Test-Time Training

By

Yu Sun

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Spring 2023

Test-Time Training

Abstract

Test-Time Training

By

Yu Sun

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alexei A. Efros, Co-chair

Professor Moritz Hardt, Co-chair

Most models in machine learning today are fixed during deployment. As a consequence, a trained model must prepare to be robust to all possible futures, even though only one of them is actually going to happen. The basic idea of test-time training is to train on this future once it arrives in the form of a test instance. Since each test instance arrives without a ground truth label, training is performed with self-supervision. This thesis explores the first steps in realizing this idea, for images, videos and robotics.

To my parents.

Contents

Acknowledgments

First, I would like to thank my two advisors: Alyosha Efros and Moritz Hardt. While most graduates open their theses this way, I presume that their advisors are all special to them in unique ways. Mine are certainly very special to me, in addition to being very special researchers themselves. In the first three years of my PhD, I did not publish anything. In fact, I have only published one paper with myself as first author and my advisors as last - the usual combination that other graduates would write a handful. I have done little for my advisors, but they have given me so much. They have been unusually generous with their advice, even after seeing how I choose to learn it the hard way. They have been giving me unselfish and unconditional support, and encouraging me to only shoot for the highest academic standards, even after seeing how often I fail to get there. Thank you, Alyosha, for being creatively provocative. If there is any hint of that quality in this thesis, it comes from you. Thank you, Moritz, for being my academic role model. It might not be surprising for a naive, first-year graduate student to feel that of her advisor. I am lucky to still feel that way towards the end.

I would like to thank Bruno Olshausen on my thesis committee, and Pieter Abbeel on my qualifying exam committee. Thanks to Koushil Sreenath, also on my qualifying exam committee, for helping me explore legged locomotion in my fourth year. Also thanks to Benjamin Burchfiel at TRI, as well as the entire manipulation team there, for hosting me as an intern in the summer of 2021.

I would like to thank Kilian Weinberger at Cornell University, for the amazing experience of doing undergraduate research in his lab for almost four years. Thanks to Matt Kusner, my first mentor in machine learning research, back when I was a clueless freshman in 2014. Also thanks to the many other students at Kilian's lab at the time who helped, inspired and collaborated with me, as well as the other faculty members who have mentored me in undergraduate research: Karthik Sridharan at Cornell, and Jeremy Buhler at Washington University in St. Louis. In hindsight, I was incredibly lucky at the beginning of my research career to have learned from all those people.

I would like to thank my long-term collaborators: Xiaolong Wang at UCSD, and Xinlei Chen at Meta AI Research. Xinlei has also been my manager at Meta, but never made me feel that way. Many of the ideas in this thesis are primitive at best, if not altogether silly. Thank you, Xiaolong and Xinlei, for supporting these silly things I bet on. Also thanks to Yossi Gandelsman, who has been teaching me a lot through our collaboration, in addition to producing almost half of this thesis.

I would like to thank my two junior collaborators: Xinhao Li and Renhao Wang, for their trust and commitment. Thanks to the collaborators in my two robotics papers: Wenlong Ma, Changhao Wang and Xiang Zhang, as well as Nicklas Hansen and Lerrel Pinto. Also thanks to Zhuang Liu and John Miller, collaborators in my first paper on test-time training.

I would like to thank my labmates in the two research groups: Chloe Hsu, Lydia Liu, Celestine Mendler-Dünner, Smitha Milli, Juanky Perdomo, Ludwig Schmidt, and Tijana Zrnic in Moritz' group, as well as Tim Brooks, Dave Epstein, Shiry Ginosar, Aleks Holynski, Allan Jabri, Toru Lin, Bill Peebles, and Assaf Shocher in Alyosha's group. Also thanks to all of my friends at the office: Anastasios Angelopoulos, Yutong Bai, Amir Bar, Shubham Goel, Hang Gao, Ashish Kumar, Boyi Li, Ruilong Li, Antonio Loquercio, Vongani Maluleke, Kartik Mangalam, Evonne Ng, George Pavlakos, Haozhi Qi, Ilija Radosavovic, Jathushan Rajasegaran, Sasha Sax, Matt Tancik, Ethan Weber, Tete Xiao, and Vickie Ye. Thank you all for making my professional life at Berkeley decorated with personal memories.

I would like to thank those friends that do not share formal, professional ties but have been especially helpful to me academically: Armin Askari, Yeshwanth Cherapanamjeri, Zihao Chen and Wenlong Mou. I would also like to thank Zitong Wang and Yaodong Yu, as well as two college friends who have been involved in every stage of my PhD – Fangzhou Xiao and Jiefu Zhang. In addition, thanks to Haggai Niv, my piano teacher, and the closest I have to a fatherly figure in the Bay Area. Also thanks to Heath Guo, who has generously opened up his Sunnyvale home to me in the past year.

Finally, I would like to thank my parents, to whom this thesis is dedicated. Words cannot express my gratitude for their support during graduate school, let alone during the younger and more vulnerable years of my life. Thank you, mom and dad!

Chapter 2 is adapted from *Test-Time Training with Self-Supervision for Generalization under Distribution Shifts* at ICML 2020, joint work with Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, Moritz Hardt.

Chapter 3 is adapted from *Test-Time Training with Masked Autoencoders* at NeurIPS 2022, joint work with Yossi Gandelsman, Xinlei Chen, Alexei A. Efros.

Chapter 4 is adapted from *Test-Time Training on Video Streams* in submission at the time of writing, joint work with Renhao Wang, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros.

Chapter 5 is adapted from *Online Learning of Unknown Dynamics for Model-Based Controllers in Legged Locomotion* at RA-L and IROS 2021, joint work with Wyatt L. Ubellacker, Wen-Loong Ma, Xiang Zhang, Changhao Wang, Noel V. Csomay-Shanklin, Masayoshi Tomizuka, Koushil Sreenath, Aaron D. Ames.

Chapter 1

Introduction

Generalization is the central theme of machine learning. Research in artificial intelligence is very much motivated by our intuitive idea of human-like generalization, as well as our attempts to make it rigorous and reproducible in machines. The traditional setting of generalization uses the notion of a statistical distribution: samples in the training and test set are drawn from the same distribution, and generalization for a parametric model is measured as its performance on the test set, after being optimized on the training set. However, this setting fails to capture the "human-like" intuition - our world rarely gives us "test samples" drawn independently from the same distribution as our past history.

This thesis explores a paradigm of algorithms – test-time training (TTT) – that shows promising results for generalization in two alternative, perhaps more realistic settings. The first, called *generalization under distribution shifts*, measures performance on test sets drawn from unknown distributions that are different from the training set. Results in this setting are presented in Chapter 2 and 3, and focus on computer vision applications. The second setting has no established name – we call it *generalization on streams*. Results on video streams are presented in Chapter 4. Results in robotics without vision are presented in Chapter 5.

The basic algorithm of TTT is simple: for each test input, first train the model on this input before making a prediction. Training can be performed in any way that does not require the ground truth label, e.g. through self-supervision. In the streaming setting, training can be extended beyond the current input, to past inputs. Of course, many technical details – the specific form of self-supervision, how optimization is performed, and whether or not to retain the model trained for each test input – are discussed in the subsequent chapters.

However, at the time of writing, our understanding of even the basic algorithm is only intuitive at best. There are two seemingly competing, yet deeply connected intuitions for how TTT might help: 1) Under distribution shifts, the test input serves as a high variance approximation to the entire test distribution, which is otherwise not available for training. 2) Allowing a different model for each test input prevents each model from wasting capacity on the rest of the input space, thus making the effective capacity of the entire system larger.

The two intuitions are seemingly competing for many reasons. The first intuition says that TTT only helps with distribution shifts, while the second says that it helps even without.

The first intuition says that the more inputs from the same test distribution we are able to obtain at a time, the more TTT helps; while the second says that the best case scenario is when TTT only uses one test input at a time and does not waste capacity on the others. These arguments about how the two intuitions disagree are concrete: they correspond to experiments in different setting, whose results can only support one intuition or the other.

Most of the results in this thesis, especially in Chapter 2 and 3, support the first intuition, that TTT helps by "knowing the unknown distribution". The author, however, believes in the second, that "locality increase capacity". Priding himself as someone who respects scientific evidence, why does the author still presume the contradicting intuition? A small reason is that results in Chapter 4 and 5, especially on the "benefits of forgetting", seem to hint at the validity of the "locality" intuition. Another reason is that the basic algorithm is more like a paradigm, whose space of numerous algorithms is barely explored; we might as well not have landed on the right algorithm for TTT that supports the second.

The biggest reason is that both the first intuition and the evidence supporting it are based on the concept of "distribution shifts", as a caricature of the phenomenon of generalization we want to study in artificial intelligence. The author believes this incomplete, if not inaccurate. Statistical concepts such as distributions are meaningful when we talk about populations; the best biological analogy of learning at the level of populations is perhaps evolution. However, learning also happens at the level of individuals; the obvious analogy here being us humans learning during our daily experience, which often serves as the inspiration for research in machine learning. The trajectory of each one of us is not very statistical, and the kind of unknown changes encountered cannot be described as distribution shifts. If TTT is only suitable for learning at the individual level, then the first intuition is irrelevant.

With some liberties taken on the concept of "distributions", one can find the two intuitions deeply connected. For example, if samples in a test set are drawn from the same distribution as training, most people would agree that there is no distribution shift. However, a test set of, say, size 50,000, can either be regarded as one test set, or 50,000 test sets, each containing only one sample. In the latter case, each of these 50,000 singleton test sets presents a tremendous "distribution shift", representing only a local part of the entire distribution. The reader can take this example as illustrating the connection between "distribution shifts" (Intuition 1) and "locality" (Intuition 2). The author prefers it as illustrating the limitation of the concept of "distributions" when thinking about generalization.

The rest of the thesis does not indulge in philosophical debates of this kind. The narrative in each chapter simply follows what best fits its empirical results – usually some version of the first intuition. In the future, the author hopes to produce enough results such that the narrative can be rewritten according to the second, making the rest of the thesis an obsolete artifact of history.

Chapter 2

Test-Time Training on Images

Supervised learning remains notoriously weak at generalization under distribution shifts. Unless training and test data are drawn from the same distribution, even seemingly minor differences turn out to defeat state-of-the-art models [146]. Adversarial robustness and domain adaptation are but a few existing paradigms that try to *anticipate* differences between the training and test distribution with either topological structure or data from the test distribution available during training. We explore a new take on generalization that *does not* anticipate the distribution shifts, but instead learns from them at test time.

We start from a simple observation. The unlabeled test sample x presented at test time gives us a hint about the distribution from which it was drawn. We propose to take advantage of this hint on the test distribution by allowing the model parameters $\boldsymbol{\theta}$ to depend on the test sample x, but not its unknown label y. The concept of a variable decision boundary $\boldsymbol{\theta}(x)$ is powerful in theory since it breaks away from the limitation of fixed model capacity, but the design of a feedback mechanism from x to $\boldsymbol{\theta}(x)$ raises new challenges in practice that we only begin to address here.

Our proposed test-time training method creates a self-supervised learning problem based on this single test sample x, updating $\boldsymbol{\theta}$ at test time before making a prediction. Self-supervised learning uses an auxiliary task that automatically creates labels from unlabeled inputs. In our experiments, we use the task of rotating each input image by a multiple of 90 degrees and predicting its angle [58].

This approach can also be easily modified to work outside the standard supervised learning setting. If several test samples arrive in a batch, we can use the entire batch for test-time training. If samples arrive in an online stream, we obtain further improvements by keeping the state of the parameters. After all, prediction is rarely a single event. The online version can be the natural mode of deployment under the additional assumption that test samples are produced by the same or smoothly changing distribution shifts.

We experimentally validate our method in the context of object recognition on several standard benchmarks. These include images with diverse types of corruption at various levels [73], video frames of moving objects [155], and a new test set of unknown shifts collected by [146]. Our algorithm makes substantial improvements under distribution shifts, while maintaining the same performance on the original distribution.

In our experiments, we compare with a strong baseline (labeled joint training) that uses both supervised and self-supervised learning at training-time, but keeps the model fixed at test time. Recent work shows that *training-time* self-supervision improves robustness [76]; our joint training baseline corresponds to an improved implementation of this work.

2.1 Method

This section describes the algorithmic details of our method. To set up notation, consider a standard K-layer neural network with parameters θ_k for layer k. The stacked parameter vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K)$ specifies the entire model for a classification task with loss function $l_m(x, y; \boldsymbol{\theta})$ on the test sample (x, y). We call this the *main task*, as indicated by the subscript of the loss function.

We assume to have training data $(x_1, y_1), \ldots, (x_n, y_n)$ drawn i.i.d. from a distribution P. Standard empirical risk minimization solves the optimization problem:

$$\min_{\boldsymbol{\theta}} \frac{1}{n} \sum_{i=1}^{n} l_m(x_i, y_i; \boldsymbol{\theta}). \tag{2.1}$$

Our method requires a *self-supervised auxiliary task* with loss function $l_s(x)$. In this chapter, we choose the rotation prediction task [58], which has been demonstrated to be simple and effective at feature learning for convolutional neural networks. The task simply rotates x in the image plane by one of 0, 90, 180 and 270 degrees and have the model predict the angle of rotation as a four-way classification problem.

The auxiliary task shares some of the model parameters $\boldsymbol{\theta}_e = (\theta_1, \ldots, \theta_\kappa)$ up to a certain $\kappa \in \{1, \ldots, K\}$. We designate those κ layers as a *shared feature extractor*. The auxiliary task uses its own task-specific parameters $\boldsymbol{\theta}_s = (\theta'_{\kappa+1}, \ldots, \theta'_K)$. We call the unshared parameters $\boldsymbol{\theta}_s$ the *self-supervised task branch*, and $\boldsymbol{\theta}_m = (\theta_{\kappa+1}, \ldots, \theta_K)$ the *main task branch*. Pictorially, the joint architecture is a Y-structure with a shared bottom and two branches. For our experiments, the self-supervised task branch has the same architecture as the main branch, except for the output dimensionality of the last layer due to the different number of classes in the two tasks.

Training is done in the fashion of multi-task learning [26]; the model is trained on both tasks on the same data drawn from P. Losses for both tasks are added together, and gradients are taken for the collection of all parameters. The joint training problem is therefore

$$\min_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_m, \boldsymbol{\theta}_s} \frac{1}{n} \sum_{i=1}^{n} l_m(x_i, y_i; \boldsymbol{\theta}_m, \boldsymbol{\theta}_e) + l_s(x_i; \boldsymbol{\theta}_s, \boldsymbol{\theta}_e). \tag{2.2}$$

Now we describe the standard version of Test-Time Training on a single test sample x. Simply put, Test-Time Training fine-tunes the shared feature extractor $\boldsymbol{\theta}_e$ by minimizing the auxiliary task loss on x. This can be formulated as

$$\min_{\boldsymbol{\theta}_e} l_s(x; \boldsymbol{\theta}_s, \boldsymbol{\theta}_e). \tag{2.3}$$

Denote $\boldsymbol{\theta}_e^*$ the (approximate) minimizer of Equation 3.2. The model then makes a prediction using the updated parameters $\boldsymbol{\theta}(x) = (\boldsymbol{\theta}_e^*, \boldsymbol{\theta}_m)$. Empirically, the difference is negligible between minimizing Equation 3.2 over $\boldsymbol{\theta}_e$ versus over both $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_s$. Theoretically, the difference exists only when optimization is done with more than one gradient step.

Test-Time Training naturally benefits from standard data augmentation techniques. On each test sample x, we perform the exact same set of random transformations as for data augmentation during training, to form a batch only containing these augmented copies of x for Test-Time Training.

Online Test-Time Training. In the standard version of our method, the optimization problem in Equation 3.2 is always initialized with parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_e, \boldsymbol{\theta}_s)$ obtained by minimizing Equation 3.1. After making a prediction on x, $\boldsymbol{\theta}_e^*$ is discarded. Outside of the standard supervised learning setting, when the test samples arrive online sequentially, the online version solves the same optimization problem as in Equation 3.2 to update the shared feature extractor $\boldsymbol{\theta}_e$. However, on test sample x_t, $\boldsymbol{\theta}$ is instead initialized with $\boldsymbol{\theta}(x_{t-1})$ updated on the previous sample x_{t-1}. This allows $\boldsymbol{\theta}(x_t)$ to take advantage of the distributional information available in $x_1, \ldots, x_{t-1}$ as well as x_t.

2.2 Empirical Results

We experiment with both versions of our method (standard and online) on three kinds of benchmarks for distribution shifts, presented here in the order of visually low to high-level. Our code is available at the project website.

Network details. Our architecture and hyper-parameters are consistent across all experiments. We use ResNets [71], which are constructed differently for CIFAR-10 [98] (26-layer) and ImageNet [148] (18-layer). The CIFAR-10 dataset contains 50K images for training, and 10K images for testing. The ImageNet contains 1.2M images for training and the 50K validation images are used as the test set. ResNets on CIFAR-10 have three groups, each containing convolutional layers with the same number of channels and size of feature maps; our splitting point is the end of the second group. ResNets on ImageNet have four groups; our splitting point is the end of the third group.

We use Group Normalization (GN) instead of Batch Normalization (BN) in our architecture, since BN has been shown to be ineffective when training with small batches, for which the estimated batch statistics are not accurate [87]. This technicality hurts Test-Time Training since each batch only contains (augmented) copies of a single image. Different from BN, GN is not dependent on batch size and achieves similar results on our baselines.

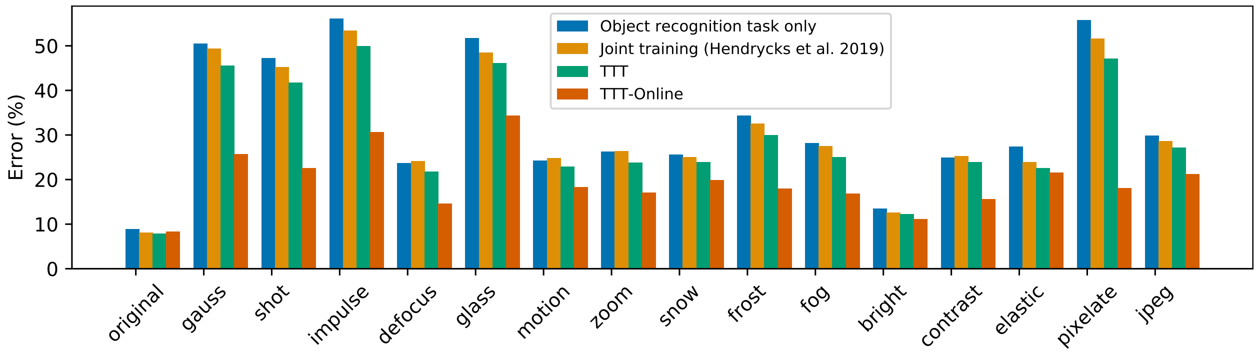


Figure 2.1: **Test error (%) on CIFAR-10-C with level 5 corruptions.** We compare our approaches, Test-Time Training (TTT) and its online version (TTT-Online), with two baselines: object recognition without self-supervision, and joint training with self-supervision but keeping the model fixed at test time. TTT improves over the baselines and TTT-Online improves even further.

Optimization details. For joint training (Equation 3.1), we use stochastic gradient descent with standard hyper-parameters as [86, 70]. For Test-Time Training (Equation 3.2), we use stochastic gradient descent with the learning rate set to that of the last epoch during training, which is 0.001 in all our experiments. We set weight decay and momentum to zero during Test-Time Training, inspired by practice in [69, 114]. For the standard version of Test-Time Training, we take ten gradient steps, using batches independently generated by the same image. For online version of Test-Time Training, we take only one gradient step given each new image. We use random crop and random horizontal flip for data augmentation. In all the tables and figures, *object recognition task only* refers to the plain ResNet model (using GN, unless otherwise specified); *joint training* refers to the model jointly trained on both the main task and the self-supervised task, fixed at test time; this has been proposed as the method in [76]; *Test-Time Training (TTT)* refers to the standard version described section 4.2; and *online Test-Time Training (TTT-Online)* refers to the online version that does not discard $\boldsymbol{\theta}(x_t)$ for x_t arriving sequentially from the same distribution. Performance for TTT-Online is calculated as the average over the entire test set; we always shuffle the test set before TTT-Online to avoid ordering artifacts.

Object Recognition on Corrupted Images

[73] propose to benchmark robustness of object recognition with 15 types of corruptions from four broad categories: noise, blur, weather and digital. Each corruption type comes in five levels of severity, with level 5 the most severe (details and sample images in the appendix). The corruptions are simulated to mimic real-world corruptions as much as possible on copies of the test set for both CIFAR-10 and ImageNet. The new test sets are named as CIFAR-10-C and ImageNet-C, respectively. In the proposed benchmark, training should be done on the

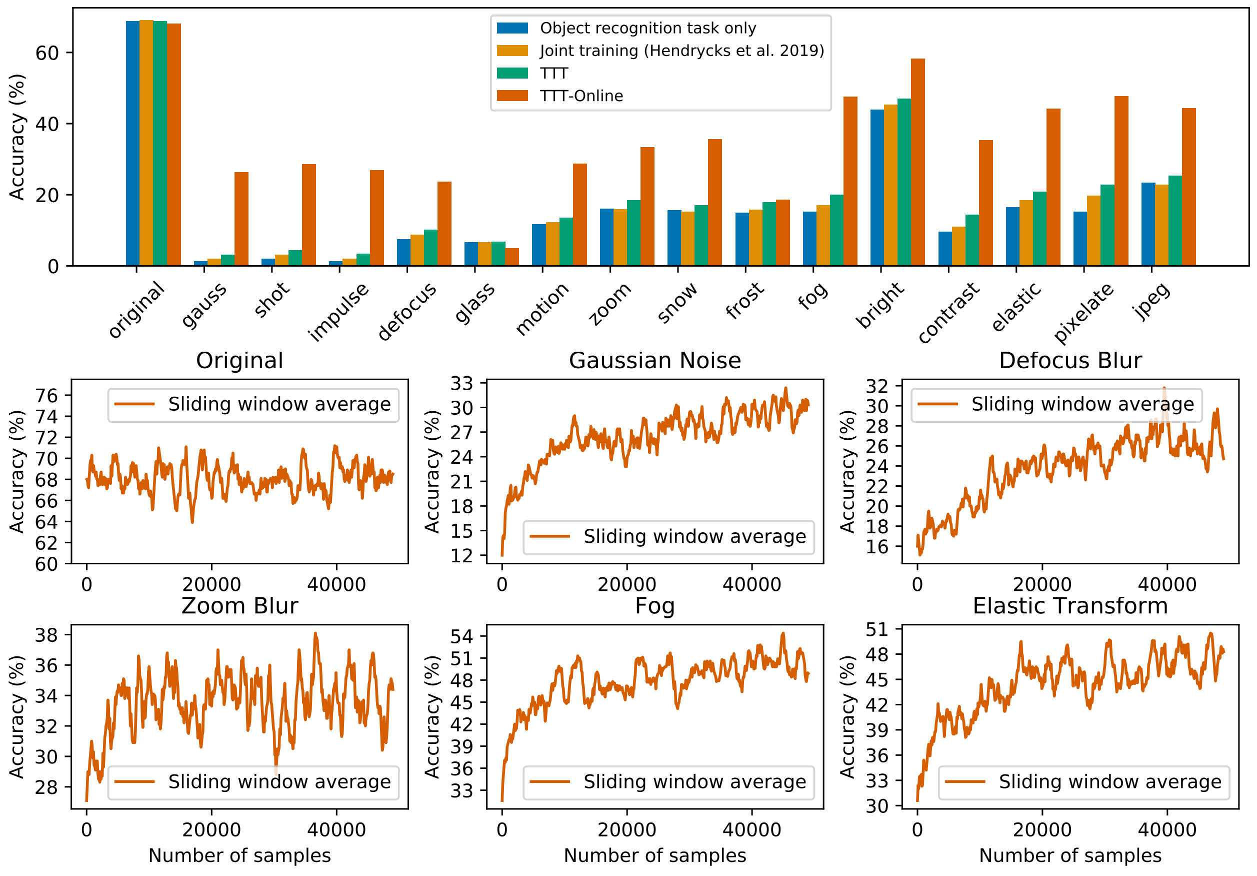


Figure 2.2: **Test accuracy (%) on ImageNet-C with level 5 corruptions.** Upper panel: Our approaches, TTT and TTT-Online, show significant improvements in all corruption types over the two baselines. Lower panel: We show the accuracy of TTT-Online as the average over a sliding window of 100 samples; TTT-Online generalizes better as more samples are evaluated (x-axis), without hurting on the original distribution. We use accuracy instead of error here because the baseline performance is very low for most corruptions.

original training set, and the diversity of corruption types should make it difficult for any methods to work well across the board if it relies too much on corruption specific knowledge. For online Test-Time Training, we take the entire test set as a stream of incoming images, and update and test on each image in an online manner as it arrives.

