



UNIVERSITY OF GENOVA

PHD COURSE: SCIENCES AND TECHNOLOGIES FOR ELECTRONICS AND TELECOMMUNICATION  
ENGINEERING  
CURRICULUM: COMPUTER VISION, PATTERN RECOGNITION AND MACHINE LEARNING

## Addressing Dataset Bias in Deep Neural Networks

by

**Ruggero Ragonesi**

Thesis submitted for the degree of *Doctor of Philosophy* (XXXIV cycle)

December 2021

Vittorio Murino

Jacopo Cavazza

Maurizio Valle

Supervisor

Co-Supervisor

Head of the PhD program

***Thesis Jury:***

Name, *University*

Name, *University*

Name, *University*

External examiner

External examiner

Internal examiner



Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni (DITEN)

## **Declaration**

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Your name  
February 2022

## Abstract

Deep Learning has achieved a tremendous success in recent years in several areas such as image classification, text translation, autonomous agents, to name a few. Deep Neural Networks are able to learn non-linear features in a data-driven fashion from complex, large scale datasets to solve tasks. However, some fundamental issues remain to be fixed: the kind of data that is provided to the neural network directly influences its capability to generalize. This is especially true when training and test data come from different distributions (the so called domain gap or domain shift problem): in this case, the neural network may learn a data representation which is representative for the training data but not for the test, thus performing poorly when deployed in actual scenarios. The domain gap problem is addressed by the so-called Domain Adaptation, for which a large literature was recently developed.

In this thesis, we first present a novel method to perform Unsupervised Domain Adaptation. Starting from the typical scenario in which we dispose of labeled source distributions and an unlabeled target distribution, we pursue a pseudo-labeling approach to assign a label to the target data and then, in an iterative way, we refine them using Generative Adversarial Networks.

Subsequently, we faced the debiasing problem. Simply speaking, bias occurs when there are factors in the data which are spuriously correlated with the task label, e.g., the background, which might be a strong clue to guess what class is depicted in an image. When this happens, neural networks may erroneously learn such spurious correlations as predictive factors, and may therefore fail when deployed on different scenarios. Learning a debiased model can be done using a supervision regarding the type of bias affecting the data, or can be done without any annotation about what are the spurious correlations.

We tackled the problem of *supervised* debiasing – where a ground truth annotation for the bias is given – under the lens of information theory. We designed a neural network architecture which learns to solve the task while achieving at the same time, statistical independence of the data embedding with respect to the bias label.

We finally addressed the *unsupervised* debiasing problem, in which there is no availability of bias annotation. we address this challenging problem by a two-stage approach: we first

split coarsely the training dataset in two subsets, samples that exhibit spurious correlations and those that do not. Second, we learn a feature representation which can accommodate both subsets and an augmented version of them.



# Table of contents

<b>List of figures</b>	<b>vii</b>
<b>List of tables</b>	<b>xi</b>
<b>I Rationale and Background</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Overview . . . . .	2
1.2 Contributions . . . . .	5
1.3 Publications . . . . .	7
1.4 Summary . . . . .	7
<b>2 Background and Related Works</b>	<b>8</b>
2.1 Domain Adaptation . . . . .	8
2.1.1 Unsupervised Domain Adaptation . . . . .	9
2.2 Out-of-Distribution generalization . . . . .	11
2.2.1 Domain Generalization . . . . .	11
2.2.2 Supervised Debiasing . . . . .	13
2.2.3 Unsupervised Debiasing . . . . .	14
<b>II Contributions</b>	<b>16</b>
<b>3 Generative Pseudo-Label refinement for Unsupervised Domain Adaptation</b>	<b>17</b>
3.1 Context . . . . .	17
3.2 Related works . . . . .	19
3.3 Robustness against label noise . . . . .	21
3.3.1 Shift noise. . . . .	21

---

3.3.2	Classifiers . . . . .	23
3.3.3	cGANs . . . . .	24
3.4	Application to UDA . . . . .	26
3.5	Experiments . . . . .	29
3.5.1	Results . . . . .	31
3.6	Conclusion . . . . .	32
<b>4</b>	<b>Learning Unbiased representations via Mutual Information Backpropagation</b>	<b>33</b>
4.1	Context . . . . .	33
4.2	Related works . . . . .	36
4.3	Problem Formulation . . . . .	37
4.4	Method . . . . .	39
4.4.1	Optimization problem . . . . .	39
4.4.2	Implementation Details . . . . .	41
4.5	Experiments . . . . .	43
4.5.1	Digit Recognition . . . . .	43
4.5.2	IMDB: Removing the Age Bias . . . . .	46
4.5.3	Learning Fair Representations . . . . .	47
4.6	Conclusions . . . . .	49
<b>5</b>	<b>Learning to learn Unbiased Models from Biased data</b>	<b>50</b>
5.1	Context . . . . .	50
5.2	Related works . . . . .	53
5.3	Method . . . . .	54
5.3.1	Bias Identification . . . . .	56
5.3.2	Bias-invariant representation learning . . . . .	56
5.4	Experiments . . . . .	58
5.4.1	Benchmarks . . . . .	59
5.4.2	Performances . . . . .	60
5.4.3	Ablation Study . . . . .	63
5.5	Conclusions . . . . .	65
<b>6</b>	<b>Conclusions</b>	<b>66</b>
	<b>References</b>	<b>68</b>

---

<b>Appendix A</b>	<b>Supplementary material for Chapter 3</b>	<b>78</b>
A.1	Architectures and Hyperparameters . . . . .	78
A.2	Are deeper architectures more resistant against shift noise? . . . . .	78
A.3	Generated images . . . . .	81
<b>Appendix B</b>	<b>Supplementary material for Chapter 4</b>	<b>86</b>
B.1	Implementation Details . . . . .	86
B.2	Discussion on the Hyper-Parameters . . . . .	88
<b>Appendix C</b>	<b>Supplementary material for Chapter 5</b>	<b>91</b>
C.1	Ablation study on $\zeta$ . . . . .	91
C.2	Ablation study on bias identification . . . . .	92
C.3	Implementation details . . . . .	93

# List of figures

1.1	Example of distributional shift from Painting domain to Photo domain (images from the PACS dataset (64)). . . . .	3
3.1	<i>Top</i> : a neural network classifier $C$ is typically not robust against shift noise, here represented by an histogram. <i>Bottom</i> : a cGAN generator $G$ is able to filter such structured noise, making it more uniform and thus tolerable from the classifier. By jointly training $C$ and $G$ , the former benefits from “cleaner” generated data, while the latter from more accurate inferred labels. . . . .	18
3.2	<i>Left</i> panel plots the fraction of images correctly generated by cGANs with different levels $n$ of uniform noise and different objectives ( <i>blue</i> : cross-entropy (37), <i>red</i> : Hinge (79), <i>green</i> : least-squares (76)), evaluated through the oracle. <i>Yellow</i> bars indicate the percentage of images with the correct label in the training set. <i>Right</i> panel shows the FID scores achieved with different levels of noise and different GAN objectives (same as <i>left</i> ). . . . .	24
3.3	Evolution of the accuracy on target test sets for SVHN $\rightarrow$ MNIST, MNIST $\rightarrow$ MNIST-M and MNIST $\rightarrow$ SVHN (from <i>left</i> to <i>right</i> ), computed throughout the training procedure described in Algorithm 1. <i>Blue</i> , <i>red</i> and <i>green</i> curves are associated with GANs trained with the cross-entropy loss (37), the Hinge loss (79) and the least-squares loss (76), respectively. Results obtained with the least-squares loss are not reported for the MNIST $\rightarrow$ MNIST-M as they are significantly worse than the ones achieved with the other options. Curves are averaged over three different runs, shades represent the confidence bands.	27

- 3.4 Graphical view of Algorithm 1. Step 1 (*top*) and step 2 (*bottom*) refer to lines 3 – 5 and 6 – 9, respectively. The module  $G$  and the module  $D$  are the generator and the discriminator of the cGAN. The module  $C$  is the classifier. Dashed boxes indicate frozen modules (not trained). Solid and dashed wires indicate image and label flows, respectively.  $\Pi_{[0,N]}$  is the discrete uniform distribution. . . . . 28
- 4.1 **Problem setting.** When learning a feature representation from the data itself (*top*), we may undesirably capture the inherent bias of the dataset (here, exemplified by colors), as opposed to learning the desired patterns (here, represented by shapes). This results in models that poorly generalize when deployed into unbiased scenarios (*bottom*). . . . . 34
- 4.2 **Model overview.** The neural network devised for the given task is the concatenation of the blue module (feature extractor  $g_\theta$ ) and the green module (logit layer  $f_\psi$ ). Solid lines indicate the forward flow, dashed lines indicate gradient backward passes. The feature extractor takes in input samples  $\mathbf{x}$  and outputs feature vectors  $\mathbf{z}$ . The logit layer takes in input the feature vectors and outputs predictions  $\tilde{\mathbf{y}}$ . To optimize for the given task, these modules can be trained by minimizing the cross-entropy between predictions and labels  $\mathbf{y}$ . The orange module (11) estimates the mutual information between the feature vectors  $\mathbf{z}$  and the attributes  $\mathbf{c}$ . To estimate the mutual information,  $T_\phi$  processes the concatenation of feature vectors and attributes from the joint distribution and the marginals. Following Belghazi et al. (11), we approximate sampling from the marginal by shuffling the batch of attributes ( $\tilde{\mathbf{c}}$ ). The estimation of the mutual information is the maximum w.r.t.  $\phi$  of the output of the orange module  $\mathcal{L}_{ne}$ . . . . . 40
- 4.3 *Left:* digit examples for each class from training (here with  $\sigma = 0.02$ ) and test set. *Right:* Women and Men images from the two splits of the training set of the IMDB dataset. . . . . 43

4.4	<b>Digit experiment – ablation study.</b> Evolution of mutual information estimation (left), test accuracy (middle) and training accuracy (right) for models trained on digits with $\sigma = 0.03$ and $\sigma = 0.045$ (top and bottom, respectively). Models are trained with Algorithm 3 with $\lambda = 0.0$ (baseline, blue), $\lambda = 0.5$ (orange) and $\lambda = 1.0$ (green). Increasing the value of the hyper-parameter $\lambda$ allows reducing the mutual information between the learned representation ( $Z$ ) and the attributes ( $C$ ). In turn, models better generalize to unbiased samples (test set). Further plots in Appendix B. . . . .	44
4.5	The two considered metrics vary as we modify the hyper-parameter $\lambda$ on the German dataset. EO ( <i>Left</i> ) is significantly reduced as we set higher values of $\lambda$ . Vice versa, test accuracy ( <i>Right</i> ) is only slightly affected. . . . .	48
5.2	Starting from the current parameter configuration $\theta$ , gradients on $\mathcal{L}(\hat{\mathcal{D}}_{bias}, f_{\theta})$ and $\mathcal{L}(\hat{\mathcal{D}}_{unbias}, f_{\theta})$ are evaluated to produce the new configuration $\theta^*$ . The regularization step using mixed data aims at producing a contribution that decreases the loss function on $\hat{\mathcal{D}}_{bias}$ , $\hat{\mathcal{D}}_{unbias}$ , and $\hat{\mathcal{D}}_{mix}$ , simultaneously, the latter estimated over the configuration $\theta^*$ . (Best viewed in color) . . . . .	58
5.3	Examples of biased training data and unbiased data (with red boundary) from Waterbirds and BAR. . . . .	59
5.4	Test accuracies at different values of $\gamma$ (from 0.8 to 0.95). We compare with ERM baseline, Nam et al. (85), and our method using the ground-truth bias knowledge as an oracle ( $\hat{\mathcal{D}}_{bias} = \mathcal{D}_{bias}$ and $\hat{\mathcal{D}}_{unbias} = \mathcal{D}_{unbias}$ ). . . . .	64
A.1	Architectures for the classifier $C$ (see Figure 4 in Chapter 3). . . . .	79
A.2	Architectures for the generator $G$ (see Figure 4 in Chapter 3). . . . .	79
A.3	Architectures for the discriminator $D$ (see Figure 4 in Chapter 3). . . . .	80
A.4	Training on the shift noise: we evaluate the accuracy on <i>clean training set</i> at the end of each epoch. Despite ResNet models are deeper and more resistant to uniform noise (93), they are not robust against shift noise. Indeed, accuracy on the <i>noisy training set</i> reaches about 100% pointing out that the models overfit noise. . . . .	80
A.5	MNIST samples generated by $G$ , trained with Algorithm 1 (SVHN $\rightarrow$ MNIST split). Each row is related to a different label code (from <i>top</i> to <i>bottom</i> , 0 to 9). . . . .	82
A.6	SVHN samples generated by $G$ , trained with Algorithm 1 (MNIST $\rightarrow$ SVHN split). Each row is related to a different label code (from <i>top</i> to <i>bottom</i> , 0 to 9). . . . .	83

A.7	MNIST-M samples generated by $G$ , trained with Algorithm 1 (MNIST $\rightarrow$ MNIST-M split). Each row is related to a different label code (from <i>top</i> to <i>bottom</i> , 0 to 9). . . . .	84
A.8	USPS samples generated by $G$ , trained with Algorithm 1 (MNIST $\rightarrow$ USPS split). Each row is related to a different label code (from <i>top</i> to <i>bottom</i> , 0 to 9). 85	85
B.1	Description of the architectures (classifiers and statistics networks) for the experiments on Digits ( <i>left</i> ), IMDB ( <i>right</i> ). . . . .	87
B.2	Description of the architectures (classifiers and statistics networks) for the experiments on German ( <i>left</i> ), Adult ( <i>right</i> ). . . . .	88
B.3	Training (cross-entropy) loss ( <i>left</i> ) and training accuracy ( <i>right</i> ) with $\lambda = 1.0$ for different number of iterations of MINE ( $K$ ) on the digit recognition task (setting $\sigma = 0.02$ ). An increased number of iterations ( $K = 20, 40, 80$ in <i>blue, orange</i> and <i>green</i> , respectively) has the effect of stabilizing the training procedure, it allows the model minimizing the loss function and fitting the training data. The charts report the average of 3 runs. . . . .	88
B.4	Values for mutual information (left column), test accuracy (middle column) and train accuracy (right column). We accounted for the different color, modelled by different $\sigma$ (check Section 4.5 of Chapter 4), and here represented by different rows. It is visible how a decrease in the (estimated) mutual information correlates with an improved performance. . . . .	89
C.1	Test accuracy for unbiased samples (green) and full set of samples (red) for different values of $\zeta$ . The $x$ axis is in logarithmic scale. For $\zeta = 0.0$ we have the weighted ERM of Eq. 5. .	92

