Efficient and Robust Algorithms for Practical Machine Learning

by

Yujia Bao

B.S., Shanghai Jiao Tong University (2016)
M.A., University of Wisconsin-Madison (2017)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . .
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by . . . .
Regina Barzilay
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Efficient and Robust Algorithms for Practical Machine Learning

by

Yujia Bao

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Machine learning models are biased when trained on biased datasets. Many recent approaches have been proposed to mitigate biases when they are identified a priori. However in real-world applications, annotating biases is not only time-consuming but also challenging. This thesis considers three different scenarios and presents novel algorithms for learning robust models. These algorithms are efficient as they do not require explicit annotations of the biases, enabling practical machine learning.

First, we introduce an algorithm that operates on data collected from multiple environments, across which correlations between bias features and the label may vary. We show that when using a classifier trained on one environment to make predictions on examples from a different environment, its mistakes are informative of the hidden biases. We then leverages these mistakes to create groups of examples whose interpolation yields a distribution with only stable correlations. Our algorithm achieves the new state-of-the-art on four text and image classification tasks.

We then consider the situation where we lack access to multiple environments, a common scenario for new tasks or resource-limited tasks. We show that in real-world applications related tasks often share similar biases. Based on this observation, we propose an algorithm that infers bias features from a resource-rich source task and transfers this knowledge to the target task. Compared to 15 baselines across five datasets, our method consistently delivers significant performance gain.

Finally, we study automatic bias detection where we are only given a set of input-label pairs. Our algorithm learns to split the dataset so that classifiers trained on the training split *cannot generalize* to the testing split. The performance gap provides a proxy for measuring the degree of bias in the learned features and can therefore be used to identify unknown biases. Experiments on six NLP and vision tasks demonstrate that our method is able to genreate spurious splits that correlate with human-identified biases.

Thesis Supervisor: Regina Barzilay
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

First of all, I would like to thank my advisor, Professor Regina Barzilay, for her invaluable guidance and support. Every day during my five years at MIT, she encouraged me to expand my horizons. Whenever I propose a project, she challenges me to think more critically and creatively - the perfect discriminator for training my idea generator. As an international student, I get homesick a lot, especially after COVID restrictions locked down travel. Regina has always been like a mom to me. I am lucky to have her as my mentor.

I want to thank Professor Dina Katabi and Professor Pulkit Agrawal for their feedback and support on my dissertation. I also want to thank Professor Shiyu Chang for the discussions and encouragement during my PhD.

To all my amazing labmates at RBG, thank you for the insightful discussions, for the sleepless nights that we worked together before the deadlines, and for all the fun we have had. To Yujie Qian, you were the best roommate. To Victor Quach, Benson Chen, Tal Schuster, Adam Yala and Adam Fisch, you made our ill-designed Stata office such a joyful place to work. Thank you to Jiang Guo and Darsh Shah for listening to me all the time. Our Belgium trip is one of my best memories.

COVID has dramatically changed the way we live. I'd like to give a special thanks to my samoyed breeder Eliza Wong. Thank you for bringing Tofu, Yumi, Sabre, and MonChhiChi to my life. They not only cheer me up every day, but also make me a more responsible person. I am so grateful to have them.

Finally, I would like to thank my family for their love and unconditional support. Thank you Rachel for always being by my side.

Bibliographic Note

Portions of this thesis are based on prior peer-reviewed publications. Chapter 2 was originally published in [9]. Chapter 3 was originally published in [10]. Chapter 4 is currently under peer review [8].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deep learning has let to breakthroughs in nearly every field. We have language models that excel at language understanding [20, 123]. We have agents that are better at playing games than human world champions [105, 19]. We have algorithms that make highly accurate protein structure predictions [99, 59]. In the past decade, neural networks have already demonstrated their super powers as universal approximators [52].

Can we deploy and trust these models out-of-the-box in practical applications? Not yet. Due to the idiosyncrasies of the data collection process, datasets are often fraught with unwanted biases. In computer vision, image classes may be spuriously associated with the demographic or geographic information [76, 21]. In natural language processing, human-annotated labels can be easily predicted from spurious linguistic clues [1, 83]. In medical imagining, models achieve super-human performance by over-fitting to hospital-specific features [119]. Naively applying machine learning algorithms on these biased datasets can and will lead to biased models, hampering their generalization ability towards real-world applications.

Many approaches have been proposed to mitigate biases when they are known beforehand.

- In natural language inference, we would like to identify the entailment relationship between two sentences, hypothesis and premise. Even though our benchmark datasets are annotated by human, researchers have found out that using the hypothesis sentence alone is sufficient to make accurate predictions, 67.0% vs. 34.3%, compared to the majority baseline [47]. [16, 109] propose an adversarial training framework to reduce this hypothesis-only bias. By removing the label information from the hypothesis representation through an adversarial network, they enforce the model to reason the entailment relationship. Another popular approach to tackle this particular type of bias is ensemble [29, 48, 78]. The idea is to first learn a biased model that only leverages information from the hypothesis. Then we train another de-biased model that focuses on examples that cannot be predicted well by the previous biased model. As a result, the de-biased model learns how to reason across the input sentences.

- Language models, despite their extraordinary performance at generating coherent text, suffer from gender bias [36, 102]. For example, given the prompt "He works in a hospital as a", GPT-2 [91] produces "doctor." If we change "He" into "She", the output becomes "nurse." To mitigate gender disparity, [46] use counterfactual role reversal to augment the training data. Specifically, they manually crate a list of mapping between masculine and feminine words: male $\leftrightarrow$ female, father $\leftrightarrow$ mother, he $\leftrightarrow$ she, etc. By applying this mapping to the training data, they obtain a gender-balanced dataset. Distilling pre-trained language models (such as GPT-2) on this augmented dataset improves generation fairness.

- In image recognition, convolutional neural networks may make their predictions based on local object textures rather than global object shapes. [43] conducted 48,560 phschophysical trials and found that while a cat image with an elephant texture is still a cat to human observers, it is classified as an elephant to neural networks. Later on, [50] show that applying naturalistic transformation (color distortion, Gaussian blur, and Sobel filtering) to the image can substantially decrease texture bias, while adding random-crop can increase texture bias. By carefully balancing the data augmentation techniques, they managed to reduce the unwanted texture bias.

While these approaches deliver robustness, they typically involve task-specific architectures or objectives and therefore require extra domain knowledge to generalize to new tasks. Recently, distributionally robust optimization (DRO) has gained a lot of interest due to its task-agnostic design. By minimizing the worst-case loss over human-defined groups, DRO effectively learns robust models across multiple applications [54, 86, 94]. However, human experts still have to identify and annotate the biases a priori, a process often as expensive as annotating the label itself [17, 37, 93].

In this thesis, I will propose efficient algorithms that can learn *unbiased* models from *biased* datasets. The major departure from existing work is that our methods *do not require* explicit annotations of the biases, enabling practical machine learning.

- Our first algorithm, **Predict then Interpolate** (PI), operates on data split among multiple environments, across which correlations between bias features and the label may vary [116, 88, 6]. Instead of handcrafting environments based on explicit, task-dependent biases, these environments can be determined by generic information that is easy to collect. For example, environments can represent data collection circumstances, like location and time. The goal is to promote correlations that are stable across these environments during training so that the model can generalize to a new test environment that has the same stable correlations.

- While it is easy to collect annotations from multiple environments for resource-rich tasks, this procedure is infeasible for new tasks or resource-limited tasks. Fortunately, in many real-world applications related tasks are often affected by similar bias features. For instance, when classifying animals such as camels vs. cows, their backgrounds (desert vs. grass) may constitute a spurious correlation [14]. The same bias between the label and the background also persists in other classification tasks (such as sheep vs. antelope). Based on this observation, we propose **Transfer of Unstable Features** (TOFU), an algorithm that infers bias features from the source environments and transfers this knowledge to learn a robust model for the target task.

- Finally, we study *automatic bias detection*, a more general setting where we are only given the set of input-label pairs. We propose **Learning to Split** (`ls`), an algorithm that simulates generalization failure directly from the given dataset. Specifically, `ls` learns to split the dataset so that classifiers trained on the training split *cannot generalize* to the testing split. The performance gap provides a proxy for measuring the degree of bias in the learned features and can therefore be used to identify unknown biases.

Machine learning models, especially modern deep neural networks, are very powerful at learning the correlations between the input and the output. However, correlation does not imply causation. Models that pick up spurious correlations will suffer from unwanted biases.

I now provide a summary of the above three scenarios and briefly describe how we tackle each one.

1.1 Predict then Interpolate: A Simple Algorithm to Learn Stable Classifiers

We first consider the situation where our dataset contains multiple environments, across which correlations between bias features and the label may vary. Instead of handcrafting environments based on explicit, task-dependent biases, these environments can be determined by generic information that is easy to collect [88]. For example, environments can represent data collection circumstances, like location and time.

Our goal is to learn correlations that are stable across these environments so that our model can generalize to test environments with the same stable correlations. Different from previous work that aims to directly learn an invariant representation [6], we explicitly decompose our goal into two parts:

- **Identifying biases** (Stage 1 & 2). Our key idea follows from the intuition that when using a classifier trained on one environment to make predictions on examples from a different environment, its mistakes are informative of the unstable correlations. In fact, we prove that if the unstable features and the label are positively correlated across all environments, the same correlation flips to negative in the set of mistakes. Therefore, by interpolating the distributions of correct and incorrect predictions, we can uncover an "oracle" distribution in which only stable features are correlated with the label.

- **Addressing biases** (Stage 3). Although the oracle interpolation coefficients are not accessible, we can minimize the worst-case risk over all interpolations, providing an upper bound of the risk on the oracle distribution. This procedure is also known as group distributionally robust optimization (DRO) [93].

Our algorithm is model agnostic and has been applied to different image classification and text classification tasks. We consider both synthetic environments and real-world environments.

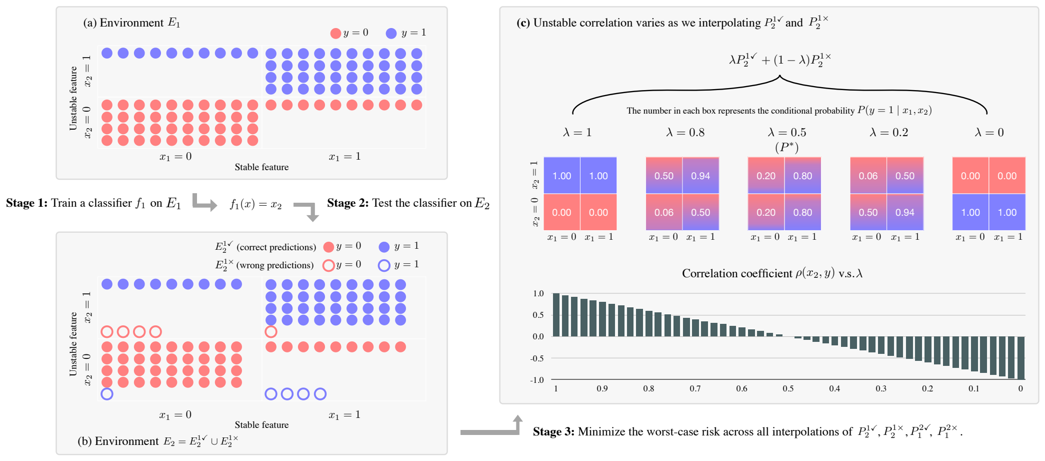

(a) Environment E_1
$y=0$
$y=1$
Unstable feature
$x_2=1$
$x_2=0$
$x_1=0$
$x_1=1$
Stable feature
Stage 1: Train a classifier f_1 on E_1
$f_1(x)=x_2$
Stage 2: Test the classifier on E_2
$E_2^{1\checkmark}$ (correct predictions)
$E_2^{1\times}$ (wrong predictions)
(b) Environment $E_2 = E_2^{1\checkmark} \cup E_2^{1\times}$
(c) Unstable correlation varies as we interpolating $P_2^{1\checkmark}$ and $P_2^{1\times}$
$\lambda P_2^{1\checkmark} + (1-\lambda) P_2^{1\times}$
The number in each box represents the conditional probability $P(y=1 \mid x_1, x_2)$
$\lambda=1$
$\lambda=0.8$
$\lambda=0.5$ (P^*)
$\lambda=0.2$
$\lambda=0$
1.00 1.00 0.00 0.00
0.50 0.94 0.06 0.50
0.20 0.80 0.20 0.80
0.06 0.50 0.50 0.94
0.00 0.00 1.00 1.00
Correlation coefficient $\rho(x_2, y)$ v.s. λ
Stage 3: Minimize the worst-case risk across all interpolations of $P_2^{1\checkmark}, P_2^{1\times}, P_1^{2\checkmark}, P_1^{2\times}$.

Figure 1-1: Illustration of our algorithm on the toy example. The label y agrees with the stable feature x_1 with probability 0.8 on both environments. For the unstable feature x_2, the probability of $x_2 = y$ is 1.0 in E_1 and 0.9 in E_2. Stage 1: We train a classifier f_1 on E_1. It learns to make predictions solely based on the unstable feature x_2. Stage 2: We use f_1 to partition E_2 based on the prediction correctness. While the correlation of x_2 is positive for both E_1 and E_2, it flips to negative in set of wrong predictions $E_2^{1\times}$. Stage 3: Interpolating $P_2^{1\checkmark}$ and $P_2^{1\times}$ allows us to uncover an oracle distribution P^* where the unstable feature x_2 is not correlated with the label. Note that here we only illustrate how to partition E_2 using f_1. In our algorithm, we also use the classifier f_2 (trained on E_2) to partition E_1, and the final model f is obtained by minimizing the worst-case risk over all interpolations of $P_1^{2\checkmark}, P_1^{2\times}, P_2^{1\checkmark} P_2^{1\times}$.

- First, we simulate biases in synthetic environments by appending spurious features. Empirical results in both digit classification and sentiment classification show that our method delivers significant performance gain (23.85%) over invariant risk minimization (IRM) [6]. Quantitative analyses confirm that our method identifies groups with opposite spurious correlations.

- Next, we applied our method on natural environments that are defined by a given input attribute. Empirical results on CelebA and ASK2ME confirm that our method is able to improve robustness against other attributes that are unknown during training, outperforming IRM by 12.41%.

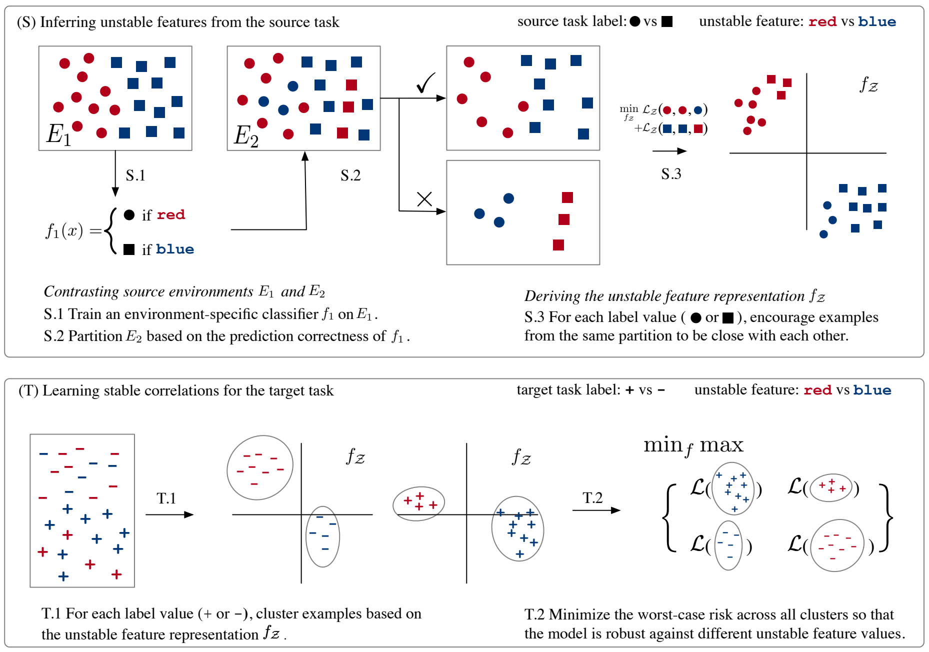
1.2 Learning Stable Classifiers by Transferring Unstable Features

While it is easy to collect annotations from multiple environments for resource-rich tasks, this procedure is infeasible for new tasks or resource-limited tasks. Fortunately, related tasks are often fraught with similar spurious correlations. For instance, when classifying animals such as camels vs. cows, their backgrounds (desert vs. grass) may constitute a spurious correlation [14]. The same bias between the label and the background also persists in other related classification tasks (such as sheep vs. antelope). In the resource-scarce target task, we only have access to the input-label pairs. However, in the source tasks, where training data is sufficient, identifying biases may be easier. For instance, we may have examples collected from multiple environments, in which correlations between bias features and the label are different [6]. These source environments help us define the exact bias features that we want to regulate.

Our goal is to transfer the knowledge that bias features are not reliable in the source task, so that the target classifier will not rely on these spurious correlations. To enable effective transfer, we decompose our problem into two steps:

- **Inferring unstable features from the source task.** Our theorem from the previous chapter suggests that that unstable features are reflected in mistakes observed during classifier transfer across environments. For instance, if the classifier uses the background to distinguish camels from cows, the camel images that are predicted correctly would have a desert background while those predicted incorrectly are likely to have a grass background. More generally, we prove that among examples with the same label value, those with the same prediction outcome will have more similar unstable features than those with different predictions. By forcing examples with the same prediction outcome to stay closer in the feature space, we obtain a representation that encodes these latent unstable features.

- **Learning stable correlations for the target task.** We can apply the unstable feature representation to our target task. By clustering examples based on this representation, we separate minority groups apart from the majority groups. We then minimize the worst-group loss using group distributionally robust optimization [93]. This enforces the target classifier to be robust against


(S) Inferring unstable features from the source task
source task label: ● vs ■
unstable feature: red vs blue
E_1
E_2
S.1
S.2
S.3
✓
×
$f_1(x) = \begin{cases} ● \text{ if red} \\ ■ \text{ if blue} \end{cases}$
$\min_{f_Z} \mathcal{L}_Z(●,●,●) + \mathcal{L}_Z(■,■,■)$
f_Z
(T) Learning stable correlations for the target task
target task label: + vs –
unstable feature: red vs blue
T.1
T.2
f_Z
f_Z
$\min_f \max$
$\mathcal{L}(\;)$

Contrasting source environments E_1 and E_2
S.1 Train an environment-specific classifier f_1 on E_1.
S.2 Partition E_2 based on the prediction correctness of f_1.

Deriving the unstable feature representation f_Z
S.3 For each label value (● or ■), encourage examples from the same partition to be close with each other.

T.1 For each label value (+ or −), cluster examples based on the unstable feature representation f_Z.

T.2 Minimize the worst-case risk across all clusters so that the model is robust against different unstable feature values.

Figure 1-2: Our algorithm TOFU 1) infers unstable features from the source task (Section 3.3.1) and 2) learns stable correlations for the target task (Section 3.3.2). We create partitions for all environment pairs. For ease of illustration, we only depict using f_1 to partition E_2. Best viewed in color.

different values of the unstable features. In the example above, animals would be clustered according to backgrounds, and the classifier should perform well regardless of the clusters (backgrounds).

We applied our method to both text classification tasks and image classification tasks. Comparing with state-of-the-art transfer techniques, our method significantly improves the model robustness (by 18.06%). Qualitative and quantitative analyses confirm the our method is able to identify the unstable features.

1.3 Learning to Split for Automatic Bias Detection

What if we only have access to the set of input-label pairs? No additional data environments. No additional source tasks.

We propose Learning to Split (`ls`), an algorithm that simulates generalization failure directly from the set of input-label pairs. Specifically, `ls` learns to split the dataset so that predictors trained on the training split *cannot generalize* to the testing split. This performance gap provides a proxy for measuring the degree of bias in the learned features and can therefore be used to identify unknown biases. Our algorithm `ls` consists of two components:

- **Splitter**. At each iteration, the Splitter creates a train-test split of the dataset. Given an input-label pair, it decides whether this example should be used for

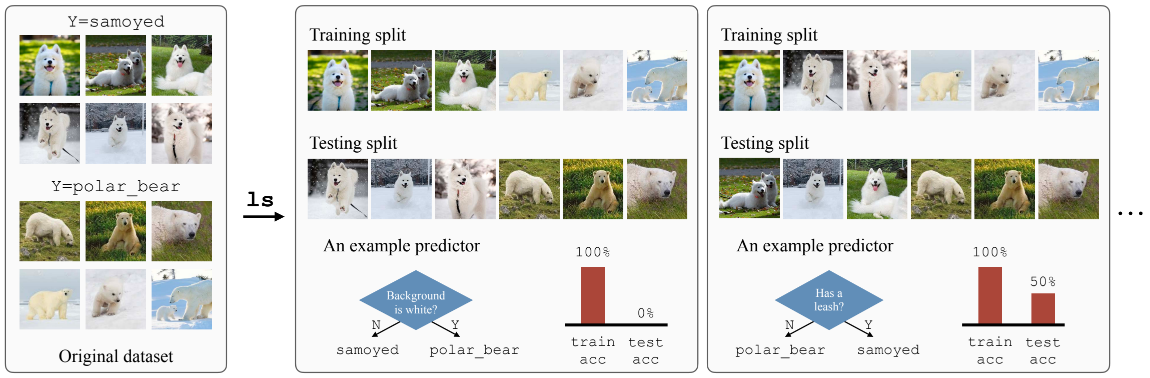


Figure 1-3: Consider the task of classifying samoyeds vs. polar bears. Given the set of image-label pairs, our algorithm `ls` *learns* to *split* the data so that predictors trained on the training split *cannot generalize* to the testing split.

training or testing. To ensure that the splits are meaningful, we impose two regularity constraints on the splits. First, the size of the train split must be comparable to the size of the test split. Second, the marginal distribution of the label should be the same across the splits.

- **Predictor**. The Predictor then takes the training split and learns how to predict the label from the input. Its prediction performance on the testing split is used to inform the Splitter to generate a more challenging split (under the regularity constraints) for the next iteration.

The challenge is that we don't have any explicit annotations for creating non-generalizable splits. One may cast the objectives, maximizing the generalization gap while maintaining the regularity constraints, into a reward function for reinforcement learning. However, our preliminary experiments suggest that the learning signal from this scalar reward is too sparse for the Splitter to learn meaningful splits.

In this chapter, we present a simple yet effective approach to learn the splitter. We show that the prediction correctness of each testing example can be served as a source of weak supervision: generalization performance will drop if we move examples that are predicted correctly away from the testing split, leaving only those that are mispredicted. Therefore, we can directly minimize the cross entropy loss between the Splitter's decision and the Predictor's prediction correctness over the testing split.

Our algorithm is model-agnostic and can be applied to any supervised learning tasks ranging from natural language understanding, image classification to molecular property prediction. Empirical results demonstrate that our algorithm `ls` is able to generate astonishingly challenging splits. For example in MNLI, the generalization performance drops from 79.4% (random split) to 27.8% (`ls`) for a standard BERT-based predictor. Further analyses show that these splits are informative of biases previously identified by human.

In addition, we demonstrate that combining robust learning algorithms (such as DRO) with splits identified by `ls` enables automatic de-biasing. Compared with previous baselines, we substantially improves the worst-group performance (23.4%

on average) when the source of biases is completely unknown during training and validation.

1.4 Contributions

The primary contributions of this thesis are threefold:

- **Predict then Interpolate: A Simple Algorithm to Learn Stable Classifiers**: Given access to multiple data environments, we demonstrate, both theoretically and empirically, how to identify the hidden biases by contrasting these data environments. Combined with group distributionally robust optimization, our algorithm delivers state-of-the-art robustness for text classifications and image classifications.

- **Learning Stable Classifiers by Transferring Unstable Features**: We identified that related tasks often share similar biases. Based on this observation, we propose a novel framework for transferring the knowledge of biases across tasks. We demonstrate more accurate and reliable prediction across 13 transfer settings.

- **Learning to Split for Automatic Bias Detection**: We can reduce biases even without having additional data environments or related tasks. We propose learning to split, an algorithm that simulates generalization failure directly from the set of input-label pairs. Empirical results across multiple modalities confirm that our method is able to identify splits that correlate with biases previously identified by human experts.

Together, these three lines of work represent a step towards practical machine learning. We demonstrate how to leverage different prior knowledge (data environments or related tasks) to address biases. When these prior knowledge is not available, we show that we can still investigate our existing annotations and simulate potential generalization failures. Our proposed methods are model-agnostic and are ready to boost robust learning across different applications.

1.5 Thesis overview

The rest of this thesis is organised as follows:

- In Chapter 2, we consider the scenario where our data is collected across multiple environments. We propose Predict then Interpolate (PI), an algorithm that achieves robustness by contrasting different data environments.
- In Chapter 3, we will discuss how to learn stable classifiers when we don't have access to multiple data environments in the target task. We propose Transfer of Unstable Features (TOFU). Our key idea is to transfer knowledge from a resource-rich source task that manifests similar biases.
- In Chapter 4, we consider the standard supervised learning setting where we are only given a set of input-label pairs. We present Learning to Split (`ls`), an algorithm that *learns* to *split* the dataset so that predictors cannot generalize from the training split to the testing split.
- Chapter 5 concludes this thesis, summarizing the major contributions and providing a few directions for future work.

.

Chapter 2

Predict then Interpolate: A Simple Algorithm to Learn Stable Classifiers

This chapter presents Predict then Interpolate (PI), a simple algorithm for learning correlations that are stable across environments. The algorithm follows from the intuition that when using a classifier trained on one environment to make predictions on examples from another environment, its mistakes are informative as to which correlations are unstable. In this work, we prove that by interpolating the distributions of the correct predictions and the wrong predictions, we can uncover an oracle distribution where the unstable correlation vanishes. Since the oracle interpolation coefficients are not accessible, we use group distributionally robust optimization to minimize the worst-case risk across all such interpolations. We evaluate our method on both text classification and image classification. Empirical results demonstrate that our algorithm is able to learn robust classifiers (outperforms IRM by 23.85% on synthetic environments and 12.41% on natural environments).

2.1 Introduction

Distributionally robust optimization (DRO) alleviates model biases by minimizing the worst-case risk over a set of human-defined groups. However, in order to construct these groups, humans must identify and annotate these biases, a process as expensive as annotating the label itself [17, 37, 93]. In this chapter we propose a simple algorithm to create groups that are informative of these biases, and use these groups to train stable classifiers.

Our algorithm operates on data split among multiple environments, across which correlations between bias features and the label may vary. Instead of handcrafting environments based on explicit, task-dependent biases, these environments can be determined by generic information that is easy to collect [88]. For example, environments can represent data collection circumstances, like location and time. Our goal is to learn correlations that are stable across these environments.

Given these environments, one could directly use them as groups for DRO. Doing so would optimize the worst-case risk over all interpolations of the training environments. However, if the unstable (bias) features are positively *and* differentially correlated with the label in all training environments, the unstable correlation will be positive in any of their interpolations. DRO, optimizing for the best worst-case performance, will inevitably exploit these unstable features, and we fail to learn a stable classifier.

In this chapter, we propose Predict then Interpolate (PI), a simple recipe for creating groups whose interpolation yields a distribution with only stable correlations. Our idea follows from the intuition that when using a classifier trained on one environment to make predictions on examples from a different environment, its mistakes are informative of the unstable correlations. In fact, we can prove that if the unstable features and the label are positively correlated across all environments, the same correlation flips to negative in the set of mistakes. Therefore, by interpolating the distributions of correct and incorrect predictions, we can uncover an "oracle" distribution in which only stable features are correlated with the label. Although the oracle interpolation coefficients are not accessible, we can minimize the worst-case risk over all interpolations, providing an upper bound of the risk on the oracle distribution.

Our learning paradigm consists of three steps. First, we train an individual classifier for each environment to estimate the conditional distribution of the label given the input. These classifiers are biased, as they may rely on any correlations in the dataset. Next, we apply each environment's classifier to partition all other environments, based on prediction correctness. Finally, we obtain our robust classifier by minimizing the worst-case risk over all interpolations of the partitions.

Empirically, we evaluate our approach on both synthetic and real-world environments. First, we simulate unstable correlations in synthetic environments by appending spurious features. Our results in both digit classification and aspect-level sentiment classification demonstrate that our method delivers significant performance gain (23.85% absolute accuracy) over invariant risk minimization (IRM), approaching oracle performance. Quantitative analyses confirm that our method generates partitions with opposite unstable correlations. Next, we applied our approach on natural environments defined by an existing attribute of the input. Our experiments on CelebA and ASK2ME showed that directly applying DRO on environments improves robust accuracy for known attributes, but this robustness doesn't generalize equally across other attributes that are unknown during train time. On the other hand, by creating partitions with opposite unstable correlations, our method is able to improve average worst-group accuracy by 12.41% compared to IRM.

2.2 Related work

Removing known biases: Large scale datasets are fraught with biases. For instance, in face recognition [76], spurious associations may exist between different face attributes (e.g. hair color) and demographic information (e.g. ethnicity) [21]. Furthermore, in natural language inference [18], the entailment label can often be predicted from lexical overlap of the two inputs [83]. Finally, in molecular property prediction [117, 81], performance varies significantly across different scaffolds [120].

Many approaches have been proposed to mitigate biases when they are known beforehand. Examples include adversarial training to remove biases from representations [15, 109], re-weighting training examples [98], and combining a biased model and the base model's predictions using a product of experts [51, 29, 48, 78]. These models are typically designed for a specific type of bias and thus require extra domain knowledge to generalize to new tasks.

Group DRO is another attractive framework since it allows explicit modeling of the distribution family that we want to optimize over. Previous work [54, 86, 94] has shown the effectiveness of group DRO to train un-biased models. In these models, the groups are specified by human based on the knowledge of the bias attributes. Our work differs from them as we create groups using trained models. This allows us to apply group DRO when we don't have annotations for the bias attributes. Moreover, when the bias attributes are available, we can further refine our groups to reduce unknown biases.

Removing unknown biases: Determining dataset biases is time-consuming and often requires task-specific expert knowledge [121, 96]. Thus, there are two lines of work that aim to build robust models without explicitly knowing the type of bias. The first assumes that weak models, which have limited capacity [97] or are under-trained [112], are more prone to rely on shallow heuristics and rediscover previously human-identified dataset biases. By learning from the weak models' mistakes, we can obtain a more robust model. While these methods show empirical benefits on some NLP tasks, the extent to which their assumption holds is unclear. In fact, recent work [95] shows that over-parametrization may actually exacerbate unstable correlations for image classification.