CIFAR-10-C. Our results on the level 5 corruptions (most severe) are shown in Figure 2.1. Across all five levels and 15 corruption types, both standard and online versions of Test-Time Training improve over the object recognition task only baseline by a large margin. The standard version always improves over joint training, and the online version often improves significantly (>10%) over joint training and never hurts by more than 0.2%. Specifically, TTT-Online contributes >24% on the three noise types and 38% on pixelation. For a learning problem with the seemingly unstable setup that abuses a single image, this kind of consistency is rather surprising.

The baseline ResNet-26 with object recognition task only has error 8.9% on the original test set of CIFAR-10. The joint training baseline actually improves performance on the original to 8.1%. More surprisingly, unlike many other methods that trade off original performance for robustness, Test-Time Training further improves on the original test set by 0.2% consistently over multiple independent trials. This suggests that our method does not choose between specificity and generality.

Separate from our method, it is interesting to note that joint training consistently improves over the single-task baseline, as discovered by [76]. [73] have also experimented with various other training methods on this benchmark, and point to Adversarial Logit Pairing (ALP) [94] as the most effective approach. Results of this additional baseline on all levels of CIFAR-10-C are shown in the appendix, along with its implementation details. While surprisingly robust under some of the most severe corruptions (especially the three noise types), ALP incurs a much larger error (by a factor of two) on the original distribution and some corruptions (e.g. all levels of contrast and fog), and hurts performance significantly when the corruptions are not as severe (especially on levels 1-3); this kind of tradeoff is to be expected for methods based on adversarial training.

	orig	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
TTT-Online	**8.2**	**25.8**	**22.6**	30.6	**14.6**	**34.4**	**18.3**	**17.1**	**20.0**	**18.0**	**16.9**	**11.2**	15.6	**21.6**	**18.1**	**21.2**
UDA-SS	9.0	28.2	26.5	**20.8**	15.6	43.7	24.5	23.8	25.0	24.9	17.2	12.7	**11.6**	22.1	20.3	22.6

Table 2.1: **Test error (%) on CIFAR-10-C with level 5 corruption.** Comparison between online Test-Time Training (TTT-Online) and unsupervised domain adaptation by self-supervision (UDA-SS) [172] with access to the entire (unlabeled) test set during training. We highlight the lower error in bold. We have abbreviated the names of the corruptions, in order: original test set, Gaussian noise, shot noise, impulse noise, defocus blur, glass blue, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic transformation, pixelation, and JPEG compression. The reported numbers for TTT-Online are the same as in Figure 2.1.

ImageNet-C. Our results on the level 5 corruptions (most severe) are shown in Figure 2.2. We use accuracy instead of error for this dataset because the baseline performance is very low for most corruptions. The general trend is roughly the same as on CIFAR-10-C. The standard version of TTT always improves over the baseline and joint training, while the online version only hurts on the original by 0.1% over the baseline, but significantly improves (by a factor of more than three) on many of the corruption types.

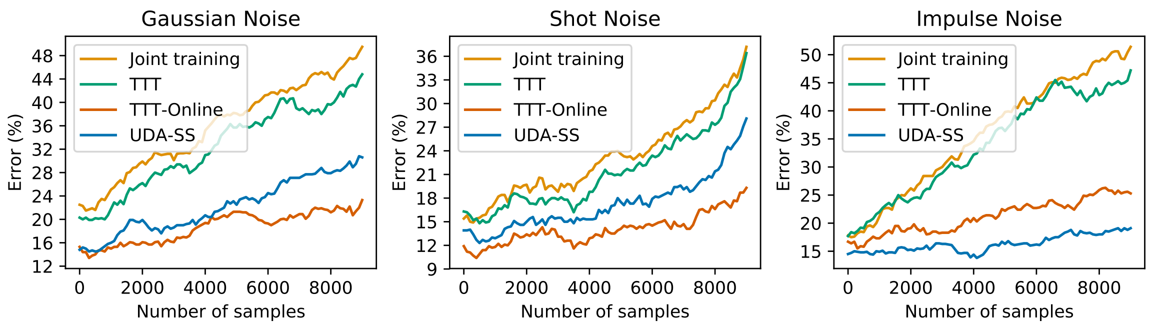


Figure 2.3: **Test error (%) on CIFAR-10-C, for the three noise types, with gradually changing distribution.** The distribution shifts are created by increasing the standard deviation of each noise type from small to large, the further we go on the x-axis. As the samples get noisier, all methods suffer greater errors the more we evaluate into the test set, but online Test-Time Training (TTT-Online) achieves gentler slopes than joint training. For the first two noise types, TTT-Online also achieves better results over unsupervised domain adaptation by self-supervision (UDA-SS) [172].

In the lower panel of Figure 2.2, we visualize how the accuracy (averaged over a sliding window) of the online version changes as more images are tested. Due to space constraints, we show this plot on the original test set, as well as every third corruption type, following the same order as in the original paper. On the original test set, there is no visible trend in performance change after updating on the 50,000 samples. With corruptions, accuracy has already risen significantly after 10,000 samples, but is still rising towards the end of the 50,000 samples, indicating room for additional improvements if more samples were available. Without seeing a single label, TTT-Online behaves as if we were training on the test set from the appearance of the plots.

Comparison with unsupervised domain adaptation. Table 2.1 empirically compares online Test-Time Training (TTT-Online) with unsupervised domain adaptation through self-supervision (UDA-SS) [172], which is similar to our method in spirit but is designed for the setting of unsupervised domain adaptation. Given labeled data from the training distribution and unlabeled data from the test distribution, UDA-SS hopes to find an invariant representation that extracts useful features for both distributions by learning to perform a self-supervised task, specifically rotation prediction, simultaneously on data from both. It then learns a labeling function on top of the invariant representation using the labeled data. In our experiments, the unlabeled data given to UDA-SS is the *entire test set itself* without the labels.

Because TTT-Online can only learn from the unlabeled test samples that have already been evaluated on, it is given less information than UDA-SS at all times. In this sense, UDA-SS should be regarded as an oracle rather than a baseline. Surprisingly, TTT-Online

Accuracy (%)	Airplane	Bird	Car	Dog	Cat	Horse	Ship	**Average**
Object recognition task only	67.9	35.8	42.6	14.7	52.0	42.0	66.7	**41.4**
Joint training [76]	70.2	36.7	42.6	15.5	52.0	44.0	66.7	**42.4**
TTT (standard version)	70.2	39.2	42.6	21.6	54.7	46.0	77.8	**45.2**
TTT-Online	70.2	39.2	42.6	22.4	54.7	46.0	77.8	**45.4**

Table 2.2: Class-wise and average classification accuracy (%) on CIFAR classes in VID-Robust, adapted from [155]. Test-Time Training (TTT) and online Test-Time Training (TTT-Online) improve over the two baselines on average, and by a large margin on "ship" and "dog" classes where the rotation task is more meaningful than in classes like "airplane".

outperforms UDA-SS on 13 out of the 15 corruptions as well as the original distribution. Our explanation is that UDA-SS has to find an invariant representation for both distributions, while TTT-Online only adapts the representation to be good for the current test distribution. That is, TTT-Online has the flexibility to forget the training distribution representation, which is no longer relevant. This suggests that in our setting, forgetting is not harmful and perhaps should even be taken advantage of.

Gradually changing distribution shifts. In our previous experiments, we have been evaluating the online version under the assumption that the test inputs x_t for $t = 1...n$ are all sampled from the same test distribution Q, which can be different from the training distribution P. This assumption is indeed satisfied for i.i.d. samples from a shuffled test set. But here we show that this assumption can in fact be relaxed to allow $x_t \sim Q_t$, where Q_t is close to Q_{t+1} (in the sense of distributional distance). We call this the assumption of gradually changing distribution shifts. We perform experiments by simulating such distribution shifts on the three noise types of CIFAR-10-C. For each noise type, x_t is corrupted with standard deviation σ_t, and $\sigma_1, ..., \sigma_n$ interpolate between the standard deviation of level 1 and level 5. So x_t is more severely corrupted as we evaluate further into the test set and t grows larger. As shown in Figure 2.3, TTT-Online still improves upon joint training (and our standard version) with this relaxed assumption, and even upon UDA-SS for the first two noise types.

Object Recognition on Video Frames

The Robust ImageNet Video Classification (VID-Robust) dataset was developed by [155] from the ImageNet Video detection dataset [148], to demonstrate how deep models for object recognition trained on ImageNet (still images) fail to adapt well to video frames. The VID-Robust dataset contains 1109 sets of video frames in 30 classes; each set is a short video clip of frames that are similar to an anchor frame. Our results are reported on the

Method	Accuracy (%)
Object recognition task only	62.7
Joint training [76]	63.5
TTT (standard version)	63.8
TTT-Online	64.3

Table 2.3: Test accuracy (%) on VID-Robust dataset [155]. TTT and TTT-Online improve over the baselines.

anchor frames. To map the 1000 ImageNet classes to the 30 VID-Robust classes, we use the max-conversion function in [155]. Without any modifications for videos, we apply our method to VID-Robust on top of the same ImageNet model as in the previous subsection. Our classification accuracy is reported in Table 2.3.

In addition, we take the seven classes in VID-Robust that overlap with CIFAR-10, and re-scale those video frames to the size of CIFAR-10 images, as a new test set for the model trained on CIFAR-10 in the previous subsection. Again, we apply our method to this dataset without any modifications. Our results are shown in Table 2.2, with a breakdown for each class. Noticing that Test-Time Training does not improve on the airplane class, we inspect some airplane samples, and observe black margins on two sides of most images, which provide a trivial hint for rotation prediction. In addition, given an image of airplanes in the sky, it is often impossible even for humans to tell if it is rotated. This shows that our method requires the self-supervised task to be both well defined and non-trivial.

CIFAR-10.1: Unknown Distribution Shifts

CIFAR-10.1 [146] is a new test set of size 2000 modeled after CIFAR-10, with the exact same classes and image dimensionality, following the dataset creation process documented by the original CIFAR-10 paper as closely as possible. The purpose is to investigate the distribution shifts present between the two test sets, and the effect on object recognition. All models tested by the authors suffer a large performance drop on CIFAR-10.1 comparing to CIFAR-10, even though there is no human noticeable difference, and both have the same human accuracy. This demonstrates how insidious and ubiquitous distribution shifts are, even when researchers strive to minimize them.

The distribution shifts from CIFAR-10 to CIFAR-10.1 pose an extremely difficult problem, and no prior work has been able to improve the performance of an existing model on this new test set, probably because: 1) researchers cannot even identify the distribution shifts, let alone describe them mathematically; 2) the samples in CIFAR-10.1 are only revealed at test time; and even if they were revealed during training, the distribution shifts are too subtle,

Method	Error (%)
Object recognition task only	17.4
Joint training [76]	16.7
TTT (standard version)	15.9

Table 2.4: Test error (%) on CIFAR-10.1 [146]. TTT is the first method to improve the performance of an existing model on this new test set.

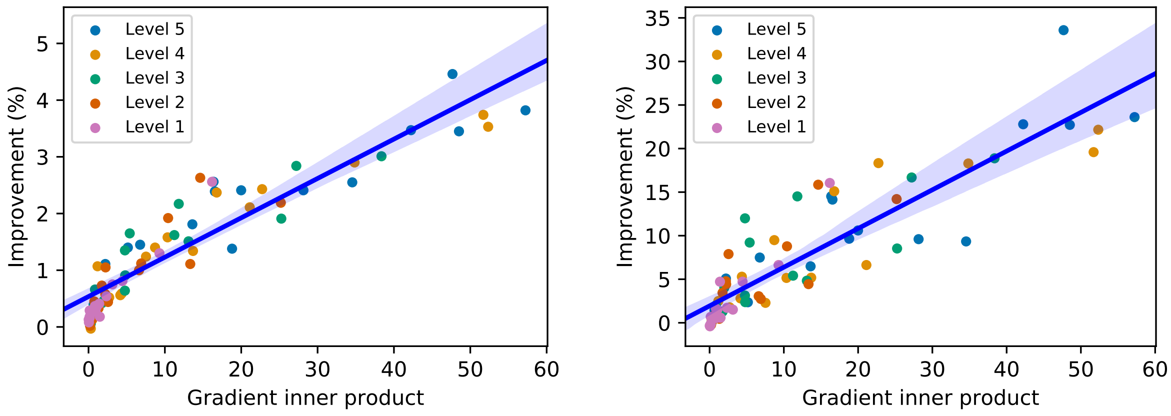


Figure 2.4: Scatter plot of the inner product between the gradients (on the shared feature extractor $\boldsymbol{\theta}_e$) of the main task l_m and the self-supervised task l_e, and the improvement in test error (%) from Test-Time Training, for the standard (left) and online (right) version. Each point is the average over a test set, and each scatter plot has 75 test sets, from all 15 types of corruptions over five levels as described in section 2.2. The blue lines and bands are the best linear fits and the 99% confidence intervals. The linear correlation coefficients are 0.93 and 0.89 respectively, indicating strong positive correlation between the two quantities, as suggested by Theorem 1.

and the sample size is too small, for domain adaptation [146].

On the original CIFAR-10 test set, the baseline with only object recognition has error 8.9%, and with joint training has 8.1%; comparing to the first two rows of Table 2.4, both suffer the typical performance drop (by a factor of two). TTT yields an improvement of 0.8% (relative improvement of 4.8%) over joint training. We recognize that this improvement is small relative to the performance drop, but see it as an encouraging first step for this very difficult problem.

2.3 Theoretical Results

This section contains our preliminary study of when and why Test-Time Training is expected to work. For convex models, we prove that positive gradient correlation between the loss functions leads to better performance on the main task after Test-Time Training. Equipped with this insight, we then empirically demonstrate that gradient correlation governs the success of Test-Time Training on the deep learning model discussed in Section 2.2.

Before stating our main theoretical result, we first illustrate the general intuition with a toy model. Consider a regression problem where $x \in \mathbb{R}^d$ denotes the input, $y_1 \in \mathbb{R}$ denotes the label, and the objective is the square loss $(\hat{y} - y_1)^2/2$ for a prediction $\hat{y}$. Consider a two layer linear network parametrized by $\boldsymbol{A} \in \mathbb{R}^{h \times d}$ and $\boldsymbol{v} \in \mathbb{R}^h$ (where h stands for the hidden dimension). The prediction according to this model is $\hat{y} = \boldsymbol{v}^\top \boldsymbol{A} x$, and the main task loss is

$$l_m(x, y_1; \boldsymbol{A}, \boldsymbol{v}) = \frac{1}{2}\left(y_1 - \boldsymbol{v}^\top \boldsymbol{A} x\right)^2 . \tag{2.4}$$

In addition, consider a self-supervised regression task that also uses the square loss and automatically generates a label y_s for x. Let the self-supervised head be parametrized by $\boldsymbol{w} \in \mathbb{R}^h$. Then the self-supervised task loss is

$$l_s(x, y_2; \boldsymbol{A}, \boldsymbol{w}) = \frac{1}{2}\left(y_2 - \boldsymbol{w}^\top \boldsymbol{A} x\right)^2 . \tag{2.5}$$

Now we apply Test-Time Training to update the shared feature extractor $\boldsymbol{A}$ by one step of gradient descent on l_s, which we can compute with y_2 known. This gives us

$$\boldsymbol{A}' \leftarrow \boldsymbol{A} - \eta\left(y_2 - \boldsymbol{w}^\top \boldsymbol{A} x\right)\left(-\boldsymbol{w} x^\top\right), \tag{2.6}$$

where $\boldsymbol{A}'$ is the updated matrix and η is the learning rate. If we set $\eta = \eta^*$ where

$$\eta^* = \frac{y_1 - \boldsymbol{v}^\top \boldsymbol{A} x}{\left(y_2 - \boldsymbol{w}^\top \boldsymbol{A} x\right) \boldsymbol{v}^\top \boldsymbol{w} x^\top x}, \tag{2.7}$$

then with some simple algebra, it is easy to see that the main task loss $l_m(x, y_1; \boldsymbol{A}', \boldsymbol{v}) = 0$. Concretely, Test-Time Training drives the main task loss down to zero with a single gradient step for a carefully chosen learning rate. In practice, this learning rate is unknown since it depends on the unknown y_1. However, since our model is convex, as long as η^* is positive, it suffices to set η to be a small positive constant (see details in the appendix). If $x \neq 0$, one sufficient condition for η^* to be positive (when neither loss is zero) is to have

$$\operatorname{sign}\left(y_1 - \boldsymbol{v}^\top \boldsymbol{A} x\right) = \operatorname{sign}\left(y_2 - \boldsymbol{w}^\top \boldsymbol{A} x\right) \tag{2.8}$$

$$\text{and} \quad \boldsymbol{v}^\top \boldsymbol{w} > 0 . \tag{2.9}$$

For our toy model, both parts of the condition above have an intuition interpretation. The first part says that the mistakes should be correlated, in the sense that predictions from both tasks are mistaken in the same direction. The second part, $v^\top \boldsymbol{w} > 0$, says that the decision boundaries on the feature space should be correlated. In fact, these two parts hold iff. $\langle \nabla l_m(\boldsymbol{A}), \nabla l_s(\boldsymbol{A}) \rangle > 0$ (see a simple proof of this fact in the appendix). To summarize, if the gradients have positive correlation, Test-Time Training is guaranteed to reduce the main task loss. Our main theoretical result extends this to general smooth and convex loss functions.

Theorem 1. *Let $l_m(x, y; \boldsymbol{\theta})$ denote the main task loss on test instance x, y with parameters $\boldsymbol{\theta}$, and $l_s(x; \boldsymbol{\theta})$ the self-supervised task loss that only depends on x. Assume that for all x, y, $l_m(x, y; \boldsymbol{\theta})$ is differentiable, convex and β-smooth in $\boldsymbol{\theta}$, and both $\|\nabla l_m(x, y; \boldsymbol{\theta})\|, \|\nabla l_s(x, \boldsymbol{\theta})\| \leq G$ for all $\boldsymbol{\theta}$. With a fixed learning rate $\eta = \frac{\epsilon}{\beta G^2}$, for every x, y such that*

$$\langle \nabla l_m(x, y; \boldsymbol{\theta}), \nabla l_s(x; \boldsymbol{\theta}) \rangle > \epsilon, \tag{2.10}$$

we have

$$l_m(x, y; \boldsymbol{\theta}) > l_m(x, y; \boldsymbol{\theta}(x)), \tag{2.11}$$

where $\boldsymbol{\theta}(x) = \boldsymbol{\theta} - \eta \nabla l_s(x; \boldsymbol{\theta})$ i.e. Test-Time Training with one step of gradient descent.

The proof uses standard techniques in optimization, and is left for the appendix. Theorem 1 reveals gradient correlation as a determining factor of the success of Test-Time Training in the smooth and convex case. In Figure 2.4, we empirically show that our insight also holds for non-convex loss functions, on the deep learning model and across the diverse set of corruptions considered in Section 2.2; stronger gradient correlation clearly indicates more performance improvement over the baseline.

2.4 Related Work

Learning on test instances. [158] provide a key inspiration for our work by showing that image super-resolution could be learned at test time simply by trying to upsample a downsampled version of the input image. More recently, [19] improve photo manipulation by adapting a pre-trained GAN to the statistics of the input image. One of the earlier examples of this idea comes from [88], who improve Viola-Jones face detection [188] by bootstrapping the more difficult faces in an image from the more easily detected faces in that same image. The online version of our algorithm is inspired by the work of [126], which makes video segmentation more efficient by using a student model that learns online from a teacher model. The idea of online updates has also been used in [92] for tracking and detection. A recent work in echocardiography [206] improves the deep learning model that tracks myocardial motion and cardiac blood flow with sequential updates. Lastly, we share the philosophy of transductive learning [183, 51], but have little in common with their classical algorithms; recent work by [178] theoretically explores this for linear prediction, in the context of debiasing the LASSO estimator.

Self-supervised learning studies how to create labels from the data, by designing various pretext tasks that can learn semantic information without human annotations, such as context prediction [44], solving jigsaw puzzles [132], colorization [101, 201], noise prediction [21], feature clustering [25]. Our paper uses rotation prediction [58]. [8] show that self-supervised learning on only a single image, surprisingly, can produce low-level features that generalize well. Closely related to our work, [76] propose that jointly training a main task and a self-supervised task (our joint training baseline in Section 2.2) can improve robustness on the main task. The same idea is used in few-shot learning [167], domain generalization [24], and unsupervised domain adaptation [172].