# List of tables

3.1	<i>Left:</i> shift noise for the split MNIST→SVHN: only 30% of the labels are correctly inferred on SVHN after training a classifier on MNIST; high degree of asymmetry (values refer to the training sets). <i>Mid-left:</i> the confusion matrix, accuracy and $\delta_A$ for a classifier trained on noisy SVHN almost reflect the initial ones, meaning that shift noise was overfitted. <i>Mid-right</i> Oracle performance on samples generated by a cGAN trained with shift noise. Not only generated images are classified better than training samples, but also residual noise is less structured (lower $\delta_A$ ). <i>Right:</i> A classifier is trained on (cleaner) cGAN-generated samples: its accuracy is slightly higher than the one of the classifier directly trained on shift noise, but more importantly inferred labels show a consistently lower amount of structure in their noise. Results are averages over 5 runs, starting from the same shift noise and accuracies refers to the training sets. . . . .	22
3.2	Classifiers tend to overfit shift noise, reflecting initial accuracy and asymmetry of the shift noise itself. They are instead significantly robust to an equivalent (in term of number of corrupted labels) amount of uniform noise $n$ (eq. 3.1). . . . .	23
3.3	cGANs trained with shift noise show robustness in generating clean samples (GAN-test), according to an oracle classifier. At the same time, they produce samples with enough variability and quality: in fact a classifier trained on generated samples outperforms a classifier trained directly on shift noise (Table 3.2) both in term of accuracy and noise uniformity. . . . .	23
3.4	Comparison between our method (Pseudo-Label Refinement - PLR) with different GAN objectives and competing algorithms. Test-set accuracies are the results of averaging over 3 different runs. (*) Uses extra SVHN data (531, 131 images). (**) Uses 1,000 target samples for cross-validation. . . .	30



4.1	<b>Digit experiment – comparison with related work.</b> Experimental results on colored digit classification for different levels of variance ( $\sigma$ ) in the color distribution. The first row reports results related to models trained via standard Empirical Risk Minimization (ERM). Below we report results obtained by competitor methods (4; 53; 81). The last row reports results achieved with our method (with $\lambda = 1.0$ ). . . . .	45
4.2	Experimental results from IMDB gender classification problem. The first row reports results obtained by setting $\lambda = 0.0$ (ERM baseline). The last row reports results obtained with our method ( <i>Ours</i> ); Each column reports results associated with the indicated test set. . . . .	46
4.3	<b>Fairness experiments</b> – We compare against results on the two datasets as reported in (30; 74; 77). For accuracy, the higher the better. For EO, the lower the better ( <i>i.e.</i> , the “fairer”). Our results were averaged across 10 different runs. . . . .	47
5.1	<b>Accuracy on whole test set.</b> Accuracy (in %) evaluated on biased + unbiased test samples for different bias ratios. Best performance in bold. . . . .	61
5.2	<b>Results on unbiased test samples.</b> Accuracy (in %) evaluated only on the unbiased samples for different bias ratios. Best performance in bold. . . . .	61
5.3	Performance on the whole (avg) and unbiased only test set: comparisons with baseline, unsupervised and supervised methods (see text). . . . .	63
5.4	<b>Ablation analysis on the augmentation strategies.</b> We report the accuracy resulting from different augmentation strategies and no augmentation, by varying the bias ratio. Our strategy results the winner over all the other mixing policies. . . . .	65
C.1	<b>Ablation study on bias identification.</b> F1 score of $\hat{\mathcal{D}}_{bias}$ , $\hat{\mathcal{D}}_{unbias}$ compared to ground truth annotations $\mathcal{D}_{bias}$ , $\mathcal{D}_{unbias}$ obtained with our Bias Identification strategy (Eq.2), using an oracle and with subsets generated randomly. We report the final test accuracy on the whole test set for the three different cases. . . . .	92

# **Part I**

## **Rationale and Background**

# Chapter 1

## Introduction

### 1.1 Overview

Deep neural networks are powerful and expressive tools that can learn non-linear data representations to perform different tasks. They allow to compute data descriptors (features) in a data-driven way, replacing the need for hand-crafted features computed by domain experts. Features learned by neural networks encode the information needed to solve a task, typically in a lower dimensional space which makes the downstream task computationally tractable.

The representation learning paradigm has been applied to large scale, complex data, leading to unprecedented results in different areas of Machine Learning such as Computer Vision, Natural Language Processing or Reinforcement Learning.

The implicit assumption is that training, validation and test data are sampled from the same distribution. This hypothesis rarely holds in practice, since data used for training a model might have been acquired in different conditions with respect to the ones that a model encounters when deployed. Such differences in the data distributions (distributional/domain shifts) severely harm the model performance and prevent from a correct generalization. In the context of computer vision, some examples of distributional shifts are different illumination conditions, style of the image, poses, etc. (see Figure 1.1). For example, if we consider a scene categorization task, *e.g.*, a self-driving car, some examples of distributional shifts are different weather conditions or unexplored environments (*e.g.*, urban vs. desert vs. country).

The distributional/domain shift issue is intimately related to the dataset bias problem. Unavoidably, every dataset has incomplete or biased information towards some specific distribution/domain. In relation with the example above, when one trains a module for a self-driving car, it is not possible to record all existing streets under all the possible light and

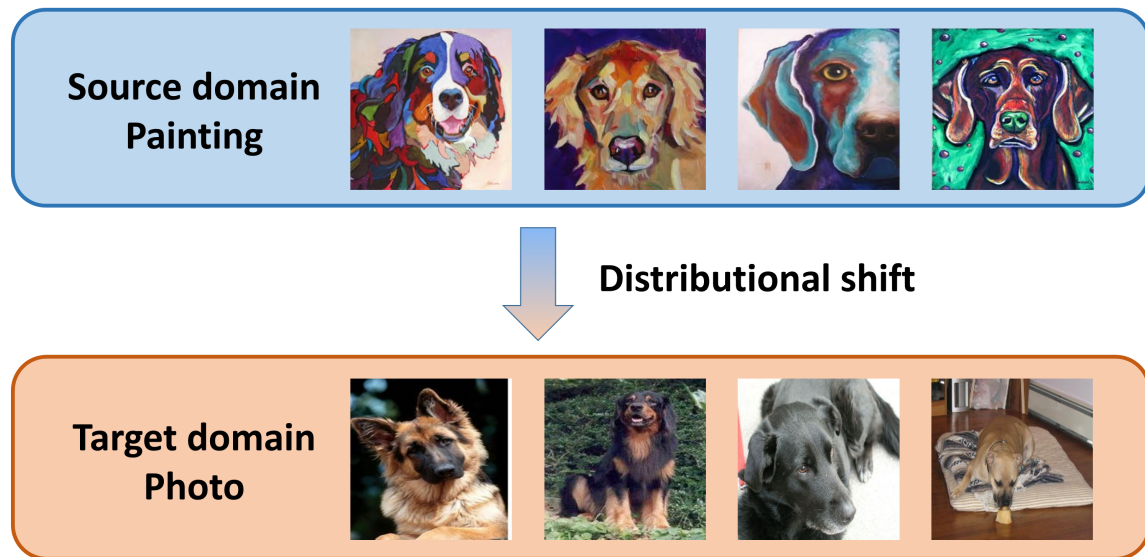


Figure 1.1 Example of distributional shift from Painting domain to Photo domain (images from the PACS dataset (64)).

weather conditions, therefore one has access only to a limited set of domains (*e.g.*, European city streets with daylight).

The set of attributes that makes one dataset unique has been called Dataset Bias (116). Such attributes may be learned during training, leading the models to rely on them for the downstream tasks. In this case, irrelevant or even harmful information is exploited by the model to solve its task, preventing it from a correct generalization. In fact, even if the model predicts correctly held-out samples from the training distribution, it may fail when such samples are drawn from another distribution.

Standard regularization techniques are in fact designed to improve model generalization on unseen data from the same distribution. Although very effective to achieve their goal, they are not sufficient to address distributional shift issues, which should be faced by tailored solutions.

Domain Adaptation (DA) considers a scenario in which one or multiple source distributions (source domains) and a target domain are available during training. Specifically, source domains samples are provided with labels, while target domain samples may come fully labeled (Supervised DA), partially labeled (Semi-supervised DA) or completely unlabeled (Unsupervised DA). Having access to samples from the target distribution is beneficial, even when no supervision is provided. In this thesis, we focus only on the Unsupervised Domain Adaptation (UDA) problem. Several techniques have been designed to learn data representation that take into account both the source and the target domains. Approaches

rely on adversarial training to learn domain invariant representations(32; 33; 117), alignment of source and target data distributions in the feature space (80; 112), translation from one domain to another (84; 97; 113). We give a detailed overview of existing methods in Chapter 2.

The main drawback of Domain Adaptation is that it assumes the target distribution (even if unlabeled) to be known a priori. This is equivalent to know in advance what are the conditions in which a model is deployed. While this is a sensible assumption in some cases, *e.g.* a robot working in a controlled environment, it may be a severe limitations in others, *e.g.* a self-driving car that operates in unpredictable weather conditions.

Out-of-Distribution (OOD) generalization problem addresses this challenging setting where the testing distribution is unknown and different from the training. A subfield of OOD generalization is Domain Generalization (DG): DG considers the supervised learning setting where one or more source domains are available at training time and the model has to generalize to unseen distributions, never observed. In practice, the target data are used only when testing the model.

In this thesis we face the problem of *learning from biased* data that requires generalization to unseen domains. In this setting, in the training set there are spurious correlations between domain attributes and task labels, *e.g.*, most fish images are set in a marine environment, hence the background is a strong cue to predict the label. When presented with such data, neural networks may learn such attributes as predictive for the class, thus failing to catch the real semantic of a category (fish shape). This is not a problem as long as we deploy the model on a test distribution with the same characteristics; nevertheless when we test it on different distributions, such spurious correlations may not hold and so the model may fail at generalizing. This situation may be even worse when the spurious correlations associated with one class at training time, are associated with another one at test time: this may lead the model to make wrong predictions with high confidence. The bias problem stems from the fact that most of the samples of a category are observed under the same domain (most fishes observed in water), whereas few or even no samples are outside of this distribution (fish in other contexts, *e.g.*, fish on a market stand). This makes the dataset highly imbalanced: one deals with many samples that bring little information (low intra-class variance) and very few samples that are very informative (high intra-class variance). In this context the representation learning paradigm has to face unique challenges to effectively learn from such data.

Learning from biased data has been addressed in two settings, namely *supervised debiasing* and *unsupervised debiasing*. In the former case, there is a ground truth label for the spurious correlations affecting the data (besides the task labels). In the latter, one is provided with only task labels and has to understand directly from data what form of bias is present, if any. Having an annotation for the bias can be used to learn models that are invariant to such attributes, achieving a debiased model. The unsupervised case instead is more challenging, but at the same time, more realistic: collecting annotations regarding the type of bias affecting the data is expensive or even impossible, whenever a control on the data collection process is not viable. Therefore, it is of utmost importance to design techniques that directly infer the bias information directly from data and subsequently account for spurious correlations in the data.

## 1.2 Contributions

In this section, the contributions brought by this thesis are introduced. First, we developed a novel algorithm to perform Unsupervised Domain Adaptation exploiting pseudo-labeling. Second, we face the debiasing problem and propose two methods, one to tackle it in the supervised scenario and another for the unsupervised case.

**Unsupervised Domain Adaptation by GAN-based pseudo-label cleaning.** We investigate and characterize the inherent resilience of conditional Generative Adversarial Networks (cGANs) against noise in their conditioning labels, and exploit this fact in the context of Unsupervised Domain Adaptation. In UDA, a classifier trained on the labelled source set can be used to infer pseudo-labels on the unlabelled target set. However, this will result in a significant amount of misclassified examples (due to the domain shift issue), which can be interpreted as noise injection in the ground-truth labels for the target set. We show that cGANs are, to some extent, robust against such “shift noise”. Indeed, cGANs trained with noisy pseudo-labels, are able to filter such noise and generate cleaner target samples. We exploit this finding in an iterative procedure where a generative model and a classifier are jointly trained: in turn, the generator allows to sample cleaner data from the target distribution, and the classifier allows to associate better labels to target samples, progressively refining target pseudo-labels. Results on common benchmarks show that our method performs better or comparably with the unsupervised domain adaptation state of the art.

**Supervised debiasing by Mutual Information.** We tackle the problem through the lens of information theory. We focus on Mutual Information as a measure of dependency between random variables. Mutual Information utterly quantifies the dependencies between the variables - not just the linear ones: it is zero if and only if the variables are jointly independent. Furthermore, Mutual Information does not change under invertible transformations (re-parameterizations) of such variables. These properties make Mutual Information the adapt measure for the analysis of the dependencies between real-valued non linearly-related variables, as it is common in computer vision problems. We argue that a good strategy to face the debiasing problem is minimizing the mutual information between the learned representation and the biased attributes. This would result in a data representation that is statistically independent from the specified bias, and that, in turn, would generalize better. We propose a novel end-to-end optimization strategy, leveraging recent findings for a differentiable estimator of Mutual Information, which simultaneously estimates and minimizes the mutual information between the learned representation and specific data attributes.

When applied on standard benchmarks, our model shows comparable or superior classification performance with respect to state-of-the-art approaches. Moreover, our method is general enough to be applicable to the problem of “algorithmic fairness”, with competitive results.

