Adaptation Based Approaches to Distribution Shift Problems

by

Marvin Mengxin Zhang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Sergey Levine, Chair
Professor Pieter Abbeel
Professor Jacob Steinhardt

Fall 2021

Adaptation Based Approaches to Distribution Shift Problems

Abstract

Adaptation Based Approaches to Distribution Shift Problems

by

Marvin Mengxin Zhang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Sergey Levine, Chair

Distribution shift in machine learning refers to the general problem where a model is evaluated on test data drawn from a different distribution than the training data distribution. In real world applications, distribution shift is oftentimes not the exception, but the norm: for example, the data distribution may change over time, the model may be tested in new and unforeseen circumstances, or distribution shift may even be a consequence of the problem definition itself. In order to realize the full potential of machine learning, therefore, effective solutions must be developed to deal with distribution shift problems.

In this thesis, we first review the many ways that distribution shift arises in machine learning settings, using several examples to ground the problem while demonstrating the ubiquity with which the problem occurs. The first few of these examples come from control and reinforcement learning, and for these examples, distribution shift is baked in to the problem formulation itself. As the shifts are easily characterized and peripheral to the main goal, handcrafted techniques can be brought to bear to handle these cases. The subsequent examples illustrate ways in which shift creeps into real world supervised learning problems, which motivates the study and development of general purpose learning paradigms that can be used to tackle these and future examples.

The paradigms we focus on in this thesis revolve around the concept of *adaptation*: leveraging the information available at test time in order to change the model to better handle the test data. We ask an open-ended question: how can and should a model adapt when faced with distribution shift? Our first proposal for answering this question is the paradigm of adaptive risk minimization: when provided examples at training time of what different shifts are likely to occur, the model should *learn to adapt* to these training shifts, thus better preparing itself for similar instances of test shift. We formalize and instantiate methods for this paradigm through the toolkit of meta-learning, demonstrating that these methods are competitive with, and oftentimes superior to, prior approaches for handling distribution shift. Our second proposal for answering the above question lies at the other end of the

spectrum: even without access to the training procedure or multiple test points, the model can still rely on inductive biases relayed by *data augmentations* in order to adapt. This leads to the method of marginal entropy minimization with one test point, a broadly applicable "slot-in replacement" for standard inference that proves to be effective for a wide range of commonly used models on a number of challenging distribution shift benchmarks.

To my mom's mom.

Contents

List of Figures

List of Tables

Acknowledgments

First, I thank my advisor, Sergey Levine. You have always been there for me every step of the way, starting all the way back to when I was an undergraduate "researcher" looking for direction and guidance. I truly appreciate all of the advice you have given me and all of the time and effort you have put in. I thank the other members of my qualifying exams and dissertation committees: Pieter Abbeel, Jacob Steinhardt, and Chelsea Finn.

I have benefited greatly from having a number of great research mentors. I thank Pieter Abbeel, who first took me in as an undergraduate searching for research opportunities and then had enough faith in me to take me on as a PhD student. Your many years of guidance and wisdom have had a lasting impact on how I conduct and think about research. I thank Matt Johnson, who perhaps has had the highest impact-to-time ratio on my graduate career, and who has been a valued mentor and friend. I thank Matt Hoffman, who graciously hosted me as an intern, exposed me to completely new ways of thinking about problems, and is an endless well of knowledge and interesting thoughts. Finally, I thank Chelsea Finn, who was a guiding force on my very first research project and then circled back around to be a guiding force on my last two projects. You are an inspiration.

I thank all of my excellent collaborators, without whom the work in this thesis and beyond would not have been possible. I have had the privilege of working with many fantastic researchers, and I have also had the privilege of being in a large research group with many lab mates and friends. I have shared the most time on this journey with JD Co Reyes, Justin Fu, Vitchyr Pong, and Avi Singh, and I consider myself fortunate for that. (We've all made it now!) I thank Coline Devin, Abhishek Gupta, Greg Kahn, and Anusha Nagabandi for always being one year wiser and thus able to answer all questions. I thank Young Geng, Laura Smith, and Nikita Dhawan, all of whom I ostensibly mentored but also taught me just as much, if not more.

I must especially thank a few of my collaborators and friends. First, I thank Abhishek Gupta, who is so many things: a close friend, a fellow slacker, a great memer, and a fantastic researcher. I thank Sharad Vikram, who showed me both what it means to be a good Bayesian and what it means to be a good programmer, but whose jump shot still needs work. I thank Ashvin Nair, who takes the initiative for social events and interview prep so I don't have to, and whose jump shot no longer needs work. Finally, I thank Michael Janner, who gave me a jump start in the middle of the pandemic, right when I needed it the most.

I thank my friends outside of my direct PhD bubble for keeping me sane, especially through the last two years. To Shreyas Chand, Matt Fong, Brian Hou, Derek Leung, Iris Wang, Patricia Weng, and George Yiu: thanks for so many reasons. Thanks for making my undergraduate experience so memorable and for always inspiring me to do better. Thanks for allowing me to be a part of your lives and share in your joyous life events. Thanks for commiserating, supporting, joking, traveling, dining, gaming, and just being all around great people. To Brian and Derek in particular: if I can do it, you can definitely do it, too.

I thank my mom, Meili Zhang; dad, Zhengsheng Yao; and brother, Yao Yao, who have helped guide me all my life and have set such great examples for me to strive for. I am lucky in many ways, but the first and most significant way is in having all of you to call family. I thank my older cousin, Yixin Ren, who beat me to the punch and set another great example for me for graduate school and beyond. To my younger cousins, Yuxin Zhang and Haoyu Ren: I hope that I can provide yet another example for you to follow in the near future. Thank you, also, to Yannie and Charlie: Charlie, whatever you choose to do in the distant future, make sure to leverage all these great examples around you. It will help.

Finally, thank you to Grace Chen. Thank you for making me laugh endlessly and harder than I thought possible. Thank you for sticking with me through all of the good, bad, and ugly. Thank you for being an inspiration with your diligence, dedication, and drive. You mean the world to me. Thank you, also, to Grace's family, for all of the support, humor, and boba. Last, but not least, thank you to Peanut for your cuteness and unconditional love. Good buddy.

Chapter 1

Introduction

Machine learning models are playing an ever increasing role in our daily lives. Smartphones come with image and speech recognition built in, autonomous cars roam the streets with constantly improving capabilities, and language models can generate remarkable emails, stories, and even code. In other areas, machine learning models are also finding use, and success, in a diverse set of applications spanning agriculture [132], medicine [49, 173], ecology [39], and much more. Much of this progress has been driven by the advent of deep learning, thanks to the ability of deep neural network models to process huge amounts of raw, high dimensional data in order to learn highly performant prediction and decision rules [69]. However, despite all of this progress, deep learning has not convincingly addressed, and has perhaps even exasperated, one fundamental issue in machine learning: *distribution shift*.

Distribution shift refers to the problem in which the model encounters a test data distribution that is different from the distribution of data it was trained on. Though the vast majority of machine learning research studies theoretical settings and empirical benchmarks in which the training and test distribution are assumed to be the same [236], this assumption is routinely violated in real world applications. Many such violations are benign, in that they do not negatively affect the model's overall accuracy. However, other shifts may represent significant challenges for machine learning performance, fairness, and safety. For example, naïvely trained deep models for autonomous driving applications can suffer at segmenting nighttime driving scenes [46], recognizing traffic signs in adverse weather conditions [225], and even detecting pedestrians with darker skin tones [248]. Other examples abound: image classifiers perform poorly at recognizing common animals in atypical backgrounds [15], speech recognition systems display higher error rates for Black speakers compared to white speakers [111], and so on, and so forth. Without effective techniques for dealing with distribution shift, therefore, we cannot realize the full potential of machine learning in the real world.

Because the distribution shift problem is so broad, it has been actively studied for decades under many different frameworks and assumptions [172], including domain adaptation [204, 43], domain generalization [21, 151], distributionally robust optimization [18, 95], and more. Many of these works have laid the foundation for formalizing and dealing with distribution shift, however, the advent of deep learning has also required rethinking prior techniques and takeaways. For example, strong regularization is important for achieving good performance on rare subgroups, due to the ability of deep networks to perfectly fit the training data [192]. Another interesting result is that standard empirical risk minimization techniques perform on par with specialized methods for domain generalization when both are carefully tuned to fit the data [74], bringing into question the necessity of such methods. Lastly, as models and datasets continue to grow, there is increasing evidence that simply training larger models on vast amounts of data can mitigate many commonly occurring instances of distribution shift [163, 87]. Thus, there remain many open questions regarding when deep neural networks are susceptible to shift and which techniques are most effective at addressing these scenarios.

This thesis tackles these questions in two distinct parts. In the first part, we review several examples of distribution shift problems occurring in deep reinforcement learning (RL) and supervised learning settings. The examples from RL serve to illustrate two key takeaways related to distribution shift. First, it is often the case that the most effective way of dealing with shift is to bring to bear human insights and domain knowledge within the specific problem. Second, when the model's accuracy under shift is not the primary goal, but rather an obstacle to be overcome, techniques for mitigating and effectively reducing shift may be the shortest path to achieving the desired goal. Though these takeaways are important for both researchers and practitioners to keep in mind, they do not apply to all problems: effective handcrafted techniques require expertise in both machine learning and the domain of interest, which may be infeasible, and improving the model's accuracy oftentimes is the ultimate goal. We illustrate these points by pivoting to examples from supervised learning. Supervised learning encompasses the majority of real world machine learning applications, thus the examples we construct here are designed to faithfully represent important real world problems. For these examples, handcrafted techniques for combating shift have had mixed success and are not designed to generalize to other examples. Thus, those working on these problems are still searching for more effective techniques, and those dealing with other instances of shift have not benefited much from the progress made on these problems.

These examples motivate the second part of this thesis, in which we study and devise general purpose methods for handling these, other, and future instances of distribution shift. In particular, we focus on paradigms and methods centered around *adaptation*, in which the model may change its parameters at test time after observing the available evidence. We argue that adaptation is particularly well suited for the typical deep learning setup, in which models are often overparameterized and capable of incorporating additional information beyond the training data. Through the toolkits of meta-learning and data augmentation, we propose two different adaptation based approaches, under varying assumptions of access to the training procedure and availability of test time data. The first approach is the adaptive risk minimization framework, which aims to optimize models such that they learn how to adapt to different shifts provided at training time, thus better equipping these models to adapt to similar instances of test shift. The second approach is the method of marginal entropy minimization with one test point, which operates directly on a wide range of pretrained models. This method adapts the model via a novel combination of confidence maximization with data augmentations to try and maximally leverage the inductive biases afforded by each technique. We evaluate these approaches on a number of supervised learning problems, including the aforementioned examples, and demonstrate that adaptation based approaches are worthy of further study alongside, and in conjunction with, the other tools for handling shift.

1.1 Outline

In Chapter 2, we provide preliminaries and notation for RL and supervised learning that will be used throughout this thesis. This exposition is supplemented by more in depth review of relevant preliminaries and related work in specific sections.

Part I: distribution shift problems

Chapter 3 and Chapter 4 comprise Part I of this thesis. This part of the thesis focuses on distribution shift *problems*, i.e., when and how does shift manifest in machine learning applications? We answer this question by providing concrete examples of shift in different domains, and we supplement this discussion with some results on prior attempts to handle and mitigate these shifts. Concretizing the discussion through these examples allows us to provide a frame of reference for an otherwise broad and unruly topic. In principle, distribution shift can refer to any scenario in which a machine learning model is trained on data from a certain distribution and evaluated on examples from a different distribution. But considering the ubiquity with which this happens in the real world, we believe that many such instances of shift are benign in that they do not significantly degrade model performance or the achievement of the downstream goal. Therefore, identifying cases of distribution shift that are both harmful and relevant to real applications should be considered alongside the methodology for handling the shifts themselves.

Chapter 3. Chapter 3 may be skipped without consequence for readers not interested in RL and related topics. In this chapter, we highlight several instances of distribution shift problems arising within control and RL applications. First, we discuss our work on learning locomotion strategies for a unique robot design known as a tensegrity robot. In this case, the model is a policy, i.e., it uses observations gathered from the environment in order to predict the action that the robot should take. The shift occurs when attempting to transfer learned policies from simulation to the real robot, and this shift is caused by two main factors. First, the observations may differ due to imperfections in the simulation and differences in the robot and environment setup. Second, the dynamics of the robot and environment may differ, i.e., taking the same action in simulation versus in the real world may lead to different future observations. For this example, we find that a combination of several techniques, mostly relying on domain knowledge and expert heuristics, is successful at combating this simulation to real world shift and allowing for the successful direct transfer of policies. In particular, the techniques include reducing the observation space to use only the most reliable sensors, constraining the action space of the robot to motor commands computed to be safe for the hardware, and injecting significant noise into the training in simulation to improve policy robustness. This work was presented at the International Conference on Robotics and Automation, 2017 [269].

Second, we detail an instance of distribution shift in model-based RL arising from using a trained dynamics model to predict state transitions under a new policy. Here, the model is predicting the future state and reward from the current state and action, and it learns to do so under the distribution of actions from the current and previous policies seen during training. As the RL training step updates the policy to try and improve the rewards obtained in the environment, the action distribution from the new policy is inevitably different and may cause the model to be evaluated for states and actions that are not well covered by its training distribution. Even more insidiously, the RL algorithm is optimizing the performance of the policy within the learned model, thus the policy may even learn to exploit the model by specifically choosing actions for which the model erroneously predicts high rewards. We find in this work that there is a particularly simple fix when it comes to dealing with this shift: reduce the shift by only asking the model to predict for one (or a small number of) time step(s) into the future. Thanks to the empirical observation that dynamics models are able to generalize to small changes in the state and action distribution when trained with sufficient data, this fix enables an efficient and powerful model-based RL algorithm that can successfully learn policies for a number of simulated robotic environments. This work appeared in Advances in Neural Information Processing Systems, 2019 [103].

Finally, we describe our work in learning tasks from human video demonstrations, in which the shift is a consequence of the mismatch in appearance and morphology between human videos and robot videos. A number of approaches have been proposed for robotic RL and imitation learning from high dimensional image observations which rely on the ability to obtain images directly from the robot, e.g., via teleoperation. These approaches make use of models which predict rewards or actions from images, thus they naturally suffer if they are instead trained with human images and queried with robot images, or vice versa. We attempt to resolve this shift via a separate learned model that translates images from a human demonstration video into synthetic, but realistic, robot images, thus enabling the use of similar models for reward and action prediction as in prior approaches. However, we find that this translation cannot perfectly mitigate the human to robot video shift, thus we incorporate this technique into a more sophisticated overall algorithm involving online human feedback and fine tuning of the learned models. This algorithm is then capable of learning complex multi-stage tasks, such as operating a coffee machine, on a real robotic system directly from videos of humans demonstrating the task. This work appeared in Robotics: Science and Systems, 2020 [208].

In each of these RL examples, the distribution shift comes as a result of the problem formulation itself – that is, our desire to transfer policies from simulation to the real world, use dynamics models to predict for updated policies, and learn from human demonstration videos. In the first example, the goal is specifically to learn locomotion strategies for tensegrity robots that transfer from simulation to the real world, thus the techniques for dealing with shift are similarly specific to the domain. In the second and third examples, the goal in these works is not to devise techniques for directly improving model performance under shift, but rather to achieve the downstream goal of learning good behavioral policies for solving the specified task, by any means necessary. As such, as we dive into in Chapter 3, these works take the path of mitigating and reducing shift through querying the models for only short horizons or on images that have been translated into the robot domain. Overall, none of these works attempt to devise general purpose solutions for dealing with shift, in terms of actually improving model accuracy, as their goals and desiderata can be achieved through more direct means.

Chapter 4. Motivated by these findings, we ask the question: are there examples of common, frequently occurring distribution shifts that would benefit from general purpose methods and solutions? This is the topic of Chapter 4, in which we shift our focus to supervised learning. As discussed above, supervised learning comprises most real world applications of machine learning, and in contrast to RL, distribution shift in these problems oftentimes directly impacts the ultimate metric of model predictive accuracy. We discuss both our work and related work on this subject. Numerous test sets and benchmark suites have been proposed which model a wide assortment of distribution shifts, and we first provide an overview of the pieces of this landscape that are relevant to the later parts of this thesis. We then present our work in this area, which complements existing benchmarks by introducing a suite of problems in domain generalization and subpopulation shift which are designed to be tractable while faithfully representing real world applications of interest, including applications in medicine, biology, ecology, and more. We named this benchmark the Wilds benchmark, and this work was presented at the International Conference on Machine Learning, 2021 [113]. We conclude this part of the thesis, and lead into the next part, by highlighting distribution shift problems we have proposed that motivate, and are set up to test, methods that adapt at test time to distribution shift.

Part II: adaptation based approaches

Chapter 5 and Chapter 6 comprise Part II of this thesis. This part of the thesis focuses on *approaches* to tackling distribution shift, with particular emphasis on approaches based on model *adaptation* at test time.

Chapter 5. In Chapter 5, we consider the problem setting in which training data are provided from multiple domains and the goal is to perform well at test time in the presence of domain shift, i.e., the test data come from a new domain or a new distribution of domains. These test scenarios correspond to the problems of domain generalization and subpopulation shift, respectively. We further make the assumption that, at test time, the test data will be available as either a batch or stream of unlabeled inputs, and the model is allowed to adapt using these inputs before making its final predictions on these data. The key question then becomes: how should the model adapt? We approach this question from the perspective of meta-learning: rather than devising a handcrafted adaptation procedure, we instantiate an adaptation model that represents an adaptation procedure with learnable parameters. Since we have access to multiple training domains, the model and adaptation model can be jointly learned by simulating domain shifts within the training data and optimizing for the model's post adaptation performance, effectively learning how to adapt to these shifts. We highlight relevant theory from prior work that provides strong theoretical justification for this framework, which we refer to adaptive risk minimization. We then detail a general algorithm and three specific methods, all inspired by meta-learning, for optimizing the objective specified by this framework. We highlight the aforementioned set of problems for which adaptation is beneficial for improved performance, and we empirically demonstrate that these methods consistently perform better than prior methods commonly used to tackle shift, including methods for distributional robustness, invariant feature learning, and test time adaptation. Finally, we evaluate our best performing method on the Wilds benchmark, and our results provide support for studying a wide range of methods, each of which may be best suited to tackle specific types of shift. This work appeared in Advances in Neural Information Processing Systems, 2021 [268].

Chapter 6. In Chapter 6, we jump to the opposite end of the spectrum in terms of training and test time assumptions. Not only do we not assume any particular characteristics of the training data, such as multiple domains, we do not even assume access to the training procedure itself – that is, we only assume access to a pretrained model. We are given only one test point which we must predict on, akin to the standard supervised learning evaluation protocol. We refer to this problem setting as test time robustness, in which the specific test point may be leveraged in order to improve the model's predictions on that point. Though the answer is now drastically different, the key question is the same as before: how should the model adapt? We probe this question by investigating existing approaches to test time robustness, including some methods that adapt certain parameters of the model and other methods that rely on data augmentations. We devise a new test time robustness method based on the synthesis of these ideas, and we refer to this method as marginal entropy minimization with one test point. This method is applicable in any test setting in which the model is adaptable and there exists a set of data augmentations designed to be invariant to the label, e.g., image crops and flips which are assumed to not change the underlying class of the image. The method works as a "slot-in" replacement for standard model inference: rather than directly predicting on the test input, we produce multiple copies of the input with different augmentations, compute the model's average, or marginal, predictive distribution across the copies, and then adapt the model by minimizing the entropy of this marginal distribution. Intuitively, this adaptation encourages the model to produce confident predictions while also making the same predictions across different augmentations of the same input. We show empirically that this simple approach works well in practice, improving results for several neural network model architectures on a range of challenging distribution shift test sets. At the time of writing, this work is under review and has been made publicly available [267].

In Chapter 7, we conclude with some general remarks on tackling distribution shift, and we highlight what we believe are important directions for future research.

Chapter 2

General Preliminaries and Notation

Broadly, this thesis studies both the reinforcement learning (RL) and supervised learning settings. In this chapter, we provide general preliminaries and notation for both settings, with more specific details deferred to sections in the appropriate later chapters.

2.1 Reinforcement Learning

In RL, we consider a Markov decision process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho)$. $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively, and $\gamma \in [0, 1)$ is the discount factor. The dynamics or transition distribution is denoted as $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$, the initial state distribution as $\rho(\mathbf{s})$, and the reward function as $r(\mathbf{s}, \mathbf{a}) \in [0, r_{\max}]$. The goal of RL is to learn a policy $\pi(\mathbf{a}|\mathbf{s})$ that maximizes the expected sum of discounted rewards, denoted by η, under the state and action distribution induced by π, p, and ρ,

$$\pi^\star = \underset{\pi}{\operatorname{argmax}}\ \eta[\pi] = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_{\rho,\pi,p}\left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t)\right] .$$

The dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ and reward $r(\mathbf{s}, \mathbf{a})$ functions are assumed to be unknown. Thus, RL algorithms must use samples collected from the MDP in order to try and find $\pi^\star$ [218]. An alternative finite horizon formulation of the MDP replaces the discount factor γ with an episode length T, and in this setting the goal is to maximize the expected sum of rewards for the entire episode,

$$\pi^\star = \underset{\pi}{\operatorname{argmax}}\ \eta[\pi] = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_{\rho,\pi,p}\left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right] .$$

Denoting the entire sequence of states and actions as τ, we can also write the expected sum of rewards as $\mathbb{E}_{\rho,\pi,p}[r(\tau)]$.

Numerous classes of RL algorithms have been proposed for optimizing $\eta[\pi]$. Broadly speaking, model-based RL methods aim to construct a model of the transition distribution,

$p_\theta(\mathbf{s}'|\mathbf{s},\mathbf{a})$, using data collected from interaction with the MDP, typically using supervised learning. The parameterization of p_θ varies based on the method: two popular parameterizations are time varying linear Gaussian models $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) = \mathcal{N}(\mathbf{f}_{\mathbf{s}t}\mathbf{s}_t + \mathbf{f}_{\mathbf{a}t}\mathbf{a}_t, \mathbf{F}_t)$ [121], which allow for tractable optimization but lack the capacity to represent complex dynamics, and neural network models [155], which are expressive but may require more data to fit. Some methods also assume that the reward function has unknown form and predict $r(\mathbf{s},\mathbf{a})$.

Oftentimes in RL and robotics, it is more natural to model the setting as a partially observed Markov decision process (POMDP), rather than an MDP [106]. POMDPs are defined by the tuple $(\mathcal{S},\mathcal{A},\mathcal{O},p,p,r,\gamma,\rho)$, which differs from the MDP definition by the addition of the observation space $\mathcal{O}$ and the emissions distribution $p(\mathbf{o}|\mathbf{s})$, and all other elements of the tuple are the same as the MDP. A finite horizon characterization of the POMDP can be described analogously. The policy $\pi(\mathbf{a}|\mathbf{o})$ is assumed to have access only to the observation $\mathbf{o}$ (and potentially the previous observations) in order to select the action $\mathbf{a}$. This setup accurately describes many scenarios in robotics: the state $\mathbf{s}$ may contain information that is difficult to obtain, such as object or joint positions provided by motion capture systems, whereas the observation $\mathbf{o}$ may rely, e.g., on only the robot's onboard sensors such as cameras and inertial measurement units (IMUs). When deployed, it is desirable to have the robot's policy depend only on measurements that can be easily obtained.

If state information is available during training, such as via an instrumented setup or through simulation, then model-based RL methods can essentially proceed as previously described, with an additional piece to compensate for having to learn $\pi(\mathbf{a}|\mathbf{o})$ rather than $\pi(\mathbf{a}|\mathbf{s})$. For example, the emissions distribution $p(\mathbf{o}|\mathbf{s})$ may also be modeled, or imitation learning may be employed such that $\pi(\mathbf{a}|\mathbf{o})$ learns to match the actions of a learned $\pi(\mathbf{a}|\mathbf{s})$ [149].

2.2 Supervised Learning

In supervised learning, we are given a training data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, consisting of N input-output pairs drawn from a training distribution $p_{\mathbf{x}y}$. In this thesis, we mainly focus on the domain of image classification, in which $\mathbf{x}$ represents raw images and y takes on discrete values corresponding to different possible classes. The goal of supervised learning is commonly stated as learning a model $g_\theta : \mathcal{X} \to \mathcal{Y}$ that minimizes the expected loss (e.g., 0-1 loss) given new input-output pairs sampled from $p_{\mathbf{x}y}$ and the model's predicted outputs. This goal assumes that the test distribution $p^\star_{\mathbf{x}y}$ is the same as $p_{\mathbf{x}y}$ and is typically referred to as risk minimization. The risk minimization objective can be written as

$$\theta^\star = \operatorname*{argmin}_\theta \; \mathbb{E}_{p_{\mathbf{x}y}} \left[\ell(g_\theta(\mathbf{x}), y)\right] .$$

When training the model, a sensible training objective, modulo regularization, is thus to minimize the empirical risk of the model on the training data [236]. That is,

$$\hat{\theta}^\star = \operatorname*{argmin}_\theta \; \frac{1}{N}\sum_{i=1}^N \ell(g_\theta(\mathbf{x}_i), y_i) .$$

When facing distribution shift, however, we may wish to change this goal, as learning a model that performs well on the training distribution $p_{\mathbf{x}y}$ does not guarantee that the model will generalize well to other distributions at test time. Thus, there are various ways we may imagine modifying the empirical risk minimization (ERM) objective, depending on the problem formulation and assumptions. Chapter 4 and Chapter 5 provide detailed instantiations of what these problem formulations and subsequent training modifications may be. In Chapter 6, we turn our attention to test time methods which do not require modifying the model training procedure, instead using various heuristics such as unsupervised adaptation and data augmentation within the test time inference procedure.

Part I

Characterizing Distribution Shift in Reinforcement Learning and Supervised Learning

Chapter 3

Distribution Shift in Reinforcement Learning

In this chapter, we present three examples of distribution shift in reinforcement learning (RL), illustrating the various ways that shift may arise when formulating and solving RL problems. Readers who are not interested in RL may skip this chapter and move on directly to Chapter 4. The examples in this chapter serve to illustrate two takeaways: that shift can often be tackled effectively using expert knowledge, without the need for general purpose techniques, and that shift can often be mitigated if it is not the ultimate goal.

In the first example below, the shift occurs as a result of taking a policy that was trained in simulation and attempting to use it to control the corresponding real robotic system, which the simulation does not perfectly emulate in terms of dynamics and actuation. In the second example, the shift is a result of using dynamics models that were trained with policies from previous iterations of the RL algorithm to predict for a new, updated policy. In the last example, the shift is a direct consequence of the stated goal, in which videos of human demonstrations are to be used as examples for a robot to perform the same task, and the discrepancy between human and robot videos must be resolved.

3.1 Deep RL for Tensegrity Robot Locomotion

Tensegrity robots are a class of robots that are composed of rigid rods connected through a network of elastic cables. These robots are lightweight, low cost, and capable of withstanding significant impacts by distributing force across the entire structure. Their compliance protects the robot and its payload during high speed landings and may also allow for greater mobility and robustness during exploration of rugged and dangerous environments [29].

However, efficient locomotion for tensegrity robots is a challenging problem. Such robots are typically controlled through actuation of motors that extend and contract their cables, thereby changing their overall shape. Many such systems are underactuated because there are usually more cables than motors. Furthermore, actuating one motor can change the entire

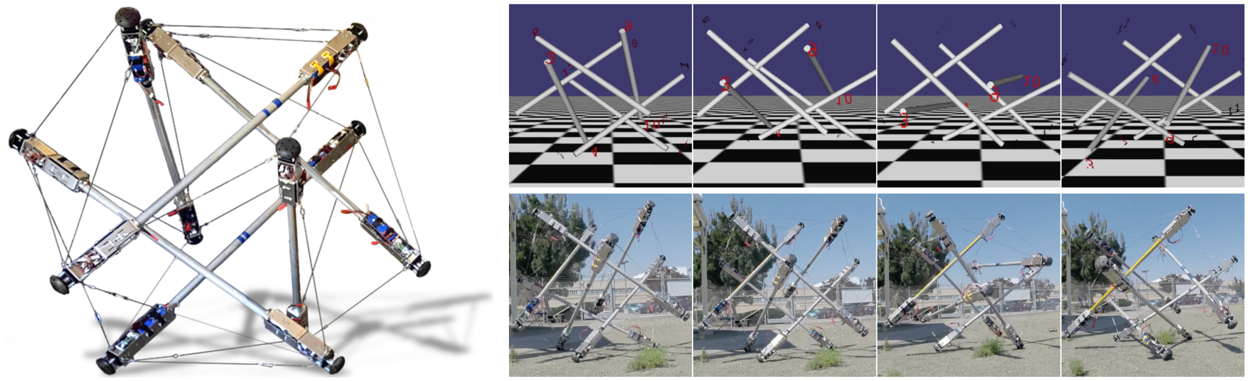


Figure 3.1: Left: The SUPERball tensegrity robot. This robot is composed of six identical rods and 24 cables, 12 of which can be actuated using the motors connected to the ends of each rod. This work uses the onboard IMUs and motor encoders on each of the end caps. Right: a rolling motion toward the camera performed by the SUPERball simulation (top) and the physical SUPERball robot (bottom). The rolling gait is learned from scratch in simulation with our proposed algorithm and, to mitigate shift arising from simulation to real world transfer, uses only the onboard accelerometer sensors for feedback.

shape of the robot, leading to complex, highly coupled dynamics. As such, hand engineering locomotion controllers is unintuitive and time consuming, and such controllers often do not generalize well to different environments. This motivates using learning algorithms to automatically discover successful and efficient behavior.

We present a method for automatically learning locomotion policies represented by general purpose neural networks, which we demonstrate by learning a gait for the Spherical Underactuated Planetary Exploration Robot ball (SUPERball), the tensegrity robot shown in Figure 3.1. We run the learning procedure on a simulated version of SUPERball rather than the real robot. Though this reduces the wear and tear on the prototype robot and the possibility of commanding unsafe actions, training in simulation and testing on the real robot introduces significant distribution shift from the policy's perspective. To deal with this shift, we find that using a reduced observation space, using a safe action space, and training with noise are sufficient for enabling direct simulation to real world transfer.

Related work in simulation to real world transfer

Many methods have been proposed for the direct transfer of policies learned in simulation to the real world, of which we highlight just a few. Cutler et al. used a transition from simple to complex simulations, which are assumed to increase in both cost and fidelity to the real world setting, to reduce the amount of training time needed in the real world [44]. Mordatch et al. optimized trajectories through an ensemble of models perturbed with noise to find trajectories that could be successfully transferred to the real world setting [150]. Our approach also introduces noise into the simulation, and we found that this can help in training policies that are successful in a wide range of simulated terrain and gravity settings as well as the real world. A similar concept arose, subsequently to this work, in simulation to real world transfer for visual tasks. Several papers found that introducing significant variability in the visual appearance of simulations, such as textures and background colors, allowed for direct transfer of object detectors and policies for visual flight and navigation [190, 229].