Adversarial robustness studies the robust risk $R_{P,\Delta}(\boldsymbol{\theta}) = \mathbb{E}_{x,y\sim P} \max_{\delta\in\Delta} l(x+\delta, y;\ \boldsymbol{\theta})$, where l is some loss function, and Δ is the set of perturbations; Δ is often chosen as the L_p ball, for $p \in \{1, 2, \infty\}$. Many popular algorithms formulate and solve this as a robust optimization problem [61, 120, 160, 144, 193, 36], and the most well known technique is adversarial training. Another line of work is based on randomized smoothing [32, 149], while some other approaches, such as input transformations [63, 165], are shown to be less effective [12]. There are two main problems with the approaches above. First, all of them can be seen as *smoothing* the decision boundary. This establishes a theoretical tradeoff between accuracy and robustness [179, 200], which we also observe empirically with our adversarial training baseline in Section 2.2. Intuitively, the more diverse Δ is, the less effective this *one-boundary-fits-all* approach can be for a particular element of Δ. Second, adversarial methods rely heavily on the mathematical structure of Δ, which might not accurately model perturbations in the real world. Therefore, generalization remains hard outside of the Δ we know in advance or can mathematically model, especially for non-adversarial distribution shifts. Empirically, [93] shows that robustness for one Δ might not transfer to another, and training on the L_∞ ball actually hurts robustness on the L_1 ball.

Non-adversarial robustness studies the effect of corruptions, perturbations, out-of-distribution examples, and real-world distribution shifts [77, 76, 80, 75]. [55] show that training on images corrupted by Gaussian noise makes deep learning models robust to this particular noise type, but does not improve performance on images corrupted by another noise type e.g. salt-and-pepper noise.

Unsupervised domain adaptation (a.k.a. transfer learning) studies the problem of distribution shifts, when an unlabeled dataset from the test distribution (target domain) is available at training time, in addition to a labeled dataset from the training distribution (source domain) [28, 60, 115, 54, 116, 180, 83, 37, 29]. The limitation of the problem setting, however, is that generalization might only be improved for this specific test distribution, which can be difficult to anticipate in advance. Prior work try to anticipate broader distributions by using multiple and evolving domains [82, 84, 81]. Test-Time Training does not anticipate any test distribution, by changing the setting of unsupervised domain adaptation, while taking inspiration from its algorithms. Our paper is a follow-up to [172], which we explain and empirically compare with in Section 2.2. Our update rule can be viewed as performing *one-sample unsupervised domain adaptation* on the fly, with the caveat that standard domain adaptation techniques might become ill-defined when there is only one sample from the target domain.

Domain generalization studies the setting where a meta distribution generates multiple environment distributions, some of which are available during training (source), while others are used for testing (target) [106, 154, 125, 15, 56, 124, 104, 52]. With only a few environments, information on the meta distribution is often too scarce to be helpful, and with many environments, we are back to the i.i.d. setting where each environment can be seen as a sample, and a strong baseline is to simply train on all the environments [105]. The setting of domain generalization is limited by the inherent tradeoff between specificity and generality of a fixed decision boundary, and the fact that generalization is again elusive outside of the meta distribution i.e. the actual P learned by the algorithm.

One (few)-shot learning studies how to learn a new task or a new classification category using only one (or a few) sample(s), on top of a general representation that has been learned on diverse samples [163, 187, 46, 145, 108, 48, 57]. Our update rule can be viewed as performing *one-shot self-supervised learning* and can potentially be improved by progress in one-shot learning.

Continual learning (a.k.a. learning without forgetting) studies the setting where a model is made to learn a sequence of tasks, and not forget about the earlier ones while training for the later [109, 117, 97, 150]. In contrast, with Test-Time Training, we are not concerned about forgetting the past test samples since they have already been evaluated on; and if a past sample comes up by any chance, it would go through Test-Time Training again. In addition, the impact of forgetting the training set is minimal, because both tasks have already been jointly trained.

Online learning (a.k.a. online optimization) is a well-studied area of learning theory [153, 67]. The basic setting repeats the following: receive x_t, predict $\hat{y}_t$, receive y_t from a worst-case oracle, and learn. Final performance is evaluated using the regret, which colloquially translates to how much worse the online learning algorithm performs in comparison to the best fixed model in hindsight. In contrast, our setting never reveals any y_t during testing even for the online version, so we do not need to invoke the concept of the worst-case oracle or the regret. Also, due to the lack of feedback from the environment after predicting, our algorithm is motivated to learn (with self-supervision) before predicting $\hat{y}_t$ instead of after. Note that some of the previously covered papers [81, 88, 126] use the term "online learning" outside of the learning theory setting, so the term can be overloaded.

Chapter 3

Test-Time Training with Masked Autoencoders

Generalization is the central theme of supervised learning, and a hallmark of intelligence. While most of the research focuses on generalization when the training and test data are drawn from the same distribution, this is rarely the case for real world deployment [176], and is certainly not true for environments where natural intelligence has emerged.

Most models today are fixed during deployment, even when the test distribution changes. As a consequence, a trained model needs to be robust to all possible distribution shifts that could happen in the future [74, 55, 185, 121]. This turns out to be quite difficult because being ready for all possible futures limits the model's capacity to be good at any particular one. But only one of these futures is actually going to happen.

This motivates an alternative perspective on generalization: instead of being ready for everything, one simply adapts to the future once it arrives. Test-time training (TTT) is one line of work that takes this perspective [171, 119, 66, 16]. The key insight is that each test input gives a hint about the test distribution. We modify the model at test time to take advantage of this hint by setting up a *one-sample learning problem*.

The only issue is that the test input comes without a ground truth label. But we can generate labels from the input itself thanks to self-supervised learning. At training time, TTT optimizes both the main task (e.g. object recognition) and the self-supervised task. Then at test time, it adapts the model with the self-supervised task alone for each test input, before making a prediction on the main task.

The choice of the self-supervised task is critical: it must be general enough to produce useful features for the main task on a wide range of potential test distributions. The self-supervised task cannot be too easy or too hard, otherwise the test input will not provide useful signal. What is a general task at the right level of difficulty? We turn to a fundamental property shared by natural images – spatial smoothness, i.e. the local redundancy of information in the xy space. Spatial autoencoding – removing parts of the data, then predicting the removed content – forms the basis of some of the most successful self-supervised tasks [186, 136, 72, 17, 196].

In this chapter, we show that masked autoencoders (MAE) [72] is well-suited for test-time training. Our simple method leads to substantial improvements on four datasets for object recognition. We also provide a theoretical analysis of our method with linear models.

3.1 Related Work

This chapter addresses the problem of generalization under distribution shifts. We first cover work in this problem setting, as well as the related setting of unsupervised domain adaptation. We then discuss test-time training and spatial autocending, the two components of our algorithm.

Problem Settings

Generalization under distribution shifts. When training and test distributions are different, generalization is intrinsically hard without access to training data from the test distribution. The robustness obtained by training or fine-tuning on one distribution shift (e.g. Gaussian noise) often does not transfer to another (e.g. salt-and-pepper noise), even for visually similar ones [55, 185]. Currently, the common practice is to avoid distribution shifts altogether by using a wider training distribution that hopefully contains the test distribution – with more training data or data augmentation [74, 204, 43].

Unsupervised domain adaptation. An easier but more restrictive setting is to use some unlabeled training data from the test distribution (target), in addition to those from the training distribution (source) [115, 116, 37, 28, 172, 83, 29, 60]. It is easier because, intuitively, most of the distribution shift happens on the inputs instead of the labels, so most of the test distribution is already known at training time through data. It is more restrictive because the trained model is only effective on the particular test distribution for which data is prepared in advance.

Test-Time Training

The idea of training on unlabeled test data has a long history. Its earliest realization is transductive learning, beginning in the 1990s [50]. Vladimir Vapnik [184] states the principle of transduction as: "Try to get the answer that you really need but not a more general one." This principle has been most commonly applied on SVMs [183, 33, 91] by using the test data to specify additional constraints on the margin of the decision boundary. Another early line of work is local learning [23, 199]: for each test input, a "local" model is trained on the nearest neighbors before a prediction is made.

In the computer vision community, [88] improves face detection by using the easier faces in a test image to bootstrap the more difficult faces in the same image. [131] creates a personalized generative model, by fine-tuning it on a few images of an individual person's face. [158] trains neural networks for super-resolution on each test image from scratch. [159] improves image retrieval by using a one-against-all classifier on the query image. [126] makes video segmentation more efficient by using a student model that learns online from a teacher model on the test videos. These are but a few examples where vision researchers find it natural to continue training during deployment.

Test-time training (TTT) [171] proposes this idea as a solution to the generalization problem in supervised learning under distribution shifts. It produces a different model for every single test input through self-supervision. This method has also been applied to several fields with domain-specific self-supervised tasks: vision-based reinforcement learning [66], legged locomotion [168], tracking objects in videos [49], natural language question answering [16], and even medical imaging [95].

The self-supervised pretext task employed by [171] is rotation prediction [58]: rotate each input in the image plane by a multiple of 90 degrees, and predict the angle as a four-way classification problem. This task is limited in generality, because it can often be too easy or too hard. For natural outdoor scenes, rotation prediction can be too easy by detecting the horizon's orientation alone without further scene understanding. On the other hand, for top-down views, it is too hard for many classes (e.g. frogs and lizards), since every orientation looks equally plausible.

TTT [171] can be viewed alternatively as one-sample unsupervised domain adaptation (UDA) – discussed in Subsection 3.1. UDA uses many unlabeled samples from the test distribution (target); TTT uses only one target sample and creates a special case – this sample can be the test input itself. We find this idea powerful in two ways: 1) We no longer need to prepare any target data in advance. 2) The learned model no longer needs to generalize to other target data – it only needs to "overfit" to the single training sample because it is the test sample.

Other papers following [171] have worked on related but different problem settings, assuming access to an entire dataset (e.g. TTT++ [112]) or batch (e.g. TENT [190]) of test inputs from the same distribution. Our paper does not make such assumptions, and evaluates on each single test sample independently.

Self-Supervision by Spatial Autoencoding

Using autoencoders for representation learning goes back to [186]. In computer vision, one of the earliest works is context encoders [136], which predicts a random image region given its context. Recent works [72, 17, 196] combine spatial autoencoding with Vision Transformers (ViT) [45] to achieve state-of-the-art performance. The most successful work is masked autoencoders (MAE) [72]: it splits each image into many (e.g. 196) patches, randomly masks out majority of them (e.g. 75%), and trains a model to reconstruct the missing patches by optimizing the mean squared error between the original and reconstructed pixels.

MAE pre-trains a large encoder and a small decoder, both ViTs [45]. Only the encoder is used for a downstream task, e.g. object recognition. The encoder features become inputs to a task-specific linear projection head. There are two common ways to combine the pre-trained

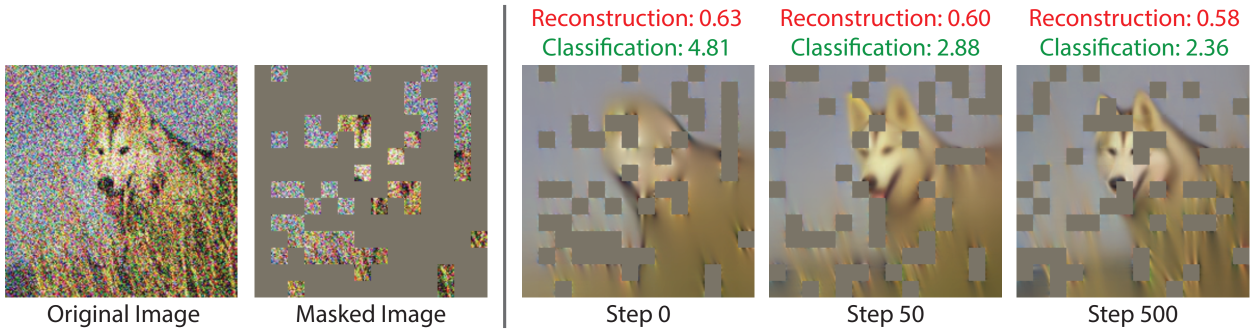


Figure 3.1: We train an MAE to reconstruct each test image at test time, masking 75% of the input patches. The three reconstructed images on the right visualize the progress of this one-sample learning problem. Loss of the main task (green) – object recognition – keeps dropping even after 500 steps of gradient descent, while the network continues to optimize for reconstruction (red). The unmasked patches are not shown on the right since they are not part of the reconstruction loss.

encoder and untrained head. 1) Fine-tuning: both the encoder and head are trained together, end-to-end, for the downstream task. 2) Linear probing: the encoder is frozen as a fixed feature extractor, only the head is trained. We refer back to these training options in the next section.

3.2 Method

At a high level, our method simply substitutes MAE [72] for the self-supervised part of TTT [171]. In practice, making this work involves many design choices.

Architecture. Our architecture is Y-shaped (like in [171]): a feature extractor f simultaneously followed by a self-supervised head g and a main task head h. Here, f is exactly the encoder of MAE, and g the decoder, both ViTs. We intentionally avoid modifying them to make clean comparison with [72]. For the main task (e.g. object recognition) head h, [72] uses a linear projection from the dimension of the encoder features to the number of classes. The authors discuss h being a linear layer as mostly a historic artifact. We also experiment with a more expressive main task head – an entire ViT-Base – to strengthen our baseline.

Training setup. Following standard practice, we start from the MAE model provided by the authors of [72], with a ViT-Large encoder, pre-trained for reconstruction on ImageNet-1k [40]. There are three ways to combine the encoder with the untrained main task head: fine-tuning, probing, and joint training. We experiment with all three, and choose probing with h being a ViT-Base, a.k.a. *ViT probing*, as our default setup, based on the following considerations:

1) **Fine-tuning**: train $h \circ f$ end-to-end. This works poorly with TTT because it does not encourage feature sharing between the two tasks. Fine-tuning makes the encoder specialize to recognition at training time, but subsequently, TTT makes the encoder specialize to reconstruction at test time, and lose the recognition-only features that h had learned to rely on. Fine-tuning is not used by [171] for the same reason.

2) **ViT probing**: train h only, with f frozen. Here, h is a ViT-Base, instead of a linear layer as used for linear probing. ViT probing is much more lightweight than both fine-tuning and joint-training. It trains 3.5 times fewer parameters than even linear fine-tuning (86M vs. 306M). It also obtains higher accuracy than linear fine-tuning without aggressive augmentations on the ImageNet validation set.

3) **Joint training**: train both $h \circ f$ and $g \circ f$, by summing their losses together. This is used by [171] with rotation prediction. But with MAE, it performs worse on the ImageNet validation set, to the best of our ability, than linear / ViT probing. If future work finds a reliable recipe for a joint training baseline, our method can easily be modified to work with that.

Training-time training. Denote the encoder produced by MAE pre-training as f_0 (and the decoder, used later for TTT, as g_0). Our default setup, ViT probing, produces a trained main task head h_0:

$$h_0 = \arg\min_h \frac{1}{n} \sum_{i=1}^{n} l_m(h \circ f_0(x_i), y_i). \tag{3.1}$$

The summation is over the training set with n samples, each consisting of input x_i and label y_i. The main task loss l_m in our case is the cross entropy loss for classification. Note that we are only optimizing the main task head, while the pre-trained encoder f_0 is frozen.

Augmentations. Our default setup, during training-time training, only uses image cropping and horizontal flips for augmentations, following the protocol in [72] for pre-training and linear probing. Fine-tuning in [72] (and [17, 196]), however, adds aggressive augmentations on top of the two above, including random changes in brightness, contrast, color and sharpness[1]. Many of them are in fact distribution shifts in our evaluation benchmarks. Training with them is analogous to training on the test set, for the purpose of studying generalization to new test distributions. Therefore, we choose not to use these augmentations, even though they would improve our results at face value.

Test-time training. At test time, we start from the main task head h_0 produced by ViT probing, as well as the MAE pre-trained encoder f_0 and decoder g_0. Once each test input x arrives, we optimize the following loss for TTT:

$$f_x, g_x = \arg\min_{f,g} l_s(g \circ f(\text{mask}(x)), x). \tag{3.2}$$

[1]Other augmentations used by fine-tuning in [72] are: interpolation, equalization, inversion, posterization, solarization, rotation, shearing, random erasing, and translation. These are taken from RandAugment [38].

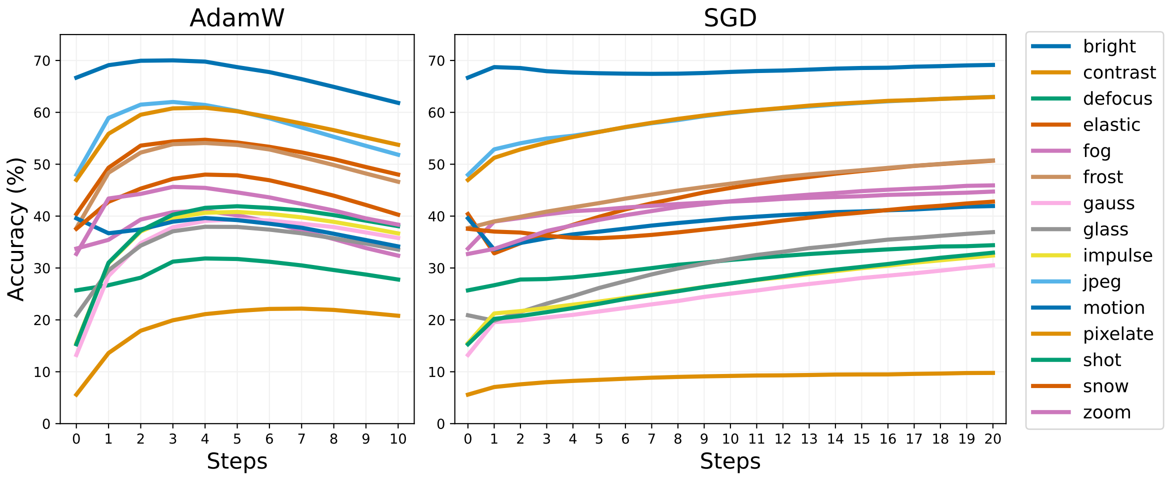


Figure 3.2: We experiment with two optimizers for TTT. MAE [72] uses AdamW for pre-training. But our results (left) show that AdamW for TTT requires early stopping, which is unrealistic for generalization to unknown distributions without a validation set. We instead use SGD, which keeps improving performance even after 20 steps (right).

The self-supervised reconstruction loss l_s computes the pixel-wise mean squared error of the decoded patches relative to the ground truth. After TTT, we make a prediction on x as $h \circ f_x(x)$. Note that gradient-based optimization for Equation 3.2 always starts from f_0 and g_0. When evaluating on a test set, we always discard f_x and g_x after making a prediction on each test input x, and reset the weights to f_0 and g_0 for the next test input. By test-time training on the test inputs independently, we do not assume that they come from the same distribution.

Optimizer for TTT. In [171], the choice of optimizer for TTT is straightforward: it simply takes the same optimizer setting as during the last epoch of training-time training of the self-supervised task. This choice, however, is not available for us, because the learning rate schedule of MAE reaches zero by the end of pre-training. We experiment with various learning rates for AdamW [118] and stochastic gradient descent (SGD) with momentum. Performance of both, using the best learning rate respectively, is shown in Figure 3.2.

AdamW is used in [72], but for TTT it hurts performance with too many iterations. On the other hand, more iterations with SGD consistently improve performance on all distribution shifts. Test accuracy keeps improving even after 20 iterations. This is very desirable for TTT: the single test image is all we know about the test distribution, so there is no validation set to tune hyper-parameters or monitor performance for early stopping. With SGD, we can simply keep training.

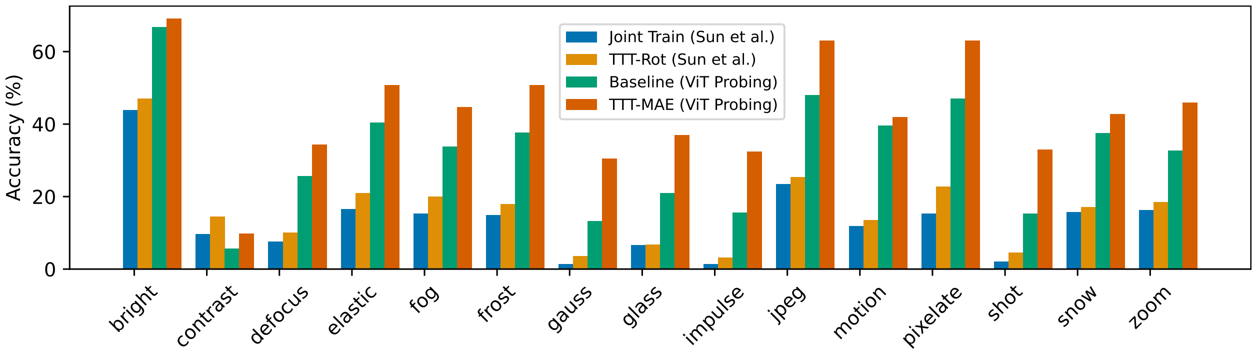


Figure 3.3: Accuracy (%) on ImageNet-C, level 5. Our method, TTT-MAE, significantly improves on top of our baseline, which already outperforms the method of [171]. See Subsection 3.3 for details. Numbers for our baseline and TTT-MAE can be found in the last two rows of Table 3.2. Numbers for Sun et al. are taken from [171].

3.3 Empirical Results

Implementation Details

In all experiments, we use MAE based on the ViT-Large encoder in [72], pre-trained for 800 epochs on ImageNet-1k [40]. Our ViT-Base head h takes as input the image features from the pre-trained MAE encoder. There is a linear layer in between that resizes those features to fit as inputs to the head, just like between the encoder and the decoder in [72]. A linear layer is also appended to the final class token of the head to produce the classification logits.

TTT is performed with SGD, as discussed, for 20 steps, using a momentum of 0.9, weight decay of 0.2, batch size of 128, and fixed learning rate of 5e-3. The choice of 20 steps is purely computational; more steps will likely improve performance marginally, judging from the trend observed in Figure 3.2. Most experiments are performed on four NVIDIA A100 GPUs; hyper-parameter sweeps are ran on an industrial cluster with V100 GPUs.

Like in pre-training, only tokens for non-masked patches are given to the encoder during TTT, whereas the decoder takes all tokens. Each image in a TTT batch has a different random mask. We use the same masking ratio as in MAE [72]: 75%. We do not use any augmentation on top of random masking for TTT. Specifically, every iteration of TTT is performed on the same 224×224 center crop of the image as we later use to make a prediction; the only difference being that predictions are made on images without masking.

We optimize both the encoder and decoder weights during TTT, together with the class token and the mask token. We have experimented with freezing the decoder and found that the difference, even for multiple iterations of SGD, is negligible. This is consistent with the observations in [171]. We also tried reconstruction loss both with and without normalized pixels, as done in [72]. Our method works well for both, but slightly better for normalized pixels because the baseline is slightly better. We also include these results in the appendix.