**Unsupervised debiasing by Meta-Learning.** We tackle the problem from a meta-learning perspective. Considering the dataset as composed of unknown biased and unbiased samples, we first identify these two subsets by a pseudo-labeling algorithm, even if coarsely. We exploit the fact that biased samples are learned more easily by a neural network, while unbiased samples are more problematic, to split the dataset. Subsequently, we apply a bi-level optimization algorithm in which, in the inner loop, we look for the best parameters guiding the training of the two subsets (biased and unbiased samples), while in the outer loop, we train the final model taking benefit from augmented data generated using Mixup (129). Properly tuning the contributions of biased and unbiased data, together with the regularization introduced by the mixed data has proved to be an effective training strategy to learn unbiased models, showing superior generalization capabilities. We highlighted how such model achieves high accuracy on unbiased samples while maintaining good performance even on samples affected by the bias. Experimental results on synthetically and realistically biased datasets surpass state-of-the-art performance, as compared to existing methods.

## 1.3 Publications

We report hereafter the list of publications and works under review, which will be illustrated in the remainder of this thesis.

- Pietro Morerio, Riccardo Volpi, **Ruggero Ragonese**, Vittorio Murino, Generative Pseudo-label Refinement for Unsupervised Domain Adaptation; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 3130-3139 (Chapter 3).
- **Ruggero Ragonese**, Riccardo Volpi, Jacopo Cavazza, Vittorio Murino, Learning Unbiased Representations via Mutual Information Backpropagation; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2021, pp. 2729-2738 (Chapter 4).
- **Ruggero Ragonese**, Valentina Sanguineti, Jacopo Cavazza, Vittorio Murino, Learning to learn Unbiased Models from Biased data. *Under review* (Chapter 5).

## 1.4 Summary

The rest of this thesis is organized as follows. In Chapter 2, the literature related to this work is illustrated. In Chapter 3, we introduce the Generative pseudo-label refinement procedure to perform UDA, and we validate it using several benchmarks. In Chapter 4, the supervised debiasing problem is addressed: the Mutual Information Neural Estimator is discussed and the architecture we propose is illustrated. We show experimental results on several benchmarks that include computer vision and algorithmic fairness. In Chapter 5, we present the method to perform unsupervised debiasing. We detail the two stages of our approach (bias identification and learning unbiased representations), and show empirical results on several datasets, both synthetic and realistic. Finally, in Chapter 6, we draw the final remarks and future work.



# Chapter 2

## Background and Related Works

In this chapter we review the existing literature on the problems we tackle, namely domain adaptation, domain generalization with a special focus on learning from biased data.

### 2.1 Domain Adaptation

The problem of leveraging one or multiple data distributions (source domains)  $P_{source}$  to train a model to perform well on a distribution of interest (target domain)  $P_{target}$  has been explored in the domain adaptation literature. The problem may be tackled in different scenarios:

- **Supervised Domain Adaptation.** At training time we are provided with a set of labeled source domains samples  $\{x^i, y^i\}_{i=1, \dots, m} \sim P_{source}(X, Y)$  and some labeled target domain samples  $\{x^i, y^i\}_{i=1, \dots, n} \sim P_{target}(X, Y)$ , usually with  $m \gg n$ .
- **Semi-supervised Domain Adaptation.** At training time we are provided with a set of labeled source domains  $\{x^i, y^i\}_{i=1, \dots, m} \sim P_{source}(X, Y)$ , some labeled target domain samples  $\{x^i, y^i\}_{i=1, \dots, n} \sim P_{target}(X, Y)$ , usually with  $m \gg n$  and some unlabeled target domain samples  $\{x^i\}_{i=1, \dots, k} \sim P_{target}(X)$ .
- **Unsupervised Domain Adaptation.** At training time we are provided with a set of labeled source domains  $\{x^i, y^i\}_{i=1, \dots, m} \sim P_{source}(X, Y)$  and an unlabeled target domain samples  $\{x^i\}_{i=1, \dots, k} \sim P_{target}(X)$ .

We focus here on the case of Unsupervised Domain Adaptation (UDA) which is the problem that we face in Chapter 3. We review existing approaches for the problem.

### 2.1.1 Unsupervised Domain Adaptation

Unsupervised Domain Adaptation (UDA) has been tackled in recent years using deep neural networks. Several methods employ the Maximum Mean Discrepancy (39) as a distance measure between source and target domains.

Tzeng et al. (118) introduce a domain loss, which is the MMD distance between feature representations of source and target domains. By jointly minimizing the task loss and the domain loss, one learns the task of interest while achieving domain confusion, which results in more general representations. Long et al. (71) propose Domain Adaptation Networks which match hidden representations of source and target domains using Reproducing Kernel Hilbert Spaces. A shared backbone is employed for both source and target, while high level features are domain specific and hence adapted using a Kernel variant of MMD proposed by Gretton et al. (39). Ganin & Lempitsky (32) introduce Domain Adversarial Neural Network (DANN) that achieves adaptation in an adversarial learning fashion. A domain classifier is “fooled” by inverting its gradients while training it. In this way one achieves domain invariance. Chopra et al. (23) define an interpolation path between source and target domains and learn intermediate representations along this path. These intermediate features are then combined to be used by a downstream classifier/regressor. Rozantsev et al. (95) avoid to use the same neural architecture for both source and target, i.e. using weight sharing, and propose a two streams model. In such model the weights in each layer are not shared across the two streams, but there is a constraint not to make them too different from each other. Sener et al. (104) propose a method that achieves a twofold goal: adaptation to reduce the domain shift and transduction to label unsupervised data points. The problem is made tractable by exploiting two heuristics regarding feature consistency. Bousmalis et al. (16) propose to learn data representations with features shared across domains and features which are domain specific. In this way one achieves a generic representation without giving up features that represent the domain’s peculiarities. Saito et al. (100) propose an asymmetric training procedure in which pseudo-labels are assigned to the target domain data. Two networks are trained on the source domains and then used to assign pseudo-labels and a third network is trained on such pseudo-labels. Haeusser et al. (42) introduce an associative loss which enforces similarity between source and target features to achieve domain invariance. Such loss is minimized together with a classification loss, that learns class discriminative features. Morerio et al. (80) and Sun & Saenko (112) propose to align features of source and target along the second order statistics while training the network to solve the classification problem. Saito et al. (101) propose an adversarial Dropout(109) procedure that learns more discriminative features for the target domain by encouraging samples to be far from the decision boundary. Shu et al.

(108) investigate the “cluster assumption”, namely decision boundaries should not cross high density data regions, to propose two algorithms (VADA and DIRT-T) that alleviate some issues in adversarial training for UDA. Mancini et al. (75) relax the assumption of having source domains as composed by samples from a single distribution and propose a method to discover sub-populations that can be treated as (sub-)domains. Such information is later exploited to learn more general data representation. Pinheiro (89) proposes a method based on similarity learning, namely for each class a prototype is computed and target domain labels are assigned based on the distance of an (unlabeled) sample and the different prototypes. Kang et al. (50) propose to use attention alignment to achieve adaptation. Specifically they propose a method to align the attention of source network with that of a target network, to have the same discriminative regions in both source and target. Damodaran et al. (27) employ Optimal Transport to reduce the discrepancy between source and target while preserving at the same time the information used by the classifier.

After the breakthrough represented by generative models like Generative Adversarial Networks (GAN’s) (37) and Variational Autoencoders (VAE’s) (56), several methods have approached UDA under an image-to-image translation paradigm, in which source images are generated under the style of target ones and vice versa.

Taigman et al. (113) introduce Domain Transfer Network (DTN) which generates images in the style of the target domain, keeping the content as in the original input. By using a constraint, images from source are translated into the target domain, while target samples are mapped into themselves. Liu & Tuzel (70) propose Coupled GAN, a specific GAN that can learn the joint distribution from few samples from the marginal distributions. This model is then used for image-to-image translation to tackle UDA, considering source and target as the marginal distributions. A follow-up work (69) further improves Coupled GAN by imposing a shared latent space between different domains. Bousmalis et al. (16) propose to train a GAN to render source domain images into the style of target domain. The overall model trains with a GAN loss and a task specific loss, e.g. classification loss. Russo et al. (97) train GAN’s in a bi-directional fashion to be able to transform source images into target ones and vice versa. To do so, they define a class consistency loss that aligns the generators in the two directions. Murez et al. (84) learn adapted features by adding extra networks and losses to the main architecture. Specifically, the features produced by an encoder are able to generate images in both source and target. Additionally, features from the two domains are imposed to be indistinguishable.

## 2.2 Out-of-Distribution generalization

Out-of-Distribution generalization (OOD) problem addresses the challenging setting where the testing distribution is unknown and different from the training one. Domain generalization (DG) considers the problem where one or multiple source domain(s) are available at training time, yet one has no access to any target domain sample: for this reason the problem is also stated as *generalizing to unseen domains*.

Learning from biased data is a special case of OOD in which training data exhibit spurious correlations that do not necessarily hold on the test distribution. Hence, the need to learn data representations that do not encode information that may be predictive at training time but harmful at test time.

When additional labeling regarding the bias is given, one can exploit this annotation to solve the task of interest without incorporating the bias information. We termed this problem **supervised debiasing**. However, a more general scenario is to deal with biased data when bias annotation is not available: we termed this problem **unsupervised debiasing**. In the latter case, the learning algorithm has to infer directly from data the information about the bias that affects the training data.

In Sec.2.2.1 we revise the first works on Domain Generalization and then the two families of approaches most related to this thesis, namely data augmentation based methods and meta-learning approaches. For a more detailed analysis of the existing literature, readers can refer to (124). Following, we detail the existing literature of learning from biased data in the cases of supervised (Sec.2.2.2) and unsupervised (Sec.2.2.3).

### 2.2.1 Domain Generalization

Khosla et al. (52) propose to learn the biases of each domain in the form of a weight and then “undo” them, compensating their contribution in the output of a model. They train a Support Vector Machine (SVM) (25) with such weights and show how the final accuracy increases compared to a baseline model. Xu et al. (127) learn an ensemble of classifiers and predict at test time which one to use based on the nuclear norm as a distance metric. Muandet et al. (83) propose Domain-Invariant Component Analysis (DICA), a kernel based optimization method that aims at learning an invariant representation across multiple domains. They show that reducing dissimilarity between domains increases the expected generalization capability. Ghifary et al. (35) introduce Multi-Task Autoencoder (MTAE), inspired by denoising autoencoders (119). Such autoencoder is made by a shared encoder and multiple

decoders, one for each domain. By learning to translate one image into other domains, the latent features are made robust towards domain shift, improving the model generalization capabilities. Li et al. (64) extend “undo bias” (52) to the neural network context and propose an end-to-end optimization procedure for training.

A body of literature tackles DG from a data augmentation perspective: augmenting the dataset can increase its diversity, providing models with additional samples. Such augmented samples can be generated in order to enhance the diversity of data while assuring their reliability. Shankar et al (105) propose Crossgrad to learn adversarial perturbations to be added to the original samples. In this way the model robustness towards domain shift is improved.

Volpi et al. (122) propose an adversarial data augmentation procedure that only require a single source domain. Samples are perturbed in the semantic space under a worst case scenario to increase the generalization. Volpi et al. (121) propose to sample image transformations from a pre-defined set and apply to the training data, in order to find the combination that causes the model to fail. Such generated samples are then subsequently employed to fine-tune the model to improve its robustness. Zhou et al. (132) introduce Domain Transformation Network (DoTNet) which transforms images to unseen domains, keeping class properties. Original and newly generated images are then used to train a more general model.

Other approaches tackle DG from a meta-learning perspective, devising methods inspired by Model Agnostic Meta-Learning (MAML) (31). In such methods, a bi-level optimization is performed: in the inner loop, a set of meta-parameters is optimized on some meta-training data. In the outer loop, the network parameters are updated on a meta-validation set, computing a gradient through a gradient. Meta-Learning has been initially devised to learn efficiently from scarce data, hence it has been successfully applied in few-shot learning problems. Nevertheless, different works have applied it to the problem of Domain Generalization. For a more complete overview on the topic, the reader can refer to (47).

Li et al. (65) cast the problem of learning from multiple domains as a multi-task problem, in which source domains are treated as meta-tasks while one held-out source domain is used as a meta-validation. The optimization steps are performed in order to keep low loss values for both meta-train and meta-validation domains, so that a better generalization is achieved. In a follow-up work (67), it is proposed to meta-learn a regulariser to help train a feature extractor to be domain invariant. The regularizer is itself a neural network whose parameters are updated in the inner loop. A similar approach is conducted by Balaji et al. (9), where a

Meta-Regularizer (MetaReg) learns the optimal parameters for a weight decay regularizer to be added to the task loss.

### 2.2.2 Supervised Debiasing

The problem of removing information about a specific attribute from data representation has been explored in recent years.

An important family of methods rely on adversarial training. Beutel et al. (14) attempt to learn data representations that are independent from a protected attribute, so to achieve a prediction that does not rely on such information. They use gradient reversal layer (32) to fool a classifier devised to predict the protected attribute from the feature representation. Zhang et al. (128) further extends this work, by placing an adversarial module on top of the predictions of the main network. The adversarial module has to predict the protected attribute from the inferred labels. Alvi et al. (4) propose to use a confusion loss (118) along with the task loss for training a debiased model. Specifically, they aim at making the prediction of the protected attribute close to a uniform distribution by means of the confusion loss. Kim et al. (53) formulate the problem of learning unbiased representations as a constrained optimization problem where mutual information (106) is used to measure the statistical dependence of the data representation with respect to the protected attribute. The optimization problem is then made tractable by introducing approximations for the mutual information and relying on the gradient reversal layer, similarly to (14).

Li and Vasconcelos. (66) propose the REPresentAtion bIas Removal (REPAIR) procedure to mitigate the bias in the data representation. The optimization problem seeks a weight distribution that penalizes “easy samples” for a classifier built on a given feature representation. Such weights help the model to remove the bias from the data representation.

Another family of approaches rely on Variational Inference models (56) to impose a prior on the data representation. Luizoz et al. (72) extend Variational Autoencoders to encourage independence between latent space and the protected attribute. Additionally, they add a penalty term based on the Maximum Mean Discrepancy (MMD) to remove any unwanted information left. Moyer et al. (81) cast the problem of invariant representation learning as an information theoretic objective, similarly to (53). They propose a tractable solution for it, inspired by the Information Bottleneck principle (115).