Learning locomotion policies for SUPERball in simulation

We encode the state of the system $\mathbf{s}$ as the position and velocity of each of the 12 bar endpoints of SUPERball, and the position and velocity of each of the 12 motors, measured in radians, for a total dimensionality of 96. This information is easily obtained in simulation but requires complex, instrumented state estimation techniques on the real robot [36]. Therefore, in order to easily deploy the learned policies "in the wild", we consider two different representations for the observation $\mathbf{o}$ which use only proprioceptive information from the robot's own sensors. The "full" 36 dimensional observation includes motor positions and also uses elevation and rotation angles and angular velocities calculated from the robot's accelerometers and magnetometers. The "limited" observation is 12 dimensional and only uses the acceleration measurement along the bar axis from each of the accelerometers. The 12 dimensional action $\mathbf{a}$ is the instantaneous desired position of each motor.

The specific RL algorithm used in this work is mirror descent GPS (MDGPS), which uses supervised learning to train the "global policy" $\pi(\mathbf{a}|\mathbf{o})$, with supervision coming from several "local policies" $\pi_i(\mathbf{a}|\mathbf{s})$ that are optimized to succeed only from a specific initial state of the task, using full state information. This local policy optimization is much simpler than the goal of the global policy, which is to succeed under partial observability from any initial condition sampled from the initial state distribution. These simplifications allow the use of simple and efficient model-based RL methods for training the local policies [121]. The local policies are optimized to minimize $\eta[\pi_i]$ subject to a bound on the KL divergence between the local policy π_i and a linearization of the global policy $\bar{\pi}_i$. Optimizing the global policy is done using the samples collected in the current iteration, which are used in a supervised fashion to approximately minimize the divergence between the global and local policies.

Due to the structure of tensegrity systems, unsafe actuation of the motors can place the robot into configurations with unacceptable risk of cable or motor failure. The configurations associated with such high tension conditions are difficult to encode analytically and even more difficult to balance against primary task objectives in the cost function. We therefore adopt a simple safety constraint approach to enable safe policy execution on the SUPERball hardware. Specifically, we estimate the cable tensions for a particular set of actuator positions using a simple forward kinematics model of SUPERball. We repeat this process for many different randomly generated motor settings and store 100 million motor positions for which all cable tensions of the robot are below the acceptable threshold. Then, when the policy outputs an action, we compute and command the nearest (ℓ_1 norm) safe action.

Distribution shift in transferring policies to the real robot

In attempting the transfer of learned policies to the real robot, we find that several design choices are crucial. First, introducing Gaussian noise and dropped sensor readings into the simulation is important. This is likely because the real world sensors and actuators are similarly noisy, and the imperfections in the hardware result in sensor and motor unreliability.

The choice of the observation representation, i.e., which sensor measurements to use as input to the policy, also turns out to be very important. We found that interfering magnetic fields near the testing grounds at NASA Ames cause the magnetometers to be unreliable and difficult to calibrate, and this results in significant distribution shift when attempting to transfer a policy using the full observation from simulation, which had no interfering magnetic fields, to the real robot. Thus, we used the limited observation representation.

Lastly, utilizing the kinematic constraints (i.e., safe actions) is extremely helpful in transferring policies learned in simulation directly onto the real robot. In simulation, the physical limits are less restrictive, and going beyond the safe limits does not have any adverse effects. By enforcing stronger constraints on what actions the policy can output, it is more likely that the policy trained with these constraints in simulation will not fail on the real robot.

The limited observation policy that was trained in simulation was successfully run directly on the real SUPERball with no tuning or changes. Over three trials of 100 s each, using the learned policy, SUPERball rolled approximately 12 m, 9 m, and 8 m measured as the linear distance from the robot's start to final position. 12 m is around the maximum distance allowed during each trial due to our limited network range. Despite the distribution shift caused by differences between the simulated and physical robot, the policy was able to successfully produce a gait on SUPERball that is more reliable, and less risky for the hardware, than any previous locomotion controller for this robot.

Summary

We demonstrated that a learned locomotion policy for the SUPERball tensegrity robot can be transferred directly from simulation to the real system. This distribution shift between the simulated and real robot was overcome by expert heuristics: carefully choosing the policy observation representation, restricting actions to known safe regions, and injecting noise into the simulation during training. We showed that, using these techniques, our learned locomotion policies are effective and generalize to the real robot.

3.2 When to Trust Your Dynamics Model

In this section, we study how to most effectively use a dynamics model for policy optimization. Most commonly used model-based approaches lack the improvement guarantees that underpin many model free methods [198]. While it is possible to apply analogous techniques to the study of model-based methods to achieve similar guarantees, it is more difficult to use such analysis to justify model usage in the first place due to pessimistic bounds on model error. In other words, the worst case distribution shift that may be induced by using the model to predict for long horizons and for an updated policy is at odds with guaranteeing monotonic improvement. However, we show that more realistic model error rates are often available in practice, especially when querying the model only for short horizons.

We devise a practical algorithm built on these insights, which we call model-based policy optimization (MBPO), that makes limited use of a predictive model to achieve pronounced improvements in performance. More specifically, we disentangle the task horizon and model horizon by querying the model only for short rollouts. We empirically demonstrate that a large amount of these short model generated rollouts can allow a policy optimization algorithm to learn substantially faster than recent model-based alternatives while retaining the asymptotic performance of the most competitive model-free algorithms.

Related work in analyzing and accounting for model error

Learned models may be incorporated into otherwise model-free methods for improvements in data efficiency. For example, model rollouts may be used to provide extra training examples for a Q-function [217], to improve the target value estimates of existing data points [58], or to provide additional context to a policy [53]. However, the performance of such approaches rapidly degrades with increasing model error [73], motivating work that interpolates between different rollout lengths [31], tunes the ratio of real to model generated data [108], or does not rely on model predictions [82]. Our approach similarly tunes model usage during policy optimization, but we show that justifying non negligible model usage during most points in training requires consideration of the model's ability to generalize outside of its training distribution. Theoretical analysis of model-based RL algorithms has been considered by bounding the discrepancy between returns under a model and those in the real environment of interest [138, 215]. Their approaches enforce a trust region around a reference policy, whereas we do not constrain the policy but instead consider rollout length based on estimated model generalization capacity. More recent work has shown that meta-learning can be a powerful tool for learning models that adapt quickly [154, 153], though no work has yet been directed at adapting to new policies, which requires out of distribution generalization.

Model-based policy optimization with deep RL

We consider a simple model-based RL approach: (1) optimize a policy under a learned model using RL, (2) collect data under the updated policy, and (3) use these data to train a new model. While conceptually straightforward, step (1) may be problematic: many RL algorithms involve iteratively updating the policy and then collecting additional data, in this case from the model, which means that the model will be queried under a different distribution of actions and future states. This distribution shift can result in a policy which, after being optimized under the model, does not perform well in the environment of interest.

To determine how well we can expect our model to generalize to new policies in practice, we empirically measure how the model error under new policies increases with policy change.

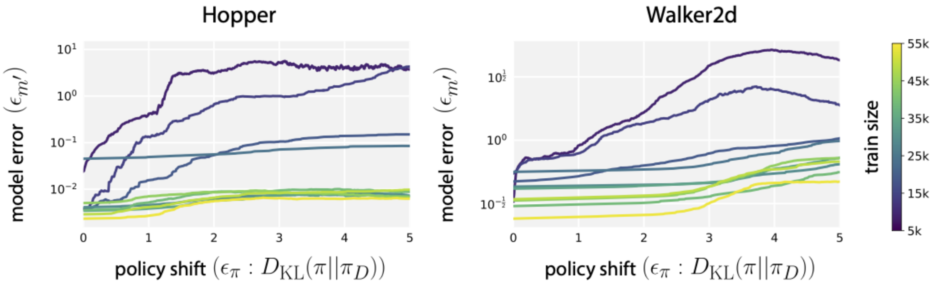


Figure 3.2: We train a predictive model on the state distribution of π_D and evaluate it on policies π of varying KL divergence from π_D without retraining. The color of each curve denotes the amount of data from π_D used to train the model corresponding to that curve. Increasing training data leads to decreasing model error on the training distribution, as well as a decreased influence of state distribution shift on model error.

We train a model on the state distribution of a data collecting policy π_D and then continue policy optimization while measuring the model's loss on all intermediate policies π during this optimization. Figure 3.2 shows that, in practice, models are able to generalize to policies that are relatively similar to π_D. We combine this insight with an additional technique for mitigating the shift that arises from querying the dynamics model for long trajectories, in which the shifted action distribution of the new policy may cause compounding changes in the future state distributions. We introduce the notion of a *branched rollout*, in which we begin a rollout from a state under the previous policy's state distribution $d_{\pi_D}(\mathbf{s})$ and run k steps according to π under the learned model p_θ. This branched rollout structure resembles the scheme proposed in the original Dyna algorithm [217], which can be viewed as a special case of a length 1 branched rollouts. Using a small k and results in more accurate model generated rollouts and, consequently, more successful policy optimization.

We devise a practical model-based RL algorithm, MBPO, based on these insights and the general approach highlighted above. Instantiating MBPO amounts to specifying three design decisions: (1) the parameterization of the model p_θ, (2) how the policy π is optimized given model samples, and (3) how to query the model for samples for policy optimization. For (1), we use a bootstrap ensemble B of dynamics models [40]. For (2), we adopt soft actor critic (SAC) [76] as our policy optimization algorithm. For (3), we use the branching strategy described above, in which model rollouts begin from the state distribution of a different policy under the true environment dynamics.

Experiments

We evaluate MBPO and state-of-the-art model-free and model-based baselines on a set of MuJoCo continuous control tasks [230]. Figure 3.3 (left) shows the learning curves for all methods. These results show that MBPO learns substantially faster than prior model-free

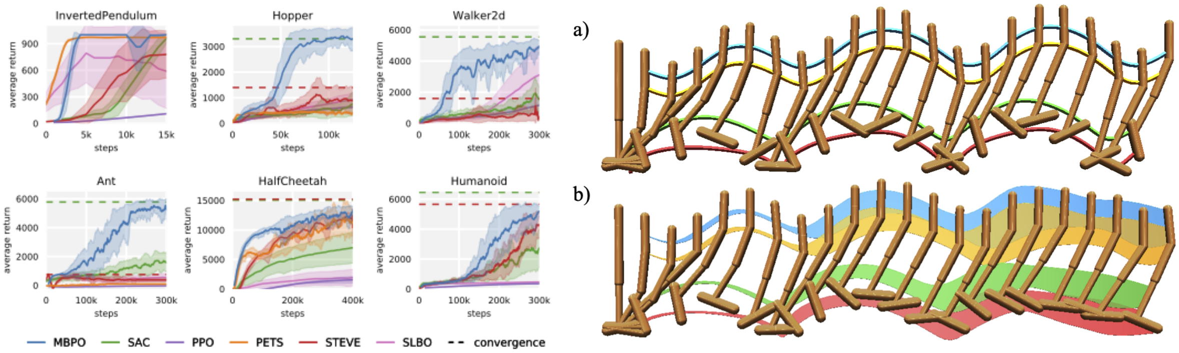


Figure 3.3: Left: Training curves of MBPO (blue) and five baselines on continuous control benchmarks. Solid curves depict the mean of five trials and shaded regions correspond to standard deviation among trials. Right: A 450-step hopping sequence performed in the real environment, with the trajectory of the body's joints traced through space (top), and the same action sequence rolled out under the model 1000 times, with shaded regions corresponding to one standard deviation away from the mean prediction (bottom). The growing uncertainty and deterioration underscore accumulation of model errors.

methods while attaining comparable final performance. This is likely due in part to the fact that short rollouts are more likely to reflect the real dynamics (Figure 3.3 right), reducing the sampling errors and distribution shift that may harm the the final learned performance.

Summary

We have investigated the role of model usage in policy optimization procedures. An empirical study of model generalization shows that predictive models can indeed perform well outside of their training distribution. We further query the model only for short, branched rollouts, thus reducing the compounding shift of long horizon trajectories. The algorithm stemming from these insights, MBPO, has asymptotic performance rivaling the best model-free algorithms, learns substantially faster than prior model-free or model-based methods, and scales to long horizon tasks that often cause model-based methods to fail.

3.3 Learning from Human Demonstration Videos

In this section, we study the setting of robotic learning from videos of human demonstrations. This setting eases the burden associated with providing demonstrations directly on the robot, as human demonstrations require comparatively minimal setup, hardware, and expertise, while also opening up a much wider source of supervision for robots to learn skills. However,

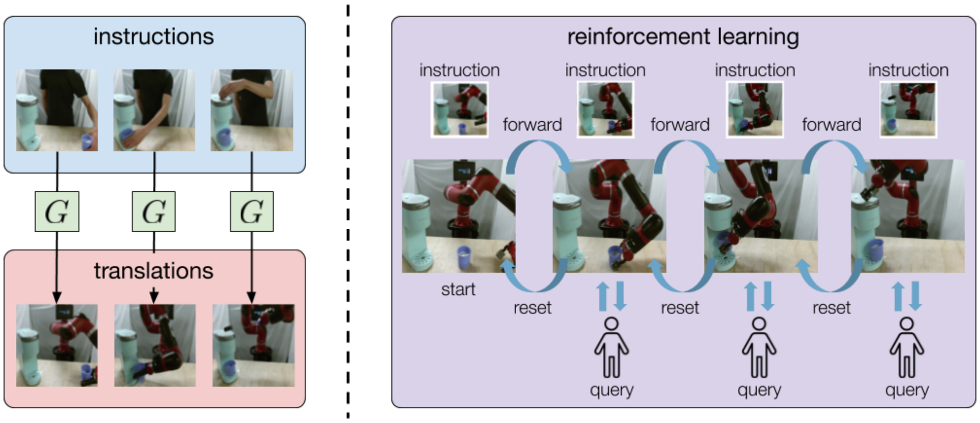


Figure 3.4: Left: Human instructions for each stage (top) are translated at the pixel level into robot instructions (bottom) via CycleGAN. Right: The robot attempts the task using a model-based RL method that extracts a reward signal from the translated images.

as we learn from visual inputs, the distribution shift caused by differences in appearance between the human and robot must be reconciled.

To tackle this challenge, we propose a robotic learning method which we name automated visual instruction-following with demonstrations (AVID). Starting from a human demonstration, our method *translates* this demonstration, at the pixel level, into images of the robot performing the task (see Figure 3.4). This translation significantly lessens the shift between human and robot demonstrations. We learn this translation via CycleGAN [273], a framework for learning unpaired image-to-image translation. In order to further mitigate the distribution shift stemming from artifacts in the translated demonstration, we reduce the demonstration to a few *instruction images*, which are then used to provide a reward for model-based RL, enabling the robot to practice the skill to learn its physical execution.

Related work in learning from human demonstrations

Prior methods for robotic imitation learning have typically used demonstrations consisting of observations and actions from the robot's observation and action space (see, e.g., [79, 20, 7]). We instead study whether we can allow robots to learn from watching a human demonstrator, relaxing standard assumptions that the expert data are given using the robot's own embodiment. Some works have studied this setting using explicit pose and object detection [118, 262], essentially resolving the correspondence problem [156] by instrumenting paired data collection or manually matching hand specified key points. Other approaches have included predictive modeling [183, 233], context translation [260, 202], learning reward representations [199, 200], and meta-learning [265]. In contrast to these works, we explicitly account for the change in embodiment through learned pixel-level translation.

Translating from human to robot demonstrations

We approach human to robot demonstration translation as an unsupervised image-to-image translation problem, where the goal is to map images from a source domain (human images, in our case) to a target domain (robot images) in the absence of paired training data. We assume there exists a bijective mapping that can be uncovered in an unsupervised manner. The approach we use to learn this translation is CycleGAN [273]. Although this approach does not exploit the temporal information that is implicit in our demonstration videos, prior work has shown that CycleGAN can nevertheless successfully translate videos frame by frame, such as translating videos of horses into videos of zebras [273].

Though we could use an imitation learning method that directly learns from the entire translated video demonstration, we find that this approach performs poorly, indicating that the distribution shift is not fully resolved due to the imperfect image translation. We address this issue by instead extracting and learning from key frames that correspond to the completion of stages. Specifically, the *instruction images* for stage i are taken from the t_i-th frame of the translated videos, where each t_i is manually specified by the user. This process is easy for the human as we use a modest number of videos and the time steps intuitively correspond to the completion of natural stages in the task (see Figure 3.4).

Model-based RL from instructions images

The RL procedure that we use is a latent space model predictive control (MPC) method [270]. To specify a reward function for each stage, we train success classifiers for each stage, where the i-th classifier is provided the instruction images at time step t_i as positive examples and all other robot images as negative examples. The log probabilities from the classifiers can then be used as a reward signal, similar to prior work [61, 207].

When the robot is attempting stage s, the planner uses the log probability of the s-th classifier as the reward function and aims to surpass a classifier threshold $\alpha \in [0, 1]$, which is a hyperparameter. Should the threshold be met during planning, the robot will query the human user, at which point the user signals either success or failure through a key press. This human feedback provides a corrective signal for classifier errors that may occur due to imperfect translations resulting in lingering distribution shift. After the human feedback, the current robot image is added to the dataset for training the s-th classifier as either a positive or negative example. Further training the classifiers on these data improves accuracy by combating the false positive problem and offsetting the artifacts in the translated images that the classifiers are initially trained with.

Experiments

We evaluate our method on two temporally extended tasks (Figure 3.5) from vision: operating a personal coffee machine and retrieving a cup from a closed drawer. We compared to a number of prior approaches in imitation learning and learning from human videos. We

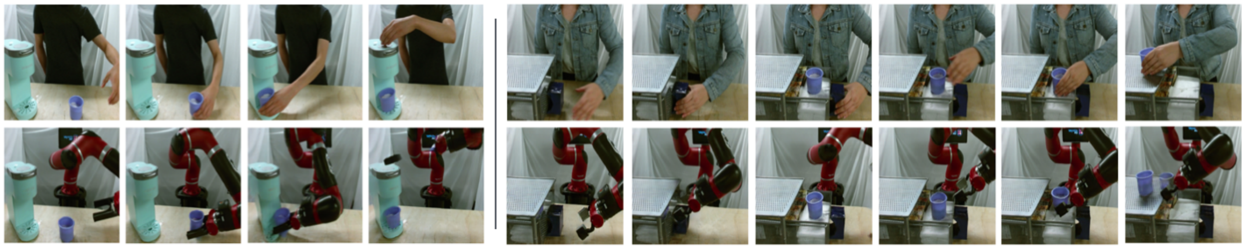


Figure 3.5: Sample sequence of instructions for each of the two tasks, coffee making (left) and cup retrieval (right). The instruction images are segmented from a human demonstration (top), then translated into the robot's domain (bottom) via CycleGAN. Note the artifacts in the generated translations, e.g., the displaced robot gripper in the last instruction image on the left and the deformed cup in the last image on the right.

		Coffee making			**Cup retrieval**				
Supervision	**Method**	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
Human Demos	AVID (ours)	**100%**	**80%**	**80%**	**100%**	**100%**	**100%**	**80%**	**70%**
	Full video ablation	70%	10%	0%	0%	0%	0%	0%	0%
	Pixel space ablation	60%	20%	0%	50%	50%	30%	10%	0%
	TCN [200]	10%	10%	0%	60%	20%	0%	0%	0%
Teleoperated Robot Demos	BCO [231]	80%	30%	0%	30%	10%	0%	0%	0%
	Behavioral Cloning	90%	90%	90%	100%	100%	60%	60%	40%

Table 3.1: We report success rates up to and including each stage for both tasks, over 10 trials. The top rows are methods that learn from human demonstrations, and we bold the best performance in this category. The bottom two rows are methods trained with direct access to demonstrations given on the real robot itself. AVID outperforms all other methods from human demonstrations, succeeding 8 times out of 10 on coffee making and 7 times out of 10 on cup retrieval, and even outperforms behavioral cloning from teleoperated demonstrations on the later stages of cup retrieval.

also evaluated an ablation of our method that learns from the full translated demonstration video, to better understand the effects of shift caused by imperfect translations.

We report success rates for both coffee making and cup retrieval in Table 3.1. Success rates are evaluated using 10 trials per method per task, with no human feedback or manual intervention. For both tasks, AVID achieves the best performance among the methods that use human demonstrations, consistently learning the earlier stages perfectly and suffering less from accumulating errors in the later stages. The full video ablation can learn to pick up the cup 70% of the time for the coffee making task, but it fails on the later stages and also fails entirely for cup retrieval. This indicates the difficulty of learning from full translated demonstrations due to the artifacts in the translations causing additional distribution shift.

Summary

We presented AVID, a method for learning visual robotic multi-stage tasks directly from videos of human demonstrations. AVID uses image-to-image translation, via CycleGAN, to generate robot instruction images that enable a stagewise model-based RL algorithm to learn from raw image observations. By using only instruction images rather than the full translated video, this translation step is able to handle most of the distribution shift arising from learning from human demonstrations, and the resulting instruction images are supplemented by the RL algorithm which uses sparse human feedback. We demonstrated that AVID is capable of learning multi-stage coffee making and cup retrieval tasks, attaining substantially better results than prior methods and ablations using human demonstrations.

3.4 Discussion

In this chapter, we illustrated three distinct examples of how distribution shift can impact RL applications. In the first example, the goal was to mitigate shift from transferring a policy for tensegrity robot locomotion trained in simulation to the real world. Combined with the benefits of learning policies, rather than hand designing them, and the robot construction itself, a number of domain specific techniques ended up proving successful in this case. Specifically, using a limited observation representation with noise added during training, along with constraining actions to known safe regions of the action space, allowed for direct simulation to real world transfer. In this case, we had the specific goal of tensegrity robot locomotion, thus the techniques for mitigating shift were also specific to task at hand.

The second and third examples were more general: the shift arose from using models to predict for new policies and learning tasks from videos of human demonstrations. However, in these examples, handling the shift was not the primary goal, but rather the shift presented a potential obstacle to reaching the goal. Thus, success for these examples instead took the route of mitigating the shifts. For example, rather than attempting to improve the model's predictive performance for new policies, the rollout length was instead truncated to reduce shift. And, because the CycleGAN could not perfectly remove the shift from human to robot videos, only specific instruction images were used, and these were further supplemented by online data collection and human feedback. These techniques cleared the path for achieving our actual desiderata of effective model-based RL and learning from humans.

In the next chapter, we turn our focus to supervised learning. The difference, compared to these examples, is that we typically wish to handle a broad shift of potential shifts, and this is our primary, rather than auxiliary, goal when learning models. The examples we provide for supervised learning will therefore motivate Chapter 5 and Chapter 6, in which we present general purpose paradigms and methods for combating shift in supervised learning settings.

Chapter 4

Distribution Shift in Supervised Learning

In this chapter, we characterize the landscape of distribution shift problems in supervised learning, focusing on benchmarks and test sets widely used in deep learning to assess model robustness and generalization. We highlight several problems that are subsequently used for evaluation in Chapter 5 and Chapter 6. We also present our own work on devising distribution shift benchmarks for supervised learning. These benchmarks aim to complement existing work by providing natural examples of when shift occurs in important real world applications, as well as examples of problems for which model adaptation is beneficial for improved performance.

4.1 Prior Work on Benchmarks for Distribution Shift

In this section, we discuss existing distribution shift benchmarks in more detail, categorizing them by how they induce their respective distribution shifts. We restrict our attention to publicly available datasets. While others have studied some proprietary datasets with realistic distribution shifts, such as the StreetView StoreFronts dataset [89] or diabetic retinopathy datasets [45], these datasets are not publicly available due to privacy and other commercial reasons.

Distribution shifts from synthetic transformations

Some of the most widely adopted benchmarks induce distribution shifts by synthetically transforming the data. Examples include rotated and translated versions of the MNIST and CIFAR datasets [251, 74]; surface variations such as color in colored MNIST [74] and texture in stylized ImageNet [65], and datasets that crop out objects and replace their backgrounds, as in the Backgrounds Challenge [255] and other similar datasets [192, 112].

ImageNet-C [84] is a widely used benchmark that augments the ImageNet test set [189] with 15 different types of image corruptions each at five different levels of severity. These corruptions include the subcategories of blur corruptions, noise corruptions, digital corruptions, and simulated weather corruptions. Other variants of image corruption benchmarks also exist for ImageNet [64] as well as other datasets such as CIFAR [84].

Benchmarks for adversarial robustness also fall in this category of distribution shifts from transformations [70, 42]. Recent work on temporal perturbations with the ImageNet-Vid-Robust and YTBB-Robust datasets [201] represents a form of distribution shift that impacts real world applications. Outside of visual object recognition, other work has used synthetic datasets and transformations to explore compositional generalization, e.g., SCAN [119].

Distribution shifts from modified data collection

A number of recent works aim to measure out of distribution generalization to test sets that are collected using a modified procedure compared to the training dataset. Examples include the CIFAR-10.1 test set [176], and for ImageNet alone, several such test sets have also been proposed. Recent examples include the ImageNetV2 dataset [177], which was constructed to be similar to the original ImageNet dataset but can still result in lower model accuracy, and ImageNet-Sketch [244], which was collected using Google image queries of sketches of ImageNet classes.

ImageNet-R [89] tests generalization across different renditions by collecting data from Flickr of images belonging to an ImageNet category plus a keyword such as “art”, “cartoon”, “graffiti”, etc. Thus, the data are a mix of real and synthetic naturally occurring images, with significantly different statistics compared to the standard ImageNet training and test sets. ImageNet-A [86] is a collection of images belonging to ImageNet categories that are selected from a larger pool based on which images are misclassified by a trained ResNet-50 model [80] while remaining visually clear to humans. These are real and naturally occuring images that have passed through an “adversarial filtration” process, thus they comprise a challenging testbed for many models trained on ImageNet.

Distribution shifts from synthetic to real transfers

Fully synthetic datasets such as SYNTHIA [187] and StreetHazards [88] have been adopted for out of distribution detection as well as other problem settings such as domain adaptation and generalization, e.g., by testing robustness to transformations in the seasons, weather, time, or architectural style [92, 240]. While the data are synthetic, they can still look realistic if a high fidelity simulator is used. In particular, synthetic benchmarks that study transfers from synthetic to real data [62, 184, 168] can be important tools for tackling real world problems: even though the data are synthesized, the synthetic to real distribution shift can still be realistic in contexts where real data are much harder to acquire than synthetic data [17].

Distribution shifts from constrained splits

Other benchmarks do not rely on transformations but instead split the data in a way that induces particular distribution shifts. These benchmarks have realistic data, e.g., the data points are derived from real world photos, but they do not necessarily reflect distribution shifts that would arise in the wild. For example, BREEDS [195] tests generalization to unseen ImageNet subclasses by holding out subclasses specified by several controllable parameters; similarly, NICO [81] considers subclasses that are defined by their context, such as dogs at home versus dogs on the beach; DeepFashion-Remixed [89] constrains the training set to include only photos from a single camera viewpoint and tests generalization to unseen camera viewpoints; BDD-Anomaly [88] uses a driving dataset but with all motorcycles, trains, and bicycles removed from the training set only; and ObjectNet [13] comprises images taken from a few prespecified viewpoints, allowing for systematic evaluation for robustness to camera angle changes but deviating from natural camera angles.

A well studied special case of this category is the class of distribution shifts obtained by combining several disparate datasets [232], training on one or more of them, and then testing on the remaining datasets. Unlike test sets such as ImageNetV2, many of these distribution shifts were constructed to be more drastic than might arise in the wild. For example, standard benchmarks for the domain adaptation problem setting include training on MNIST but testing on SVHN street signs [234, 92]. Standard benchmarks for the domain generalization problem setting include transfers across datasets containing different renditions (e.g., photos, clipart, sketches) in DomainNet [167], PACS [123], and Office-Home [237], and VLCS [57], which tests generalization across similar visual object recognition datasets. Recently, DomainBed collected many of these test sets together into a benchmark suite, and surprisingly, their main result was that a well tuned implementation of empirical risk minimization (ERM) was competitive with state of the art specialized methods across the suite of problems [74]. We discuss this result in greater detail in the later chapters.

4.2 WILDS: A Benchmark of in-the-Wild Distribution Shifts

In this section, we consider two common types of distribution shifts: *domain generalization* and *subpopulation shift* (Figure 4.1). Both of these shifts arise naturally in many real world scenarios, and prior work has shown that they can substantially degrade model performance. In domain generalization, the training and test distributions comprise data from related but distinct domains, such as patients from different hospitals [266], images taken by different cameras [15], bioassays from different cell types [125], or satellite images from different countries and time periods [105]. In subpopulation shift, we consider test distributions that are subpopulations of the training distribution, with the goal of doing well even on the worst case subpopulation; e.g., we might seek models that perform well on all demographic subpopulations, including minority individuals [32].

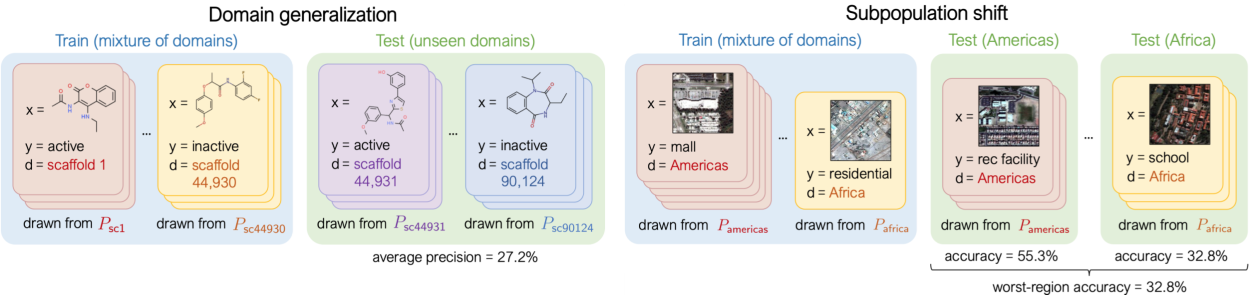
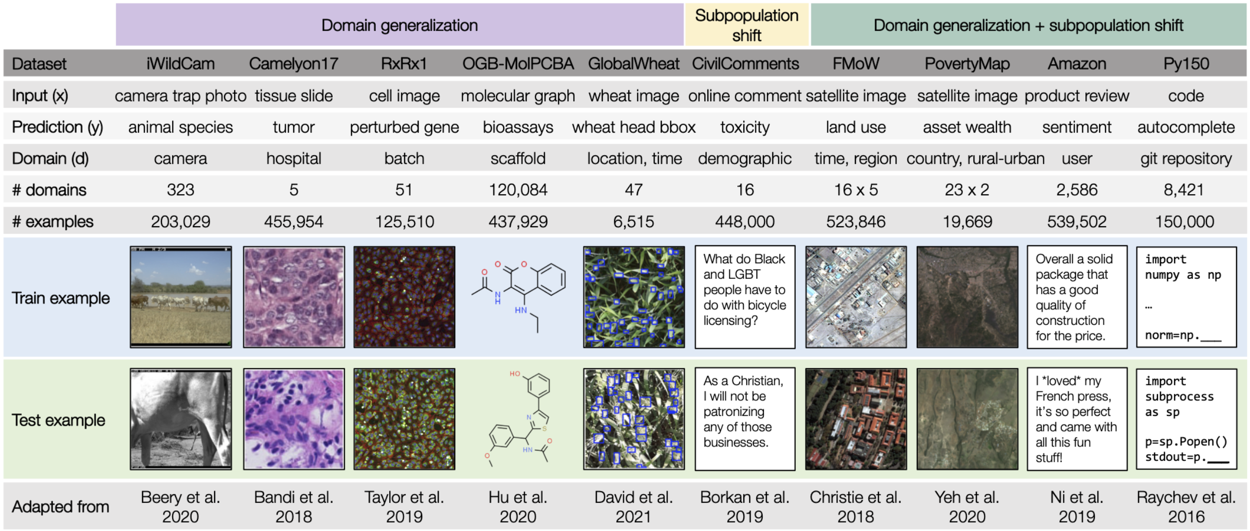


Figure 4.1: In each WILDS dataset, each data point is associated with a domain. Each domain corresponds to a distribution over data points which are similar in some way, e.g., molecules with the same scaffold, or satellite images from the same region. We study two types of distribution shifts. **Left:** In *domain generalization*, we train and test on disjoint sets of domains. The goal is to generalize to domains unseen during training. **Right:** In *subpopulation shift*, the training and test domains overlap, but their relative proportions differ. We typically assess models by their worst performance over test domains, each of which correspond to a subpopulation of interest.