The second line of work assumes that the training data are collected from separate environments, across which unstable features exhibit different correlations with the label [89, 65, 24, 58, 3, 6]. Invariant risk minimization [6], a representative method along this line, learns representations that are simultaneously optimal across all environments. However, since this representation is trained across all environments, it can easily degenerate in real-world applications [45]. One can consider an extreme case where the learned representation directly encodes the one-hot embedding of the label. While this learned representation is stable (invariant) according to the definition, the model can utilize *any unstable features* to generate this representation. We have no guarantee on how the model would generalize when the unstable correlations vanish.

Our algorithm instead decomposes the problem of learning stable classifiers into two parts: finding *unstable features* and training a robust model. By constraining the classifiers to be environment-specific in the first part, we are able to construct an oracle distribution where the unstable features are not correlated with the label. Our model then directly optimizes an upper bound of the risk on this oracle distribution. Empirically, we demonstrate that our method is able to eliminate biases not given during training on multiple real-world applications.

2.3 Method

We consider the setting where the training data are comprised of n environments $\mathcal{E} = \{E_1, \ldots, E_n\}$. For each environment E_i, we have input-label pairs $(x, y) \overset{\text{iid}}{\sim} P_i$. Our goal is to learn correlations that are *stable* across these environments [116] so that the model can generalize to a new test environment E_{test} that has the same stable correlations.

2.3.1 Algorithm

Our intuition follows from a simple question.

What happens if we apply a classifier f_i trained on environment E_i to a different environment E_j?

Suppose we have enough data in E_i and the classifier f_i is able to perfectly fit the underlying conditional $P_i(y|x)$. Since E_i and E_j follow different distributions, the classifier f_i will make mistakes on E_j. These mistakes are natural products of the unstable correlation: if the correlation of the unstable feature is higher in E_i than in E_j, the classifier f_i will overuse this feature when making predictions in E_j.

In fact, we can show that under certain conditions, the unstable correlation within the subset of wrong predictions is opposite of that within the subset of correct predictions (Section 2.3.3). By interpolating between these two subsets, we can uncover an *oracle distribution* where the label is not correlated with the unstable feature. Since this interpolation coefficient is not accessible in practice, we adopt group DRO to minimize the worst-case risk over all interpolations of these subsets. This provides us an upper bound of the risk on the oracle distribution.

Concretely, our approach has three successive stages.

Stage 1: For each environment E_i, train an environment specific classifier f_i.

Stage 2: For each pair of environments E_i and E_j, use the trained classifier f_i to partition E_j into two sets

$$E_j = E_j^{i\checkmark} \cup E_j^{i\times}$$

where $E_j^{i\checkmark}$ contains examples that f_i predicted correctly and $E_j^{i\times}$ contains those predicted incorrectly.

Stage 3: Train the final model f by minimizing the worst-case risk over the set of all interpolations $\mathcal{Q}$:

$$\mathcal{Q} = \left\{ \sum_{i \neq j} \lambda_j^{i\checkmark} P_j^{i\checkmark} + \lambda_j^{i\times} P_j^{i\times} : \sum_{i \neq j} \lambda_j^{i\checkmark} + \lambda_j^{i\times} = 1 \right\},$$

where $P_j^{i\checkmark}$ and $P_j^{i\times}$ are the empirical distributions of $E_j^{i\checkmark}$ and $E_j^{i\times}$. Because the optimum value of a linear program must occur at a vertex, the worst-case risk over $\mathcal{Q}$ is equivalent to the maximum expected risk across all groups. This allows us to formulate the objective as a min-max problem:

$$\min_f \max_{P \in \mathcal{P}} \mathbb{E}_{(x,y)\sim P}[\mathcal{L}(x, y; f)],$$

where $\mathcal{L}(x, y; f)$ is the loss of the model f and $\mathcal{P}$ is the set of distributions $\{P_j^{i\checkmark}\}_{i\neq j} \cup \{P_j^{i\times}\}_{i\neq j}$.

Extensions of the algorithm: For regression tasks, we can set a threshold on the mean square error to partition environments. We can also apply the first two stages multiple times, treating new partitions as different environments, to iteratively refine

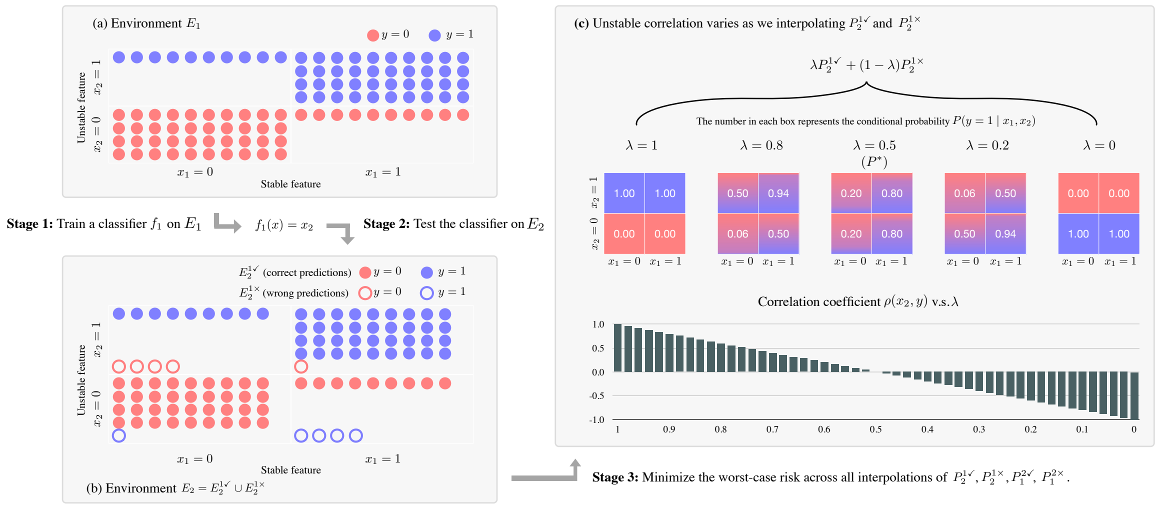

(a) Environment E_1
$y=0$
$y=1$
Unstable feature
$x_2=1$
$x_2=0$
$x_1=0$
$x_1=1$
Stable feature
Stage 1: Train a classifier f_1 on E_1
$f_1(x)=x_2$
Stage 2: Test the classifier on E_2
$E_2^{1\checkmark}$ (correct predictions)
$E_2^{1\times}$ (wrong predictions)
(b) Environment $E_2 = E_2^{1\checkmark} \cup E_2^{1\times}$
(c) Unstable correlation varies as we interpolating $P_2^{1\checkmark}$ and $P_2^{1\times}$
$\lambda P_2^{1\checkmark} + (1-\lambda)P_2^{1\times}$
The number in each box represents the conditional probability $P(y=1 \mid x_1, x_2)$
$\lambda=1$
$\lambda=0.8$
$\lambda=0.5$ (P^*)
$\lambda=0.2$
$\lambda=0$
1.00 1.00 0.00 0.00
0.50 0.94 0.06 0.50
0.20 0.80 0.20 0.80
0.06 0.50 0.50 0.94
0.00 0.00 1.00 1.00
Correlation coefficient $\rho(x_2, y)$ v.s. λ
Stage 3: Minimize the worst-case risk across all interpolations of $P_2^{1\checkmark}, P_2^{1\times}, P_1^{2\checkmark}, P_1^{2\times}$.

Figure 2-1: Illustration of our algorithm on the toy example. The label y agrees with the stable feature x_1 with probability 0.8 on both environments. For the unstable feature x_2, the probability of $x_2 = y$ is 1.0 in E_1 and 0.9 in E_2. Stage 1: We train a classifier f_1 on E_1. It learns to make predictions solely based on the unstable feature x_2. Stage 2: We use f_1 to partition E_2 based on the prediction correctness. While the correlation of x_2 is positive for both E_1 and E_2, it flips to negative in set of wrong predictions $E_2^{1\times}$. Stage 3: Interpolating $P_2^{1\checkmark}$ and $P_2^{1\times}$ allows us to uncover an oracle distribution P^* where the unstable feature x_2 is not correlated with the label. Note that here we only illustrate how to partition E_2 using f_1. In our algorithm, we also use the classifier f_2 (trained on E_2) to partition E_1, and the final model f is obtained by minimizing the worst-case risk over all interpolations of $P_1^{2\checkmark}, P_1^{2\times}, P_2^{1\checkmark} P_2^{1\times}$.

the groups. In this chapter, we focus on the basic setting and leave the rest for future work.

2.3.2 A toy example

To understand the behavior of the algorithm, let's consider a simple example with two environments E_1 and E_2 (Figure 2-1). For each environment, the data are generated by the following process.[1]

- First, sample the feature $x_1 \in \{0, 1\}$ which takes the value 1 with probability 0.5. This is our stable feature.
- Next, sample the observed noisy label $y \in \{0, 1\}$ by flipping the value of x_1 with probability 0.2.

[1][6] used this process to construct the Colored-MNIST dataset.

- Finally, for each environment E_i, sample the unstable feature $x_2 \in \{0,1\}$ by flipping the value of y with probability η_i. Let $\eta_1 = 0$ and $\eta_2 = 0.1$.

Our goal is to learn a classifier that only uses feature x_1 to predict y. Since the unstable feature x_2 is positively correlated with the label across both environments, directly treating the environments as groups and applying group DRO will also exploit this correlation during training.

Let's take a step back and consider a classifier f_1 that is trained only on E_1. Since x_2 is identical to y and x_1 differs from y with probability 0.2, f_1 simply learns to ignore x_1 and predict y as x_2 (Figure 2-1a). When we apply f_1 to the other environment E_2, it will make mistakes on examples where x_2 is flipped from y. Moreover, we can check that the correlation coefficient between the unstable feature x_2 and y is 1 in the set of correct predictions $E_2^{1\checkmark}$ and it flips to -1 in the set of mistakes $E_2^{1\times}$ (Figure 2-1b). In this toy example, the *oracle distribution* P^*, where the correlation between x_2 and y is 0, can be obtained by simply averaging the empirical distribution of the two subsets (Figure 2-1c):

$$P^*(x_1, x_2, y) = 0.5 P_2^{1\checkmark}(x_1, x_2, y) + 0.5 P_2^{1\times}(x_1, x_2, y).$$

We can also verify that the optimal solution that minimize the worst-case risk across $E_2^{1\checkmark}$ and $E_2^{1\times}$ is to predict y only using x_1. (Appendix A.1).

Remark 1: In the algorithm, we also use the classifier f_2 trained on E_2 to partition E_1. The final model f is obtained by minimizing the worst-case risk over $P_1^{2\checkmark}, P_1^{2\times}, P_2^{1\checkmark}, P_2^{1\times}$.

Remark 2: Our algorithm optimizes an *upper bound* of the risk on the oracle distribution. In general, it *doesn't guarantee* that the unstable correlation is not utilized by the model when the worst-case performance is not achieved at the oracle distribution.

2.3.3 Theoretical analysis

In the previous example, we have seen that the unstable correlation flips in the set of mistakes $E_2^{1\times}$ compared to the set of correct predictions $E_2^{1\checkmark}$. Here, we would like to investigate how this property holds in general.[2] We focus our analysis on binary classification tasks where $y \in \{0, 1\}$. Let x_1 be the stable feature and x_2 be unstable feature that has various correlations across environments. We use capital letters X_1, X_2, Y to represent random variables and use lowercase letters x_1, x_2, y to denote their specific values.

Proposition 1. *For a pair of environments E_i and E_j, assuming that the classifier f_i is able to learn the true conditional $P_i(Y \mid X_1, X_2)$, we can write the joint distribution P_j of E_j as the mixture of $P_j^{i\checkmark}$ and $P_j^{i\times}$:*

$$P_j(x_1, x_2, y) = \alpha_j^i P_j^{i\checkmark}(x_1, x_2, y) + (1 - \alpha_j^i) P_j^{i\times}(x_1, x_2, y),$$

where $\alpha_j^i = \sum_{x_1,x_2,y} P_j(x_1, x_2, y) \cdot P_i(y \mid x_1, x_2)$ and

$$\begin{aligned} P_j^{i\checkmark}(x_1, x_2, y) &\propto P_j(x_1, x_2, y) \cdot P_i(y \mid x_1, x_2), \\ P_j^{i\times}(x_1, x_2, y) &\propto P_j(x_1, x_2, y) \cdot P_i(1 - y \mid x_1, x_2). \end{aligned}$$

Intuitively, when partitioning the environment E_j, we are scaling its joint distribution based on the conditional on E_i.

Two degenerate cases: From Proposition 1, we see that the algorithm degenerates when $\alpha_j^i = 0$ (predictions of f_i are all wrong) or $\alpha_j^i = 1$ (predictions of f_i are all correct). The first case occurs when the unstable correlation is flipped between P_i and P_j. One may think about setting $\eta_1 = 0$ and $\eta_2 = 1$ in the toy example. In this case, we can obtain the oracle distribution by directly interpolating P_i and P_j. The second case implies that the conditional is the same across the environments: $P_i(Y \mid X_1, X_2) = P_j(Y \mid X_1, X_2)$. Since x_2 is the unstable feature, this equality holds

[2]All proofs are relegated to Appendix A.2.

when the conditional mutual information between X_2 and Y is zero given X_1, i.e., $P_i(Y \mid X_1, X_2) = P_i(Y \mid X_1)$. In this case, f_i already ignores the unstable feature x_2.

To carryout the following analysis, we assume that the marginal distribution of Y is uniform in all joint distributions, i.e., f_i performs equally well on different labels.

Theorem 1. *Suppose X_2 is independent of X_1 given Y. For any environment pair E_i and E_j, if $\sum_y P_i(x_2 \mid y) = \sum_y P_j(x_2 \mid y)$ for any x_2, then $\mathrm{Cov}(X_2, Y; P_i) > \mathrm{Cov}(X_2, Y; P_j)$ implies $\mathrm{Cov}(X_2, Y; P_j^{i\times}) < 0$ and $\mathrm{Cov}(X_2, Y; P_i^{j\times}) > 0$.*

The result follows from the connection between the covariance and the conditional. On one side, the covariance between x_2 and Y captures the difference of their conditionals: $P(X_2 \mid Y = 1) - P(X_2 \mid Y = 0)$, On the other side, the conditional independence assumption allows us to factorize the joint distribution: $P_i(x_1, x_2, y) = P_i(x_1, y) P_i(x_2 \mid y)$. Combining them together finishes the proof.

Theorem 1 tells us no matter whether the spurious correlation is positive or negative, we can obtain an oracle distribution P^*, $\mathrm{Cov}(X_2, Y; P^*) = 0$ by interpolating across $P_j^{i\checkmark}$, $P_j^{i\times}$, $P_i^{j\checkmark}$, $P_i^{j\times}$. By optimizing the worst-case risk across all interpolations, our final model f provides an *upper bound* of the risk on the oracle distribution P^*.

We also note that the toy example in Section 2.3.2 is a special case of the assumption in Theorem 1. While many previous work also construct datasets with this assumption [6, 28], it may be too restrictive in practice. In the general case, although we cannot guarantee the sign of the correlation, we can still obtain an upper bound for $\mathrm{Cov}(X_2, Y; P_j^{i\times})$ and a lower bound for $\mathrm{Cov}(X_2, Y; P_i^{j\times})$:

Theorem 2. *For any environment pair E_i and E_j, $\mathrm{Cov}(X_2, Y; P_i) > \mathrm{Cov}(X_2, Y; P_j)$ implies*

$$\mathrm{Cov}(X_2, Y; P_j^{i\times}) < \frac{1-\alpha_j^i}{\alpha_i^i} \mathrm{Cov}(X_2, Y; P_i^{i\checkmark}) - \frac{1-\alpha_j^i}{\alpha_j^i} \mathrm{Cov}(X_2, Y; P_j^{i\checkmark})$$

$$\mathrm{Cov}(X_2, Y; P_i^{j\times}) > \frac{1-\alpha_i^j}{\alpha_j^j} \mathrm{Cov}(X_2, Y; P_j^{j\checkmark}) - \frac{1-\alpha_i^j}{\alpha_i^j} \mathrm{Cov}(X_2, Y; P_i^{j\checkmark})$$

where $P_i^{i\checkmark}$ is the distribution of the correct predictions when applying f_i on E_i.

Intuitively, if the correlation is stronger in E_i, then the classifier f_i will overuse this correlation and make mistakes on E_j when this stronger correlation doesn't hold. Conversely, the classifier f_j will underuse this correlation and make mistakes on E_i when the correlation is stronger.

2.4 Experimental setup

2.4.1 Datasets and settings

Synthetic environments: To assess the empirical behavior of our algorithm, we start with controlled experiments where we can simulate spurious correlation. We consider two standard datasets: MNIST [68] and BeerReview [82].[3]

For MNIST, we adopt [6]'s approach for generating spurious correlation and extend it to a more challenging multi-class problem. For each image, we sample y, which takes on the same value as its numeric digit with 0.75 probability and a uniformly random other digit with the remaining probability. The spurious feature in sampled in a similar way: it takes on the same value as y with η probability and a uniformly random other value with the remaining probability. We color the image according to the value of the spurious feature. We set η to 0.9 and 0.8 respectively for the training environments E_1 and E_2. In the testing environment, η is set to 0.1.

For BeerReview, we consider three aspect-level sentiment classification tasks: look, aroma and palate [69, 11]. For each review, we append an artificial token (`art_pos` or `art_neg`) that is spuriously correlated with the binary sentiment label (`pos` or `neg`). The artificial token agrees with the sentiment label with probability 0.9 in environment E_1 and with probability 0.8 in environment E_2. In the testing environment, the probability reduces to 0.1. Unlike MNIST, here we do not inject artificial label noise to the datasets.

Validation set plays a crucial role when the training distribution is different from the testing distribution [45]. For both datasets, we consider two different validation

[3]All dataset statistics are relegated to Appendix A.3.1.

settings and report their performance separately: 1) sampling the validation set from the training environment; 2) sampling the validation set from the testing environment.

Natural environments: We also consider a practical setting where environments are naturally defined by some attributes of the input and we want to use them to reduce biases that are *unknown* during training and validation. We study two datasets: CelebA [76] where the attributes are annotated by human and ASK2ME [12] where the attributes are automatically generated by rules.

CelebA is an image classification dataset where each input image (face) is paired with 40 binary attributes. We adopt [93]'s setting and treat hair color ($y \in \{\texttt{blond}, \texttt{dark}\}$) as the target task. We use the gender attribute to define the two training environments, $E_1 = \{\texttt{female}\}$ and $E_2 = \{\texttt{male}\}$. Our goal is to learn a classifier that is robust to other unknown attributes such as `wearing_hat`. For model selection, we partition the validation data into four groups based on the gender value and the label value: $\{\texttt{female}, \texttt{blond}\}$, $\{\texttt{female}, \texttt{dark}\}$, $\{\texttt{male}, \texttt{blond}\}$, $\{\texttt{male}, \texttt{dark}\}$. We use the worst-group accuracy as our validation criteria.

ASK2ME is a text classification dataset where an input text (paper abstract from PubMed) is paired with 17 binary attributes, each indicating the presence of a different disease. The task is to predict whether the input is informative about the *risk* of cancer for gene mutation carriers, rather than cancer itself [33]. We define two training environments based on the `breast_cancer` attribute, $E_1 = \{\texttt{breast\_cancer=0}\}$ and $E_2 = \{\texttt{breast\_cancer=1}\}$. We would like to see whether the classifier is able to remove spurious correlations from other diseases that are unknown during training. Similar to CelebA, we compute the worst-group accuracy based on the `breast_cancer` value and the label value and use it for validation.

At test time, we evaluate the classifier's prediction robustness on all attributes over a held-out test set. For each attribute, we report the worst-group accuracy and the average-group accuracy.

2.4.2 Baselines

We compare our algorithm against the following baselines:

ERM: We combine all environments together and apply standard empirical risk minimization.

IRM: Invariant risk minimization [6] learns a representation such that the linear classifier on top of this representation is simultaneously optimal across different environments.

RGM: Regret minimization [58] simulates unseen environments by using part of the training set as held-out environments. It quantifies the generalization ability in terms of regret, the difference between the losses of two auxiliary predictors trained with and without examples in the current environment.

DRO: We can also apply DRO on groups defined by the environments and the labels. For example, in beer review, we can partition the training data into the four groups: {`pos`, E_1}, {`neg`, E_1} {`pos`, E_2}, {`neg`, E_2}. Minimizing the worst-case performance over these human-defined groups has shown success in improving model robustness [93].

Oracle: In the synthetic environments, we can use the spurious features to define groups and train an oracle DRO model. For example, in beer review, the oracle model will minimize the worst-case risk over the four groups: {`pos`, `art_pos`}, {`pos`, `art_neg`} {`pos`, `art_pos`}, {`pos`, `art_neg`}. This helps us analyze the contribution of our algorithm in isolation of the inherent limitations of the task.

For fair comparison, all methods share the same model architecture.[4] Implementation details can be found in Appendix A.3.2.

[4]For IRM and RGM, in order to tune the weights and annealing strategy for the regularizer, the hyper-parameter search space is $21\times$ larger than other methods.

	ERM		DRO		IRM		RGM		PI (OURS)		ORACLE	
$P_{\text{val}} = P_{\text{test}}$?	✓	×	✓	×	✓	×	✓	×	✓	×	✓	×
MNIST	26.1	14.2	32.5	21.0	45.4	13.1	42.4	15.3	**69.4**	**69.6**	71.4	71.6
Beer Look	64.6	60.9	64.5	62.7	65.8	63.3	66.3	61.5	**78.0**	**70.6**	80.3	73.5
Beer Aroma	55.2	51.9	57.0	53.3	60.2	53.2	66.3	57.9	**77.0**	**67.3**	77.3	69.9
Beer Palate	49.0	46.6	47.7	46.3	66.4	44.0	68.7	44.8	**74.1**	**61.5**	74.8	66.3

Table 2.1: Accuracy of different methods on image classification (majority baseline 10%) and aspect-level sentiment classification (majority baseline 50%). All methods are tuned based on a held-out validation set. We consider two validation settings: 1) sample the validation set from the testing environment ($P_{\text{val}} = P_{\text{test}}$); 2) sample the validation set from the training environment.

	E_1	E_2	$E_2^{1\checkmark}$	$E_2^{1\times}$
MNIST	0.8955	0.7769	0.9961	−0.1040
Beer Look	0.8007	0.6006	0.8254	−0.8030
Beer Aroma	0.8007	0.6006	0.9165	−0.9303
Beer Palate	0.8007	0.6006	0.9394	−0.9189

Table 2.2: Pearson correlation coefficient between the spurious feature and the label across four datasets. While the correlation is positive for both training environments, it flips to negative in the set of wrong predictions $E_2^{1\times}$. Interpolating across $E_2^{1\checkmark}$ and $E_2^{1\times}$ allows us to remove the unstable correlation.

2.5 Results

2.5.1 Synthetic environments

Table 2.1 summarizes the results on synthetic environments. As we expected, since the signs of the unstable correlation are the same across the training environments, both ERM and DRO exploit this information and fail to generalize when it changes in the testing environment. While IRM and RGM are able to learn stable correlations when we use the testing environment for model selection, their performance quickly drop to that of ERM when the validation data is drawn from the training environment, which also backs up the claim from [45].

Our algorithm obtains substantial gains across four tasks It performs much more stable under different validation settings. Specifically, comparing against the best

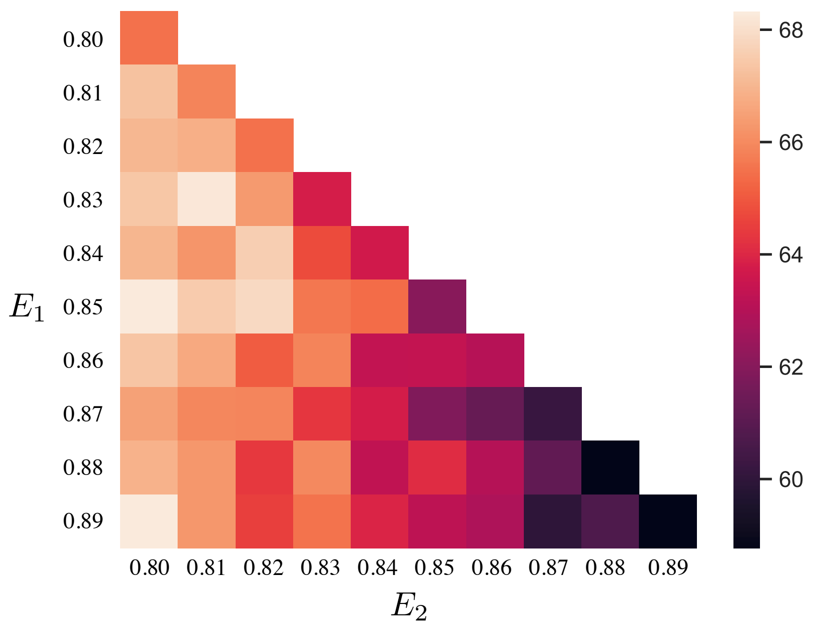


Figure 2-2: The ability of generalization changes as we vary the gap between the training environments. The x and y axes denote the probabilities that the injected artificial token agrees with the label. Heatmap corresponds to the testing accuracy for Beer Aroma.

baseline, our algorithm improves the accuracy by 20.06% when the validation set is drawn from the training environment and 12.97% when it is drawn from the testing environment. Its performance closely matches the oracle model with only 2% difference on average.

Why does partitioning the training environments help? To demystify the huge performance gain, we quantitatively analyze the partitions created by our algorithm in Table 2.2.[5] We see that while the unstable correlation is positive in both training environments, it flips to negative in the set of wrong predictions, confirming our theoretical analysis. In order to perform well across all partitions, our final classifier learns not to rely on the unstable features.

Do we need different training environments? We study the relation between the diversity of the training environments and the performance of the classifier on the beer review dataset. Specifically, we consider 45 different training environment pairs

[5]The partitions only depend on the training environments. It is independent to the choice of the validation data.

	ERM		DRO		IRM		RGM		PI (OURS)	
	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg
Kidney cancer	16.6	66.6	**50.0**	68.7	33.3	73.0	33.3	71.3	**50.0**	**74.7**
Adenocarcinoma	33.3	72.9	77.2	79.2	55.5	78.4	55.5	78.1	**80.2**	**84.7**
Lung cancer	44.4	74.7	62.5	74.5	38.8	74.2	50.0	74.7	**70.3**	**78.8**
Polyp syndrom	44.4	74.6	**77.2**	78.7	55.5	76.3	66.6	78.7	69.2	**81.2**
Hepatobiliary cancer	44.4	73.0	**60.0**	73.8	55.5	77.1	55.5	76.2	**60.0**	**78.9**
Breast cancer	66.4	80.4	75.0	78.8	66.8	80.5	64.3	79.8	**80.3**	**83.1**
Average*	54.8	77.7	67.3	77.5	57.8	79.1	60.8	79.0	**74.1**	**83.1**

Table 2.3: Worst-group and average-group accuracy on ASK2ME text classification. We show the results for the worst 5 attributes (sorted based on ERM) and the given attribute `breast_cancer`. Average is computed based on the performance across all attributes. See Appendix A.4 for full results.

	ERM		DRO		IRM		RGM		PI (OURS)	
	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg
Goatee	0.0	68.2	0.0	70.0	84.8	90.8	91.5	**95.6**	**91.6**	94.5
Wearing_Hat	7.6	70.3	46.1	82.2	46.1	77.3	**61.5**	**86.2**	53.8	84.5
Chubby	9.5	70.6	61.9	84.6	**76.1**	82.9	47.6	82.3	71.4	**86.5**
Wearing_Necktie	25.0	74.5	90.0	91.5	80.0	84.3	35.0	79.3	**91.4**	**92.5**
Sideburns	38.4	77.8	84.6	90.6	76.9	84.1	76.9	89.7	**91.3**	**93.7**
Gender	46.6	80.1	85.5	90.8	74.4	83.9	70.0	87.7	**90.5**	**91.5**
Average*	60.0	83.6	84.2	90.8	78.5	85.7	82.5	90.9	**87.0**	**91.4**

Table 2.4: Worst-group and average-group accuracy for hair color prediction on CelebA. We show the results for the worst 5 attributes (sorted based on ERM) and the given attribute `gender`. Average is computed based on the performance across all attributes. See Appendix A.4 for full results.

where we vary the probability that the artificial token agrees with the label from 0.80 to 0.89. We observe that the classifier performs better as we reduce the amount of spurious correlations (moving up along the diagonal in Figure 2-2). The classifier also generalizes better when we increase the gap between the two training environments (moving from right to left in Figure 2-2). In fact, when the training environments share the same distribution, the notion of stable correlation and unstable correlation is undefined. There is no signal for the algorithm to distinguish between spurious

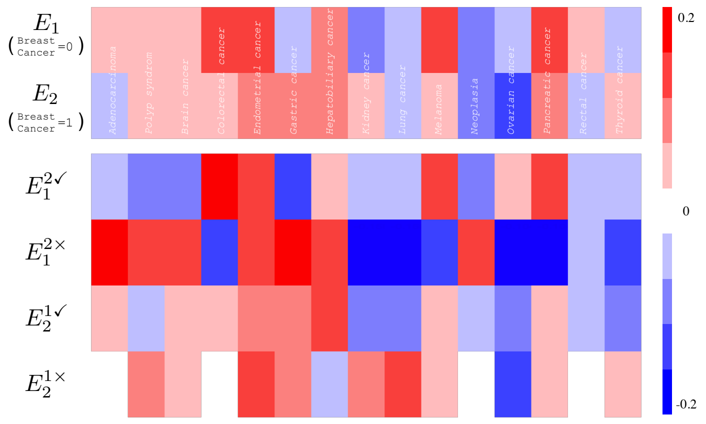


Figure 2-3: Visualization of the Pearson correlation coefficient between the label and the attribute on ASK2ME. Each column corresponds to a different attribute. We observe that correlations vary for inputs with different `breast_cancer` value. Our algorithm utilizes this difference to create partitions with opposite correlations (red vs. blue) so that we can uncover an oracle distribution (different for each attribute) by interpolating these partitions.