	Frost		*JPEG*		*Brightness*		*Elastic*	
	[171]	Ours	[171]	Ours	[171]	Ours	[171]	Ours
Jellyfish	-3	**0**	-6	**5**	**2**	-1	-2	**3**
Holster	-1	**4**	-2	**6**	**1**	0	-1	**14**
Chiton	-1	**3**	-3	**8**	-1	**2**	-3	**8**
Fire salamander	1	**3**	-1	**3**	2	**5**	**2**	**2**
Spaghetti squash	-3	**4**	-2	**5**	2	**3**	1	**6**

Table 3.1: Changes after TTT (-Rot and -MAE) in number of correctly classified images, out of 50 total for each category. On these rotation invariant classes, TTT-Rot [171] hurts performance, while TTT-MAE still helps, thanks to the generality of MAE as a self-supervised task. The four corruptions are selected for being the *most accurate* categories of the TTT-Rot baseline.

ImageNet-C

ImageNet-C [74] is a benchmark for object recognition under distribution shifts. It contains copies of the ImageNet [40] validation set with 15 types of corruptions, each with 5 levels of severity. Due to space constraints, results in the main text are on level 5, the most severe, unless stated otherwise. Results on the other four levels are in the appendix.

In the spirit of treating these distribution shifts as truly unknown, we do not use training data, prior knowledge, or data augmentations derived from these corruptions. This complies with the stated rule of the benchmark, that the corruptions should be used only for evaluation, and the algorithm being evaluated should not be corruption-specific [74]. This rule, in our opinion, helps community progress: the numerous distribution shifts outside of research evaluation cannot be anticipated, and an algorithm cannot scale if it relies on information that is specific to a few test distributions.

Main results. Our main results on ImageNet-C appear in Figure 3.3. Following the convention in [171], we plot accuracy only on the level-5 corruptions. Results on the other levels are in the appendix. All results here use the default training setup discussed in Section 4.2: take a pre-trained MAE encoder, then perform ViT probing for object recognition on the original ImageNet training set. Our baseline (green) applies the fixed model to the corrupted test sets. TTT-MAE (red) on top of our baseline significantly improves performance.

Comparing reconstruction vs. rotation prediction. The baseline for TTT-Rot, taken from [171], is called Joint Train in Figure 3.3. This is a ResNet [70] with 16-layers, after joint training for rotation prediction and object recognition. Because our baseline is much more advanced than the ResNet baseline, the former already outperforms the reported results of TTT-Rot, for all corruptions except contrast. [2] So comparison to [171] is more meaningful in relative terms: TTT-MAE has higher performance gains in all corruptions than TTT-Rot, on top of their respective baselines.

Rotation invariant classes. As discussed in Section 4.1, rotation prediction [58] is often too hard to be helpful, in contrast to a more general task like spatial autoencoding. In fact, we find entire classes that are rotation invariant, and show random examples from them in Table 3.1. [3] Not surprisingly, these images are usually taken from top-down views; rotation prediction on them can only memorize the auxiliary labels, without forming semantic features as intended. This causes TTT-Rot of [171] to hurt performance on these classes, as seen in Table 3.1. Also not surprisingly, TTT-MAE is agnostic to rotation invariance and still helps on these classes.

Training-time training. In Section 4.2, we discussed our choice of ViT probing instead of fine-tuning or joint training. The first three rows of Table 3.2 compare accuracy of these three designs. As discussed, joint training does not achieve satisfactory performance on most corruptions. While fine-tuning initially performs better than ViT probing, it is not amenable to TTT. The first three rows are only for training-time training, after which a fixed model is applied during testing. The last row is TTT-MAE after ViT probing, which performs the best across all corruption types. Numbers in the last two rows are the same as, respectively, for Baseline (ViT Probing) and TTT-MAE in Figure 3.3.

Other ImageNet Variants

We evaluate TTT-MAE on two other popular benchmarks for distribution shifts. ImageNet-A [78] contains real-world, unmodified, and naturally occurring examples where popular ImageNet models perform poorly. ImageNet-R [79] contains renditions of ImageNet classes, e.g. art, cartoons and origami. Both TTT-MAE and the baseline use the same models and algorithms, with exactly the same hyper-parameters as for ImageNet-C (Subsection 3.3). Our method continues to outperform the baseline by large margins: see Table 3.3.

[2]It turns out that for the contrast corruption type, the ResNet baseline of TTT-Rot is much better than our ViT baseline, even though the latter is much larger; TTT-MAE improves on the ViT baseline nevertheless.

[3]We find 5 of these classes through the following: 1) Take the ImageNet pre-trained ResNet-18 from the official PyTorch website [135]. 2) Run it on the original validation set, and a copy rotated 90-degree. 3) Eliminate classes for which the original accuracy is lower than 60%. 4) For each class still left, compare accuracy on the original vs. rotate images, and choose the 5 classes for which the normalized difference is the smallest.

	brigh	cont	defoc	elast	fog	frost	gauss	glass	impul	jpeg	motn	pixel	shot	snow	zoom
Joint Train	62.3	4.5	26.7	39.9	25.7	30.0	5.8	16.3	5.8	45.3	30.9	45.9	7.1	25.1	31.8
Fine-Tune	67.5	7.8	33.9	32.4	36.4	38.2	22.0	15.7	23.9	51.2	37.4	51.9	23.7	37.6	37.1
ViT Probe	68.3	6.4	24.2	31.6	38.6	38.4	17.4	18.4	18.2	51.2	32.2	49.7	18.2	35.9	32.2
TTT-MAE	**69.1**	**9.8**	**34.4**	**50.7**	**44.7**	**50.7**	**30.5**	**36.9**	**32.4**	**63.0**	**41.9**	**63.0**	**33.0**	**42.8**	**45.9**

Table 3.2: Accuracy (%) on ImageNet-C, level 5. The first three rows are fixed models without test-time training, comparing the three design choices discussed in Section 4.2. The third row, ViT probing, is our default baseline throughout the paper; it has the same numbers as the baseline in Figure 3.3. The last row is our method: TTT-MAE after ViT probing. This achieves the best performance across all corruption types; it has the same numbers as TTT-MAE in Figure 3.3.

ImageNet-A		*ImageNet-R*	
Baseline	TTT-MAE	Baseline	TTT-MAE
15.3	**21.3**	31.3	**38.9**

Table 3.3: Accuracy (%) on ImageNet-A [78] and ImageNet-R [79]; see Subsection 3.3 for details. Baseline is ViT Probing – the default – trained on the original ImageNet training set; it is in fact the same model as for ImageNet-C. Our method improves on the baseline for both datasets, using the same hyper-parameters as the rest of the paper.

Portraits Dataset

The Portraits dataset [59] contains American high school yearbook photos labeled by gender, taken over more than a century. It contains real-world distribution shifts in visual appearance over the years. We sort the entire dataset by year and split it into four equal parts, with 5062 images each. Between the four splits, there are visible low-level differences such as blurriness and contrast, as well as high-level change like hair-style and smile. [4]

We create three experiments: for each, we train on one of the first three splits and test on the fourth. Performance is measured by accuracy in binary gender classification. As expected, baseline performance increases slightly as the training split gets closer to test split in time.

Our model is the same ImageNet pre-trained MAE + ViT-Base (with sigmoid for binary classification), and probing is performed on the training split. We use exactly the same hyper-parameters for TTT. Our results are shown in Table 3.4. Our method improves on the baseline for all three experiments.

[4]The Portraits dataset was only accessed by the UC Berkeley authors.

Train Split	1	2	3
Baseline	76.1	76.5	78.2
TTT-MAE	**76.4**	**76.7**	**79.4**

Table 3.4: Accuracy (%) for binary classification on the Portraits dataset; see Subsection 3.3 for details. For each column, we train on the indicated split, and test on the fourth. Baseline is still ViT Probing (except with a sigmoid in the end). Our method improves on the baseline for all three training splits. We perform both regular training and TTT using the same hyper-parameters as the rest of the paper.

3.4 Theoretical Results

Why does TTT help? The theory of [171] gives an intuitive but shallow explanation: when the self-supervised task happens to propagate gradients that correlate with those of the main task. But this evasive theory only delegates one unknown – the test distribution, to another – the magical self-supervised task with correlated gradients, without really answering the question, or showing any concrete example of such a self-supervised task.

In this chapter, we show that autoencoding, i.e. reconstruction, is a self-supervised task that makes TTT help. To do so with minimal mathematical overhead, we restrict ourselves to the linear world, where our models are linear and the distribution shifts are produced by linear transformations. We analyze autoencoding with dimensionality reduction instead of masking, as we believe the two are closely related in essence.

The most illustrative insight, in our opinion, is that under distribution shifts, TTT finds a better *bias-variance trade-off* than applying a fixed model. The fixed model is biased because it is completely based on biased training data that do not represent the new test distribution. The other extreme is to completely forget the training data, and train a new model from scratch on each test input; this is also undesirable because the single test input is high variance, although unbiased by definition.

A sweet spot of the bias-variance trade-off is to perform TTT while remembering the training data in some way. In deep learning, this is usually done by initializing with a trained model for SGD, like for our paper and [171]. In this section, we retain memory by using part of the covariance matrix derived from training data.

It is well known that linear autoencoding is equivalent to principle component analysis (PCA). This equivalence simplifies the mathematics by giving us closed form solutions to the optimization problems both during training and test-time training. For the rest of the section, we use the term PCA instead of linear autoencoding, following convention of the theory community.

Problem setup. Let $x, y \sim P$, the training distribution, and assume that the population covariance matrix $\Sigma = \mathrm{Cov}(x)$ is known. PCA performs spectral decomposition on $\Sigma = UDU^\top$, and takes the top k eigenvectors $u_1, \ldots, u_k$ to project $x \in \mathbb{R}^d$ to $\mathbb{R}^k$. Throughout this section, we denote u_i as the ith column of the matrix U, and likewise for other matrices.

Assume that for each x, the ground truth y is a linear function of the PCA projection with $k = 1$:

$$y = w u_1^\top x, \tag{3.3}$$

for some known weight $w \in \mathbb{R}$. For mathematical convenience, we also assume the following about the eigenvalues, i.e. the diagonal entries of D:

$$\sigma_1 > \sigma_2 = \sigma_3 = \ldots = \sigma_d = \sigma. \tag{3.4}$$

At test time, nature draws a new $x, y \sim P$, but we can only see $\tilde{x}$, a corrupted version of x. We model a corruption as an unknown orthogonal transformation R, and $\tilde{x} = Rx$.

Algorithms. The baseline is to blithely apply a fixed model and predict

$$\hat{y} = w u_1^\top \tilde{x}, \tag{3.5}$$

as an estimate of y. Intuitively, this will be inaccurate if corruption is severe i.e. R is far from I. Now we derive the PCA version of TTT. We first construct a new (and rather trivial) covariance matrix with $\tilde{x}$, the only piece of information we have about the corruption. Let $\alpha \in [0, 1]$ be a hyper-parameter. We then form a linear combination of Σ with our new covariance matrix:

$$M(\alpha) = (1 - \alpha) \cdot \Sigma + \alpha \cdot \tilde{x}\tilde{x}^\top. \tag{3.6}$$

Denote its spectral decomposition as $M(\alpha) = V(\alpha) \cdot S(\alpha) \cdot V^\top(\alpha)$. We then predict

$$\hat{y} = w v_1^\top \tilde{x}. \tag{3.7}$$

Bias-variance trade-off. $\alpha = 0$ is equivalent to the baseline, and $\alpha = 1$ means that we exclusively use the single test input and completely forget about the training data. In statistical terms, α presents a bias-variance trade-off: $\alpha \downarrow 0$ means more bias, $\alpha \uparrow 1$ means more variance.

Theorem. Define the prediction risk as $\mathbb{E}[|\hat{y} - y|]$, where the expectation is taken over the corrupted test distribution. This risk is *strictly dominated* when $\alpha = 0$. That is, TTT with some hyper-parameter choice $\alpha > 0$ is, on average, strictly better than the baseline.

The proof is given in the appendix.

Remark on the assumptions. The assumption in Equation 3.3 is mild; it basically just says that PCA is at least helpful on the training distribution. It is also mild to assume that Σ and w are known in this context, since the estimated covariance matrix and weight asymptotically approach the population ground truth as the training set grows larger. The assumption on the eigenvalues in Equation 3.6 greatly simplifies the math, but a more complicated version of our analysis should exist without it. Lastly, we believe that our general statement should still hold for invertible linear transformation instead of orthogonal transformations, since they share the same intuition.

Chapter 4

Test-Time Training on Video Streams

Computer vision models deployed in the real world often take input data in the form of continuous video streams. However, most such models are trained with large collections of still images, e.g. the COCO dataset [110], causing a mismatch between training and testing. At test time, such models are applied to each video frame-by-frame, as if it was a collection of independent images. Thus, model predictions across visually similar frames can be very inconsistent. Simple averaging or temporal smoothing across predictions offer little improvement.

For a trained model that stays fixed at test time, a video is indeed no different from a collection of unordered frames. To take advantage of the temporal nature of our problem, we keep updating the model as it sees more of the video. At each timestep, before the model makes a prediction on the current frame, we first fine-tune it on this frame and recent frames in the past. After prediction, the model repeats the same process for the next frame, initializing with parameters from the current timestep.

Since there is no ground truth label on the test video, the fine-tuning technique inside this outer loop must not rely on explicit supervision. We experiment with a wide variety of baseline techniques for fine-tuning with self-supervision [171, 53], batch statistics [190, 151], and self-training using pseudo-labels [164]. While the techniques above have been introduced in the past, we make many design innovations to make them work effectively on video streams, inside our outer loop.

In terms of results, we make two contributions. First, we significantly improve prediction quality and visual consistency on three real-world datasets, for four tasks: instance, panoptic and semantic segmentation, and colorization. Prior work have considered the paradigm of updating a model on test videos for either short clips of synthetic data [171, 189], or depth estimation [174, 175, 203, 205, 119]. Our paper is the first to demonstrate the potential of this paradigm on real data, for general tasks that do not rely on stereo-specific knowledge. To facilitate future research in this under-explored direction, we collect a new dataset – COCO Videos – with dense annotations on very long videos of daily-life scenes. Figure 4.1 visualizes our results on COCO Videos for panoptic segmentation.

Our second contribution is more thought-provoking. Conventional wisdom in the continual

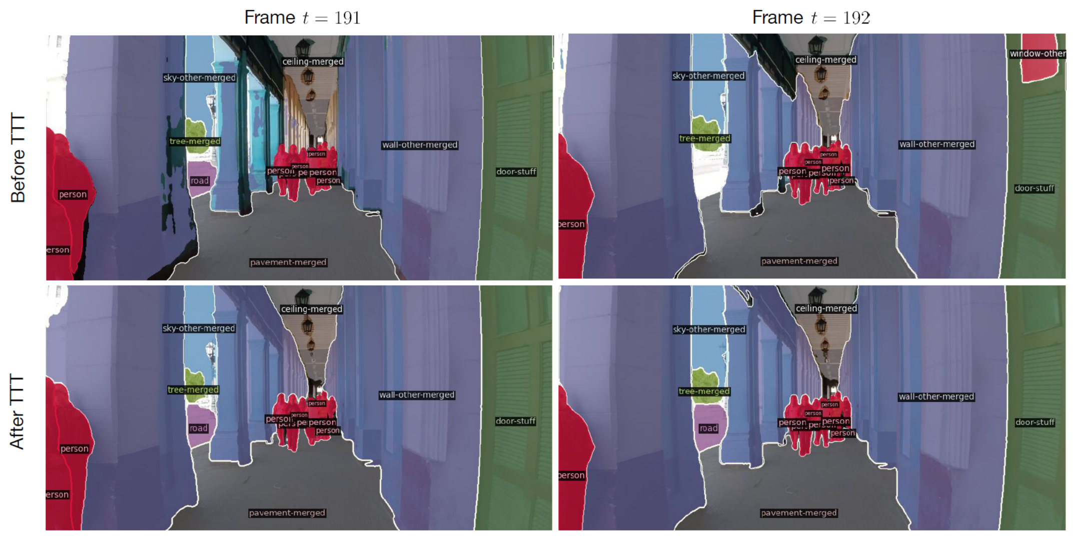


Figure 4.1: Panoptic segmentation predictions for adjacent frames from a video in our new COCO Videos dataset. **Top**: Baseline results produced by a fixed model. Predictions are inconsistent between the two frames. **Bottom**: Results after test-time training (TTT), by the same model, on the same frames as top. Predictions are now consistent and correct. Please zoom in to see the instance labels.

learning community believes that forgetting is harmful. Specifically, the best accuracy is achieved by remembering everything with an infinite replay buffer, given unlimited time and memory. In our experiments, however, forgetting can in fact be beneficial for accuracy. This is intuitive in hindsight, since frames from the distant past are less relevant for making a prediction in the present, and training on these frames creates bias. To make our intuition rigorous, we theoretically analyze the prediction error after test-time training, and characterize forgetting as finding a sweet spot in a bias-variance trade-off.

Our analysis suggests that the observed benefit of forgetting is not particular to our algorithm or technique, but a general property of our streaming setting. To the best of our knowledge, we are the first to discover the benefit of forgetting in a setting that can be instantiated with large-scale, real-world data. This discovery is consistent with studies in neuroscience [62], and the ubiquity of real-world data in our setting suggests that it might be more representative of how natural agents, e.g. humans, operate.

4.1 Related Work

Continual Learning

In the field of continual a.k.a. lifelong learning, a model learns a sequence of tasks in temporal order, and is asked to perform well on all of them [182, 64]. Here is the basic problem setting.

Each task is defined by a data distribution P_t, which produces a training set D_t^{tr} and a test set D_t^{te}. At each time t, the model is evaluated on all the test sets $D_1^{\text{te}}, \ldots, D_t^{\text{te}}$ of the past and present, and average performance is reported.

One solution is to train the model on all of $D_1^{\text{tr}}, \ldots, D_t^{\text{tr}}$, which collectively have the same distribution as the test sets. This is often referred to as the oracle with an infinite replay buffer that remembers everything. However, due to constraints in computation and memory, the model at time t is only allowed to train on D_t^{tr}, or a limited piece of memory (a.k.a. replay buffer) that contains a small fraction of the past training sets. Most solutions, therefore, focus on how to avoid forgetting, since majority of the past data can only be retained by the model parameters [150, 109, 117, 157, 97, 57].

Among the vast literature on continual learning, many papers extend beyond the basic setting above. Due to space constraints, we point to a few of the most relevant ones. [3] use continuous instead of discrete tasks across time. [141] and [47] perform self-supervised learning on unlabeled training sets, and evaluate the learned features on the test sets. [81], [103] and [134] use not only unlabeled training sets $D_1^{\text{tr}}, \ldots, D_t^{\text{tr}}$ (target), but also a labeled training set D_0^{tr} (source), thus sharing some terminology and techniques with the field of unsupervised domain adaptation. [117] and [42] use alternative metrics, e.g. forward transfer, to justify forgetting for reasons other than computational; they evaluate on synthetic datasets, e.g. rotated MNIST.

Online Learning on Videos at Test Time

The idea of training on test data has been explored in many papers over the past decade [88, 158, 131, 66, 137, 168, 49, 16, 95, 107, 112]. Here we discuss the ones working with videos.

The most relevant paper is [189]. They keep updating the model on each video without ground truth label, for the task of semantic segmentation. Inspired by their work, we use the same outer loop, but make design innovations that significantly improve performance, and expand the scope of results to four tasks on three datasets. All of our experiments and improvements are on real-world data. In comparison, only one experiment in [189] uses data without synthetic corruptions (CityScapes), where the videos are orders of magnitude shorter than ours (see Table 4.2), and their best method only makes 1.4% relative improvement. By controlling memory explicitly through a sliding window, we are also able to make novel investigations into the benefits of forgetting.

Another recent work on videos is [14]. They treat each video as a dataset of unordered frames instead of a stream, and use the same model on each video. In particular, they always have access to the entire video, including future frames. In contrast, we have access to only the current and past frames, and our model keeps learning over time. In addition, all of our results are on real world videos, while [14] experiment only on videos with artificial corruptions. From what we understand, their corruptions are also i.i.d. across frames of even the same video.

4.2 Approach

At a high level, most of the components of our algorithm have been introduced in the past. In practice, it requires many design innovations to combine them effectively.

Outer Loop

We consider a test video as a smoothly changing sequence of frames $x_1, \ldots, x_T$. We want to evaluate an algorithm on the test video following its temporal order, as if it was consumed by a human. Ground truth labels should never used by the algorithm. For each time t, the algorithm makes a prediction on x_t after receiving it from the environment, before seeing any future frame. Naturally, the past frames $x_1, \ldots, x_{t-1}$ are also available at the time of prediction, in addition to x_t. This is a key difference between the setting of [189] and ours.

Given a trained model F that takes a single image as input, the most naive baseline is to blithely run F frame-by-frame, predicting $F(x_t)$ at each timestep. Currently, this is indeed the most common mode of deploying a single-image model on videos. How to do better? We improve F by updating it on the test video. For each timestep t, we repeat the following: 1) Receive x_t. 2) Produce F_t through training on x_t and, if helpful, some of $x_1, \ldots, x_{t-1}$. 3) Predict $F_t(x_t)$. The second step is the key of this outer loop.

Baseline Techniques for Model Update

In principle, any technique that does not use ground truth labels can be used to update F inside the loop. We next discuss the most promising ones according to prior work.

Masked Autoencoders

[53] shows that masked autoencoders (MAE) [72] work well for test-time training (TTT) on still images. The self-supervised task – in this case, reconstruction from an input image with masked patches – needs its own prediction head. This requires us to modify the network architecture. We first split F into two parts as $F = h \circ f$, where f is the feature extractor, and h is the prediction head for the main task – in our case, segmentation or colorization. For F composed of a sequence of layers, this split is realized by making f the first few layers, and h the rest. Next we add a prediction head g, for the self-supervised task, after f. For symmetry, g has the same architecture as h, except the last layer mapping to a different output space.

Initializing g from scratch is undesirable for TTT since performance of the self-supervised task would then start from chance level. We prepare our model for TTT by training it jointly on both tasks, using the training set of images with labels. This design is different from that in [53], which first trains $g \circ f$, then trains h and keeps g frozen. We find their design to perform much worse in our setting. Denote the main task loss as ℓ_m, and the self-supervised loss as ℓ_s. We optimize those two losses together to produce:

$$g_0, h_0, f_0 = \arg\min_{g,h,f} \frac{1}{n} \sum_{i=1}^{n} [\ell_m(h \circ f(x_i), y_i) \\ + \ell_s(g \circ f(\tilde{x}_i), x_i)].$$

The summation is over the training set with n samples, each consisting of input x_i and label y_i. $\tilde{x}_i$ is x_i transformed as input for the self-supervised task, in our case, by masking 80% of the patches.

Given a single test image x (and $\tilde{x}$ transformed for the self-supervised task), [53] formulates a one-sample learning problem that solves for

$$g', f' = \arg\min_{g,f} \ell_s(g \circ f(\tilde{x}), x), \tag{4.1}$$

initializing from g_0 and f_0. A prediction for the main task is then made as $h_0 \circ f'(x)$. However, naively applying the above does not help performance in our experiments. In the next subsection, we are able to use many more frames for this learning problem, thanks to our streaming setting.