Despite their ubiquity in real world deployments, these types of distribution shifts are underrepresented in the datasets widely used in the community today [66]. Most of these datasets were designed for the standard i.i.d. setting, with training and test sets from the same distribution, and prior work on retrofitting them with distribution shifts has focused on shifts that are cleanly characterized but not always likely to arise in real world deployments. For instance, many recent papers have studied datasets with shifts induced by synthetic transformations, such as changing the color of MNIST digits [8], or by disparate data splits, such as generalizing from cartoons to photos [123]. Datasets like these are important testbeds for systematic studies; but to develop and evaluate methods for real world shifts, we need to complement them with datasets that capture shifts in the wild.

Here, we present WILDS, a curated benchmark of 10 datasets with evaluation metrics and train/test splits representing a broad array of shifts that models face in the wild (Figure 4.2). WILDS datasets span many important applications: animal species categorization [14], tumor identification [12], bioassay prediction [254, 96], genetic perturbation classification [222], wheat head detection [48], text toxicity classification [25], land use classification [38], poverty mapping [263], sentiment analysis [158], and code completion [175, 137]. These datasets reflect natural distribution shifts arising from different cameras, hospitals, molecular scaffolds, experiments, demographics, countries, time periods, users, and codebases.

WILDS builds on extensive data collection efforts by domain experts, who are often forced to grapple with distribution shifts to make progress in their applications, such as the ones listed above. To design WILDS, we worked with them to identify, select, and adapt datasets that fulfilled the following criteria:



	Domain generalization					Subpopulation shift	Domain generalization + subpopulation shift			
Dataset	iWildCam	Camelyon17	RxRx1	OGB-MolPCBA	GlobalWheat	CivilComments	FMoW	PovertyMap	Amazon	Py150
Input (x)	camera trap photo	tissue slide	cell image	molecular graph	wheat image	online comment	satellite image	satellite image	product review	code
Prediction (y)	animal species	tumor	perturbed gene	bioassays	wheat head bbox	toxicity	land use	asset wealth	sentiment	autocomplete
Domain (d)	camera	hospital	batch	scaffold	location, time	demographic	time, region	country, rural-urban	user	git repository
# domains	323	5	51	120,084	47	16	16 x 5	23 x 2	2,586	8,421
# examples	203,029	455,954	125,510	437,929	6,515	448,000	523,846	19,669	539,502	150,000

Train example						What do Black and LGBT people have to do with bicycle licensing?			Overall a solid package that has a good quality of construction for the price.	`import numpy as np`
										`…`
										`norm=np.___`

Test example						As a Christian, I will not be patronizing any of those businesses.			I *loved* my French press, it's so perfect and came with all this fun stuff!	`import subprocess as sp`
										`p=sp.Popen()`
										`stdout=p.___`

Adapted from	Beery et al. 2020	Bandi et al. 2018	Taylor et al. 2019	Hu et al. 2020	David et al. 2021	Borkan et al. 2019	Christie et al. 2018	Yeh et al. 2020	Ni et al. 2019	Raychev et al. 2016

Figure 4.2: The Wilds benchmark contains 10 datasets across a diverse set of application areas, data modalities, and dataset sizes. Each dataset comprises data from different domains, and the benchmark is set up to systematically evaluate models on distribution shifts across these domains.

1. **Distribution shifts with performance drops.** The train/test splits reflect shifts that substantially degrade model performance, i.e., with a large gap between in distribution and out of distribution performance.

2. **Real world relevance.** The training/test splits and evaluation metrics are motivated by real world scenarios and chosen in conjunction with domain experts.

3. **Potential leverage.** Distribution shift benchmarks must be nontrivial but also possible to solve, as models cannot be expected to generalize to arbitrary distribution shifts. We constructed each Wilds dataset to have training data from multiple domains, with domain annotations and other metadata available at training time. We hope that these can be used to learn robust models: e.g., for domain generalization, one could use these annotations to learn models that are invariant to domain specific features, while for subpopulation shift, one could learn models that perform uniformly well across each subpopulation.

We chose the Wilds datasets to collectively encompass a diverse set of tasks, data modalities, dataset sizes, and numbers of domains, so as to enable evaluation across a broad range of real world distribution shifts. To make the Wilds datasets more accessible, we have substantially modified most of them, e.g., to clarify the distribution shift, standardize the data splits, or preprocess the data for use in standard frameworks.

Datasets are significant catalysts for machine learning research. Likewise, benchmarks that curate and standardize datasets – e.g., the GLUE and SuperGLUE benchmarks for language understanding [241, 242] and the Open Graph Benchmark for graph machine learning [96] – can accelerate research by focusing community attention, easing development on multiple datasets, and enabling systematic comparisons between approaches. In this spirit, we hope that WILDS will facilitate the development of methods and models that are robust to real world distribution shifts and can therefore be deployed reliably in the wild.

WILDS problem settings

Each WILDS dataset is associated with a type of domain shift: domain generalization, subpopulation shift, or a hybrid of both (Figure 4.2). In each setting, we can view the overall data distribution as a mixture of D domains $\mathscr{D} = \{1, \dots, D\}$. Each domain $d \in \mathscr{D}$ corresponds to a fixed data distribution $p^{(d)}_{\mathbf{x}y}$ over $(\mathbf{x}, y, d)$, where all points sampled from $p^{(d)}_{\mathbf{x}y}$ have domain d. We encode the domain shift by assuming that the training distribution $p_{\mathbf{x}y} = \sum_{d \in \mathscr{D}} q_d \, p^{(d)}_{\mathbf{x}y}$ has mixture weights q_d for each domain d, while the test distribution $p^{\star}_{\mathbf{x}y} = \sum_{d \in \mathscr{D}} q^{\star}_d \, p^{(d)}_{\mathbf{x}y}$ is a different mixture of domains with weights $q^{\star}_d$. For convenience, we define the set of training domains as $\mathscr{D}^{\mathsf{train}} = \{d \in \mathscr{D} \mid q_d > 0\}$ and, likewise, the set of test domains as $\mathscr{D}^{\mathsf{test}} = \{d \in \mathscr{D} \mid q^{\star}_d > 0\}$.

At training time, the learning algorithm gets to see the domain annotations d, i.e., the training set comprises points $(x, y, d) \sim p_{\mathbf{x}y}$. At test time, the model gets either x or (x, d) drawn from $p^{\star}_{\mathbf{x}y}$, depending on the application.

Domain generalization (Figure 4.1-Top). In domain generalization, we aim to generalize to test domains $\mathscr{D}^{\mathsf{test}}$ that are disjoint from the training domains $\mathscr{D}^{\mathsf{train}}$, i.e., $\mathscr{D}^{\mathsf{train}} \cap \mathscr{D}^{\mathsf{test}} = \emptyset$. To make this problem tractable, the training and test domains are typically similar to each other: e.g., in CAMELYON17-WILDS, we train on data from some hospitals and test on a different hospital, and in IWILDCAM2020-WILDS, we train on data from some camera traps and test on different camera traps. We typically seek to minimize the average error on the test distribution.

Subpopulation shift (Figure 4.1-Bottom). In subpopulation shift, we aim to perform well across a wide range of domains seen during training time. Concretely, all test domains are seen at training, with $\mathscr{D}^{\mathsf{test}} \subseteq \mathscr{D}^{\mathsf{train}}$, but the proportions of the domains can change, with $q^{\star} \neq q$. We typically seek to minimize the maximum error over all test domains. For example, in CIVILCOMMENTS-WILDS, the domains d represent particular demographics, some of which are a minority in the training set, and we seek high accuracy on each of these subpopulations without observing their demographic identity d at test time.

Hybrid settings. It is not always possible to cleanly define a problem as domain generalization or subpopulation shift; for example, a test domain might be present in the training set but at a very low frequency. In Wilds, we also consider some hybrid settings that combine both problem settings. For example, in FMoW-wilds, the inputs are satellite images and the domains correspond to the year and geographical region in which they were taken. We simultaneously consider domain generalization across time (the training/test sets comprise images taken before/after a certain year) and subpopulation shift across regions (there are images from the same regions in the training and test sets, and we seek high performance across all regions).

WILDS datasets

We now briefly describe each Wilds dataset (Figure 4.2). For each dataset, we consider a problem setting – domain generalization, subpopulation shift, or a hybrid – that we believe best reflects the real world challenges in the corresponding application area. To avoid confusion between our modified datasets and their original sources, we append -wilds to the dataset names.

Domain generalization: iWildCam2020-wilds. Animal populations have declined 68% on average since 1970 [72]. To better understand and monitor wildlife biodiversity loss, ecologists commonly deploy camera traps – heat or motion activated static cameras placed in the wild [246] – and then learn models to process the data collected [247, 220, 16, 161, 2]. Typically, these models would be trained on photos from existing camera traps and then used across new camera trap deployments. However, across different camera traps, there is drastic variation in illumination, color, camera angle, background, vegetation, and relative animal frequencies, which results in models generalizing poorly to new camera trap deployments [15].

We study this shift on a variant of the iWildCam 2020 dataset [14], where the input $\mathbf{x}$ is a photo from a camera trap, the label y is one of 182 animal species, and the domain d specifies the identity of the camera trap. The training and test sets comprise photos from disjoint sets of camera traps. As leverage, we include over 200 camera traps in the training set, capturing a wide range of variation. We evaluate models by their macro F1 scores, which emphasizes performance on rare species, as rare and endangered species are the most important to accurately monitor.

Domain generalization: Camelyon17-wilds. Models for medical applications are often trained on data from a small number of hospitals, but with the goal of being deployed more generally across other hospitals. However, variations in data collection and processing can degrade model accuracy on data from new hospital deployments [266, 3]. In histopathology applications – studying tissue slides under a microscope – this variation can arise from, e.g., differences in the patient population or in slide staining and image acquisition [238, 114, 224].

We study this shift on a patch-based variant of the Camelyon17 dataset [12], where the input $\mathbf{x}$ is a 96x96 patch of a whole slide image of a lymph node section from a patient with potentially metastatic breast cancer, the label y is whether the patch contains tumor, and the domain d specifies which of 5 hospitals the patch was from. The training and test sets comprise class balanced patches from separate hospitals, and we evaluate models by their average accuracy. Prior work suggests that staining differences are the main source of variation between hospitals in similar datasets [224]. As we have training data from multiple hospitals, a model could use that as leverage to learn to be robust to stain variation.

Domain generalization: RxRx1-wilds. High throughput screening techniques that can generate large amounts of data are now common in many fields of biology, including transcriptomics [78], genomics [54, 272], proteomics and metabolomics [223], and drug discovery [28, 140, 219, 26]. Such large volumes of data, however, need to be created in experimental batches, or groups of experiments executed at similar times under similar conditions. Despite attempts to carefully control experimental variables such as temperature, humidity, and reagent concentration, measurements from these screens are confounded by technical artifacts that arise from differences in the execution of each batch. These *batch effects* make it difficult to draw conclusions from data across experimental batches [120, 165, 210, 162, 34].

We study the shift induced by batch effects on a variant of the RxRx1 dataset [222], where the input $\mathbf{x}$ is a 3-channel image of cells obtained by fluorescent microscopy [27], the label y indicates which of the 1,139 genetic treatments (including no treatment) the cells received, and the domain d specifies the batch in which the imaging experiment was run. The training and test sets consist of disjoint experimental batches; as leverage, the training set has images from 33 different batches, with each batch containing one sample for every class. We assess a model's ability to normalize batch effects while preserving biological signal by evaluating accuracy on images of treated cells in the out of distribution test set.

Domain generalization: OGB-MolPCBA. Accurate prediction of the biochemical properties of small molecules can significantly accelerate drug discovery by reducing the need for expensive lab experiments [205, 99]. However, the experimental data available for training such models is limited compared to the extremely diverse and combinatorially large universe of candidate molecules that we would want to make predictions on [24, 211, 139, 145]. This means that models need to generalize to out of distribution molecules that are structurally different from those seen in the training set.

We study this shift on the OGB-MolPCBA dataset, which is directly adopted from the Open Graph Benchmark [96] and originally from MoleculeNet [254]. It is a multi label classification dataset, where the input $\mathbf{x}$ is a molecular graph, the label y is a 128-dimensional binary vector where each component corresponds to a biochemical assay result, and the domain d specifies the scaffold (i.e., a cluster of molecules with similar structure). The training and test sets comprise molecules with disjoint scaffolds; for leverage, the training set has molecules from over 40,000 scaffolds. We evaluate models by averaging the Average Precision (AP) across each of the 128 assays.

Domain generalization: GlobalWheat-wilds. Models for automated, high throughput plant phenotyping – measuring the physical characteristics of plants and crops, such as wheat head density and counts – are important tools for crop breeding [226, 181] and agricultural field management [203]. These models are typically trained on data collected in a limited number of regions, even for crops grown worldwide such as wheat [142, 258, 235, 10]. However, there can be substantial variation between regions, due to differences in crop varieties, growing conditions, and data collection protocols. Prior work on wheat head detection has shown that this variation can significantly degrade model performance on regions unseen during training [48].

We study this shift in an expanded version of the Global Wheat Head Dataset [48], a large set of wheat images collected from 12 countries around the world. It is a detection dataset, where the input $\mathbf{x}$ is a cropped overhead image of a wheat field, the label y is the set of bounding boxes for each wheat head visible in the image, and the domain d specifies an image acquisition session (i.e., a specific location, time, and sensor with which a set of images was collected). The data split captures location shift, with training and test sets comprising images from disjoint countries. As leverage, we include images from 18 acquisition sessions over 5 countries in the training set. We evaluate model performance on unseen countries by measuring accuracy at a fixed Intersection over Union (IoU) threshold, and averaging across acquisition sessions to account for imbalances in the numbers of images in them.

Subpopulation shift: CivilComments-wilds. Automatic review of user generated text is an important tool for moderating the sheer volume of text written on the Internet. We focus here on the task of detecting toxic comments. Prior work has shown that toxicity classifiers can pick up on biases in the training data and spuriously associate toxicity with the mention of certain demographics [164, 50]. These types of spurious correlations can significantly degrade model performance on particular subpopulations [192].

We study this problem on a variant of the CivilComments dataset [25], a large collection of comments on online articles taken from the Civil Comments platform. The input $\mathbf{x}$ is a text comment, the label y is whether the comment was rated as toxic, and the domain d is a 8-dimensional binary vector where each component corresponds to whether the comment mentions one of the 8 demographic identities *male*, *female*, *LGBTQ*, *Christian*, *Muslim*, *other religions*, *Black*, and *White*. The training and test sets comprise comments on disjoint articles, and we evaluate models by the lowest true positive/negative rate over each of these 8 demographic groups; these groups overlap with each other, deviating slightly from the standard subpopulation shift framework detailed above. Models can use the provided domain annotations as leverage to learn to perform well over each demographic group.

Hybrid: FMoW-wilds. Models for satellite imagery can enable global scale monitoring of sustainability and economic challenges, aiding policy and humanitarian efforts in applications such as deforestation tracking [77], population density mapping [228], crop yield prediction [245], and other economic tracking applications. As satellite data constantly change due to human activity and environmental processes, these models must be robust to distribution shifts over time. Moreover, as there can be disparities in the data available between regions, these models should ideally have uniformly high accuracies instead of only doing well on data rich regions and countries.

We study this problem on a variant of the Functional Map of the World dataset [38], where the input $\mathbf{x}$ is an RGB satellite image, the label y is one of 62 building or land use categories, and the domain d represents the year the image was taken and its geographical region (Africa, the Americas, Oceania, Asia, or Europe). The different regions have different numbers of examples, e.g., there are far fewer images from Africa than the Americas. The training set comprises data from before 2013, while the test set comprises data from 2016 and after; years 2013 to 2015 are reserved for the validation set. We evaluate models by their test accuracy on the worst geographical region, which combines both a domain generalization problem over time and a subpopulation shift problem over regions. As we provide both time and region annotations, models can leverage the structure across both space and time to improve robustness.

Hybrid: PovertyMap-wilds. Global scale poverty estimation is a specific remote sensing application which is essential for targeted humanitarian efforts in poor regions [1, 56]. However, ground truth measurements of poverty are lacking for much of the developing world, as field surveys for collecting the ground truth are expensive [23]. This motivates the approach of training models on countries with ground truth labels and then deploying them on different countries where we have satellite data but no labels [256, 105, 263].

We study this shift through a variant of the poverty mapping dataset [263], where the input $\mathbf{x}$ is a multispectral satellite image, the output y is a real valued asset wealth index from surveys, and the domain d represents the country the image was taken in and whether the image is of an urban or rural area. The training and test set comprise data from disjoint sets of countries, and we evaluate models by the correlation of their predictions with the ground truth. Specifically, we take the lower of the correlations over the urban and rural subpopulations, as prior work has shown that accurately predicting poverty within these subpopulations is especially challenging. As poverty measures are highly correlated across space [104, 186], methods can utilize the provided location coordinates, and the country and urban/rural annotations, to improve robustness.

Hybrid: Amazon-wilds. In many consumer facing applications, models are trained on data collected on one set of users and then deployed across a wide range of potentially new users. These models can perform well on average but poorly on some users [221, 35, 127, 111]. These large performance disparities across users are practical concerns in consumer facing applications, and they can also indicate that models are exploiting biases or spurious correlations in the data [11, 67].

We study a variant of the Amazon review dataset [158], where the input $\mathbf{x}$ is the review text, the label y is the corresponding 1 to 5 star rating, and the domain d identifies the user who wrote the review. The training and test sets comprise reviews from disjoint sets of users; for leverage, the training set has reviews from 5,008 different users. As our goal is to train models with consistently high performance across users, we evaluate models by the 10th percentile of per user accuracies.

Hybrid: Py150-wilds. Code completion models – autocomplete tools used by programmers to suggest subsequent source code tokens, such as the names of API calls – are commonly used to reduce the effort of software development [185, 30, 157, 170, 60]. These models are typically trained on data collected from existing codebases but then deployed more generally across other codebases, which may have different distributions of API usages [160, 171, 5]. This shift across codebases can cause substantial performance drops in code completion models. Moreover, prior studies of real world usage of code completion models have noted that they can generalize poorly on some important subpopulations of tokens such as method names [83].

We study a variant of the Py150 Dataset [175, 137], where the goal is to predict the next token given the context of previous tokens. The input $\mathbf{x}$ is a sequence of source code tokens, the label y is the next token, and the domain d specifies the repository that the source code belongs to. The training and test sets comprise code from disjoint GitHub repositories. As leverage, we include over 5,300 repositories in the training set, capturing a wide range of source code variation. We evaluate models by their accuracy on the subpopulation of class and method tokens.

WILDS experimental results

Many algorithms have been proposed for training models that are more robust to particular distribution shifts than standard ERM models. Unlike ERM, these algorithms tend to utilize domain annotations during training, with the goal of learning a model that can generalize across domains. In this section, we evaluate several representative algorithms from prior work and show that out of distribution (OOD) performance drops still remain.

Domain generalization baselines. Methods for domain generalization typically involve adding a penalty to the ERM objective that encourages some form of invariance across domains. We include two such methods as representatives:

- **CORAL** [213], which penalizes differences in the means and covariances of the feature distributions (i.e., the distribution of last layer activations in a neural network) for each domain. Conceptually, CORAL is similar to other methods that encourage feature representations to have the same distribution across domains [234, 135, 62, 130, 126].

Dataset	Setting	ERM	CORAL	IRM	DRNN
iWildCam2020-wilds	Domain gen.	31.0 (1.3)	**32.8 (0.1)**	15.1 (4.9)	23.9 (2.1)
Camelyon17-wilds	Domain gen.	**70.3 (6.4)**	59.5 (7.7)	64.2 (8.1)	68.4 (7.3)
RxRx1-wilds	Domain gen.	**29.9 (0.4)**	28.4 (0.3)	8.2 (1.1)	23.0 (0.3)
OGB-MolPCBA	Domain gen.	**27.2 (0.3)**	17.9 (0.5)	15.6 (0.3)	22.4 (0.6)
GlobalWheat-wilds	Domain gen.	**49.2 (1.5)**	—	—	46.1 (1.6)
CivilComments-wilds	Subpop. shift	56.0 (3.6)	65.6 (1.3)	66.3 (2.1)	**70.0 (2.0)**
FMoW-wilds	Hybrid	**32.3 (1.3)**	31.7 (1.2)	30.0 (1.4)	30.8 (0.8)
PovertyMap-wilds	Hybrid	**0.45 (0.06)**	0.44 (0.06)	0.43 (0.07)	0.39 (0.06)
Amazon-wilds	Hybrid	**53.8 (0.8)**	52.9 (0.8)	52.4 (0.8)	53.3 (0.0)
Py150-wilds	Hybrid	**67.9 (0.1)**	65.9 (0.1)	64.3 (0.2)	65.9 (0.1)

Table 4.1: The out of distribution (OOD) test performance of models trained with different baseline algorithms: CORAL, originally designed for unsupervised domain adaptation; IRM, for domain generalization; and DRNN, for subpopulation shifts. Higher is better for all evaluation metrics. Overall, these algorithms failed to improve over ERM, except on CivilComments-wilds where they perform better but still do not close the in distribution gap. For GlobalWheat-wilds, we omit CORAL and IRM as those methods do not port straightforwardly to detection settings. Parentheses show standard deviation across 3+ replicates.

- **IRM** [8], which penalizes feature distributions that have different optimal linear classifiers for each domain. This builds on earlier work on invariant predictors [169].

Subpopulation shift baselines. In subpopulation shift settings, our aim is to train models that perform well on all relevant subpopulations. We test the following approach:

- **DRNN** [95, 192], which uses distributionally robust optimization (DRO) to explicitly minimize the loss on the worst case domain during training. DRNN builds on a maximin approach developed in earlier work [146].

Setup. We trained ERM, CORAL, IRM, and DRNN models on each dataset. While DRNN was originally developed for subpopulation shifts, for completeness, we also experiment with using it for domain generalization. In that setting, DRNN models aim to achieve similar performance across domains: e.g., in Camelyon17-wilds, where the domains are hospitals, DRNN optimizes for the training hospital with the highest loss. Similarly, we also test CORAL and IRM on subpopulation shifts, where they encourage models to learn invariant representations across subpopulations. We used an OOD validation set for early stopping and to tune the penalty weights for the CORAL and IRM algorithms. More experimental details are provided in Section A.1.

Results. Table 4.1 shows that models trained with CORAL, IRM, and DRNN generally fail to improve over models trained with ERM. The exception is the CivilComments-wilds subpopulation shift dataset, where the worst performing subpopulation is a minority domain. By upweighting the minority domain, DRNN obtains an OOD accuracy of 70.0% on the worst performing subpopulation compared to 56.0% for ERM, though this is still substantially below the ERM model's average accuracy of 92.2% over the entire test set. CORAL and IRM also perform well on CivilComments-wilds, though their gains stem largely from how our implementation heuristically upsamples the minority domain. All other datasets involve domain generalization; the failures here are consistent with other recent findings on standard domain generalization datasets [74].

These results indicate that training models to be robust to distribution shifts in the wild remains a significant open challenge. However, we are optimistic about future progress for two reasons. First, current methods were mostly designed for other problem settings besides domain generalization, e.g., CORAL for unsupervised domain adaptation and DRNN for subpopulation shifts. Second, compared to existing distribution shift datasets, the Wilds datasets generally contain diverse training data from many more domains as well as metadata on these domains, which future algorithms might be able to leverage.

Summary

We presented Wilds, a benchmark of distribution shift problems designed with real world applications in mind, spanning areas including ecology, economics, medicine, social issues, and more. Wilds datasets are set up as domain generalization and subpopulation shift problems, and these formulations make assumptions about the presence of domain annotations within the training data which is reasonable and realistic in many cases. We evaluated a number of prior methods that were developed for these and related problem formulations and found that ERM oftentimes performs better than these methods for the proposed datasets. Nevertheless, we believe that introducing these datasets will encourage the development of more powerful methods that better leverage the assumptions that fit real world applications, in order to improve past standard ERM. In Chapter 5, we will revisit some of these datasets, and in the rest of this chapter, we will introduce another suite of problems focused on adaptation that will also be used in Chapter 5.

4.3 Distribution Shift Problems for Adaptation

In this section, we propose a set of problems in which it is both feasible and helpful (and perhaps even necessary) to assume access to a batch or stream of inputs at test time. As a running example which we concretely instantiate later in this section, consider a handwriting classification model that, after training on data from past users, is deployed to new end users. Each new user represents a new test distribution that differs from the training distribution. Thus, each test setting involves dealing with shift. In Figure 4.3, we visualize a batch of

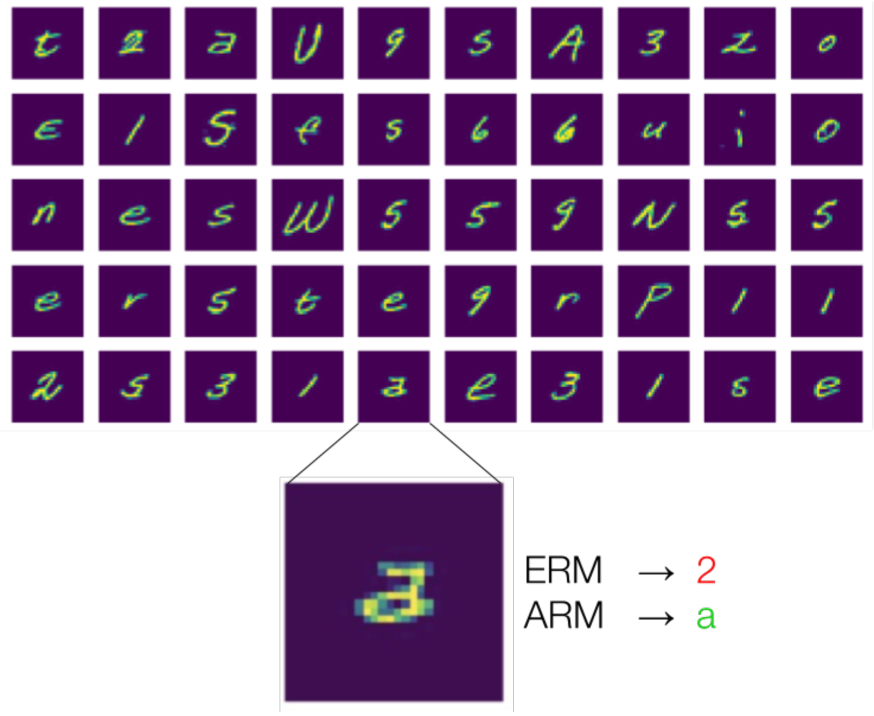


Figure 4.3: An example of ambiguous data points in handwriting classification, described in detail later in this section.

50 examples from a test user, and we highlight an ambiguous example which may be either a “2” (written with a loop) or an “a” (in the double-storey style) depending on the user’s handwriting. Due to the biases in the training data, an ERM trained model incorrectly classifies this example as “2”. However, we can see that the batch of images from this test user contains other examples of “2” (written without loops) and “a” (also double-storey) from this user. Can we somehow leverage this unlabeled data to better handle test shifts caused by new users?

Leveraging this test assumption does not require labels for any test data and is feasible in many practical setups. For handwriting classification, we do not access only single handwritten characters from an end user, but rather collections of characters such as sentences or paragraphs. Unlabeled adaptation has been shown empirically to be useful for distribution shift problems [216, 197, 243], such as for dealing with image corruptions [84]. Taking inspiration from these findings, we propose a number of problems which, similar to Wilds, can be categorized as domain generalization and subpopulation shift problems. Thus, in addition to Wilds problems, we also evaluate on the problems in this section in Chapter 5, and our empirical results indicate that adaptation is beneficial in dealing with these instances of distribution shift.

Description of distribution shift problems

We propose four image classification problems which we believe can supplement existing benchmarks for domain generalization and subpopulation shift.

Rotated MNIST. We study a modified version of MNIST where images are rotated in 10 degree increments, from 0 to 130 degrees. We treat each rotation as a separate domain, and all domains appear in the training and test data, which makes this a subpopulation shift problem. We construct a training set of 32292 data points using 90% of the original training set – separating out a validation set – by sampling and applying random rotations to each image. The rotations are not dependent on the image or label, but certain rotations are sampled much less frequently. Rotations of 0 through 20 degrees, inclusive, have 7560 data points each, 30 through 50 degrees have 2160 points each, 60 through 80 have 648, 90 through 110 have 324 each, and 120 to 130 have only 108 points each.

In this setting, we hypothesize that adaptation can specialize the model to specific domains, in particular the *rare domains* in the training set. For each test evaluation, we generate images from the MNIST test set with a certain rotation. We measure both worst case and average accuracy across domains.

Federated Extended MNIST (FEMNIST). The extended MNIST (EMNIST) dataset consists of images of handwritten uppercase and lowercase letters, in addition to digits [41]. FEMNIST is the same dataset, but it also provides the metadata of which user generated each data point [35]. FEMNIST, and EMNIST in general, is a significantly more challenging dataset compared to MNIST due to its larger label space (62 compared to 10 classes), label imbalance (almost half of the data points are digits), and inherent ambiguities (e.g., lowercase versus uppercase "o") [41]. In processing the dataset, we filter out users with fewer than 100 examples, leaving 262, 50, and 35 unique users and a total of 62732, 8484, and 8439 data points in the training, validation, and test splits, respectively. The smallest users contain 104, 119, and 140 data points, respectively.