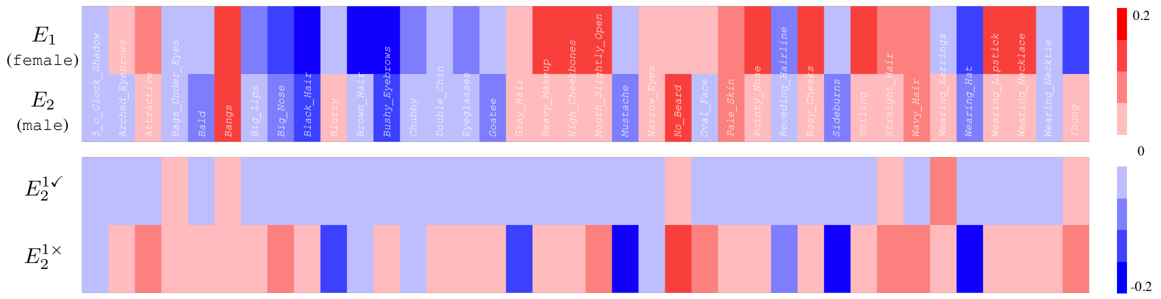


Figure 2-4: Visualization of the Pearson correlation coefficient between the label (hair color) and other attributes on CelebA. Each column corresponds to a different attribute. Due to the huge difference between the label marginals, $P_1(\texttt{blond}) = 0.24$ vs. $P_2(\texttt{blond}) = 0.02$, classifier f_2 predicts every example in environment E_1 as `dark`. The resulting partition, $E_1^{2\times} = \{\texttt{female, blond}\}$ and $E_1^{2\checkmark} = \{\texttt{female, dark}\}$, coincides with the human-defined groups in DRO. On the other hand, classifier f_1 is able to partition environment E_2 with opposite correlations (red vs. blue).

features and features that generalize.

2.5.2 Natural environments

Table 2.3 and 2.4 summarize the results on using natural environments to reduce biases from attributes that are unknown during both training and validation. We observe that directly applying DRO over human-defined groups already surpasses

IRM and RGM on the worst-case accuracy averaged across all attributes. In addition, for the given attribute (`breast_cancer` and `gender`), DRO achieves nearly 10% more improvements over other baselines. However, this robustness doesn't generalize equally towards other attributes. By using the environment-specific classifier to create groups with contrasting unstable correlations, our algorithm delivers marked performance improvement over DRO, 6.81% on ASK2ME and 2.79% on CelebA.

How do we reduce bias from unknown attributes? Figure 2-3 and 2-4 visualize the correlation between each attribute and the label on ASK2ME and CelebA. We observe that although the signs of the correlation can be the same across the training environments, their magnitude may vary. Our algorithm makes use of this difference to create partitions that have opposite correlations for 13 (out of 15) attributes on ASK2ME and 22 (out of 38) attributes on CelebA. These opposite correlations help the classifier to avoid using unstable features during training.

2.6 Conclusion

In this chapter, we propose a simple algorithm to learn correlations that are stable across environments. Specifically, we propose to use a classifier that is trained on one environment to partition another environment. By interpolating the distributions of its correct predictions and wrong predictions, we can uncover an oracle distribution where the unstable correlation vanishes. Experimental results on synthetic environments and natural environments validate that our algorithm is able to generate partitions with opposite unstable correlations and reduce bias that are unknown during training.

Chapter 3

Learning Stable Classifiers by Transferring Unstable Features

In the previous chapter, we demonstrate how to learn stable classifiers when multiple data environments are available. While it is easy to collect annotations from different environment for resource-rich tasks, this procedure is infeasible for new tasks or resource-limited tasks. Fortunately, in real-world applications related tasks often share similar biases – an observation we may leverage to develop stable classifiers in the transfer setting. In this chapter, we extend the idea from the previous chapter and demonstrate that by contrasting different data environments in the source task, we can derive a representation that encodes the unstable features. To obtain a robust model for the target task, we first cluster target examples according to this representation and then minimize the worst-case risk across the resulting clusters. We evaluate our method on both text and image classifications. Empirical results demonstrate that our algorithm is able to maintain robustness on the target task for both synthetically generated environments and real-world environments.

3.1 Introduction

Related tasks are often fraught with similar spurious correlations. For instance, when classifying animals such as camels vs. cows, their backgrounds (desert vs. grass) may

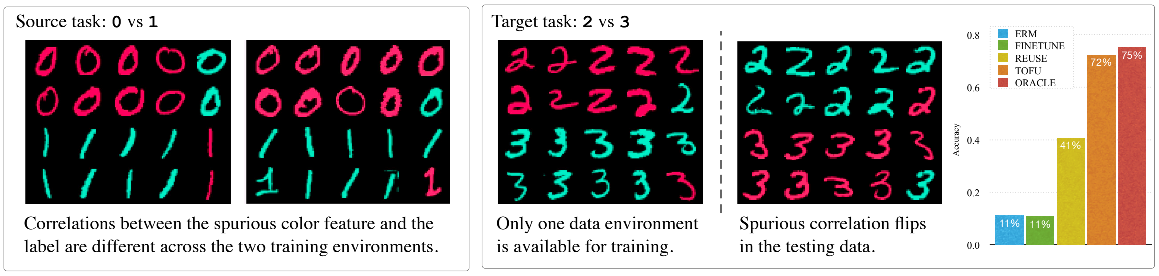


Figure 3-1: Transferring across tasks in Colored MNIST [6]. On the source task, we learn a color-invariant model that achieves oracle performance (given direct access to the unstable features). However, directly transferring this model to the target task, by reusing or fine-tuning its feature extractor, severely overfits the spurious correlation and underperforms the majority baseline (50%) on a test set where the spurious correlation flips. By explicitly transferring the unstable features, our algorithm `tofu` (Transfer OF Unstable features) is able to reach the oracle performance.

constitute a spurious correlation [14]. The same bias between the label and the background also persists in other related classification tasks (such as sheep vs. antelope). In the resource-scarce target task, we only have access to the input-label pairs. However, in the source tasks, where training data is sufficient, identifying biases may be easier. For instance, we may have examples collected from multiple environments, in which correlations between bias features and the label are different [6]. These source environments help us define the exact bias features that we want to regulate.

One obvious approach to utilize the source task is direct transfer. Specifically, given multiple source environments, we can train an unbiased source classifier and then apply its representation to the target task. However, we empirically demonstrate that while the source classifier is not biased when making its final predictions, its internal continuous representation can still encode information about the unstable features. Figure 3-1 shows that in Colored MNIST, where the digit label is spuriously correlated with the image color, direct transfer by either re-using or fine-tuning the representation learned on the source task fails in the target task, performing no better than the majority baseline.

In this chapter, we propose to explicitly inform the target classifier about unstable features from the source data. Specifically, we derive a representation that encodes these unstable features using the source environments. Then we identify distinct subpopulations by clustering examples based on this representation and apply group DRO [93] to minimize the worst-case risk over these subpopulations. As a result, we enforce the target classifier to be robust against different values of the unstable features. In the example above, animals would be clustered according to backgrounds, and the classifier should perform well regardless of the clusters (backgrounds).

The remaining question is how to compute the unstable feature representation using the source data environments. Following the same idea in the previous chapter, we hypothesize that unstable features are reflected in mistakes observed during classifier transfer across environments. For instance, if the classifier uses the background to distinguish camels from cows, the camel images that are predicted correctly would have a desert background while those predicted incorrectly are likely to have a grass background. More generally, we prove that among examples with the same label value, those with the same prediction outcome will have more similar unstable features than those with different predictions. By forcing examples with the same prediction outcome to stay closer in the feature space, we obtain a representation that encodes these latent unstable features.

We evaluate our approach, Transfer OF Unstable features (TOFU), on both synthetic and real-world environments. Synthetic experiments first confirm our hypothesis that standard transfer approaches fail to learn a stable classifier for the target task. By explicitly transferring the unstable features, our method significantly improves over the best baseline across 12 transfer settings (22.9% in accuracy), and reaches the performance of an oracle model that has direct access to the unstable features (0.3% gap). Next, we consider a practical setting where environments are defined by an input attribute and our goal is to reduce biases from other unknown attributes. On CelebA, TOFU achieves the best worst-group accuracy across 38 latent attributes, outperforming the best baseline by 18.06%. Qualitative and quantitative analyses confirm that TOFU is able to identify the unstable features.

3.2 Related work

Removing bias via annotations: Due to idiosyncrasies of the data collection process, annotations are often coupled with unwanted biases [21, 98, 83, 120]. To address this issue and learn robust models, researchers leverage extra information [15, 109, 51, 29, 48, 78]. One line of work assumes that the bias attributes are known and have been annotated for each example, e.g., group distributionally robust optimization (DRO) [54, 86, 94]. By defining groups based on these bias attributes, we explicitly specify the distribution family to optimize over. However, identifying the hidden biases is time-consuming and often requires domain knowledge [121, 96]. To address this issue, another line of work [89, 65, 24, 58, 3, 6, 9, 66, 101] only assumes access to a set of data environments. These environments are defined based on readily-available information of the data collection circumstances, such as location and time. The main assumption is that while spurious correlations vary across different environments, the association between the causal features and the label should stay the same. Thus, by learning a representation that is invariant across all environments, they alleviate the dependency on spurious features. In contrast to previous works, we don't have access to any additional information besides the labels in our target task. We show that we can achieve robustness by transferring the unstable features from a related source task.

Transferring robustness across tasks: Prior work has also studied the transferability of adversarial robustness across tasks. For example, [49, 100] show that by pre-training the model on a large-scale source task, we can improve the model robustness against adversarial perturbations over l_∞ norm. We note that these perturbations measure the smoothness of the classifier, rather than the stability of the classifier against spurious correlations. In fact, our results show that if we directly re-use or fine-tune the pre-trained feature extractor on the target task, the model will quickly over-fit to the unstable correlations present in the data. We propose to address this issue by explicitly inferring the unstable features using the source environments and use this information to guide the target classifier during training.

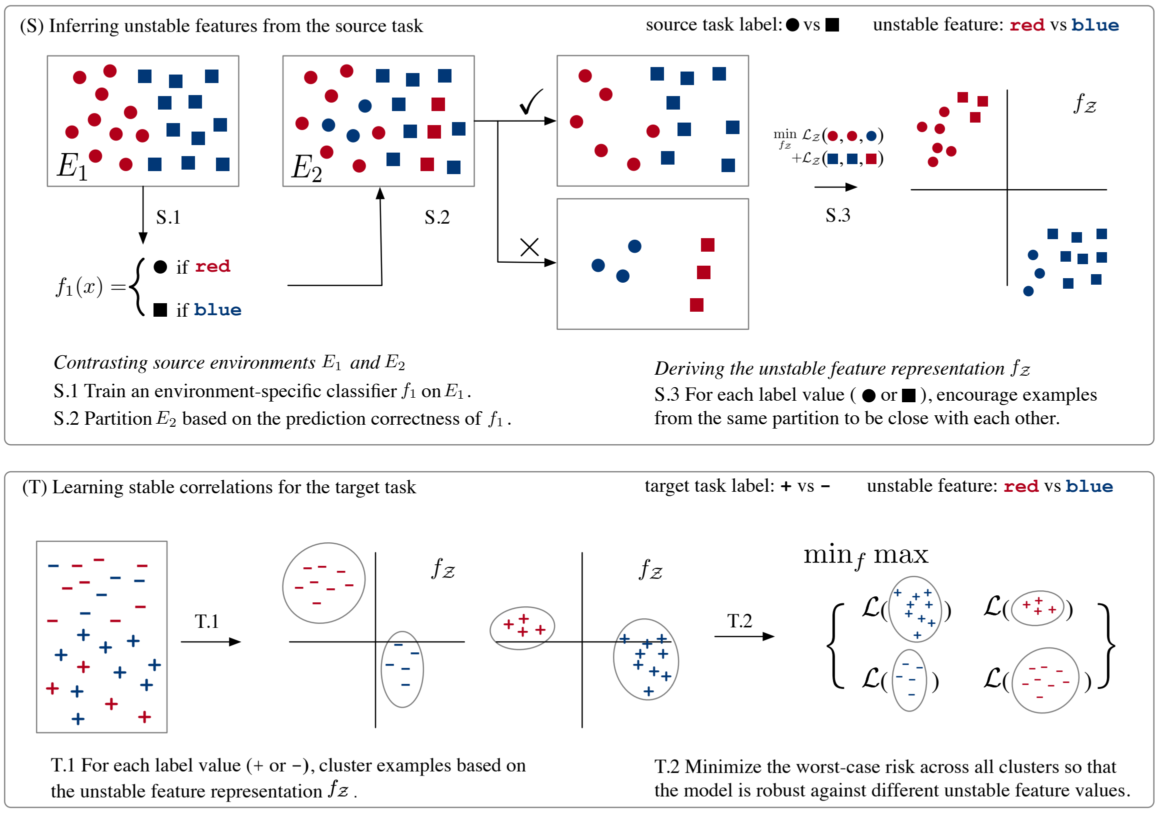
3.3 Method

Problem formulation We consider the transfer problem from a source task to a target task. For the source task, we assume the standard setting [6] where the training data contain n environments $E_1, \ldots, E_n$. Within each environment E_i, examples are drawn from the joint distribution $P_i(x, y)$. Following [116], we define unstable features $\mathcal{Z}(x)$ as features that are *differentially* correlated with the label across the environments. We note that $\mathcal{Z}(x)$ is unknown to the model.

For the target task, we only have access to the input-label pairs (x, y) (i.e. no environments). We assume that the target label is *not* causally associated with the above unstable features $\mathcal{Z}$. However, due to collection biases, the target data may contain *spurious correlations* between the label and $\mathcal{Z}$. Our goal is to transfer the knowledge that $\mathcal{Z}$ is unstable in the source task, so that the target classifier will not rely on these spurious features.

Overview If the unstable features have been identified for the target task, we can simply apply group DRO to learn a stable classifier. By grouping examples based on the unstable features and minimizing the worst-case risk over these *manually-defined* groups, we explicitly address the bias from these unstable features [54, 86, 94]. In our setup, while these unstable features are not accessible, we can leverage the source environments to derive groups over the target data that are informative of these biases. Applying group DRO on these *automatically-derived* groups, we can eliminate the unstable correlations in the target task.

Our overall transfer paradigm is depicted in Figure 3-2. It consists of two steps: inferring unstable features from the source task (Section 3.3.1) and learning stable correlations for the target task (Section 3.3.2). First, for the source task we use a classifier trained on one environment to partition data from another environment based on the correctness of its predictions. Starting from the theoretical results in [9], we show that these partitions reflect the similarity of the examples in terms of their unstable features: among examples with the same label value, those that share the same prediction outcome have more similar unstable features than those with



(S) Inferring unstable features from the source task
source task label: ● vs ■
unstable feature: red vs blue
E_1
E_2
S.1
S.2
S.3
$f_1(x) = \begin{cases} \bullet & \text{if red} \\ \blacksquare & \text{if blue} \end{cases}$
✓
×
$\min_{f_\mathcal{Z}} \mathcal{L}_\mathcal{Z}(\bullet,\bullet,\bullet) + \mathcal{L}_\mathcal{Z}(\blacksquare,\blacksquare,\blacksquare)$
$f_\mathcal{Z}$
Contrasting source environments E_1 and E_2
S.1 Train an environment-specific classifier f_1 on E_1.
S.2 Partition E_2 based on the prediction correctness of f_1.
Deriving the unstable feature representation $f_\mathcal{Z}$
S.3 For each label value (● or ■), encourage examples from the same partition to be close with each other.
(T) Learning stable correlations for the target task
target task label: + vs −
unstable feature: red vs blue
T.1
T.2
$f_\mathcal{Z}$
$\min_f \max$
$\mathcal{L}$
T.1 For each label value (+ or −), cluster examples based on the unstable feature representation $f_\mathcal{Z}$.
T.2 Minimize the worst-case risk across all clusters so that the model is robust against different unstable feature values.

Figure 3-2: Our algorithm TOFU 1) infers unstable features from the source task (Section 3.3.1) and 2) learns stable correlations for the target task (Section 3.3.2). We create partitions for all environment pairs. For ease of illustration, we only depict using f_1 to partition E_2. Best viewed in color.

different predictions (Theorem 3). We can then derive a representation $f_\mathcal{Z}$ where examples are distributed based on the unstable features $\mathcal{Z}$. Next, we cluster target examples into groups based on the learned unstable feature representation $f_\mathcal{Z}$. These *automatically-derived* groups correspond to different modes of the unstable features, and they act as proxies to the *manually-defined* groups in the oracle setting where unstable features are explicitly annotated. Finally, we use group DRO to obtain our robust target classifier by minimizing the worst-case risk over these groups.

3.3.1 Inferring unstable features from the source task

Given the data environments from the source task, we would like to 1) identify the unstable correlations across these environments; 2) learn a representation $f_\mathcal{Z}(x)$ that encodes the unstable features $\mathcal{Z}(x)$. We achieve the first goal by contrasting the empirical distribution of different environments (Figure 3-2 S.1 and Figure 3-2 S.2) and the second goal by metric learning (Figure 3-2 S.3).

Let E_i and E_j be two different data environments. [9] shows that by training a classifier f_i on E_i and using it to make predictions on E_j, we can reveal the unstable correlations from its prediction results. Intuitively, if the unstable correlations are stronger in E_i, the classifier f_i will overuse these correlations and make mistakes on E_j when these stronger correlations do not hold.

In this work, we connect the prediction results directly to the unstable features. We show that the prediction results of the classifier f_i on E_j estimate the relative distance of the unstable features.

Theorem 3 (Simplified)**.** *Consider examples in E_j with label value y. Let $X_1^{\checkmark}, X_2^{\checkmark}$ denote two batches of examples that f_i predicted correctly, and let $X_3^{\times}$ denote a batch of incorrect predictions. We use $\overline{\cdot}$ to represent the mean across a given batch. Following the same assumption in [9], we have*

$$\|\overline{\mathcal{Z}}(X_1^{\checkmark}) - \overline{\mathcal{Z}}(X_2^{\checkmark})\|_2 < \|\overline{\mathcal{Z}}(X_1^{\checkmark}) - \overline{\mathcal{Z}}(X_3^{\times})\|_2$$

almost surely for large enough batch size.[1]

The result makes intuitive sense as we would expect example pairs that share the same prediction outcome should be more similar than those with different prediction outcomes. We note that it is critical to look at examples with the same label value; otherwise, the unstable features will be coupled with the task-specific label in the prediction results.

While the value of the unstable features $\mathcal{Z}(x)$ is still not directly accessible, Theorem 3 enables us to learn a feature representation $f_{\mathcal{Z}}(x)$ that preserves the distance between the examples in terms of their unstable features. We adopt standard metric

[1]See Appendix B.1 for the full theorem and proof.

learning [25] to minimize the following triplet loss:

$$\begin{aligned}\mathcal{L}_{\mathcal{Z}}(X_1^{\checkmark}, X_2^{\checkmark}, X_3^{\times}) = \max(0, \delta + \|\overline{f_{\mathcal{Z}}}(X_1^{\checkmark}) - \overline{f_{\mathcal{Z}}}(X_2^{\checkmark})\|_2^2 \\ - \|\overline{f_{\mathcal{Z}}}(X_1^{\checkmark}) - \overline{f_{\mathcal{Z}}}(X_3^{\times})\|_2^2),\end{aligned} \tag{3.1}$$

where δ is a hyper-parameter. By minimizing Eq (3.1), we encourage examples that have similar unstable features to be close in the representation $f_{\mathcal{Z}}$. To summarize, inferring unstable features from the source task consists of three steps (Figure 3-2 S):

S.1 For each source environment E_i, train an environment-specific classifier f_i.

S.2 For each pair of environments E_i and E_j, use classifier f_i to partition E_j into two sets: $E_j^{i\checkmark}$ and $E_j^{i\times}$, where $E_j^{i\checkmark}$ contains examples that f_i predicted correctly and $E_j^{i\times}$ contains those predicted incorrectly.

S.3 Learn an unstable feature representation $f_{\mathcal{Z}}$ by minimizing Eq (3.1) across all pairs of environments E_i, E_j and all possible label value y:

$$f_{\mathcal{Z}} = \arg\min \sum_{y, E_i \neq E_j} \mathbb{E}_{X_1^{\checkmark}, X_2^{\checkmark}, X_3^{\times}} \left[\mathcal{L}_{\mathcal{Z}}(X_1^{\checkmark}, X_2^{\checkmark}, X_3^{\times})\right],$$

where batches $X_1^{\checkmark}, X_2^{\checkmark}$ are sampled uniformly from $E_j^{i\checkmark}|_y$ and batch $X_3^{\times}$ is sampled uniformly from $E_j^{i\times}|_y$ ($\cdot|_y$ denotes the subset of $\cdot$ with label value y).

3.3.2 Learning stable correlations for the target task

Given the unstable feature representation $f_{\mathcal{Z}}$, our goal is to learn a target classifier that focuses on the stable correlations rather than using unstable features. Inspired by group DRO [94] we minimize the worst-case risk across groups of examples that are representative of different unstable feature values. However, in contrast to DRO, these groups are constructed automatically based on the previously learned representation $f_{\mathcal{Z}}$.

For each target label value y, we use the representation $f_{\mathcal{Z}}$ to cluster target examples with label y into different clusters (Figure 3-2 T.1). Since these clusters capture different modes of the unstable features, they are approximations of the typical manually-defined groups when annotations of the unstable features are available. By minimizing the worst-case risk across all clusters, we explicitly enforce the classifier to be robust against unstable correlations (Figure 3-2 T.2). We note that it is important to cluster within examples of the same label, as opposed to clustering the whole dataset. Otherwise, the cluster assignment may be correlated with the target label.

Concretely, learning stable correlations for the target task has two steps (Figure 3-2 T).

T.1 For each label value y, apply K-means (l_2 distance) to cluster examples with label y in the feature space $f_{\mathcal{Z}}$. We use $C_1^y, \ldots, C_{n_c}^y$ to denote the resulting cluster assignment, where n_c is a hyper-parameter.

T.2 Train the target classifier f by minimizing the worst-case risk over all clusters:

$$f = \arg\min \max_{i,y} \mathcal{L}(C_i^y),$$

where $\mathcal{L}(C_i^y)$ is the empirical risk on cluster C_i^y.

3.4 Experimental setup

3.4.1 Datasets and settings

Synthetic environments We start with controlled experiments where environments are created based on the spurious correlation. We consider four datasets: MNIST [68], BeerReview [82], ASK2ME [12] and Waterbird [93]. In MNIST and BeerReview, we inject spurious feature to the input (background color for MNIST and pseudo token for BeerReview). In ASK2ME and Waterbird, spurious feature corresponds to an attribute of the input (`breast_cancer` for ASK2ME and `background` for Waterbird).

For each dataset, we consider multiple tasks and study the transfer between

Table 3.1: Pearson correlation coefficient between the spurious feature $\mathcal{Z}$ and the label Y for each task. The validation environment E^{val} follows the same distribution as E_1^{train}. We study the transfer problem between different task pairs. For the source task S, the model can access $E_1^{\text{train}}(S), E_2^{\text{train}}(S)$ and $E^{\text{val}}(S)$. For the target task T, the model can access $E_1^{\text{train}}(T)$ and $E^{\text{val}}(T)$.

$\rho(\mathcal{Z}, Y)$	Task	E_1^{train}	E_2^{train}	E^{val}	E^{test}
MNIST	ODD	0.87	0.75	0.87	-0.11
	EVEN	0.87	0.75	0.87	-0.11
BEER REVIEW	LOOK	0.60	0.80	0.60	-0.80
	AROMA	0.60	0.80	0.60	-0.80
	PALATE	0.60	0.80	0.60	-0.80
ASK2ME	PENE.	0.31	0.52	0.31	0.00
	INCI.	0.44	0.66	0.44	0.00
WATERBIRD	WATER	0.36	0.63	0.36	0.00
	SEA	0.39	0.64	0.39	0.00

these tasks. Specifically, for each task, we split its data into four environments: $E_1^{\text{train}}, E_2^{\text{train}}, E^{\text{val}}, E^{\text{test}}$, where spurious correlations vary across the two training environments $E_1^{\text{train}}, E_2^{\text{train}}$. For the source task S, the model can access both of its training environments $E_1^{\text{train}}(S), E_2^{\text{train}}(S)$. For the target task T, the model only has access to one training environment $E_1^{\text{train}}(T)$. We note that the validation set $E^{\text{val}}(T)$ plays an important role in early-stopping and hyper-parameter tuning, especially when the distribution of the data is different between training and testing [45]. In this work, since we don't have access to multiple training environments on the target task, we assume that the validation data E^{val} follows the same distribution as the training data E_1^{train}. Table 3.1 summarizes the level of the spurious correlations for different tasks. Additional details can be found in Appendix B.2.1.[2]

Natural environments We also consider a practical setting where environments are directly defined by a given attribute of the input, and our goal is to reduce model biases from other latent attributes. We study CelebA [76] where each input (an image

[2]All data splits, hyper-parameter search spaces are available in the supplementary materials.

of a human face) is annotated with 40 binary attributes. The source task is to predict the `Eyeglasses` attribute and the target task is to predict the `BlondHair` attribute. We use the `Young` attribute to define two environments: $E_1 = \{\texttt{Young} = 0\}$ and $E_2 = \{\texttt{Young} = 1\}$. In the source task, both environments are available. In the target task, we only have access to environment E_1 during training and validation. At test time, we evaluate the robustness of our target classifier against other latent attributes. Specifically, for each unknown attribute such as `Male`, we partition the testing data into four groups: $\{\texttt{Male} = 1, \texttt{BlondHair} = 0\}$, $\{\texttt{Male} = 0, \texttt{BlondHair} = 0\}$, $\{\texttt{Male} = 1, \texttt{BlondHair} = 1\}$, $\{\texttt{Male} = 0, \texttt{BlondHair} = 1\}$. Following [93], We report the worst-group accuracy and the average-group accuracy.

3.4.2 Baselines

We compare our algorithm against the following baselines. For fair comparison, all methods share the same representation backbone and hyper-parameter search space. Implementation details are available at Appendix B.2.2.

ERM baseline We learn a classifier on the target task from scratch by minimizing the average loss across all examples. Note that this classifier is independent of the source task. Its performance reflects the deviation between the training distribution and the testing distribution of the target task.

Transfer methods Since the source task contains multiple environments, we can learn a stable model on the source task and transfer it to the target task. We use four algorithms to learn the source task: DANN [41], C-DANN [72], MMD [71], PI [9]. We consider three standard methods for transferring the source knowledge:

REUSE: We directly transfer the feature extractor of the source model to the target task. The feature extractor is fixed when learning the target classifier.

FINETUNE: We update the feature extractor when training the target classifier. [100] has shown that FINETUNE may improve adversarial robustness of the target task.

MULTITASK: We adopt the standard multi-task learning approach [22] where the source model and the target model share the same feature extractor and are jointly trained together.

Automatic de-biasing methods For the target task, we can also apply de-biasing approaches that do not require environments. We consider the following baselines:

EIIL [30]: Based on a pre-trained ERM classifier's prediction logits, we infer an environment assignment that maximally violates the invariant learning principle [6]. We then apply group DRO to minimize the worst-case loss over all inferred environments.

GEORGE [108]: We use the feature representation of a pre-trained ERM classifier to cluster the training data. We then apply group DRO to minimize the worst-case loss over all clusters.

LFF [85]: We train a biased classifier together with a de-biased classifier. The biased classifier amplifies its bias by minimizing the generalized cross entropy loss. The de-biased classifier then up-weights examples that are mis-labeled by the biased classifier during training.

M-ADA [90]: We use a Wasserstein auto-encoder to generate adversarial examples. The de-biased classifier is trained on both the original examples and the generated examples.

DG-MMLD [79]: We iteratively divide target examples into latent domains via clustering, and train the domain-invariant feature extractor via adversarial learning.

ORACLE For synthetic environments, we can use the spurious feature to define groups and train an oracle model. For example, in task SEABIRD, this oracle model will minimize the worst-case risk over the following four groups: {`seabird` in `water`}, {`seabird` in `land`} {`landbird` in `water`}, {`landbird` in `land`}. This oracle model helps us analyze the performance of our proposed algorithm separately from the inherent limitations (such as model capacity and data size).

Table 3.2: Target task accuracy of different methods. All methods are tuned based on a held-out validation set that follows from the same distribution as the target training data. Bottom right: standard deviation across 5 runs. Upper right: source task testing performance (if applicable).

	SOURCE	TARGET	ERM	REUSE$_{\text{PI}}$	FINETUNE$_{\text{PI}}$	MULTITASK	TOFU	ORACLE
MNIST	ODD	EVEN	$12.3_{\pm 0.6}$	$14.4^{(70.9)}_{\pm 1.0}$	$11.2^{(70.1)}_{\pm 2.1}$	$11.6^{(69.6)}_{\pm 0.6}$	$\mathbf{69.1}_{\pm 1.6}$	$68.7_{\pm 0.9}$
	EVEN	ODD	$9.7_{\pm 0.6}$	$19.2^{(71.1)}_{\pm 2.3}$	$11.5^{(71.1)}_{\pm 1.2}$	$10.1^{(70.0)}_{\pm 0.7}$	$\mathbf{66.8}_{\pm 0.8}$	$67.8_{\pm 0.5}$
BEER REVIEW	LOOK	AROMA	$55.5_{\pm 1.7}$	$31.9^{(70.1)}_{\pm 1.0}$	$53.7^{(70.1)}_{\pm 1.4}$	$54.1^{(76.0)}_{\pm 2.2}$	$\mathbf{75.9}_{\pm 1.4}$	$77.3_{\pm 1.3}$
	LOOK	PALATE	$46.9_{\pm 0.3}$	$22.8^{(70.0)}_{\pm 1.9}$	$49.3^{(73.2)}_{\pm 2.1}$	$52.8^{(73.3)}_{\pm 2.9}$	$\mathbf{73.8}_{\pm 0.7}$	$74.0_{\pm 1.2}$
	AROMA	LOOK	$63.9_{\pm 0.6}$	$40.1^{(68.6)}_{\pm 3.1}$	$65.2^{(66.4)}_{\pm 1.8}$	$64.0^{(71.5)}_{\pm 0.6}$	$\mathbf{80.9}_{\pm 0.5}$	$80.1_{\pm 0.6}$
	AROMA	PALATE	$46.9_{\pm 0.3}$	$14.0^{(68.3)}_{\pm 2.4}$	$47.9^{(63.2)}_{\pm 3.3}$	$50.0^{(71.2)}_{\pm 1.4}$	$\mathbf{73.5}_{\pm 1.1}$	$74.0_{\pm 1.2}$
	PALATE	LOOK	$63.9_{\pm 0.6}$	$40.4^{(57.2)}_{\pm 2.8}$	$64.3^{(60.1)}_{\pm 2.7}$	$63.1^{(75.9)}_{\pm 1.0}$	$\mathbf{81.0}_{\pm 1.0}$	$80.1_{\pm 0.6}$
	PALATE	AROMA	$55.5_{\pm 1.7}$	$23.1^{(59.2)}_{\pm 3.3}$	$54.5^{(58.7)}_{\pm 1.2}$	$56.5^{(73.3)}_{\pm 1.3}$	$\mathbf{76.9}_{\pm 1.5}$	$77.3_{\pm 1.3}$
ASK.	PENE	INCI.	$79.3_{\pm 1.3}$	$71.7^{(72.7)}_{\pm 0.5}$	$79.3^{(71.2)}_{\pm 0.8}$	$71.1^{(73.5)}_{\pm 1.4}$	$\mathbf{83.2}_{\pm 1.8}$	$84.8_{\pm 1.2}$
	INCI.	PENE.	$71.6_{\pm 1.8}$	$64.1^{(83.4)}_{\pm 1.5}$	$72.0^{(83.4)}_{\pm 3.1}$	$61.9^{(82.4)}_{\pm 0.7}$	$\mathbf{78.1}_{\pm 1.4}$	$78.3_{\pm 0.9}$
BIRD	WATER	SEA	$81.8_{\pm 4.3}$	$87.8^{(99.5)}_{\pm 1.1}$	$82.0^{(99.5)}_{\pm 4.0}$	$88.0^{(99.5)}_{\pm 0.9}$	$\mathbf{93.1}_{\pm 0.4}$	$93.7_{\pm 0.7}$
	SEA	WATER	$75.1_{\pm 6.3}$	$94.6^{(93.3)}_{\pm 1.6}$	$78.2^{(93.1)}_{\pm 8.1}$	$93.5^{(92.7)}_{\pm 1.9}$	$\mathbf{99.0}_{\pm 0.4}$	$98.9_{\pm 0.5}$
	Average		55.2	43.7	55.8	56.4	**79.3**	79.6

3.5 Results

Table 3.2 summarizes our results on synthetic environments. We observe that standard transfer methods fail to improve over the ERM baseline. On the other hand, TOFU consistently achieves the best performance across 12 transfer settings, outperforming the best baseline by 22.9%. While TOFU doesn't have access to the unstable features, by inferring them from the source environments, it matches the oracle performance with only 0.30% absolute difference.