Sagawa et al. (99) reduce bias in the learned representation relying on robust optimization (12). A worst case objective is formulated, in which one seeks to optimize for the group

(represented by the pair class/domain) that has the highest loss value. In practice, such optimization leads to compute sample weights, resembling other importance weighting approaches (20).

### 2.2.3 Unsupervised Debiasing

Few recent works attempted to solve the debiasing problem in the unsupervised case, namely without requiring any information regarding the type of bias affecting the data.

**Ensembling approaches.** Bahng et al. (8) formalized the cross-bias generalization problem and proposed a method (ReBias) to tackle it. Cross-bias refers to the scenario in which non essential cues to recognize a class are present in the training data but not necessarily in the test data. ReBias trains a model that is biased by design towards texture and colors: in practice the receptive field of the convolutional layers are smaller so that the feature maps focus mostly on low level patterns such as colors/textures. Provided with such biased model, a debiased model is trained by pursuing statistical independence between the embeddings of the former and the latter. Statistical independence is achieved by using the Hilbert Schmidt Independence Criterion (HSIC) (40) so that the debiased model seeks for alternative ways to predict the target with respect to the biased model.

Nam et al. (85) showed that samples which exhibit spurious correlations are usually learned faster than those that do not. They consider the training data as composed by “bias aligned” samples which are more numerous and have shortcuts, and “bias conflicting” samples which are sparse and do not follow any particular shortcut. To tackle the unsupervised debiasing, they train a model to be biased (similarly to (8)) by using the Generalized Cross-Entropy (GCE) loss (130). GCE loss is lower for samples which are already correctly predicted, hence a model trained in this way amplifies the bias present in the data. A second model is trained to be debiased: the Cross Entropy values of each sample in both model are cast into weights that modulate the learning rate for each data point: bias conflicting samples are learned slower than the bias aligned ones, hence the debiased network can learn more general data representation.

**Sample reweighting.** Liu et al. proposed Just Train Twice (JTT) (68) to tackle unsupervised debiasing with a two-stage approach. First they split the training data into two subsets which resemble ideally the bias aligned and bias conflicting samples: a model is trained via Empirical Risk Minimization for  $T$  epochs and thereafter the samples that correctly predicted are assigned to one subset while those incorrectly predicted are assigned to the other one.

Secondly, samples which have been wrongly classified are then upsampled to rebalance the dataset.

**Data augmentation.** Kim et al. (54) propose a two-stage approach (similarly to (68)) to identify the biased and unbiased samples and, subsequently, use such information to learn more general data representations. In the first stage, they train a model via Empirical Risk Minimization and combine information about the training accuracy and the confidence on the softmax predictions. Samples which are wrongly classified with high confidence are likely to exhibit some spurious correlation. The feature learning part involves a data augmentation step where patches of biased and unbiased samples are swapped so to generate more diverse data. the newly generated samples are then attached to the initial training data and a debiased model is trained on them.



**Part II**

**Contributions**

# Chapter 3

## Generative Pseudo-Label refinement for Unsupervised Domain Adaptation

### 3.1 Context

Unsupervised Domain Adaptation (UDA) addresses the problem of learning models that perform well on a *target* domain for which ground truth annotations are not provided. During the training phase, one can leverage unlabeled samples from this distribution and labelled samples from a *source* distribution, separated by the so-called *domain shift* (116), *i.e.*, drawn from two different data distributions. In this chapter, we address UDA from a novel perspective, by casting the problem in the setting of *learning with noisy labels* (93). We start from the very simple realization that, given a model trained on the source domain, we can infer a set of (pseudo-)labels for the target domain. Typically, due to the domain shift, a consistent number of labels are wrongly inferred, resulting in a noisy-labelled target set, with an amount of noise proportional to the classification error. Previous work (93) has shown that deep learning-based classifiers are robust against label noise, provided a sufficiently large training set. For this reason, one might hope that such resilience could be exploited to train a classifier for the target domain with the supervision of the inferred, noisy pseudo-labels. The idea is that the classifier might disregard label noise to some extent, providing target accuracy higher than the original model trained on source samples.

However, we empirically show that such strategies alone cannot compete with existing UDA methods in terms of accuracy on the target set. Indeed, while deep models' robustness against noisy labels is remarkable when the noise distribution is nearly uniform, we show that they are not robust against the label noise resulting from the domain shift. Although *a priori* assumptions cannot be made on such noise, we empirically observe in a variety of

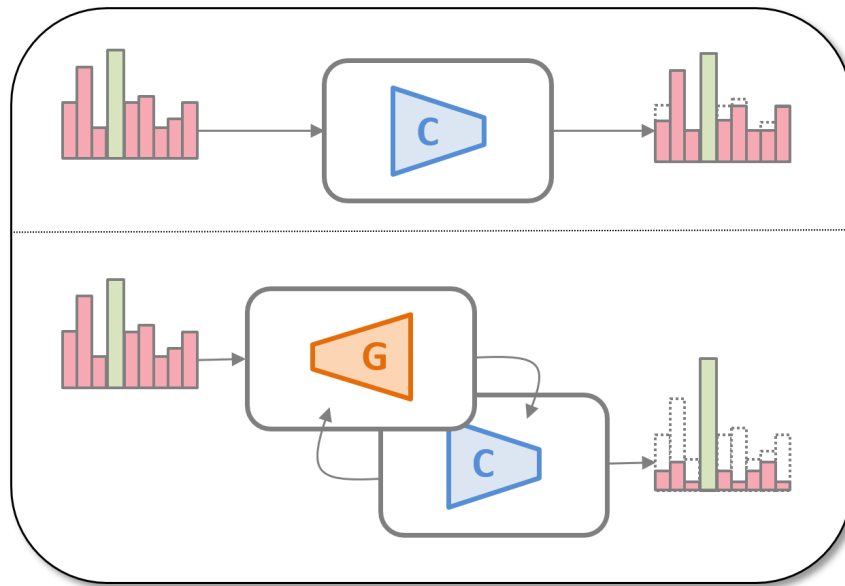


Figure 3.1 *Top*: a neural network classifier  $C$  is typically not robust against shift noise, here represented by an histogram. *Bottom*: a cGAN generator  $G$  is able to filter such structured noise, making it more uniform and thus tolerable from the classifier. By jointly training  $C$  and  $G$ , the former benefits from “cleaner” generated data, while the latter from more accurate inferred labels.

adaptation problems heavy deviations from uniform noise, meaning that misclassifications are not evenly distributed across classes. We term such highly structured noise “*shift noise*”.

While classifiers are not robust against such more structured kind of noise, we observe that conditional Generative Adversarial Networks (78) (cGAN) are. A cGAN model can be trained to generate samples conditioning on the desired classes of an arbitrary distribution. It was shown that this class of models can be *made* more resistant against label noise (114), but we provide empirical evidence that—to some extent—they are inherently robust against it, without any modification from the standard training procedure (37; 78). This means, in practice, that training a cGAN on some noisy-labelled samples will result in a model that generates samples that are “cleaner” than the training ones. A natural idea that follows this finding, is trying to generate cleaner target samples to train better performing models for the target domain. However, although cGANs are to some extent resistant to noisy-labels, they are not robust enough to generate target samples that allow to train competitive models.

Interestingly though, we observe that, even if the noise reduction is not sufficient to train competitive target models, *the labels of the generated samples obey a noise distribution which is closer to the uniform than the shift noise one.*

In this chapter, we explore the two facts above (classifier robustness against uniform noise (93) and cGAN robustness to shift noise), and jointly exploit them for UDA. We devise a UDA strategy based on the properties of both classifiers and cGANs to filter out noise in the labels. We propose an iterative procedure, where we alternately optimize the losses associated with a cGAN and with a classifier. Throughout the training phase, the classifier can benefit from more and more reliable conditionally-generated data, while a cGAN can exploit more and more reliable pseudo-labels inferred by the classifier (see Figure 3.1). Source samples are only exploited to train an initial classifier. After this step, the problem is faced in a fully unsupervised fashion, reducing the noise on the labels of the empirical target distribution over iterations during training. Results on standard UDA benchmarks show the effectiveness of our approach.

## 3.2 Related works

**Unsupervised Domain Adaptation.** In UDA, we are given a set of samples from a source distribution in the form  $\{x_s, y_s\} \sim p_{source}$ , and a set of samples from a target distribution of interest in the form  $\{x_t\} \sim p_{target}$  (no labels). The goal is to perform well on data from the target distribution. Different approaches allow to solve this problem efficiently in a plethora of tasks. Adversarial training has been effectively used to map source and target samples in a common feature space (32; 33; 117; 120). Other works aim at aligning the second order statistics of source and target features (80; 112). More recently, image-to-image translation methods, that learn the mapping from the source space to the target one and vice-versa, have been proposed (17; 46; 69; 70; 97; 103; 113). In general, one can design models for UDA that leverage labeled source samples that are “rendered” with the style of target samples (and vice-versa). Other works propose different successful solutions to face the adaptation problem (*e.g.*, (18; 41; 100; 101; 102; 104)). Since the latter are only related to our work for the common goal, they are not detailed in this section.

Our approach is somehow related to image-to-image translation methods, since we exploit generated samples to train a classifier for the target domain. In particular, PixelDA (17) is the most related method, since it leverages a training procedure where a GAN and a classifier are jointly trained. However, the latter makes a strong assumption on the relationship between source and target domains: “*the differences between the domains are primarily low-level (due to noise, resolution, illumination, color) rather than high-level (types of objects, geometric variations, etc)*”. In this work, we generate target images using a simple cGAN, namely mapping noise vectors from a latent space into the image space, merely conditioning on label

codes. This difference comes with two main advantages: our architecture and loss functions are much simpler than the ones adopted for image-to-image translation, and we do not have to make such strong assumptions on the gap between the two domains.

Our method is substantially different from most UDA solutions also because we do not need source samples throughout the adaptation procedure, but only to pre-train the model  $M_{\theta_s}$ , used to assign pseudo-labels to target samples. Indeed, solutions that align source/target feature statistics (80; 112), map samples from both distributions in a common feature space via adversarial training (32; 33), or translate images between domains (17; 46; 69; 70; 97; 103; 113), are typically based on objectives that depend on both source and target samples. In our case, the independence from source samples during the adaptation procedure brings a number of advantages. The main one is that the training procedure designed for a certain target can be used *as is*, regardless of the source domain, the only difference being the model  $M_{\theta_s}$  used for the first, initial label inference. Moreover, many adaptation methods require additional hyperparameters to balance different loss terms (17; 32; 33; 69; 97; 103; 113) that depend on both source and target samples. The latter is a huge drawback because in UDA we do not have target labels for hyperparameter cross-validation.

**Learning with pseudo-labels.** Our joint training procedure for UDA is related to the approach by Lee et al. (48). In this work, a method for semi-supervised learning is proposed, where, as training proceeds, inference is performed on unlabeled samples, and the pseudo-labels obtained are interpreted as correct and used for training a classifier. Part of our method has similarities to this idea since, during our training procedure, we infer pseudo-labels for the target samples. However, we are different in that we use them to train a generative model.

**Generative Adversarial Networks.** The original formulation by Goodfellow et al. (37) is defined by the following minimax game between a network  $D$  (discriminator) and a network  $G$  (generator)

$$\min_{\theta_D} \max_{\theta_G} \mathcal{L}_{GAN} = \mathbb{E}_{x \sim p_x} [-\log D(x; \theta_D)] + \mathbb{E}_{z \sim p_z} [-\log(1 - D(G(z; \theta_G); \theta_D))]$$

Solving such optimization problem makes  $D$  classify samples from the data distribution as *real* and samples generated by  $G$  as *fake*. Conversely, it makes  $G$  generate samples that  $D$  would classify as *real*.

A straightforward extension is the concatenation of label codes to the input before it is fed to  $G$  and  $D$ , in order to condition on the class from which data are generated. This extension is termed *conditional* GAN (cGAN, (78)), and represents the class of models this work focuses on, being our method based on class-conditioned image generation.

Several alternatives to the original GAN formulation (37) have been proposed. Two examples are substituting the cross-entropy loss with the least-squares loss (76) or with the Hinge loss (79). Also more elaborated alternatives have been introduced (7; 13; 58). To date, the superiority of one objective function over the others is not fully clear (73), and the main advancements on GAN research have been related to architectural choices (91) and different training procedures (19; 51; 79; 133).

### 3.3 Robustness against label noise

In this section we first formalize the problem and describe the concept of *shift noise*, as the noise resulting from inferring labels in a domain that is different from the training one. Armed with the formal definition, we explore the robustness of ConvNets (60) and cGANs against such peculiar and highly structured noise. Following standardized UDA benchmarks used by the main competing algorithms (17; 69; 70; 97; 113), namely works that rely on GANs to perform adaptation, we train models on MNIST (61) and test on SVHN (86), MNIST-M (32) and USPS (29), we train on SVHN and test on MNIST, and we train on USPS and test on MNIST. For brevity, we define the procedure of training on source and testing on target as *source*  $\rightarrow$  *target* (, MNIST  $\rightarrow$  SVHN). The conclusions we draw in this section will motivate the algorithmic choices we will introduce in Sec. 3.4

#### 3.3.1 Shift noise.

In the UDA setting, given a model  $M_{\theta_s}(x)$ , trained on a source domain  $\mathcal{S} = \{x_s^{(i)}, y_s^{(i)}\}_{i=1}^n$ , we can infer a pseudo-label  $\tilde{y} = M_{\theta_s}(x_t)$  for each target sample  $x_t$ . Mis-classification on the target set will result in a noisy set of pseudo-label associations  $\mathcal{T} = \{x_t^{(i)}, \tilde{y}^{(i)}\}_{i=1}^m$ .