We treat each user as a domain. We measure each method's worst case and average accuracy across 35 test users, which are held out and thus *disjoint* from the training users. Thus, this problem is a domain generalization problem. As discussed above, adaptation may help for this problem for specializing the model and resolving ambiguous data points.

Corrupted image datasets. CIFAR-10-C and Tiny ImageNet-C [84] augment the CIFAR-10 [115] and Tiny ImageNet test sets with common image corruptions that vary in type and severity. The original goal of these augmented test sets was to benchmark how well methods could handle these corruptions without access to *any* corruptions during training [84]. Thus, successful methods for these problems typically have relied on domain knowledge and heuristics designed specifically for image classification. For example, prior work has shown that carefully designed test time adaptation procedures are effective for these problems [216, 197, 243]. One possible reason for this phenomenon is that convolutional networks are biased toward texture [65], which is distorted by corruptions, thus adaptation can help the model recover its performance for each corruption type.

We study whether these test sets can be extended into domain generalization problems, to test whether training on various corruptions can improve performance on novel corruptions at test time. To do so, we modify the original dataset construction protocol [84]. We construct training, validation, and test splits with 56, 17, and 22 domains, respectively, where we define domains based on the type and severity of the corruption. We split domains such that corruptions in the training, validation, and test sets are disjoint. Specifically, the training set consists of Gaussian noise, shot noise, defocus blur, glass blur, zoom blur, snow, frost, brightness, contrast, and pixelate corruptions of all severity levels. Similarly, the validation set consists of speckle noise, Gaussian blur, and saturate corruptions, and the test set consists of impulse noise, motion blur, fog, and elastic transform corruptions of all severity levels. For two corruptions, spatter and JPEG compression, we include lower severities (1-3) in the training set and higher severities (4-5) in the validation and test sets. In this way, we are constructing a more challenging test setting, in which the test domains are not sampled identically as the training domains, since the corruption types are largely different between the two sets. For the training and validation sets, each domain consists of 1000 images for CIFAR-10-C and 2000 images for Tiny ImageNet-C, giving training sets of size 56000 and 112000, respectively. We use the full test set of 10000 images for each domain, giving a total of 220000 test images for both datasets.

In Chapter 5, we evaluate on these problems as well as WILDS problems to study whether, and when, adaptation at test time leads to improve model performance when faced with distribution shift.

Part II

Adapting to Distribution Shift via Meta-Learning and Data Augmentations

Chapter 5

Adaptive Risk Minimization: Learning to Adapt to Domain Shift

In Part I, we provided examples in RL and supervised learning for distribution shift problems that are important to solve and representative of real world applications. In this chapter and Chapter 5, we turn our focus to approaches for solving the supervised learning problems, with an emphasis on paradigms and methods that use *adaptation* as the main tool for combating shift. As discussed in the last section of Chapter 4, one way in which we can formulate the problem setting for test time adaptation, similar to prior works [216, 197, 243], is to assume access to a batch of stream of unlabeled test inputs. From the perspective of domain generalization and subpopulation shift problems, we can imagine that observing multiple inputs may provide greater information about the underlying data distribution, which may aid in prediction. In this chapter, we expand on this intuition to develop the adaptive risk minimization paradigm for tackling these problems through adaptation, and we evaluate on the specific problems proposed in Chapter 4.

5.1 Introduction

Prior benchmarks for domain generalization and similar settings typically center around *invariances* – i.e., in these benchmarks, there is a consistent input-output relationship across all domains, and the goal is to learn this relationship while ignoring the spurious correlations within the domains (see, e.g., [74]). Thus, prior methods aim for generalization to shifts by discovering this relationship, through techniques such as robust optimization and learning an invariant feature space [126, 8, 192]. These methods are appealing in that they make minimal assumptions about the information provided at test time – in particular, they do not require test labels, and the learned model can be immediately applied to predict on a single point. Nevertheless, these methods also have limitations, such as in dealing with problems where the input-output relationship varies across domains, e.g., the handwriting classification example from Chapter 4.

Our main contribution in this chapter is to introduce the framework of adaptive risk minimization (ARM), which proposes the following objective: optimize the model such that it can maximally leverage the unlabeled adaptation phase to handle domain shift. To do so, we instantiate a set of methods that, given a set of training domains, *meta-learns* a model that is adaptable to these domains. These methods are straightforward extensions of existing meta-learning approaches, thereby demonstrating that tools from the meta-learning toolkit can be readily adapted to tackle domain shift. Our experiments test on several image classification problems, derived from benchmarks for federated learning [35] and image classifier robustness [84], in which training and test domains share structure that can be leveraged for improved performance. We also evaluate on the Wilds suite of distribution shift problems [113], which have been curated to faithfully represent important real world problems. Empirically, we demonstrate that the proposed ARM methods, by leveraging meta-training and test time adaptation, are often able to outperform prior state-of-the-art methods by 1-4% test accuracy.

5.2 Related Work

A number of prior works have studied distribution shift in various forms [172]. In this section, we review prior work in domain generalization, group robustness, meta-learning, and adaptation.

Invariance and robustness to domains. As discussed above, a number of frameworks leverage training domains to address test time shift. The terminology in prior work is scattered and, depending on the application, includes terms such as “groups”, “datasets”, “subpopulations”, and “users”; in this chapter, we adopt the term “domains” which we believe is an appropriate unifying term. A number of testbeds for this problem setting have been proposed for image classification, including generalizing to new datasets [57], new image types [123, 167], and underrepresented demographics [192].

Prior benchmarks typically assume the existence of a consistent input-output relationship across domains that is learnable by the specified model, thus motivating methods such as learning an invariant feature space [126, 129, 8] or optimizing for worst case group performance [95, 192]. In particular, methods for domain generalization – sometimes referred to as multi-source domain adaptation [214] or zero shot domain adaptation [261] – have largely focused on learning invariant features [62, 213, 126, 129, 167]. DomainBed [74] provides a comprehensive survey of domain generalization benchmarks and finds that, surprisingly, ERM is competitive with the state of the art across all the benchmarks considered. In Section B.1, we discuss this finding as well as the performance of an ARM method on this benchmark suite. In contrast to these benchmarks, we identify different problems for which adaptation is helpful, and we find that, on these problems, ARM methods consistently outperform ERM and other non adaptive methods for robustness and invariance.

Meta-learning. Meta-learning [196, 19, 227, 91] has been most extensively studied in the context of few shot *labeled* adaptation [194, 239, 174, 59, 209]. Our aim is not to address few-shot recognition problems, nor to propose a novel meta-learning algorithm, but rather to extend meta-learning paradigms to problems requiring unlabeled adaptation, with the primary goal of tackling distribution shift. This aim differs from previous work in meta-learning for domain generalization [124, 52], which seek to meta-train models for non adaptive generalization performance. We demonstrate how paradigms such as contextual meta-learning [63, 180] can be readily extended using the ARM framework.

Some other meta-learning methods adapt using both labeled and unlabeled data, either in the semi supervised learning setting [178, 271, 128] or the transductive learning setting [159, 134, 6, 94]. These works all assume access to labeled data for adaptation, whereas we propose methods and problems for purely unlabeled adaptation. Prior methods in meta-learning for unlabeled adaptation include DAML [265], which adapts a policy to imitate human demonstrations in the context of robotic learning; meta-learning unsupervised update rules [147], which meta-learns an update rule for unsupervised representation learning but still requires labels to learn a predictive model; and Tailoring [4], which meta-learns adaptive models based on task specific unsupervised objectives. Unlike these prior works, we propose a general framework for tackling distribution shift problems by meta-learning unsupervised adaptation strategies. This framework simplifies the extension of meta-learning paradigms to these problems, encapsulates previous approaches such as DAML [265], and sheds light on how to improve existing strategies such as adaptation via batch normalization [131].

Adaptation to shift. Unlabeled adaptation has primarily been studied separately from meta-learning. Domain adaptation is a prominent framework that assumes access to test examples at *training* time [43, 249], similar to transductive learning [236]. As such, most domain adaptation methods consider the problem setting where there is a single test distribution [204, 47, 68, 62, 234, 37], and some of these methods are difficult to apply to problems where there are multiple test distributions. Certain domain adaptation methods have also been applied in the domain generalization setting, such as methods for learning invariant features [62, 213, 126], and we compare to these methods in our experiments.

Some prior works have also proposed adaptive methods for domain generalization [151, 117]. Other prior work has provided a theoretic study of domain generalization and established favorable generalization bounds for models that can adapt to domain shift at test time [21, 22]. We summarize some of these results in the next section. In comparison, our work establishes a framework that makes explicit the connection between adaptation to domain shift and meta-learning, allowing us to devise new methods in a straightforward and principled manner. These methods are amenable to expressive models such as deep neural networks, which enables us to propose and evaluate on problems with raw image observations.

Test time adaptation has also been studied for dealing with label shift [188, 133, 212] and crafting favorable inductive biases for the domain of interest. For image classification, techniques such as normalizing via the test inputs [131] and optimizing self-supervised surrogate losses [216] have proven effective for adapting to image corruptions [84]. We compare to these prior methods in our experiments and empirically demonstrate the advantage of using training domains to learn how to adapt.

5.3 Preliminaries and Notation

Let $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ represent the input and output, respectively. We can formalize the domain generalization problem setting using the following data generation process [22]: first, a joint data distribution $p_{\mathbf{x}y}^{(d)}$ is sampled from a set of distributions $\mathcal{P}_{\mathbf{x}y}$, and then some data points are sampled from $p_{\mathbf{x}y}^{(d)}$.[1] Overloading terminology, we refer to each $p_{\mathbf{x}y}^{(d)}$ as a domain, e.g., a particular dataset or user, thus $\mathcal{P}_{\mathbf{x}y}$ represents the set of all possible domains. We assume that the training dataset is composed of data from S runs of this generative process, organized by domain. An equivalent characterization which we will use for clarity is that, within the training set, there are S domains, and each data point $(\mathbf{x}^{(i)}, y^{(i)})$ is annotated with a domain label $d^{(i)}$. Each $d^{(i)}$ is an integer that takes on a value between 1 and S, indicating which $p_{\mathbf{x}y}^{(d)}$ generated the i-th training point (though, of course, we do not have access to, or knowledge of, $p_{\mathbf{x}y}^{(d)}$ itself). At test time, there may be multiple evaluation settings, where each setting is considered separately and contains only *unlabeled* data sampled via a new run of the same generative process. This data may represent, e.g., a new dataset or user, and the test domains are likely to be distinct from the training domains when $|\mathcal{P}_{\mathbf{x}y}|$ is large or infinite.

Our formal goal is to optimize for expected performance, e.g., classification accuracy, at test time. To do so, let us first consider predictive models of the form $f : \mathcal{X} \times \mathcal{P}_{\mathbf{x}} \to \mathcal{Y}$, where the model f takes in not just an input $\mathbf{x}$ but also the marginal input distribution $p_{\mathbf{x}}^{(d)} \in \mathcal{P}_{\mathbf{x}}$ that $\mathbf{x}$ was sampled from. We refer to f as an *adaptive* model, as it has the opportunity to use $p_{\mathbf{x}}^{(d)}$ to adapt its predictions on $\mathbf{x}$. The underlying assumption is that $p_{\mathbf{x}}^{(d)}$ provides information about $p_{y|\mathbf{x}}^{(d)}$, i.e., $p_{\mathbf{x}}^{(d)}$ is used as a surrogate input in place of $p_{\mathbf{x}y}^{(d)}$. In the worst case, if $p_{\mathbf{x}}^{(d)}$ and $p_{y|\mathbf{x}}^{(d)}$ are sampled independently, then the model does not benefit at all from knowing $p_{\mathbf{x}}^{(d)}$. In many problems, however, we expect knowledge about $p_{\mathbf{x}}^{(d)}$ to be useful, e.g., for resolving ambiguity as in the handwriting classification example in Chapter 4.

Theoretically, when $p_{\mathbf{x}}^{(d)}$ provides information about $p_{y|\mathbf{x}}^{(d)}$, and when training and test domains are drawn from the same distribution over $\mathcal{P}_{\mathbf{x}y}$, we can establish favorable generalization bounds for the expected performance of f in adapting to domain shift at test time. We can formalize this as follows. First, define a *prediction model* to be a non adaptive model of the form $g : \mathcal{X} \to \mathcal{Y}$, and define the risk for a prediction model g and loss function ℓ, under a data distribution $p_{\mathbf{x}y}^{(d)}$, as

$$\mathcal{R}(g, p_{\mathbf{x}y}^{(d)}) \triangleq \mathbb{E}_{p_{\mathbf{x}y}^{(d)}} \left[\ell(g(\mathbf{x}), y) \right] .$$

[1] Formally, the number of points sampled is another random variable with support over the positive integers.

Further, define the Bayes optimal risk for ℓ under $p^{(d)}_{\mathbf{x}y}$ as

$$\mathcal{R}^{\star}(p^{(d)}_{\mathbf{x}y}) \triangleq \min_{g} \mathcal{R}(g, p^{(d)}_{\mathbf{x}y}) \,.$$

Let μ denote the distribution on $\mathcal{P}_{\mathbf{x}y}$ from which training and test domains $p^{(d)}_{\mathbf{x}y}$ are sampled. To avoid overlapping terms, define the *adaptive risk* for an adaptive model f and ℓ, under μ, to be

$$\mathcal{E}(f, \mu) \triangleq \mathbb{E}_{\mu}\left[\mathbb{E}_{p^{(d)}_{\mathbf{x}y}}\left[\ell(f(\mathbf{x}, p^{(d)}_{\mathbf{x}}), y)\right]\right] \,. \tag{5.1}$$

We state the following result from prior work [22], which details a condition on μ under which $\mathcal{E}$ is a strongly principled objective for learning adaptive models.

Lemma 9 from [22]. Let $f^{\star}$ denote a minimizer of $\mathcal{E}$ for the given μ. If μ is a distribution on $\mathcal{P}_{\mathbf{x}y}$ such that μ-almost surely it holds that $p^{(d)}_{y|\mathbf{x}} = M(p^{(d)}_{\mathbf{x}})$ for some deterministic mapping M, then for μ-almost all $p^{(d)}_{\mathbf{x}y}$, we have

$$\mathcal{R}(f^{\star}(\cdot, p^{(d)}_{\mathbf{x}}), p^{(d)}_{\mathbf{x}y}) = \mathcal{R}^{\star}(p^{(d)}_{\mathbf{x}y}) \implies \mathcal{E}(f^{\star}, \mu) = \mathbb{E}_{\mu}\left[\mathcal{R}^{\star}(p^{(d)}_{\mathbf{x}y})\right] \,.$$

In other words, an adaptive model which minimizes the adaptive risk $\mathcal{E}$ coincides with a Bayes optimal decision function for $p^{(d)}_{\mathbf{x}y}$, *for μ-almost all domains $p^{(d)}_{\mathbf{x}y}$.*

Remark. The required condition on μ – that $p^{(d)}_{y|\mathbf{x}}$ is determined by $p^{(d)}_{\mathbf{x}}$ – holds if, and only if, an expert (or oracle) is able to correctly label inputs from a given domain provided only information about the input distribution. This condition holds for all of the testbeds detailed in Chapter 4. The condition does not hold for, e.g., standard few shot learning testbeds, where it is possible for two domains with identical input distributions to shuffle their label orderings differently [239]. Thus, these problems are outside the scope of this chapter.

This result provides strong justification for learning adaptive models f by minimizing the adaptive risk $\mathcal{E}$. However, a practical instantiation of this approach requires some approximations. First, we do not know and cannot input $p^{(d)}_{\mathbf{x}}$ to f in most cases. Instead, we instantiate f such that it takes in a batch of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_K$, all from the same domain, where K can vary. f makes predictions on the whole batch, which also serves as an empirical approximation (i.e., a histogram) $\hat{p}^{(d)}_{\mathbf{x}}$ of $p^{(d)}_{\mathbf{x}}$ [22]. In our exposition, we will assume that a batch of unlabeled points is available at test time for adaptation. However, we also experiment with the *streaming* setting where the test inputs are observed one at a time and adaptation occurs incrementally.

Notice that, if we instead passed in an approximation $\hat{p}^{(d)}_{\mathbf{x}y}$ of $p^{(d)}_{\mathbf{x}y}$ to the model, such as a batch of *labeled* data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$, then this setup would resemble the standard few shot meta-learning problem [239]. Formally, a meta-learning model takes in both an input $\mathbf{x}$ and $\hat{p}^{(d)}_{\mathbf{x}y}$, which approximates the distribution that $\mathbf{x}$ was sampled from and thus can be used to adapt the prediction on $\mathbf{x}$. Compared to our problem setting, the meta-learning formalism can tackle a wider range of problems but also requires more restrictive assumptions, specifically, labels at test time via $\hat{p}_{\mathbf{x}y}^{(d)}$. *Transductive meta-learning* methods further assume that, in addition to $\hat{p}_{\mathbf{x}y}^{(d)}$, a full batch of inputs $\mathbf{x}_1, \dots, \mathbf{x}_K$ is passed into the model, which allows for better estimation of the input distribution $p_{\mathbf{x}}^{(d)}$ [159, 134, 94]. The model then makes predictions on this entire batch. In meta-learning terminology, $\hat{p}_{\mathbf{x}y}^{(d)}$ and $\mathbf{x}_1, \dots, \mathbf{x}_K$ are often referred to as the *support* and *query*, respectively. Therefore, another interpretation of the adaptive models that we study in this chapter is that they resemble transductive meta-learning models, but they are given only the unlabeled query and not the labeled support set.

In the next section, we expand on this connection to develop the ARM framework, which then allows us to bring forward tools from meta-learning to tackle problems in domain generalization and subpopulation shift, which we will collectively refer to as "domain shift" problems.

5.4 Adaptive Risk Minimization

In this section, we formally describe the ARM framework, which defines an objective for training adaptive models to tackle domain shift. Furthermore, we propose a general meta-learning algorithm as well as specific methods for optimizing the ARM objective. In the next section, we test these ARM methods on problems for which unlabeled adaptation can potentially be leveraged for better test performance.

Devising the ARM objective

We wish to learn an adaptive model $f : \mathcal{X}^K \to \mathcal{Y}^K$ to tackle domain shift. As noted, meta-learning methods for labeled adaptation study a similar form of model, and a common approach in many of these methods is to define f such that it is composed of two parts: first, a *learner* which ingests the data and produces parameters, and second, a *prediction model* which uses these parameters to make predictions [239, 59]. We will follow a similar strategy which will allow us to easily extend and design meta-learning methods towards our goal.

In particular, we will decompose the model f into two modules: a standard prediction model $g(\cdot\,;\theta) : \mathcal{X} \to \mathcal{Y}$, that is parameterized by $\theta \in \Theta$ and predicts y given $\mathbf{x}$, and an *adaptation model* $h(\cdot\,,\cdot\,;\phi) : \Theta \times \mathcal{X}^K \to \Theta$, which is parameterized by ϕ. h takes in the prediction model parameters θ and K unlabeled data points and uses the K points to produce adapted parameters θ'. This is analogous to the learner in meta-learning, however, h adapts the model parameters using only unlabeled data.[2]

[2] For some meta-learning methods, the learner does not take as input the unadapted model parameters [239], and we also devise some methods of this form. In the formalism above, these methods simply ignore the input θ.

Algorithm 1 Meta-Learning for ARM

```
// Training procedure
Require: # training steps $T$, batch size $K$, learning rate $\eta$
1: Initialize: $\theta, \phi$
2: for $t = 1, \ldots, T$ do
3:     Sample $d$ uniformly from training domains
4:     Sample $(\mathbf{x}_k, y_k) \sim \hat{p}_{\mathbf{x}y}^{(d)}$ for $k = 1, \ldots, K$
5:     $\theta' \leftarrow h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi)$
6:     $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{(\theta,\phi)} \sum_{k=1}^{K} \ell(g(\mathbf{x}_k; \theta'), y_k)$
7: end for

// Test time adaptation procedure
Require: $\theta$, $\phi$, test batch $\mathbf{x}_1, \ldots, \mathbf{x}_K$
7: $\theta' \leftarrow h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi)$
8: $\hat{y}_k \leftarrow g(\mathbf{x}_k; \theta')$ for $k = 1, \ldots, K$
```

The ARM objective is to optimize ϕ and θ such that h can adapt g using unlabeled data sampled according to a particular domain d. This can be expressed as the optimization problem

$$\min_{\theta,\phi} \hat{\mathcal{E}}(\theta, \phi) = \mathbb{E}_{p_d} \left[\mathbb{E}_{\hat{p}_{\mathbf{x}y}^{(d)}} \left[\frac{1}{K} \sum_{k=1}^{K} \ell(g(\mathbf{x}_k; \theta'), y_k) \right] \right] , \text{ where } \theta' = h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi) . \quad (5.2)$$

Note that $\hat{\mathcal{E}}$ is the empirical form of the adaptive risk in Equation 5.1 for the form of f we have defined. Mimicking the generative process that we assume generated the training data, p_d is a categorical distribution over $\{1, \ldots, S\}$ which places uniform probability mass on each training domain, and $\hat{p}_{\mathbf{x}y}^{(d)}$ assigns uniform probability to only the training points within a particular domain. As we have established theoretically, we expect the trained models to perform well at test time if the test domains are sampled independently and identically – i.e., from the same distribution over $\mathcal{P}_{\mathbf{x}y}$ – as the training domains. In practice, similar to how meta-learned few shot classification models are evaluated on new and unseen meta-test classes [239, 59], we empirically show in our experiments that the trained models can generalize to test domains that are not sampled identically to the training domains.

Optimizing the ARM objective

Algorithm 1 presents a general meta-learning approach for optimizing the ARM objective. As described above, h outputs updated parameters θ' using an unlabeled batch of data (line 5). This mimics the adaptation procedure at test time, where we do not assume access to labels (lines 7-8). However, the training update itself does rely on the labels (line 6). We assume that h is differentiable with respect to its input θ and ϕ, thus we use gradient updates on both θ and ϕ to optimize for *post adaptation* performance on a mini batch of data sampled according to a particular domain d. In practice, we also sample mini batches of domains, rather than just one domain (as written in line 3), to provide a better gradient signal for optimizing ϕ and θ.

Together, Equation 5.2 and Algorithm 1 shed light on a number of ways to devise methods for solving the ARM problem. First, we can extend meta-learning paradigms to the ARM problem setting, and any paradigm in which the adaptation model h can be augmented to operate on unlabeled data is readily applicable. As an example, we propose the ARM-CML method, which is inspired by recent works in contextual meta-learning (CML) [63, 180]. Second, we can enhance prior unlabeled adaptation methods by incorporating a meta-training phase that allows the model to better leverage the adaptation. To this end, we propose the ARM-BN method, based on the general approach of adapting using batch normalization (BN) statistics of the test inputs [131, 197, 107, 152]. Third, we can incorporate existing methods for meta-learning unlabeled adaptation to solve domain shift problems. We demonstrate this by proposing the ARM-LL method, which is based on the robotic imitation learning method from prior work which adapts via a learned loss (LL) [265]. All of these methods are straightforward extensions of existing meta-learning and adaptation methods, and this is intentional – we aim to show how existing tools can be readily adapted to tackle domain generalization problems.

ARM-CML. In ARM-CML, the parameters ϕ of h define the weights of a *context network* $\mathrm{f}_{\mathrm{cont}}(\cdot\,; \phi) : \mathcal{X} \to \mathbb{R}^D$, parameterized by the adaptation model parameters ϕ. We also instantiate the model with a *prediction network* $\mathrm{f}_{\mathrm{pred}}(\cdot\,, \cdot\,; \theta) : \mathcal{X} \times \mathbb{R}^D \to \mathcal{Y}$, parameterized by θ. When given a mini batch of inputs, $\mathrm{f}_{\mathrm{cont}}$ processes each example $\mathbf{x}_k$ in the mini batch separately and outputs $\mathbf{c}_k \in \mathbb{R}^D$ for $k = 1, \ldots, K$, which are averaged together into a *context* $\mathbf{c} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{c}_k$. D is a hyperparameter, and in our experiments, we choose D to be the dimensionality of $\mathbf{x}$, such that we can concatenate each image $\mathbf{x}_k$ and the context $\mathbf{c}$ along the channel dimension to produce the input to $\mathrm{f}_{\mathrm{pred}}$. In other words, $\mathrm{f}_{\mathrm{pred}}$ processes each $\mathbf{x}_k$ separately to produce an estimate of the output $\hat{y}_k$, but it additionally receives $\mathbf{c}$ as input. In this way, $\mathrm{f}_{\mathrm{cont}}$ can provide information about the entire batch of K unlabeled data points to $\mathrm{f}_{\mathrm{pred}}$ for predicting the correct outputs.

Note that the difference between ARM-CML and prior contextual meta-learning approaches is that, in prior approaches, the context network processes both inputs and outputs to produce each $\mathbf{c}_k$. ARM-CML is designed for the domain generalization setting in which we do not assume access to labels at test time, thus we meta-train for unlabeled adaptation performance at training time.

ARM-BN. ARM-BN is a particularly simple method that is applicable for any model g that has BN layers [100]. Practically, training g via ARM-BN follows the same protocol as standard BN [100] except for two key differences: first, the training batches are sampled from a single domain, rather than from the entire dataset, and second, the normalization

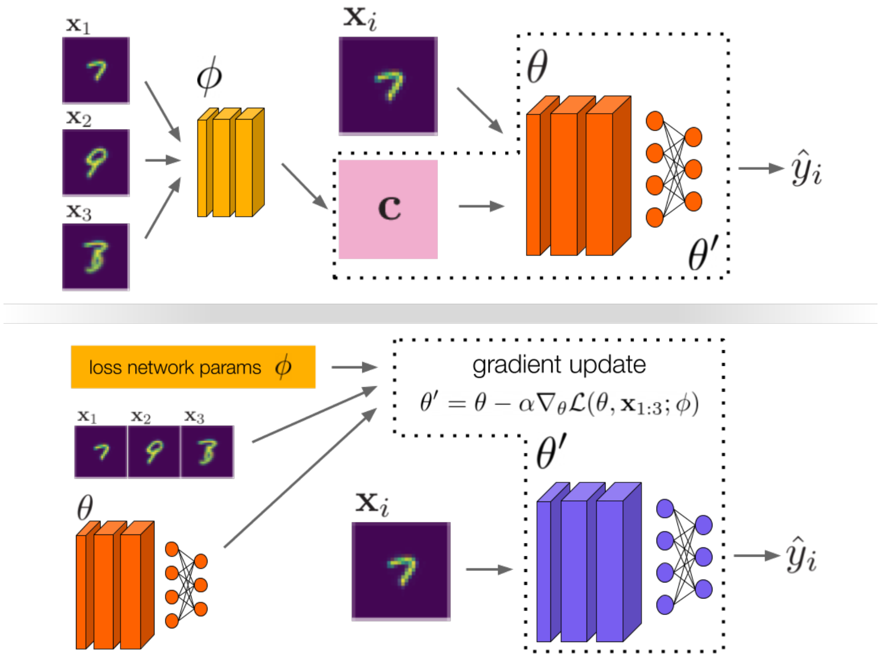


Figure 5.1: In the contextual approach (top), $\mathbf{x}_1, \ldots, \mathbf{x}_K$ are summarized into a context $\mathbf{c}$, and we propose two methods for this summarization, either through a separate context network or using batch normalization activations in the model itself. $\mathbf{c}$ can then be used by the model to infer additional information about the input distribution. In the gradient based approach (bottom), an unlabeled loss function $\mathcal{L}$ is used for gradient updates to the model parameters, in order to produce parameters that are specialized to the test inputs and can produce more accurate predictions.

statistics are recomputed at test time rather than using a training running average. As noted, this second difference has been explored by several works as a method for test time adaptation, but the first difference is novel to ARM-BN. Following Algorithm 1, ARM-BN defines a meta-training procedure in which g learns to adapt – i.e., compute normalization statistics – using batches of training points sampled from the same domain. We empirically show in our experiments that, for problems where BN adaptation already has a favorable inductive bias, such as for image classification, further introducing meta-training boosts its performance. We believe that other test time adaptation methods, such as those based on optimizing surrogate losses [216, 243], may similarly benefit from their corresponding meta-training procedures.

At a high level, ARM-BN operates in a similar fashion to ARM-CML, thus we group these methods together into the umbrella of contextual approaches, shown in Figure 5.1 (top). The interpretation of ARM-BN through the contextual approach is that h replaces the running statistics used by standard BN with statistics computed on the batch of inputs, which then serves as the context $\mathbf{c}$. Thus, for ARM-BN, there is no context network, and h has no parameters beyond the model parameters θ involved in computing BN statistics. The model g is again specified via a prediction network $\mathrm{f}_{\mathrm{pred}}$, which must have BN layers. BN typically tracks a running average of the first and second moments of the activations in these layers, which are then used at test time. ARM-BN defines h such that it swaps out these moments for the moments computed via the activations on the test batch, thus giving us adapted parameters θ' if we view the moments as part of the model parameters. This method is remarkably simple, and in deep learning libraries such as PyTorch [166], implementing ARM-BN involves changing a single line of code. However, as shown by our experiments, this method also performs very well empirically, and the adaptation effectiveness is further boosted by meta-training.

ARM-LL. ARM-LL, depicted in Figure 5.1 (bottom), follows the gradient based meta-learning paradigm [59] and learns parameters θ that are amenable to gradient updates on a loss function in order to quickly adapt to a new problem. In other words, h produces $\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi)$, where α is a hyperparameter. Note that the loss function $\mathcal{L}$ used in the gradient updates is different from the original supervised loss function ℓ, in that it operates on only the inputs $\mathbf{x}$, rather than the input output pairs that ℓ receives. We follow the general implementation of this approach proposed in DAML [265]. We define g to produce output features $\mathbf{o} \in \mathbb{R}^{|\mathcal{Y}|}$ that are used as logits when making predictions. We then define the unlabeled loss function $\mathcal{L}$ to be the composition of g and a *loss network* $\mathrm{f}_{\mathrm{loss}}(\cdot\,; \phi) : \mathbb{R}^{|\mathcal{Y}|} \to \mathbb{R}$, which takes in the output features from g and produces a scalar. As we desire that the loss is bounded below, we use the ℓ_2-norm of these scalars across the batch of inputs as the loss for updating θ. In other words,

$$h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi) = \theta - \alpha \nabla_\theta \|\mathbf{v}\|_2 \,, \text{ where } \mathbf{v} = [\mathrm{f}_{\mathrm{loss}}(g(\mathbf{x}_1; \theta); \phi), \ldots, \mathrm{f}_{\mathrm{loss}}(g(\mathbf{x}_K; \theta); \phi)] \,.$$

In our experiments, we used $\alpha = 0.1$ and 1 gradient step for both meta-training and meta-testing.