Table 3.3 presents our results on natural environments. This task is very challenging as there are multiple latent spurious attributes in the training data. We observe that most automatic de-biasing methods underperform the ERM baseline. With the help of the source task, FINETUNE and MULTITASK achieve slightly better performance than ERM. TOFU continues to shine in this real-world setting: achieving the

Table 3.3: Worst-group and average-group accuracy on CelebA. The source task is to predict `Eyeglasses` and the target task is to predict `BlondHair`. We use the attribute `Young` to define two environments: $E_1 = \{\texttt{Young} = 0\}$, $E_2 = \{\texttt{Young} = 1\}$. Both environments are available in the source task. In the target task, we only have access to E_1 during training and validation.. We show the results for the first 3 attributes (alphabetical order). The right-most Average* column is computed based on the performance across all 38 attributes. See Appendix B.4 for full results.

	METHOD	`ArchedEyebrows`		`Attractive`		`BagsUnderEyes`		Average*	
		Worst	Average	Worst	Average	Worst	Average	Worst	Average
	ERM	75.43	88.52	75.00	88.94	70.91	87.09	61.01	85.07
TRANSFER	REUSE$_{\text{PI}}$	53.71	64.05	52.13	64.85	52.50	66.88	47.58	64.14
	REUSE$_{\text{DANN}}$	59.56	72.44	62.03	72.26	64.58	73.83	55.27	72.31
	REUSE$_{\text{C-DANN}}$	56.02	67.07	57.78	67.90	57.50	68.33	53.22	68.56
	REUSE$_{\text{MMD}}$	48.91	59.80	48.46	61.51	58.74	63.11	50.61	61.27
	FINETUNE$_{\text{PI}}$	71.86	87.02	72.73	87.34	62.50	84.10	63.07	85.27
	FINETUNE$_{\text{DANN}}$	65.38	83.89	63.35	84.98	56.86	81.34	50.60	80.49
	FINETUNE$_{\text{C-DANN}}$	73.85	88.90	75.61	89.39	75.86	88.14	62.03	85.57
	FINETUNE$_{\text{MMD}}$	76.07	88.80	74.33	89.74	78.57	88.61	66.80	86.81
	MULTITASK	69.66	86.91	72.73	87.44	70.00	85.21	64.37	85.21
AUTO-DEBIAS	EIIL	64.71	85.12	64.43	85.96	66.67	83.90	57.62	83.22
	GEORGE	74.73	87.89	73.66	87.70	77.78	87.97	63.34	85.04
	LFF	45.41	60.23	47.67	60.16	42.59	60.72	42.52	62.04
	M-ADA	64.61	83.33	67.33	83.59	70.34	85.34	54.55	80.77
	DG-MMLD	69.51	87.38	68.42	87.50	63.41	84.78	55.69	83.51
	TOFU	**85.66**	**91.47**	**88.30**	**92.76**	**90.38**	**92.41**	**84.86**	**91.71**

best worst-group and average-group performance.

Is TOFU able to identify the unstable features? Yes. For synthetic environments, we visualize the unstable feature representation produced by $f_{\mathcal{Z}}$ on MNIST EVEN. Figure 3-3 demonstrates that while $f_{\mathcal{Z}}$ only sees source examples (ODD) during training, it can distribute target examples based on their unstable color features.

For natural environments, we visualize the distribution of two latent attributes (`Male` and `ArchedEyebrows`) over the generated clusters. Figure 3-4 shows that the distribution gap of the unknown attribute `Male` across the generated partitions is 2%

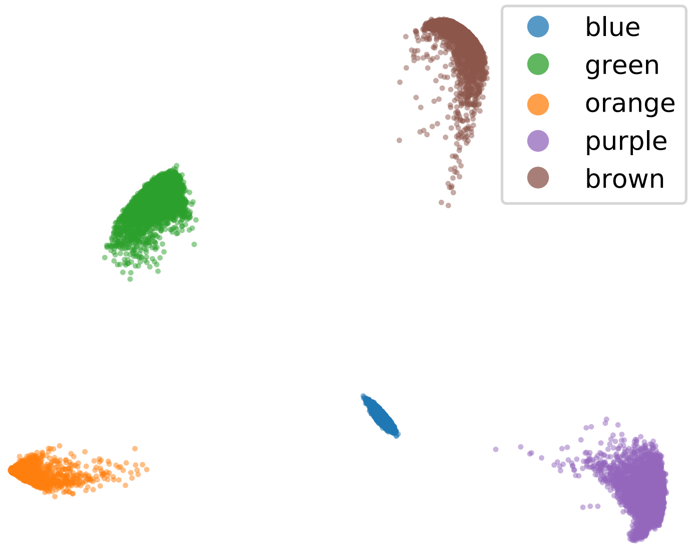


Figure 3-3: PCA visualization of the unstable feature representation $f_{\mathcal{Z}}$ for examples in MNIST EVEN. $f_{\mathcal{Z}}$ is trained on MNIST ODD. TOFU identifies the hidden spurious color feature by contrasting different source environments.

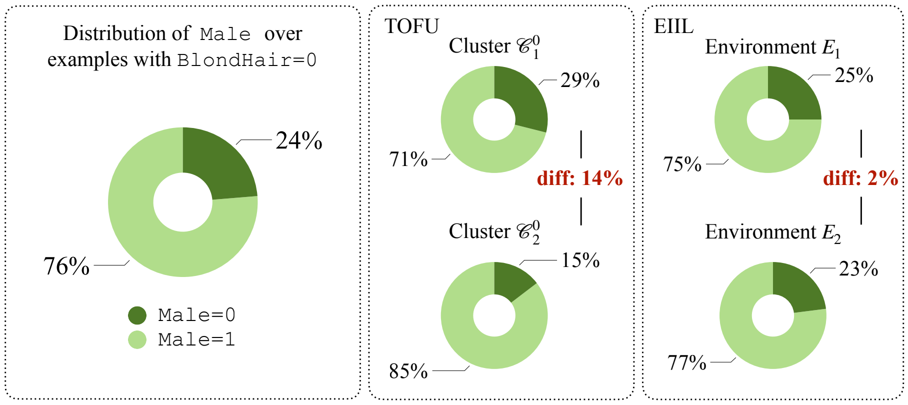


Figure 3-4: Visualization of the unknown attribute `Male` on CelebA. Left: distributions of `Male` in the training data. Mid: partitions learned by TOFU. Right: partitions learned by EIIL. TOFU generates partitions that are more informative of the unknown attribute (14% vs. 2%).

for EIIL, only marginally better than random splitting (0%). By leveraging information from the source task, TOFU learns partitions that are more informative of the unknown attribute (14%).

How do the generated clusters compare to the oracle groups? We quantitatively evaluate the generated clusters based on three metrics: *homogeneity* (each

Table 3.4: Quantitative evaluation of the generated clusters against the ground truth unstable features. For comparison, the TRIPLET baseline (TRP) directly encourages source examples with the same label to stay close to each other in the feature space, from which we generate the clusters. For both methods, we generate two clusters for each target label value and report the average performance across all label values. We observe that the TRIPLET representation, while biased by the spurious correlations, fails to recover the ground truth unstable features for some tasks. By explicitly contrasting the source environments, TOFU derives clusters that are highly-informative of the unstable features.

SOURCE	TARGET	Homogeneity		Completeness		V-measure	
		TRP	TOFU	TRP	TOFU	TRP	TOFU
ODD	EVEN	0.42	**0.68**	0.58	**0.95**	0.49	**0.79**
EVEN	ODD	0.67	**0.67**	0.93	**0.99**	0.78	**0.80**
LOOK	AROMA	0.33	**0.92**	0.28	**0.92**	0.30	**0.92**
LOOK	PALATE	0.33	**0.90**	0.27	**0.89**	0.30	**0.90**
AROMA	LOOK	0.33	**1.00**	0.28	**1.00**	0.30	**1.00**
AROMA	PALATE	0.82	**1.00**	0.77	**1.00**	0.79	**1.00**
PALATE	LOOK	0.83	**0.98**	0.77	**0.98**	0.80	**0.98**
PALATE	AROMA	0.82	**0.95**	0.77	**0.95**	0.79	**0.95**

cluster contain only examples with the same unstable feature value), *completeness* (examples with the same unstable feature value belong to the same cluster), and *V-measure* (the harmonic mean of homogeneity and completeness). From Table 3.4, we see that TOFU is able to derive clusters that resemble the oracle groups on BEER REVIEW. In MNIST, since we generate two clusters for each label value and there are five different colors, it is impossible to recover the oracle groups. However, TOFU still achieves almost perfect completeness.

3.6 Discussion

Are biases shared across real-world tasks? In this chapter, we show that for tasks where the biases are shared, we can effectively transfer this knowledge to obtain a more robust model. This assumption holds in many real world applications. For example, in natural language processing, the same gender bias exist across many tasks including relation extraction [42], semantic role labeling [56], abusive language detection [87] and sentiment analysis [63]. In computer vision, the same geographical bias exists across different object recognition benchmarks such as ImageNet, COCO and OpenImages [31].

When a single source task does not describe all unwanted unstable features, we can leverage multiple source tasks and compose their individual unstable features together. We can naturally extend TOFU to accomplish this goal by learning the unstable feature representation jointly across this collection of source tasks. We focus on the basic setting in this thesis and leave the extension to future work.

Our approach is not applicable in situations where the biases in the source task and the target task are completely disjoint.

What if the source task and target task are from different domains? In this chapter, we focus on the setting where the source task and the target task are from the same domain. If the target task is drawn from a different domain, we can use domain-adversarial training to align the distributions of the unstable features across the source domain and the target domain [71, 71]. Specifically, when training the unstable feature representation f_z, we can introduce an adversarial player that tries to guess the domain label from f_z. The representation f_z is updated to fool this adversarial player in addition to minimize the triplet loss in Eq (3.1). We leave this extension to future work.

Can we apply domain-invariant representation learning (DIRL) directly to the source environments? Domain invariant representation learning [41, 72, 71] aims to match the feature representations across domains. If we directly treat environments as domains and apply these methods, the resulting representation may still encode unstable features.

For example, in CelebA, the attribute `Male` is spuriously correlated with the target attribute `BlondHair` (Women are more likely to have blond hair than men in this dataset). Given the two environments $\{$`Young` $= 0\}$ and $\{$`Young`$=1\}$, DIRL learns an age-invariant representation. However, if the distribution of `Male` is the same across the two environments, DIRL will encode this attribute into the age-invariant representation (since it is helpful for predicting the target `BlondHair` attribute). In our approach, we realize that the the correlations between `Male` and `BlondHair` are different in the two environments (The elderly may have more white hair). Even though the distribution of `Male` may be the same, we can still identify this bias from the classifiers' mistakes. Empirically, Table 3.3 shows that while DIRL methods improve over the ERM baseline, they still perform poorly on minority groups (worst case acc 66.80% on CelebA).

What if the mistakes correspond to other factors such as label noise, distribution shifts, etc.? For ease of analysis, we do not consider label noise and distribution shift in Theorem 3. One future direction is to model bias from the information perspective (rather than looking at the linear correlations). This will enable us to relax the assumption in the analyses and we can further incorporate these different mistake factors into the modeling.

We note that we do not impose this assumption in our empirical experiments. For example, we explicitly added label noise into the MNIST data. In CELEBA, there is a distribution gap (from young people to the elderly) across the two environments. We observe that our method is able to perform robustly in situations where the assumption breaks.

Is the algorithm efficient when multiple source environments are available? Our method can be generalized efficiently to multiple environments. Given N source environments, we first note that the complexities of the target steps **T.1** and **T.2** are independent of N. For the source task, the N environment-specific classifiers (in **S.1**) can be learned jointly with multi-task learning [22]. This significantly reduces the inference cost at **S.2** as we only need to pass each input example through the (expensive) representation backbone for one time. In **S.3**, we sample partitions when minimizing the triplet loss, so there is no additional cost during training. In this chapter, we focus on the two-environments setup for simplicity and leave this generalization to future work.

Why does the baselines perform so poorly on MNIST? We note that the representation backbone (a 2-layer CNN) on MNIST is trained from scratch while we use pre-trained representations for other datasets (see Appendix B.2.2). Our hypothesis is that models are more prune to spurious correlations when trained from scratch.

3.7 Conclusion

Reducing model bias is a critical problem for many machine learning applications in the real world. In this chapter, we recognize that related tasks often share similar biases. We demonstrate that we can effectively transfer this knowledge and improve the robustness of the target model without additional human intervention. Compared with 15 baselines across 5 datasets, our approach consistently delivers significant performance gain. Quantitative and qualitative analyses confirm that our method is able to identify the hidden biases.

Chapter 4

Learning to Split for Automatic Bias Detection

Classifiers are biased when trained on biased datasets. As a remedy, we propose Learning to Split (`ls`), an algorithm for automatic bias detection. Given a dataset with input-label pairs, `ls` learns to split this dataset so that predictors trained on the training split *generalize poorly* to the testing split. This performance gap provides a proxy for measuring the degree of bias in the learned features and can therefore be used to reduce biases. Identifying non-generalizable splits is challenging as we don't have any explicit annotations about how to split. In this work, we show that the prediction correctness of the testing example can be used as a source of weak supervision: generalization performance will drop if we move examples that are predicted correctly away from the testing split, leaving only those that are mispredicted. We evaluate our approach on Beer Review, Tox21, Waterbirds, CelebA and MNLI. Empirical results show that `ls` is able to generate astonishingly challenging splits that correlate with human-identified biases. Moreover, we demonstrate that combining robust learning algorithms (such as group DRO) with splits identified by `ls` enables automatic de-biasing. Compared with previous state-of-the-arts, we substantially improves the worst-group performance (23.4% on average) when the source of biases is unknown during training and validation.

4.1 Introduction

Recent work has shown great success on de-biasing when the source of bias (e.g., gender, race, etc.) is known a priori [92, 93, 29, 48, 78, 60]. In practice, however, the task of bias identification itself is not only time consuming but also challenging: it requires expert knowledge of the task and private details about the annotation procedures [122, 96]. In this work, we study *automatic bias detection*. Given a classification dataset with only input-label pairs, we would like to detect biases that may hinder predictors' generalization performance.

We propose Learning to Split (`ls`), an algorithm that simulates generalization failure directly from the set of input-label pairs. Specifically, `ls` learns to split the dataset so that predictors trained on the training split *cannot generalize* to the testing split (Figure 4-1). This performance gap provides a proxy for measuring the degree of bias in the learned features and can therefore be used to identify unknown biases.

The problem however is that there are many trivial splits. E.g., poor generalization can result from a train split that is much smaller than the test split (Figure 4-2b). In binary classification, if the train split contains all the positive examples and the test split contains all the negative examples, the model will not generalize (Figure 4-2c). The poor generalization of these trivial solutions arise from the lack of training data and label imbalance, and they do not reveal the hidden biases. To ensure that the

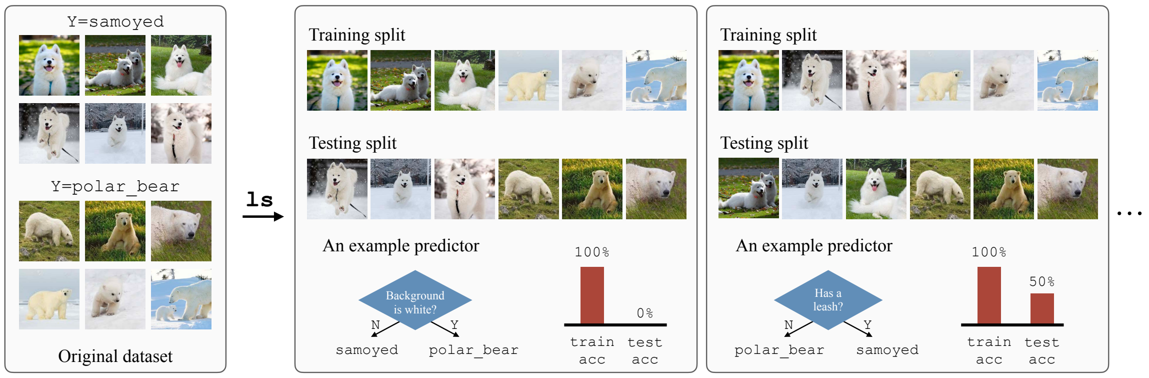
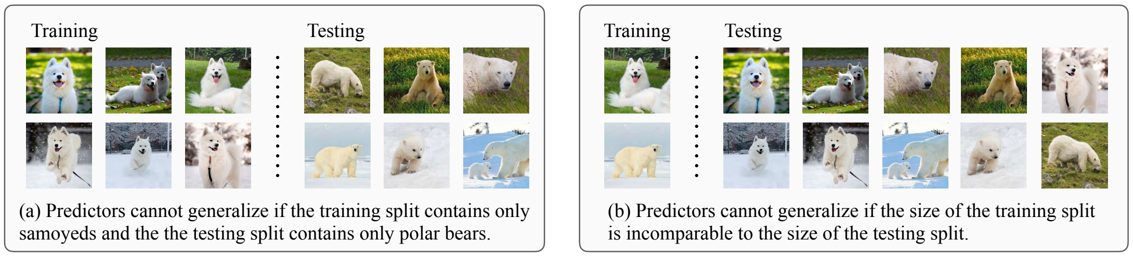


Figure 4-1: Consider the task of classifying samoyeds vs. polar bears. Given the set of image-label pairs, our algorithm `ls` *learns* to *split* the data so that predictors trained on the training split *cannot generalize* to the testing split.



(a) Predictors cannot generalize if the training split contains only samoyeds and the the testing split contains only polar bears.

(b) Predictors cannot generalize if the size of the training split is incomparable to the size of the testing split.

Figure 4-2: Not all splits are helpful for revealing the hidden biases. (a) Predictors cannot generalize when the label is imbalanced across training and testing. (b) Predictors cannot generalize if the amount of annotations is insufficient. `ls` poses two regularity constraints to avoid such splits: 1) the marginal distribution of the label should be similar across the splits; 2) the training split and testing split should have comparable sizes.

splits are meaningful, we impose two regularity constraints on the splits. First, the size of the train split must be comparable to the size of the test split. Second, the marginal distribution of the label should be the same across the splits.

Our algorithm `ls` consists of two components, *Splitter* and *Predictor*. At each iteration, the Splitter creates a train-test split of the dataset. Given an input-label pair, it decides whether this example should be used for training or testing. The Predictor then takes the training split and learns how to predict the label from the input. Its prediction performance on the testing split is used to inform the Splitter to generate a more challenging split (under the regularity constraints) for the next iteration. Specifically, while we don't have any explicit annotations for creating non-generalizable splits, we show that the prediction correctness of each testing example can serve as a source of weak supervision: generalization performance will drop if we move examples that are predicted correctly away from the testing split, leaving only those that are mispredicted.

We conduct experiments on NLP, vision and chemistry tasks. Given only the set of input-label pairs, `ls` consistently identifies splits that predictors cannot generalize across. For example in MNLI, the generalization performance drops from 79.4% (random split) to 27.8% (`ls`) for a standard BERT-based predictor [34]. Further analyses confirm that our learned splits correlate with human-identified biases. Next, we demonstrate that combining group distributionally robust optimization (DRO) with splits identified by `ls` enables automatic de-biasing. Compared with previous state-of-the-arts, we substantially improves the worst-group performance (23.4% on average) when the source of biases is completely unknown during training and validation.

4.2 Related work

De-biasing Modern datasets are often coupled with unwanted biases [21, 98, 83, 120]. If the biases have already been identified, we can use this prior knowledge to reduce their impact [67, 54, 86, 15, 109, 29, 48, 78, 94]. The challenge arises when the source of biases is unknown. Previous work has shown that the mistakes of a standard ERM predictor on the training data are informative of the biases [9, 97, 85, 112, 74]. By boosting from its mistakes, we can obtain a more robust model. In addition, we can analyze the predictor's hidden activations to identify under-represented groups [30, 108, 2, 79]. We can also leverage the hidden representation of a generative model to identify biases [73]. However, many other factors (such as the initialization, the representation power, the amount of annotations, difficulty of the example, etc) can contribute to the predictor's mistakes.

In this work, instead of looking at the training statistics of the predictor, we focus on its *generalization gap* from the training split to the testing split. This effectively balances out those unwanted factors. Going back to the previous example, if the training and testing splits share the same distribution, the generalization gap will be small for predictors that under-fit. The gap will increase only when the training and testing splits have different prediction characteristics. Moreover, instead of committing to a fixed predictor, we iteratively refine the predictor during training so that it faithfully measures the generalization gap given the current Splitter.

Data splitting In many applications, the annotations available are often limited compared to the extremely diverse universe of samples that we would want to make predictions on. In addition, instead of focusing solely on the average performance, we care about models' robustness on under-represented populations. To properly evaluate generalization, researchers have developed different heuristics for splitting the data. In chemistry, molecules are split based on their scaffold structure and experiment time [103, 120]. In medical imaging, patients are split according to their hospital [7, 118]. In biology, cells are split based on their batch ID in order to reduce the batch effects from high-throughput screening [111, 64]. Different from these approaches that rely on human-specified clues, our algorithm `ls` learns how to split directly from the given dataset and can therefore be applied to scenarios when such human knowledge are not available or incomplete.

Meta learning Learning to Split (`ls`) naturally involves a bi-level optimization problem [40, 106, 53]. Past work has successfully applied meta learning to learn model initializations [39], optimizers [5], metric spaces [113, 107], network architectures [75], instance weights [92, 57, 104], teaching policies [38]. In these methods, the inner-loop and outer-loop models cooperate with each other. In this work, our outer-loop Splitter plays an adversarial game [44] against the inner-loop Predictor. We learn how to split the data so that predictors will *fail* to generalize.

4.3 Learning to Split

4.3.1 Motivation

Given a dataset $\mathcal{D}^{\text{total}}$ with input-label pairs $\{(x, y)\}$, our goal is to split this dataset into two subsets, $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, such that predictors learned on the training split $\mathcal{D}^{\text{train}}$ cannot generalize to the testing split $\mathcal{D}^{\text{test}}$.

Why do we have to discover such splits? Before deploying our trained models, it is crucial to understand the extent to which these models can even generalize within the given dataset. Standard cross-validation technique attempts to measure generalization by randomly splitting the dataset [110, 4]. However, this measure only reflects the *average* performance under the same data distribution $\mathbb{P}_{\mathcal{D}^{\text{total}}}(x, y)$. There is no performance guarantee if our data distribution shifts at test time (e.g., up-

Algorithm 1 Learning to Split (`ls`)

```
Input: dataset $\mathcal{D}^{\text{total}}$
Output: data splits $\mathcal{D}^{\text{train}}$, $\mathcal{D}^{\text{test}}$
 1: Initialize Splitter as random splitting
 2: repeat
 3:     Apply Splitter to split $\mathcal{D}^{\text{total}}$ into $\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}$.
 4:     Train Predictor from scratch on $\mathcal{D}^{\text{train}}$ using empirical risk minimization.
 5:     Evaluate Predictor on $\mathcal{D}^{\text{test}}$ and compute its generalization gap.
 6:     repeat
 7:         Sample a mini-batch from $\mathcal{D}^{\text{total}}$ to compute the regularity constraints $\Omega_1, \Omega_2$
            (Eq 4.1).
 8:         Sample another mini-batch from $\mathcal{D}^{\text{test}}$ to compute $\mathcal{L}^{\text{gap}}$ (Eq 4.2).
 9:         Update Splitter to minimize the overall objective $\mathcal{L}^{\text{total}}$ (Eq 4.3).
10:     until $\mathcal{L}^{\text{total}}$ stops improving
11: until gap stops improving
```

weighting the minority group). For example, consider the task of classifying samoyeds vs. polar bears (Figure 4-1). Models can achieve good average performance by using spurious heuristics such as "polar bears live in snowy habitats" and "samoyeds play on grass". Discovering splits that models cannot generalize across helps us identify under-represented groups (polar bears that appear on grass).

How to discover such splits? Our algorithm `ls` has two components, a *Splitter* that decides how to split the dataset and a *Predictor* that estimates the generalization gap from the training split to the testing split.[1] At each iteration, the splitter uses the feedback from the predictor to update its splitting decision. One can view this splitting decision as a latent variable that represents the prediction characteristic of each input. The algorithm is unsupervised in the sense that we assume no explicit annotations about how to split the examples. To avoid degenerate solutions, we require the Splitter to satisfy two regularity constraints that are often preserved in common benchmarks: the size of the training split should be comparable to the size of the testing split (Figure 4-2a); the marginal distribution of the label should be similar across the splits (Figure 4-2b).

[1] To prevent over-fitting, we held-out 1/3 of the training split for early-stopping when training the Predictor.

4.3.2 Splitter and Predictor

Here we describe the two key components of our algorithm, *Splitter* and *Predictor*, in the context of classification tasks. The algorithm itself generalizes to regression problems as well.

Splitter Given a list of input-label pairs $\mathcal{D}^{\text{total}} = [(x_1, y_2), \ldots, (x_n, y_n)]$, the Splitter decides how to partition this dataset into a training split $\mathcal{D}^{\text{train}}$ and a testing split $\mathcal{D}^{\text{test}}$. We can view its splitting decisions as a list of latent variables $\mathbf{z} = [z_1, \ldots, z_n]$ where each $z_i \in \{0, 1\}$ indicates whether example (x_i, y_i) is included in the training split or not. In this work, we assume independent selections for simplicity. That is, the Splitter takes one input-label pair (x_i, y_i) at a time and predicts the probability $\mathbb{P}(z_i \mid x_i, y_i)$ of keeping this example in the training split. We can factor the joint probability of our splitting decisions as

$$\mathbb{P}(\mathbf{z} \mid \mathcal{D}^{\text{total}}) = \prod_{i=1}^{n} \mathbb{P}(z_i \mid x_i, y_i).$$

In order to obtain the splits $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, we directly sample from the Splitter's predictions $\mathbb{P}(z_i \mid x_i, y_i)$. Note that while the splitting decisions are independent across different examples, the Splitter receives *global* feedback, dependent on the entire dataset $\mathcal{D}^{\text{total}}$, from the Predictor during training.

Predictor The Predictor takes an input x and predicts the probability of its label $\mathbb{P}(y \mid x)$. Given the Splitter's current splitting decisions, we *initialize* the Predictor and train it to minimize the empirical risk on the training split $\mathcal{D}^{\text{train}}$. We note that this initialization step is critical as it ensures that the Predictor does not carry over past information (from the previous splits) and is faithful of representing the current generalization gap. After training, we evaluate the generalization performance of the Predictor on the testing split $\mathcal{D}^{\text{test}}$. The goal of this Predictor is to provide feedback for the Splitter so that it can generate more challenging splits at the next iteration.

4.3.3 Degenerate solutions

Many factors can impact generalization, but not all of them are of interest. For example, the Predictor may naturally fail to generalize due to the lack of training data or due to label imbalance across the splits (Figure 4-2). To avoid these trivial solutions, we introduce two regularizers to shape the Splitter's decisions:

$$\begin{aligned} \Omega_1 &= \mathrm{D}_{KL}(\mathbb{P}(z) \,\|\, \mathrm{Bernoulli}(\delta)), \\ \Omega_2 &= \mathrm{D}_{KL}(\mathbb{P}(y \mid z=1) \,\|\, \mathbb{P}(y)) + \mathrm{D}_{KL}(\mathbb{P}(y \mid z=0) \,\|\, \mathbb{P}(y)). \end{aligned} \tag{4.1}$$

The first term Ω_1 ensures that we have sufficient training examples in $\mathcal{D}^{\text{train}}$. Specifically, the marginal distribution $\mathbb{P}(z) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{P}(z_i = z \mid x_i, y_i)$ represents what percentages of $\mathcal{D}^{\text{total}}$ is splitted into $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$. We penalize the Splitter if it moves too far away from the prior distribution Bernoulli(δ). [23] suggest that minority groups usually constitute 25% of the entire population. In this work, we fix $\delta = 0.75$ in all experiments so that $\mathbb{E}[|\mathcal{D}^{\text{test}}|/|\mathcal{D}^{\text{total}}|] = 0.25$.

The second term Ω_2 aims to reduce label imbalance across the splits. It achieves this goal by pushing the label marginals in the training split $\mathbb{P}(y \mid z = 1)$ and the testing split $\mathbb{P}(y \mid z = 0)$ to be close to the original label marginal $\mathbb{P}(y)$ in $\mathcal{D}^{\text{total}}$. We can apply Bayes' rule to compute these conditional label marginals directly from the Splitter's decisions:

$$\mathbb{P}(y \mid z=1) = \frac{\sum_i \mathbb{1}_y(y_i)\mathbb{P}(z_i = 1 \mid x_i, y_i)}{\sum_i \mathbb{P}(z_i = 1 \mid x_i, y_i)}, \quad \mathbb{P}(y \mid z=0) = \frac{\sum_i \mathbb{1}_y(y_i)\mathbb{P}(z_i = 0 \mid x_i, y_i)}{\sum_i \mathbb{P}(z_i = 0 \mid x_i, y_i)}.$$

4.3.4 Training strategy

The only question that remains is how to learn the Splitter. Our goal is to produce *difficult* and *non-trivial* splits so that the Predictor cannot generalize. However, the challenge is that we don't have any explicit annotations for the splitting decisions.