Self-Training

Self-training is a popular technique in semi-supervised learning [143, 147, 207, 9, 7] and domain adaptation [100, 208, 122, 111, 166]. It is also evaluated in [189], but is shown to produce inferior performance. Here we incorporate novel design choices that make it suitable for TTT.

We assume that for each test image x, the prediction $\hat{y}$ is also of the same shape in 2D, which is true in semantic segmentation and colorization. We also assume that F outputs a estimated confidence map $\hat{c}$ of the same shape as $\hat{y}$. Specifically, for pixel $x[i,j]$, $\hat{y}[i,j]$ is the predicted class of this pixel, and $\hat{c}[i,j]$ is the estimated confidence of $\hat{y}[i,j]$.

Self-training repeats many iterations of the following:
1) Start with an empty set of labels D for this iteration.
2) Loop over every $[i,j]$ location, add pseudo-label $\hat{y}[i,j]$ to D if $\hat{c}[i,j] > \lambda$, for a fixed threshold λ. 3) Train F to fit this iteration's set D, as if the selected pseudo-labels are ground truth labels. Our first design improvement is incorporating the confidence threshold λ. In [189], all predictions are pseudo-labels, regardless of confidence. Experiments show that for low λ, or with $\lambda = 0$ in [189], self-training is noisy and unstable, as expected.

However, for high λ, there is limited learning signal, e.g. little gradient, since f is already very confident about the pseudo-label. Our second design improvement, inspired by [164], is to make learning more challenging with an already confident prediction, by masking image patches in x. In [164], masking is applied sparingly on 2.5% of the pixels in average. We mask 80% or even 90% of the pixels, inspired by [72].

Adapting the Normalization Layers

Prior work [151] shows that simply recalculating the batch normalization (BN) statistics works well for unsupervised domain adaptation. [189] applies this technique to video streams by accumulating the statistics with a forward pass on each frame once it is revealed. Since modern transformers use layer normalization (LN) instead, we apply the same technique to LN. This is labeled as *LN Adapt* [151] in the tables.

The normalization layers (BN and LN) also contain parameters that modify the statistics. Tent [190] is an objective for learning only those parameters at test time, by minimizing the softmax entropy of the predicted distribution over classes. We update the LN statistics and parameters with Tent, in the same loop as our method.

Test-Time Training with Memory

We can naively apply any of the baseline techniques to a video by making the input x_t instead of x. However, this misses point of using a video. Like the single-image baseline using a fixed model, single-image TTT treats a video as a collection of unordered frames. Neither of the two can improve over time, which is only possible through memory, by retaining some information from the past frames $x_1, \ldots, x_{t-1}$ to help prediction on x_t.

Because evaluation is performed at each timestep only on the current frame, memory design should favor past data that are most relevant to the present. Fortunately, with the help of nature, the most relevant frames usually happen to be the most recent due to temporal smoothness – observations close in time tend to be similar. We design memory that favors recent frames in the following two ways.

Implicit memory. To initialize TTT on x_t, we have two choices. The first is g_0 and f_0. For each timestep, this resets the model parameters before TTT to those at the beginning of the video. Once a prediction is made, the new parameters after TTT are discarded. The second is g_{t-1} and f_{t-1}. Algorithmically, this simply keeps the models parameters. Results show that the second is usually better under temporal smoothness. It creates an implicit memory, since information carries over from the previous model, optimized on previous frames. It also happens to be more biologically plausible: we humans rarely reset our minds. In [53], only the first is discussed, because it does not assume smoothness between different inputs.

Explicit memory. While implicit memory is easy to implement, it is hard to control. One can certainly manipulate, for example, the learning rate, number of iterations, and regularization coefficients to influence the effect of initialization, but these delicate hyper-parameters also affect the optimization solver. To isolate the amount of memory for reliable analysis, we have to decouple the optimization problem from its solver.

While [189] uses the implicit memory discussed above, it still performs TTT one image at a time like in [171] and [53]. We add explicit control to memory by remembering past frames in a sliding window, also known as a replay buffer in continual learning. For each iteration of stochastic gradient decent, we uniformly sample a batch with replacement from the window, and calculate the total loss as the sum over all losses for elements in the batch.

Let k denote the window size. As an example, for TTT with MAE, we solve the following optimization problem at each timestep t:

$$g_t, f_t = \arg\min_{g,f} \frac{1}{k} \sum_{t'=t-k+1}^{t} \ell_s(g \circ f(\tilde{x}_{t'}), x_{t'}), \tag{4.2}$$

before predicting $h_0 \circ f_t(x_t)$. Masking is applied independently within and across batches.

4.3 Results

We experiment with four applications on three real-world datasets: 1) semantic segmentation on a public dataset of urban driving videos; 2) instance and panoptic segmentation on a new dataset we collected and annotated, of videos with COCO objects in daily scenes; 3) colorization on our new dataset and a collection of black and white films.

Architecture. Our default is Mask2Former [30], which has recently achieved state-of-the-art performance on many semantic, instance and panoptic segmentation benchmarks. Our method is generally applicable to modern network architectures, and does not rely on any particular property of Mask2Former. Our Mask2Former uses a Swin-S [113] backbone. For MAE, this is also the shared feature extractor (a.k.a. encoder) f. Everything following the backbone in the original architecture is taken as the main task head h, and our self-supervised head (a.k.a. decoder) g copies the architecture of h except the last layer.

Class balancing. [189] proposes a heuristic that is applicable when implicit memory is used: record the number of predicted classes, for the initial model F_0 and the current model F_t. They reset the model parameters when the difference is large enough, since the predictions of F_t have likely collapsed. To compare with [189], we also evaluate this heuristic on self-training and Tent, appended with the *-Class* label.

Semantic Segmentation on KITTI-STEP

Dataset. KITTI-STEP [191] contains 12 training videos and 9 validation videos of urban driving scenes. At the rate of 10 frames-per-second, these videos are the longest – up to 106 seconds – among public datasets with dense pixel-wise annotations. Since we do not perform regular training on KITTI-STEP, we select hyper-parameters on the validation set, and use the larger training set as the test set. We report results both.

Training. KITTI-STEP has exactly the same 19 categories as CityScapes [34], another driving dataset of individual images instead of videos. We take the publicly available Mask2Former model pre-trained on CityScapes images. For TTT-MAE, this is also the initialization for joint training, labeled *Joint MAE*.

Main results. Quantitative results with ablations, measured by mean intersection over union (IoU), are in Table 4.1. We make a few observations:

1. Using both forms of memory, TTT-MAE performs the best. For semantic segmentation, such improvements on a state-of-the-art baseline are considered highly significant. Joint

Method	**Validation**	**Test**	**Time**
	Fixed models.		
DeepLab	42.0	41.6	-
SegFormer	53.1	50.8	-
Mask2Former	53.8	52.5	-
Joint MAE	53.5 (-0.6%)	52.5 (0.0%)	1.0
	No memory.		
TTT-MAE	53.6 (+0.2%)	52.5 (0.0%)	5.2
Self-Train	54.3 (+0.9%)	53.1 (+1.1%)	115.6
ST Improve	53.8 (0.0%)	52.8 (+0.6%)	121.4
LN Adapt	53.6 (-0.4%)	52.5 (0.0%)	2.8
Tent	53.7 (-0.2%)	52.6 (+0.2%)	3.4
	Implicit memory only: no reset.		
TTT-MAE	55.4 (+3.6%)	53.8 (+2.5%)	5.0
Self-Train	12.5 (-76.8%)	10.5 (-80.0%)	108.8
ST-Class	55.2 (+2.6%)	53.7 (+2.3%)	111.0
ST Improve	53.9 (+0.2%)	53.0 (+1.0%)	114.6
LN Adapt	53.8 (0.0%)	52.5 (0.0%)	2.6
Tent	53.8 (0.0%)	52.4 (-0.2%)	3.0
Tent-Class	53.8 (0.0%)	52.5 (0.0%)	5.2
	Explicit memory only: window size 16.		
TTT-MAE	53.7 (+0.4%)	52.8 (+0.6%)	5.6
Self-Train	54.0 (+0.4%)	52.9 (+0.8%)	133.6
ST Improve	54.7 (+1.7%)	53.4 (+1.7%)	137.4
LN Adapt	53.8 (0.0%)	52.5 (0.0%)	3.0
Tent	53.9 (+0.2%)	52.7 (+0.4%)	3.0
	Both memory (ours): no reset, window size 16.		
TTT-MAE	**57.2 (+6.9%)**	**55.4 (+5.5%)**	5.2
Self-Train	11.4 (-78.8%)	9.2 (-82.5%)	126.2
ST-Class	56.1 (+4.3%)	54.4 (+3.6%)	127.8
ST Improve	57.0 (+5.9%)	55.4 (+5.5%)	129.4
LN Adapt	53.8 (0.0%)	52.5 (0.0%)	2.6
Tent	53.8 (0.0%)	52.6 (+0.2%)	2.8
Tent-Class	53.9 (+0.2%)	52.7 (+0.4%)	4.6

Table 4.1: Results for semantic segmentation in mean IoU (%) on KITTI-STEP. For implementations based on Mask2Former, the numbers in parenthesis indicate relative improvements. Average inference time is in seconds per frame. In all experiments, timing is performed on a single A100 GPU with 80G memory.

Dataset	Length	Frames	Rate
CityScapes-VPS [96]	1.8	3000	17
DAVIS [139]	3.5	3455	30
YouTube-VOS [197]	4.5	123,467	30
KITTI-STEP [191]	40	8,008	10
COCO Videos (Ours)	309	23,925	10

Table 4.2: Comparison of video datasets with annotations for segmentation. The columns are: average length per video in seconds, total number of frames in the entire dataset, and the rate in frames per second. The videos in our new dataset – COCO Videos – are orders of magnitude longer than other publicly available ones.

training alone does not improve on baseline, indicating that the improvements of TTT-MAE come from updating the model at test time.

2. Our TTT-MAE optimizes for only 1 iteration per frame, and is only 5x slower than the baseline. Comparing with [53], which optimizes for 20 iterations per image, we take advantage of temporal smoothness and implicit memory to get a better initialization for every frame.

3. When implicit memory is used, performance of self-training [189] is very bad without class balancing. This is consistent with the observations in [189]. Our two design improvements, labeled *ST Improve* in Table 4.1, make self-training almost competitive with TTT-MAE.

4. Baseline techniques that adapt the normalization layers alone help little in these evaluations. In [189], they are also shown to barely improve results on the single real-world dataset (CityScapes), even though they improve significantly on the other datasets with synthetic corruptions.

Additional baselines. Mask2Former was not evaluated by the authors on KITTI-STEP. To verify that the pre-trained model (69M) is already the state-of-the-art on KITTI-STEP, we compare its performance with two other popular models of comparable size: SegFormer B4 [195] (64.1M), and DeepLabV3+/RN101 [27] (62.7M), used by [189]. We also experiment with test-time augmentation of the input, applying the default recipe in the codebase to the baseline, budgeting 100 seconds per frame. This improves mean IoU on the validation set modestly by 1.2%.

Temporal smoothing. As a sanity check, we also implement temporal smoothing on the baseline, by averaging the predictions across a short window. The window size is selected to optimize performance after smoothing on the validation set. This improves the baseline by only 0.4% mean IoU on average.

Method	**Instance (AP)**	**Panoptic (PQ)**	**Time**
Fixed models.			
Mask2Former	4.52	9.34	-
Joint MAE	4.53 (+0.2%)	9.21 (-1.4%)	1.3
No memory.			
TTT-MAE	6.59 (+45.8%)	13.22 (+41.5%)	4.6
Implicit memory only: no reset.			
TTT-MAE	6.90 (+52.7%)	14.11 (+57.2%)	3.6
LN Adapt	4.62 (+2.2%)	10.03 (+8.3%)	2.8
Tent	4.99 (+10.4%)	10.74 (+16.8%)	3.2
Explicit memory only: window size 16.			
TTT-MAE	6.71 (+48.5%)	13.25 (+41.9%)	4.4
LN Adapt	4.76 (+5.3%)	9.48 (+1.5%)	3.6
Tent	5.88 (+30.1%)	11.38 (+21.8%)	3.4
Both memory (ours): no reset, window size 16.			
TTT-MAE	**8.12 (+79.6%)**	**15.43 (+65.2%)**	3.8
LN Adapt	4.51 (-0.2%)	9.35 (+0.1%)	3.4
Tent	5.57 (+23.2%)	10.39 (+11.2%)	3.6
Tent-Class	5.88 (+30.1%)	11.63 (+24.5%)	5.0

Table 4.3: Results for instance and panoptic segmentation on COCO Videos. The metrics for instance and panoptic segmentation are, respectively, average precision (AP) and panoptic quality (PQ). Average inference time is in seconds per frame.

COCO Videos

While KITTI-STEP already contains the longest annotated videos among publicly available datasets, they are still far too short for studying long-term phenomenon such as memory and forgetting. KITTI-STEP videos are also limited to driving scenarios, a small subset of the diverse scenarios in our daily lives. These limitations motivate us to annotate our own videos.

We collected 7 videos, each about 5 minutes, annotated by professionals, in the same format as for COCO instance and panoptic segmentation [110]. To put things into perspective, each of the 7 videos alone contains more frames, at the same rate, than all of the videos combined in the KITTI-STEP validation set. We compare this new dataset with other public datasets in Table 4.2.

All videos are egocentric, similar to the visual experience of a human walking around. In particular, they do not follow any tracked object like in Oxford Long-Term Tracking [181] or ImageNet-Vid [156]. Objects leave and enter the camera's view all the time. Unlike KITTI-STEP and CityScapes that focus on self-driving scenes, our videos are both indoors and

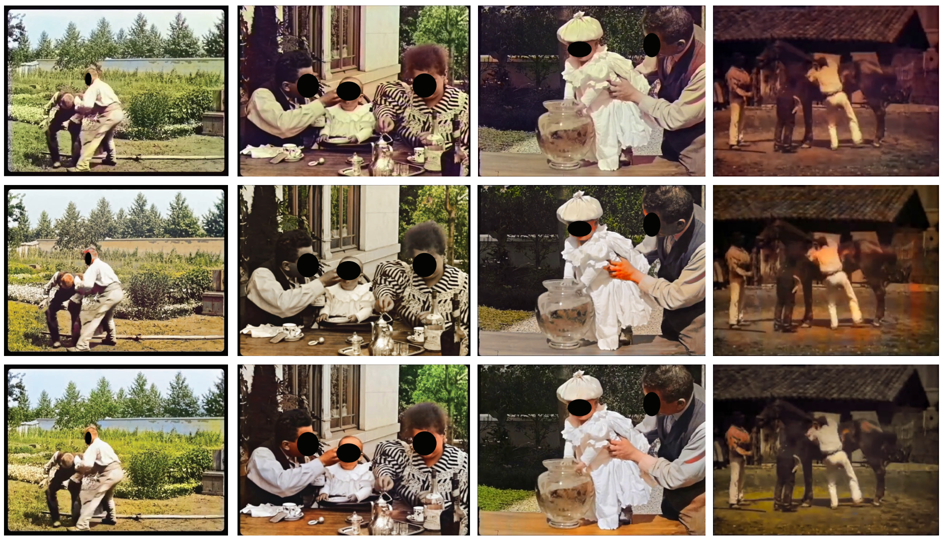


Figure 4.2: Samples results for video colorization on the Lumiere Brothers films. **Top**: Results using [201]. **Middle**: Results using our own baseline, which is already comparable, if not superior to [201]. **Bottom**: Results after applying our best performing method on the baseline. Our colors are more vibrant and consistent within regions.

Method	**FID** $\downarrow$	**IS** $\uparrow$	**LPIPS** $\uparrow$	**PSNR** $\uparrow$	**SSIM** $\uparrow$
Zhang et al.[201]	62.39	5.00 $\pm$ 0.19	0.180	22.27	**0.924**
Mask2Former [30]	59.96	5.23 $\pm$ 0.12	0.216	20.42	0.881
Ours	**56.47**	**5.31** $\pm$ **0.18**	**0.237**	**22.97**	0.901

Table 4.4: Results for video colorization on COCO Videos. FID: Fréchet Inception Distance. IS: Inception Score (standard deviation is naturally available). LPIPS: Learned Perceptual Image Patch Similarity. PSNR: Peak Signal-to-Noise Ratio. SSIM: Structural Similarity.

outdoors, taken from diverse locations such as sidewalks, markets, schools, offices, restaurants, parks and households. We plan to expand this dataset into 10 videos by rebuttal.

For context, our baseline is the state-of-the-art on the COCO validation set, with 44.9 AP for instance and 53.6 PQ for panoptic segmentation. The fact that baseline performance drops to single digits in Table 4.3 speaks about the challenging nature of our videos, and the fragility of single-image models when evaluated on videos in the wild.

Because COCO Videos is very large-scale, we only use TTT-MAE – the best-performing baseline technique – for the ablated versions. Self-training is not applicable here because for instance and panoptic segmentation, the model does not return a confidence per object instance. This points to a limitation of self-training that makes it not as generally applicable as TTT-MAE.

Training. We take the publicly available Mask2Former model pre-trained on images in the COCO training set, for instance and panoptic segmentation respectively. Analogous to our procedure for KITTI-STEP, joint training for TTT-MAE is also on COCO images, and our 7 videos are only used for evaluation. We use exactly the same hyper-parameters as tuned on the KITTI-STEP validation set, for all algorithms considered. That is, all of our results for COCO Videos were completed in a single run.

Main results. Quantitative results are in Table 4.3. Figure 4.1 provides a snapshot of our qualitative results. Please see supplementary materials for more visualizations. Using both forms of memory, TTT-MAE improves on the baseline by almost 80% relative performance on instance segmentation, and 65% on panoptic. Improvements of this magnitude on a state-of-the-art baseline is usually considered dramatic, especially given the hyper-parameters are tuned on another dataset. Here TTT-MAE is only 3x slower than the baseline. Comparing with KITTI-STEP where it is 5x slower, the augmentations for COCO are much simpler and faster. Methods with reset are slower than without because of checkpoint loading.

Video Colorization

The goal of colorization is to add realistic RGB colors to a gray-scale image. Since many old films were taken in black and white, single-image models are often applied to colorize videos [102, 198]. For this application, we try to demonstrate the generality of our approach, not to achieve the state-of-the-art in colorization. Because running on COCO Videos is expensive, we only evaluate our best-performing method – TTT-MAE with both forms of memory.

Dataset. For quantitative results, we colorize COCO Videos, by processing the 7 videos into black and white; this enables us to compare with the original videos in RGB. For qualitative results, we also colorize the 10 original black-and-white Lumiere Brothers films from 1895, roughly 40 seconds each, at the rate of 10 fps.

Training. Following [201], we simply treat colorization as a supervised learning problem. We use the same architecture as for segmentation – Swin Transformer with two heads, trained on ImageNet [40] to predict the colors given input images processed as gray-scale. We only make the necessary changes to map to a different output space, and do not use domain-specific techniques, e.g., perceptual losses, adversarial learning, and diffusion models. Our bare-minimal baseline already achieves results comparable, if not superior, to those in [201]. Our method uses exactly the same hyper-parameters as for segmentation. All of our colorization experiments were completed in a single run.

Main results. Quantitative results are in Table 4.4. Figure 4.2 provides a snapshot of our qualitative results. Our method outperforms the baseline and [201] on all metrics except SSIM. It is a field consensus [201, 202] that PSNR and SSIM often misrepresent performance because colorization is inherently multi-modal. but we still include them for completeness. Please see supplementary materials for the complete set of the original and colorized videos. Our method visually improves the quality in all of them comparing to the baseline, especially in terms of consistency across frames.

Setting. Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable with L-Lipschitz derivatives (colloquially, f is smooth). For mathematical convenience, assume that the inputs x_t move by a constant interval $\delta > 0$, that is, $x_t = t\delta$. After prediction, we are given feedback in the form of a noisy label

$$y_t = f(x_t) + \epsilon_t,$$

where ϵ_t is any random variable with mean 0 and variance $\sigma^2 < \infty$; specifically, these noise terms do not need to be identically distributed. The noisy labels model the potentially inaccurate pseudo-labels we generate for self-training.

Algorithm. We perform test-time training on a linear model with those noisy pseudo-labels. Consider a fixed sliding window of size k, and a local ordinary least squares estimator with data $\{(x_{t-k}, y_{t-k}), \ldots, (x_{t-1}, y_{t-1})\}$. Its current prediction on x_t is $\hat{y}_t = \alpha_t x_t + \beta_t$, where α_t and β_t are the current weight and bias.

Theorem. The average prediction loss (a.k.a. risk) for estimating f can be decomposed as:

$$\mathbb{E}\left[f(x_t) - \hat{y}_t\right]^2 = B^2 + V,$$

where B and V are called the bias and variance, and

$$B = \left(f(x_t) - \mathbb{E}[\hat{y}_t]\right)\right)^2 \leq 6k^2 L\delta,$$

$$V = \mathbb{V}(\hat{y}_t) \leq 4\sigma^2/k.$$

The proof is given in Section 1 of the appendix.

Discussion. Our theorem indicates that the phenomenon of *forgetting emerges naturally when optimizing for the prediction risk.* With a smaller window, i.e. more forgetting, the bias decreases while the variance increases. This matches our intuition derived from empirical observations: a smaller window contains data more relevant for prediction at our current time and place, i.e. less *dataset bias*, but we also pay a price of $O(1/k)$ because the data is noisy with $\sigma > 0$.

Chapter 5

Test-Time Training in Robotics

Many popular frameworks for controller design are based on the robot's model of dynamics. In the real world, however, this model can often turn out to be inaccurate, due to, for example, misspecification of the robot's physical parameters, mechanical wear and tear, and deployment-time interventions such as additional payload. While a well designed controller is robust to small inaccuracies in the dynamics, large deviations may significantly degrade its performance. Our goal is to make corrections to the model behind the controller during deployment, through online learning using onboard sensors. Since the nature of a model is to predict the future given the past, data for supervised learning of dynamics can be collected automatically without human supervision, as time goes on and the future is revealed.

Because data are generated along the controller's trajectory that we are trying to improve, they might not contain enough information about the entire system. Nevertheless, we find it sufficient to limit the scope of learning to a *local* neighborhood of the current point in the current trajectory, instead of the entire system, if the learned model is updated in real time as the trajectory evolves.

Fortunately, even globally complex systems, such as the highly nonlinear hybrid systems for legged locomotion, can be locally simple. Therefore, we also find it sufficient to learn with only a *time-varying, locally linear model*, which is computationally feasible to be updated in real time.

We first develop the intuition of online learning into a method for controllers that drive the outputs to the desired behavior based on control-affine models. We then analyze this method's theoretical properties, and evaluate it in two applications for legged locomotion.