Table 3.1 (first column) provides an example, associated with the split MNIST  $\rightarrow$  SVHN. First of all, we can observe that misclassification noise in the confusion matrices, which we term *shift noise*, is not uniformly distributed across classes. Moreover, shift noise is also different from the structured noise analyzed by Rolnick et al. (93), where the correct label is always assumed to have the highest probability. The only *a priori* assumption we can make concerns the amount of shift noise, which must at least guarantee an accuracy higher

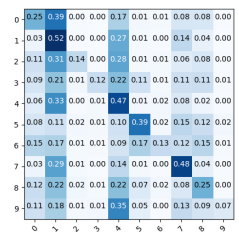
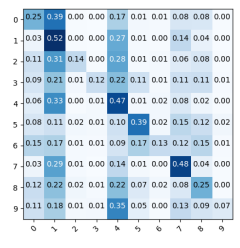
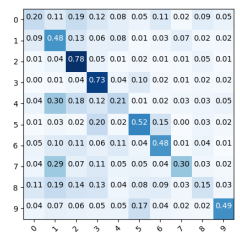
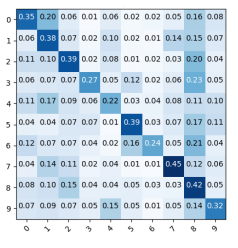
Shift noise	Classifier	GAN-test (107)	GAN-train (107)
			
$a = 0.300$ $\delta_A = 0.374$	$a = 0.321 \pm 0.002$ $\delta_A = 0.374 \pm 0.0001$	$a = 0.419 \pm 0.012$ $\delta_A = 0.252 \pm 0.012$	$a = 0.337 \pm 0.010$ $\delta_A = 0.270 \pm 0.006$

Table 3.1 *Left*: shift noise for the split MNIST→SVHN: only 30% of the labels are correctly inferred on SVHN after training a classifier on MNIST; high degree of asymmetry (values refer to the training sets). *Mid-left*: the confusion matrix, accuracy and  $\delta_A$  for a classifier trained on noisy SVHN almost reflect the initial ones, meaning that shift noise was overfitted. *Mid-right* Oracle performance on samples generated by a cGAN trained with shift noise. Not only generated images are classified better than training samples, but also residual noise is less structured (lower  $\delta_A$ ). *Right*: A classifier is trained on (cleaner) cGAN-generated samples: its accuracy is slightly higher than the one of the classifier directly trained on shift noise, but more importantly inferred labels show a consistently lower amount of structure in their noise. Results are averages over 5 runs, starting from the same shift noise and accuracies refers to the training sets.

than random chance for  $M_{\theta_s}(x)$  on  $\mathcal{T}$ . Note that given an accuracy  $a$  for  $M_{\theta_s}(x_t)$ , the same accuracy can be obtained by injecting *uniform* noise in a fraction  $n$  of the labels:

$$n = (1 - a) \frac{c}{c - 1}, \quad (3.1)$$

where  $c$  is the number of classes. Vice-versa, randomizing a fraction  $n$  of the labels, one would get a fraction of correct predictions (i.e. accuracy) equal to

$$a = 1 - n \frac{c - 1}{c}. \quad (3.2)$$

Note that randomizing a fraction  $n$  of labels does not imply they are all wrong, since, on average,  $1/c$  of them will be assigned the correct class.

As already mentioned, no hypotheses on the shape of the distribution of shift noise can be put forward, making it difficult to characterize it. However, a useful estimate of its amount

Split	Shift noise		Classifier		Equiv. unif. noise		Classifier	
	$a$	$\delta_A$	$a$	$\delta_A$	$a$	$\delta_A$	$a$	$\delta_A$
SVHN → MNIST	0.669	0.208	0.660 ± 0.001	0.241 ± 0.003	0.669	0.017	0.879 ± 0.043	0.029 ± 0.006
MNIST → SVHN	0.300	0.374	0.321 ± 0.002	0.374 ± 0.000	0.300	0.157	0.293 ± 0.006	0.161 ± 0.008
MNIST → MNIST-M	0.550	0.153	0.557 ± 0.003	0.153 ± 0.000	0.550	0.014	0.619 ± 0.002	0.023 ± 0.001
USPS → MNIST	0.608	0.273	0.619 ± 0.001	0.273 ± 0.000	0.608	0.013	0.881 ± 0.036	0.026 ± 0.006
MNIST → USPS	0.819	0.150	0.807 ± 0.002	0.150 ± 0.000	0.819	0.024	0.919 ± 0.006	0.025 ± 0.001

Table 3.2 Classifiers tend to overfit shift noise, reflecting initial accuracy and asymmetry of the shift noise itself. They are instead significantly robust to an equivalent (in term of number of corrupted labels) amount of uniform noise  $n$  (eq. 3.1).

Split	Shift noise		GAN-test		GAN-train	
	$a$	$\delta_A$	$a$	$\delta_A$	$a$	$\delta_A$
SVHN → MNIST	0.669	0.208	0.737 ± 0.018	0.134 ± 0.006	0.725 ± 0.014	0.219 ± 0.007
MNIST → SVHN	0.300	0.374	0.419 ± 0.012	0.252 ± 0.012	0.337 ± 0.010	0.270 ± 0.006
MNIST → MNIST-M	0.550	0.153	0.565 ± 0.057	0.189 ± 0.067	0.536 ± 0.092	0.174 ± 0.068
USPS → MNIST	0.608	0.273	0.772 ± 0.006	0.114 ± 0.005	0.692 ± 0.018	0.239 ± 0.009
MNIST → USPS	0.819	0.150	0.810 ± 0.005	0.135 ± 0.010	0.824 ± 0.005	0.143 ± 0.004

Table 3.3 cGANs trained with shift noise show robustness in generating clean samples (GAN-test), according to an oracle classifier. At the same time, they produce samples with enough variability and quality: in fact a classifier trained on generated samples outperforms a classifier trained directly on shift noise (Table 3.2) both in term of accuracy and noise uniformity.

of structure is given by the asymmetry of the confusion matrix  $M$ , defined as:

$$\delta_A(M) = \frac{\|M - M^T\|_F}{2\|M\|_F}. \quad (3.3)$$

In general,  $0 \leq \delta_A(M) \leq 1$ . We have  $\delta_A(M) = 0$  for symmetric matrices, thus  $\delta_A(M) \approx 0$  for uniform noise, since  $M$  would be approximately symmetric. For shift noise,  $0 < \delta_A(M) < 1$ . The lower  $\delta_A$  the more uniform the noise. Values for  $\delta_A$  are given for all the considered benchmarks in Table 3.2, together with the amount of correctly inferred labels (*i.e.* accuracy).

### 3.3.2 Classifiers

As already mentioned, a study on the robustness of classifiers against label noise is proposed by Rolnick et al. (93). We integrate their findings by training a set of classifiers with labels corrupted by shift noise. In practice, we train a first ancillary classifier  $M_{\theta_s}(x)$  on a source domain  $\mathcal{S}$ , and then use it to assign labels  $\tilde{y} = M_{\theta_s}(x_t)$  for the samples of a target domain



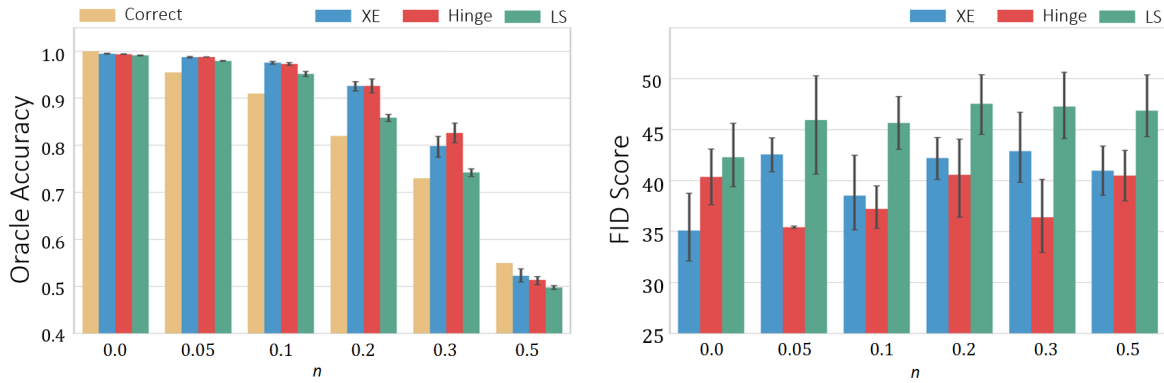


Figure 3.2 *Left* panel plots the fraction of images correctly generated by cGANs with different levels  $n$  of uniform noise and different objectives (*blue*: cross-entropy (37), *red*: Hinge (79), *green*: least-squares (76)), evaluated through the oracle. *Yellow* bars indicate the percentage of images with the correct label in the training set. *Right* panel shows the FID scores achieved with different levels of noise and different GAN objectives (same as *left*).

$\mathcal{T}$ . Eventually, we train a new classifier from scratch on the noisy set  $\mathcal{T} = \{x_t^{(i)}, \tilde{y}^{(i)}\}_{i=1\dots m}$ , where labels  $\tilde{y}$  are corrupted by shift noise deriving from misclassifications produced by  $M_{\theta_s}$ .

Classifiers tend to tolerate uniform noise in training labels to a good extent (93). Differently, as reported in Table 3.2 - columns 2 and 3 - we notice that a classifier trained with shift noise in the training labels is perfectly able to (over)fit it, being accuracies on the training set and noise asymmetry in the confusion matrices nearly the same. This can also be noticed when comparing the first and second columns of Table 3.1.

Note that this behavior does not depend on the *amount* of noise, but only on its nature. In fact, training the same classifiers with the equivalent amount of uniform noise (eq. 3.1) produces higher accuracies (together with, of course, nearly null  $\delta_A$ ), as shown in Table 3.2, columns 4 and 5. Note also that we train all classifiers till convergence, since we have no means for early stopping. This reflects the UDA setting where no target validation labels are provided.

Since deeper architectures, and Residual Networks (43) in particular, are more robust against uniform noise (93), one could wonder whether they could prove more resilience against shift noise than shallow models: this does not happen as we show in the experiments provided in the Supplementary Material.

### 3.3.3 cGANs

Given the success of GANs in UDA, we investigate their properties in terms of robustness against both uniform and shift noise, since never investigated before.

**Uniform noise.** We consider the MNIST dataset (61) and assume to have an oracle to classify which class a sample belongs to. In practice, this oracle is a ConvNet trained on MNIST, that achieves  $> 99\%$  accuracy on both the training and the test sets. Accuracy of such oracle on GAN-generated samples is referred to as the *GAN-test* metric (107).

One might genuinely expect that training a cGAN with, *e.g.*, a fraction of noisy labels  $n = 0.1$  will result in  $\sim 10\%$  of mis-generated samples. We show in the following that this does not occur. We train the cGAN with different levels of uniform noise  $n$  and evaluate the output of the generator through the oracle, by comparing the label code given in input to the cGAN and the output of the oracle fed with the generated image.

Figure 3.2 (*left*) reports our findings. Yellow bars indicate the percentage of correct labels in the training set. Blue, red and green bars indicate the percentage of samples correctly generated by cGANs trained with cross-entropy (37), Hinge (79) and least-squares (76) losses, respectively. As it can be observed, when the level of noise  $\alpha$  is reasonably below some threshold, the amount of images correctly generated (, correctly classified by the oracle) is always consistently higher than the amount of clean training samples, meaning that the cGAN can ignore noisy labels to some extent.

Several objectives have been proposed for the GAN formulation, which theoretically minimize different divergences between the data distribution  $p_d(x)$  and the generated one  $p_g(x)$ . For instance, (i) the original GAN, that uses the cross-entropy loss, is proven to minimize the Jensen-Shannon divergence  $D_{JS}(p_d||p_g)$  (37); (ii) the least-squares GAN (76) is proven to minimize the Pearson (87) divergence  $D_P(p_d + p_g||2p_g)$ ; (iii) a GAN with a Hinge loss is proven to minimize the reverse KL divergence  $D_{KL}(p_g||p_d)$  (79). In principle, being the KL divergence not symmetric, minimizing  $D_{KL}(p_d||p_g)$  would place high probability everywhere the data occurs, while  $D_{KL}(p_g||p_d)$  should enforce low probability wherever the data does *not* occur (36), thus yielding models more prone to mode collapse (38). Furthermore, it has been suggested that the Pearson divergence is more resistant to outliers than the KL divergence (110; 111), and we can interpret samples with noisy label as outliers in the conditional distributions. We are thus interested in understanding how the different objectives, theoretically associated with different divergences, behave in presence of noisy labels

There seem to be some differences between the three objectives: the least-squares GAN (76) appears to be less resistant to noisy samples. However, since there is no substantial gap between a Hinge GAN and a cross-entropy GAN, we will only use the latter from now on in this section.

Figure 3.2 (*right*) reports the FID scores (Fréchet Inception Distance (45)) for the same models. The FID is an indirect measure of image quality, accounting for the distance between the training and the generated distributions (the lower, the better). Interestingly, there seem to be no correlation between the amount of noisy labels and the overall quality of the images.

**Shift Noise.** As for classifiers, we train several cGANs on  $\mathcal{T} = \{x_t^{(i)}, \tilde{y}^{(i)}\}_{i=1\dots m}$ , we try to generate image sets starting from shift-noisy labels. In order to assess model performances, we exploit the metrics proposed by (107), *GAN-test* and *GAN-train*, which specifically deal with classifiers. As already mentioned, the *GAN-test* is the accuracy of an “oracle” classifier trained on real images and evaluated on generated images. This metric tries to capture the precision (i.e., image quality) of GANs. We thus train an oracle classifier for each target set and use it to test the corresponding cGAN trained with shift noise. Ground-truth labels for evaluating the oracle are those fed into the cGAN for class-conditional image generation. The *GAN-train* is instead the accuracy of a classifier trained on generated images and evaluated on real test images. This metric tries to capture the recall (i.e., diversity) of samples generated (107).

Accuracies and asymmetries of such classifiers are reported in Table 3.3, and confusion matrices are shown in Table 3.1. Interestingly, the samples generated by the CGANs not only induce a better accuracy on the oracle classifier, but also *significantly reduce the amount of asymmetry of the confusion matrices*. This also happens for a classifier trained on generated samples, although to a lower amount.

**Summary.** In conclusion, training a cGAN on the set  $\mathcal{T} = \{x_t^{(i)}, \tilde{y}^{(i)}\}_{i=1\dots m}$  allows to “filter” noise in  $\tilde{y}^{(i)}$  in two respects: *i*) by reducing the amount of shift noise in the generated data and *ii*) by reducing the asymmetry of shift noise, making its distribution more alike uniform noise and thus more tolerable for classifiers (93). As a matter of fact, Tables 3.1 and 3.3 show that classifiers trained on generated samples (*GAN-train* column) perform better than classifiers trained with shift noise (*Classifier* column).