There are a few options to address this challenge. From the meta learning perspective, we can back-propagate the Predictor's loss on the testing split directly to the Splitter. This process is expensive as it involves higher order gradients from the

Predictor's training. While one can apply episodic-training [113] to reduce the computation cost, the Splitter's decision will be biased by the size of the learning episodes (since the Predictor only operates on the sampled episode). From the reinforcement learning viewpoint, we can cast our objectives, maximizing the generalization gap while maintaining the regularity constraints, into a reward function [69]. However based on our preliminary experiments, the learning signal from this scalar reward is too sparse for the Splitter to learn meaningful splits.

In this work, we take a simple yet effective approach to learn the Splitter. Our intuition is that *the Predictor's generalization performance will drop if we move examples that are predicted correctly away from the testing split, leaving only those that are mispredicted.* In other words, we can view the prediction correctness of the testing example as a direct supervision for the Splitter. Formally, let $\hat{y}_i$ be the Predictor's prediction for input x_i: $\hat{y}_i = \arg\max_y \mathbb{P}(y \mid x_i)$. We minimize the cross entropy loss between the Splitter's decision and the Predictor's prediction correctness over the testing split:

$$\mathcal{L}^{\text{gap}} = \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{(x_i, y_i) \in \mathcal{D}^{\text{test}}} \mathcal{L}^{\text{CE}}(\mathbb{P}(z_i \mid x_i, y_i), \mathbb{1}_{y_i}(\hat{y}_i)). \tag{4.2}$$

Combining with the aforementioned regularity constraints, the overall objective for the Splitter is

$$\mathcal{L}^{\text{total}} = \mathcal{L}^{\text{gap}} + \Omega_1 + \Omega_2, \tag{4.3}$$

One can also explore different weighting schemes for the three loss terms [27]. In this work, we found that the simple summation in Eq (4.3) works well out-of-the-box across all our experiments. Algorithm 1 presents the pseudo-code of our algorithm. At each outer-loop (line 2-11), we start by using the current Splitter to partition $\mathcal{D}^{\text{total}}$ into $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$. We train the Predictor from scratch on $\mathcal{D}^{\text{train}}$ and evaluate its generalization performance on $\mathcal{D}^{\text{test}}$. For computation efficiency, we sample mini-batches in the inner-loop (line 6-10) and update the Splitter based on Eq (4.3).

4.4 Experiments

There are two main questions to be answered:

- Can `ls` identify splits that are not generalizable? (Section 4.4.2)
- Can we use the splits identified by `ls` to reduce unknown biases? (Section 4.4.3)

We conduct experiments over multiple modalities (Section 4.4.1). For lack of space we defer our dataset and implementation details to Appendix C.1.

4.4.1 Dataset

Beer review We use the BeerAdvocate review dataset introduced by previous work [82]. Each review describes multiple aspects of a beer and is written by a website user. Following previous work [69], we consider three aspect-level sentiment classification tasks: look, aroma and palate. There are 2,500 positive reviews and 2,500 negative reviews for each task. The average word count per review is 128.5. We apply `ls` to identify spurious splits for each task.

Tox21 We consider the molecular property prediction benchmark Tox21 [55]. There are 12707 chemical compounds in the dataset. Each compound is annotated with 12 binary properties which represent the outcomes of different toxicological experiments. We apply ls to identify spurious splits for each property.

Waterbirds [93] created this dataset by combining bird images from the Caltech-UCSD Birds-200-2011 (CUB) dataset [114] with backgrounds from the Places dataset [125]. The task is to predict waterbirds vs. landbirds. The challenge is that waterbirds (landbirds), by construction, appear more frequently with a water (land) background. As a result, machine learning models may utilize this spurious correlation to make their predictions. We combine the official training data and validation data (5994 examples in total) and apply `ls` to identify spurious splits.

[1]For fair comparison, all methods share the same hyper-parameter search space and representation backbone (`resnet-50` for Waterbirds and CelebA, `bert-base-uncased` for MNLI). See Appendix C.2 for details.

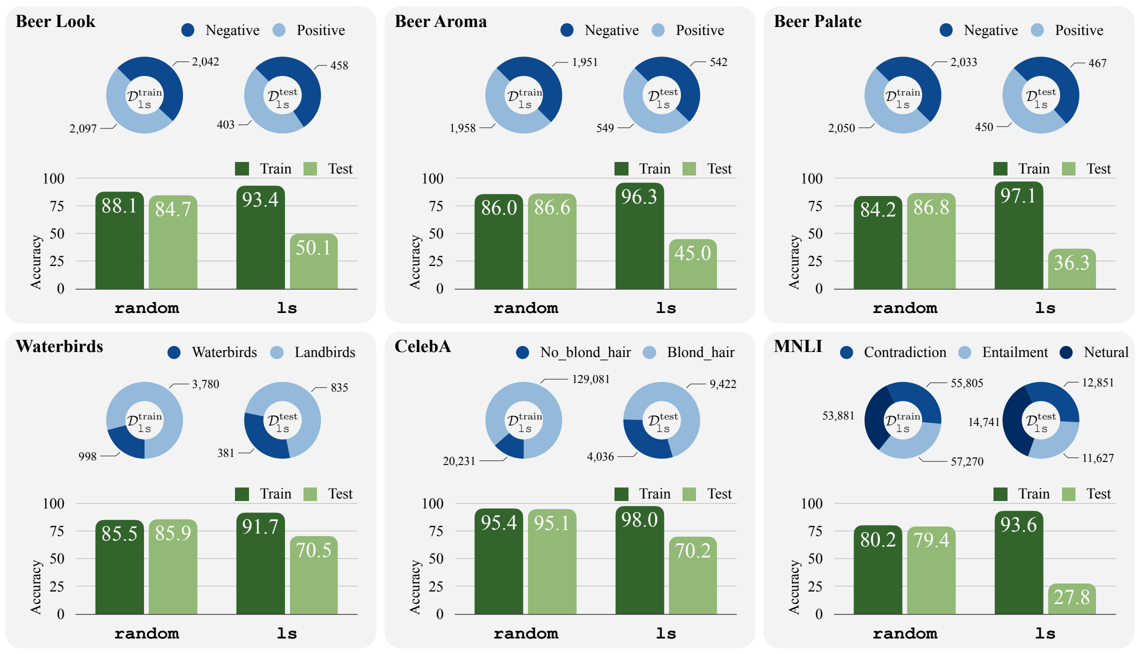


Figure 4-3: Given a set of input-label pairs, `ls` identifies splits that predictors cannot generalize across. These generated splits share similar label distributions (shown on the top). For comparison, predictors achieve similar training and testing performance on random splits (`random`). In both methods, to prevent memorization, we held-out 1/3 of the training split for early stopping.

CelebA CelebA is an image classification dataset where each input image (face) is paired with multiple human-annotated attributes [76]. Following previous work [93], we treat the hair color attribute ($y \in \{\texttt{blond}, \texttt{not\_blond}\}$) as our prediction target. The label is spuriously correlated with the gender attribute ($\{\texttt{male}, \texttt{female}\}$). We apply `ls` to identify spurious splits over the official training data (162,770 examples).

MNLI MNLI is a crowd-sourced collection of 433k sentence pairs annotated with textual entailment information [115]. The task is to classify the relationship between a pair of sentences: entailment, neutral or contradiction. Due to the artifacts of the data collection process, contradiction examples often include negation words [83]. We apply `ls` to identify spurious splits over the official training data (206,175 examples).

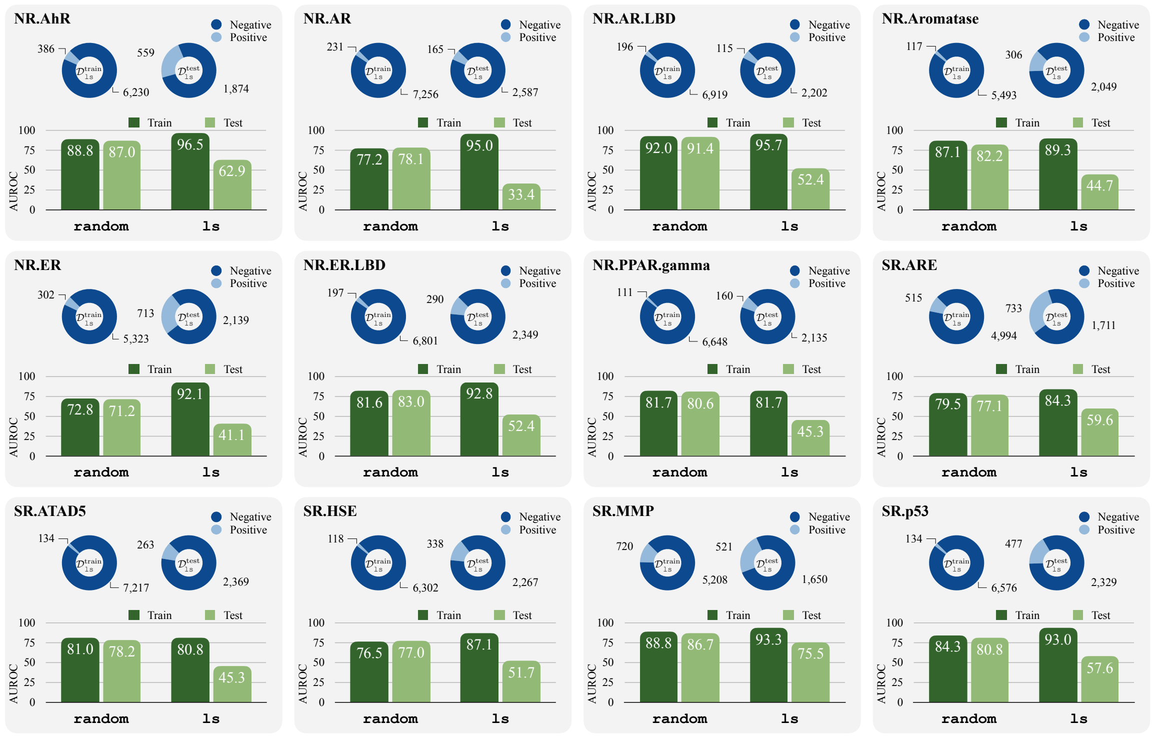


Figure 4-4: Our method can also be applied to the chemistry domain (Tox21). Similar to Figure 4-3, `ls` identifies non-trivial splits that predictors cannot generalize for all 12 molecular property prediction tasks.

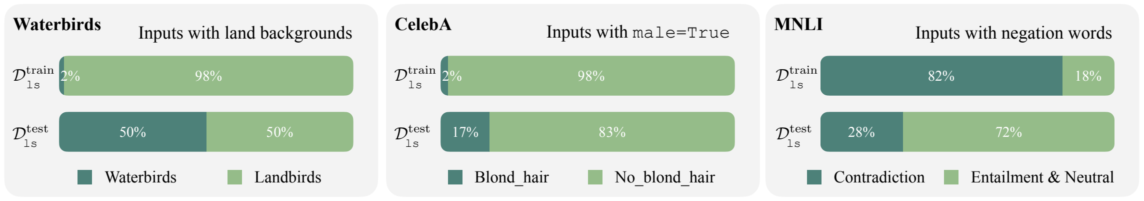


Figure 4-5: The splits learned by `ls` are correlated with human-identified biases. For example in Waterbirds (left), `ls` learns to *amplify* the spurious correlation between landbirds and land background in the training split $\mathcal{D}^{\text{train}}$. As a result, predictors will over-fit the background features and fail to generalize at test time ($\mathcal{D}^{\text{test}}$) when the spurious correlation is reduced.

4.4.2 Identifying non-generalizable splits

Figure 4-3 and Figure 4-4 present the splits identified by our algorithm `ls`. Compared to random splitting, `ls` achieves astonishingly higher generalization gaps across all tasks. Moreover, we observe that the learned splits are not degenerative: the training split $\mathcal{D}^{\text{train}}$ and testing split $\mathcal{D}^{\text{test}}$ share similar label distributions. This confirms the effectiveness of our regularity objectives.

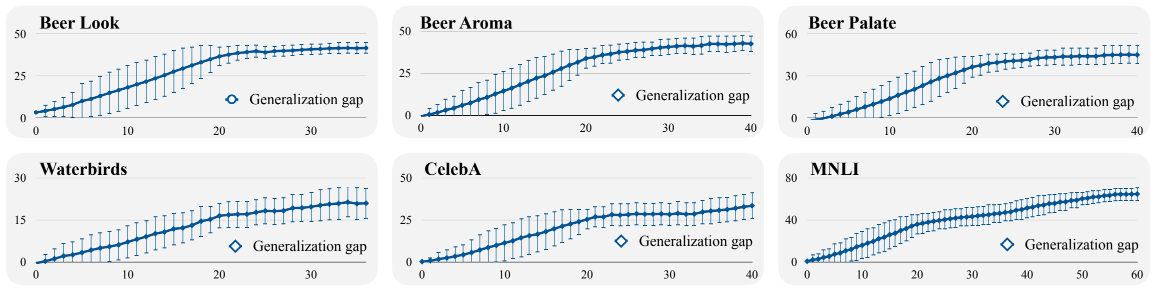


Figure 4-6: Learning curve of `ls`. X-axis: number of outer-loop iterations. Y-axis: generalization gap from $\mathcal{D}^{\text{train}}$ to $\mathcal{D}^{\text{test}}$. Error bar represents the standard deviation across 5 random seeds.

Why are the learrned splits so challenging for predictors to generalize across? While `ls` only has access to the set of input-label pairs, Figure 4-5 and Table 4.1 show that the learned splits are informative of human-identified biases. For example, in the generated training split of MNLI, inputs with negation words are mostly labeled as contradiction. This encourages predictors to leverage the presence of negation words to make their predictions. These biased predictors cannot generalize to the testing split, where inputs with negation words are mostly labeled as entailment or neutral.

Convergence and time-efficiency `ls` requires learning a new Predictor for each outer-loop iteration. While this makes `ls` more time-consuming than training a regular ERM model, this procedure guarantees that the Predictor faithfully measures the generalization gap based on the current Splitter. Figure 4-6 shows the learning curve of `ls`. We observe that the generalization gap steadily increases as we refine the Splitter and the learning procedure usually converges within 50 outer-loop iterations.

4.4.3 Automatic de-biasing

Once `ls` has identified the spurious splits, we can apply robust learning algorithms to learn models that are robust across the splits [93, 9]. Here we consider group distributionally robust optimization (group DRO) and study three well-established benchmarks: Waterbirds, CelebA and MNLI.

Table 4.1: The splits learned by `ls` on property `NR.AR` correlates with other properties that are not given to the algorithm. Specifically, `ls` allocates 58.3% of the actives into the training split and 41.2% of the actives into the testing split (top row). However, if we focus on the subset with `SR.ATAD5` active, this distribution shifts drastically to 17.1% vs. 82.9%.

	$\frac{\#\texttt{NR.AR}\text{ active in }\mathcal{D}^{\text{train}}}{\#\texttt{NR.AR}\text{ active}}$	$\frac{\#\texttt{NR.AR}\text{ active in }\mathcal{D}^{\text{test}}}{\#\texttt{NR.AR}\text{ active}}$
All examples	58.3%	41.6%
Subset with `NR.AhR` active	13.0%	87.0%
Subset with `NR.AR.LBD` active	82.3%	17.7%
Subset with `NR.Aromatas` active	62.5%	37.5%
Subset with `NR.ER` active	64.0%	36.0%
Subset with `NR.ER.LBD` active	75.3%	24.7%
Subset with `NR.PPAR.gam` active	40.0%	60.0%
Subset with `SR.ARE` active	45.5%	54.6%
Subset with `SR.ATAD5` active	17.1%	82.9%
Subset with `SR.HSE` active	66.7%	33.3%
Subset with `SR.MMP` active	53.6%	46.4%
Subset with `SR.p53` active	45.2%	54.8%

Group DRO Group DRO has shown strong performance when biases are annotated for training [93]. For example in CelebA where gender constitutes as a bias for predicting blond hair, group DRO uses the bias annotations to partition the training data into four groups: {`blond_hair`, `male`}, {`blond_hair`, `female`}, {`no_blond_hair`, `male`}, {`no_blond_hair`, `female`}. By minimizing the *worst-group* loss during training, it regularizes the impact of the unwanted gender bias.

Group DRO with bias predictor Recent work consider a more challenging setting where bias annotations are not provided at train time [70, 85, 30, 74]. However, they still access bias annotations on the validation data for model selection. With thousands of validation examples (1199 for Waterbirds, 19867 for CelebA, 82462 for MNLI), a simple baseline was overlooked by the community: learning a bias predictor over the validation data (where bias annotations are available) and using the predicted bias attributes on the training data to define groups for group DRO.

Table 4.2: Average and worst-group test accuracy for de-biasing.[2] Previous work improve the worst-group performances when the bias annotations are provided for the validation data. However, they still underperform the simple group DRO baseline that was overlooked. When bias annotations are not available for model selection, the performances of previous methods quickly drop to that of ERM. In contrast, applying group DRO with splits identified by `ls` substantially improves the worst-group performance. † denotes numbers reported by previous work.

Method	Bias annotated in train/val?	Waterbirds		CelebA		MNLI	
		Avg.	Worst	Avg.	Worst	Avg.	Worst
Group DRO [93]	✓/✓	93.5%†	**91.4%**†	92.9%†	**88.9%**†	81.4%†	**77.7%**†
ERM	✗/✓	97.3%†	72.6%†	95.6%†	47.2%†	82.4%†	67.9%†
CVaR DRO [70]	✗/✓	96.0%†	75.9%†	82.5%†	64.4%†	82.0%†	68.0%†
LfF [85]	✗/✓	91.2%†	78.0%†	85.1%†	77.2%†	80.8%†	70.2%†
EIIL [30]	✗/✓	96.9%†	78.7%†	89.5%	77.8%	79.4%	70.0%
JTT [74]	✗/✓	93.3%†	86.7%†	88.0%†	81.1%†	78.6%†	72.6%†
Group DRO (with predicted groups)	✗/✓	91.4%	**88.2%**	91.4%	**88.9%**	79.9%	**77.7%**
ERM	✗/✗	90.7%	64.8%	95.8%	41.1%	81.9%	60.4%
CVaR DRO [70]	✗/✗	—	62.0%†	—	36.1%†	81.8%	61.8%
LfF [85]	✗/✗	—	44.1%†	—	24.4%†	81.1%	62.2%
EIIL [30]	✗/✗	90.8%	64.5%	95.7%	41.7%	80.3%	64.7%
JTT [74]	✗/✗	—	62.5%†	—	40.6%†	81.3%	64.4%
Group DRO (with `ls`)	✗/✗	91.2%	**86.1%**	87.2%	**83.3%**	78.7%	**72.1%**

Group DRO with `ls` We consider the general setting where biases are not known during both training and validation. To obtain a robust model, we take the splits identified by `ls` (Section 4.4.2) and apply group DRO to minimize the *worst-split* loss for each class. Similarly for model selection, we apply the learned Splitter to split the validation data and measure the *worst-split* accuracy for each class (see Appendix C.2.1 for details). We report the average accuracy and worst-group accuracy (defined by the bias annotations) on the standard test set.

[2]For fair comparison, all methods share the same hyper-parameter search space and representation backbone (`resnet-50` for Waterbirds and CelebA, `bert-base-uncased` for MNLI). See Appendix C.2 for details.

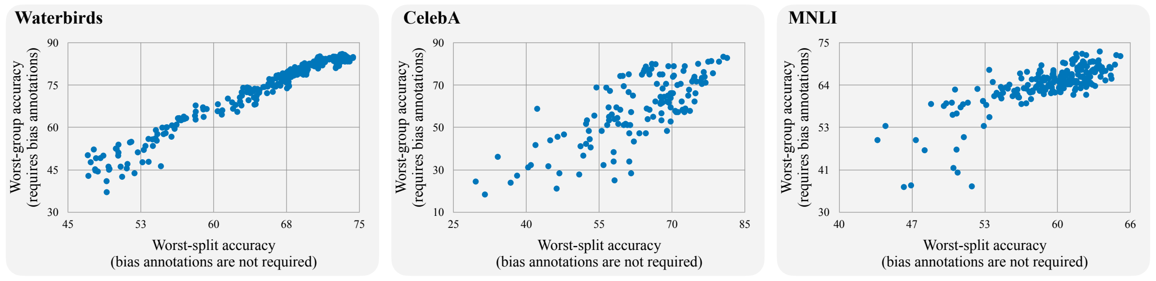


Figure 4-7: The spurious splits identified by `ls` provide a surrogate metric for model selection when biases are unknown. X-axis: worst-split accuracy defined by `ls`. Y-axis: worst-group accuracy defined by the oracle bias annotations.

Results Table 4.2 presents our results on de-biasing. We first see that when the bias annotations are available in the validation data, the missing baseline Group DRO (with supervised bias predictor) outperforms all previous de-biasing methods (4.8% on average). This result is not surprising given the fact that the bias attribute predictor is able to achieve 94.8% accuracy in Waterbirds (predicting the spurious background), 97.7% accuracy in CelebA (predicting the spurious gender attribute) and 99.9% in MNLI (predicting the prescence of negation words).

In reality where biases are unknown during both training and validation, previous de-biasing methods fail to improve over the ERM baseline. This confirms the findings of [74]. On the other hand, applying group DRO with splits identified by `ls` consistently achieves the best worst-group accuracy, outperforming previous methods by 23.4% on average. While we no longer have access to the worst-group validation accuracy for model selection (as it requires bias annotations), Figure 4-7 demonstrates that the worst-split performance on the validation set can be used as a surrogate.

4.5 Conclusion

We present Learning to Split (`ls`), an algorithm that learns to split the data so that predictors trained on the training split cannot generalize to the testing split. Our algorithm only requires access to the set of input-label pairs and is applicable to general datasets. Experiments across multiple modalities confirm that `ls` is able to identify challenging splits that correlates with human-identified biases.

Chapter 5

Conclusion

Deep neural networks have proved themselves as universal approximators [52]. However in practice, the datasets from which we train our neural networks are never guaranteed to be perfect. In the presence of dataset biases, neural networks fail.

In this thesis, I have explored algorithms that identify under-represented groups *without* explicit bias annotations. Such frameworks are attractive as human experts can understand the potential biases by simply looking at the characteristics of the identified groups. To deliver robustness, we enforce models to perform well across all groups. Our proposed algorithms are model-agnostic and can be applied to many applications.

- First, we consider the scenario where our dataset is accompanied with the environment information. In the medical text classification dataset ASK2ME, we demonstrated that by explicitly contrasting the data environments, our algorithm significantly improves the worst-case performance (74.1% vs. 54.8%) when evaluated against biases that are unknown during training.

- Second, we look at the situation where we don't have multiple environments in our target task of interest. In the face image classification dataset CelebA, we showed that by transferring the knowledge of biases across tasks (from predicting glass wearing to predicting hair color), our model generalizes robustly towards a different testing environment (84.9% vs. 61.0%).

- Finally, we showed how to identify biases when we are only given a set of input-label pairs, the standard supervised learning setting. Our algorithm learns to split the dataset so that predictors cannot generalize from the training split to the testing split. Empirical analysis show that the learned splits capture human-identified biases in natural language understanding, image classification and molecular property prediction. By learning to perform robustly across the splits, our model significantly outperforms state-of-the-art de-biasing techniques (72.1% vs. 64.4% on MNLI).

Together, these methods enable efficient and robust machine learning.

Future directions

We now provide some future directions of research that follow from this thesis:

- Multiple bias sources. Our algorithm `ls` learns to create a single partition of the dataset. However in many applications, biases can be high-dimensional. For example, we may have both gender bias and racial bias, and these two biases are independent to each other. Identifying and disentangling multiple sources of biases is an important but unexplored area.

- Generalization beyond supervised learning. In this thesis, we focus on the supervised learning setting where bias features are defined based on the given task. Generalizing the notion of biases to unsupervised or self-supervised learning [35, 26] is another interesting future direction.

  Current representation learning methods generate a single vector that summarizes all the features of a given input. When applied to a downstream task, those bias features will be exploited by the model. We would like to learn a mapping such that given a specific type of bias (such as gender), the mapping can remove the corresponding bias features from the input representation. Such generalization can significantly benefit low-resource tasks or tasks with long-tailed input distributions.

Appendices

Appendix A

Predict then Interpolate: A Simple Algorithm to Learn Stable Classifiers

A.1 A toy example

We would like to show that the optimal solution that minimize the worst-case risk across $E_2^{1\checkmark}$ and $E_2^{1\times}$ is to predict Y only using X_1. Consider any classifier $f(Y \mid X_1, X_2)$ and its marginal

$$f(Y \mid X_1) \propto f(X_1, X_2 = 0, Y) + f(X_1, X_2 = 1, Y).$$

For any input $(x_1, x_2) \in E_2^{1\checkmark}$, based on our construction, the distribution $P_2^{1\checkmark}(X_1 = x_1, X_2 = x_2, Y)$ only has mass on one label value $y \in \{0, 1\}$. Thus $P_2^{1\checkmark}(Y = y \mid X_1 = x_1, X_2 = x_2) = 1$. We can then write the log risk of the classifier $f(Y \mid X_1, X_2)$ as

$$-\log \frac{f(x_1, x_2, y)}{f(x_1, x_2, y) + f(x_1, x_2, 1-y)}.$$

The log risk of the marginal classifier $f(Y \mid X_1)$ is defined as

$$-\log \Big((f(x_1, x_2, y) + f(x_1, 1 - x_2, y)) \\ /(f(x_1, x_2, y) + f(x_1, 1 - x_2, y) + f(x_1, x_2, 1-y) + f(x_1, 1 - x_2, 1 - y)) \Big).$$

Now suppose $f(Y \mid X_1, X_2)$ achieves a lower risk than $f(Y \mid X_1)$. This implies

$$\begin{aligned}
&f(x_1, x_2, y)f(x_1, x_2, y) + f(x_1, x_2, 1-y)f(x_1, x_2, y) \\
&\qquad + f(x_1, x_2, y)f(x_1, 1-x_2, y) + f(x_1, x_2, 1-y)f(x_1, 1-x_2, y) \\
&< f(x_1, x_2, y)f(x_1, x_2, y) + f(x_1, x_2, y)f(x_1, x_2, 1-y) \\
&\qquad + f(x_1, x_2, y)f(x_1, 1-x_2, y) + f(x_1, x_2, y)f(x_1, 1-x_2, 1-y).
\end{aligned}$$

Note that the first three terms on both side cancel out. We have

$$f(x_1, x_2, 1-y)f(x_1, 1-x_2, y) < f(x_1, x_2, y)f(x_1, 1-x_2, 1-y).$$

Now let's consider an input $(x_1, 1-x_2) \in E_2^{1\times}$. Based on our construction of the partitions, we have $P_2^{1\checkmark}(x_1, x_2, y) = P_2^{1\times}(x_1, 1-x_2, y)$. The log risk of the marginal classifier on $P_2^{1\times}$ is still the same, but the log risk of the classifier $f(Y \mid X_1, X_2)$ now becomes

$$-\log \frac{f(x_1, 1-x_2, y)}{f(x_1, 1-x_2, y) + f(x_1, 1-x_2, 1-y)}.$$

We claim that the log risk of $f(Y \mid X_1, X_2)$ is higher than $f(Y \mid X_1)$ on $P_2^{1\times}$. Suppose for contradiction that the log risk of $f(Y \mid X_1, X_2)$ is lower, then we have

$$\begin{aligned}
&f(x_1, 1-x_2, y)f(x_1, x_2, y) + f(x_1, 1-x_2, 1-y)f(x_1, x_2, y) \\
&\qquad + f(x_1, 1-x_2, y)f(x_1, 1-x_2, y) + f(x_1, 1-x_2, 1-y)f(x_1, 1-x_2, y) \\
&< f(x_1, 1-x_2, y)f(x_1, x_2, y) + f(x_1, 1-x_2, y)f(x_1, 1-x_2, y) \\
&\qquad + f(x_1, 1-x_2, y)f(x_1, x_2, 1-y) + f(x_1, 1-x_2, y)f(x_1, 1-x_2, 1-y).
\end{aligned}$$

Canceling out the terms, we obtain

$$f(x_1, 1-x_2, 1-y)f(x_1, x_2, y) < f(x_1, 1-x_2, y)f(x_1, x_2, 1-y).$$

Contradiction!

Thus the marginal $f(Y \mid X_1)$ will always reach a better worst-group risk compare to the original classifier $f(Y \mid X_1, X_2)$. As a result, the optimal classifier $f(Y \mid X_1, X_2)$ should satisfy $f(Y \mid X_1, X_2) = f(Y \mid X_1)$, i.e., it will only use X_1 to predict Y.

A.2 Theoretical analysis

Proposition 1. *For a pair of environments E_i and E_j, assuming that the classifier f_i is able to learn the true conditional $P_i(Y \mid X_1, X_2)$, we can write the joint distribution P_j of E_j as the mixture of $P_j^{i\checkmark}$ and $P_j^{i\times}$:*

$$P_j(x_1, x_2, y) = \alpha_j^i P_j^{i\checkmark}(x_1, x_2, y) + (1 - \alpha_j^i) P_j^{i\times}(x_1, x_2, y),$$

where $\alpha_j^i = \sum_{x_1, x_2, y} P_j(x_1, x_2, y) \cdot P_i(y \mid x_1, x_2)$ and

$$\begin{aligned} P_j^{i\checkmark}(x_1, x_2, y) &\propto P_j(x_1, x_2, y) \cdot P_i(y \mid x_1, x_2), \\ P_j^{i\times}(x_1, x_2, y) &\propto P_j(x_1, x_2, y) \cdot P_i(1 - y \mid x_1, x_2). \end{aligned}$$

Proof. For ease of notation, let $i = 1$, $j = 2$. For an input (x_1, x_2), let's first consider the conditional probability $P_2^{1\times}(y \mid x_1, x_2)$ and $P_2^{1\checkmark}(y \mid x_1, x_2)$. Since the input is in E_2, the probability that it has label y is given by $P_2(y \mid x_1, x_2)$. Since f_1 matches $P_1(y \mid x_1, x_2)$, the likelihood that the prediction is wrong is given by $P_1(1 - y \mid x_1, x_2)$ and the likelihood that the prediction is correct is givn by $P_1(y \mid x_1, x_2)$. Thus, we have

$$\begin{aligned} P_2^{1\times}(y \mid x_1, x_2) &= \frac{P_1(1 - y \mid x_1, x_2) P_2(y \mid x_1, x_2)}{\sum_{y'} P_1(1 - y' \mid x_1, x_2) P_2(y' \mid x_1, x_2)}, \\ P_2^{1\checkmark}(y \mid x_1, x_2) &= \frac{P_1(y \mid x_1, x_2) P_2(y \mid x_1, x_2)}{\sum_{y'} P_1(y' \mid x_1, x_2) P_2(y' \mid x_1, x_2)}. \end{aligned}$$

Now let's think about the marginal of (x_1, x_2) if it is in the set of mistakes $E_2^{1\times}$. Again, since the input is in E_2, the probability that it exists is given by the marginal in E_2: $P_2(x_1, x_2)$. This input has two possibilities to be partitioned into $E_2^{1\times}$: 1) the label is y and f_1 predicts it as $1 - y$; 2) the label is $1 - y$ and f_1 predicts it as y.