Conventions

In this chapter, vectors ($\mathbf{a}, \boldsymbol{\alpha}$) are bold and lowercase, matrices ($\mathbf{A}, \boldsymbol{\Omega}$) are bold and uppercase, scalars and functions (of all type signatures) are not bold. We assemble matrices and vectors like in MATLAB: $[\mathbf{A}, \mathbf{B}]$ concatenates $\mathbf{A}$ and $\mathbf{B}$ horizontally with a comma, and $[\mathbf{A}; \mathbf{B}]$ concatenates them vertically with a semicolon. $\mathbf{0}_n$ denotes the $n \times n$ matrix of zeros, and $\mathbf{1}_n$ denotes the $n \times n$ identity matrix. Also, $\|\cdot\|$ denotes the 2-norm for vectors (Euclidean

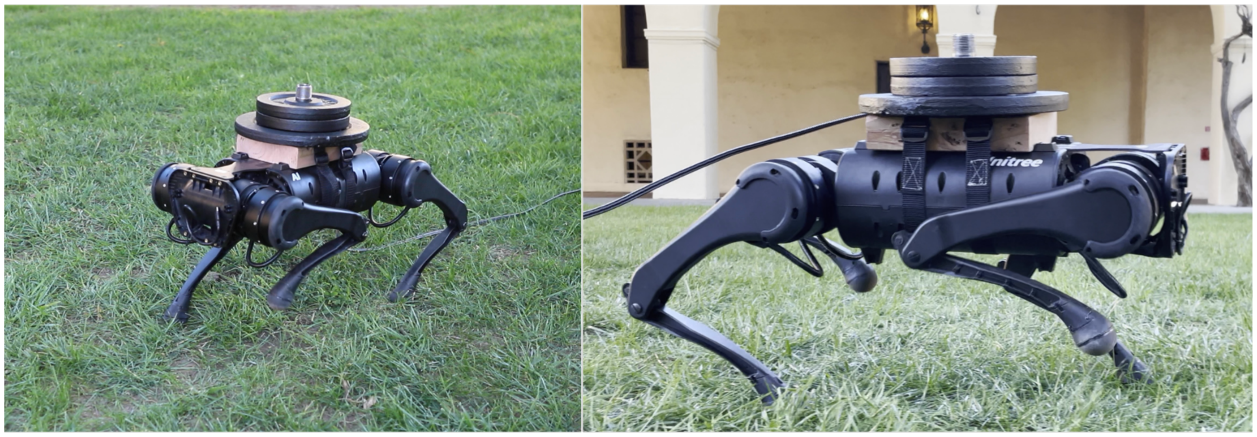


Figure 5.1: The 12 kg A1 robot carrying 10 kg of payload with our method, tested for trotting in place and walking forward.

norm) and matrices (spectral norm), unless stated otherwise. We express quantities in the nominal dynamics $\bar{\alpha}$ with a bar, in the residual dynamics $\tilde{\alpha}$ with a tilde, and in the true (plant) dynamics α without anything on top.

5.1 Related Work

System identification

Given a system with known form but unknown parameters, system identification (sysID) estimates these parameters from signals given by the system ([11]). Recent papers have applied sysID for inertial parameters of a humanoid ([13, 123]). The parameters are assumed to be constant in time, and estimation is performed before the deployment of a controller. Thinking of identification as training and deployment as testing, sysID trains a model before deployment, and keeps the model fixed during testing. Since the goal is to model the system's behavior globally across the entire state space, sysID usually requires driving the system to diverse enough states, using diverse enough inputs. This requirement is known as persistence of excitation in control theory, and might be difficult to satisfy without many samples from the plant. In contrast, we only model the system's behavior locally, around the small neighborhood of our current state, learning a linear model even for complex systems with relatively few samples.

Learning dynamics

There is also a developing community in machine learning, modeling dynamics of the environment from interactions and observations ([109, 2, 18, 194, 89, 5, 142, 129]). It has roughly the same goal as sysID, but often uses powerful tools from deep learning, and does not assume any specific form of the system; here, learning often produces a general prediction model. We diverge from this community in the global vs. local aspect (like from sysID), but embrace its philosophy of learning a general model with parameters that might not be interpretable.

Adaptive control

The intuition of adaptive control is to change the controller's parameters during deployment ([10]). Online system identification ([161]) is the most relevant sub-field, since it directly concerns the model behind the controller. It has been successfully applied in manipulation ([162, 133, 20, 90, 99]), and for the location and inertial parameters of the center of mass of a quadruped ([177]). For online sysID, the parameters considered are very specific, and estimation relies on the physics of the model and the particular controller for the application. Our work considers parameters in a much more general sense closer to that of the machine learning community. Our parameters are functions of the state, thus are inherently time-varying and abstract. In fact, in the control-affine form, every term of our dynamics is updated in real time as the state evolves. Furthermore, unlike sysID (online or not) whose goal is to identify the parameters, our goal is simply to give accurate predictions for the next timestep, again closer to the goal of learning. This allows our method to not rely on the specific meanings of the parameters and instead work with general model-based controllers. Another relevant sub-field is L1 adaptive control ([130, 85]), which, like our work, concerns the residual dynamics, but does not use learning.

Online learning

Our work performs supervised learning online, which has long been a subject of research in machine learning ([152, 22, 128]). The two central questions are: where does the label come from, and how is learning evaluated. Traditionally ([68]), learning has been evaluated with regret, and labels can come from a potentially adversarial oracle. Recently, the computer vision community has been using self-supervised tasks to provide labels ([173, 169, 170, 65, 103]), and the continual learning community has been evaluating with forward and backward predictions ([109]).

5.2 Method

Unknown Dynamics and Linear Residual Models

Given a robotic system that is characterized by rigid-body dynamics, we denote $\mathbf{x} \in \mathbb{R}^n$ as its state, $\mathbf{u} \in \mathbb{R}^m$ its vector of control inputs, and $\mathbf{y} \in \mathbb{R}^d$ its vector of outputs. The output dynamics can almost always be written as a second-order system of the following form ([6]),

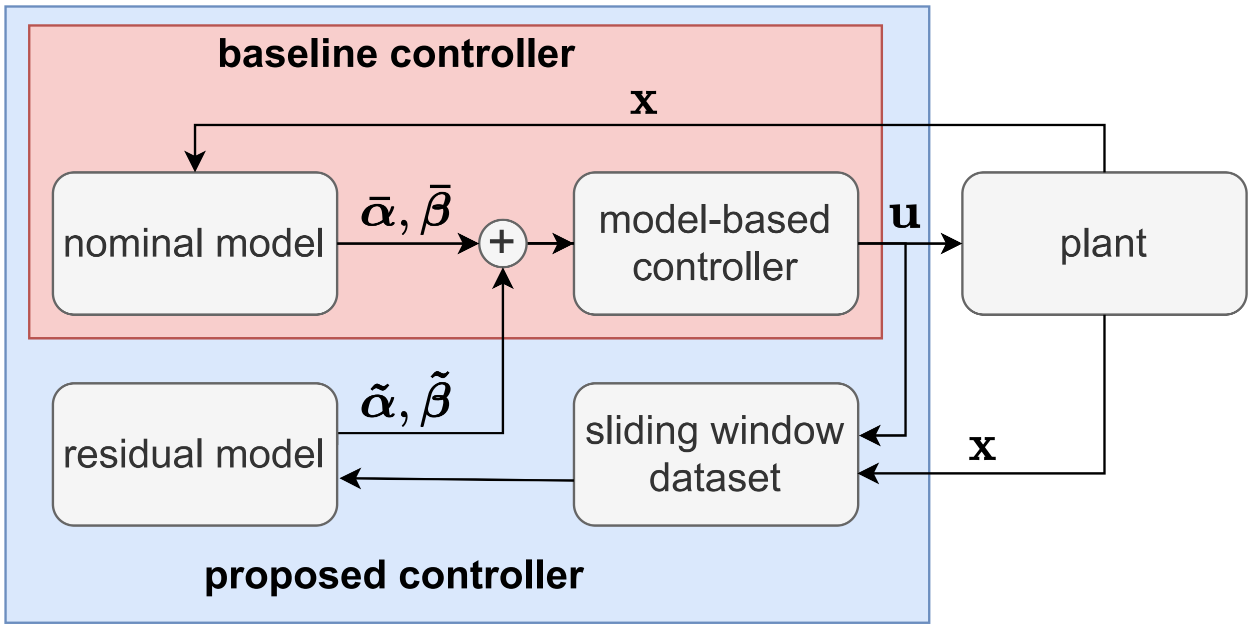


Figure 5.2: **Block diagram of our method.** $\bar{\boldsymbol{\alpha}}$ and $\bar{\boldsymbol{\beta}}$ are time-varying parameters of the nominal model for a system's output dynamics, assumed to be control-affine. As the model-based controller is running, data are collected into the sliding window dataset, and supervised learning is performed to estimate residual parameters $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$; they are then used to improve the model behind the controller. See Section 4.2 for more details.

known as control-affine ([127]):

$$\ddot{\mathbf{y}} = \bar{\alpha}(\mathbf{x})\mathbf{u} + \bar{\beta}(\mathbf{x}). \tag{5.1}$$

We consider model-based controllers whose goal is to drive the vector of tracking errors $\boldsymbol{\eta} = [\mathbf{y}; \dot{\mathbf{y}}]$ to zero.

The bars on top of the variables imply that they come from our assumed *nominal model*, which in reality can never be completely accurate. The unknown real-world dynamics are called the *true (plant) model*, denoted without the bars as α, β. We often use an alternative set of notations to write equation (5.1) simply as:

$$\ddot{\mathbf{y}} = \bar{\boldsymbol{\alpha}}\mathbf{u} + \bar{\boldsymbol{\beta}}, \tag{5.2}$$

in order to emphasize the role of $\bar{\boldsymbol{\alpha}}$ and $\bar{\boldsymbol{\beta}}$ as time-varying parameters of the output dynamics.

To make corrections to the nominal model, we incorporate two residual parameters and obtain the following form:

$$\ddot{\mathbf{y}} = (\bar{\boldsymbol{\alpha}} + \tilde{\boldsymbol{\alpha}})\,\mathbf{u} + \left(\bar{\boldsymbol{\beta}} + \tilde{\boldsymbol{\beta}}\right), \tag{5.3}$$

where $\tilde{\boldsymbol{\alpha}}$ is called the *weight* and $\tilde{\boldsymbol{\beta}}$ is called the *bias*. They are written as time-varying parameters, and have the same dimensions as $\bar{\boldsymbol{\alpha}}$ and $\bar{\boldsymbol{\beta}}$ respectively. The tildes on top of them emphasize that they are estimated from data.

To better understand these residual parameters, we manipulate equation (5.3) into:

$$\ddot{\mathbf{y}} - \left(\bar{\boldsymbol{\alpha}}\mathbf{u} + \bar{\boldsymbol{\beta}}\right) = \tilde{\boldsymbol{\alpha}}\mathbf{u} + \tilde{\boldsymbol{\beta}}. \tag{5.4}$$

Intuitively, the above equation says that the goal of learning is to make the *residual model* on the right-hand side account for the prediction errors of the nominal model on the left-hand side. It also reveals the role of labels vs. covariates, as we explain next in the context of learning.

Data Collection and Online Learning

For real systems, sensor data can only be collected at discrete sampling intervals. We denote each sampling timestep by an integer subscript, which converts equation (5.4) into:

$$\ddot{\mathbf{y}}_t - \left(\bar{\boldsymbol{\alpha}}_t\mathbf{u}_t + \bar{\boldsymbol{\beta}}_t\right) = \tilde{\boldsymbol{\alpha}}_t\mathbf{u}_t + \tilde{\boldsymbol{\beta}}_t. \tag{5.5}$$

Note that we are merely sampling a continuous system at discrete timesteps, so continuous-time concepts such as acceleration are still well defined. We collect a dataset of the form $\mathcal{D} := \{\text{label}_s, \text{covariate}_s\}_{s=t-k,\ldots,t-1}$, where s is the index of discrete timesteps, and k denotes the fixed size of the sliding time window. From equation (5.5), we have

$$\mathcal{D} = \left\{\ddot{\mathbf{y}}_s - \left(\bar{\boldsymbol{\alpha}}_s\mathbf{u}_s + \bar{\boldsymbol{\beta}}_s\right), \mathbf{u}_s\right\}_{s=t-k,\ldots,t-1}.$$

Given a dataset, our method solves regularized least squares a.k.a. ridge regression on the labels and covariates. The weight of the solution is $\tilde{\boldsymbol{\alpha}}_t$, and bias is $\tilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$. Note that in textbook-style least squares, the weight is a vector, and the label and bias are scalars; for our learning problem, the weight is a matrix in $\mathbb{R}^{d\times m}$, and the label and bias are vectors in $\mathbb{R}^d$. But we can simply reduce this to d independent vector-scalar least squares problems. The same regularization is added independently to these d problems, since they share the same covariates; thus inversion of the covariance matrix, the most computationally costly step, is only performed once.

The solved parameters are then immediately used by the model-based controller to produce $\mathbf{u}_t$. In both of our later examples, the baseline controller solves for $\mathbf{u}_t$ in an application-specific optimization problem with the assumed nominal parameters $\bar{\boldsymbol{\alpha}}_t$ and $\bar{\boldsymbol{\beta}}_t$. We simply substitute these with $\bar{\boldsymbol{\alpha}}_t + \tilde{\boldsymbol{\alpha}}_{\boldsymbol{t}}$ and $\bar{\boldsymbol{\beta}}_t + \tilde{\boldsymbol{\beta}}_{\boldsymbol{t}}$ respectively, as shown in Figure 5.2.

Learning is performed online, as the controller is running with the learned parameters. At the beginning, all residual parameters are initialized to zero, because there is not enough data to learn them. Once we are k steps into the trajectory, we have enough data to form $\mathcal{D}$ as above and solve for the residual parameters; informed by them, the controller generates an improved trajectory, which in turn generates new data that are more relevant as time goes on.

The fact that $\mathcal{D}$ only keeps the k most recent data points implements a natural forgetting mechanism. In reinforcement learning terms, $\mathcal{D}$ is called the *replay buffer*, which stores the off-policy data that are not generated by the current controller; in our case, data in $\mathcal{D}$ are generated by the old controllers using the residual parameters from previous timesteps. Because we learn small, local models, we *encourage forgetting* so that our model capacity can be used only for the neighborhood of our current state. This is in contrast to the vast literature in reinforcement learning [192, 128, 140, 109], where the goal is to learn a large, global model; there the replay buffer contains as much historical data as possible, and various techniques are implemented to *discourage forgetting.*

Our method can also be viewed as bootstrapping from a "bad" controller based on an inaccurate model to a better one. This might not be feasible, however, if the initial model deviates too much from the plant. For example, if the nominal model is so far off that the robot loses balance immediately, no useful information will be contained in the data collected. Fortunately, when deviations happen gradually over time, there will more likely be enough information for learning to maintain a controller that keeps generating useful data.

Theoretical Analysis

Suppose the true (plant) output dynamics is control-affine:

$$\ddot{\mathbf{y}}_t = \boldsymbol{\alpha}_t \mathbf{u}_t + \boldsymbol{\beta}_t. \tag{5.6}$$

We prove that our method stabilizes the tracking errors under two assumptions. The main theorem illustrates our intuition of learning in a local time window under smoothly varying dynamics, and characterizes the role of k, our window size.

Denote errors in the nominal model's prediction as

$$\hat{\ddot{\mathbf{y}}}_t := \ddot{\mathbf{y}}_t - \left(\bar{\boldsymbol{\alpha}}_t \mathbf{u}_t + \bar{\boldsymbol{\beta}}_t\right) = \hat{\boldsymbol{\alpha}}_t \mathbf{u}_t + \hat{\boldsymbol{\beta}}_t, \tag{5.7}$$

with $\hat{\boldsymbol{\alpha}}_t := \boldsymbol{\alpha}_t - \tilde{\boldsymbol{\alpha}}_t$, and $\hat{\boldsymbol{\beta}}_t := \boldsymbol{\beta}_t - \tilde{\boldsymbol{\beta}}_t$.

Denote the prediction of the residual model as

$$\tilde{\ddot{\mathbf{y}}}_t := \tilde{\boldsymbol{\alpha}}_t \mathbf{u}_t + \tilde{\boldsymbol{\beta}}_t. \tag{5.8}$$

Assumption 1. *The model-based controller can stabilize the tracking errors* $\boldsymbol{\eta} = [\mathbf{y}; \dot{\mathbf{y}}]$ *if for some* $\epsilon > 0$,

$$\left\| \ddot{\mathbf{y}}_t - \left(\left(\bar{\boldsymbol{\alpha}}_t + \tilde{\boldsymbol{\alpha}}_t\right) \mathbf{u}_t + \left(\bar{\boldsymbol{\beta}}_t + \tilde{\boldsymbol{\beta}}_t\right) \right) \right\| < \epsilon. \tag{5.9}$$

Assumption 2. $\|\hat{\boldsymbol{\alpha}}_{t+1} - \hat{\boldsymbol{\alpha}}_t\| < \delta_\alpha$, $\|\hat{\boldsymbol{\beta}}_{t+1} - \hat{\boldsymbol{\beta}}_t\| < \delta_\beta$.

In words, Assumption 1 says that the proposed model-based controller works when the proposed (nominal plus residual) model is relatively accurate; Assumption 2 says that the deviations in dynamics are relatively smooth (in the space of parameters) over time.

In addition, we denote the motor torque saturation as $\|\mathbf{u}\| < B$. Denote $\mathbf{u}'_t = [\mathbf{u}_t; 1] \in \mathbb{R}^{m+1}$, and

$$\mathbf{U}' = \left[[\mathbf{u}_1; 1]^\top; ...; [\mathbf{u}_k; 1]^\top \right] \in \mathbb{R}^{k\times(m+1)}. \tag{5.10}$$

We set $k \geq m+1$, so $\sigma_{\min}(\mathbf{U}') > 0$, i.e. the covariance matrix of ordinary least squares (OLS) has rank $m+1$.

Theorem 2. *Given the above assumptions, if*

$$\frac{(B+1)^2\sqrt{d}}{\sigma_{min}(\mathbf{U}')} k\sqrt{k}(\delta_\alpha + \delta_\beta) < \epsilon, \tag{5.11}$$

then the model-based controller stabilizes $\boldsymbol{\eta}$.

Note that any claim of stability in Theorem 1 is completely inherited from the baseline controller, when Assumption 1 holds. Our method is agnostic to the exact type of stability e.g. exponential / asymptotic, which depends on the underlying baseline, and is orthogonal to the theory we develop.

In Theorem 1, B, d, δ_α and δ_β are constants determined by the application. ϵ is the model-based controller's tolerance for model inaccuracy, also independent of our method. The only quantity we tune is k, the window size, which strongly affects $\sigma_{\min}(\mathbf{U}')$. With a large k, we pay a factor of $k\sqrt{k}$, intuitively due to the lag in our dataset. With a small k, we pay for the decrease in $\sigma_{\min}(\mathbf{U}')$, as $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\boldsymbol{\beta}}$ become more sensitive to noise. The user should tune k to find a sweet spot in the middle. In practice, we use regularized least squares instead of OLS, so $\sigma_{\min}(\mathbf{U}')$ is always > 0 and more noise tolerant, making the balance less delicate w.r.t. choice of k. We use $k = 100$ in both of our applications (100 and 200 ms respectively).

Before proving Theorem 1, we state two lemmas.

Lemma 1. *For* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and* $\mathbf{b} \in \mathbb{R}^m$, *if* $\|\mathbf{A}\| \leq \delta_A$ *and* $\|\mathbf{b}\| \leq \delta_b$, *then* $\|[\mathbf{A}, \mathbf{b}]\| \leq \delta_A + \delta_b$.

Lemma 2. *Let* $\mathbf{y}_t \in \mathbb{R}^d$, $\mathbf{u}_t \in \mathbb{R}^m$ *and* $\mathbf{A}_t \in \mathbb{R}^{d\times m}$. *Let* $\mathbf{y}_t = \mathbf{A}_t\mathbf{u}_t$ *for* $t = 1, ..., k$, *and* $\tilde{\mathbf{A}}$ *be the OLS estimator of the dataset* $\{(\mathbf{y}_1, \mathbf{u}_1), ..., (\mathbf{y}_k, \mathbf{u}_k)\}$. *If for* $t = 1, ..., k+1$, $\|\mathbf{A}_{t+1} - \mathbf{A}_t\| < \delta_A$, *and* $\|\mathbf{u}_t\| < B$, *then*

$$\|\mathbf{A}_t - \tilde{\mathbf{A}}_t\| < \frac{B\sqrt{d}}{\sigma_{min}(\mathbf{U})} k\sqrt{k}\delta_A, \tag{5.12}$$

where $\mathbf{U} = [\mathbf{u}_1^T; ...; \mathbf{u}_k^T] \in \mathbb{R}^{k\times m}$.

Proof of Theorem 1

By triangle inequality, we have $\|\mathbf{u}'_t\| < B+1$. Also define $\hat{\mathbf{A}}_t = [\hat{\boldsymbol{\alpha}}_t, \hat{\boldsymbol{\beta}}_t] \in \mathbb{R}^{d\times m+1}$, and similarly $\tilde{\mathbf{A}}_t = [\tilde{\boldsymbol{\alpha}}_t, \tilde{\boldsymbol{\beta}}_t]$. Combining Assumption 2 and Lemma 1, we have $\|\hat{\mathbf{A}}_{t+1} - \hat{\mathbf{A}}_t\| < \delta_\alpha + \delta_\beta$.

Now

$$\left\|\ddot{\mathbf{y}}_t - \left((\bar{\boldsymbol{\alpha}}_t + \tilde{\boldsymbol{\alpha}}_t)\,\mathbf{u}_t + \left(\bar{\boldsymbol{\beta}}_t + \tilde{\boldsymbol{\beta}}_t\right)\right)\right\| \tag{5.13}$$

$$= \left\|\hat{\ddot{\mathbf{y}}}_t - \tilde{\ddot{\mathbf{y}}}_t\right\| = \left\|(\hat{\boldsymbol{\alpha}}_t - \tilde{\boldsymbol{\alpha}}_t)\,\mathbf{u}_t + \left(\hat{\boldsymbol{\beta}}_t - \tilde{\boldsymbol{\beta}}_t\right)\right\| \tag{5.14}$$

$$= \left\|\left(\hat{\mathbf{A}}_t - \tilde{\mathbf{A}}_t\right)\mathbf{u}'_t\right\| \leq (B+1)\left\|\hat{\mathbf{A}}_t - \tilde{\mathbf{A}}_t\right\|. \tag{5.15}$$

By definition, $\tilde{\mathbf{A}}_t$ is the least squares solution on $\mathcal{D}$. We then apply Assumption 1 and Lemma 2 to finish the proof.

5.3 Results

Video of our experiments is available at `https://youtu.be/Je_2Y-FQpKw` ([1]). Simulations are performed in the PyBullet ([35]) physics engine.