### 3.4 Application to UDA

In this section, we detail the method designed to face UDA, based on the insights and the findings reported so far.

As already mentioned, we can interpret data from the target distribution  $p_{target}$ , with pseudo-labels (inferred through a classifier trained on data from the source distribution

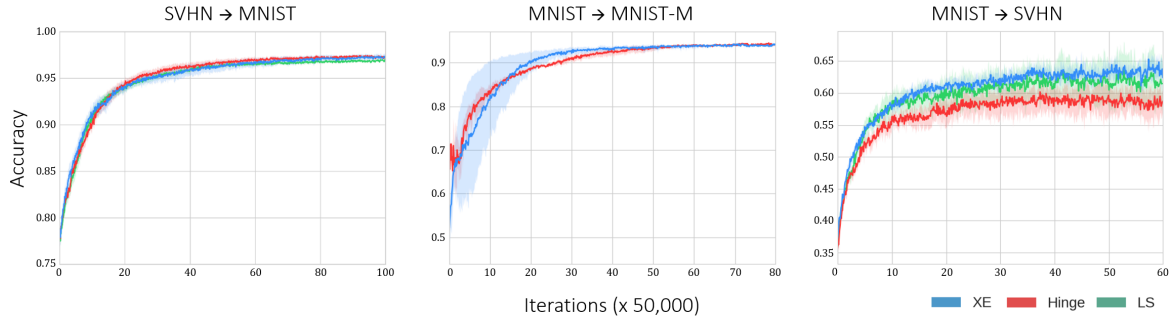


Figure 3.3 Evolution of the accuracy on target test sets for SVHN  $\rightarrow$  MNIST, MNIST  $\rightarrow$  MNIST-M and MNIST  $\rightarrow$  SVHN (from *left to right*), computed throughout the training procedure described in Algorithm 1. *Blue*, *red* and *green* curves are associated with GANs trained with the cross-entropy loss (37), the Hinge loss (79) and the least-squares loss (76), respectively. Results obtained with the least-squares loss are not reported for the MNIST  $\rightarrow$  MNIST-M as they are significantly worse than the ones achieved with the other options. Curves are averaged over three different runs, shades represent the confidence bands.

$p_{source}$ ), as a dataset polluted with label noise. From this perspective, training a cGAN on such empirical, noisy distribution should allow us to generate cleaner samples, as suggested by the findings reported in Section 3.3. In turn, a classifier trained on generated data will perform better than the one trained on source data, since noise in the target labels has been reduced in both amount and asymmetry. Starting from these two insights, we define a training procedure where we simultaneously train a classifier  $C$  and the modules  $G$  and  $D$  that define a cGAN (see Figure 3.1).

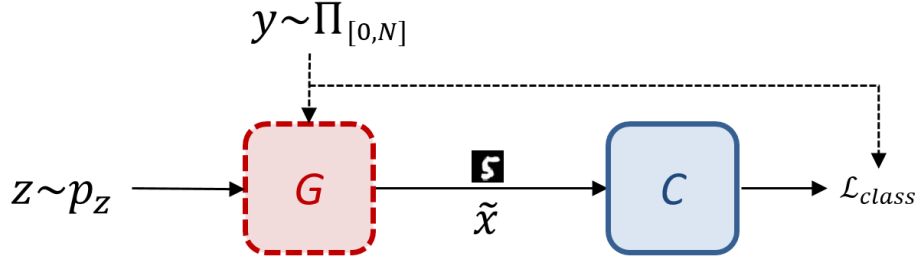
The pre-train step of our method consists in training a model  $M_{\theta_s}$  on labeled data from the source distribution

$$\min_{\theta_c} \mathcal{L}_{class} \mathbb{E}_{x,y \sim p_{source}} H(x,y; \theta_s), \quad (3.4)$$

where  $H$  is the cross-entropy loss. Equipped with this classifier, we can straightforwardly infer pseudo-labels for each target sample as  $\tilde{y}_t = C(x_t)$ . Typically, an unknown percentage of these labels will be wrong, due to the domain shift between  $p_{source}$  and  $p_{target}$ . We obviously do not know which labels are correct and which are not, but this is irrelevant for the devised strategy.

Before starting the joint training procedure, we also need to train a cGAN on the noisy target distribution, as in the previous section. This is necessary because we will train  $C$  on generated data, and thus starting with randomly-initialized  $G$  and  $D$  would result in a random classifier  $C$ , and consequently in non-informative pseudo-labels.

Step 1: training  $C$



Step 2: training  $G$  and  $D$

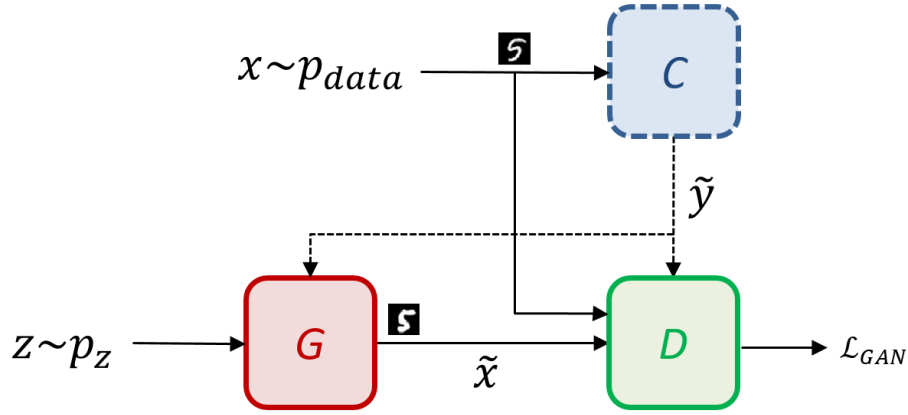


Figure 3.4 Graphical view of Algorithm 1. Step 1 (*top*) and step 2 (*bottom*) refer to lines 3 – 5 and 6 – 9, respectively. The module  $G$  and the module  $D$  are the generator and the discriminator of the cGAN. The module  $C$  is the classifier. Dashed boxes indicate frozen modules (not trained). Solid and dashed wires indicate image and label flows, respectively.  $\Pi_{[0,N]}$  is the discrete uniform distribution.

We train the cGAN in a standard fashion, alternating between the following minimax game. Note that we report here the objective as defined in Goodfellow et al. (37), but in our experiments we also test least-squares GANs (76) and Hinge-GANs (79).

$$\min_{\theta_D} \max_{\theta_G} \mathcal{L}_{GAN} \mathbb{E}_{x, \tilde{y} \sim p_{target}} [-\log D(x|\tilde{y})] + \mathbb{E}_{z \sim p_z} [-\log(1 - D(G(z|\tilde{y})|\tilde{y}))] \quad (3.5)$$

Armed with the pre-trained modules  $C$ ,  $G$  and  $D$ , we can start the training procedure, which is defined by Algorithm 1. In short, we alternate until convergence between (i) updating the weights of the classifier  $\theta_c$  via stochastic gradient descent, with labeled target samples uniformly generated via  $G$  (lines 3 – 5), and (ii) training the weights of the discriminator  $\theta_D$  and of the generator  $\theta_G$  via stochastic gradient descent, with target samples from  $p_{target}$ ,

**Algorithm 1** Pseudo-Label Refinement (PLR)

---

**Input:** target data distribution  $p_{target}$ , noise distribution  $p_z$ , pre-trained  $\theta_D^0, \theta_G^0, \theta_C^0$ , step sizes  $\eta, \delta$

**Output:** learned weights  $\theta_D, \theta_G, \theta_C$

- 1: **Initialize:**  $\theta_D \leftarrow \theta_D^0, \theta_G \leftarrow \theta_G^0, \theta_C \leftarrow \theta_C^0$
- 2: **while** not done **do**
- 3:     Sample  $z \sim p_z$  and  $y \sim \Pi_{[0,N]}$
- 4:     Generate  $\tilde{x} = G(z|y)$
- 5:      $\theta_C \leftarrow \theta_C - \eta \nabla_{\theta_C} \mathcal{L}_{class}(\theta_C; \tilde{x}, y)$
- 6:     Sample  $x \sim p_{target}$  and  $z \sim p_z$
- 7:     Infer  $\tilde{y} = C(x)$
- 8:      $\theta_D \leftarrow \theta_D - \delta \nabla_{\theta_D} \mathcal{L}_{GAN}(\theta_D; z, x, \tilde{y})$
- 9:      $\theta_G \leftarrow \theta_G - \delta \nabla_{\theta_G} \mathcal{L}_{GAN}(\theta_G; z, \tilde{y})$
- 10: **end while**

---

with pseudo-labels inferred via the classifier  $C$  (lines 6 – 9). Alternating the two steps will progressively reduce the amount and asymmetry of the initial shift noise in the target set.

In Figure 3.4, we show the computation flow of the proposed system: *top* (step 1) and *bottom* (step 2) panels represent modules corresponding to lines 3 – 5 and 6 – 9 of Algorithm 1, respectively.

The output of Algorithm 1 is twofold: (i) the trained modules of the cGAN ( $G$  and  $D$ ), and (ii) the trained classifier  $C$ , which is the module finally used to classify target samples. In the next section, we report performances obtained using  $C$  in UDA benchmarks.

## 3.5 Experiments

We test Algorithm 1 on a variety of UDA benchmarks. In every experiment, we run Algorithm 1 until convergence, intended as convergence of the cGAN minimax game, and use accuracy on target dataset test sets (fed to the classifier  $C$ ) as a metric to evaluate our models and compare them with other adaptation approaches.

**Benchmarks.** We test our method on the following cross-dataset digit classification problems: SVHN  $\leftrightarrow$  MNIST, MNIST  $\rightarrow$  MNIST-M and USPS  $\leftrightarrow$  MNIST, following protocols on which UDA algorithms based on GANs are tested (17; 70; 97; 103; 113). In order to work with comparable sizes, we resized all images to  $32 \times 32$ . For each experiment, we use a ConvNet with architecture *conv-pool-conv-pool-fc-fc-softmax*. For the GAN architectures, we

	SVHN $\rightarrow$ MNIST	MNIST $\rightarrow$ SVHN	MNIST $\rightarrow$ MNIST-M	USPS $\rightarrow$ MNIST	MNIST $\rightarrow$ USPS
Train on source	0.682	0.314	0.548	0.612	0.783
DANN (32)	0.739	-	0.767	-	-
ADDA (117)	$0.760 \pm 0.018$	-	-	$0.901 \pm 0.008$	$0.894 \pm 0.002$
DIFA (120)	$0.897 \pm 0.020$	-	-	$0.897 \pm 0.005$	$0.962 \pm 0.002$
MECA (80)	0.952	-	-	-	-
ATT (100)	0.862	0.528	0.942	-	-
AD (101)	$0.950 \pm 0.187$	-	-	$0.931 \pm 0.127$	$0.961 \pm 0.029$
MCD (102)	$0.962 \pm 0.004$	-	-	$0.941 \pm 0.003$	$0.965 \pm 0.003$
CoGAN (70)	-	-	-	0.931 (69)	0.957 (69)
DTN* (113)	0.849	-	-	-	-
UNIT* (69)	0.905	-	-	0.936	0.960
PixelDA** (17)	-	-	0.982	-	0.959
SBADA** (97)	0.761	0.611	0.994	0.950	0.976
GenToAd (103)	$0.924 \pm 0.009$	-	-	$0.908 \pm 0.013$	$0.953 \pm 0.007$
CycADA (46)	$0.904 \pm 0.004$	-	-	$0.965 \pm 0.001$	$0.956 \pm 0.002$
<b>Ours (PLR)</b>					
<i>Cross-entropy</i>	$0.973 \pm 0.006$	$0.634 \pm 0.026$	$0.943 \pm 0.002$	$0.918 \pm 0.013$	$0.893 \pm 0.019$
<i>Least-squares</i>	$0.969 \pm 0.003$	$0.618 \pm 0.060$	-	$0.916 \pm 0.019$	$0.903 \pm 0.013$
<i>Hinge</i>	$0.973 \pm 0.003$	$0.586 \pm 0.041$	$0.938 \pm 0.002$	$0.891 \pm 0.010$	$0.907 \pm 0.022$
Train on target	0.992	0.913	0.964	0.992	0.999

Table 3.4 Comparison between our method (Pseudo-Label Refinement - PLR) with different GAN objectives and competing algorithms. Test-set accuracies are the results of averaging over 3 different runs. (\*) Uses extra SVHN data (531, 131 images). (\*\*) Uses 1,000 target samples for cross-validation.

draw inspiration from DCGAN (91), though considering different objectives (cross-entropy, least-squares, Hinge). All details regarding architectures and training procedures are reported in the Appendix A.

### 3.5.1 Results

We report in Figure 3.3 the plots showing the evolution of test accuracy in different experiments throughout the training procedure defined by Algorithm 1. As it can be observed, the performance on target domain of the classifier trained on target samples generated via the cGAN is improved over iterations. It is worth highlighting the monotonic increase of performance: early stopping is not feasible in UDA, thus an unstable algorithm is of scarce utility.

A particularly important result is the one related to the MNIST  $\rightarrow$  SVHN split. The large gap between the two domains, and the fact that labels are provided for the easier, more biased dataset makes this split particularly difficult to tackle (32; 33). Our method allows to generate SVHN samples that make the classifier  $C$  – trained on them – better generalizing to the target distribution, improving performance of  $\sim 30\%$  with respect to the baseline. The complete analysis of the obtained results, also in comparison with the state-of-the-art methods, is illustrated in the following.

**Comparison with other methods.** Table 3.4 compares the proposed method performance (Pseudo-Label Refinement, PLR) with the results obtained by several works in the literature. It is worth to note that, nowadays, research in UDA reached a point where it is difficult to state the superiority of a method over the others. Indeed, Table 3.4 shows that there is not a single method that performs better than the others in *every* benchmark.