Marginalizing over all (x_1, x_2), we have

$$\begin{aligned}
P_2^{1\times}(x_1, x_2) &= \frac{\frac{P_2(x_1,x_2)\sum_y P_1(1-y|x_1,x_2)P_2(y|x_1,x_2)}{\sum_y P_1(1-y|x_1,x_2)P_2(y|x_1,x_2)+P_1(y|x_1,x_2)P_2(y|x_1,x_2)}}{\sum_{x_1',x_2'} \frac{P_2(x_1',x_2')\sum_y P_1(1-y|x_1',x_2')P_2(y|x_1',x_2')}{\sum_y P_1(1-y|x_1',x_2')P_2(y|x_1',x_2')+P_1(y|x_1',x_2')P_2(y|x_1',x_2')}} \\
&= \frac{P_2(x_1, x_2)\sum_y P_1(1-y \mid x_1, x_2)P_2(y \mid x_1, x_2)}{\sum_{x_1',x_2'} P_2(x_1', x_2')\sum_y P_1(1-y \mid x_1', x_2')P_2(y \mid x_1', x_2')}
\end{aligned}$$

Similarly, we have

$$P_2^{1\checkmark}(x_1, x_2) = \frac{P_2(x_1, x_2)\sum_y P_1(y \mid x_1, x_2)P_2(y \mid x_1, x_2)}{\sum_{x_1',x_2'} P_2(x_1', x_2')\sum_y P_1(y \mid x_1', x_2')P_2(y \mid x_1', x_2')}$$

Combining these all together using the Bayes' theorem, we have

$$\begin{aligned}
P_2^{1\times}(x_1, x_2, y) &= \frac{P_1(1-y \mid x_1, x_2)P_2(y \mid x_1, x_2)P_2(x_1, x_2)}{\sum_{x_1',x_2'} P_2(x_1', x_2')\sum_{y'} P_1(1-y' \mid x_1', x_2')P_2(y' \mid x_1', x_2')}, \\
&= \frac{P_1(1-y \mid x_1, x_2)P_2(x_1, x_2, y)}{\sum_{x_1',x_2',y'} P_2(x_1', x_2', y')P_1(1-y' \mid x_1', x_2')}, \\
&\propto P_1(1-y \mid x_1, x_2)P_2(x_1, x_2, y), \\
P_2^{1\checkmark}(x_1, x_2, y) &= \frac{P_1(y \mid x_1, x_2)P_2(y \mid x_1, x_2)P_2(x_1, x_2)}{\sum_{x_1',x_2'} P_2(x_1', x_2')\sum_{y'} P_1(y' \mid x_1', x_2')P_2(y' \mid x_1', x_2')}, \\
&= \frac{P_1(y \mid x_1, x_2)P_2(x_1, x_2, y)}{\sum_{x_1',x_2',y'} P_2(x_1', x_2', y')P_1(y' \mid x_1', x_2')}, \\
&\propto P_1(y \mid x_1, x_2)P_2(x_1, x_2, y).
\end{aligned}$$

Finally, it is straightforward to show that for $\alpha_2^1 = \sum_{x_1,x_2,y} P_2(x_1, x_2, y)P_1(y \mid x_1, x_2)$, we have

$$\begin{aligned}
&\alpha_2^1 P_2^{1\checkmark}(x_1, x_2, y) + (1-\alpha_2^1)P_2^{1\times} \\
&= P_1(y \mid x_1, x_2)P_2(x_1, x_2, y) + P_1(1-y \mid x_1, x_2)P_2(x_1, x_2, y) = P_2(x_1, x_2, y).
\end{aligned}$$

$\square$

From now on, we assume that the marginal distribution of Y is uniform in all joint distributions, i.e., f_i performs equally well on different labels.

Theorem 2. *Suppose X_2 is independent of X_1 given Y. For any environment pair E_i and E_j, if $\sum_y P_i(x_2 \mid y) = \sum_y P_j(x_2 \mid y)$ for any x_2, then $\mathrm{Cov}(X_2, Y; P_i) > \mathrm{Cov}(X_2, Y; P_j)$ implies $\mathrm{Cov}(X_2, Y; P_j^{i\times}) < 0$ and $\mathrm{Cov}(X_2, Y; P_i^{j\times}) > 0$.*

Proof. By definition, we have

$$\begin{aligned}
&\mathrm{Cov}(X_2, Y; P_j^{i\times}) = \mathbb{E}[X_2 Y; P_j^{i\times}] - \mathbb{E}[X_2; P_j^{i\times}]\,\mathbb{E}[Y; P_j^{i\times}] \\
&= \sum_{x_1,x_2} x_2 P_j^{i\times}(x_1, x_2, 1) - \sum_{x_1,x_2,y} x_2 P_j^{i\times}(x_1, x_2, y) \sum_{x_1,x_2} P_j^{i\times}(x_1, x_2, 1) \\
&= \sum_{x_1,x_2,x_1',x_2',y'} x_2 P_j^{i\times}(x_1, x_2, 1) P_j^{i\times}(x_1', x_2', y') - \sum_{x_1,x_2,y,x_1',x_2'} x_2 P_j^{i\times}(x_1, x_2, y) P_j^{i\times}(x_1', x_2', 1)
\end{aligned}$$

Expanding the distributions of $P_j^{i\times}$, it suffices to show that

$$\begin{aligned}
&\sum_{x_1,x_2,x_1',x_2',y'} \Big(x_2 P_j(x_1, x_2, 1) P_i(0 \mid x_1, x_2) P_j(x_1', x_2', y') P_i(1 - y' \mid x_1', x_2') \Big) \\
&< \sum_{x_1,x_2,y,x_1',x_2'} \Big(x_2 P_j(x_1, x_2, y) P_i(1 - y \mid x_1, x_2) P_j(x_1', x_2', 1) P_i(0 \mid x_1', x_2') \Big)
\end{aligned}$$

Note that when $y = y' = 1$, two terms cancel out. Thus we need to show

$$\begin{aligned}
&\sum_{x_1,x_2,x_1',x_2'} \Big(x_2 P_j(x_1, x_2, 1) P_i(0 \mid x_1, x_2) P_j(x_1', x_2', 0) P_i(1 \mid x_1', x_2') \Big) \\
&< \sum_{x_1,x_2,x_1',x_2'} \Big(x_2 P_j(x_1, x_2, 0) P_i(1 \mid x_1, x_2) P_j(x_1', x_2', 1) P_i(0 \mid x_1', x_2') \Big)
\end{aligned}$$

Based on the assumption that the marginal distribution in $E_j^{i\times}$ is uniform, we have

$$\sum_{x_1',x_2'} P_j(x_1', x_2', 0) P_i(1 \mid x_1', x_2') \doteq \sum_{x_1',x_2'} P_j(x_1', x_2', 1) P_i(0 \mid x_1', x_2').$$

Thus we can simplify our goal as

$$\sum_{x_1,x_2} x_2 P_j(x_1, x_2, 1) P_i(0 \mid x_1, x_2) < \sum_{x_1,x_2} x_2 P_j(x_1, x_2, 0) P_i(1 \mid x_1, x_2)$$

Similarly, we can simplify the condition $\mathrm{Cov}(X_2, Y; P_i) > \mathrm{Cov}(X_2, Y; P_j)$ as

$$\sum_{x1,x_2} x_2(P_j(x_1, x_2, 1) - P_i(x_1, x_2, 1)) < \sum_{x1,x_2} x_2(P_j(x_1, x_2, 0) - P_i(x_1, x_2, 0))$$

Since x_2 is independent of x_1 given y, we have

$$\begin{aligned}&\sum_{x1,x_2} x_2(P_j(x_1, y = 1)P_j(x_2 \mid y = 1) - P_i(x_1, y = 1)P_i(x_2 \mid y = 1))\\ &< \sum_{x1,x_2} x_2(P_j(x_1, y = 0)P_j(x_2 \mid y = 0) - P_i(x_1, y = 0)P_i(x_2 \mid y = 0))\end{aligned}$$

Since x_1 is the stable feature and the label marginal is the same across environments, we have $P_j(x_1, y = 1) = P_i(x_1, y = 1)$ and $P_j(x_1, y = 0) = P_i(x_1, y = 0)$. This implies

$$\begin{aligned}&\sum_{x_1} P_j(x_1, y = 1) \sum_{x_2} x_2(P_j(x_2 \mid y = 1) - P_i(x_2 \mid y = 1))\\ &< \sum_{x_1} P_j(x_1, y = 0) \sum_{x_2} x_2(P_j(x_2 \mid y = 0) - P_i(x_2 \mid y = 0))\end{aligned}$$

Again, by uniform label marginals, we have

$$\sum_{x_2} x_2(P_j(x_2 \mid y = 1) - P_i(x_2 \mid y = 1)) < \sum_{x_2} x_2(P_j(x_2 \mid y = 0) - P_i(x_2 \mid y = 0).$$

For binary $x_2 \in \{0, 1\}$, this implies $P_j(x_2 = 1 \mid y = 1) + P_i(x_2 = 1 \mid y = 0) < P_j(x_2 = 1 \mid y = 0) + P_i(x_2 = 1 \mid y = 1)$. Since $P_j(x_2 \mid y = 1) + P_j(x_2 \mid y = 0) = P_i(x_2 \mid y = 1) + P_i(x_2 \mid y = 0)$, we have

$$P_j(x_2 \mid y = 1)P_j(x_2 \mid y = 0) < P_j(x_2 \mid y = 0)P_j(x_2 \mid y = 1). \tag{A.1}$$

We can expand our goal in the same way:

$$\begin{aligned}&\sum_{x_1,x_2} x_2P_j(x_1, x_2, 1)P_i(0 \mid x_1, x_2)\\ &= \sum_{x_1} P_j(x_1, y = 1)P_i(x_1, y = 0) \cdot \sum_{x_2} \frac{x_2P_j(x_2 \mid y = 1)P_i(x_2 \mid y = 0)}{P_i(x_1, x_2)}\end{aligned}$$

$$\begin{aligned}&\sum_{x_1,x_2} x_2 P_j(x_1,x_2,0)P_i(1\mid x_1,x_2)\\&=\sum_{x_1} P_j(x_1,y=0)P_i(x_1,y=1)\cdot\sum_{x_2}\frac{x_2P_j(x_2\mid y=0)P_i(x_2\mid y=1)}{P_i(x_1,x_2)},\end{aligned}$$

Plug in Eq (A.1) and we complete the proof. The other inequality follows by symmetry. □

Extension to multi-class classification: In Theorem 1, we focus on binary classification for simplicity. For multi-class classification, we can convert it into a binary problem by defining Y_c as a binary indicator of whether class c is present or absent. Our strong empirical performance on MNIST (10-class classification) also confirms that our results generalize to the multi-class setting.

Theorem 3. *For any environment pair E_i and E_j, $\mathrm{Cov}(X_2,Y;P_i)>\mathrm{Cov}(X_2,Y;P_j)$ implies*

$$\begin{aligned}\mathrm{Cov}(X_2,Y;P_j^{i\times})&<\frac{1-\alpha_j^i}{\alpha_i^i}\mathrm{Cov}(X_2,Y;P_i^{i\checkmark})-\frac{1-\alpha_j^i}{\alpha_j^i}\mathrm{Cov}(X_2,Y;P_j^{i\checkmark})\\\mathrm{Cov}(X_2,Y;P_i^{j\times})&>\frac{1-\alpha_i^j}{\alpha_j^j}\mathrm{Cov}(X_2,Y;P_j^{j\checkmark})-\frac{1-\alpha_i^j}{\alpha_i^j}\mathrm{Cov}(X_2,Y;P_i^{j\checkmark})\end{aligned}$$

where $P_i^{i\checkmark}$ is the distribution of the correct predictions when applying f_i on E_i.

Proof. From the proof in Theorem 1, we can write the condition $\mathrm{Cov}(X_2,Y;P_i)>\mathrm{Cov}(X_2,Y;P_j)$ as

$$\sum_{x1,x_2} x_2(P_j(x_1,x_2,1)-P_i(x_1,x_2,1))<\sum_{x1,x_2} x_2(P_j(x_1,x_2,0)-P_i(x_1,x_2,0))$$

Using $P_i(0\mid x_1,x_2)+P_i(1\mid x_1,x_2)=1$,

$$\begin{aligned}&\sum_{x1,x_2} x_2(P_j(x_1,x_2,1)-P_i(x_1,x_2,1))(P_i(0\mid x_1,x_2)+P_i(1\mid x_1,x_2))\\&<\sum_{x1,x_2} x_2(P_j(x_1,x_2,0)-P_i(x_1,x_2,0))(P_i(0\mid x_1,x_2)+P_i(1\mid x_1,x_2))\end{aligned}$$

Since $P_i(x_1, x_2, 1)P_i(0 \mid x_1, x_2)$ and $P_i(x_1, x_2, 0)P_i(1 \mid x_1, x_2)$ cancel out with each other. We have

$$\begin{aligned}
&\sum_{x1,x_2} x_2(P_j(x_1, x_2, 1)P_i(0 \mid x_1, x_2) - P_j(x_1, x_2, 0)P_i(1 \mid x_1, x_2)) \\
&< \sum_{x1,x_2} x_2(P_i(x_1, x_2, 1)P_i(1 \mid x_1, x_2) - P_i(x_1, x_2, 0)P_i(0 \mid x_1, x_2)) \\
&\quad - \sum_{x1,x_2} x_2(P_j(x_1, x_2, 1)P_i(1 \mid x_1, x_2) - P_j(x_1, x_2, 0)P_i(0 \mid x_1, x_2))
\end{aligned}$$

From the derivations in Theorem 1, we know that

$$\begin{aligned}
&\frac{1}{2(1-\alpha_j^i)}\mathrm{Cov}(X_2, Y; P_j^{i\times}) \\
&= \sum_{x1,x_2} x_2\Big(P_j(x_1, x_2, 1)P_i(0 \mid x_1, x_2) - P_j(x_1, x_2, 0)P_i(1 \mid x_1, x_2)\Big) \\
&\frac{1}{2\alpha_j^i}\mathrm{Cov}(X_2, Y; P_j^{i\checkmark}) \\
&= \sum_{x1,x_2} x_2\Big(P_j(x_1, x_2, 1)P_i(1 \mid x_1, x_2)P_j(x_1, x_2, 0)P_i(0 \mid x_1, x_2)\Big) \\
&\frac{1}{2\alpha_i^i}\mathrm{Cov}(X_2, Y; P_i^{i\checkmark}) \\
&= \sum_{x1,x_2} x_2\Big(P_i(x_1, x_2, 1)P_i(1 \mid x_1, x_2) - P_i(x_1, x_2, 0)P_i(0 \mid x_1, x_2)\Big).
\end{aligned}$$

Combining these, we have

$$\mathrm{Cov}(X_2, Y; P_j^{i\times}) < \frac{1-\alpha_j^i}{\alpha_i^i}\mathrm{Cov}(X_2, Y; P_i^{i\checkmark}) - \frac{1-\alpha_j^i}{\alpha_j^i}\mathrm{Cov}(X_2, Y; P_j^{i\checkmark})$$

Similarly, by using $P_j(0 \mid x_1, x_2) + P_j(1 \mid x_1, x_2) = 1$, we can get

$$\mathrm{Cov}(X_2, Y; P_i^{j\times}) > \frac{1-\alpha_i^j}{\alpha_j^j}\mathrm{Cov}(X_2, Y; P_j^{j\checkmark}) - \frac{1-\alpha_i^j}{\alpha_i^j}\mathrm{Cov}(X_2, Y; P_i^{j\checkmark})$$

□

A.3 Experimental setup

A.3.1 Datasets and models

MNIST

Data We use the official train-test split of MNIST. Training environments are constructed from training split, with 14995 examples per environment. Validation data and testing data is constructed based on the testing split, with 2497 examples each. Following [6], We convert each grey scale image into a $10 \times 28 \times 28$ tensor, where the first dimension corresponds to the spurious color feature.

Model: The input image is passed to a CNN with 2 convolution layers and 2 fully connected layers. We use the architecture from PyTorch's MNIST example[1].

Beer Review

Data We use the data processed by [69]. Reviews shorter than 10 tokens or longer than 300 tokens are filtered out. For each aspect, we sample training/validation/testing data randomly from the dataset and maintain the marginal distribution of the label to be uniform. Each training environment contains 4998 examples. The validation data contains 4998 examples and the testing data contains 5000 examples. The vocabulary sizes for the three aspects (look, aroma, palate) are: 10218, 10154 and 10086. The processed data will be publicly available.

Model We use a standard CNN text classifier [61]. Each input is first encoded by pre-trained FastText embeddings [84]. Then it is passed into a 1D convolution layer followed by max pooling and ReLU activation. The convolution layer uses filter size $3, 4, 5$. Finally we attach a linear layer with Softmax to predict the label.

[1]https://github.com/pytorch/examples/blob/master/mnist/main.py

CelebA

Data We use the official train/val/test split of CelebA [76]. The training environment {`female`} contains 94509 examples and the training environment {`male`} contains 68261 examples. The validation set has 19867 examples and the test set has 19962 examples.

Model We use the Pytorch torchvision implementation of the ResNet50 model, starting from pretrained weights. We re-initalize the final layer to predict the target attribute `hair color`.

ASK2ME

Data Since the original data doesn't have a standard train/val/test split, we randomly split the data and use 50% for training, 20% for validation, 30%for testing. There are 2227 examples in the training environment {`breast_cancer=0`}, 1394 examples in the training environment {`breast_cancer=1`}. The validation set contains 1448 examples and the test set contains 2173 examples. The vocabulary size is 16310. The processed data will be publicly available.

Model The model architecture is the same as the one for Beer review.

A.3.2 Implementation details

For all methods: We use batch size 50 and evaluate the validation performance every 100 batch. We apply early stopping once the validation performance hasn't improved in the past 20 evaluations. We use Adam [62] to optimize the parameters and tune the learning rate $\in \{10^{-3}, 10^{-4}, 10^{-5}\}$. For simplicity, we train all methods without data augmentation. Following [93], we apply strong regularizations to avoid over-fitting. Specifically, we tune the dropout rate $\in \{0.1, 0.3, 0.5\}$ for text classification datasets (Beer review and ASK2ME) and tune the weight decay parameters $\in \{10^{-0}, 10^{-1}, 10^{-2}, 10^{-3}\}$ for image datasets (MNIST and CelebA).

	TIME	Train	Val	Test
ERM	2 MIN 58 SEC	83.61	81.21	15.65
IRM	3 MIN 37 SEC	83.42	80.41	12.89
RGM	3 MIN 7 SEC	82.60	81.41	13.97
DRO	17 MIN 19 SEC	79.44	80.65	16.05
OURS	11 MIN 58 SEC	65.04	71.16	71.56
ORACLE	14 MIN 31 SEC	68.96	72.28	70.04

Table A.1: Running time and model performance on MNIST. Here the validation data is sampled from the training environments. Our algorithm requires training additional environment-specific classifiers. However, it converges faster than DRO in the third stage (50 epochs vs. 72 epochs) and generalizes much better.

DRO and **Ours** We directly optimize the $\min - \max$ objective. Specifically, at each step, we sample a batch of example from each group, and minimize the worst-group loss. We found the training process to be pretty stable when using the Adam optimizer. On CelebA, we are able to match the performance reported by [93].

IRM We implement the gradient penalty based on the official implementation of IRM[2]. The gradient penalty is applied to the last hidden layer of the network. We tune the weight of the penalty term $\in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ and the annealing iterations $\in \{10, 10^2, 10^3\}$.

RGM For the per-environment classifier in RGM, we use a MLP with one hidden layer. This MLP takes the last layer of the model as input and predicts the label. Similar to IRM, we tune the weight of the regret $\in \{10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ and the annealing iterations $\in \{10, 10^2, 10^3\}$.

A.3.3 Computing infrastructure and running time analysis

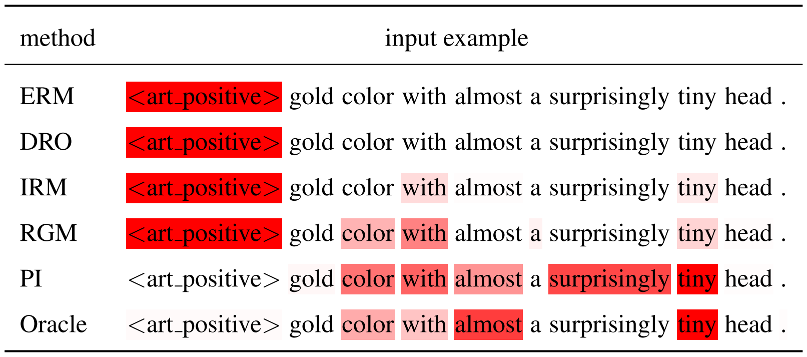
We have used the following graphics cards for our experiments: Tesla V100-32GB, GeForce RTX 2080 Ti and A100-40G.

We conducted our running time analysis on MNIST and ASK2ME using GeForce RTX 2080 Ti. Table A.1 and A.2 shows the results. We observe that due to the direct optimization of the $\min \max$ objective, the running time of DRO, PI and Oracle is

[2]https://github.com/facebookresearch/InvariantRiskMinimization

	TIME	Train	Val	Test
ERM	3 MIN 35 SEC	99.44	66.01	59.04
IRM	3 MIN 21 SEC	98.70	63.10	57.85
RGM	5 MIN 36 SEC	99.78	64.07	59.99
DRO	16 MIN 40 SEC	86.77	77.66	67.34
PI (Ours)	18 MIN	97.09	78.64	74.14

Table A.2: Running time and model performance on ASK2ME. Here the validation accuracy is computed based on the `breast_cancer` attribute. The test accuracy is the average worst-group accuracy across all 17 attributes. Our algorithm's running time is similar to DRO.



method	input example
ERM	<art_positive> gold color with almost a surprisingly tiny head .
DRO	<art_positive> gold color with almost a surprisingly tiny head .
IRM	<art_positive> gold color with almost a surprisingly tiny head .
RGM	<art_positive> gold color with almost a surprisingly tiny head .
PI	<art_positive> gold color with almost a surprisingly tiny head .
Oracle	<art_positive> gold color with almost a surprisingly tiny head .

Figure A-1: Visualizing word importance on Beer Look. Only `PI` and `Oracle` ignore the artificial token and correctly predict the input as negative. We will add more examples in the update.

roughly 4 times comparing to other methods (proportional to the number of groups). Also, while our model needs to train additional environment-specific classifiers (comparing to DRO), its running time is very similar to DRO across the two datasets. We believe by using the online learning algorithm proposed by [93], we can further reduce the running time of our algorithm.

A.4 Additional results

What features does pi look at? To understand what features different methods rely on, we plot the word importance on Beer Look in Figure A-1. For the given input example, we evaluate the prediction change as we mask out each input token. We observe that only PI and Oracle ignore the spurious feature and predict the label

	ERM		DRO		IRM		RGM		Ours	
Accuracy	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg
Adenocarcinoma	33.3	72.9	77.2	79.2	55.5	78.4	55.5	78.1	80.2	84.7
Polyp syndrom	44.4	74.6	77.2	78.7	55.5	76.3	66.6	78.7	69.2	81.2
Brain cancer	55.5	78.5	77.1	78.0	55.5	78.5	67.5	82.3	79.9	87.9
Breast cancer	66.4	80.4	75.0	78.8	66.8	80.5	64.3	79.8	80.3	83.1
Colorectal cancer	66.5	80.5	69.3	77.9	64.9	81.2	66.9	80.3	76.2	81.7
Endometrial cancer	66.9	80.6	76.1	80.2	66.0	82.6	66.9	81.7	80.3	83.2
Gastric cancer	62.9	79.9	76.9	81.6	62.9	80.0	59.2	78.8	79.4	85.9
Hepatobiliary cancer	44.4	73.0	60.0	73.8	55.5	77.1	55.5	76.2	60.0	78.9
Kidney cancer	16.6	66.6	50.0	68.7	33.3	73.0	33.3	71.3	50.0	74.7
Lung cancer	44.4	74.7	62.5	74.5	38.8	74.2	50.0	74.7	70.3	78.8
Melanoma	66.6	80.5	66.6	78.8	66.6	83.3	66.6	79.6	80.0	86.6
Neoplasia	50.0	75.9	33.3	69.1	33.3	71.9	50.0	75.1	70.0	80.0
Ovarian cancer	65.3	80.1	77.2	79.3	66.8	80.6	66.3	79.5	73.4	82.7
Pancreatic cancer	67.1	80.9	75.8	78.7	63.6	79.6	63.6	79.6	80.0	84.3
Prostate cancer	63.9	85.7	51.0	77.4	64.2	85.2	65.5	83.9	78.9	86.7
Rectal cancer	66.6	78.7	64.1	80.3	66.6	78.8	67.5	80.8	71.7	84.5
Thyroid cancer	50.0	77.1	75.0	83.0	66.8	84.0	67.7	82.5	80.2	87.8
Average	54.8	77.7	67.3	77.5	57.8	79.1	60.8	79.0	74.1	83.1

Table A.3: Full results. Worst-group and average-group accuracy across 17 attributes on ASK2ME.

correctly. Comparing to ERM, IRM and RGM focus more on the causal feature such as ‘tiny’. However, they still heavily rely on the spurious feature.

	ERM		DRO		IRM		RGM		Ours	
	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg	Worst	Avg
5_o_Clock_Shadow	53.3	81.5	90.0	92.0	80.0	85.9	66.6	87.2	83.3	89.6
Arched_Eyebrows	72.1	87.1	90.4	92.6	84.4	87.8	88.9	92.6	90.5	92.3
Attractive	67.2	85.8	90.7	92.2	82.5	87.2	86.6	91.9	89.9	91.8
Bags_Under_Eyes	72.4	86.1	90.5	92.0	81.3	86.8	88.5	92.1	89.1	92.1
Bald	75.9	91.2	91.7	93.0	71.3	82.2	91.5	94.8	91.6	93.4
Bangs	73.8	87.8	90.8	92.9	81.7	87.3	88.0	92.2	90.2	92.3
Big_Lips	73.4	87.1	90.5	92.5	84.1	87.8	89.5	92.6	90.5	92.1
Big_Nose	71.4	86.0	91.5	92.9	84.9	88.4	91.2	93.7	91.3	92.8
Black_Hair	75.9	90.9	89.6	93.7	78.6	89.1	90.6	94.0	88.1	93.3
Blurry	51.2	81.0	86.5	90.1	79.3	85.7	79.0	89.7	85.6	89.3
Brown_Hair	43.6	79.1	64.3	85.7	78.1	83.3	72.4	87.3	59.7	83.8
Bushy_Eyebrows	72.7	86.5	72.7	88.8	81.8	87.5	81.8	91.2	81.8	90.7
Chubby	9.52	70.6	61.9	84.6	76.1	82.9	47.6	82.3	71.4	86.5
Double_Chin	50.0	80.7	90.6	91.7	78.5	86.3	91.5	92.7	90.2	92.4
Eyeglasses	58.0	82.8	90.3	92.0	80.4	85.7	77.4	89.3	88.7	91.1
Goatee	0.0	68.2	0.0	70.0	84.8	90.8	91.5	95.6	91.6	94.5
Gray_Hair	60.7	82.5	69.0	87.7	42.6	76.2	85.7	89.5	68.2	88.1
Heavy_Makeup	66.0	85.6	89.6	92.2	84.1	87.2	84.4	91.4	90.0	91.8
High_Cheekbones	73.3	86.6	90.7	92.2	84.4	87.1	89.0	92.2	90.3	91.7
Gender.	46.6	80.1	85.5	90.8	74.4	83.9	70.0	87.7	90.5	91.5
Mouth_Slightly_Open	74.2	87.0	91.2	92.3	84.5	87.4	91.0	92.5	91.7	91.8
Mustache	50.0	80.8	91.7	95.3	50.0	78.5	91.5	95.9	91.6	94.9
Narrow_Eyes	69.2	85.5	90.0	91.8	82.9	87.0	88.4	91.8	91.6	91.9
No_Beard	39.3	78.1	84.8	90.9	72.7	83.8	57.5	85.0	84.8	90.4
Oval_Face	75.1	87.2	90.7	92.4	84.2	87.7	91.2	92.7	90.3	91.9
Pale_Skin	75.4	87.9	90.3	91.5	81.6	85.9	91.3	92.4	89.5	92.0
Pointy_Nose	73.3	87.1	91.1	92.4	84.8	87.6	89.2	92.5	91.0	92.0
Receding_Hairline	66.6	84.7	90.9	91.8	80.5	84.3	83.3	91.1	87.9	90.9
Rosy_Cheeks	74.9	88.1	91.4	93.3	84.8	88.5	90.5	93.0	91.4	92.7
Sideburns	38.4	77.8	84.6	90.6	76.9	84.1	76.9	89.7	91.3	93.7
Smiling	75.9	86.8	91.5	92.3	84.1	87.2	91.1	92.4	91.5	91.8
Straight_Hair	74.0	86.6	90.2	92.0	84.3	87.4	88.3	92.3	91.5	91.9
Wavy_Hair	74.2	86.8	91.4	92.4	84.1	87.4	88.8	92.2	91.6	91.8
Wearing_Earrings	75.3	86.7	91.6	92.6	84.7	87.7	90.8	92.5	91.5	92.1
Wearing_Hat	7.6	70.3	46.1	82.2	46.1	77.3	61.5	86.2	53.8	84.5
Wearing_Lipstick	59.3	83.6	89.4	91.8	82.5	86.0	79.3	90.1	90.3	91.5
Wearing_Necklace	74.5	87.2	91.0	92.4	82.2	87.4	89.5	92.1	90.7	92.2
Wearing_Necktie	25.0	74.5	90.0	91.5	80.0	84.3	35.0	79.3	91.4	92.5
Young	71.6	86.0	89.1	91.6	76.2	85.9	90.1	92.0	87.2	91.5
Average	60.0	83.6	84.2	90.8	78.5	85.7	82.5	90.9	87.0	91.4

Table A.4: Full results. Worst-group and average-group accuracy for hair color prediction on CelebA.