Simulation for Bipedal Walking

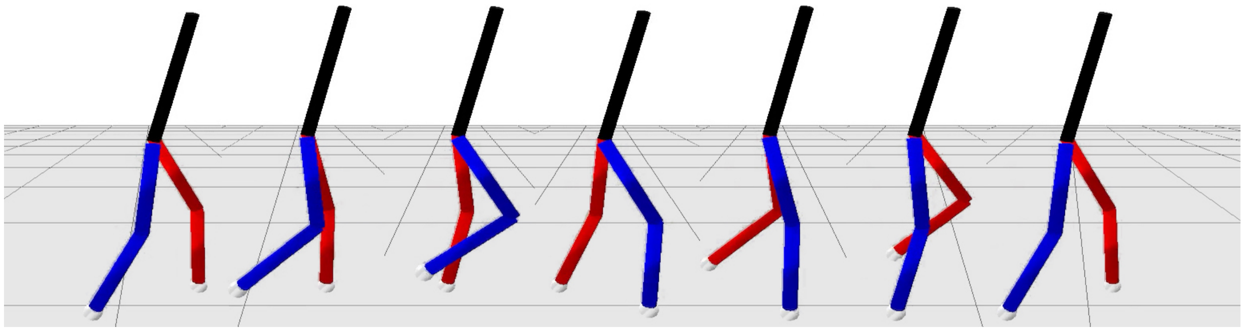
Our baseline controller is taken from [192], which introduces its own setting and method for unknown dynamics. We perform simulation in their setting, and make comparison with their method.

The problem setting is based on RABBIT ([31]), an under-actuated planar five-link bipedal robot with seven degree-of-freedom; virtual constraints and controller design are based on [4]. Model uncertainty is introduced in [192] by scaling the mass of each link by a factor of two in the real environment. The baseline CLF-QP controller falls in a few steps in this setting, due to the significant difference in dynamics between the nominal and true model.

By querying the plant, [192] uses model-free reinforcement learning (RL) to train a policy that directly adds on the original control inputs $\mathbf{u}$, without reasoning about the unknown dynamics in the model space. Specifically, the commanded control inputs take the form $\mathbf{u} + u_{\boldsymbol{\theta}}(\mathbf{x})$, where u is a neural network policy with parameters $\boldsymbol{\theta}$. Their reward is designed to encourage $\dot{V} < -cV$, where the value of V is obtained by simulating in the plant. After 20,000 samples from the plant simulated using the true dynamics, their method trains a policy which walks in the true dynamics without falling.

Our method walks stably in the same setting, training completely online without querying the plant at all before deployment. In fact, Fig. 5.3 shows that our method enjoys smaller impulses of control inputs and better tracking performance than the RL-based method, even though the latter had privileged access to the plant before deployment to optimize exactly for these metrics.

Online learning enables us to treat the plant as truly unknown, in terms of both data and mathematical representation, while only the latter is unknown for methods that train



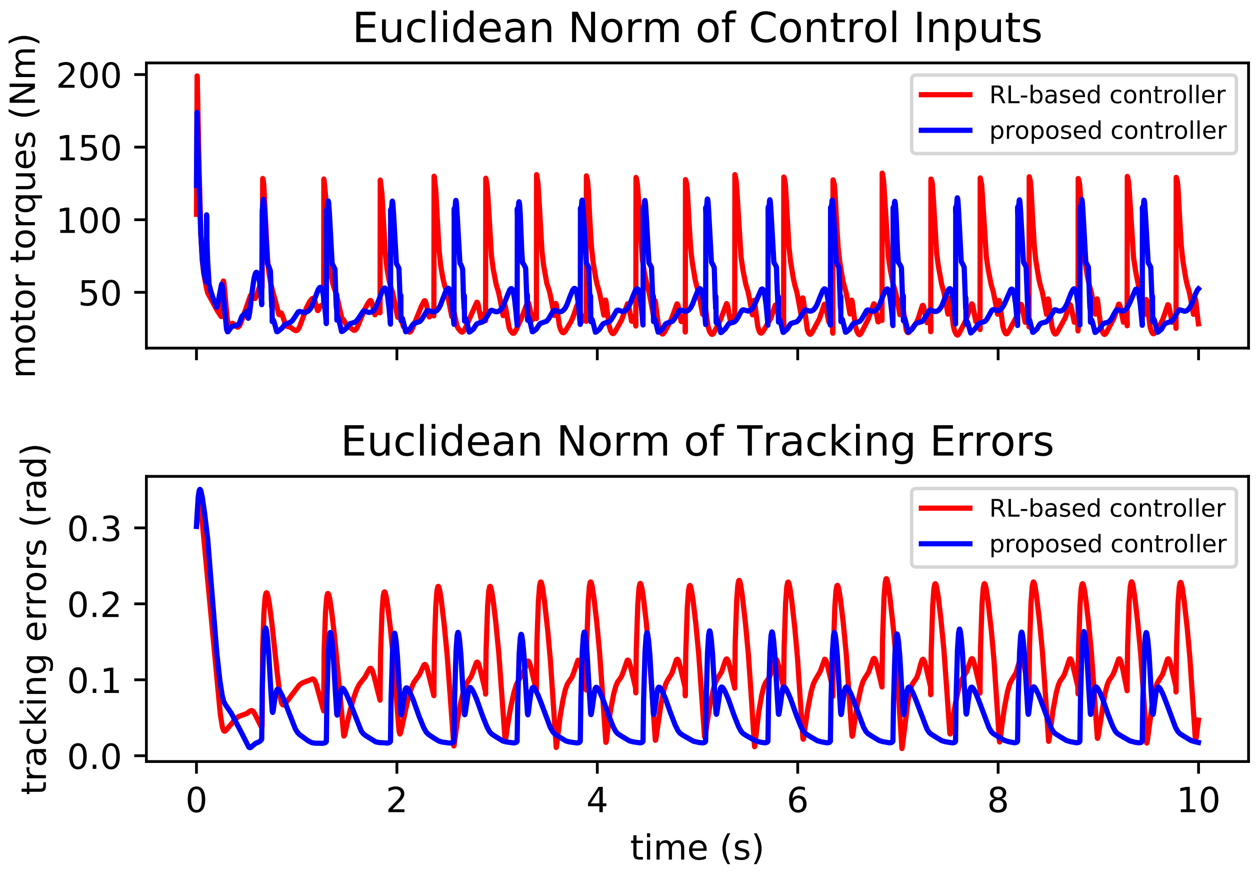


Figure 5.3: **Bipedal walking with mass of each link scaled by two.** Both our method and that of [192] walk stably. Their RL-based method trains on 20,000 samples from the real environment before deployment. Our method trains completely online and does not sample from or anticipate the real environment, treating it as truly unknown until the robot is deployed, and results in smaller impulses of control inputs and better tracking performance. The top panel visualizes the gait generated by our method.

offline like in [192]. This philosophical difference prevents our controller from overfitting on the training environment. In particular, our controller still walks stably under the original dynamics without scaling, where the policy trained with the scaled links fails, because it overfits to the scaled dynamics.

In addition, our controller walks stably in all environments below, where the baseline and the RL-based method cannot:

1. scaling the control inputs by half, in order to simulate transmission inefficiencies and motor wear and tear;

2. scaling the mass of the torso by four, in order to simulate payload on the back of the humanoid;

3. scaling the mass of the right leg by four, as an example of asymmetric changes in dynamics.

We keep the same hyper-parameters for all the experiments above, including a windows size of 100 ms (where $k = 100$ and each timestep is 1 ms). The robot is still able to walk under the scaled dynamics with a window size of 10 or 1000, but has higher norm of control inputs and tracking errors.

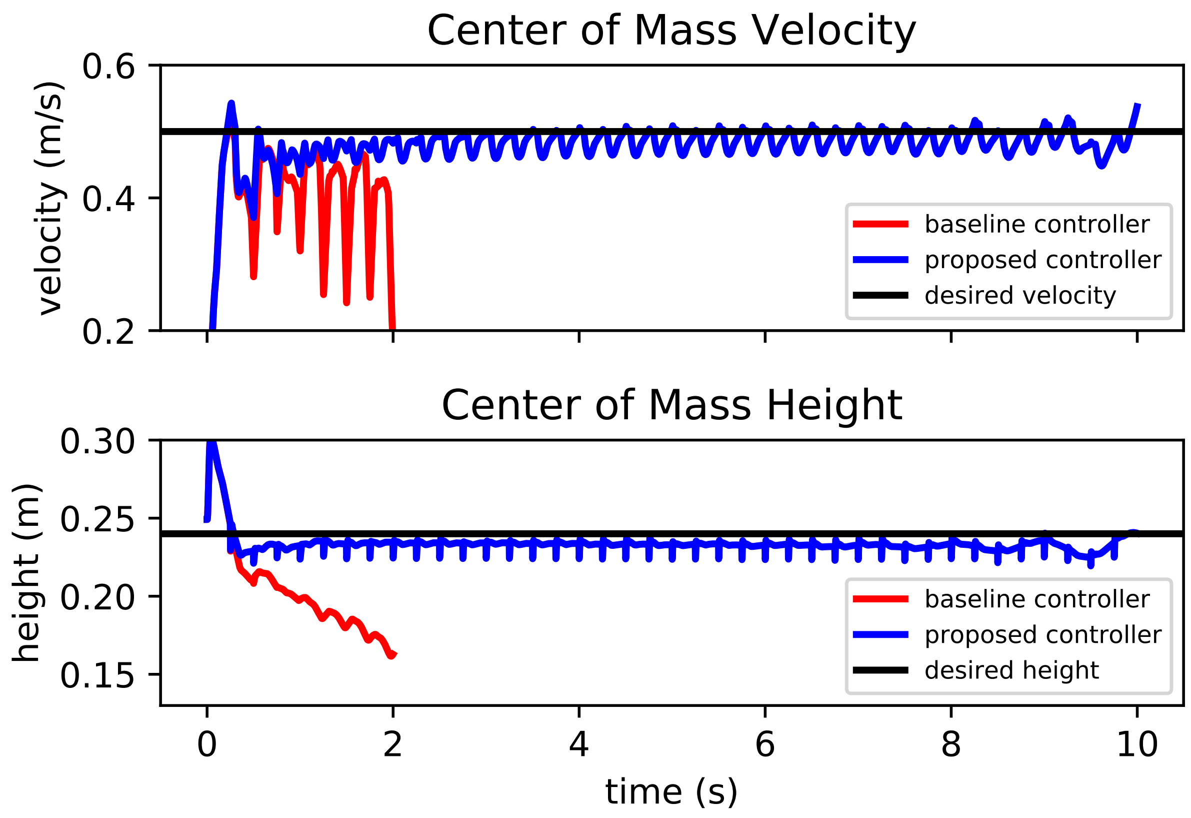
Simulation for Quadrupedal Robot

Our baseline controller is based on [41] and used subsequently in [39] and [138]. Our implementation is modified from the publicly available code of [138] on an Unitree A1 quadruped, and keeps their original parameters unless stated otherwise. The A1 weighs 12 kg and has 12 motors, three for each leg, with the stated torque limit of 35.5 Nm. We experiment in PyBullet using Unitree's URDF description, and also on a real robot. In both simulation and real world, we use a window size of $k = 100$ (like for the biped); the controller runs at a frequency of 500 Hz, making the dataset window 200 ms.

We command our robot to walk with linear velocity of 0.5 m / s in the x-direction, while maintaining CoM height of 0.24 m. Both the baseline and the proposed method can walk stably without payload, while tracking the desired velocity and height. With 6 kg of payload, however, the baseline can barely walk at 2 / 3 the desired velocity, and sags to 2 / 3 the desired height; the robot falls with 7 kg.

The proposed method walks stably with 12 kg of payload (same as its body mass), while tracking the desired velocity up to 0.05 m / s, and the desired height up to 0.01 m; all motors torques are less than 35.5 Nm. With more than 12 kg, however, tracking becomes less accurate, and with 15 kg the robot falls. Since the payload is carried from very the beginning of simulation, the robot visibly sags for the first fifth of a second, as we collect data before we can estimate the residual parameters. With 12 kg it soon recovers from the sag, but for larger payloads it struggles to get back.

Next, we experiment with gradually changing dynamics. We start with an empty payload, and increase its mass by 5 kg / s, that is, 0.001 kg per timestep, once simulation begins. The



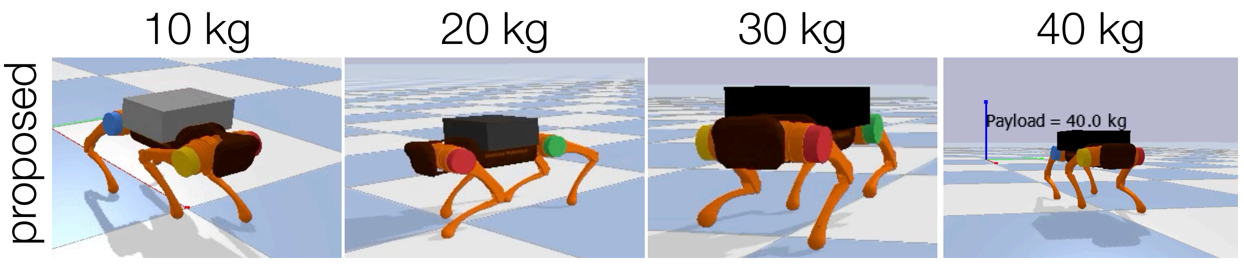


Figure 5.4: **Quadruped walking with payload in simulation.** We start with an empty payload, and increase its mass by 5 kg / s once simulation begins. The baseline has completely fallen in 2 s, but the proposed method still walks stably after 10 s (50 kg). The bottom visualizations are captured when the payload reaches the specified mass. The torque limit is reached at 25 kg.

tracking errors are shown in Figure 5.4. The baseline falls within 2 s. We have tried to improve the baseline by tuning the PD gains for $\ddot{\mathbf{p}}_d$, but found it ineffective. This observation is reasonable, since larger gains only make $\ddot{\mathbf{p}}_d$ more aggressive, but cannot help if the model-based controller fails to achieve it using the nominal dynamics. The proposed method walks

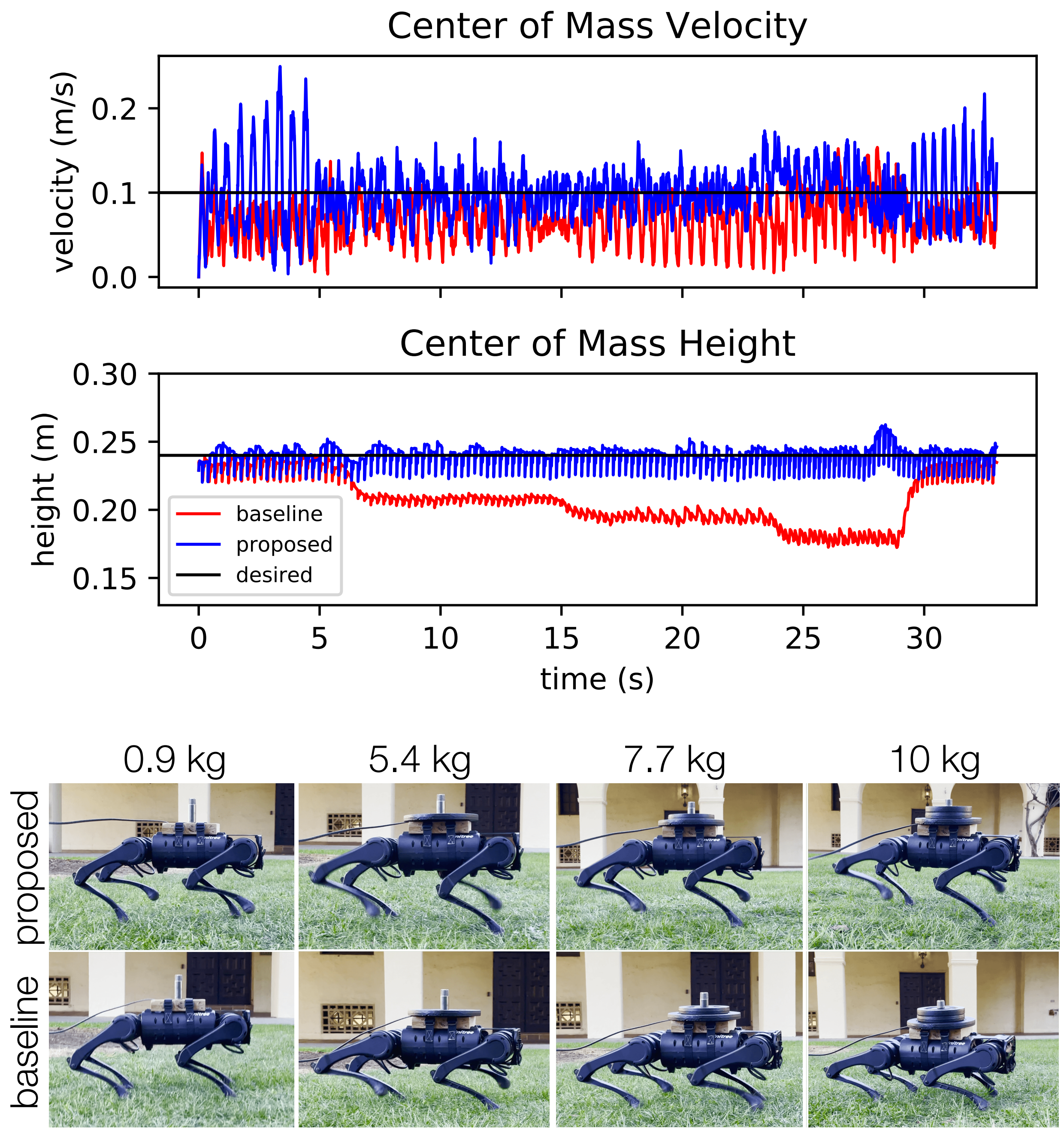


Figure 5.5: **Quadruped walking with payload in the real world.**

stably even when the payload reaches 50 kg. Motor torques reach the specified limit at 25 kg (5 s), but the URDF allows simulation to keep running.

Hardware Experiments for Quadrupedal Robot

To facilitate hardware testing, we fit the Unitree A1 quadruped with a loading rig designed to hold up three standard 1 inch weight plates. The rig allows for incremental, discrete changes in load while the quadruped is in operation. The rig itself weighs 0.9 kg.

The experiments were designed to compare the performance of the baseline and proposed controllers under varying load conditions during operation. Two tests for each controller were performed: a step-in-place test and a 0.1 m/s forward motion test. The load conditions for the tests are shown in Table 1. Due to the manual loading process, the duration of each load varies by a small amount of transition time, typically less than 1 s. To protect the hardware from possible damage, we do not load beyond 10 kg, and limit operation at this load to 5 s.

Table 5.1: Load Conditions for Hardware Experiments

Load (kg)	0.9	5.4	7.7	10	0.9
Duration (s)	5	10	10	5	5

In the transition from simulation to hardware, we had to address the problem of acceleration estimation from noisy measurements. [138] uses a Kalman filter to fuse IMU and joint encoder measurements and produce a CoM velocity measurement. From this, first order difference is then used to compute a CoM acceleration estimate for the learning algorithm. Two parameters for the Kalman filter, namely the window size and IMU variance value, are tuned to give a final acceleration estimate with suitable trade-off between lag and noise. The window size is modified from 120 to 60 samples, and the IMU variance is modified from equal to the encoder variance to 5 times the encoder variance. Ultimately, after tuning, the estimator produces acceptable linear acceleration estimates, but the angular terms proved too noisy to be useful. As such, we proceeded with hardware experiments with learning enabled for only the linear terms.

The hardware experiments were performed on the A1 on flat, grassy terrain. Both the baseline and proposed methods perform nominally with low load, but as the weight increases, the baseline controller sags in body height and is unable to maintain forward velocity. The proposed controller does not suffer this degradation and is able to maintain desired body height and forward velocity for the range of load conditions. Results for the forward walking test are summarized in Figure 5.5. Video comparison of both trot-in-place and forward walking is available in [1].

Bibliography

[1] Supplementary video. `https://youtu.be/Je_2Y-FQpKw`.

[2] Pulkit Agrawal et al. “Learning to poke by poking: Experiential learning of intuitive physics”. In: *NIPS* (2016).

[3] Rahaf Aljundi, Klaas Kelchtermans, and Tinne Tuytelaars. “Task-free continual learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11254–11263.

[4] Aaron D Ames et al. “Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics”. In: *IEEE Transactions on Automatic Control* 59.4 (2014), pp. 876–891.

[5] Brandon Amos et al. “Differentiable mpc for end-to-end planning and control”. In: *arXiv preprint arXiv:1810.13400* (2018).

[6] V.I. Arnold. *Mathematical Methods of Classical Mechanics*. 2nd ed. 60. Springer-Verlag New York, 1989.

[7] Yuki Asano et al. “Labelling unlabelled videos from scratch with multi-modal self-supervision”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4660–4671.

[8] Yuki M Asano, Christian Rupprecht, and Andrea Vedaldi. “Surprising Effectiveness of Few-Image Unsupervised Feature Learning”. In: *arXiv preprint arXiv:1904.13132* (2019).

[9] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. “Self-labelling via simultaneous clustering and representation learning”. In: *arXiv preprint arXiv:1911.05371* (2019).

[10] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[11] Karl Johan Åström and Peter Eykhoff. “System identification—a survey”. In: *Automatica* 7.2 (1971), pp. 123–162.

[12] Anish Athalye, Nicholas Carlini, and David Wagner. “Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples”. In: *arXiv preprint arXiv:1802.00420* (2018).

[13] Ko Ayusawa, Gentiane Venture, and Yoshihiko Nakamura. “Identification of humanoid robots dynamics using floating-base motion dynamics”. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 2854–2859.

[14] Fatemeh Azimi et al. “Self-supervised test-time adaptation on video data”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2022, pp. 3439–3448.

[15] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. “Metareg: Towards domain generalization using meta-regularization”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 998–1008.

[16] Pratyay Banerjee, Tejas Gokhale, and Chitta Baral. “Self-supervised test-time learning for reading comprehension”. In: *arXiv preprint arXiv:2103.11263* (2021).

[17] Hangbo Bao, Li Dong, and Furu Wei. “BEiT: BERT Pre-Training of Image Transformers”. In: *CoRR* abs/2106.08254 (2021). arXiv: `2106.08254`. URL: `https://arxiv.org/abs/2106.08254`.

[18] Peter W Battaglia et al. “Interaction networks for learning about objects, relations and physics”. In: *NIPS* (2016).

[19] David Bau et al. “Semantic photo manipulation with a generative image prior”. In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), p. 59.

[20] Harry Berghuis, Herman Roebbers, and Henk Nijmeijer. “Experimental comparison of parameter estimation methods in adaptive robot control”. In: *Automatica* 31.9 (1995), pp. 1275–1285.

[21] Piotr Bojanowski and Armand Joulin. “Unsupervised learning by predicting noise”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 517–526.

[22] Léon Bottou and Yann LeCun. “Large scale online learning”. In: *Advances in neural information processing systems* 16 (2004), pp. 217–224.