First, our method shows performance comparable with the state of the art in the SVHN  $\rightarrow$  MNIST split benchmark, significantly outperforming more complex image-to-image translation methods (69; 97; 103; 113) that not only rely on more complicated architectures, but also present a training procedure where the objective is weighted by different hyperparameters (which, as previously mentioned, is a significant drawback in UDA).

Next, an important result is the one related to MNIST  $\rightarrow$  SVHN. As discussed above, this is a rather challenging split, and several methods (*e.g.*, (32; 69; 70; 80; 113; 117; 120)) do not show results on this benchmark. Furthermore, we tested the implementation of PixelDA (17) provided by the authors and could not observe any sign of adaptation. Our algorithm, with the cross-entropy loss as GAN objective, is the best performing method by a statistically significant margin. We also stress that Russo et al. (97), the second best performing method



on this split, use 1,000 samples from SVHN to cross-validate the hyperparameters, thus making the working setup much easier than ours.

On MNIST  $\rightarrow$  MNIST-M, the performance achieved with our method is comparable with Saito et al. (100) and below the one achieved by methods that perform hyperparameter cross-validation (17; 97).

## 3.6 Conclusion

We introduce the concept of shift noise and analyze the robustness of classifiers and cGANs against such highly structured noise. We empirically show that, while classifiers are generally not robust against this kind of label noise, cGANs are more resilient against it, and furthermore generate samples with a more uniform noise distribution. Inspired by these findings, we design a training procedure that progressively allows to generate cleaner samples from the target distributions, and in turn to train better classifiers.

For future work, we hope to extend the devised algorithm to more realistic UDA benchmarks, such as Office-31 (98) and VisDA (88). The limitation towards this goal is the current computational expense in training GANs that generate high-resolution samples (19; 51).

# Chapter 4

## Learning Unbiased representations via Mutual Information Backpropagation

### 4.1 Context

In this chapter, we are interested in learning representations that are discriminative for the supervised learning task of interest, while being invariant to certain specified *biased attributes* of the data. By “biased attribute”, we mean an inherent bias of the dataset, which is assumed to be known and follows a certain distribution during training. At test time, the distribution of such attribute may abruptly change, thus tampering the generalization capability of the model and affecting its performance for the given task (4; 53; 81).

One intuitive example is provided in Figure 4.1: we seek to train a *shape classifier*, but each shape has a distinct color – the biased attribute. Unfortunately, a model can fit the training distribution by discriminating either the color or the shape. Among the two options, we are interested in the latter only, because the first one does not allow generalizing to shapes with different colors. Thus, if we were capable of learning a classifier while unlearning the color, we posit that it would better generalize to shapes with arbitrary colors. Like other prior works (4; 53; 72; 81), we operate in a scenario where the labels of biased attributes are assumed to be known. An example of application domain in which the hypothesis of having known labels for the bias holds, is algorithmic fairness (30; 57; 125; 128), where the user specifies which attributes the algorithm has to be invariant to (, learning a face recognition system which is not affected by gender or ethnicity biases).

We tackle this problem through the lens of information theory. Since mutual information can be used to quantify the nonlinear dependency of the learned feature space with respect to the dataset bias, we argue that a good strategy to face the aforementioned problem

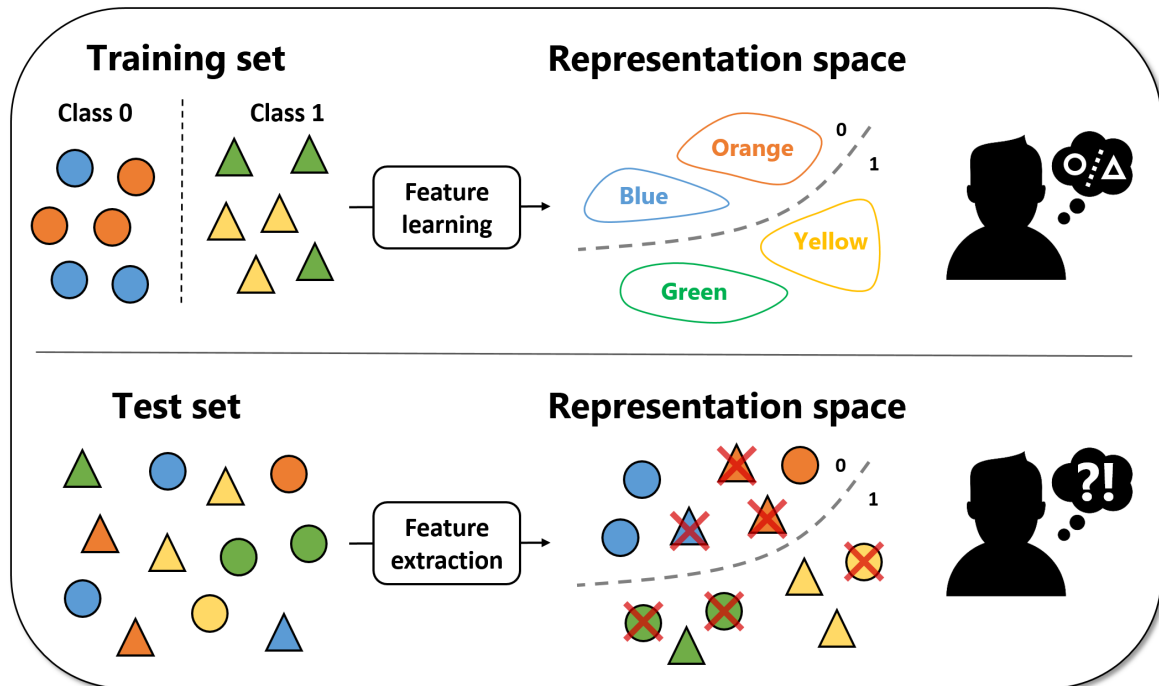


Figure 4.1 **Problem setting.** When learning a feature representation from the data itself (*top*), we may undesirably capture the inherent bias of the dataset (here, exemplified by colors), as opposed to learning the desired patterns (here, represented by shapes). This results in models that poorly generalize when deployed into unbiased scenarios (*bottom*).

is minimizing the mutual information between the learned representation and the biased attributes. This would result in a data representation that is statistically independent from the specified bias, and that, in turn, would generalize better.

Unfortunately, the estimation of the mutual information is not a trivial problem (90). In the context of representation learning, two bodies of work proposed solutions to the problem of learning unbiased representations via information theoretic measures: one that relies on adversarial training (4; 53), and one based on variational inference (81). Adversarial methods (4; 53) learn unbiased representations by “fooling” a classifier trained to predict the attribute from the learned representation. Such condition is argued to be a proxy for the minimization of the mutual information (53).

Other approaches rely on variational inference, which properly formalizes the prior and the conditional dependences among variables. However, when implementing those methods in practice, approximations need to be done to replace the computationally intractable posterior with an auxiliary distribution, but at the cost of several assumptions of independence

among the variables. Moreover, such methods are more problematic to scale to complex computer vision tasks, and have been applied mostly on synthetic or toy datasets (72; 81).

Due to the aforementioned difficulties, we leverage the mathematical soundness of mutual information to design a computational pipeline that is alternative to standard adversarial training. We rely on a neural estimator for the mutual information (MINE (11)). This module provides a more reliable estimate of the mutual information (90), while still being fully differentiable and, therefore, trainable via backpropagation (96).

Endowed with this model, we propose a training scheme where we alternate between (i) optimizing the estimator and (ii) learning a representation that is both discriminative for the desired task and statistically independent from the specified bias.

A key strength of the proposed approach is that the module that estimates the mutual information is not competing with the feature extractor (differently from existing adversarial methods (53)). For this reason, MINE can be trained until convergence at every training step, avoiding the need to carefully balance between steps (i) and (ii), and guaranteeing an updated estimate of the mutual information throughout the training process.

In adversarial methods such as (53), where the estimate for the mutual information is modeled via a discriminator that the feature extractor seeks to fool (32; 33), one cannot train an optimal discriminator at every training iteration. Indeed, if one trains an optimal bias discriminator, the feature extractor will no longer be able to fool it, due to the fact that gradients will become too small (5) – and the adversarial game will not reach optimality. This difference is a key novelty of the proposed computational pipeline, which scores favorably with respect to prior work on different computer vision benchmarks, from color-biased classification to age-invariant recognition of people attributes.

Furthermore, a critical aspect of this line of work (4; 53) is how to balance between learning the desired task and “unlearning” the dataset bias, which is a core, open issue (128). The training strategy proposed in this chapter allows for a very simple solution to this important problem. Indeed, as we will show later in the experimental analysis, a very effective approach is selecting the models whose learned representation distribution has the lowest mutual information with that of the biased attribute. We empirically show that these models are also the ones that better generalize to unbiased settings. Most notably, this also provides us with a simple cross-validation strategy for the hyper-parameters: without using any *validation data*, we can select the optimal model as the one that achieves the best fitting to the data, while better minimizing the mutual information. The importance of this contribution is that, when dealing with biased datasets, also the validation set will likely suffer from the same bias, making hyper-parameter selection a thorny problem. Our proposed

method properly responds to this problem, whereas former works have not addressed the issue (53).

## 4.2 Related works

The problem of learning unbiased representations has been explored in several sub-fields. In the following section, we cover the most related literature, with particular focus on works that approach our same problem formulation, highlighting similarities and differences.

In **domain adaptation** (15; 49; 98), the goal is learning representations that generalize well to a (target) domain of interest, for which only unlabeled – or partially labeled – samples are available at training time, leveraging annotations from a different (source) distribution. In domain generalization, the goal is to better generalize to unseen domains, by relying on one or more source distributions (64; 83). Adversarial approaches for domain adaptation (32; 33; 117; 120) and domain generalization (105; 134) are very related to our work: their goal is indeed learning representations that do not contain the domain bias, and therefore better generalize in out-of-distribution settings. Differently, in our problem formulation we aim at learning representations that are invariant towards specific attributes that are given at training time.

A similar formulation is related to the so-called “**algorithmic fairness**” (57). The problem here is learning representations that do not rely on sensitive attributes (such as, *e.g.*, gender, age or ethnicity), in order to prevent from learning discriminant capabilities towards protected categories. Our methods can be applied in this setting, in order to minimize the mutual information between the learned representation and the sensitive attribute (whose distribution might be biased for what concerns the training set). In these settings, it is important to notice that a “fairer” representation does not necessarily generalize better than a standard one: the trade-off between accuracy and fairness is termed “fairness price” (30; 57; 125; 128).

There is a number of works that share our same goal and problem formulation. Alvi et al. (4) learn unbiased representations through the minimization of a confusion loss, learning a representation that does not inherit information related to specified attributes. Kim et al. (53) propose to minimize the mutual information between learned features and the bias. However, they face the optimization problem through standard adversarial training: in practice, in their implementation (kim), the authors rely on a discriminator trained to detect the bias as an estimator for the mutual information, and learn unbiased representations by trying to fool this module, drawing inspiration from the solution proposed by Ganin and Lempitsky (32) for domain adaptation. Moyer et al. (81) also introduce a penalty term based on mutual

information, to achieve representations that are invariant to some factors. In contrast with related works (4; 53; 81), it shows that adversarial training is not necessary to minimize such objective, and the problem is approached in terms of variational inference, relying on Variational Auto-Encoders (VAEs (56)). Closely related to Moyer et al., other works (26; 72) impose a prior on the representation and the underlying data generative factors (, feature vectors are distributed as a factorized Gaussian).

Our proposed solution provides several advantages to existing adversarial approaches (4; 53) and VAE based ones (81). With respect to adversarial strategies, our method has the advantage of relying on a module estimating the mutual information (11) that is not competing with the network trained to learn an unbiased representation. In our computational pipeline, we do not learn unbiased representation by “fooling” the estimator, but by minimizing the information that it measures. The difference is subtle (for instance, we also end up approaching a minmax optimization problem), but brings a crucial advantage: in standard adversarial methods, the discriminator (estimator) cannot be trained until convergence at every training step, otherwise gradients flowing through it would be close to zero almost everywhere in the parameter space (5), preventing from learning an unbiased representation. In our case, the estimator can be trained until convergence at every training step, improving the quality of its measure without any drawbacks. Furthermore, our solution can easily scale to large architectures (, for complex computer vision tasks) in a straightforward fashion. While this is true also for adversarial methods (4; 53), we posit that it might not be the case for methods based on VAEs (81), where one has to simultaneously train a feature extractor/encoder and a decoder.

### 4.3 Problem Formulation

We operate in a setting where data are shaped as triplets  $(\mathbf{x}, \mathbf{y}, \mathbf{c})$ , where  $\mathbf{x}$  represents a generic datapoint,  $\mathbf{y}$  denotes the ground truth label related to a task of interest and  $\mathbf{c}$  encodes a vector of given attributes. We are interested in learning a representation  $\mathbf{z}$  of  $\mathbf{x}$  that allows performing well on the given task, with the constraint of not retaining information related to  $\mathbf{c}$ . In other words, we desire to learn a model that, when fed with  $\mathbf{x}$ , produces a representation  $\mathbf{z}$  which is maximally discriminative with respect to  $\mathbf{y}$ , while being invariant with respect to  $\mathbf{c}$ .

In this work, we formalize the invariance of  $\mathbf{z}$  with respect to  $\mathbf{c}$  through the lens of information theory, imposing a null mutual information  $I$ . Specifically, we constrain the discriminative training (finalized to learn the task of interest) by imposing  $I(Z, C) = 0$ , where

$Z$  and  $C$  are the random variables associated with  $\mathbf{z}$  and  $\mathbf{c}$ , respectively. In formulæ, we obtain the following constrained optimization

$$\min_{\theta, \psi} \mathcal{L}_{task}(\theta, \psi), \quad s.t. \quad I(Z, C) = 0 \quad (4.1)$$

where  $\theta$  and  $\psi$  define the two sets of parameters of the objective  $\mathcal{L}_{task}$ , which can be tailored to learn the task of interest. With  $\theta$ , we refer to the trainable parameters of a module  $g_\theta$  that maps a datapoint  $\mathbf{x}$  into the corresponding feature representation  $\mathbf{z}$  (that is,  $\mathbf{z} = g_\theta(\mathbf{x})$ ). With  $\psi$ , we denote the trainable parameters of a classifier that predicts  $\tilde{\mathbf{y}}$  from a feature vector  $\mathbf{z}$  (that is,  $\tilde{\mathbf{y}} = f_\psi(\mathbf{z})$ ). The constraint  $I(Z, C) = 0$  does not depend upon  $\psi$ , but only upon  $\theta$ , since  $\mathbf{z}$  obeys to  $p_Z$  and  $\mathbf{z} = g_\theta(\mathbf{x})$ . The variable  $\mathbf{c}_i$  does neither depend on  $\theta$  nor  $\psi$ , since it is given as ground truth.