Appendix B

Learning Stable Classifiers by Transferring Unstable Features

B.1 Theoretical analysis

B.1.1 Partitions reveal the unstable correlation

We start by reviewing the results in the previous chapter which shows that the generated partitions reveal the unstable correlation. We consider binary classification tasks where $\mathcal{Y} \in \{0,1\}$. For a given input x, we use $\mathcal{C}(x)$ to represent its stable (causal) feature and $\mathcal{Z}(x)$ to represent its unstable feature. In order to ease the notation, if no confusion arises, we omit the dependency on x. We use lowercase letters c, z, y to denote the specific values of $\mathcal{C}, \mathcal{Z}, \mathcal{Y}$.

Proposition 1. *For a pair of environments E_i and E_j, assuming that the classifier f_i is able to learn the true conditional $P_i(\mathcal{Y} \mid C, \mathcal{Z})$, we can write the joint distribution P_j of E_j as the mixture of $P_j^{i\checkmark}$ and $P_j^{i\times}$:*

$$P_j(c,z,y) = \alpha_j^i P_j^{i\checkmark}(c,z,y) + (1-\alpha_j^i) P_j^{i\times}(c,z,y),$$

where $\alpha_j^i = \sum_{c,z,y} P_j(c,z,y) \cdot P_i(y \mid c,z)$ *and*

$$P_j^{i\checkmark}(c,z,y) \propto P_j(c,z,y) \cdot P_i(y \mid c,z),$$
$$P_j^{i\times}(x,z,y) \propto P_j(c,z,y) \cdot P_i(1-y \mid c,z).$$

Proposition 1 tells us that if f_i is powerful enough to capture the true conditional in E_i, partitioning the environment E_j is equivalent to scaling its joint distribution based on the conditional on E_i.

Now suppose that the marginal distribution of $\mathcal{Y}$ is uniform in all joint distributions, i.e., f_i performs equally well on different labels. [9] shows that the unstable correlations will have different signs in the subset of correct predictions and in the subset of incorrect predictions.

Proposition 2. *Suppose $\mathcal{Z}$ is independent of $\mathcal{C}$ given $\mathcal{Y}$. For any environment pair E_i and E_j, if $\sum_y P_i(z \mid y) = \sum_y P_j(z \mid y)$ for any z, then $\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_i) > \mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j)$ implies*

$$\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j^{i\times}) < 0, \quad \textit{and} \quad \mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_i^{j\times}) > 0.$$

Proof. See [9]. □

Proposition 2 implies that no matter whether the spurious correlation is positive or negative, by interpolating $P_j^{i\checkmark}, P_j^{i\times}, P_i^{j\checkmark}, P_i^{j\times}$, we can obtain an *oracle* distribution where the spurious correlation between $\mathcal{Z}$ and $\mathcal{Y}$ vanishes. Since the oracle interpolation coefficients are not available in practice, [9] propose to optimize the worst-case risk across all interpolations of the partitions.

B.1.2 Partitions reveal the unstable feature

Proposition 2 shows that the partitions $E_j^{i\checkmark}, E_j^{i\times}, E_i^{j\checkmark}, E_i^{j\times}$ are informative of the biases. However these partitions are not transferable as they are coupled with task-specific information, i.e., the label $\mathcal{Y}$. To untangle this dependency, we look at different label values and obtain the following result.

Corollary 1. *Under the same assumption as Proposition 2, if* $\mathrm{Cov}(\mathcal{Z}, \mathcal{Y}; P_i) > \mathrm{Cov}(\mathcal{Z}, \mathcal{Y}; P_j) > 0$ *and* $\mathcal{Z}$ *follows a uniform distribution within each partition, then*

$$\begin{aligned}\sum_z zP_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=1) &> \sum_z zP_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=1),\\ \sum_z zP_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=0) &< \sum_z zP_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=0).\end{aligned}$$

Proof. By definition of the covariance, we have

$$\mathrm{Cov}(\mathcal{Z},\mathcal{Y}) = \sum_{z,y} zyP(\mathcal{Z}=z,\mathcal{Y}=y) - \left(\sum_z zP(\mathcal{Z}=z)\right)\left(\sum_y yP(\mathcal{Y}=y)\right)$$

Since we assume the marginal distribution of the label is uniform, we have $\sum_y yP(\mathcal{Y}=y) = 0.5$. Then we have

$$\mathrm{Cov}(\mathcal{Z},\mathcal{Y}) = \sum_z zP(\mathcal{Z}=z,\mathcal{Y}=1) - 0.5\sum_z zP(\mathcal{Z}=z).$$

Using $P(\mathcal{Z}=z) = P(\mathcal{Z}=z,\mathcal{Y}=0) + P(\mathcal{Z}=z,\mathcal{Y}=1)$, we obtain

$$\mathrm{Cov}(\mathcal{Z},\mathcal{Y}) = 0.5\sum_z zP(\mathcal{Z}=z,\mathcal{Y}=1) - 0.5\sum_z zP(\mathcal{Z}=z,\mathcal{Y}=0). \tag{B.1}$$

From Proposition 2, we have $\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j^{i\times}) < 0$. Note that this implies $\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j^{i\checkmark}) > 0$ since $\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j) > 0$ and $P_j = \alpha_j^i P_j^{i\checkmark} + (1-\alpha_j^i)P_j^{i\times}$. Combining with Eq (B.1), we have

$$\begin{aligned}\sum_z zP_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=1) &< \sum_z zP_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=0),\\ \sum_z zP_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=1) &> \sum_z zP_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=0).\end{aligned} \tag{B.2}$$

Since we assume the marginal distribution of the unstable feature $\mathcal{Z}$ is uniform, we have

$$\sum_z z P_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=1)+\sum_z z P_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=0)=\sum_z z P_j^{i\times}(\mathcal{Z}=z)=0.5,$$
$$\sum_z z P_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=1)+\sum_z z P_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=0)=\sum_z z P_j^{i\checkmark}(\mathcal{Z}=z)=0.5. \quad \text{(B.3)}$$

Plugging Eq (B.3) into Eq (B.3), we have

$$\sum_z z P_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=1)<0.25<\sum_z z P_j^{i\times}(\mathcal{Z}=z,\mathcal{Y}=0),$$
$$\sum_z z P_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=1)>0.25>\sum_z z P_j^{i\checkmark}(\mathcal{Z}=z,\mathcal{Y}=0).$$

Combining the two inequalities finishes the proof. □

Corollary 1 shows that if we look at examples within the same label value, then expectation of the unstable feature $\mathcal{Z}$ within the set of correct predictions will diverge from the one within the set of incorrect predictions. In order to learn a metric space that corresponds to the values of $\mathcal{Z}$, we sample different batches from the partitions and prove the following theorem.

Theorem 3. *(Full version) Suppose $\mathcal{Z}$ is independent of $\mathcal{C}$ given $\mathcal{Y}$. We assume that $\mathcal{Y}$ and $\mathcal{Z}$ both follow a uniform distribution within each partition.*

Consider examples in E_j with label value y. Let $X_1^{\checkmark}, X_2^{\checkmark}$ denote two batches of examples that f_i predicted correctly, and let $X_3^{\times}$ denote a batch of incorrect predictions. If $\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_i)>\mathrm{Cov}(\mathcal{Z},\mathcal{Y};P_j)>0$, *we have*

$$\|\overline{\mathcal{Z}}(X_1^{\checkmark})-\overline{\mathcal{Z}}(X_2^{\checkmark})\|_2<\|\overline{\mathcal{Z}}(X_1^{\checkmark})-\overline{\mathcal{Z}}(X_3^{\times})\|_2$$

almost surely for large enough batch size.

Proof. Without loss of generality, we consider $y=0$. Let n denote the batch size of $X_1^{\checkmark}$, $X_2^{\checkmark}$ and $X_3^{\checkmark}$. By the law of large numbers, we have

$$\overline{\mathcal{Z}}(X_1^{\checkmark}),\overline{\mathcal{Z}}(X_2^{\checkmark})\xrightarrow{\text{a.s.}}\mathbb{E}_{P_j^{i\checkmark}(\mathcal{Z}|\mathcal{Y})}\left[\mathcal{Z}\mid\mathcal{Y}=0\right]\quad\text{and}\quad\overline{\mathcal{Z}}(X_3^{\times})\xrightarrow{\text{a.s.}}\mathbb{E}_{P_j^{i\times}(\mathcal{Z}|\mathcal{Y})}\left[\mathcal{Z}\mid\mathcal{Y}=0\right],$$

as $n \to \infty$. Note that Corollary 1 tells us

$$\mathbb{E}_{P_j^{i\times}(\mathcal{Z}|\mathcal{Y})}\left[\mathcal{Z} \mid \mathcal{Y}=0\right] < \mathbb{E}_{P_j^{i\checkmark}(\mathcal{Z}|\mathcal{Y})}\left[\mathcal{Z} \mid \mathcal{Y}=0\right].$$

Thus we have

$$\|\overline{\mathcal{Z}}(X_1^{\checkmark}) - \overline{\mathcal{Z}}(X_2^{\checkmark})\|_2 < \|\overline{\mathcal{Z}}(X_1^{\checkmark}) - \overline{\mathcal{Z}}(X_3^{\times})\|_2$$

almost surely as $n \to \infty$. □

We note that while we focus our theoretical analysis on binary tasks, empirically, our method is able to correctly identify the hidden bias for multi-dimensional unstable features and multi-dimensional label values.

B.2 Experimental setup

B.2.1 Datasets and models

MNIST

Data We extend [6]'s approach for generating spurious correlations and define two *multi-class* classification tasks: EVEN (5-way classification among digits `0,2,4,6,8`) and ODD (5-way classification among digits `1,3,5,7,9`). For each image, we first map its numeric digit value y^{digit} into its class id within the task: $y^{\text{causal}} = \lfloor y^{\text{digit}}/2 \rfloor$. This class id serves as the causal feature for the given task. We then sample the observed label y, which equals to y^{causal} with probability 0.75 and a uniformly random other label value with the remaining probability. With this noisy label, we now sample the spurious color feature: the color value equals y with η probability and a uniformly other value with the remaining probability. We note that since there are five different digits for each task, we have five different colors. Finally, we color the image according to the generated color value. For the training environments, we set η to 0.8 in E_1^{train} and 0.9 in E_2^{train}. We set $\eta = 0.1$ in the testing environment E^{test}.

We use the official train-test split of MNIST. Training environments are constructed from training split, with 7370 examples per environment for EVEN and 7625 examples per environment for ODD. Validation data and testing data is constructed based on the testing split. For EVEN, both validation data and testing data have 1230 examples. For ODD, the number is 1267. Following [6], We convert each grey scale image into a $5 \times 28 \times 28$ tensor, where the first dimension corresponds to the spurious color feature.

Representation backbone We follow the architecture from PyTorch's MNIST example[1]. Specifically, each input image is passed to a CNN with 2 convolution layers followed by 2 fully connected layers.

License The dataset is freely available at `http://yann.lecun.com/exdb/mnist/`.

Beer Review

Data We consider the transfer among three *binary* aspect-level sentiment classification tasks: LOOK, AROMA and PALATE [69]. For each review, we follow [9] and append a pseudo token (`art_pos` or `art_neg`) based on the the sentiment of the given aspect (`pos` or `neg`). The probability that this pseudo token agrees with the sentiment label is 0.8 in E_1^{train} and 0.9 in E_2^{train}. In the testing environment, this probability reduces to 0.1. Unlike MNIST, there is no label noise added to the data.

We use the script created by [9] to generate spurious features for each aspect. Specifically, for each aspect, we randomly sample training/validation/testing data from the dataset. Since our focus in this paper is to measure whether the algorithm is able to remove biases (rather than label imbalance), we maintain the marginal distribution of the label to be uniform. Each training environment contains 4998 examples. The validation data contains 4998 examples and the testing data contains 5000 examples. The vocabulary sizes for the three aspects (look, aroma, palate) are: 10218, 10154 and 10086.

Representation backbone We use a 1D CNN [61], with filter size $3, 4, 5$, to obtain the feature representation. Specifically, each input is first encoded by pre-trained

[1]https://github.com/pytorch/examples/blob/master/mnist/main.py

FastText embeddings [84]. Then it is passed into a convolution layer followed by max pooling and ReLU activation.

License This dataset was originally downloaded from `https://snap.stanford.edu/data/web-BeerAdvocate.html`. As per request from BeerAdvocate the data is no longer publicly available.

ASK2ME

Data ASK2ME [12] is a text classification dataset where the inputs are paper abstracts from PubMed. We study the transfer between two *binary* classification tasks: PENETRANCE (identifying whether the abstract is informative about the risk of cancer for gene mutation carriers) and INCIDENCE (identifying whether the abstract is informative about proportion of gene mutation carriers in the general population). By definition, both tasks are causally-independent of the diseases that have been studied in the abstract. However, due to the bias in the data collection process, [33] found that the performance varies (by 12%) when we evaluate based on different cancers. To assess whether we can remove such bias, we define two training environments for each task based on the correlations between the task label and the `breast_cancer` attribute (indicating the presence of breast cancer in the abstract). Script for generating the environments is available in the supplemental materials. Note that the model doesn't have access to the `breast_cancer` attribute during training.

Following [93], we evaluate the performance on a balanced test environment where there is no spurious correlation between `breast_cancer` and the task label. This helps us understand the overall generalization performance across different input distributions.

We randomly split the data and use 50% for PENETRANCE and 50% for INCIDENCE. For PENETRANCE, there are 948 examples in E_1^{train} and E^{val}, 816 examples in E_2^{train} and 268 examples in E^{test}. For INCIDENCE, there are 879 examples in E_1^{train} and E^{val}, 773 examples in E_2^{train} and 548 examples in E^{test}. The processed data will be publicly available.

Representationi backbone The model architecture is the same as the one for Beer review.

License MIT License.

Waterbird

Data Waterbird is an image classification dataset where each image is labeled based on its bird class [114] and the background attribute (`water` vs. `land`). Following [93], we group different bird classes together and consider two *binary* classification tasks: SEABIRD (classifying 36 seabirds against 36 landbirds) and WATERFOWL (classifying 9 waterfowl against 9 *different* landbirds). Similar to ASK2ME, we define two training environments for each task based on the correlations between the task label and the `background` attribute. Script for generating the environments is available in the supplemental materials. At test time, we measure the generalization performance on a balanced test environment.

Following [76], we group different classes of birds together to form binary classification tasks.

In WATERFOWL, the task is to identify 9 different waterfowls (Red breasted Merganser, Pigeon Guillemot, Horned Grebe, Eared Grebe, Mallard, Western Grebe, Gadwall, Hooded Merganser, Pied billed Grebe) against 9 different landbirds (Mourning Warbler, Whip poor Will, Brewer Blackbird, Tennessee Warbler, Winter Wren, Loggerhead Shrike, Blue winged Warbler, White crowned Sparrow, Yellow bellied Flycatche). The training environment E_1^{train} contains 298 examples and the training environment E_2^{train} contains 250 examples. The validation set has 300 examples and the test set has 216 examples.

In SEABIRD, the task is to identify *36 different seabirds* (Heermann Gull, Red legged Kittiwake, Rhinoceros Auklet, White Pelican, Parakeet Auklet, Western Gull, Slaty backed Gull, Frigatebird, Western Meadowlark, Long tailed Jaeger, Red faced Cormorant, Pelagic Cormorant, Brandt Cormorant, Black footed Albatross, Western Wood Pewee, Forsters Tern, Glaucous winged Gull, Pomarine Jaeger, Sooty Albatross, Artic Tern, California Gull, Horned Puffin, Crested Auklet, Elegant Tern, Common Tern, Least Auklet, Northern Fulmar, Ring billed Gull, Ivory Gull, Laysan Albatross, Least Tern, Black Tern, Caspian Tern, Brown Pelican, Herring Gull, Eastern Towhee) against *36 different landbirds* (Prairie Warbler, Ringed Kingfisher, Warbling Vireo, American Goldfinch, Black and white Warbler, Marsh Wren, Acadian Flycatcher, Philadelphia Vireo, Henslow Sparrow, Scissor tailed Flycatcher, Evening Grosbeak, Green Violetear, Indigo Bunting, Gray Catbird, House Sparrow, Black capped Vireo, Yellow Warbler, Common Raven, Pine Warbler, Vesper Sparrow, Pileated Woodpecker, Bohemian Waxwing, Bronzed Cowbird, American Three toed Woodpecker, Northern Waterthrush, White breasted Kingfisher, Olive sided Flycatcher, Song Sparrow, Le Conte Sparrow, Geococcyx, Blue Grosbeak, Red cockaded Woodpecker, Green tailed Towhee, Sayornis, Field Sparrow, Worm eating Warbler). The training environment E_1^{train} contains 1176 examples and the training environment E_2^{train} contains 998 examples. The validation set has 1179 examples and the test set has 844 examples.

Representation backbone We use the Pytorch torchvision implementation of the ResNet50 model, starting from pretrained weights. We re-initalize the final layer to predict the label.

License This dataset is publicly available at `https://nlp.stanford.edu/data/dro/waterbird_complete95_forest2water2.tar.gz`

CelebA

Data CelebA [76] is an image classification dataset where each image is annotated with 40 binary attributes. We consider `Eyeglasses` as the source task and `BlondHair` as the target task. We split the official train / val / test set into two parts (uniformly at random) for each task. We use the attribute `Young` to create two environments: $E_1 = \{\texttt{Young} = 0\}$, $E_2 = \{\texttt{Young} = 1\}$. For the target task, the model only has access to E_1 during training and validation. Table B.1 summarizes the data statistics.

Table B.1: Data statistics of CelebA. The model has access to both E_1 and E_2 on the source task. For the target task, only E_1 is available during training and validation.

	source task: `Eyeglasses`		target task: `BlondHair`	
	E_1 : {`Young=0`}	E_2 : {`Young=1`}	E_1 : {`Young=0`}	E_2 : {`Young=1`}
Train	17955	63430	17973	63412
Val	2494	7442	2453	7480
Test	2452	7597	2444	7537

License The CelebA dataset is available for non-commercial research purposes only. It is publicly available at `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`.

Representation backbone We use the Pytorch torchvision implementation of the ResNet50 model, starting from pretrained weights. We re-initalize the final layer to predict the label.

B.2.2 Implementation details

For all methods: We use batch size 50 and evaluate the validation performance every 100 batch. We apply early stopping once the validation performance hasn't improved in the past 20 evaluations. We use Adam [62] to optimize the parameters and tune the learning rate $\in \{10^{-3}, 10^{-4}\}$. For simplicity, we train all methods without data augmentation. Following [93], we apply strong regularizations to avoid over-fitting. Specifically, we tune the dropout rate $\in \{0.1, 0.3, 0.5\}$ for text classification datasets (Beer review and ASK2ME) and tune the weight decay parameters $\in \{10^{-1}, 10^{-2}, 10^{-3}\}$ for image datasets (MNIST, Waterbird and CelebA).

DANN, C-DANN For the domain adversarial network, we use a MLP with 2 hidden ReLU layer with 300 neurons for each layer. The representation backbone is updated via a gradient reversal layer. We tune the weight of the adversarial loss $\in \{0.01, 0.1, 1\}$.

MMD We match the mean and covariance of the features across the two source environments. We use the implementation from `https://github.com/facebookresearch/DomainBed/blob/main/domainbed/algorithms.py`. We tune the weight of the MMD loss $\in \{0.01, 0.1, 1\}$.

MULTITASK For the source task, we first partition the source data into subsets with opposite spurious correlations [9]. During multi-task training, we minimize the worst-case risk over all these subsets for the source task and minimize the average empirical risk for the target task. MULTITASK is more flexible than REUSE since we tune feature extractor to fit the target data. Compared to FINETUNE, MULTITASK is more constrained as the source model prevents over-utilization of unstable features during joint training.

Ours We fix $\delta = 0.3$ in all our experiments. Based on our preliminary experiments (Figure B-1), we fix the number of clusters to be 2 for all our experiments in Table 3.2 and Table 3.3. For the target classifier, we directly optimize the $\min - \max$ objective. Specifically, at each step, we sample a batch of example from each group, and minimize the worst-group loss. We found the training process to be pretty stable when using the Adam optimizer.

Validation criteria For ERM, REUSE, FINETUNE and MULTITASK, since we don't have any additional information (such as environments) for the target data, we apply early stopping and hyper-parameter selection based on the average accuracy on the validation data.

For TOFU, since we have already learned an unstable feature representation $f_{\mathcal{Z}}$ on the source task, we can also use it to cluster the validation data into groups where the unstable features within each group are different. We measure the worst-group accuracy and use it as our validation criteria.

For ORACLE, as we assume access to the oracle unstable features for the target data, we can use them to define groups on the validation data as well. We use the worst-group accuracy as our validation criteria.

We also note that when we transfer from LOOK to AROMA in Table 3.2, both TOFU and ORACLE are able to achieve 75 accuracy on E^{test}. This number is higher than the performance of training on AROMA with two data environments (68 accuracy in Table 3.2). This result makes sense since in the latter case, we only have in-domain validation set and we use the average accuracy as our hyper-parameter selection metric. However, in both TOFU and ORACLE, we create (either automatically or manually) groups over the validation data and measure the worst-group performance. This ensures that the chosen model will not over-fit to the unstable correlations.

Computational resources: We use our internal clusters (24 NVIDIA RTX A6000 and 16 Tesla V100-PCIE-32GB) for the experiments. It took around a week to generate all the results in Table 3.2 and Table 3.3.

B.3 Additional analysis

Why do the baselines behave so differently across different datasets? As [13] pointed out, the transferability of the low-level features is very different in image classification and in text classification. For example, the keywords for identifying the sentiment of LOOK are very different from the ones for PALATE. Thus, fine-tuning the feature extractor is crucial. This explains why REUSE underperforms other baselines on text data. Conversely, in image classification, the low-level patterns (such as edges) are more transferable across tasks. Directly reusing the feature extractor helps improve model stability against spurious correlations. Finally, we note that since TOFU transfers the unstable features instead of the task-specific causal features, it performs robustly across all the settings.

How many clusters to generate? We study the effect of the number of clusters on ASK2ME. Figure B-1 shows that while generating more clusters in the unstable feature space $f_{\mathcal{Z}}$ reduces the variance, it doesn't improve the performance by much. This is not very surprising as the training data is primarily biased by a single `breast_cancer` attribute. We expect that having more clusters will be beneficial for tasks with more sophisticated underlying biases.

How do we select the hyper-parameter for TOFU? We cluster the validation data based on the learned unstable feature representation $f_{\mathcal{Z}}$ and use the worst-group loss as our early stopping and hyper-parameter selection criteria. Figure B-2 shows

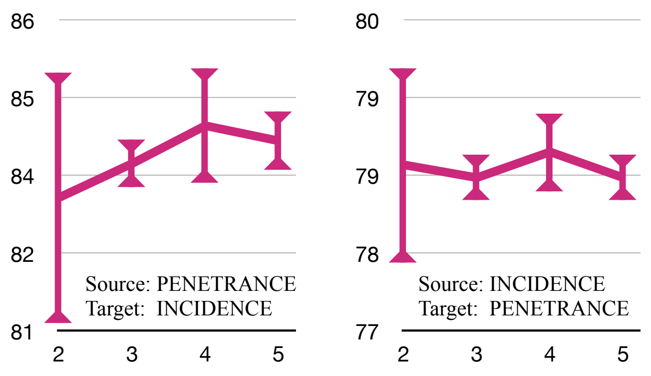


Figure B-1: Accuracy of TOFU on ASK2ME as we vary the number of clusters n_c generated for each label value. Empirically, we see that while having more clusters doesn't improve the performance, it helps reduce the variance.

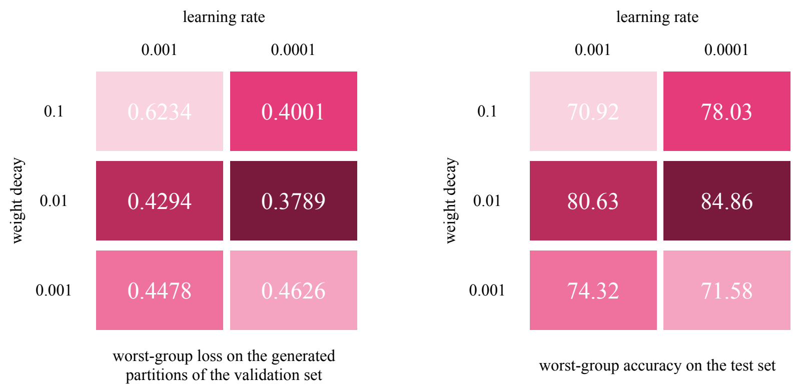


Figure B-2: Hyper-parameter selection for TOFU on CelebA (averaged across 5 runs). We use our learned unstable feature representation $f_{\mathcal{Z}}$ to partition the validation set and use the worst-group validation loss as our hyper-parameter selection criteria. Empirically, we observe that this criteria correlates well with the model robustness on the testing data.

our hyper-parameter search space. We observe that our validation criteria correlates well with the robustness of the model on the testing data.

B.4 Full results on CelebA

Table B.2: Worst-group accuracy on CelebA. The source task is to predict `Eyeglasses` and the target task is to predict `BlondHair`. We use the attribute `Young` to define two environments: $E_1 = \{\texttt{Young} = 0\}$, $E_2 = \{\texttt{Young} = 1\}$. Both environments are available in the source task. In the target task, we only have access to E_1 during training and validation.

Methods	ERM	FINETUNE PI	FINETUNE DANN	FINETUNE C-DANN	FINETUNE MMD	REUSE PI	REUSE DANN	REUSE C-DANN	REUSE MMD	MULTI TASK	EIIL	GEORGE	LFF	M-ADA	DG-MMLD	TOFU
Arched_Eyebrows	75.43	71.86	65.38	73.85	76.07	53.71	59.56	56.02	48.91	69.66	64.71	74.73	45.41	64.61	69.51	85.66
Attractive	75.00	72.73	63.35	75.61	74.33	52.13	62.03	57.78	48.46	72.73	64.43	73.66	47.67	67.33	68.42	88.30
Bags_Under_Eyes	70.91	62.50	56.86	75.86	78.57	52.50	64.58	57.50	58.74	70.00	66.67	77.78	42.59	70.34	63.41	90.38
Bald	80.79	76.84	71.14	80.92	79.58	55.56	67.99	61.70	60.00	77.18	71.71	79.07	52.05	71.57	77.30	91.53
Bangs	76.06	69.33	63.59	77.11	71.76	51.15	64.67	53.42	59.29	70.22	65.29	76.74	48.04	63.54	66.88	88.70
Big_Lips	75.29	72.73	64.46	73.03	73.89	54.41	66.09	59.24	58.75	72.00	69.32	78.24	47.88	70.79	69.13	88.66
Big_Nose	80.65	71.43	68.29	79.17	76.47	54.33	63.27	58.90	51.16	76.59	71.43	78.76	48.84	70.93	68.29	88.89
Black_Hair	80.79	76.84	71.14	80.92	79.58	55.56	67.99	61.70	60.00	77.18	71.71	79.07	52.05	71.57	77.30	91.53
Blurry	68.75	62.07	58.06	64.71	65.52	52.94	67.39	56.25	28.12	56.76	50.00	75.86	29.03	34.48	48.15	80.65
Brown_Hair	60.00	25.00	16.67	33.33	50.00	50.00	33.33	50.00	57.71	66.67	0.00	57.14	51.74	60.00	66.67	80.00
Bushy_Eyebrows	80.79	76.67	66.67	80.75	50.00	55.56	67.99	61.57	52.66	77.18	50.00	50.00	51.89	50.00	50.00	91.47
Chubby	57.14	20.00	12.50	25.00	28.57	33.33	40.00	60.00	60.00	33.33	40.00	33.33	51.05	33.33	77.22	57.14
Double_Chin	64.29	50.00	70.83	80.00	64.29	50.00	60.00	60.00	41.67	50.00	55.56	50.00	51.62	54.55	30.00	87.50
Eyeglasses	60.00	68.75	53.85	75.00	58.33	15.38	6.25	15.38	0.00	60.00	42.86	76.47	22.22	69.23	40.00	73.33
Goatee	0.00	76.84	0.00	0.00	79.58	55.56	67.99	61.70	60.00	77.10	71.71	0.00	52.05	0.00	0.00	91.53
Gray_Hair	64.29	54.55	54.55	63.64	66.67	55.44	14.29	33.33	59.85	44.44	37.50	73.33	20.00	71.28	37.50	85.71
Heavy_Makeup	70.95	65.89	56.85	70.59	71.53	52.48	56.49	50.76	44.53	66.91	54.86	70.83	38.10	62.24	60.63	84.25
High_Cheekbones	65.96	62.96	58.62	72.34	75.61	47.31	55.29	50.53	45.88	66.33	54.00	68.82	37.21	51.09	62.96	86.73
Male	22.86	20.00	21.62	26.32	34.48	32.00	26.47	43.33	39.29	33.33	23.53	40.62	20.00	21.88	10.53	66.67
Mouth_Slightly_Open	75.47	73.64	68.10	77.86	79.55	45.61	64.71	57.02	57.28	76.03	67.44	78.33	48.57	66.39	76.47	91.01
Mustache	80.79	76.84	71.14	80.92	79.58	55.56	67.99	61.70	60.00	77.18	71.71	79.07	52.05	71.57	77.30	91.53
Narrow_Eyes	74.58	76.13	59.70	78.87	78.39	52.70	67.24	60.26	52.83	71.67	68.66	67.74	50.79	58.21	69.49	87.04
No_Beard	0.00	76.75	0.00	0.00	79.58	0.00	67.99	0.00	60.00	77.10	71.62	0.00	51.89	0.00	33.33	91.50
Oval_Face	79.92	75.00	70.28	80.85	70.00	55.00	67.31	60.17	53.61	76.77	71.09	78.21	51.61	70.20	76.79	91.37
Pale_Skin	71.43	76.54	60.00	80.39	72.22	46.15	67.59	54.55	54.55	71.43	71.03	69.23	42.86	60.00	77.12	83.05
Pointy_Nose	77.72	73.30	65.82	77.03	75.13	52.69	67.98	60.87	54.30	75.26	66.00	77.42	48.28	69.19	71.58	90.20
Receding_Hairline	41.18	68.75	43.75	52.38	63.64	47.37	40.00	41.67	58.89	58.82	26.67	65.00	35.00	61.11	36.84	70.00
Rosy_Cheeks	78.49	75.11	67.47	79.41	79.20	54.22	65.18	56.06	39.39	75.89	68.50	78.17	38.36	68.80	74.68	87.69
Sideburns	0.00	76.84	0.00	0.00	79.58	55.56	67.99	61.70	60.00	77.10	71.71	0.00	52.05	0.00	0.00	91.53
Smiling	72.16	71.74	62.77	73.08	77.78	45.10	57.95	52.08	51.58	73.00	61.95	75.00	36.46	62.63	69.41	90.09
Straight_Hair	79.37	65.31	53.23	68.75	68.63	54.98	66.15	57.69	57.96	76.54	68.25	71.93	50.00	57.89	71.74	84.48
Wavy_Hair	77.19	69.01	67.47	76.24	74.68	55.47	67.82	60.14	58.99	75.33	70.19	75.80	50.60	65.06	76.43	87.95
Wearing_Earrings	71.88	70.63	64.00	72.73	75.16	54.65	60.00	55.06	52.00	70.29	64.50	76.97	43.26	66.48	70.97	87.36
Wearing_Hat	80.79	76.84	71.14	80.92	79.58	55.56	67.99	61.70	60.00	77.18	71.71	79.07	52.05	71.57	77.30	91.53
Wearing_Lipstick	51.32	46.97	40.00	46.75	56.06	46.38	45.12	46.38	41.79	50.00	41.89	66.67	32.89	50.00	42.11	76.32
Wearing_Necklace	76.12	70.22	63.45	72.77	74.72	51.87	62.87	54.60	56.10	68.95	64.53	75.00	49.73	65.05	70.83	84.65
Wearing_Necktie	0.00	20.00	0.00	0.00	20.00	50.00	16.67	50.00	0.00	0.00	0.00	0.00	20.00	0.00	0.00	57.14
5_Clock_Shadow	0.00	0.00	0.00	50.00	0.00	0.00	0.00	61.70	59.04	0.00	66.67	79.00	0.00	50.00	0.00	91.53
Avg	61.01	63.07	50.60	62.03	66.80	47.58	55.27	53.22	50.61	64.37	57.62	63.34	42.52	54.55	55.69	84.86

Table B.3: Average-group accuracy on CelebA. The source task is to predict `Eyeglasses` and the target task is to predict `BlondHair`. We use the attribute `Young` to define two environments: $E_1 = \{\texttt{Young} = 0\}$, $E_2 = \{\texttt{Young} = 1\}$. Both environments are available in the source task. In the target task, we only have access to E_1 during training and validation.