[23] Léon Bottou and Vladimir Vapnik. “Local learning algorithms”. In: *Neural computation* 4.6 (1992), pp. 888–900.

[24] Fabio M Carlucci et al. “Domain Generalization by Solving Jigsaw Puzzles”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2229–2238.

[25] Mathilde Caron et al. “Deep clustering for unsupervised learning of visual features”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 132–149.

[26] Rich Caruana. “Multitask learning”. In: *Machine learning* 28.1 (1997), pp. 41–75.

[27] Liang-Chieh Chen et al. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017).

[28] Minmin Chen, Kilian Q Weinberger, and John Blitzer. “Co-training for domain adaptation”. In: *Advances in neural information processing systems*. 2011, pp. 2456–2464.

[29] Xilun Chen et al. “Adversarial deep averaging networks for cross-lingual sentiment classification”. In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 557–570.

[30] Bowen Cheng et al. “Masked-attention mask transformer for universal image segmentation”. In: *arXiv preprint arXiv:2112.01527* (2021).

[31] Christine Chevallereau et al. “Rabbit: A testbed for advanced control theory”. In: *IEEE Control Systems Magazine* 23.5 (2003), pp. 57–79.

[32] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *arXiv preprint arXiv:1902.02918* (2019).

[33] Ronan Collobert et al. “Large scale transductive SVMs.” In: *Journal of Machine Learning Research* 7.8 (2006).

[34] Marius Cordts et al. “The cityscapes dataset for semantic urban scene understanding”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[35] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2019.

[36] Francesco Croce, Maksym Andriushchenko, and Matthias Hein. “Provable robustness of relu networks via maximization of linear regions”. In: *arXiv preprint arXiv:1810.07481* (2018).

[37] Gabriela Csurka. “Domain adaptation for visual applications: A comprehensive survey”. In: *arXiv preprint arXiv:1702.05374* (2017).

[38] Ekin D. Cubuk et al. “RandAugment: Practical data augmentation with no separate search”. In: *CoRR* abs/1909.13719 (2019). arXiv: 1909.13719. URL: http://arxiv.org/abs/1909.13719.

[39] Xingye Da et al. “Learning a contact-adaptive controller for robust, efficient legged locomotion”. In: *4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.* (2020).

[40] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[41] Jared Di Carlo et al. “Dynamic locomotion in the mit cheetah 3 through convex model-predictive control”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–9.

[42] Natalia Diaz-Rodriguez et al. “Don’t forget, there is more than forgetting: new metrics for Continual Learning”. In: *arXiv preprint arXiv:1810.13166* (2018).

[43] Samuel Dodge and Lina Karam. “Quality resilient deep neural networks”. In: *arXiv preprint arXiv:1703.08119* (2017).

[44] Carl Doersch, Abhinav Gupta, and Alexei A Efros. “Unsupervised visual representation learning by context prediction”. In: *Proceedings of the IEEE International Conference on Computer Vision.* 2015, pp. 1422–1430.

[45] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[46] Li Fei-Fei, Rob Fergus, and Pietro Perona. “One-shot learning of object categories”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611.

[47] Enrico Fini et al. “Self-supervised models are continual learners”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2022, pp. 9621–9630.

[48] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70.* JMLR. org. 2017, pp. 1126–1135.

[49] Yang Fu et al. “Learning to track instances without video annotations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2021, pp. 8680–8689.

[50] A. Gammerman, V. Vovk, and V. Vapnik. “Learning by Transduction”. In: *In Uncertainty in Artificial Intelligence.* Morgan Kaufmann, 1998, pp. 148–155.

[51] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. “Learning by transduction”. In: *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence.* Morgan Kaufmann Publishers Inc. 1998, pp. 148–155.

[52] Chuang Gan, Tianbao Yang, and Boqing Gong. “Learning attributes equals multi-source domain generalization”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 87–97.

[53] Yossi Gandelsman et al. “Test-Time Training with Masked Autoencoders”. In: *Advances in Neural Information Processing Systems* (2022).

[54] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 2096–2030.

[55] Robert Geirhos et al. “Generalisation in humans and deep neural networks”. In: *Advances in Neural Information Processing Systems.* 2018, pp. 7538–7550.

[56] Muhammad Ghifary et al. “Domain generalization for object recognition with multi-task autoencoders”. In: *Proceedings of the IEEE international conference on computer vision.* 2015, pp. 2551–2559.

[57] Spyros Gidaris and Nikos Komodakis. "Dynamic few-shot visual learning without forgetting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4367–4375.

[58] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised representation learning by predicting image rotations". In: *arXiv preprint arXiv:1803.07728* (2018).

[59] Shiry Ginosar et al. "A Century of Portraits: A Visual Historical Record of American High School Yearbooks". In: *CoRR* abs/1511.02575 (2015). arXiv: 1511.02575. URL: http://arxiv.org/abs/1511.02575.

[60] Boqing Gong et al. "Geodesic flow kernel for unsupervised domain adaptation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 2066–2073.

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

[62] Lauren Gravitz. "The importance of forgetting". In: *Nature* 571.July (2019), S12–S14.

[63] Chuan Guo et al. "Countering adversarial images using input transformations". In: *arXiv preprint arXiv:1711.00117* (2017).

[64] Raia Hadsell et al. "Embracing change: Continual learning in deep neural networks". In: *Trends in cognitive sciences* 24.12 (2020), pp. 1028–1040.

[65] Nicklas Hansen et al. "Self-Supervised Policy Adaptation during Deployment". In: (ICLR, 2021).

[66] Nicklas Hansen et al. "Self-supervised policy adaptation during deployment". In: *arXiv preprint arXiv:2007.04309* (2020).

[67] Elad Hazan et al. "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

[68] Elad Hazan. "Introduction to online convex optimization". In: *Foundations and Trends in Optimization: Vol. 2: No. 3-4, pp 157-325* (2019).

[69] Kaiming He, Ross Girshick, and Piotr Dollár. "Rethinking imagenet pre-training". In: *arXiv preprint arXiv:1811.08883* (2018).

[70] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[71] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[72] Kaiming He et al. "Masked Autoencoders Are Scalable Vision Learners". In: *CoRR* abs/2111.06377 (2021). arXiv: 2111.06377. URL: https://arxiv.org/abs/2111.06377.

[73] Dan Hendrycks and Thomas Dietterich. "Benchmarking neural network robustness to common corruptions and perturbations". In: *arXiv preprint arXiv:1903.12261* (2019).

[74] Dan Hendrycks and Thomas G. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *CoRR* abs/1903.12261 (2019). arXiv: `1903.12261`. URL: `http://arxiv.org/abs/1903.12261`.

[75] Dan Hendrycks and Kevin Gimpel. “A baseline for detecting misclassified and out-of-distribution examples in neural networks”. In: *arXiv preprint arXiv:1610.02136* (2016).

[76] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. “Using pre-training can improve model robustness and uncertainty”. In: *arXiv preprint arXiv:1901.09960* (2019).

[77] Dan Hendrycks et al. “Improving Model Robustness and Uncertainty Estimates with Self-Supervised Learning”. In: *arXiv preprint* (2019).

[78] Dan Hendrycks et al. “Natural Adversarial Examples”. In: *CVPR* (2021).

[79] Dan Hendrycks et al. “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization”. In: *ICCV* (2021).

[80] Dan Hendrycks et al. “Using trusted data to train deep networks on labels corrupted by severe noise”. In: *Advances in neural information processing systems*. 2018, pp. 10456–10465.

[81] Judy Hoffman, Trevor Darrell, and Kate Saenko. “Continuous manifold based adaptation for evolving visual domains”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 867–874.

[82] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. “Algorithms and theory for multiple-source adaptation”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 8246–8256.

[83] Judy Hoffman et al. “Cycada: Cycle-consistent adversarial domain adaptation”. In: *arXiv preprint arXiv:1711.03213* (2017).

[84] Judy Hoffman et al. “Discovering latent domains for multisource domain adaptation”. In: *European Conference on Computer Vision*. Springer. 2012, pp. 702–715.

[85] Naira Hovakimyan and Chengyu Cao. *L1 adaptive control theory: Guaranteed robustness with fast adaptation*. SIAM, 2010.

[86] Gao Huang et al. “Deep networks with stochastic depth”. In: *European conference on computer vision*. Springer. 2016, pp. 646–661.

[87] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[88] Vidit Jain and Erik Learned-Miller. “Online domain adaptation of a pre-trained cascade of classifiers”. In: *CVPR 2011*. IEEE. 2011, pp. 577–584.

[89] Michael Janner et al. “Reasoning about physical interactions with object-oriented prediction and planning”. In: *arXiv preprint arXiv:1812.10972* (2018).

[90] S. Jin, C. Wang, and M. Tomizuka. "Robust Deformation Model Approximation for Robotic Cable Manipulation". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 6586–6593. DOI: 10.1109/IROS40897.2019.8968157.

[91] Thorsten Joachims. *Learning to classify text using support vector machines*. Vol. 668. Springer Science & Business Media, 2002.

[92] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection". In: *IEEE transactions on pattern analysis and machine intelligence* 34.7 (2011), pp. 1409–1422.

[93] Daniel Kang et al. "Transfer of Adversarial Robustness Between Perturbation Types". In: *arXiv preprint arXiv:1905.01034* (2019).

[94] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. "Adversarial logit pairing". In: *arXiv preprint arXiv:1803.06373* (2018).

[95] Neerav Karani et al. "Test-time adaptable neural networks for robust medical image segmentation". In: *Medical Image Analysis* 68 (2021), p. 101907.

[96] Dahun Kim et al. "Video panoptic segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9859–9868.

[97] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[98] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. Citeseer, 2009.

[99] Zhian Kuang et al. "Feedback-based Digital Higher-order Terminal Sliding Mode for 6-DOF Industrial Manipulators". In: *American Control Conference* (2021).

[100] Ananya Kumar, Tengyu Ma, and Percy Liang. "Understanding self-training for gradual domain adaptation". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5468–5479.

[101] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Colorization as a Proxy Task for Visual Understanding". In: *CVPR*. 2017.

[102] Chenyang Lei and Qifeng Chen. "Fully automatic video colorization with self-regularization and diversity". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3753–3761.

[103] Da Li and Timothy Hospedales. "Online meta-learning for multi-source and semi-supervised domain adaptation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 382–403.

[104] Da Li et al. "Deeper, broader and artier domain generalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5542–5550.

[105] Da Li et al. "Episodic training for domain generalization". In: *arXiv* (2019).

[106] Ya Li et al. "Deep domain generalization via conditional invariant adversarial networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 624–639.

[107] Yizhuo Li et al. "Test-time personalization with a transformer for human pose estimation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 2583–2597.

[108] Zhenguo Li et al. "Meta-SGD: Learning to learn quickly for few-shot learning". In: *arXiv preprint arXiv:1707.09835* (2017).

[109] Zhizhong Li and Derek Hoiem. "Learning without forgetting". In: *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017), pp. 2935–2947.

[110] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2014. URL: `http://arxiv.org/abs/1405.0312`.

[111] Xiaofeng Liu et al. "Energy-constrained self-training for unsupervised domain adaptation". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 7515–7520.

[112] Yuejiang Liu et al. "TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive?" In: *Advances in Neural Information Processing Systems* 34 (2021).

[113] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[114] Zhuang Liu et al. "Rethinking the value of network pruning". In: *arXiv preprint arXiv:1810.05270* (2018).

[115] Mingsheng Long et al. "Learning transferable features with deep adaptation networks". In: *arXiv preprint arXiv:1502.02791* (2015).

[116] Mingsheng Long et al. "Unsupervised domain adaptation with residual transfer networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 136–144.

[117] David Lopez-Paz and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6467–6476.

[118] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[119] Xuan Luo et al. "Consistent video depth estimation". In: *ACM Transactions on Graphics (ToG)* 39.4 (2020), pp. 71–1.

[120] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083* (2017).

[121] Chengzhi Mao et al. "Discrete representations strengthen vision transformer robustness". In: *arXiv preprint arXiv:2111.10493* (2021).

[122] Ke Mei et al. "Instance adaptive self-training for unsupervised domain adaptation". In: *European conference on computer vision*. Springer. 2020, pp. 415–430.

[123] Michael Mistry, Stefan Schaal, and Katsu Yamane. "Inertial parameter estimation of floating base humanoid systems using partial force sensing". In: *2009 9th IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2009, pp. 492–497.

[124] Saeid Motiian et al. "Unified deep supervised domain adaptation and generalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5715–5725.

[125] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. "Domain generalization via invariant feature representation". In: *International Conference on Machine Learning*. 2013, pp. 10–18.

[126] Ravi Teja Mullapudi et al. "Online Model Distillation for Efficient Video Inference". In: *arXiv preprint arXiv:1812.02699* (2018).

[127] Richard M Murray et al. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[128] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. "Deep online learning via meta-learning: Continual adaptation for model-based RL". In: *ICLR 2019* (2019).

[129] Anusha Nagabandi et al. "Deep dynamics models for learning dexterous manipulation". In: *Conference on Robot Learning*. PMLR. 2020, pp. 1101–1112.

[130] Quan Nguyen and Koushil Sreenath. "L1 adaptive control for bipedal robots with control Lyapunov function based quadratic programs". In: *2015 American Control Conference (ACC)*. IEEE. 2015, pp. 862–867.

[131] Yotam Nitzan et al. "MyStyle: A Personalized Generative Prior". In: *arXiv preprint arXiv:2203.17272* (2022).

[132] Mehdi Noroozi and Paolo Favaro. "Unsupervised learning of visual representations by solving jigsaw puzzles". In: *European Conference on Computer Vision*. Springer. 2016, pp. 69–84.

[133] Romeo Ortega and Mark W Spong. "Adaptive motion control of rigid robots: A tutorial". In: *Automatica* 25.6 (1989), pp. 877–888.

[134] Theodoros Panagiotakopoulos et al. "Online Domain Adaptation for Semantic Segmentation in Ever-Changing Conditions". In: *arXiv preprint arXiv:2207.10667* (2022).

[135] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[136] Deepak Pathak et al. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2536–2544.

[137] Deepak Pathak et al. "Curiosity-driven Exploration by Self-supervised Prediction". In: *ICML*. 2017.

[138] Xue Bin Peng et al. "Learning Agile Robotic Locomotion Skills by Imitating Animals". In: *Robotics: Science and Systems*. July 2020. DOI: 10.15607/RSS.2020.XVI.064.

[139] Jordi Pont-Tuset et al. "The 2017 DAVIS Challenge on Video Object Segmentation". In: *arXiv:1704.00675* (2017).

[140] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. "Off-policy temporal-difference learning with function approximation". In: *ICML*. 2001, pp. 417–424.

[141] Senthil Purushwalkam, Pedro Morgado, and Abhinav Gupta. "The Challenges of Continuous Self-Supervised Learning". In: *arXiv preprint arXiv:2203.12710* (2022).

[142] Haozhi Qi et al. "Learning Long-term Visual Dynamics with Region Proposal Interaction Networks". In: *ICLR* (2021).

[143] Ilija Radosavovic et al. "Data Distillation: Towards Omni-Supervised Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[144] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples". In: *arXiv preprint arXiv:1801.09344* (2018).

[145] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *IEEE transactions on pattern analysis and machine intelligence* (2016).

[146] Benjamin Recht et al. "Do CIFAR-10 Classifiers Generalize to CIFAR-10?" In: *arXiv preprint arXiv:1806.00451* (2018).

[147] Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. "Semi-supervised self-training of object detection models". In: *arXiv* (2005).

[148] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[149] Hadi Salman et al. "Provably Robust Deep Learning via Adversarially Trained Smoothed Classifiers". In: *arXiv preprint arXiv:1906.04584* (2019).

[150] Adam Santoro et al. "Meta-learning with memory-augmented neural networks". In: *International conference on machine learning*. 2016, pp. 1842–1850.

[151] Steffen Schneider et al. "Improving robustness against common corruptions by covariate shift adaptation". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11539–11551.

[152] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and trends in Machine Learning* 4.2 (2011), pp. 107–194.

[153] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". In: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[154] Shiv Shankar et al. "Generalizing across domains via cross-gradient training". In: *arXiv preprint arXiv:1804.10745* (2018).

[155] Vaishaal Shankar et al. "Do Image Classifiers Generalize Across Time?" In: *arXiv* (2019).

[156] Vaishaal Shankar et al. "Do image classifiers generalize across time?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9661–9669.

[157] Hanul Shin et al. "Continual learning with deep generative replay". In: *Advances in neural information processing systems* 30 (2017).

[158] Assaf Shocher, Nadav Cohen, and Michal Irani. ""Zero-Shot" Super-Resolution using Deep Internal Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3118–3126.

[159] Abhinav Shrivastava et al. "Data-driven Visual Similarity for Cross-domain Image Matching". In: *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)* 30.6 (2011).

[160] Aman Sinha, Hongseok Namkoong, and John Duchi. "Certifying some distributional robustness with principled adversarial training". In: *arXiv preprint arXiv:1710.10571* (2017).

[161] Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*. Vol. 199. 1. Prentice hall Englewood Cliffs, NJ, 1991.

[162] Jean-Jacques E Slotine and Weiping Li. "On the adaptive control of robot manipulators". In: *The international journal of robotics research* 6.3 (1987), pp. 49–59.

[163] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4077–4087.

[164] Kihyuk Sohn et al. "A simple semi-supervised learning framework for object detection". In: *arXiv preprint arXiv:2005.04757* (2020).

[165] Yang Song et al. "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples". In: *arXiv preprint arXiv:1710.10766* (2017).

[166] Teo Spadotto et al. "Unsupervised domain adaptation with multiple domain discriminators and adaptive self-training". In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, pp. 2845–2852.

[167] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. "Boosting Supervision with Self-Supervision for Few-shot Learning". In: *arXiv preprint arXiv:1906.07079* (2019).

[168] Yu Sun et al. “Online learning of unknown dynamics for model-based controllers in legged locomotion”. In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 8442–8449.

[169] Yu Sun et al. “Test-time training for out-of-distribution generalization”. In: (2019).

[170] Yu Sun et al. “Test-Time Training with Self-Supervision for Generalization under Distribution Shifts”. In: *ICML*. 2020.

[171] Yu Sun et al. “Test-time training with self-supervision for generalization under distribution shifts”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9229–9248.

[172] Yu Sun et al. “Unsupervised Domain Adaptation through Self-Supervision”. In: *arXiv preprint* (2019).

[173] Yu Sun et al. “Unsupervised domain adaptation through self-supervision”. In: *arXiv preprint arXiv:1909.11825* (2019).

[174] Alessio Tonioni et al. “Learning to adapt for stereo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9661–9670.

[175] Alessio Tonioni et al. “Real-time self-adaptive deep stereo”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 195–204.

[176] Antonio Torralba and Alexei A Efros. “Unbiased look at dataset bias”. In: *CVPR 2011*. IEEE. 2011, pp. 1521–1528.

[177] Guido Tournois et al. “Online payload identification for quadruped robots”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 4889–4896.

[178] Nilesh Tripuraneni and Lester Mackey. “Debiasing Linear Prediction”. In: *arXiv preprint arXiv:1908.02341* (2019).

[179] Dimitris Tsipras et al. “Robustness may be at odds with accuracy”. In: *arXiv preprint arXiv:1805.12152* (2018).

[180] Eric Tzeng et al. “Adversarial discriminative domain adaptation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7167–7176.

[181] Jack Valmadre et al. “Long-term tracking in the wild: A benchmark”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 670–685.

[182] Gido M Van de Ven and Andreas S Tolias. “Three scenarios for continual learning”. In: *arXiv preprint arXiv:1904.07734* (2019).

[183] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[184] Vladimir Vapnik and S. Kotz. *Estimation of Dependences Based on Empirical Data: Empirical Inference Science (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387308652.

[185] Igor Vasiljevic, Ayan Chakrabarti, and Gregory Shakhnarovich. "Examining the impact of blur on recognition by convolutional networks". In: *arXiv preprint arXiv:1611.05760* (2016).

[186] Pascal Vincent et al. "Extracting and Composing Robust Features with Denoising Autoencoders". In: ICML '08. Helsinki, Finland: Association for Computing Machinery, 2008, pp. 1096–1103. ISBN: 9781605582054. DOI: 10.1145/1390156.1390294. URL: https://doi.org/10.1145/1390156.1390294.

[187] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Advances in neural information processing systems*. 2016, pp. 3630–3638.

[188] Paul Viola, Michael Jones, et al. "Rapid object detection using a boosted cascade of simple features". In: *CVPR (1)* 1.511-518 (2001), p. 3.

[189] Riccardo Volpi et al. "On the Road to Online Adaptation for Semantic Image Segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 19184–19195.

[190] Dequan Wang et al. "Tent: Fully test-time adaptation by entropy minimization". In: *arXiv preprint arXiv:2006.10726* (2020).

[191] Mark Weber et al. "Step: Segmenting and tracking every pixel". In: *arXiv preprint arXiv:2102.11859* (2021).

[192] Tyler Westenbroek et al. "Learning min-norm stabilizing control laws for systems with unknown dynamics". In: *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE. 2020, pp. 737–744.

[193] Eric Wong and J Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope". In: *arXiv preprint arXiv:1711.00851* (2017).

[194] Jiajun Wu et al. "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning". In: *Advances in neural information processing systems* 28 (2015), pp. 127–135.

[195] Enze Xie et al. "SegFormer: Simple and efficient design for semantic segmentation with transformers". In: *Advances in Neural Information Processing Systems* 34 (2021).

[196] Zhenda Xie et al. "SimMIM: A Simple Framework for Masked Image Modeling". In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[197] Ning Xu et al. "Youtube-vos: Sequence-to-sequence video object segmentation". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 585–601.

[198] Bo Zhang et al. "Deep exemplar-based video colorization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8052–8061.

[199] Hao Zhang et al. "SVM-KNN: Discriminative nearest neighbor classification for visual category recognition". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 2126–2136.

[200] Hongyang Zhang et al. "Theoretically principled trade-off between robustness and accuracy". In: *arXiv preprint arXiv:1901.08573* (2019).

[201] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful image colorization". In: *European conference on computer vision*. Springer. 2016, pp. 649–666.

[202] Richard Zhang et al. "Real-time user-guided image colorization with learned deep priors". In: (2017).

[203] Zhenyu Zhang et al. "Online depth learning against forgetting in monocular videos". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4494–4503.

[204] Stephan Zheng et al. "Improving the robustness of deep neural networks via stability training". In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, pp. 4480–4488.

[205] Yiran Zhong, Hongdong Li, and Yuchao Dai. "Open-world stereo video matching with deep rnn". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 101–116.

[206] Wentao Zhu et al. "Neural Multi-Scale Self-Supervised Registration for Echocardiogram Dense Tracking". In: *arXiv preprint arXiv:1906.07357* (2019).

[207] Barret Zoph et al. "Rethinking pre-training and self-training". In: *Advances in neural information processing systems* 33 (2020), pp. 3833–3845.

[208] Yang Zou et al. "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 289–305.