5.5 Experiments

Our experiments are designed to answer the following questions:

1. Do ARM methods learn models that can better adapt to tackle domain shift?

2. How do ARM methods compare to methods for robustness, invariance, and adaptation?

3. Can models trained via ARM methods adapt successfully in the streaming test setting?

We evaluate on the problems we proposed in Chapter 4. A key characteristic of these problems is the potential for adaptation to improve test performance, and this differs from prior benchmarks such as the problems compiled by DomainBed [74]. In Section B.1, we compare our testbeds to DomainBed and group robustness benchmarks, and we briefly discuss the results in DomainBed, which also evaluate ARM-CML.

Comparisons and ablations

We compare the ARM methods against several prior methods designed for robustness, invariance, and adaptation. We describe the comparisons here and provide additional details in Section B.2.

Test time adaptation. We evaluate the general approach of using test batches to compute BN statistics [131, 197, 107, 152], which we term BN adaptation. We also compare to test time training (TTT) [216], which adapts the model at test time using a self-supervised rotation prediction loss. These methods have previously achieved strong results for image classification, likely because they constitute favorable inductive biases for improving on the true classification task [216].

Ablations. We also include ablations of the ARM-CML and ARM-LL methods, which sample training batches of unlabeled examples uniformly from the entire training set, rather than sampling from a single domain.[3] These "context ablation" and "learned loss ablation" are similar to test time adaptation methods in that they do not require training domains, thus they allow us to evaluate whether or not meta-training on domain shifts is important for improved performance.

Group robustness and invariance. We compare to DRNN [192], as well as a strong upweighting (UW) baseline evaluated in this work that samples uniformly from each group, and so we also evaluate this approach in our experiments. Additionally, we compare to domain adversarial neural networks (DANN) [62] and maximum mean discrepancy (MMD) feature learning [126], two state-of-the-art methods for adversarial learning of invariant predictive features. For the Wilds datasets, we include the numbers reported in Chapter 4 for DRNN and two other invariance methods, correlation alignment (CORAL) [213] and invariant risk minimization (IRM) [8].

Robustness and invariance methods assume access to training domains but not test batches, whereas adaptation methods assume the opposite. Thus, at a high level, we can view the comparisons to these methods as evaluating the importance of each of these assumptions for the specified problems.

Quantitative evaluation and comparisons

The results for the four proposed benchmarks for adaptation are presented in Table 5.1. The best results, stratified by classes of methods, are bolded. Across all of these problems, ARM

[3] Note that the corresponding ablation of ARM-BN is simply the BN adaptation method.

	MNIST		**FEMNIST**		**CIFAR-10-C**		**Tiny ImageNet-C**	
Method	**WC**	**Avg**	**WC**	**Avg**	**WC**	**Avg**	**WC**	**Avg**
ERM	74.5 ± 1.4	93.6 ± 0.4	62.4 ± 0.4	79.1 ± 0.3	54.1 ± 0.3	70.4 ± 0.1	20.3 ± 0.5	41.9 ± 0.1
UW*	**80.3 ± 1.2**	**95.1 ± 0.1**	**65.7 ± 0.7**	80.3 ± 0.6	—	—	—	—
DRNN	79.9 ± 0.7	**94.9 ± 0.1**	57.5 ± 1.7	76.5 ± 1.2	49.3 ± 0.9	65.7 ± 0.5	14.2 ± 0.2	31.6 ± 1.0
DANN	78.8 ± 0.8	**94.9 ± 0.1**	**65.4 ± 1.0**	**81.7 ± 0.3**	**53.9 ± 2.2**	**69.8 ± 0.3**	**20.4 ± 0.7**	**40.9 ± 0.2**
MMD	**82.4 ± 0.9**	**95.3 ± 0.3**	62.4 ± 0.7	79.8 ± 0.4	**52.2 ± 0.3**	**69.5 ± 0.1**	**19.7 ± 0.2**	40.1 ± 0.1
BN adaptation	78.0 ± 0.3	94.4 ± 0.1	65.7 ± 1.5	80.0 ± 0.5	60.6 ± 0.3	70.9 ± 0.1	26.5 ± 0.3	**42.8 ± 0.0**
TTT	**81.1 ± 0.3**	**95.4 ± 0.1**	**68.6 ± 0.4**	**84.2 ± 0.1**	**61.5 ± 0.3**	**71.7 ± 0.5**	**27.6 ± 0.5**	37.7 ± 0.3
CML ablation	63.5 ± 1.8	90.1 ± 0.2	61.8 ± 0.8	81.6 ± 0.5	58.8 ± 0.1	69.6 ± 0.2	26.3 ± 0.6	42.5 ± 0.1
LL ablation	**79.9 ± 1.1**	**95.0 ± 0.3**	64.1 ± 1.6	80.8 ± 0.2	**60.9 ± 0.4**	**71.3 ± 0.0**	21.6 ± 2.1	32.6 ± 3.2
ARM-CML	**88.0 ± 0.8**	**96.3 ± 0.4**	**70.9 ± 1.4**	**86.4 ± 0.3**	**61.2 ± 0.4**	70.3 ± 0.2	**29.1 ± 0.4**	**43.3 ± 0.1**
ARM-BN	83.3 ± 0.5	95.6 ± 0.1	64.5 ± 3.2	83.2 ± 0.5	**61.7 ± 0.3**	**72.4 ± 0.3**	**28.3 ± 0.3**	**43.3 ± 0.1**
ARM-LL	**88.9 ± 0.8**	**96.9 ± 0.2**	67.0 ± 0.9	84.3 ± 0.7	**61.2 ± 0.7**	**72.5 ± 0.4**	25.4 ± 0.1	35.7 ± 0.4

Table 5.1: Worst case (WC) and average (Avg) top 1 accuracy on all testbeds, where means and standard errors are reported across three separate runs of each method. Horizontal lines separate methods that make use of (from top to bottom): neither, training domains, test batches, or both. ARM methods consistently achieve greater robustness, measured by WC, and Avg performance compared to prior methods. *UW is identical to ERM for CIFAR-10-C and Tiny ImageNet-C.

methods increase both worst case and average accuracy compared to all other methods. ARM-CML performs well across all tasks, and despite its simplicity, ARM-BN achieves the best performance overall on the corrupted image testbeds, demonstrating the effectiveness of meta-training on top of an already strong adaptation procedure. BN adaptation and TTT are the strongest prior methods, as these adaptation procedures constitute inductive biases that are generally well suited for image classification. However, ARM methods are comparatively less reliant on favorable inductive biases and consistently attain better results. In general, we observe poor performance from robustness methods, varying performance from invariance methods, strong performance from adaptation methods, and the strongest performance from ARM methods.

When we cannot access a batch of test points all at once, and instead the points are observed in a streaming fashion, we can augment the proposed ARM methods to perform sequential model updates. For example, ARM-CML and ARM-BN can update their average context and normalization statistics, respectively, after observing each new test point. In Figure 5.2, we study this test setting for the Tiny ImageNet-C problem. We see that both models trained with ARM-CML and ARM-BN are able to achieve near their original worst case and average accuracy within observing 50 data points, well before the training batch size of 100. This result demonstrates that ARM methods are applicable for problems where test points must be observed one at a time, provided that the model is permitted to adapt using each point. We provide streaming results on rotated MNIST in Section B.4.

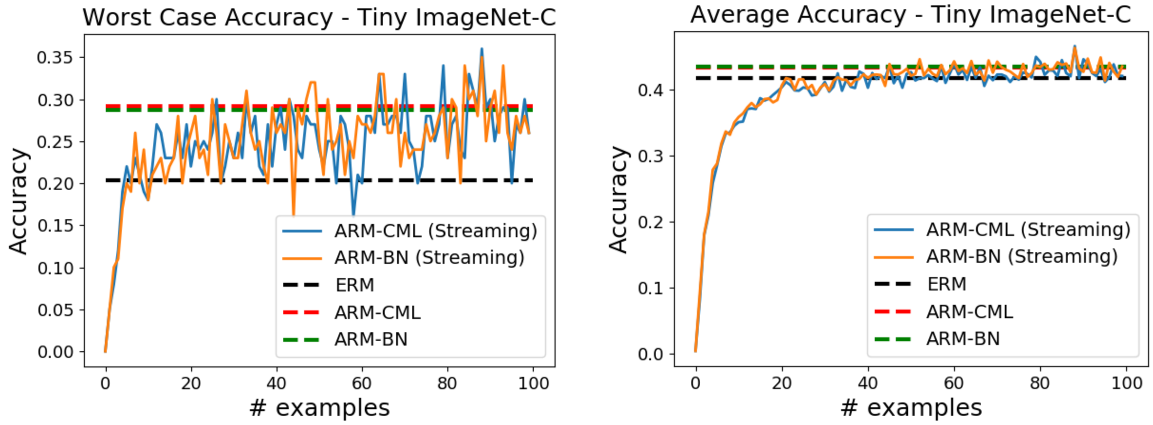


Figure 5.2: In the streaming setting, ARM methods reach strong performance on Tiny ImageNet-C after fewer than 50 data points, despite using training batch sizes of 100. This highlights that the trained models are able to adapt successfully in the standard streaming evaluation setting.

	iWildCam		**Camelyon17**	**RxRx1**	**FMoW**		**PovertyMap**	
Method	**Acc**	**Macro F1**	**Acc**	**Acc**	**WC Acc**	**Avg Acc**	**WC Pearson r**	**Pearson r**
ERM	**71.6 ± 2.5**	31.0 ± 1.3	70.3 ± 6.4	29.9 ± 0.4	**32.3 ± 1.25**	**53.0 ± 0.55**	0.45 ± 0.06	0.78 ± 0.04
DRNN	**72.7 ± 2.0**	23.9 ± 2.1	68.4 ± 7.3	23.0 ± 0.3	**30.8 ± 0.81**	**52.1 ± 0.5**	0.39 ± 0.06	0.75 ± 0.07
CORAL	**73.3 ± 4.3**	**32.8 ± 0.1**	59.5 ± 7.7	28.4 ± 0.3	**31.7 ± 1.24**	50.5 ± 0.36	0.44 ± 0.06	0.78 ± 0.05
IRM	59.8 ± 3.7	15.1 ± 4.9	64.2 ± 8.1	8.2 ± 1.1	**30.0 ± 1.37**	50.8 ± 0.13	0.43 ± 0.07	0.77 ± 0.05
BN adaptation	46.4 ± 1.0	13.8 ± 0.3	**88.6 ± 1.4**	20.0 ± 0.2	30.2 ± 0.26	51.6 ± 0.16	0.39 ± 0.17	0.82 ± 0.06
ARM-BN	**70.3 ± 2.4**	23.2 ± 2.7	**87.2 ± 0.9**	**31.2 ± 0.1**	24.6 ± 0.04	42.0 ± 0.21	0.49 ± 0.21	0.84 ± 0.05

Table 5.2: Results on the Wilds image testbeds. Different methods are best suited for different problems, motivating the need for a wide range of methods. ARM-BN struggles on FMoW but performs well on the other datasets, in particular RxRx1.

Wilds results

Finally, we present results on the Wilds benchmark [113] in Table 5.2. We evaluate BN adaptation and ARM-BN on these testbeds. We see that, on these real world distribution shift problems, different methods perform well for different problems. CORAL, a method for invariance [213], performs best on the iWildCam animal classification problem [14], whereas no methods outperform ERM by a significant margin on the FMoW [38] or PovertyMap [263] satellite imagery problems. ARM-BN performs particularly poorly on the FMoW problem. However, it performs well on PovertyMap and significantly improves performance on the RxRx1 [222] problem of treatment classification from medical images. On the other medical imagery problem of Camelyon17 [12] tumor identification, adaptation in general boosts performance dramatically. These results indicate the need to consider a wide range of tools, including meta-learning and adaptation, for combating distribution shift.

5.6 Discussion

We presented adaptive risk minimization (ARM), a framework and problem formulation for learning models that can adapt in the face of domain shift at test time using only a batch of unlabeled test examples. We devised an algorithm and instantiated a set of methods for optimizing the ARM objective that meta-learns models that are adaptable to different domains of training data. Empirically, we observed that ARM methods consistently improve performance in terms of both average and worst case metrics, as compared to a number of prior approaches for handling domain shift.

Though we provided contextual meta-learning as a concrete example, a number of other meta-learning paradigms would also be interesting to extend to the ARM setting. For example, few shot generative modeling objectives would be a natural fit for unlabeled adaptation [55, 90, 252]. Another exciting direction for future work is to explore the problem setting where domains are not provided at training time. As discussed in Section B.4, in this setting, we can instead construct domains via unsupervised learning techniques. One promising approach is to generate a diverse set of domains in order to learn generally effective adaptation strategies [93]. Robustness and invariance methods cannot be used easily with multiple different groupings, learned or otherwise, as techniques such as group weighted loss functions [192] and domain classifiers [62] are not immediately extendable to this setup. Thus, ARM methods may be uniquely suited to be paired with domain learning.

Chapter 6

MEMO: Test Time Robustness via Adaptation and Augmentation

At the end of Chapter 5, we discussed possible ways in which a model could learn to adapt to distribution shift without access to training domains. In this chapter, we take the question one step further: what can we do if we do not even have access to the model training procedure itself? This is the case for the setting of test time robustness, in which a pretrained model may leverage only the current test point that it must predict on in order to try and improve its prediction. We will see in this chapter that methods based on adaptation via entropy minimization, but modified to be more effective for single test inputs, and data augmentations produce strong inductive biases for improving test time robustness.

As the methods we study in this chapter make virtually no assumptions about the model training procedure, the information available at training time (e.g., domains), or the availability of multiple test points, we are free to choose from a much broader pool of test sets to evaluate on. In order to provide a thorough analysis of the most commonly studied state-of-the-art models, we will primarily focus on challenging test sets for ImageNet classification, namely, ImageNet-C [84], ImageNet-R [89], and ImageNet-A [86].

6.1 Introduction

Deep neural network models have achieved excellent performance on many machine learning problems, such as image classification, but are often brittle and susceptible to issues stemming from *distribution shift*. For example, deep image classifiers may degrade precipitously in accuracy when encountering input perturbations, such as noise or changes in lighting [84] or domain shifts which occur naturally in real world applications [113]. Therefore, robustification of deep models against these test shifts is an important and active area of study.

Most prior works in this area have focused on techniques for training time robustification, including utilizing larger models and datasets [163], various forms of adversarial training [192, 250], and aggressive data augmentation [264, 85, 122, 89]. Employing these techniques requires modifying the training process, which may not be feasible if, e.g., it involves heavy computation or non public data. Furthermore, these techniques do not rely on any information about the test points that the model must predict on, even though these test points may provide significant information for improving model robustness. Recently, several works have proposed methods for improving accuracy via *adaptation* after seeing the test data, typically by updating a subset of the model's weights [216, 101], normalization statistics [197], or both [243, 268]. Though effective at handling test shifts, these methods sometimes still require specialized training procedures, and they typically rely on extracting distributional information via batches or even entire sets of test inputs, thus introducing additional assumptions.

We are interested in studying and devising methods for improving model robustness that are "plug and play", i.e., they can be readily used with a wide variety of pretrained models and test settings. Such methods may simply constitute a different, more robust way to perform inference with preexisting models, under virtually the same assumptions as standard test time inference. In this work, we focus on methods for *test time robustness*, in which the specific test input may be leveraged in order to improve the model's prediction on that point. Though broad applicability is our primary goal, we also want methods that synergize with other robustification techniques, in order to achieve greater performance than using either set of techniques in isolation. To satisfy both of these desiderata, we devise a novel test time robustness method based on adaptation and augmentation. As illustrated in Figure 6.1, when presented with a test point, we propose to adapt the model by augmenting the test point in different ways and ensuring that the model makes the same predictions across these augmentations, thus respecting the invariances encoded in the data augmentations. We further encourage the model to make confident predictions, thus arriving at the proposed method: minimize the *marginal entropy* of the model's predictions across the augmented versions of the test point.

We refer to the proposed method as **m**arginal **e**ntropy **m**inimization with **o**ne test point (MEMO), and this is the primary contribution of our work. MEMO makes direct use of pretrained models without any assumptions about their particular training procedure or architecture, while requiring only a single test input for adaptation. In our experiments, we demonstrate empirically that MEMO consistently improves the performance of ResNet [80] and vision transformer [51] models on several challenging ImageNet distribution shift benchmarks, achieving several new state-of-the-art results for these models in the setting in which only one test point is available. In particular, MEMO consistently outperforms non adaptive marginal distribution predictions (between 1-10% improvement) on corruption and rendition shifts – tested by the ImageNet-C [84] and ImageNet-R [89] datasets, respectively – indicating that adaptation plays a crucial role in improving predictive accuracy. MEMO encourages both invariance across augmentations and confident predictions, and an ablation study in our experiments shows that both components are important for maximal performance gains. Also, MEMO is, to the best of our knowledge, the first adaptation method to improve performance (by 1-4% over standard model evaluation) on ImageNet-A [86], demonstrating that MEMO is more broadly applicable on a wide range of distribution shifts.

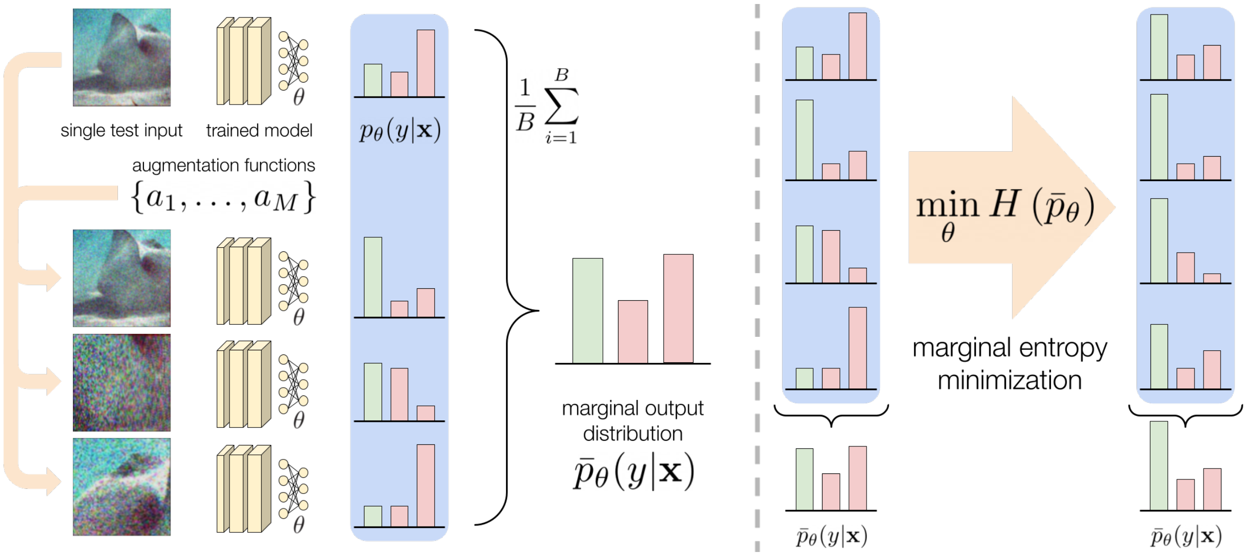

single test input
trained model
θ
$p_\theta(y|\mathbf{x})$
$\frac{1}{B}\sum_{i=1}^{B}$
augmentation functions
$\{a_1, \ldots, a_M\}$
marginal output distribution
$\bar{p}_\theta(y|\mathbf{x})$
$\min_\theta H(\bar{p}_\theta)$
marginal entropy minimization
$\bar{p}_\theta(y|\mathbf{x})$
$\bar{p}_\theta(y|\mathbf{x})$

Figure 6.1: A schematic of our overall approach. Left: at test time, we have a trained model that outputs a probabilistic predictive distribution and has adaptable parameters θ, a single test input $\mathbf{x}$, and a set of data augmentation functions $\{a_1, \ldots, a_M\}$. Note that we do not assume access to the model training procedure or multiple test inputs for adaptation. We perform different augmentations to $\mathbf{x}$ and pass these augmented inputs to the model in order to estimate the marginal output distribution averaged over augmentations. Right: rather than using this distribution to make the final prediction, we instead perform a gradient update on the model to minimize the entropy of this marginal distribution, thus encouraging the model predictions to be invariant across different augmentations while maintaining confident predictions. The final prediction is then made on the original data point, i.e., the predictive distribution in the top right of the schematic.

6.2 Related Work

The general problem of distribution shift has been studied under a number of frameworks [172], including domain adaptation [204, 43, 249], domain generalization [21, 151, 74], and distributionally robust optimization [18, 95, 192], to name just a few. These frameworks typically leverage additional training or test assumptions in order to make the distribution shift problem more tractable. Largely separate from these frameworks, various empirical methods have also been proposed for dealing with shift, such as increasing the model and training dataset size or using heavy training augmentations [163, 264, 89]. The focus of this work is complementary to these efforts: the proposed MEMO method is applicable to a wide range of pretrained models, including those trained via robustness methods, and can achieve further performance gains via test time adaptation.

Prior test time adaptation methods generally either make significant training or test time assumptions. Some methods update the model using batches or even entire datasets of test inputs, such as by computing batch normalization (BN) statistics on the test set [131, 107, 152, 197], or minimizing the (conditional) entropy of model predictions across a batch of test data [243]. The latter approach is closely related to MEMO. The differences are that MEMO minimizes *marginal* entropy using single test points and data augmentation and adapts all of the model parameters rather than just those associated with normalization layers, thus not requiring multiple test points or specific model architectures. Other test time adaptation methods can be applied to single test points but require specific training procedures or models [216, 98, 197]. Test time training (TTT) [216] requires a specialized model with a rotation prediction head, as well as a different procedure for training this model. Prior work has also shown that BN adaptation can be effective even with only one test point [197], and we refer to this approach as "single point" BN adaptation. As we discuss in the next section, MEMO synergizes well with single point BN adaptation.

A number of works have noted that varying forms of strong data augmentation on the training set can improve the resulting model's robustness [264, 85, 122, 89]. Data augmentations are also sometimes used on the test data directly by averaging the model's outputs across augmented copies of the test point [116, 206], i.e., predicting according to the model's marginal output distribution. This technique, which we refer to as test time augmentation (TTA), has been shown to be useful both for improving model accuracy and calibration [9] as well as handling distribution shift [148]. We take this idea one step further by explicitly adapting the model such that its marginal output distribution has low entropy. This extracts an additional learning signal for improving the model, and furthermore, the adapted model can then make its final prediction on the clean test point rather than the augmented copies. We empirically show in our experiments that these differences lead to improved performance over this non adaptive TTA baseline.

6.3 Robustness via Adaptation and Augmentation

Data augmentations are typically used to train the model to respect certain invariances – e.g., changes in lighting or viewpoint do not change the underlying class label – but, especially when faced with distribution shift, the model is not guaranteed to obey the same invariances at test time. In this section, we introduce MEMO, a method for test time robustness that adapts the model such that it respects these invariances on the test input. We use "test time robustness" specifically to refer to techniques that operate directly on pretrained models and single test inputs – single point BN adaptation and TTA, as described in the previous section, are examples of prior test time robustness methods.

In the test time robustness setting, we are given a trained model $g_\theta : \mathcal{X} \to \mathcal{Y}$ with parameters $\theta \in \Theta$. We do not require any special training procedure and do not make any assumptions about the model, except that θ is adaptable and that g_θ produces a conditional output distribution $p_\theta(y|\mathbf{x})$ that is differentiable with respect to θ.[1] All standard deep

[1] Single point BN adaptation also assumes that the model has batch normalization layers, and, as shown

neural network models satisfy these assumptions. A single point $\mathbf{x} \in \mathcal{X}$ is presented to g_θ, for which it must predict a label $\hat{y} \in \mathcal{Y}$ immediately. Note that this is precisely identical to the standard test time inference procedure for regular supervised learning models – in effect, we are simply modifying how inference is done, without any additional assumptions on the training process or on test time data availability. This makes test time robustness methods a simple "slot-in" replacement for the ubiquitous and standard test time inference process. We assume sampling access to a set of augmentation functions $\mathcal{A} \triangleq \{a_1, \dots, a_M\}$ that can be applied to the test point $\mathbf{x}$. We use these augmentations and the self-supervised objective detailed below to adapt the model before it predicts on $\mathbf{x}$. When given a set of test inputs, the model adapts and predicts on each test point independently. We do not assume access to any ground truth labels.

Marginal entropy minimization with one test point

Given a test point $\mathbf{x}$ and set of augmentation functions $\mathcal{A}$, we sample B augmentations from $\mathcal{A}$ and apply them to $\mathbf{x}$ in order to produce a batch of augmented data $\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_B$. The model's average, or *marginal*, output distribution with respect to the augmented points is given by

$$\bar{p}_\theta(y|\mathbf{x}) \triangleq \mathbb{E}_{\mathcal{U}(\mathcal{A})}\left[p_\theta(y|a(\mathbf{x}))\right] \approx \frac{1}{B}\sum_{i=1}^{B} p_\theta(y|\tilde{\mathbf{x}}_i)\,, \tag{6.1}$$

where the expectation is with respect to uniformly sampled augmentations $a \sim \mathcal{U}(\mathcal{A})$.

What properties do we desire from this marginal distribution? To answer this question, consider the role that data augmentation typically serves during training. For each training point $(\mathbf{x}^{\text{train}}, y^{\text{train}})$, the model g_θ is trained using multiple augmented forms of the input $\tilde{\mathbf{x}}_1^{\text{train}}, \dots, \tilde{\mathbf{x}}_E^{\text{train}}$. g is trained to obey the invariances between the augmentations and the label – no matter the augmentation on $\mathbf{x}^{\text{train}}$, g should predict the same label y^{train}, and it should do so confidently. We seek to devise a similar learning signal during test time, when no ground truth labels are available. That is, after adapting:

1. the model g_θ predictions should be invariant across augmentations of the test point, and
2. the model g_θ should be confident in its predictions, even for heavy augmentations of the test point, due to the knowledge that augmentations do not change the underlying label.

Optimizing the model for more confident predictions can be justified from the assumption that the true underlying decision boundaries between classes lie in low density regions of the data space [71]. With these two goals in mind, we propose to adapt the model using the entropy of its marginal output distribution over augmentations (Equation 6.1), i.e.,

$$\ell(\theta; \mathbf{x}) \triangleq H\left(\bar{p}_\theta(\cdot|\mathbf{x})\right) = -\sum_{y \in \mathcal{Y}} \bar{p}_\theta(y|\mathbf{x}) \log \bar{p}_\theta(y|\mathbf{x})\,. \tag{6.2}$$

empirically in our experiments, this is an assumption that we do not require but can also benefit from.

Algorithm 2 Test time robustness via MEMO

```
Require: trained model $g_\theta$, test point $\mathbf{x}$, # augs $B$, learning rate $\eta$, update rule $G$
1: Sample $a_1, \dots, a_B \overset{\text{i.i.d.}}{\sim} \mathcal{U}(\mathcal{A})$ and get augmented points $\tilde{\mathbf{x}}_i = a_i(\mathbf{x})$ for $i \in \{1, \dots, B\}$
2: Compute Monte Carlo estimate $\tilde{p} = \frac{1}{B} \sum_{i=1}^{B} p_\theta(y|\tilde{\mathbf{x}}_i) \approx \bar{p}_\theta(y|\mathbf{x})$ and $\tilde{\ell} = H(\tilde{p}) \approx \ell(\theta; \mathbf{x})$
3: Adapt model parameters via update rule $\theta' \leftarrow G(\theta, \eta, \tilde{\ell})$
4: Predict $\hat{y} \triangleq \arg\max_y p_{\theta'}(y|\mathbf{x})$
```

Optimizing this objective encourages both confidence and invariance to augmentations, since the entropy of $\bar{p}_\theta(\cdot|\mathbf{x})$ is minimized when the model outputs the same (confident) prediction regardless of the augmentation. We can directly use gradient based optimization to adapt θ according to this objective. We use only one gradient step per test point, because empirically we found this to be sufficient for improved performance while being more computationally efficient. After this step, we can use the adapted model, which we denote $g_{\theta'}$, to predict on the original test input $\mathbf{x}$.

Algorithm 2 presents the overall method MEMO for test time adaptation. Though prior test time adaptation methods must carefully choose which parameters to adapt in order to avoid degenerate solutions [243], our adaptation procedure simply adapts all of the model's parameters θ (line 3). As discussed above, the model g_θ adapts using augmented data but makes its final prediction on the original point (line 4), which may be easier to predict on.

Composing MEMO with prior methods

An additional benefit of MEMO is that it synergizes with other approaches for handling distribution shift. In particular, MEMO can be composed with prior methods for training robust models and adapting model statistics, thus leveraging the performance improvements of each technique.

Pretrained robust models. Since MEMO makes no assumptions about, or modifications to, the model training procedure, performing adaptation on top of pretrained robust models, such as those trained with heavy data augmentations, is as simple as using any other pretrained model. Crucially, we find that, in practice, the set of augmentations that we use at test time $\mathcal{A}$ does not have to match the augmentations that were used to train the model. This is important as we require a few properties from the test time augmentations: that they can be easily sampled and are applied directly to the model input $\mathbf{x}$. These properties do not hold for, e.g., data augmentation techniques based on image translation models, such as DeepAugment [89], or feature mixing, such as moment exchange [122]. However, we can still use models trained with these data augmentation techniques as our starting point for adaptation, thus allowing us to improve upon their state-of-the-art results. As noted above, using pretrained models is not as easily accomplished for adaptation methods which require complicated or specialized training procedures and model architectures, such as TTT [216]

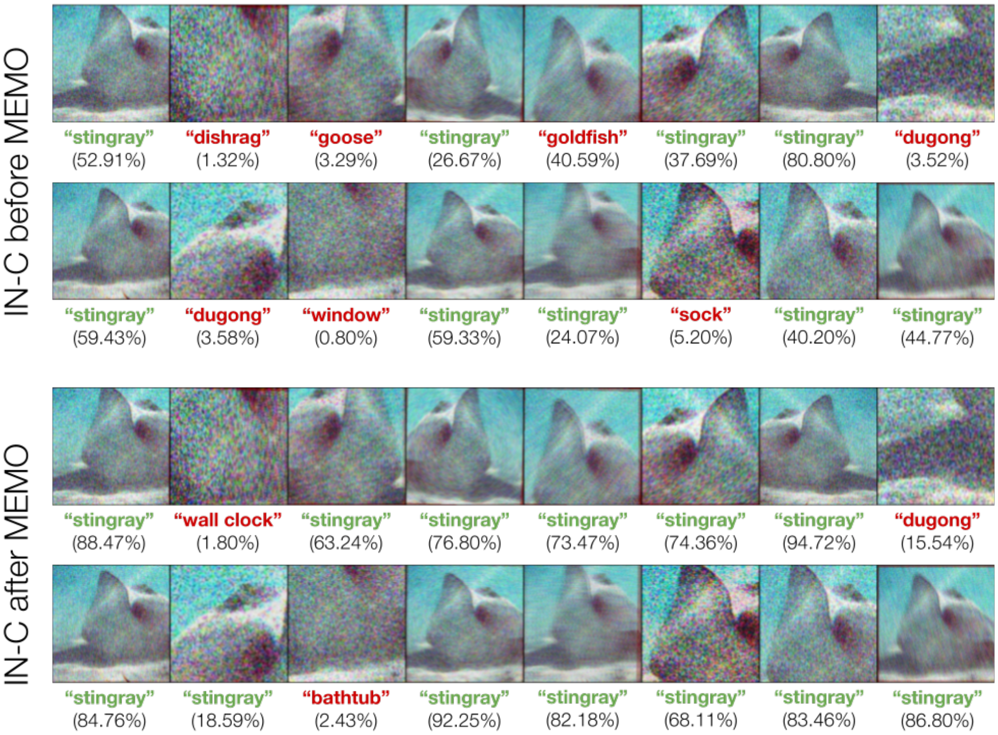


Figure 6.2: We visualize augmentations of a randomly chosen data point from the "Gaussian Noise level 3" ImageNet-C test set. Even for a robust model trained with heavy data augmentations [89], both its predictive accuracy and confidence drop sharply when encountering test shift. As shown in the bottom two rows, these drops can be remedied via MEMO.

or ARM [268]. In our experiments, we use AugMix as our set of augmentations [85], as it satisfies the above properties and yields significant diversity as depicted in Figure 6.2.