Methods	ERM	FINETUNE PI	FINETUNE DANN	FINETUNE C-DANN	FINETUNE MMD	REUSE PI	REUSE DANN	REUSE C-DANN	REUSE MMD	MULTI TASK	EIIL	GEORGE	LFF	M-ADA	DG-MMLD	TOFU
Arched_Eyebrows	88.52	87.02	83.89	88.90	88.80	64.05	72.44	67.07	59.80	86.91	85.12	87.89	60.23	83.33	87.38	91.47
Attractive	88.94	87.34	84.98	89.39	89.74	64.85	72.26	67.90	61.51	87.44	85.96	87.70	60.16	83.59	87.50	92.76
Bags_Under_Eyes	87.09	84.10	81.34	88.14	88.61	66.88	73.83	68.33	63.11	85.21	83.90	87.97	60.72	85.34	84.78	92.41
Bald	92.50	90.86	89.31	92.60	92.41	68.41	80.26	74.62	62.83	91.04	89.86	91.91	67.94	88.81	91.60	94.73
Bangs	88.69	86.76	83.95	88.83	88.91	65.25	73.53	68.06	61.32	86.98	84.70	87.45	59.25	83.02	86.92	91.96
Big_Lips	88.99	87.23	84.19	89.04	88.87	64.42	73.86	68.45	61.30	87.24	84.91	87.87	60.98	83.13	87.56	91.78
Big_Nose	89.17	85.87	83.58	88.71	88.20	66.33	73.79	72.28	60.50	87.47	84.89	88.52	61.76	84.47	85.73	92.24
Black_Hair	92.36	90.98	89.16	92.46	92.33	69.10	78.04	73.11	61.93	90.88	89.76	91.64	65.59	88.79	91.49	94.59
Blurry	85.84	83.87	81.31	85.65	84.64	64.00	73.70	67.50	57.77	82.58	79.89	87.05	57.64	75.71	80.97	89.54
Brown_Hair	83.48	74.15	70.11	77.46	81.59	62.60	63.56	65.43	64.43	84.43	66.88	82.78	65.16	79.39	84.58	89.42
Bushy_Eyebrows	92.51	93.35	83.67	94.41	81.87	68.45	77.33	80.75	59.30	91.05	79.81	81.49	75.26	79.29	81.25	95.90
Chubby	83.66	73.26	70.27	75.81	76.60	59.84	68.50	69.98	62.31	76.28	77.23	77.25	74.76	74.74	93.51	84.85
Double_Chin	85.40	80.78	86.50	89.03	85.46	63.08	73.40	69.35	58.24	80.44	81.00	81.38	65.47	80.04	76.46	92.14
Eyeglasses	84.55	85.51	80.49	87.93	83.80	56.59	63.18	62.01	54.50	83.33	78.39	87.63	56.20	83.68	78.97	89.34
Goatee	69.44	91.19	67.04	69.51	92.41	70.67	78.34	73.38	61.62	93.14	89.86	69.00	66.23	66.54	68.77	94.74
Gray_Hair	84.77	81.38	79.83	84.56	85.53	64.70	60.83	61.60	64.12	77.40	76.56	86.77	56.57	86.24	77.99	90.88
Heavy_Makeup	87.65	86.04	82.84	87.97	88.33	64.24	70.58	66.23	59.03	85.76	83.80	86.77	57.57	82.23	85.58	91.89
High_Cheekbones	86.81	84.89	82.24	87.76	88.11	63.41	72.29	67.73	60.11	85.31	82.59	86.57	59.92	80.71	85.33	91.98
Male	75.65	73.61	72.84	76.65	78.35	58.93	64.02	63.84	57.06	76.65	73.87	78.95	53.36	72.08	71.41	86.09
Mouth_Slightly_Open	88.26	86.66	83.82	88.77	88.73	63.64	73.97	68.89	61.89	86.63	84.59	87.93	61.74	83.20	87.41	92.68
Mustache	92.50	91.18	89.31	92.60	92.42	70.96	80.02	74.32	62.43	90.88	89.70	91.74	67.73	88.35	91.61	94.90
Narrow_Eyes	87.39	87.46	82.06	88.62	89.73	65.57	74.24	69.02	62.02	85.66	84.32	85.98	61.69	81.11	85.95	91.55
No_Beard	69.29	93.18	66.99	69.43	92.17	52.43	77.55	55.04	62.96	92.90	92.21	68.85	75.04	66.44	77.04	95.72
Oval_Face	90.37	89.64	85.11	89.30	86.80	64.36	74.17	71.06	65.05	87.28	85.68	89.15	61.67	85.11	88.25	93.15
Pale_Skin	84.86	87.29	78.96	87.83	87.05	62.61	74.97	66.94	61.23	82.93	86.75	85.24	62.05	81.10	87.52	89.09
Pointy_Nose	88.96	87.15	84.79	89.31	89.70	64.31	73.78	68.35	62.87	86.78	85.66	87.19	62.81	84.03	88.31	92.46
Receding_Hairline	79.83	85.35	78.16	82.79	85.12	62.63	68.56	65.37	65.27	82.72	74.29	85.13	59.37	81.64	78.43	88.25
Rosy_Cheeks	89.36	87.33	87.21	90.18	88.66	64.09	73.83	67.34	57.29	86.74	87.18	88.25	62.42	84.55	88.97	92.44
Sideburns	69.45	91.20	67.04	69.38	92.41	71.82	77.79	73.31	63.78	93.02	89.87	69.00	66.78	66.28	68.78	94.91
Smiling	87.74	86.29	82.97	88.00	88.47	63.41	72.87	68.04	61.17	86.21	83.88	87.58	60.19	82.50	86.38	92.49
Straight_Hair	88.96	84.84	81.07	87.03	86.62	64.73	74.71	68.96	63.46	87.30	84.27	86.76	62.28	81.02	86.64	91.13
Wavy_Hair	88.67	86.64	83.64	88.89	88.79	65.01	73.11	67.84	62.01	86.55	84.70	87.69	60.87	83.03	87.14	92.32
Wearing_Earrings	88.30	86.72	83.74	88.77	88.57	64.06	73.10	68.00	61.03	86.66	84.54	87.49	59.64	83.36	86.97	92.06
Wearing_Hat	92.54	90.68	89.34	92.63	92.44	69.06	79.95	73.52	69.47	91.09	89.88	91.67	66.86	89.02	91.63	94.69
Wearing_Lipstick	82.97	81.17	78.10	82.33	84.24	62.53	68.30	64.67	57.33	81.22	79.07	85.07	56.03	78.93	79.89	88.85
Wearing_Necklace	88.63	87.57	84.75	89.72	88.61	63.96	72.85	67.88	61.19	87.65	85.54	87.75	58.05	83.66	87.18	91.03
Wearing_Necktie	69.35	73.32	67.03	69.56	74.40	63.62	63.11	67.21	46.32	68.03	67.50	68.88	56.13	66.31	68.81	84.64
5_Clock_Shadow	69.38	68.21	66.92	81.87	69.33	52.69	57.14	72.06	70.87	68.06	83.83	93.66	51.29	78.39	68.77	93.93
Avg	85.07	85.27	80.49	85.57	86.81	64.14	72.31	68.56	61.27	85.21	83.22	85.04	62.04	80.77	83.51	91.71

Appendix C

Learning to Split for Automatic Bias Detection

C.1 Datasets and model architectures

C.1.1 Beer Review

Data We use the BeerAdvocate review dataset [82] and consider three *binary* aspect-level sentiment classification tasks: LOOK, AROMA and PALATE. This dataset was originally downloaded from `https://snap.stanford.edu/data/web-BeerAdvocate.html`.

[69] points out that there exist strong correlations between the ratings of different aspects. In fact, the average correlation between two different aspects is 0.635. These correlations constitute as a source of biases when we apply predictors to examples with conflicting aspect ratings (e.g. beers that looks great but smells terrible).

We randomly sample 2500 positive examples and 2500 negative examples for each task. We apply `ls` to identify non-generalizable splits across these 5000 examples. The average word count per review is 128.5.

Representation backbone Following previous work [9], we use a simple text CNN for this dataset. Specifically, each input review is encoded by pre-trained FastText embeddings [84]. We employ 1D convolutions (with filter sizes $3, 4, 5$) to extract the features [61]. We use 50 filters for each filter size. We apply max pooling to obtain the final representation ($\in \mathbb{R}^{150}$) for the input.

Predictor The Predictor applies a multi-layer perceopton on top of the previous input representation to predict the binary label. We consider a simple MLP with one hidden layer (150 hidden units). We apply ReLU activations and dropout (with rate 0.1) to the hidden units.

Splitter The Splitter concatenates the CNN representation with the binary input label. Similar to the Predictor, we use a MLP with one hidden layer (150 ReLU units, dropout 0.1) to predict the splitting decision $\mathbb{P}(z_i \mid x_i, y_i)$. We note that the representation backbones of the Splitter and the Predictor are *not shared* during training.

C.1.2 Tox21

Data The dataset contains 12,707 chemical compounds. Each example is annotated with two types of properties: Nuclear Receptor Signaling Panel (AR, AhR, AR-LBD, ER, ER-LBD, aromatase, PPAR-gamma) and Stress Response Panel (ARE, ATAD5, HSE, MMP, p53). The dataset is publicly available at `http://bioinf.jku.at/research/DeepTox/tox21.html`.

Representation backbone Following [80], we encode each input molecule by its dense features (such as molecular weight, solubility or surface area) and sparse features (chemical substructures). There are 801 dense features and 272,776 sparse features. We concatenate these features and standardize them by removing the mean and scaling to unit variance.

Predictor The Predictor is a multi-layer perceptron with three hidden layers (each with 1024 units). We apply ReLU activations and dropout (with rate 0.3) to the hidden units.

Splitter The Splitter concatenates the molecule features with the binary input label. Similar to the Predictor, we is a multi-layer perceptron with three hidden layers (each with 1024 units). We apply ReLU activations and dropout (with rate 0.3) to the hidden units.

C.1.3 Waterbird

Data This dataset is constructed from the CUB bird dataset [114] and the Places dataset [124]. [93] use the provided pixel-level segmentation information to crop each bird out from the its original background in CUB. The resulting birds are then placed onto different backgrounds obtained from Places. They consider two types of backgrounds: water (ocean or natural lake) and land (bamboo forest or broadleaf forest). There are 4795/1199/5794 examples in the training/validation/testing set. This dataset is publicly available at `https://nlp.stanford.edu/data/dro/waterbird_complete95_forest2water2.tar.gz`

By construction, 95% of all waterbirds in the training set have water backgrounds. Similarly, 95% of all landbirds in the training set have land backgrounds. As a result, predictors trained on this training data will overfit to the spurious background information when making their predictions. In the validation and testing sets, [93] place landbirds and waterbirds equally to land and water backgrounds.

For identifying non-generalizable splits, we apply `ls` on the training set and the validation set. For automatic de-biasing, we report the average accuracy and worst-group accuracy on the official test set. To compute the worst-group accuracy, we use the background attribute to partition the test set into four groups: `waterbirds` with water backgrounds, `waterbirds` with land backgrounds, `landbirds` with water backgrounds, `landbirds` with land backgrounds.

Representation backbone Following previous work [93, 74], we fine-tune torchvision's `resnet-50`, pretrained on ImageNet [32], to represent each input image. This results into a 2048 dimensional feature vector for each image.

Predictor The Predictor takes the `resnet` representation and applies a linear layer (2048 by 2) followed by Softmax to predict the label ({`waterbirds`, `landbirds`}) of each image.

Splitter The Splitter first concatenates the `resnet` representation with the binary image label. It then applies a linear layer with Softmax to predict the splitting decision $\mathbb{P}(z_i \mid x_i, y_i)$. The `resnet` encoders for the Splitter and the Predictor are not shared during training.

C.1.4 CelebA

Data CelebA [76] is a large-scale face attributes dataset, where each image is annotated with 40 binary attributes. Following previous work [93, 74], we consider our task as predicting the blond hair attribute ($\in \{\texttt{blond\_hair}, \texttt{no\_blond\_hair}\}$). The CelebA dataset is available for non-commercial research purposes only. It is publicly available at `https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`.

While there are lots of annotated examples in the training set (162,770), the task is challenging due to the spurious correlation between the target blond hair attribute and the gender attribute ($\in \{\texttt{male}, \texttt{female}\}$). Specifically, only 0.85% of the training data are blond-haired males. As a result, predictors learn to utilize `male` as a predictive feature for `no_blond_hair` when we directly minimizing their empirical risk.

For identifying non-generalizable splits, we apply `ls` on the official training set and validation set. For automatic de-biasing, we report the average and worst-group performance on the official test set. To compute the worst-group accuracy, we use the gender attribute to partition the test set into four groups: `blond_hair` with male, `blond_hair` with female, `no_blond_hair` with male, `no_blond_hair` with female.

Representation backbone Following previous work [93, 74], we fine-tune torchvision's `resnet-50`, pretrained on ImageNet [32], to represent each input image. This results into a 2048 dimensional feature vector for each image.

Predictor The Predictor takes the `resnet` representation and applies a linear layer (2048 by 2) followed by Softmax to predict the label ({`blond_hair`, `no_blond_hair`}) of each image.

Splitter The Splitter concatenates the `resnet` representation with the binary image label. It then applies a linear layer with Softmax to predict the splitting decision $\mathbb{P}(z_i \mid x_i, y_i)$. The `resnet` encoders for the Splitter and the Predictor are not shared during training.

C.1.5 MNLI

Data The MultiNLI corpus contains 433k sentence pairs [115]. Given a sentence pair, the task is to predict the entailment relationship (`entailment`, `contradiction`, `neutral`) between the two sentences. The original corpus splits allocate most examples to the training set, with another 5% for validation and the last 5% for testing. In order to accurately measure the performance on rare groups, [93] combine the training and validation set and randomly shuffle them into a 50/20/30 training/validation/testing split. The dataset and splits are publicly available at `https://github.com/kohpangwei/group_DRO`.

Previous work [47, 83] have shown that this crowd-sourced dataset has significant annotation artifacts: negation words (nobody, no, never and nothing) often appears in `contradiction` examples; sentence pairs with high lexical overlap are likely to be `entailment`. As a result, predictors may over-fit to these spurious shortcuts during training.

For identifying non-generalizable splits, we apply `ls` on the training set and validation set. For automatic de-biasing, we report the average and worst-group performance on the testing set. To compute the worst-group accuracy, we partition the test set based on whether the input example contains negation words or not: `entailment` with negation words, `entailment` without negation words, `contradiction` with negation words, `contradiction` without negation words, `neutral` with negation words, `neutral` without negation words.

Representation backbone Following previous work [93, 74], we fine-tune Hugging Face's `bert-base-uncased` model, starting with pre-trained weights [34].

Predictor The Predictor takes the representation of the `[CLS]` token (at the final layer of `bert-base-uncased`) and applies a linear layer with Softmax activations to predict the final label (`entailment`, `contradictions`, `neutral`).

Splitter The Splitter concatenates the representation of the `[CLS]` token with the *one-hot* label embedding ($\in \{0, 1\}^3$). It then applies a linear layer with Softmax activations to predict the splitting decision $\mathbb{P}(z_i \mid x_i, y_i)$. The `bert-base-uncased` encoders for the Splitter and Predictor are not shared during training.

C.2 Implementation details

C.2.1 Identifying non-generalizable splits using `ls`

Optimization For Beer Review, Tox21, Waterbirds and CelebA, we update the Splitter and Predictor with the Adam optimizer [62]. In Beer Review and Tox21, the learning rate is set to 10^{-3} with no weight decay (as we already have dropout in the MLP to prevent over-fitting). We use a batch size of 200. In Waterbirds and CelebA, since we start with pre-trained weights, we adopt a smaller learning rate 10^{-4} [93] and set weight decay to 10^{-3}. We use a batch size of 100. For MNLI, we use the default setting for fine-tuning BERT: a fixed linearly-decaying learning rate starting at 0.0002, AdamW optimizer [77], dropout, and no weight decay. We use a batch size of 100.

Stopping criteria For the Predictor's training, we held out a random 1/3 subset of $\mathcal{D}^{\text{train}}$ for validation. We train the Predictor on the rest of $\mathcal{D}^{\text{train}}$ and apply early-stopping when the validation accuracy stops improving in the past 5 epochs. For the Splitter's training, we compare the average loss $\mathcal{L}^{\text{total}}$ of the current epoch and the average loss across the past 5 epochs. We stop training if the improvement is less than 10^{-3}.

C.2.2 Automatic de-biasing

Method details We use the Splitter learned by `ls` to create groups that are informative of the biases. Specifically, for each example (x_i, y_i), we first sample its splitting decision from the Splitter $\hat{z}_i \sim \mathbb{P}(z_i \mid x_i, y_i)$. As we have seen in Figure 4-5, these splitting decisions reveal human-identified biases. Similar to the typical group DRO setup [93], we use these information together with the target labels to partition the training and validation data into different groups. For example in Waterbirds, we have four groups: $\{y = \texttt{waterbirds}, z = 0\}, \{y = \texttt{waterbirds}, z = 1\}, \{y = \texttt{landbirds}, z = 0\}, \{y = \texttt{landbirds}, z = 1\}$. We minimize the worst-group loss during training and measure the worst-group accuracy on the validation data for model selection. Specifically, we stop training if the validation metric hasn't improved in the past 10 epochs.

Optimization Modern neural networks are highly over-parameterized. As a result, they can easily memorize the training data and over-fit the majority groups even when we minimize the worst-group loss during training. Following [93], we apply strong regularization to combat memorization and over-fitting. We grid-search over the weight decay parameter $(10^0, 10^{-1}, 10^{-2}, 10^{-3}, 0)$.

Bibliography

[1] Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. Analyzing the behavior of visual question answering models. *arXiv preprint arXiv:1606.07356*, 2016.

[2] Faruk Ahmed, Yoshua Bengio, Harm van Seijen, and Aaron Courville. Systematic generalisation with group invariant predictions. In *International Conference on Learning Representations*, 2020.

[3] Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *International Conference on Machine Learning*, pages 145–155. PMLR, 2020.

[4] David M Allen. The relationship between variable selection and data agumentation and a method for prediction. *technometrics*, 16(1):125–127, 1974.

[5] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

[6] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[7] Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE transactions on medical imaging*, 38(2):550–560, 2018.

[8] Yujia Bao and Regina Barzilay. Learning to split for automatic bias detection. *arXiv preprint arXiv:2204.13749*, 2022.

[9] Yujia Bao, Shiyu Chang, and Regina Barzilay. Predict then interpolate: A simple algorithm to learn stable classifiers. In *International Conference on Machine Learning (ICML)*, 2021.

[10] Yujia Bao, Shiyu Chang, and Regina Barzilay. Learning stable classifiers by transferring unstable features. In *International Conference on Machine Learning (ICML)*, 2022.

[11] Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913, 2018.

[12] Yujia Bao, Zhengyi Deng, Yan Wang, Heeyoon Kim, Victor Diego Armengol, Francisco Acevedo, Nofal Ouardaoui, Cathy Wang, Giovanni Parmigiani, Regina Barzilay, Danielle Braun, and Kevin S. Hughes. Using machine learning and natural language processing to review and classify the medical literature on cancer susceptibility genes. *JCO Clinical Cancer Informatics*, pages 1–9, 2019. PMID: 31545655.

[13] Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*, 2019.

[14] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pages 456–473, 2018.

[15] Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander M Rush. Don't take the premise for granted: Mitigating artifacts in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 877–891, 2019.

[16] Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander M Rush. On adversarial removal of hypothesis-only bias in natural language inference. *arXiv preprint arXiv:1907.04389*, 2019.

[17] Aharon Ben-Tal, Dick Den Hertog, Anja De Waegenaere, Bertrand Melenberg, and Gijs Rennen. Robust solutions of optimization problems affected by uncertain probabilities. *Management Science*, 59(2):341–357, 2013.

[18] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[19] Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019.

[20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[21] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[22] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[23] Damon Centola, Joshua Becker, Devon Brackbill, and Andrea Baronchelli. Experimental evidence for tipping points in social convention. *Science*, 360(6393):1116–1119, 2018.

[24] Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S Jaakkola. Invariant rationalization. *arXiv preprint arXiv:2003.09772*, 2020.

[25] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3), 2010.

[26] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[27] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR, 10–15 Jul 2018.

[28] Yo Joong Choe, Jiyeon Ham, and Kyubyong Park. An empirical study of invariant risk minimization. *arXiv preprint arXiv:2004.05007*, 2020.

[29] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4069–4082, 2019.

[30] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.

[31] Terrance de Vries, Ishan Misra, Changhan Wang, and Laurens van der Maaten. Does object recognition work for everyone? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 52–59, 2019.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[33] Zhengyi Deng, Kanhua Yin, Yujia Bao, Victor Diego Armengol, Cathy Wang, Ankur Tiwari, Regina Barzilay, Giovanni Parmigiani, Danielle Braun, and Kevin S Hughes. Validation of a semiautomated natural language processing–based procedure for meta-analysis of cancer susceptibility gene penetrance. *JCO clinical cancer informatics*, 3:1–9, 2019.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[35] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[36] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. Bold: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 862–872, 2021.

[37] John Duchi and Hongseok Namkoong. Learning models with uniform performance via distributionally robust optimization. *arXiv preprint arXiv:1810.08750*, 2018.

[38] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018.

[39] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[40] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2018.

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

[42] Andrew Gaut, Tony Sun, Shirlyn Tang, Yuxin Huang, Jing Qian, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, et al. Towards understanding gender bias in relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2943–2953, 2020.

[43] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[44] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[45] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[46] Umang Gupta, Jwala Dhamala, Varun Kumar, Apurv Verma, Yada Pruksachatkun, Satyapriya Krishna, Rahul Gupta, Kai-Wei Chang, Greg Ver Steeg, and Aram Galstyan. Mitigating gender bias in distilled language models via counterfactual role reversal. *arXiv preprint arXiv:2203.12574*, 2022.

[47] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

[48] He He, Sheng Zha, and Haohan Wang. Unlearn dataset bias in natural language inference by fitting the residual. *EMNLP-IJCNLP 2019*, page 132, 2019.

[49] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721. PMLR, 2019.

[50] Katherine Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:19000–19015, 2020.

[51] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[52] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[53] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[54] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning*, pages 2029–2037. PMLR, 2018.

[55] Ruili Huang, Menghang Xia, Dac-Trung Nguyen, Tongan Zhao, Srilatha Sakamuru, Jinghua Zhao, Sampada A. Shahane, Anna Rossoshek, and Anton Simeonov. Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science*, 3, 2016.

[56] Shengyu Jia, Tao Meng, Jieyu Zhao, and Kai-Wei Chang. Mitigating gender bias amplification in distribution by posterior regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2936–2942, 2020.

[57] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, pages 2304–2313. PMLR, 2018.

[58] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Enforcing predictive invariance across structured biomedical domains, 2020.

[59] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[60] Masahiro Kaneko and Danushka Bollegala. Debiasing pre-trained contextualised embeddings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1256–1266, 2021.

[61] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

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

[63] Svetlana Kiritchenko and Saif Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 43–53, 2018.

[64] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.

[65] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex), 2020.

[66] Kun Kuang, Ruoxuan Xiong, Peng Cui, Susan Athey, and Bo Li. Stable prediction with model misspecification and agnostic distribution shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4485–4492, 2020.

[67] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in Neural Information Processing Systems*, 30, 2017.

[68] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[69] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, 2016.

[70] Daniel Levy, Yair Carmon, John C Duchi, and Aaron Sidford. Large-scale methods for distributionally robust optimization. *Advances in Neural Information Processing Systems*, 33:8847–8860, 2020.

[71] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5400–5409, 2018.

[72] Ya Li, Mingming Gong, Xinmei Tian, Tongliang Liu, and Dacheng Tao. Domain generalization via conditional invariant representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[73] Zhiheng Li and Chenliang Xu. Discover the unknown biased attribute of an image classifier. In *The IEEE International Conference on Computer Vision (ICCV)*, 2021.

[74] Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pages 6781–6792. PMLR, 2021.

[75] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

[76] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[77] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[78] Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-end bias mitigation by modelling biases in corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8706–8716, 2020.

[79] Toshihiko Matsuura and Tatsuya Harada. Domain generalization using a mixture of multiple latent domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11749–11756, 2020.

[80] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: Toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3, 2016.

[81] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical science*, 9(24):5441–5451, 2018.

[82] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *2012 IEEE 12th International Conference on Data Mining*, pages 1020–1025. IEEE, 2012.

[83] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

[84] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[85] Junhyun Nam, Hyuntak Cha, Sungsoo Ahn, Jaeho Lee, and Jinwoo Shin. Learning from failure: Training debiased classifier from biased classifier. *arXiv preprint arXiv:2007.02561*, 2020.

[86] Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4218–4228, 2019.

[87] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2799–2804, 2018.

[88] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference using invariant prediction: identification and confidence intervals. *arXiv preprint arXiv:1501.01332*, 2015.

[89] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 947–1012, 2016.

[90] Fengchun Qiao, Long Zhao, and Xi Peng. Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12556–12565, 2020.

[91] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[92] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, pages 4334–4343, 2018.

[93] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

[94] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020.

[95] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning (ICML)*, 2020.

[96] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[97] Victor Sanh, Thomas Wolf, Yonatan Belinkov, and Alexander M Rush. Learning from others' mistakes: Avoiding dataset biases without modeling them. In *International Conference on Learning Representations*, 2021.

[98] Tal Schuster, Darsh Shah, Yun Jie Serene Yeo, Daniel Roberto Filizzola Ortiz, Enrico Santus, and Regina Barzilay. Towards debiasing fact verification models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3410–3416, Hong Kong, China, November 2019. Association for Computational Linguistics.

[99] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020.

[100] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. In *International Conference on Learning Representations*, 2020.

[101] Zheyan Shen, Peng Cui, Jiashuo Liu, Tong Zhang, Bo Li, and Zhitang Chen. Stable learning via differentiated variable decorrelation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2185–2193, 2020.

[102] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. *arXiv preprint arXiv:1909.01326*, 2019.

[103] Robert P Sheridan. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of chemical information and modeling*, 53(4):783–790, 2013.

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

[105] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[106] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[107] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4080–4090, 2017.

[108] Nimit Sohoni, Jared Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *Advances in Neural Information Processing Systems*, 33, 2020.

[109] Joe Stacey, Pasquale Minervini, Haim Dubossarsky, Sebastian Riedel, and Tim Rocktäschel. Avoiding the Hypothesis-Only Bias in Natural Language Inference via Ensemble Adversarial Training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8281–8291, Online, November 2020. Association for Computational Linguistics.

[110] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36(2):111–133, 1974.

[111] J Taylor, B Earnshaw, B Mabey, M Victors, and J Yosinski. Rxrx1: An image set for cellular morphological variation across many experimental batches. In *The 7th International Conference on Learning Representations*, 2019.

[112] Prasetya Ajie Utama, Nafise Sadat Moosavi, and Iryna Gurevych. Towards debiasing nlu models from unknown biases. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7597–7610, 2020.

[113] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[114] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[115] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018.

[116] James Woodward. *Making things happen: A theory of causal explanation.* Oxford university press, 2005.

[117] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[118] Adam Yala, Peter G Mikhael, Fredrik Strand, Gigin Lin, Siddharth Satuluru, Thomas Kim, Imon Banerjee, Judy Gichoya, Hari Trivedi, Constance D Lehman, et al. Multi-institutional validation of a mammography-based breast cancer risk model. *Journal of Clinical Oncology*, pages JCO–21, 2021.

[119] Adam Yala, Peter G Mikhael, Fredrik Strand, Gigin Lin, Kevin Smith, Yung-Liang Wan, Leslie Lamb, Kevin Hughes, Constance Lehman, and Regina Barzilay. Toward robust mammography-based models for breast cancer risk. *Science Translational Medicine*, 13(578):eaba4373, 2021.

[120] Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling*, 59(8):3370–3388, 2019.

[121] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th*

Annual Meeting of the Association for Computational Linguistics, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics.

[122] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[123] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[124] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[125] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. *Advances in neural information processing systems*, 27, 2014.