Adapting BN statistics. Prior work has shown that, even when presented with just a single test point, partially adapting the estimated mean and variance of the activations in each batch normalization (BN) layer of the model can still be effective in some cases for handling shift [197]. In this setting, to prevent overfitting to the test point, the channelwise mean and variance $[\boldsymbol{\mu}_{\text{test}}, \boldsymbol{\sigma}^2_{\text{test}}]$ estimated from this point are mixed with the the mean and variance $[\boldsymbol{\mu}_{\text{train}}, \boldsymbol{\sigma}^2_{\text{train}}]$ computed during training according to a prior strength N, i.e.,

$$\boldsymbol{\mu} \triangleq \frac{N}{N+1}\boldsymbol{\mu}_{\text{train}} + \frac{1}{N+1}\boldsymbol{\mu}_{\text{test}}\,, \boldsymbol{\sigma}^2 \triangleq \frac{N}{N+1}\boldsymbol{\sigma}^2_{\text{train}} + \frac{1}{N+1}\boldsymbol{\sigma}^2_{\text{test}}\,.$$

This technique is also straightforward to combine with MEMO: we simply use the adapted BN statistics whenever computing the model's output distribution. That is, we adapt the BN statistics alongside all of the model parameters for MEMO. Following the suggestion in prior work [197], we set $N = 16$ for all of our experiments in the next section.

6.4 Experiments

Our experiments aim to answer the following questions:

1. How does MEMO compare to prior methods for test time adaptation, which make additional training and test assumptions, and test time robustness?

2. Can MEMO be successfully combined with a wide range of pretrained models?

3. Which aspects of MEMO are the most important for strong performance?

We conduct experiments on two distribution shift benchmarks for CIFAR-10 [115] and three distribution shift benchmarks for ImageNet [189]. Specifically, for CIFAR-10, we evaluate on the CIFAR-10-C [84] and CIFAR-10.1 [176] test sets, and for ImageNet, we evaluate on the ImageNet-C [84], ImageNet-R [89], and ImageNet-A [86] test sets.

To answer question (1), we compare to test time training (TTT) [216] in the CIFAR-10 experiments, for which we train ResNet-26 models following their protocol and specialized architecture. We do not compare to TTT for the ImageNet experiments due to the computational demands of training state-of-the-art models and because TTT does not report competitive ImageNet results. For the ImageNet experiments, we compare to Tent [243] and BN adaptation, which can be used with pretrained models but require multiple test inputs (or even the entire test set) for adaptation. We provide BN adaptation with 256 test inputs at a time and, again following prior work [197], set the prior strength $N = 256$ accordingly.

For Tent, we use test batch sizes of 64 and, for ResNet-50 models, test both "online" adaptation – where the model adapts continually through the entire evaluation – and "episodic" adaptation – where the model is reset after each test batch [243]. Note that the evaluation protocols are different for these two methods: whereas MEMO is tasked with predicting on each test point immediately after adaptation, BN adaptation predicts on a batch of 256 test points after computing BN statistics on the batch, and Tent predicts on a batch of 64 inputs after adaptation but also, in the online setting, continually adapts throughout evaluation. In all experiments, we further compare to single point BN adaptation [197] and the TTA baseline that simply predicts according to the model's marginal output distribution over augmentations $\bar{p}_\theta(y|\mathbf{x})$ (Equation 6.1) [116, 9]. Full details on our experimental protocol are provided in Section C.1.

To answer question (2), we apply MEMO on top of multiple pretrained models with different architectures, trained via several different procedures. For CIFAR-10, we train our own ResNet-26 [80] models. For ImageNet, we use the best performing ResNet-50 robust models from prior work, which includes those trained with DeepAugment and AugMix augmentations [89] as well as those trained with moment exchange and CutMix [122]. To evaluate the generality of prior test time robustness methods and MEMO, we also evaluate the small robust vision transformer (RVT*-small), which provides superior performance on all three ImageNet distribution shift benchmarks compared to the robust ResNet-50 models [144].

Finally, to answer (3), we conduct ablative studies: first to determine the relative importance of maximizing confidence (via entropy minimization) versus enforcing invariant

	CIFAR-10 Error (%)	CIFAR-10.1 Error (%)	CIFAR-10-C Average Error (%)
ResNet-26 [80]	9.2	18.4	22.5
+ TTA	**7.3** (−1.9)	14.8 (−3.6)	19.9 (−2.6)
+ MEMO (ours)	**7.3 (−1.9)**	**14.7 (−3.7)**	**19.6 (−2.9)**
+ Joint training* [216]	8.1	16.7	22.8
+ TTT* [216]	7.9 (−0.2)	15.9 (−0.8)	21.5 (−1.3)

Table 6.1: Results for the original CIFAR-10 test set, CIFAR-10.1, and CIFAR-10-C. MEMO outperforms TTT despite not making any training assumptions. *Results from prior work [216].

predictions across augmented copies of each test point, and second to determine the importance of the particular augmentation functions used. The comparison to the non adaptive TTA baseline also helps determine whether simply augmenting the test point is sufficient or if adaptation is additionally helpful.

Main results

We summarize results for CIFAR-10, CIFAR-10.1, and CIFAR-10-C in Table 6.1, with full CIFAR-10-C results in Section C.3. We use indentations to indicate composition, e.g., TTT is performed at test time on top of their specialized joint training procedure. Across all corruption types in CIFAR-10-C, MEMO consistently improves test error compared to the baselines, non adaptive TTA, and TTT. MEMO also provides a larger performance gain on CIFAR-10.1 compared to TTT. We find that the non adaptive TTA baseline is competitive for these relatively simple test sets, though it is worse than MEMO for CIFAR-10-C. Of these three test sets, CIFAR-10-C is the only benchmark that explicitly introduces distribution shift, which suggests that adaptation is useful when the test shifts are more prominent. Both TTA and MEMO are also effective at improving performance for the original CIFAR-10 test set where there is no distribution shift, providing further support for the widespread use of augmentations in standard evaluation protocols [116, 9].

We summarize results for ImageNet-C, ImageNet-R, and ImageNet-A in Table 6.2, with complete ImageNet-C results in Section C.3. We again use indentations to indicate composition, e.g., the best results on ImageNet-C for our setting are attained through a combination of starting from a model trained with DeepAugment and AugMix [89] and using MEMO on top. For both ImageNet-C and ImageNet-R, and for both the ResNet-50 and RVT*-small models, combining MEMO with robust training techniques leads to new state-of-the-art performance among methods that observe only one test point at a time. We highlight in gray the methods that require multiple test points for adaptation, and we list in bold the best results

	ImageNet-C	ImageNet-R	ImageNet-A
	mCE ↓	Error (%)	Error (%)
Baseline ResNet-50 [80]	76.7	63.9	100.0
+ TTA	77.9 (+1.2)	61.3 (−2.6)	98.4 (−1.6)
+ Single point BN	71.4 (−5.3)	61.1 (−2.8)	99.4 (−0.6)
+ MEMO (ours)	69.9 (−6.8)	58.8 (−5.1)	99.1 (−0.9)
+ BN ($N = 256, n = 256$)	61.6 (−15.1)	59.7 (−4.2)	99.8 (−0.2)
+ Tent (online) [243]	54.4 (−22.3)	57.7 (−6.2)	99.8 (−0.2)
+ Tent (episodic)	64.7 (−12.0)	61.0 (−2.9)	99.7 (−0.3)
+ DeepAugment+AugMix [89]	53.6	53.2	96.1
+ TTA	55.2 (+1.6)	51.0 (−2.2)	93.5 (−2.6)
+ Single point BN	51.3 (−2.3)	51.2 (−2.0)	95.4 (−0.7)
+ MEMO (ours)	**49.8** **(−3.8)**	**49.2** **(−4.0)**	94.8 (−1.3)
+ BN ($N = 256, n = 256$)	45.4 (−8.2)	48.8 (−4.4)	96.8 (+0.7)
+ Tent (online)	**43.5** **(−10.1)**	**46.9** **(−6.3)**	96.7 (+0.6)
+ Tent (episodic)	47.1 (−6.5)	50.1 (−3.1)	96.6 (+0.5)
+ MoEx+CutMix [122]	74.8	64.5	91.9
+ TTA	75.7 (+0.9)	62.7 (−1.8)	89.5 (−2.4)
+ Single point BN	71.0 (−3.8)	62.6 (−1.9)	91.1 (−0.8)
+ MEMO (ours)	69.1 (−5.7)	59.4 (−3.3)	**89.0** **(−2.9)**
+ BN ($N = 256, n = 256$)	60.9 (−13.9)	61.6 (−2.9)	93.9 (+2.0)
+ Tent (online)	54.0 (−20.8)	58.7 (−5.8)	94.4 (+2.5)
+ Tent (episodic)	66.2 (−8.6)	63.9 (−0.6)	94.7 (+2.8)
RVT*-small [144]	49.4	52.3	73.9
+ TTA	53.0 (+3.6)	49.0 (−3.3)	**68.9** **(−5.0)**
+ Single point BN	48.0 (−1.4)	51.1 (−1.2)	74.4 (+0.5)
+ MEMO (ours)	**40.6** **(−8.8)**	**43.8** **(−8.5)**	69.8 (−4.1)
+ BN ($N = 256, n = 256$)	44.3 (−5.1)	51.0 (−1.3)	78.3 (+4.4)
+ Tent (online)	46.8 (−2.6)	50.7 (−1.6)	82.1 (+8.2)
+ Tent (adapt all)	44.7 (−4.7)	74.1 (+21.8)	81.1 (+7.2)

Table 6.2: Test results for ImageNet-C, ImageNet-R, and ImageNet-A. MEMO achieves new state-of-the-art performance on each benchmark for ResNet-50 models for the single test point setting. For RVT*-small, MEMO substantially improves performance across all benchmarks and reaches a new state of the art for ImageNet-C and ImageNet-R.

from these methods which outperform the test time robustness methods. As Table 6.2 and prior work both show [197, 243], accessing multiple test points can be powerful for benchmarks such as ImageNet-C and ImageNet-R, in which inferred statistics from the test input distribution may aid in prediction. In the case of Tent, which adapts online, the model has adapted using the entire test set by the end of evaluation. However, these methods do not help, and oftentimes even hurt, for ImageNet-A. Furthermore, we find that these methods are less effective with the RVT*-small model, which may indicate their sensitivity to model architecture choices. Therefore, for this model, we also test a modification of Tent which adapts all parameters, and we find that this version of Tent works better for ImageNet-C but is significantly worse for ImageNet-R.

MEMO results in substantial improvement for ImageNet-A and is competitive with TTA on this problem. No prior test time adaptation methods have reported improvements on ImageNet-A, and some have reported explicit negative results [197]. As discussed, it is reasonable for adaptation methods that rely on multiple test points to achieve greater success on other benchmarks such as ImageNet-C, in which a batch of inputs provides significant information about the specific corruption that must be dealt with. In contrast, ImageNet-A does not have such obvious characteristics associated with the input distribution, as it is simply a collection of images that are difficult to classify. As MEMO instead extracts a learning signal from single test points, it is, to the best of our knowledge, the first test time adaptation method to report successful results on this testbed. When used on top of a model trained with moment exchange and CutMix [122], MEMO achieves state-of-the-art performance among ResNet-50 models and single test point methods. TTA generally offers larger performance gains on ImageNet-A and also results in the highest overall accuracy when combined with the RVT*-small model; however, TTA performs worse than MEMO on ImageNet-R and consistently *decreases* accuracy on ImageNet-C compared to standard evaluation. We view the consistency with which MEMO outperforms the best prior methods, which change across different test sets, to be a major advantage of the proposed method.

Ablative study

MEMO increases model robustness at test time via adaptation and augmentation. In this section, we ablate the adaptation procedure, and in Section C.2 we ablate the use and choice of augmentations.

From the results above, we conclude that adaptation generally provides additional benefits beyond simply using TTA to predict via the marginal output distribution $\bar{p}_\theta(y|\mathbf{x})$. However, we can disentangle two distinct self-supervised learning signals that may be effective for adaptation: encouraging invariant predictions across different augmentations of the test point, and encouraging confidence via entropy minimization. The marginal entropy objective in Equation 6.2 encapsulates both of these learning signals, but it cannot easily be decomposed into these pieces. Thus, we instead use two ablative adaptation methods that each only make use of one of these learning signals.

First, we consider optimizing the pairwise cross entropy between each pair of augmented

	CIFAR-10	CIFAR-10.1	CIFAR-10-C
	Error (%)	Error (%)	Average Error (%)
ResNet-26 [80]	9.2	18.4	22.5
+ MEMO (ours)	**7.3 (−1.9)**	**14.7 (−3.7)**	**19.6 (−2.9)**
$-\ \ell$ (Equation 6.2) $+\ \ell_{\text{PCE}}$	7.6 (−1.6)	15.3 (−3.1)	20.0 (−2.5)
$-\ \ell$ (Equation 6.2) $+\ \ell_{\text{CE}}$	7.6 (−1.6)	**14.7 (−3.7)**	20.0 (−2.5)
	ImageNet-C	ImageNet-R	ImageNet-A
	mCE ↓	Error (%)	Error (%)
RVT*-small [144]	49.4	52.3	73.9
+ MEMO (ours)	**40.6 (−8.8)**	**43.8 (−8.5)**	69.8 (−4.1)
$-\ \ell$ (Equation 6.2) $+\ \ell_{\text{CE}}$	41.2 (−8.2)	44.2 (−8.1)	**69.7 (−4.2)**

Table 6.3: Ablating the adaptation objective to test pairwise cross entropy and conditional entropy (CE) based adaptation. MEMO generally performs the best, indicating that both encouraging invariance across augmentations and confidence are helpful in adapting the model.

points, i.e.,

$$\ell_{\text{PCE}}(\theta; \mathbf{x}) \triangleq \frac{1}{B \times (B-1)} \sum_{i=1}^{B} \sum_{j \neq i} \sum_{y \in \mathcal{Y}} p_\theta(y|\tilde{\mathbf{x}}_i) \log p_\theta(y|\tilde{\mathbf{x}}_j)\,,$$

Where $\tilde{\mathbf{x}}_i$ again refers to the i-th sampled augmentation applied to $\mathbf{x}$. Intuitively, this loss function encourages the model to adapt such that it produces the same predictive distribution for all augmentations of the test point, but it does not encourage the model to produce confident predictions. Conversely, as an objective that encourages confidence but not invariance, we also consider optimizing conditional entropy on the batch of augmented points, i.e.,

$$\ell_{\text{CE}}(\theta; \mathbf{x}) \triangleq \frac{1}{B} \sum_{i=1}^{B} H(p_\theta(\cdot|\tilde{\mathbf{x}}_i))\,.$$

This ablation is effectively a version of the episodic variant of Tent [243] that produces augmented copies of a single test point rather than assuming access to a test batch. We first evaluate these ablations on the CIFAR-10 test sets. We use the same adaptation procedure outlined in Algorithm 2, with ℓ replaced with the above objectives, and we keep the same hyperparameter values.

The results are presented in Table 6.3. We see that MEMO, i.e., marginal entropy minimization, generally performs better than adaptation with either of the alternative objectives. This supports the hypothesis that both invariance across augmentations and confidence are important learning signals for self-supervised adaptation. When faced with CIFAR-10.1, we see poor performance from the pairwise cross entropy based adaptation method. On the original CIFAR-10 test set and CIFAR-10-C, the ablations perform nearly identically and uniformly worse than MEMO. To further test the ℓ_{CE} ablation, we also evaluate it on the ImageNet test sets for the RVT*-small model. We find that, similarly, minimizing conditional entropy generally improves performance compared to the baseline evaluation. MEMO is more performant for ImageNet-C and ImageNet-R, again indicating the benefits of encouraging invariance to augmentations. Adaptation via ℓ_{CE} performs slightly better for ImageNet-A, though for this problem, TTA is still the best method.

6.5 Discussion

We presented MEMO, a method for test time robustification again distribution shift via adaptation and augmentation. MEMO does not require access or changes to the model training procedure and is thus broadly applicable for a wide range of pretrained models. Furthermore, MEMO adapts at test time using single test inputs, thus it does not assume access to multiple test points as in several recent methods for test time adaptation [197, 243, 268]. On a range of distribution shift benchmarks for CIFAR-10 and ImageNet classification, and for both ResNet and vision transformer models, MEMO consistently improves performance at test time and achieves several new state-of-the-art results for these models in the single test point setting.

Inference via MEMO is more computationally expensive than standard model inference, primarily because adaptation is performed per test point and thus inference cannot be batched. When deployed in the real world, it is natural to expect that test points will arrive one at a time and batched inference will not be possible. However, MEMO is also more computationally expensive due to its augmentation and adaptation procedure. In Section C.2 we further study the tradeoff between efficient and model accuracy, and we find that smaller values of B can lead to favorable tradeoffs in terms of significantly speeding up inference without sacrificing much in terms of accuracy. One interesting direction for future work to further improve efficiency is to develop techniques for selectively determining when to adapt the model. For example, with well calibrated models [75], we may run simple “feedforward” inference when the prediction confidence is over a certain threshold, thus achieving better efficiency. Additionally, it would be interesting to explore MEMO in the test setting where the model is allowed to continually adapt as more test data is observed. In our preliminary experiments in this setting, MEMO tended to lead to degenerate solutions, e.g., the model predicting a constant label with maximal confidence, and this may potentially be rectified by carefully choosing which parameters to adapt [243] or regularizing the model such that it does not change too drastically from the pretrained model.

Chapter 7

Conclusion

Distribution shift in machine learning is a broad topic, and in this thesis, we have discussed it broadly.

In Part I, we covered numerous examples of how shift occurs in machine learning problems, for both RL and supervised learning settings. This part of the thesis focused on the distribution shift *problems* themselves, in order to illustrate how shift arises in machine learning applications and motivate the need for addressing these shifts. Chapter 3 presented three examples from RL: distribution shift arising from the simulation to real world transfer of a policy for tensegrity robot locomotion, from using dynamics models to predict for updated policies, and from attempting to learn tasks directly from human demonstration videos. In the first example of simulation to real world transfer, improving model (in this case, policy) performance under shift was the goal itself – however, for this specific platform, we could rely on domain knowledge and handcrafted techniques to achieve this goal. In the second and third examples, the shift instead represented an obstacle to the actual goal of learning to solve the task at hand. For these cases, techniques were instead devised to effectively mitigate the shift rather than tackling it head on – specifically, using truncated horizons for model predictions, and supplementing translated instruction images with human feedback.

Chapter 4 shifted focus to supervised learning problems. In this setting, there is a clear need for general purpose frameworks and methods for dealing with distribution shift, and the abundance of challenging problems demonstrates this point. We reviewed a number of prior works which have proposed various benchmarks for domain adaptation, domain generalization, general robustness, and more. We then presented our work aiming to complement this growing landscape in two ways. First, we proposed Wilds, a benchmark of domain generalization and subpopulation shift problems carefully curated to faithfully represent important real world applications. Second, we introduced additional problems in the same settings geared toward studying methods for adaptation, as there is strong intuition that adaptation is beneficial, perhaps even crucial, for achieving strong results for these problems.

Chapter 5, the first of two chapters in Part II, helped confirm this intuition with strong empirical evidence. We proposed the ARM framework, in which models are meta-trained such that they learn how to adapt to domain shift at test time. We provided strong theoretical justification for the ARM objective based on prior work, and we proposed a general algorithm and three specific methods for instantiating and optimizing the ARM objective inspired by meta-learning and test time adaptation. On the aforementioned problems for adaptation, as well as Wilds problems, ARM methods outperformed prior robustness, invariance, and adaptation methods in terms of both worst case and average accuracy.

Chapter 6 continued Part II by introducing another approach for adaptation to distribution shift, however from very different assumptions. Specifically, almost all assumptions were stripped away: access to training domains or other types of training information, access to even the model training procedure itself, and access to multiple unlabeled inputs at test time. In this sparse setup, we demonstrated that various techniques such as adaptation and data augmentation still incorporated favorable inductive biases for model improvement – inspired by these insights, we proposed the MEMO test time robustness method which utilizes both adaptation and augmentation. Here, we showed on several different challenging ImageNet test sets, on a range of model architectures and models trained with different procedures, that MEMO resulted in significant accuracy gains compared to the baseline test time inference procedure as well as prior methods for test time robustness. To summarize, Part II presented adaptation based approaches to distribution shift problems from Part I, specifically the supervised learning problems from Chapter 4.

From this thesis, there can be a number of takeaways. The first takeaway from Chapter 3 is perhaps obvious: distribution shift problems do not always have to be solved in the most general way, or even solved at all. Oftentimes, incorporating domain expertise or finding ways to sidestep the shift are the most direct and effective paths toward the end goal, and this is the path that many practitioners should take when encountering shift. The second takeaway from Chapter 4 and Chapter 5 is that there may not even exist a "one size fits all" approach to solving distribution shift problems anyway, simply due to the enormity and diversity of the set of problems. This motivates both the first takeaway as well as the study and development of additional general purpose methods that can better address important real world instances of shift. The third takeaway from Chapter 5 and Chapter 6 is that, though it may not solve all distribution shift problems, approaches based on adaptation appear to be a strong candidate for further study, in particular in the era of deep learning. As mentioned in Chapter 1, deep neural network models typically have the capacity to absorb additional information even after training on large datasets. Whether this information comes from inferring statistics of the test distribution, harnessing useful inductive biases for the test input, or something else, it is likely that these techniques will beat out methods that use a static model at test time. Indeed, we found that MEMO could further improve the performance of a large ResNext-101 model trained on billions of images from social media [257, 143] (Section C.2), lending further support to the potential of adaptation to already robust models.

Another interesting avenue for future work, that was not considered in depth in this thesis, is to study the adaptation setting in which some amount of labeled data is available at test time. Though this assumption is more demanding, it greatly expands the scope of shifts that can be handled in principle. For example, shifts that cause drastic changes in the domain, such as new classes or even a completely different set of classes, may be tackled from the perspective of labeled adaptation. Paradigms such as unsupervised or weakly supervised learning, massive scale pretraining, and meta-learning may all play key roles in the development of performant methods for this setting.

Bibliography

[1] B. Abelson, K. Varshney, and J. Sun. "Targeting Direct Cash Transfers to the Extremely Poor". In: *International Conference on Knowledge Discovery and Data Mining (KDD).* 2014.

[2] J. Ahumada et al. "Wildlife Insights: A Platform to Maximize the Potential of Camera Trap and Other Passive Sensor Wildlife Data for the Planet". In: *Environmental Conservation* (2020).

[3] E. AlBadawy, A. Saha, and M. Mazurowski. "Deep Learning for Segmentation of Brain Tumors: Impact of Cross-Institutional Training and Testing". In: *Med Phys.* (2018).

[4] F. Alet et al. "Tailoring: Encoding Inductive Biases by Optimizing Unsupervised Objectives at Prediction Time". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2021.

[5] M. Allamanis and M. Brockschmidt. "Smartpaste: Learning to Adapt Source Code". In: *arXiv preprint arXiv:1705.07867* (2017).

[6] A. Antoniou and A. Storkey. "Learning to Learn via Self-Critique". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2019.

[7] B. Argall et al. "A Survey of Robot Learning from Demonstration". In: *Robotics and Autonomous Systems* (2009).

[8] M. Arjovsky et al. "Invariant Risk Minimization". In: *arXiv preprint arXiv:1907.02893* (2019).

[9] A. Ashukha et al. "Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning". In: *International Conference on Learning Representations (ICLR).* 2020.

[10] T. Ayalew, J. Ubbens, and I. Stavness. "Unsupervised Domain Adaptation for Plant Organ Counting". In: *European Conference on Computer Vision (ECCV).* 2020.

[11] M. Badgeley et al. "Deep Learning Predicts Hip Fracture Using Confounding Patient and Healthcare Variables". In: *npj Digital Medicine* (2019).

[12] P. Bandi et al. “From Detection of Individual Metastases to Classification of Lymph Node Status at the Patient Level: the CAMELYON17 Challenge”. In: *IEEE Transactions on Medical Imaging* (2018).

[13] A. Barbu et al. “ObjectNet: A Large-Scale Bias-Controlled Dataset for Pushing the Limits of Object Recognition Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[14] S. Beery, E. Cole, and A. Gjoka. “The iWildCam 2020 Competition Dataset”. In: *arXiv preprint arXiv:2004.10340* (2020).

[15] S. Beery, G. van Horn, and P. Perona. “Recognition in Terra Incognita”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[16] S. Beery, D. Morris, and S. Yang. “Efficient Pipeline for Camera Trap Image Review”. In: *arXiv preprint arXiv:1907.06772* (2019).

[17] M. Bellemare et al. “Autonomous Navigation of Stratospheric Balloons using Reinforcement Learning”. In: *Nature* (2020).

[18] A. Ben-Tal et al. “Robust Solutions of Optimization Problems Affected by Uncertain Probabilities”. In: *Management Science* (2013).

[19] S. Bengio et al. “On the Optimization of a Synaptic Learning Rule”. In: *Optimality in Artificial and Biological Neural Networks*. 1992.

[20] A. Billard and M. Mataric. “Learning Human Arm Movements by Imitation: Evaluation of a Biologically Inspired Connectionist Architecture”. In: *Robotics and Autonomous Systems* (2001).

[21] G. Blanchard, G. Lee, and C. Scott. “Generalizing from Several Related Classification Tasks to a New Unlabeled Sample”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2011.

[22] G. Blanchard et al. “Domain Generalization by Marginal Transfer Learning”. In: *Journal of Machine Learning Research (JMLR)* (2021).

[23] J. Blumenstock, G. Cadamuro, and R. On. “Predicting Poverty and Wealth from Mobile Phone Metadata”. In: *Science* (2015).

[24] R. Bohacek, C. McMartin, and W. Guida. “The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective”. In: *Medicinal Research Reviews* (1996).

[25] D. Borkan et al. “Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification”. In: *World Wide Web Conference (WWW)*. 2019.

[26] M. Boutros, F. Heigwer, and C. Laufer. “Microscopy-Based High-Content Screening”. In: *Cell* (2015).

[27] M. Bray et al. “Cell Painting, a High-Content Image-Based Assay for Morphological Profiling using Multiplexed Fluorescent Dyes”. In: *Nature Protocols* (2016).

[28] J. Broach and J. Thorner. “High-Throughput Screening for Drug Discovery”. In: *Nature* (1996).

[29] J. Bruce et al. “Design and Evolution of a Modular Tensegrity Robot Platform”. In: *International Conference on Robotics and Automation (ICRA)*. 2014.

[30] M. Bruch, M. Monperrus, and M. Mezini. “Learning from Examples to Improve Code Completion Systems”. In: *European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 2009.

[31] J. Buckman et al. “Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[32] J. Buolamwini and T. Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification”. In: *Conference on Fairness, Accountability and Transparency (FAT*)*. 2018.

[33] J. Byrd and Z. Lipton. “What is the Effect of Importance Weighting in Deep Learning?” In: *International Conference on Machine Learning (ICML)*. 2019.

[34] J. Caicedo et al. “Data-Analysis Strategies for Image-Based Cell Profiling”. In: *Nature Methods* (2017).

[35] S. Caldas et al. “LEAF: A Benchmark for Federated Settings”. In: *Workshop on Federated Learning for Data Privacy and Confidentiality*. 2019.

[36] K. Caluwaerts et al. “State Estimation for Tensegrity Robots”. In: *International Conference on Robotics and Automation (ICRA)*. 2016.

[37] F. Carlucci et al. “AutoDIAL: Automatic DomaIn Alignment Layers”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[38] G. Christie et al. “Functional Map of the World”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[39] S. Christin, É. Hervet, and N. Lecomte. “Applications for Deep Learning in Ecology”. In: *Methods in Ecology and Evolution* (2019).

[40] K. Chua et al. “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[41] G. Cohen et al. “EMNIST: An Extension of MNIST to Handwritten Letters”. In: *arXiv preprint arXiv:1702.05373* (2017).

[42] F. Croce et al. “RobustBench: A Standardized Adversarial Robustness Benchmark”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[43] G. Csurka. “Domain Adaptation for Visual Applications: A Comprehensive Survey”. In: *arXiv preprint arXiv:1702.05374* (2017).

[44] M. Cutler, T. Walsh, and J. How. “Real-World Reinforcement Learning via Multifidelity Simulators”. In: *IEEE Transactions on Robotics* (2016).

[45] A. D’Amour et al. “Underspecification Presents Challenges for Credibility in Modern Machine Learning”. In: *arXiv preprint arXiv:2011.03395* (2020).

[46] D. Dai and L. Van Gool. “Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime”. In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[47] H. Daumé III. “Frustratingly Easy Domain Adaptation”. In: *Conference of the Association for Computational Linguistics (ACL)*. 2007.

[48] E. David et al. “Global Wheat Head Detection (GWHD) dataset: A Large and Diverse Dataset of High-Resolution RGB-Labelled Images to Develop and Benchmark Wheat Head Detection Methods”. In: *Plant Phenomics* (2020).

[49] R. Deo. “Machine Learning in Medicine”. In: *Circulation* (2015).

[50] L. Dixon et al. “Measuring and Mitigating Unintended Bias in Text Classification”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[51] A. Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[52] Q. Dou et al. “Domain Generalization via Model-Agnostic Learning of Semantic Features”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[53] Y. Du and K. Narasimhan. “Task-Agnostic Dynamics Priors for Deep Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2019.

[54] C. Echeverri and N. Perrimon. “High-Throughput RNAi Screening in Cultured Cells: A User’s Guide”. In: *Nature Reviews Genetics* (2006).

[55] H. Edwards and A. Storkey. “Towards a Neural Statistician”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[56] J. Espey et al. “Data for Development: A Needs Assessment for SDG Monitoring and Statistical Capacity Development”. In: *Sustainable Development Solutions Network* (2015).

[57] C. Fang, Y. Xu, and D. Rockmore. “Unbiased Metric Learning: On the Utilization of Multiple Datasets and Web Images for Softening Bias”. In: *International Conference on Computer Vision (ICCV)*. 2013.

[58] V. Feinberg et al. “Model-Based Value Estimation for Efficient Model-Free Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2018.

[59] C. Finn, P. Abbeel, and S. Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: *International Conference on Machine Learning (ICML)*. 2017.

[60] C. Franks et al. "CACHECA: A Cache Language Model Based Code Suggestion Tool". In: *International Conference on Software Engineering (ICSE)*. 2015.

[61] J. Fu et al. "Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[62] Y. Ganin and V. Lempitsky. "Unsupervised Domain Adaptation by Backpropagation". In: *International Conference on Machine Learning (ICML)*. 2015.

[63] M. Garnelo et al. "Conditional Neural Processes". In: *International Conference on Machine Learning (ICML)*. 2018.

[64] R. Geirhos et al. "Generalisation in Humans and Deep Neural Networks". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[65] R. Geirhos et al. "ImageNet-trained CNNs Are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness". In: *International Conference on Learning Representations (ICLR)*. 2019.

[66] R. Geirhos et al. "Shortcut Learning in Deep Neural Networks". In: *Nature Machine Intelligence* (2020).

[67] M. Geva, Y. Goldberg, and J. Berant. "Are We Modeling the Task or the Annotator? An Investigation of Annotator Bias in Natural Language Understanding Datasets". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[68] B. Gong et al. "Geodesic Flow Kernel for Unsupervised Domain Adaptation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[69] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[70] I. Goodfellow, G. Shlens, and C. Szegedy. "Explaining and Harnessing Adversarial Examples". In: *International Conference on Learning Representations (ICLR)*. 2015.

[71] Y. Grandvalet and Y. Bengio. "Semi-Supervised Learning by Entropy Minimization". In: *Advances in Neural Information Processing Systems (NIPS)*. 2005.

[72] M. Grooten, T. Peterson, and R. Almond. *Living Planet Report 2020 - Bending the Curve of Biodiversity Loss*. WWF, Gland, Switzerland, 2020.

[73] S. Gu et al. "Continuous Deep Q-Learning with Model-based Acceleration". In: *International Conference on Machine Learning (ICML)*. 2016.

[74] I. Gulrajani and D. Lopez-Paz. "In Search of Lost Domain Generalization". In: *International Conference on Learning Representations (ICLR)*. 2021.

[75] C. Guo et al. "On Calibration of Modern Neural Networks". In: *International Conference on Machine Learning (ICML)*. 2017.

[76] T. Haarnoja et al. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning (ICML)*. 2018.

[77] M. Hansen et al. "High-Resolution Global Maps of 21st-Century Forest Cover Change". In: *Science* (2013).

[78] J. Harrill et al. "Considerations for Strategic Use of High-Throughput Transcriptomics Chemical Screening Data in Regulatory Decisions". In: *Current Opinion in Toxicology* (2019).

[79] G. Hayes and J. Demiris. "A Robot Controller using Learning by Imitation". In: *International Symposium on Intelligent Robotic Systems (SIRS).* 1994.

[80] K. He et al. "Deep Residual Learning for Image Recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016.

[81] Y. He, Z. Shen, and P. Cui. "Towards Non-I.I.D. Image Classification: A Dataset and Baselines". In: *Pattern Recognition* (2020).

[82] N. Heess et al. "Learning Continuous Control Policies by Stochastic Value Gradients". In: *Advances in Neural Information Processing Systems (NIPS).* 2015.

[83] V. Hellendoorn et al. "When Code Completion Fails: A Case Study on Real-World Completions". In: *International Conference on Software Engineering (ICSE).* 2019.

[84] D. Hendrycks and T. Dietterich. "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations". In: *International Conference on Learning Representations (ICLR).* 2019.

[85] D. Hendrycks et al. "AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty". In: *International Conference on Learning Representations (ICLR).* 2020.

[86] D. Hendrycks et al. "Natural Adversarial Examples". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2021.

[87] D. Hendrycks et al. "Pretrained Transformers Improve Out-of-Distribution Robustness". In: *Conference of the Association for Computational Linguistics (ACL).* 2020.

[88] D. Hendrycks et al. "Scaling Out-of-Distribution Detection for Real-World Settings". In: *arXiv preprint arXiv:1911.11132* (2020).

[89] D. Hendrycks et al. "The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization". In: *International Conference on Computer Vision (ICCV).* 2021.

[90] L. Hewitt et al. "The Variational Homoencoder: Learning to Learn High Capacity Generative Models from Few Examples". In: *Conference on Uncertainty in Artificial Intelligence (UAI).* 2018.

[91] S. Hochreiter, A. Younger, and P. Conwell. "Learning to Learn using Gradient Descent". In: *International Conference on Artificial Neural Networks (ICANN).* 2001.

[92] J. Hoffman et al. "CyCADA: Cycle Consistent Adversarial Domain Adaptation". In: *International Conference on Machine Learning (ICML).* 2018.

[93] K. Hsu, S. Levine, and C. Finn. “Unsupervised Learning via Meta-Learning”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[94] S. Hu et al. “Empirical Bayes Transductive Meta-Learning with Synthetic Gradients”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[95] W. Hu et al. “Does Distributionally Robust Supervised Learning Give Robust Classifiers?” In: *International Conference on Machine Learning (ICML)*. 2018.

[96] W. Hu et al. “Open Graph Benchmark: Datasets for Machine Learning on Graphs”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[97] G. Huang et al. “Densely Connected Convolutional Networks”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[98] Y. Huang et al. “Neural Networks with Recurrent Generative Feedback”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[99] J. Hughes et al. “Principles of Early Drug Discovery”. In: *British Journal of Pharmacology* (2011).

[100] S. Ioffe and C. Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *International Conference on Machine Learning (ICML)*. 2015.

[101] Y. Iwasawa and Y. Matsuo. “Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[102] E. Jang, S. Gu, and B. Poole. “Categorical Reparameterization with Gumbel-Softmax”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[103] M. Janner et al. “When to Trust Your Model: Model-Based Policy Optimization”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[104] N. Jean, S. Xie, and S. Ermon. “Semi-supervised Deep Kernel Learning: Regression with Unlabeled Data by Minimizing Predictive Variance”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[105] N. Jean et al. “Combining Satellite Imagery and Machine Learning to Predict Poverty”. In: *Science* (2016).

[106] L. Kaelbling, M. Littman, and A. Cassandra. “Planning and Acting in Partially Observable Stochastic Domains”. In: *Artificial Intelligence* (1998).

[107] A. Kaku et al. “Be Like Water: Robustness to Extraneous Variables via Adaptive Feature Normalization”. In: *arXiv preprint arXiv:2002.04019* (2020).

[108] G. Kalweit and J. Boedecker. “Uncertainty-driven Imagination for Continuous Deep Reinforcement Learning”. In: *Conference on Robot Learning (CoRL)*. 2017.

[109] D. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations (ICLR)*. 2015.

[110] D. Kingma and M. Welling. “Auto-Encoding Variational Bayes”. In: *International Conference on Learning Representations (ICLR).* 2014.

[111] A. Koenecke et al. “Racial Disparities in Automated Speech Recognition”. In: *Proceedings of the National Academy of Sciences* (2020).

[112] P. Koh et al. “Concept Bottleneck Models”. In: *International Conference on Machine Learning (ICML).* 2020.

[113] P. Koh et al. “WILDS: A Benchmark of in-the-Wild Distribution Shifts”. In: *International Conference on Machine Learning (ICML).* 2021.

[114] D. Komura and S. Ishikawa. “Machine Learning Methods for Histopathological Image Analysis”. In: *Computational and Structural Biotechnology Journal* (2018).

[115] A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images.* Tech. rep. University of Toronto, 2009.

[116] A. Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems (NIPS).* 2012.

[117] A. Kumagai and T. Iwata. “Zero-shot Domain Adaptation with Domain Semantic Descriptors”. In: *arXiv preprint arXiv:1807.02927* (2018).

[118] Y. Kuniyoshi, M. Inaba, and H. Inoue. “Learning by Watching: Extracting Reusable Task Knowledge from Visual Observation of Human Performance”. In: *IEEE Transactions on Robotics and Automation* (1994).

[119] B. Lake and M. Baroni. “Generalization without Systematicity: On the Compositional Skills of Sequence-to-Sequence Recurrent Networks”. In: *International Conference on Machine Learning (ICML).* 2018.

[120] J. Leek et al. “Tackling the Widespread and Critical Impact of Batch Effects in High-Throughput Data”. In: *Nature Reviews Genetics* (2010).

[121] S. Levine and P. Abbeel. “Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics”. In: *Advances in Neural Information Processing Systems (NIPS).* 2014.

[122] B. Li et al. “On Feature Normalization and Data Augmentation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2021.

[123] D. Li et al. “Deeper, Broader and Artier Domain Generalization”. In: *International Conference on Computer Vision (ICCV).* 2017.

[124] D. Li et al. “Learning to Generalize: Meta-Learning for Domain Generalization”. In: *AAAI Conference on Artificial Intelligence (AAAI).* 2018.

[125] H. Li, D. Quang, and Y. Guan. “Anchor: Trans-Cell Type Prediction of Transcription Factor Binding Sites”. In: *Genome Research* (2019).

[126] H. Li et al. “Domain Generalization with Adversarial Feature Learning”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[127] T. Li et al. “Fair Resource Allocation in Federated Learning”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[128] X. Li et al. “Learning to Self-Train for Semi-Supervised Few-Shot Classification”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[129] Y. Li et al. “Deep Domain Generalization via Conditional Invariant Adversarial Networks”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[130] Y. Li et al. “Domain Generalization via Conditional Invariant Representations”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[131] Y. Li et al. “Revisiting Batch Normalization for Practical Domain Adaptation”. In: *International Conference on Learning Representations Workshop (ICLRW)*. 2017.

[132] K. Liakos et al. “Machine Learning in Agriculture: A Review”. In: *Sensors* (2018).

[133] Z. Lipton, Y. Wang, and A. Smola. “Detecting and Correcting for Label Shift with Black Box Predictors”. In: *International Conference on Machine Learning (ICML)*. 2018.

[134] Y. Liu et al. “Learning to Propagate Labels: Transductive Propagation Network for Few-Shot Learning”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[135] M. Long et al. “Conditional Adversarial Domain Adaptation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[136] I. Loshchilov and F. Hutter. “Decoupled Weight Decay Regularization”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[137] S. Lu et al. “CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation”. In: *arXiv preprint arXiv:2102.04664* (2021).

[138] Y. Luo et al. “Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees”. In: *International Conference on Learning Representations (ICLR)*. 2019.

[139] J. Lyu et al. “Ultra-Large Library Docking for Discovering New Chemotypes”. In: *Nature* (2019).

[140] R. Macarron et al. “Impact of High-Throughput Screening in Biomedical Research”. In: *Nature reviews Drug discovery* (2011).

[141] C. Maddison, A. Mnih, and Y. Teh. “The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables”. In: *International Conference on Learning Representations (ICLR)*. 2017.

[142] S. Madec et al. “Ear Density Estimation from High Resolution RGB Imagery using Deep Learning Technique”. In: *Agricultural and Forest Meteorology* (2019).

[143] D. Mahajan et al. “Exploring the Limits of Weakly Supervised Pretraining”. In: *European Conference on Computer Vision (ECCV).* 2018.

[144] X. Mao et al. “Towards Robust Vision Transformer”. In: *arXiv preprint arXiv:2105.07926* (2021).

[145] K. McCloskey et al. “Machine Learning on DNA-Encoded Libraries: A New Paradigm for Hit Finding”. In: *Journal of Medicinal Chemistry* (2020).

[146] N. Meinshausen and P. Bühlmann. “Maximin Effects in Inhomogeneous Large-Scale Data”. In: *Annals of Statistics* (2015).

[147] L. Metz et al. “Meta-Learning Update Rules for Unsupervised Representation Learning”. In: *International Conference on Learning Representations (ICLR).* 2019.

[148] D. Molchanov et al. “Greedy Policy Search: A Simple Baseline for Learnable Test-Time Augmentation”. In: *Conference on Uncertainty in Artificial Intelligence (UAI).* 2020.

[149] W. Montgomery and S. Levine. “Guided Policy Search as Approximate Mirror Descent”. In: *Advances in Neural Information Processing Systems (NIPS).* 2016.

[150] I. Mordatch, K. Lowrey, and E. Todorov. “Ensemble-CIO: Full-body Dynamic Motion Planning that Transfers to Physical Humanoids”. In: *International Conference on Intelligent Robots and Systems (IROS).* 2015.

[151] K. Muandet, D. Balduzzi, and B. Schölkopf. “Domain Generalization via Invariant Feature Representation”. In: *International Conference on Machine Learning (ICML).* 2013.

[152] Z. Nado et al. “Evaluating Prediction-Time Batch Normalization for Robustness under Covariate Shift”. In: *arXiv preprint arXiv:2006.10963* (2020).

[153] A. Nagabandi, C. Finn, and S. Levine. “Deep Online Learning via Meta-Learning: Continual Adaptation for Model-Based RL”. In: *International Conference on Learning Representations (ICLR).* 2019.

[154] A. Nagabandi et al. “Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning”. In: *International Conference on Learning Representations (ICLR).* 2019.

[155] A. Nagabandi et al. “Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning”. In: *International Conference on Robotics and Automation (ICRA).* 2018.

[156] C. Nehaniv and K. Dautenhahn. “The Correspondence Problem”. In: *Imitation in Animals and Artifacts* (2002).

[157] A. Nguyen and T. Nguyen. “Graph-Based Statistical Language Model for Code”. In: *International Conference on Software Engineering (ICSE).* 2015.

[158] J. Ni, J. Li, and J. McAuley. “Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019.

[159] A. Nichol, J. Achiam, and J. Schulman. “On First-Order Meta-Learning Algorithms”. In: *arXiv preprint arXiv:1803.02999* (2018).

[160] M. Nita and D. Notkin. “Using Twinning to Adapt Programs to Alternative APIs”. In: *International Conference on Software Engineering (ICSE)*. 2010.

[161] M. Norouzzadeh et al. “A Deep Active Learning System for Species Identification and Counting in Camera Trap Images”. In: *Methods in Ecology and Evolution* (2020).

[162] V. Nygaard, E. Rødland, and E. Hovig. “Methods that Remove Batch Effects while Retaining Group Differences May Lead to Exaggerated Confidence in Downstream Analyses”. In: *Biostatistics* (2016).

[163] A. Orhan. “Robustness Properties of Facebook’s ResNeXt WSL Models”. In: *arXiv preprint arXiv:1907.07640* (2019).

[164] J. Park, J. Shin, and P. Fung. “Reducing Gender Bias in Abusive Language Detection”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2018.

[165] H. Parker and J. Leek. “The Practical Effect of Batch on Genomic Prediction”. In: *Statistical Applications in Genetics and Molecular Biology* (2012).

[166] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[167] X. Peng et al. “Moment Matching for Multi-Source Domain Adaptation”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[168] X. Peng et al. “VisDA: A Synthetic-to-Real Benchmark for Visual Domain Adaptation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[169] J. Peters, P. Bühlmann, and N. Meinshausen. “Causal Inference using Invariant Prediction: Identification and Confidence Intervals”. In: *Journal of the Royal Statistical Society (JRSS), Series B* (2016).

[170] S. Proksch, J. Lerch, and M. Mezini. “Intelligent Code Completion with Bayesian Networks”. In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* (2015).

[171] S. Proksch et al. “Evaluating the Evaluations of Code Recommender Systems: A Reality Check”. In: *International Conference on Automated Software Engineering (ASE)*. 2016.

[172] J. Quiñonero-Candela et al. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[173] A. Rajkomar, J. Dean, and I. Kohane. “Machine Learning in Medicine”. In: *The New England Journal of Medicine* (2019).

[174] S. Ravi and H. Larochelle. "Optimization as a Model for Few-Shot Learning". In: *International Conference on Learning Representations (ICLR).* 2017.

[175] V. Raychev, P. Bielik, and M. Vechev. "Probabilistic Model for Code with Decision Trees". In: *ACM SIGPLAN Notices* (2016).

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

[177] B. Recht et al. "Do ImageNet Classifiers Generalize to ImageNet?" In: *International Conference on Machine Learning (ICML).* 2019.

[178] M. Ren et al. "Meta-Learning for Semi-Supervised Few-Shot Classification". In: *International Conference on Learning Representations (ICLR).* 2018.

[179] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *arXiv preprint arXiv:1506.01497* (2015).

[180] J. Requeima et al. "Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes". In: *Advances in Neural Information Processing Systems (NeurIPS).* 2019.

[181] M. Reynolds et al. "Breeder Friendly Phenotyping". In: *Plant Science* (2020).

[182] D. Rezende, S. Mohamed, and D. Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *International Conference on Machine Learning (ICML).* 2014.

[183] N. Rhinehart and K. Kitani. "First-Person Activity Forecasting with Online Inverse Reinforcement Learning". In: *International Conference on Computer Vision (ICCV).* 2017.

[184] S. Richter et al. "Playing for Data: Ground Truth from Computer Games". In: *European Conference on Computer Vision (ECCV).* 2016.

[185] R. Robbes and M. Lanza. "How Program History Can Improve Code Completion". In: *International Conference on Automated Software Engineering (ASE).* 2008.

[186] E. Rolf, M. Jordan, and B. Recht. "Post-Estimation Smoothing: A Simple Baseline for Learning with Side Information". In: *International Conference on Artificial Intelligence and Statistics (AISTATS).* 2020.

[187] G. Ros et al. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016.

[188] A. Royer and C. Lampert. "Classifier Adaptation at Prediction Time". In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2015.

[189] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* (2015).

[190] F. Sadeghi and S. Levine. “CAD2RL: Real Single-Image Flight without a Single Real Image”. In: *Robotics: Science and Systems (RSS)*. 2017.

[191] S. Sagawa et al. “An Investigation of Why Overparameterization Exacerbates Spurious Correlations”. In: *International Conference on Machine Learning (ICML)*. 2020.

[192] S. Sagawa et al. “Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-case Generalization”. In: *International Conference on Learning Representations (ICLR)*. 2020.

[193] V. Sanh et al. “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter”. In: *arXiv preprint arXiv:1910.01108* (2019).

[194] A. Santoro et al. “Meta-Learning with Memory-Augmented Neural Networks”. In: *International Conference on Machine Learning (ICML)*. 2016.

[195] S. Santurkar, D. Tsipras, and A. Madry. “BREEDS: Benchmarks for Subpopulation Shift”. In: *International Conference on Learning Representations (ICLR)*. 2021.

[196] J. Schmidhuber. “Evolutionary Principles in Self-Referential Learning”. In: *Diploma thesis, Institut f. Informatik, Tech. Univ. Munich* (1987).

[197] S. Schneider et al. “Improving Robustness against Common Corruptions by Covariate Shift Adaptation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[198] J. Schulman et al. “Trust Region Policy Optimization”. In: *International Conference on Machine Learning (ICML)*. 2015.

[199] P. Sermanet, K. Xu, and S. Levine. “Unsupervised Perceptual Rewards for Imitation Learning”. In: *Robotics: Science and Systems (RSS)*. 2017.

[200] P. Sermanet et al. “Time-Contrastive Networks: Self-Supervised Learning from Video”. In: *International Conference on Robotics and Automation (ICRA)*. 2018.

[201] V. Shankar et al. “Do Image Classifiers Generalize Across Time?” In: *arXiv preprint arXiv:1906.02168* (2019).

[202] P. Sharma, D. Pathak, and A. Gupta. “Third-Person Visual Imitation Learning via Decoupled Hierarchical Control”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[203] Y. Shi et al. “Unmanned Aerial Vehicles for High-Throughput Phenotyping and Agronomic Research”. In: *PLOS One* (2016).

[204] H. Shimodaira. “Improving Predictive Inference under Covariate Shift by Weighting the Log-Likelihood Function”. In: *Journal of Statistical Planning and Inference (JSPI)* (2000).

[205] B. Shoichet. “Virtual Screening of Chemical Libraries”. In: *Nature* (2004).

[206] C. Shorten and T. Khoshgoftaar. “A Survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* (2019).

[207] A. Singh et al. “End-to-End Robotic Reinforcement Learning without Reward Engineering”. In: *Robotics: Science and Systems (RSS)*. 2019.

[208] L. Smith et al. “AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos”. In: *Robotics: Science and Systems*. 2020.

[209] J. Snell, K. Swersky, and R. Zemel. “Prototypical networks for few-shot learning”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017.

[210] C. Soneson, S. Gerster, and M. Delorenzi. “Batch Effect Confounding Leads to Strong Bias in Performance Estimates Obtained by Cross-Validation”. In: *PlOS One* (2014).

[211] T. Sterling and J. Irwin. “ZINC 15 – Ligand Discovery for Everyone”. In: *Journal of Chemical Information and Modeling* (2015).

[212] M. Sulc and J. Matas. “Improving CNN Classifiers by Estimating Test-time Priors”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[213] B. Sun, J. Feng, and K. Saenko. “Return of Frustratingly Easy Domain Adaptation”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2016.

[214] S. Sun, H. Shi, and Y. Wu. “A Survey of Multi-Source Domain Adaptation”. In: *Information Fusion* (2015).

[215] W. Sun et al. “Dual Policy Iteration”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[216] Y. Sun et al. “Test-Time Training with Self-Supervision for Generalization under Distribution Shifts”. In: *International Conference on Machine Learning (ICML)*. 2020.

[217] R. Sutton. “Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming”. In: *International Conference on Machine Learning (ICML)*. 1990.

[218] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[219] D. Swinney and J. Anthony. “How Were New Medicines Discovered?” In: *Nature reviews Drug discovery* (2011).

[220] M. Tabak et al. “Machine Learning to Classify Animal Species in Camera Trap Images: Applications in Ecology”. In: *Methods in Ecology and Evolution* (2019).

[221] R. Tatman. “Gender and Dialect Bias in YouTube’s Automatic Captions”. In: *Workshop on Ethics in Natural Language Processing*. 2017.

[222] J. Taylor et al. “RxRx1: An Image Set for Cellular Morphological Variation Across Many Experimental Batches”. In: *International Conference on Learning Representations (ICLR). AI for Social Good Workshop*. 2019.

[223] M. Taylor, J. Lukowski, and C. Anderton. “Spatially Resolved Mass Spectrometry at the Single Cell: Recent Innovations in Proteomics and Metabolomics”. In: *Journal of the American Society for Mass Spectrometry* (2021).

[224] D. Tellez et al. “Quantifying the Effects of Data Augmentation and Stain Color Normalization in Convolutional Neural Networks for Computational Pathology”. In: *Medical Image Analysis* (2019).

[225] D. Temel et al. “CURE-TSR: Challenging Unreal and Real Environments for Traffic Sign Recognition”. In: *arXiv preprint arXiv:1712.02463* (2017).

[226] K. Thorp et al. “High-Throughput Phenotyping of Crop Water Use Efficiency via Multispectral Drone Imagery and a Daily Soil Water Balance Model”. In: *Remote Sensing* (2018).

[227] S. Thrun and L. Pratt. *Learning to Learn*. Springer Science & Business Media, 1998.

[228] T. Tiecke et al. “Mapping the World Population One Building at a Time”. In: *arXiv preprint arXiv:1712.05839* (2017).

[229] J. Tobin et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[230] E. Todorov, T. Erez, and Y. Tassa. “MuJoCo: A Physics Engine for Model-Based Control”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2012.

[231] F. Torabi, G. Warnell, and P. Stone. “Behavioral Cloning from Observation”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2018.

[232] A. Torralba and A. Efros. “Unbiased Look at Dataset Bias”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011.

[233] A. Tow et al. “What Would You Do? Acting by Learning to Predict”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[234] E. Tzeng et al. “Adversarial Discriminative Domain Adaptation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[235] J. Ubbens et al. “AutoCount: Unsupervised Segmentation and Counting of Organs in Field Images”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[236] V. Vapnik. *Statistical Learning Theory*. Wiley New York, 1998.

[237] H. Venkateswara et al. “Deep Hashing Network for Unsupervised Domain Adaptation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[238] M. Veta et al. “Mitosis Counting in Breast Cancer: Object-Level Interobserver Agreement and Comparison to an Automatic Method”. In: *PlOS One* (2016).

[239] O. Vinyals et al. “Matching Networks for One Shot Learning”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2016.

[240] R. Volpi et al. “Generalizing to Unseen Domains via Adversarial Data Augmentation”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[241] A. Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *International Conference on Learning Representations (ICLR)*. 2019.

[242] A. Wang et al. "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[243] D. Wang et al. "Tent: Fully Test-Time Adaptation by Entropy Minimization". In: *International Conference on Learning Representations (ICLR)*. 2021.

[244] H. Wang et al. "Learning Robust Global Representations by Penalizing Local Predictive Power". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[245] S. Wang et al. "Weakly Supervised Deep Learning for Segmentation of Remote Sensing Imagery". In: *Remote Sensing* (2020).

[246] O. Wearn and P. Glover-Kapfer. "Camera-Trapping for Conservation: A Guide to Best-Practices". In: *WWF Conservation Technology Series* (2017).

[247] B. Weinstein. "A Computer Vision for Animal Ecology". In: *Journal of Animal Ecology* (2018).

[248] B. Wilson, J. Hoffman, and J. Morgenstern. "Predictive Inequity in Object Detection". In: *arXiv preprint arXiv:1902.11097* (2019).

[249] G. Wilson and D. Cook. "A Survey of Unsupervised Deep Domain Adaptation". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* (2020).

[250] E. Wong, L. Rice, and J. Kolter. "Fast is Better Than Free: Revisiting Adversarial Training". In: *International Conference on Learning Representations (ICLR)*. 2020.

[251] D. Worrall et al. "Harmonic Networks: Deep Translation and Rotation Equivariance". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[252] M. Wu et al. "Meta-Amortized Variational Inference and Learning". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2019.

[253] Y. Wu and K. He. "Group Normalization". In: *European Conference on Computer Vision (ECCV)*. 2018.

[254] Z. Wu et al. "MoleculeNet: A Benchmark for Molecular Machine Learning". In: *Chemical Science* (2018).

[255] K. Xiao et al. "Noise or Signal: The Role of Image Backgrounds in Object Recognition". In: *International Conference on Learning Representations (ICLR)*. 2021.

[256] M. Xie et al. "Transfer Learning from Deep Features for Remote Sensing and Poverty Mapping". In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2016.

[257] S. Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[258] H. Xiong et al. “TasselNetv2: In-Field Counting of Wheat Spikes with Context-Augmented Local Regression Networks”. In: *Plant Methods* (2019).

[259] K. Xu et al. “How Powerful are Graph Neural Networks?” In: *International Conference on Learning Representations (ICLR)*. 2018.

[260] Liu. Y. et al. “Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation”. In: *International Conference on Robotics and Automation (ICRA)*. 2018.

[261] Y. Yang and T. Hospedales. “A Unified Perspective on Multi-Domain and Multi-Task Learning”. In: *International Conference on Learning Representations (ICLR)*. 2015.

[262] Y. Yang et al. “Robot Learning Manipulation Action Plans by “Watching” Unconstrained Videos from the World Wide Web”. In: *AAAI Conference on Artificial Intelligence (AAAI)*. 2015.

[263] C. Yeh et al. “Using Publicly Available Satellite Imagery and Deep Learning to Understand Economic Well-being in Africa”. In: *Nature Communications* (2020).

[264] D. Yin et al. “A Fourier Perspective on Model Robustness in Computer Vision”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[265] T. Yu et al. “One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning”. In: *Robotics: Science and Systems (RSS)*. 2018.

[266] J. Zech et al. “Variable Generalization Performance of a Deep Learning Model to Detect Pneumonia in Chest Radiographs: A Cross-Sectional Study”. In: *PLOS Medicine* (2018).

[267] M. Zhang, S. Levine, and C. Finn. “MEMO: Test Time Robustness via Adaptation and Augmentation”. In: *arXiv preprint arXiv:2110.09506* (2021).

[268] M. Zhang et al. “Adaptive Risk Minimization: Learning to Adapt to Domain Shift”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[269] M. Zhang et al. “Deep Reinforcement Learning for Tensegrity Robot Locomotion”. In: *International Conference on Robotics and Automation (ICRA)*. 2017.

[270] M. Zhang et al. “SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning”. In: *International Conference on Machine Learning (ICML)*. 2019.

[271] R. Zhang et al. “MetaGAN: An Adversarial Approach to Few-Shot Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2018.

[272] Y. Zhou et al. “High-Throughput Screening of a CRISPR/Cas9 Library for Functional Genomics in Human Cells”. In: *Nature* (2014).

[273] J. Zhu et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: *International Conference on Computer Vision (ICCV)*. 2017.

Appendix A

Appendices for Distribution Shift in Supervised Learning

A.1 Additional Experimental Details for WILDS

Model architectures

We used standard model architectures for each dataset: ResNet and DenseNet for images [80, 97], DistilBERT for text [193], a Graph Isomorphism Network (GIN) for graphs [259], and Faster-RCNN [179] for detection.

Model hyperparameters

As our goal is high out of distribution (OOD) performance, we use a separate OOD validation set for early stopping and hyperparameter selection. Relative to the training set, this OOD validation set reflects a distribution shift similar to, but distinct from, the test set. For example, in iWildCam2020-wilds, the training, validation, and test sets each comprise photos from distinct sets of camera traps.

For each hyperparameter setting, we used early stopping to pick the epoch with the best OOD validation performance (as measured by the specified metrics for each dataset), and then we picked the model hyperparameters with the best early stopped validation performance. We found that this gave similar or slightly better OOD test performance than selecting hyperparameters using the in distribution (ID) validation set.

Using the OOD validation set for early stopping means that even if the training procedure does not explicitly use additional metadata, as in ERM, the metadata might still be implicitly (but mildly) used for model selection in one of two related ways. First, the metric might use the metadata directly (e.g., by computing the accuracy over different subpopulations defined in the metadata). Second, the OOD validation set is generally selected according to this metadata (e.g., comprising data from a disjoint set of domains as the training set). We expect that implicitly using the metadata in these ways should increase the OOD performance of each model. Nevertheless, as the results show, there are still large gaps between OOD and ID performance.

In general, we selected model hyperparameters with ERM and used the same hyperparameters for the other algorithm baselines (e.g., CORAL, IRM, or DRNN). For CORAL and IRM, we did a subsequent grid search over the weight of the penalty term, using the defaults from DomainBed [74]. Specifically, we tried penalty weights of $\{0.1, 1, 10\}$ for CORAL and penalty weights of $\{1, 10, 100, 1000\}$ for IRM. We fixed the step size hyperparameter for DRNN to its default value of 0.01 [192].

Replicates

We typically use a fixed train/validation/test split and report results averaged across 3 replicates (random seeds for model initialization and minibatch order), as well as the unbiased standard deviation over those replicates. There are three exceptions to this. For PovertyMap-wilds, we report results averaged over 5-fold cross validation, as model training is relatively fast on this dataset. For Camelyon17-wilds, results vary substantially between replicates, so we report results averaged over 10 replicates instead. Similarly, for CivilComments-wilds, we report results averaged over 5 replicates.

Baseline algorithms

For all classification datasets, we train models against the cross entropy loss. For the PovertyMap-wilds regression dataset, we use the mean squared error loss.

We adapted the implementations of CORAL from DomainBed [74]; IRM from the original authors [8]; and DRNN from the original authors [192]. We note that CORAL was originally proposed in the context of domain adaptation [213], where it was shown to substantially improve performance on standard domain adaptation benchmarks, and it was subsequently adapted for domain generalization [74].

Following these implementations, we use minibatch stochastic optimizers to train models under each algorithm, and we sample uniformly from each domain regardless of the number of training examples in it. This means that the CORAL and IRM algorithms optimize for their respective penalty terms plus a reweighted ERM objective that weights each domain equally (i.e., effectively upweighting minority domains). The DRNN objective is unchanged, as it still optimizes for the domain with the worst loss, but the uniform sampling improves optimization stability.

Both CORAL and IRM are designed for models with featurizers, i.e., models that first map each input to a feature representation and then predict based on the representation. To estimate the feature distribution for a domain, these algorithms need to see a sufficient number of examples from that domain in a minibatch. However, some of our datasets have large numbers of domains, making it infeasible for each minibatch to contain examples from all domains. For these algorithms, our data loaders form a minibatch by first sampling a few domains, and then sampling examples from those domains. For consistency in our experiments, we used the same total batch size for these algorithms and for ERM and DRNN, with a default of 8 examples per domain in each minibatch (e.g., if the batch size was 32, then in each minibatch we would have 8 examples × 4 domains).

For DRNN, each example in the minibatch is sampled independently with uniform probabilities across domains, and therefore each minibatch does not need to only comprise a small number of domains. We note that reweighting methods like DRNN are effective only when the training loss is non vanishing, which we achieve through early stopping [33, 192, 191].

Appendix B

Appendices for Adaptive Risk Minimization

B.1 Contrasting the Benchmarks Evaluated in ARM with Prior Benchmarks

For ARM, we primarily evaluated on problems for which unlabeled adaptation is feasible, helpful, and potentially crucial, inspired by important real world problem settings such as federated learning. Thus, the problems we focused on differ from prior work in domain shift, which have different implicit and explicit goals when designing and choosing benchmarks. In particular, many prior benchmarks assume the existence of a consistent input-output relationship across domains, for which various methods can be designed to try and better uncover this relationship. Compared to problems where adaptation is important, we can roughly characterize these benchmarks as having a conditional distribution $p_{y|\mathbf{x}}^{(d)}$ that is more stable across domains and thus does not depend as much on the marginal $p_{\mathbf{x}}^{(d)}$. As discussed, the less information the marginal provides about the conditional, the less we expect domain generalization strategies to improve over ERM [22]. And, indeed, DomainBed provides a comprehensive survey of domain generalization benchmarks and find that, though ERM is sometimes outperformed by certain methods on certain benchmarks, ERM is competitive with the state of the art on average across the benchmarks [74].

DomainBed also evaluated ARM-CML across the whole suite and found middling performance across most of the testbeds in the benchmark. This negative result provides further evidence that adaptation may not be well suited to these problems, at least in their standard formulations, and that different approaches are suited for different problems. Similarly, adaptation and ARM methods also do not improve performance on some of the WILDS domain generalization problems [113], potentially due to the marginal $p_{\mathbf{x}}^{(d)}$ not providing much information about $p_{y|\mathbf{x}}^{(d)}$, or other factors such as the lack of training domains or shared structure between domains. One potentially interesting result found was that ARM-CML did outperform ERM and all prior methods on one toy problem in DomainBed: the colored MNIST benchmark [8]. For a non adaptive model, the goal as originally proposed is to disregard color and learn the invariant relationship between digits and labels [8]. Irrespective of the original motivations, though, an adaptive model is in theory capable of learning a more flexible classification strategy for this problem, in that it may leverage information about the current domain in order to produce better predictions. Viewed this way, it becomes clear why ARM-CML can learn a more performant solution for the colored MNIST problem. This result on a toy problem provides further motivation for identifying and studying real world problems for which adaptation can be beneficial, alongside other benchmarks geared toward discovering invariances.

Specifically when viewing learning adaptation as a meta-learning problem, as in ARM, we may pose additional hypotheses about a problem's desired properties. For example, in meta-learning, each task is viewed as a "higher level" data point, and this generally motivates constructing many different tasks so as to prevent the learner from overfitting to the tasks. We extend this intuition to our work in that our problems have tens to hundreds of domains, whereas the benchmarks in DomainBed have between 3 to 6 domains [74]. Note that the overall dataset sizes are still comparable, so previous benchmarks typically also have orders of magnitude more data per domain. Depending on the scenario, it may be difficult to either collect data from many domains, or conversely it may be difficult to collect many data points from any single domain. For example, the FEMNIST dataset naturally contains hundreds of users each contributing at most hundreds of examples, but it would be difficult to collect orders of magnitude more data from any given user. These practical considerations should also factor into the choice of algorithm for solving any particular problem.

Prior testbeds used in group distributionally robust optimization (DRO) typically also contain a small number of groups [192], and these testbeds also have a couple of other important differences. First, group DRO testbeds typically use the same training and test groups and measure worst case performance, which differs from domain generalization, meta-learning, and most problems considered in this work, which construct or hold out disjoint sets of domains for testing. Second, prior group DRO testbeds use label information to construct groups, in that data within each group will all have the same label. This is not an issue for non adaptive models, however, classification in this setup becomes much easier for adaptive models and particularly if training with an ARM method, as the model simply needs to learn to adapt to output a constant label. Thus, we were motivated to identify and set up problems which are distinct from both prior work in domain generalization and group DRO, in order to properly evaluate ARM and prior methods in settings for which adaptation is beneficial.

B.2 Additional Dataset and Experimental Setup Details

B.3 Additional Experimental Details

Code for Table 5.1 results is available from `https://github.com/henrikmarklund/arm`.

In our experiments, we use several different computing clusters with either NVIDIA Titan X Pascal, RTX 2080 Ti, or V100 GPUs, and all experiments use 1 GPU. When reporting our results, we run each method across three seeds and reported the mean and standard error across seeds. Standard error is calculated as the sample standard deviation divided by $\sqrt{3}$. We checkpoint models after every epoch of training, and at test time, we evaluate the checkpoint with the best worst case validation accuracy. Training hyperparameters and details for how we evaluate validation and test accuracy are provided for each experimental domain below. All hyperparameter settings were selected in preliminary experiments using validation accuracy only.

Rotated MNIST details

We trained all models for 200 epochs with mini batch sizes of 300. We use Adam updates with learning rate 0.0001. We construct an additional level of mini batching for our method such that the batch dimensions of the data mini batches are 6×50 rather than just 300, and each of the inner mini batches contain examples from the same rotation. We refer to the outer batch dimension as the *meta batch size* and the inner dimension as the batch size. All methods are still trained for the same number of epochs and see the same amount of data. DRNN uses an additional learning rate hyperparameter for their robust loss, which we set to 0.01 across all experiments [192].

We compute validation accuracy every 10 epochs. We estimate validation accuracy on each rotation by randomly sampling 300 of the held out 6000 original training points and applying the specific rotation, resampling for each validation evaluation. This is effectively the same procedure as the test evaluation, which randomly samples 3000 of the 10000 test points and applies a specific rotation.

We retain the original $28 \times 28 \times 1$ image dimensionality, and we divide inputs by 256. We use convolutional neural networks for all methods with varying depths to account for parameter fairness. For all methods that do not use a context network, the network has four convolution layers with 128 5×5 filters, followed by 4×4 average pooling, one fully connected layer of size 200, and a linear output layer. Rectified linear unit (ReLU) nonlinearities are used throughout, and BN [100] is used for the convolution layers. The first two convolution layers use padding, and the last two convolution layers use 2×2 max pooling. For ARM-CML and the context ablation, we remove the first two convolution layers for the prediction network, but we incorporate a context network. The context network uses two convolution layers with 64 filters of size 5 × 5, with ReLU nonlinearities, BN, and padding, followed by a final convolution layer with 12 5 × 5 filters with padding.

FEMNIST details

We keep all hyperparameters the same as for rotated MNIST, except we set the meta batch size for ARM methods to be 2, and we use stochastic gradient updates with learning rate 0.0001, momentum 0.9, and weight decay 0.0001. For DANN, we use Adam updates with learning rate 0.0001 as stochastic gradient updates were unsuccessful for this method. We compute validation accuracy every epoch by iterating through the data of each validation user once, and this procedure is the same as test evaluation. Note that all methods will sometimes receive small batch sizes as each user's data size may not be a multiple of 50. Though this may affect ARM methods, we demonstrate in the streaming experiments that ARM-CML and ARM-BN can adapt using small batch sizes. The network architectures are the same as the architectures used for rotated MNIST, except that, when applicable, the last layer of the context network has only 1 filter of size 5 × 5.

CIFAR-10-C and Tiny ImageNet-C details

In these experiments, we use a support size of 100 and meta batch size of 3. For CIFAR-10-C, we use the same convolutional network architecture as for rotated MNIST and FEMNIST, except for the first layer which needs to be modified to handle RGB images. For Tiny ImageNet-C, we fine tune ResNet-50 [80] models pretrained on ImageNet. The context ablation and ARM-CML additionally use small convolutional context networks, and the learned loss ablation and ARM-LL use small fully connected loss networks. For this domain, we further incorporate BN adaptation into the context ablation and ARM-CML, as we found this technique to generally be very helpful when dealing with image corruptions. The images are first normalized by the ImageNet mean and standard deviation. For CIFAR-10-C, we train models from scratch for 100 epochs, and for Tiny ImageNet-C we fine tune for 50 epochs. We use stochastic gradient updates with learning rate 0.01, momentum 0.9, and weight decay 0.0001. We evaluate validation accuracy after every epoch and perform model selection based on the worst case accuracy over domains. We perform test evaluation by randomly sampling 3000 images from each domain and computing worst case and average accuracy across domains.

B.4 Additional Experiments

Additional comparison to unsupervised domain adaptation

In Table B.1, we provide an additional comparison to unsupervised domain adaptation (UDA). A number of methods have been proposed for UDA, and for simplicity, we compare to

	Rotated MNIST	
Method	**WC**	**Avg**
DANN (DG)	78.8 ± 0.8	94.9 ± 0.1
DANN (UDA)	82.4 ± 1.6	94.9 ± 0.2
ARM-CML	$\mathbf{88.0 \pm 0.8}$	$\mathbf{96.3 \pm 0.4}$

Table B.1: Comparing to DANN [62] as an unsupervised domain adaptation (UDA) method, in which the particular test domain is known at training time. Note that this involves retraining models for each test evaluation, and ARM-CML is still more performant by leveraging meta-training and adaptation.

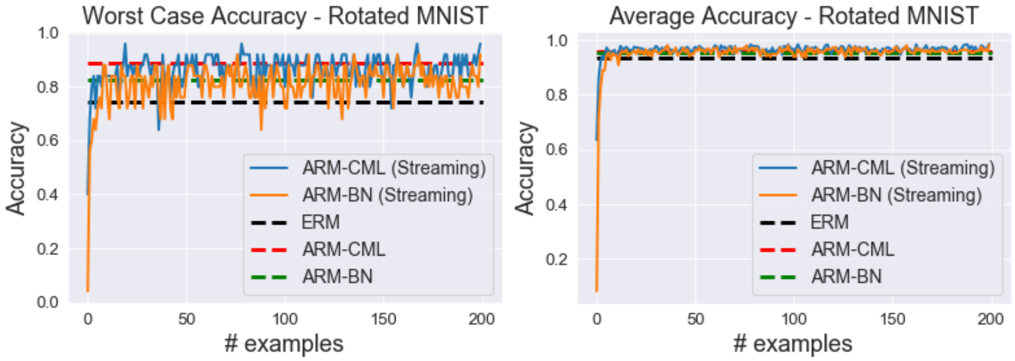


Figure B.1: On rotated MNIST, ARM methods reach strong performance in the streaming setting after fewer than 10 data points, again despite meta-training with batch sizes of 50.

DANN [62], which we evaluated as a domain generalization algorithm but was originally proposed for UDA. When faced with multiple test shifts, UDA methods run training separately for each shift, as they assume access to unlabeled samples from the test distribution at training time. For rotated MNIST, where there are 14 test groups, evaluating DANN as a UDA method involved 42 separate training runs, as we still used 3 training seeds per test evaluation. We see Table B.1 that DANN in this setting performs better in terms of worst case accuracy, which is not surprising given each model's ability to specialize to a particular test domain. However, by leveraging meta-training and adaptation, ARM-CML still performs the best on this problem.

Additional results with loosened assumptions

In Figure B.1, we include results for the rotated MNIST problem in the test streaming setting. We can see the same general trend as for Tiny ImageNet-C, where the models

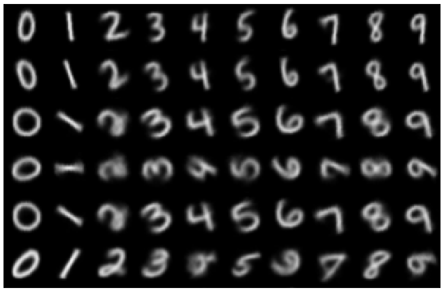


Figure B.2: VAE samples conditioned on different values of y (x axis) and z (y axis). The VAE learns to use z to represent rotations.

Method	**WC**	**Avg**
ERM	74.5 ± 1.4	93.6 ± 0.4
TTT	$\mathbf{81.1 \pm 0.3}$	$\mathbf{95.4 \pm 0.1}$
ARM-CML	$\mathbf{81.7 \pm 0.3}$	$\mathbf{95.2 \pm 0.3}$

Table B.2: Using learned domains, ARM-CML outperforms ERM and matches the performance of TTT on rotated MNIST. This result may be improved by techniques for learning more diverse domains.

trained via ARM methods are able to adapt successfully, and in this easier domain these models require fewer than 10 test inputs to reach their performances in the batch setting, where adaptation is performed with batches of 50 points.

In the case of unknown domains, one option is to use unsupervised learning techniques to discover domain structure in the training data. To test this option, we focus on rotated MNIST and ARM-CML, which performs the best on this dataset, and train a variational autoencoder (VAE) [110, 182] with discrete latent variables [102, 141] using the training images and labels. We define the latent variable z to be Categorical with 12 possible discrete values, which we purposefully choose to be smaller than the number of rotations. The VAE is not given any information about the ground truth d; however, we weakly encode the notion that z is independent of y by conditioning the decoder on the label. We use the VAE inference network to assign domains to the training data, and we run ARM-CML using these learned domains. In Table B.2, we see that ARM-CML in this setting outperforms ERM and is competitive with TTT, which as discussed earlier encodes a strong inductive bias, via rotation prediction, for solving this task. Figure B.2 visualizes samples from the VAE for different values of y and z, which shows that the VAE learns to encode rotation information using z. This result suggests that, when domain information is not provided, a viable approach may be to learn domains which then enables the use of ARM methods.

Appendix C

Appendices for Marginal Entropy Minimization with One Test Point

C.1 Experimental Setup and Protocol

We select hyperparameters using the four disjoint validation corruptions provided with CIFAR-10-C and ImageNet-C [84]. As the other benchmarks are only test sets and do not provide validation sets, we use the same hyperparameters found using the corruption validation sets and do not perform any additional tuning. For the ResNet models that we evaluate, we use stochastic gradients as the update rule G; for ResNet-26 models, we set the number of augmentations $B = 32$ and the learning rate $\eta = 0.005$; and for ResNet-50 models, we set $B = 64$ and $\eta = 0.00025$. For the robust vision transformer, we use AdamW [136] as the update rule G, with learning rate $\eta = 0.00001$ and weight decay 0.01, and $B = 64$.

In the CIFAR evaluation, we compare to TTT, which, as noted, can also be applied to single test inputs but requires a specialized training procedure [216]. Thus, the ResNet-26 model we use for our method closely follows the modifications that TTT propose, in order to provide a fair point of comparison. In particular, TTT elect to use group normalization [253] rather than BN, thus single point BN adaptation is not applicable for this model architecture. As noted before, TTT also requires the joint training of a separate rotation prediction head, thus further changing the model architecture, while MEMO directly adapts the standard pretrained model.

The TTA results are obtained using the same AugMix augmentations as for MEMO. The single point BN adaptation results use $N = 16$, as suggested by prior work [197]. As noted, the BN adaptation results (using multiple test points) are obtained using $N = 256$ as the prior strength and batches of 256 test inputs for adaptation. For Tent, we use the hyperparameters suggested in prior work [243]: we use stochastic gradients with learning rate 0.00025 and momentum 0.9, the adaptation is performed with test batches of 64 inputs, and the method is run online, i.e., prediction and adaptation occur simultaneously and the model is allowed to continuously adapt through the entire test epoch. Since Tent did not experiment with transformer models, we also attempted to run Tent with Adam [109] and AdamW [136] and various hyperparameters for the RVT*-small model; however, we found that this generally resulted in worse performance than using stochastic gradient updates with the aforementioned hyperparameters.

We obtain the baseline ResNet-50 parameters directly from the `torchvision` library. The parameters for the ResNet-50 trained with DeepAugment and AugMix are obtained from `https://drive.google.com/file/d/1QKmc_p6-qDkh51WvsaS9HKFv8bX5jLnP`. The parameters for the ResNet-50 trained with moment exchange and CutMix are obtained from `https://drive.google.com/file/d/1cCvhQKV93pY-jj8f5jITywkB9EabiQDA`. The parameters for the small robust vision transformer (RVT*-small) model are obtained from `https://drive.google.com/file/d/1g40huqDVthjS2H5sQV3ppcfcWEzn9ekv`.

C.2 Additional Experiments

In this section, we analyze the importance of using augmentations during adaptation, study the tradeoffs between efficiency and accuracy for MEMO, and present results with ResNext101 models [257, 143, 163] on ImageNet-A.

Analysis on augmentations

One may first wonder: are augmentations needed in the first place? In the test time robustness setting when only one test point is available, how would simple entropy minimization fare? We answer this question in Table C.1 by evaluating the episodic variant of Tent (i.e., with model resetting after each batch) with a test batch size of 1. This approach is also analogous to a variant of MEMO that does not use augmentations, since for one test point and no augmented copies, conditional and marginal entropy are the same. Similar to MEMO, we also incorporate single point BN adaptation with $N = 16$, in place of the standard BN adaptation that Tent typically employs using batches of test inputs. The results in Table C.1 indicate that entropy minimization on a single test point generally provides marginal performance gains beyond just single point BN adaptation. This empirically shows that using augmentations is important for achieving the reported results.

We also wish to understand the importance of the choice of augmentation functions $\mathcal{A}$. As mentioned, we used AugMix [85] in the previous experiments as it best fit our criteria: AugMix requires only the input $\mathbf{x}$, and randomly sampled augmentations lead to diverse augmented data points. A simple alternative is to instead use the "standard" set of augmentations commonly used in ImageNet training, i.e., random resized cropping and random horizontal flipping. We evaluate this ablation of using MEMO with standard augmentations also on the CIFAR-10 test sets, again with the same hyperparameter values. From the results in Table C.2, we can see that MEMO is still effective with simpler augmentation functions. This is true particularly for the cases where there is no test shift, as in the original CIFAR-10 test set, or subtle shifts as in CIFAR-10.1; however, for the more severe and systematic

	ImageNet-C	ImageNet-R	ImageNet-A
	mCE ↓	Error (%)	Error (%)
Baseline ResNet-50 [80]	76.7	63.9	100.0
+ Single point BN	71.4 (−5.3)	61.1 (−2.8)	99.4 (−0.6)
+ MEMO (ours)	69.9 (−6.8)	58.8 (−5.1)	99.1 (−0.9)
+ Tent (episodic, batch size 1) [243]	70.4 (−6.3)	60.0 (−3.9)	99.3 (−0.7)
+ DeepAugment+AugMix [89]	53.6	53.2	96.1
+ Single point BN	51.3 (−2.3)	51.2 (−2.0)	95.4 (−0.7)
+ MEMO (ours)	**49.8 (−3.8)**	**49.2 (−4.0)**	94.8 (−1.3)
+ Tent (episodic, batch size 1)	50.7 (−2.9)	50.7 (−2.5)	95.2 (−0.9)
+ MoEx+CutMix [122]	74.8	64.5	91.9
+ Single point BN	71.0 (−3.8)	62.6 (−1.9)	91.1 (−0.8)
+ MEMO (ours)	69.1 (−5.7)	59.4 (−3.3)	**89.0 (−2.9)**
+ Tent (episodic, batch size 1)	69.9 (−4.9)	61.7 (−2.8)	90.6 (−1.3)
RVT*-small [144]	49.4	52.3	73.9
+ Single point BN	48.0 (−1.4)	51.1 (−1.2)	74.4 (+0.5)
+ MEMO (ours)	**40.6 (−8.8)**	**43.8 (−8.5)**	**69.8 (−4.1)**
+ Tent (episodic, batch size 1)	47.9 (−1.5)	50.9 (−1.4)	74.4 (+0.5)

Table C.1: Evaluating the episodic version of Tent with a batch size of 1, which corresponds to a simple entropy minimization approach for the test time robustness setting. This approach also uses single point BN adaptation, and entropy minimization does not provide much additional gain.

CIFAR-10-C shifts, using heavier AugMix data augmentations leads to greater performance gains over the standard augmentations. Furthermore, this ablation was conducted using the ResNet-26 model, which was trained with standard augmentations – for robust models, AugMix may offer greater advantages at test time since these models were exposed to heavy augmentations during training.

Analyzing the tradeoff between efficiency and accuracy

In Figure C.1, we analyze the % test error of MEMO adaptation on ImageNet-R as a function of the efficiency of adaptation, measured in seconds per evaluation. We achieve various tradeoffs by varying the number of augmented copies $B = \{1, 2, 4, 8, 16, 32, 64, 128\}$. We note that small values of B such as 4 and 8 can already provide significant performance gains, indicating that a practical tradeoff between efficiency and accuracy is possible. For large B, the wall clock time is dominated by computing the augmentations – in our implementation, we do not compute augmentations in parallel, though in principle this is possible for AugMix

	CIFAR-10 Error (%)	CIFAR-10.1 Error (%)	CIFAR-10-C Average Error (%)
ResNet-26 [80]	9.2	18.4	22.5
+ MEMO (ours)	7.3 (−1.9)	14.7 (−3.7)	**19.6** **(−2.9)**
− AugMix [85] + standard augs	**7.2** **(−2.0)**	**14.6** **(−3.8)**	20.2 (−2.3)

Table C.2: Ablating the augmentation functions to test standard augmentations (random resized cropping and horizontal flips). When changing the augmentations used, the post adaptation performance generally does not change much.

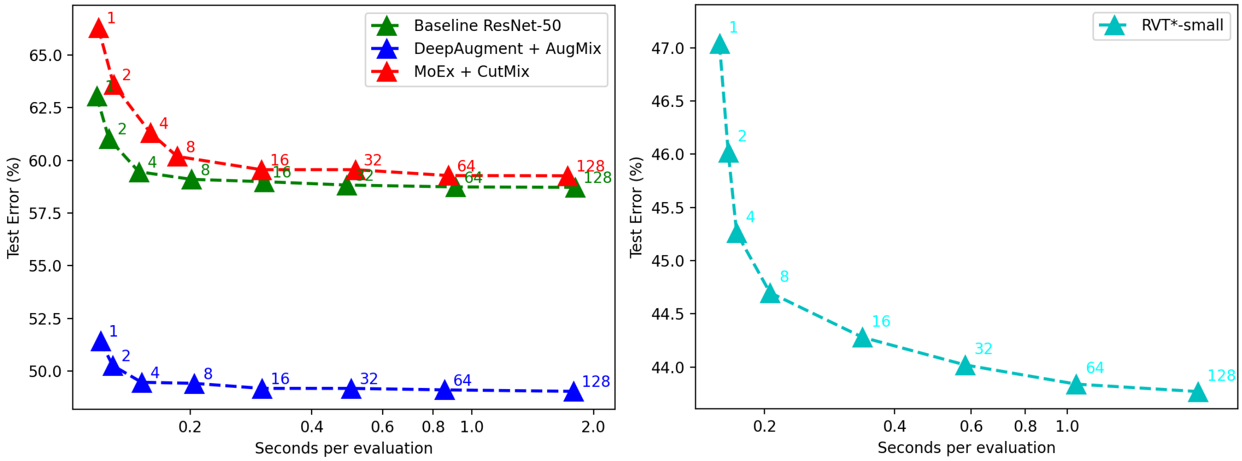


Figure C.1: Plotting MEMO efficiency as seconds per evaluation (x axis) and % test error on ImageNet-R (y axis) for the ResNet-50 models (left) and RVT*-small (right) while varying $B = \{1, 2, 4, 8, 16, 32, 64, 128\}$. Note the log scale on the x axis.

and should improve efficiency overall. These experiments used four Intel Xeon Skylake 6130 CPUs and one NVIDIA TITAN RTX GPU.

Evaluating ResNext101 models on ImageNet-A

ResNext-101 models [257] have been found to achieve higher accuracies on ImageNet-A [86], particularly when trained with massive scale weakly supervised pretraining [143, 89]. In this section, we evaluate whether MEMO can successfully adapt these models and further improve performance on this challenging test set. We use the same hyperparameters as for the robust vision transformer with no additional tuning: AdamW [136] as the update rule G, learning rate $\eta = 0.00001$, weight decay 0.01, and $B = 32$ due to memory limits. We obtain the baseline ResNext-101 (32x8d) parameters, pretrained on ImageNet, directly from

	ImageNet-A Error (%)
Baseline ResNext-101 (32x8d) [257]	90.0
+ TTA	83.2 (−6.8)
+ Single point BN	88.8 (−1.2)
+ MEMO (ours)	84.3 (−5.7)
+ WSL on billions of images [143]	54.9
+ TTA	49.1 (−5.8)
+ Single point BN	58.9 (+4.0)
+ MEMO (ours)	**43.2** **(−11.7)**

Table C.3: ImageNet-A results for the ResNext-101 models.

the `torchvision` library. We also evaluate a ResNext-101 (32x8d) pretrained with weakly supervised learning (WSL) on billions of Instagram images [143], and we obtained the parameters from `https://download.pytorch.org/models/ig_resnext101_32x8-c38310e5.pth`. For the WSL model, we did not use single point BN adaptation for MEMO as we found this technique to be actually harmful to performance, and this corroborates previous findings [197].

Table C.3 summarizes the results. We can see that, similar to Table 6.2, Both TTA and MEMO significantly improve upon the baseline model evaluation. TTA performs best for the baseline ResNext-101 model. However, MEMO ultimately achieves the best accuracy by a significant margin, as it is more successful at adapting the WSL model, which has a much higher accuracy. This suggests that combining MEMO with other large pretrained models may be an interesting direction for future work.

C.3 Full CIFAR-10-C and ImageNet-C results

In the following tables, we present test results broken down by corruption and level for CIFAR-10-C. We omit joint training and TTT because these results are available in prior work [216]. Our test results for ImageNet-C can be obtained from `https://docs.google.com/spreadsheets/d/1U5hUIJ1sRoMGwZY2WI8vH0VJTXvdX5vHMy6nej4PHE4`.

	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
ResNet-26	48.4	44.8	50.3	24.1	47.7	24.5	24.1	24.1	33.1	28.0	14.1	29.7	25.6	43.7	28.3
+ TTA	43.4	39.6	42.9	28.3	44.7	26.3	26.3	21.4	28.5	23.3	12.1	32.9	21.7	43.2	21.7
+ MEMO (ours)	43.5	39.8	43.3	26.4	44.4	25.1	25.0	20.9	28.3	22.8	11.9	28.3	21.1	42.8	21.7

Table C.4: Test error (%) on CIFAR-10-C level 5 corruptions.

	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
ResNet-26	43.8	37.2	39.3	14.8	48.0	19.9	18.7	22.0	24.9	15.1	11.4	16.8	19.1	27.9	24.9
+ TTA	39.5	32.0	31.8	15.4	45.0	20.9	20.2	18.9	21.7	12.9	9.3	16.8	17.7	25.7	18.9
+ MEMO (ours)	39.7	32.3	32.2	14.7	45.0	20.0	19.2	18.7	21.1	12.5	9.3	15.2	16.9	25.2	18.9

Table C.5: Test error (%) on CIFAR-10-C level 4 corruptions.

	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
ResNet-26	40.0	33.8	26.4	11.5	37.3	20.0	16.6	20.0	24.7	12.2	10.5	13.6	15.0	18.4	22.7
+ TTA	34.3	27.7	20.3	11.3	32.9	20.7	16.7	16.3	21.1	9.8	8.5	12.5	13.7	14.5	17.2
+ MEMO (ours)	34.4	27.9	20.5	10.8	32.8	19.8	16.1	16.1	20.9	9.6	8.6	11.7	13.2	14.5	17.2

Table C.6: Test error (%) on CIFAR-10-C level 3 corruptions.

	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
ResNet-26	30.1	21.8	21.1	9.7	38.3	15.3	13.8	21.2	17.6	10.5	9.7	11.6	12.9	15.4	21.3
+ TTA	25.3	16.9	15.8	8.5	33.8	15.3	13.7	17.8	14.5	8.7	7.8	10.0	11.0	12.0	16.1
+ MEMO (ours)	25.3	16.9	15.9	8.4	33.5	14.6	13.0	17.7	14.4	8.5	7.7	9.6	10.7	11.9	16.2

Table C.7: Test error (%) on CIFAR-10-C level 2 corruptions.

	gauss	shot	impul	defoc	glass	motn	zoom	snow	frost	fog	brit	contr	elast	pixel	jpeg
ResNet-26	20.8	16.5	15.8	9.2	38.9	11.8	12.8	13.9	13.4	9.7	9.4	9.6	13.1	12.0	16.4
+ TTA	15.8	12.8	11.8	7.3	35.1	10.8	12.5	11.0	10.8	7.4	7.4	7.7	11.4	9.2	12.5
+ MEMO (ours)	16.1	12.9	11.9	7.4	34.7	10.4	12.1	11.0	10.7	7.4	7.3	7.5	10.9	9.2	12.5

Table C.8: Test error (%) on CIFAR-10-C level 1 corruptions.