In order to optimize the objective in (4.1), we must adopt an estimator of the mutual information. Before detailing our approach, in the following paragraph we cover the background required for a basic understanding of mutual information estimation, with focus on the path we pursue in this work.

**Background on information theory.** The mutual information between two random variables  $X, Z$  is given by

$$I(X, Z) = \int p_{X,Z}(x, z) \log \frac{p_{X,Z}(x, z)}{p_X(x) \cdot p_Z(z)} dx dz,$$

where  $p_{X,Z}$  denotes the joint probability of the two variables and  $p_X, p_Z$  represent the two marginals. As an alternative to covariance and other linear indicators of statistical dependence, mutual information can account for generic inter-relationships between  $X, Z$ , going beyond simple correlation (21; 22).

The main drawback with mutual information relates to its difficult computation, since the probability distributions  $p_X, p_Z$  and  $p_{X,Z}$  are not known in practice. Even in that case, exact calculation is possible only when dealing with discrete  $X, Z$  or for very particular distribution  $p_X, p_Z$ , *e.g.* Gaussian (106). In all other cases, the integral needs to be approximated. Furthermore, the methods applied to compute the mutual information are frequently problem-specific, since assuming restrictive priors over  $X, Z$ .

Recently, a general purpose and efficient estimator for mutual information has been proposed by Belghazi et al. (11). They propose a neural network based approximation to compute the following lower bound for the mutual information  $I$ :

$$\mathcal{L}_{ne} := \mathbb{E}_{(\mathbf{z}, \mathbf{c}) \sim \hat{p}_{Z,C}} [T_\phi(\mathbf{z}|\mathbf{c})] - \log \mathbb{E}_{\mathbf{z} \sim \hat{p}_Z, \tilde{\mathbf{c}} \sim \hat{p}_C} [\exp T_\phi(\mathbf{z}|\tilde{\mathbf{c}})] \quad (4.2)$$

When implementing  $T_\phi$  as a feed-forward neural network, the maximization in Eq. 4.2 can be efficiently solved via backpropagation (11).

As a result, we can approximate  $I(X, Z)$  with  $\hat{I}_\phi(X, Z)$ , the so-called ‘‘Mutual Information Neural Estimator’’ (MINE (11)). An appealing aspect of MINE is its fully differentiable nature, that enables end-to-end optimization of objectives that rely on mutual information computations.

## 4.4 Method

In the following, we detail how we approach Eq. (4.1), both in terms of theoretical foundations and practical implementation.

### 4.4.1 Optimization problem

In order to proceed with a more tractable problem, we consider the Lagrangian of Eq. (4.1)

$$\min_{\theta, \psi} \mathcal{L} := \mathcal{L}_{task}(\theta, \psi) + \lambda I(Z, C) \quad (4.3)$$

where the first term is a loss associated with the task of interest, whose minimization ensures that the learned representation is sufficient for our purposes. The second term is the mutual information between the learned representation and the given attributes. The hyper-parameter  $\lambda$  balances the trade-off between optimizing for a given task and minimizing the mutual information.

Concerning the first term of the objective, we will consider classification tasks throughout this work, and thus we assume that our aim is minimizing the cross-entropy loss between the output of the model  $\tilde{\mathbf{y}}$  and the ground truth  $\mathbf{y}$ .

$$\mathcal{L}_{task} = \frac{1}{N} \sum_{i=1}^N \mathbf{y}_i^T \log s(\tilde{\mathbf{y}}_i) \quad (4.4)$$

where  $s$  is the softmax function and  $N$  is the number of given datapoints.

Concerning the second term of the objective in Eq. (4.1), as already mentioned, the analytical formulation of the mutual information is of scarce utility to evaluate  $I(Z, C)$ .



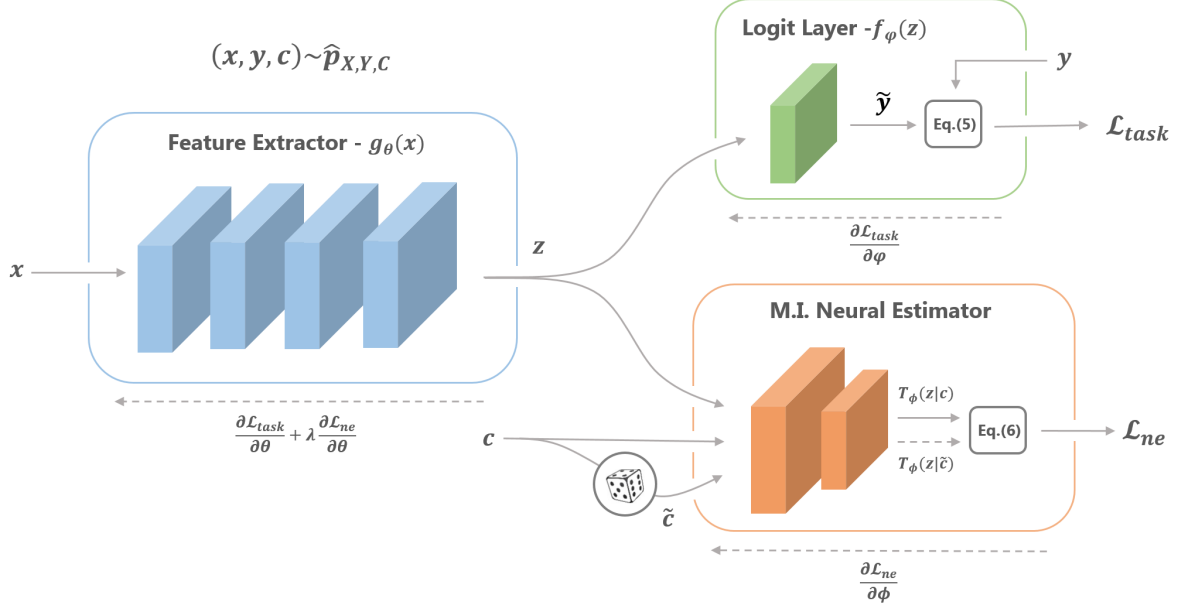


Figure 4.2 **Model overview.** The neural network devised for the given task is the concatenation of the blue module (feature extractor  $g_\theta$ ) and the green module (logit layer  $f_\psi$ ). Solid lines indicate the forward flow, dashed lines indicate gradient backward passes. The feature extractor takes in input samples  $\mathbf{x}$  and outputs feature vectors  $\mathbf{z}$ . The logit layer takes in input the feature vectors and outputs predictions  $\tilde{\mathbf{y}}$ . To optimize for the given task, these modules can be trained by minimizing the cross-entropy between predictions and labels  $\mathbf{y}$ . The orange module (11) estimates the mutual information between the feature vectors  $\mathbf{z}$  and the attributes  $\mathbf{c}$ . To estimate the mutual information,  $T_\phi$  processes the concatenation of feature vectors and attributes from the joint distribution and the marginals. Following Belghazi et al. (11), we approximate sampling from the marginal by shuffling the batch of attributes ( $\tilde{\mathbf{c}}$ ). The estimation of the mutual information is the maximum w.r.t.  $\phi$  of the output of the orange module  $\mathcal{L}_{ne}$ .

Indeed, we do not explicitly know the probability distributions that the learned representation and the attributes obey to. Therefore, we need an estimator for the mutual information  $\hat{I}(Z, C)$ , with the requirement of being differentiable with respect to the model parameters  $\theta$ .

In order to attain our targeted goal, we take advantage of the work by Belghazi et al. (11), and exploit a second neural network  $T_\phi$  (“statistics network”) to estimate the mutual information. We therefore introduce the additional loss function  $\mathcal{L}_{ne}$  (Eq. (4.2)) that, once maximized, provides an estimate of the mutual information

$$\hat{I}_{ne}(Z, C) = \max_{\phi} \mathcal{L}_{ne}. \quad (4.5)$$

In Eq. (4.2), the notation  $\hat{p}$  reflects that we rely on the empirical distributions of features and attributes, the operator “|” indicates vector concatenations and “ne” stands for “neural estimator” (11). The loss  $\mathcal{L}_{ne}$  also depends on  $\theta$ , since Eq. (4.2) depends on  $\mathbf{z}$ . Combining the pieces together, we obtain the following problem:

$$\min_{\theta, \psi} \{ \mathcal{L}_{task}(\theta, \psi) + \lambda \widehat{I}_{ne}(Z, C) \} = \underbrace{\min_{\theta, \psi} \{ \mathcal{L}_{task}(\theta, \psi) + \lambda \underbrace{\max_{\phi} \mathcal{L}_{ne}(\phi, \theta)}_{\text{MI estimation}} \}}_{\text{Representation learning}} \quad (4.6)$$

Intuitively, the inner maximization problem ensures a reliable estimate of the mutual information between the learned representation and the attributes. The outer minimization problem is aimed at learning a representation that is at the same time optimal for the given task and unbiased with respect to the attributes.

#### 4.4.2 Implementation Details

In this paragraph, we detail our approach that allows training an arbitrary neural network by optimizing the objective in Eq. (4.6). Concerning the modules introduced in Section 4.3, we implement the feature extractor  $g_{\theta}$  (which computes features  $\mathbf{z}$  from datapoints  $\mathbf{x}$ ) and the classifier  $f_{\psi}$  (which predicts labels  $\tilde{\mathbf{y}}$  from  $\mathbf{z}$ ) as feed-forward neural networks. The classifier  $f_{\psi}$  is implemented as a shallow logit layer to accomplish predictions on the task of interest. As already mentioned, the model  $T_{\phi}$  is also a neural network; it accepts in input the concatenation of feature vectors  $\mathbf{z}$  and attribute vectors  $\mathbf{c}$ , and through Eq. (4.2) allows estimating the mutual information between the two random variables. The nature of the modules allow to optimize the objective functions in (4.6) via backpropagation (96). Figure 5.3 portrays the connections between the different elements, and how the losses (4.4) and (4.2) originate.

A crucial point that needs to be addressed when jointly optimizing the two terms of Eq. (4.6) is that, while the distribution of the attributes  $\hat{p}_C$  is static, the distribution of the feature embeddings  $\hat{p}_Z$  depends on  $\theta$ , which changes throughout the learning process. For this reason, the mutual information estimator needs to be constantly updated during training, because an estimate  $\widehat{I}_{ne}(Z_t, C)$ , associated with  $\theta_t$  at step  $t$ , is no longer reliable at step  $t + 1$ . To cope with this issue, we devise an iterative procedure where, prior to every gradient descent update on  $(\theta, \psi)$ , we update MINE on the current model, through the inner maximizer in Eq. (4.6). This guarantees a reliable mutual information estimation.

**Algorithm 2** Learning Unbiased Representations

---

```

1: Input: Dataset  $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^N$ , initialized weights  $\theta_0, \psi_0, \phi_0$ , learning rates  $\alpha, \eta$ ,
   hyper-parameters  $\lambda, K, T$ .
2: Output: learned weights  $\theta, \psi$ 
3: Initialize:  $\theta \leftarrow \theta_0, \psi \leftarrow \psi_0, \phi \leftarrow \phi_0$ 

4: for  $t = 1, \dots, T$  do
5:   for  $k = 1, \dots, K$  do (estimate MI)
6:     sample mini-batches  $\{(\mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^m, \{\tilde{\mathbf{c}}^{(i)}\}_{i=1}^m$ 
7:     evaluate  $\mathcal{L}_{ne}$  (Eq. (4.2))
8:      $\phi \leftarrow \phi + \eta \nabla_{\phi} \mathcal{L}_{ne}$ 
9:   end for
10:  sample mini-batches  $\{(\mathbf{x}^{(i)}, \mathbf{x}^{(i)}, \mathbf{c}^{(i)})\}_{i=1}^n, \{\tilde{\mathbf{c}}^{(i)}\}_{i=1}^n$ 
11:  evaluate  $\mathcal{L}_{task}$  (Eq. (4.4)) and  $\mathcal{L}_{ne}$  (Eq. (4.2))
12:   $\theta \leftarrow \theta - \alpha \nabla_{\theta} (\mathcal{L}_{task} + \lambda \mathcal{L}_{ne})$ 
13:   $\psi \leftarrow \psi - \alpha \nabla_{\psi} \mathcal{L}_{task}$ 
14: end for

```

---

One key difference with standard adversarial methods(4; 53) is that we can train MINE until convergence prior to each gradient descent step on the feature extractor, without the risk of obtaining gradients whose magnitude is close to zero (5), since our estimator is not a discriminator (being the mutual information unbounded, sometimes gradient clipping is actually beneficial (11)). The full training procedure is detailed in Algorithm 2.

**Training techniques.** We list some techniques that we could appreciate to generally increase the stability of the proposed training procedure. Hyper-parameters choice is discussed in Appendix B.

(a) Despite MINE (11) can estimate the mutual information between continuous random variables, we observed that the estimation is eased (in terms of speed and stability) if the attribute labels  $\mathbf{c}$  are discrete. (b) We observed an increased stability in training MINE (11) for lower-dimensional representations  $\mathbf{z}$  and attributes  $\mathbf{c}$ . For this reason, as we will discuss in Section 4.5, feature extractors with low-dimensional embedding layer are favored. (c) The feature extractor  $g$  receives gradients related to both  $\mathcal{L}_{task}$  and  $\mathcal{L}_{ne}$ : since the mutual information is unbounded, the latter may dominate the former. Following Belghazi et al. (11), we overcome this issue via gradient clipping (we refer to original work for details). (d) We observed that training MINE requires large mini-batches: when this was unfeasible due to memory issues, we relied on gradient accumulation. (e) We observed that using

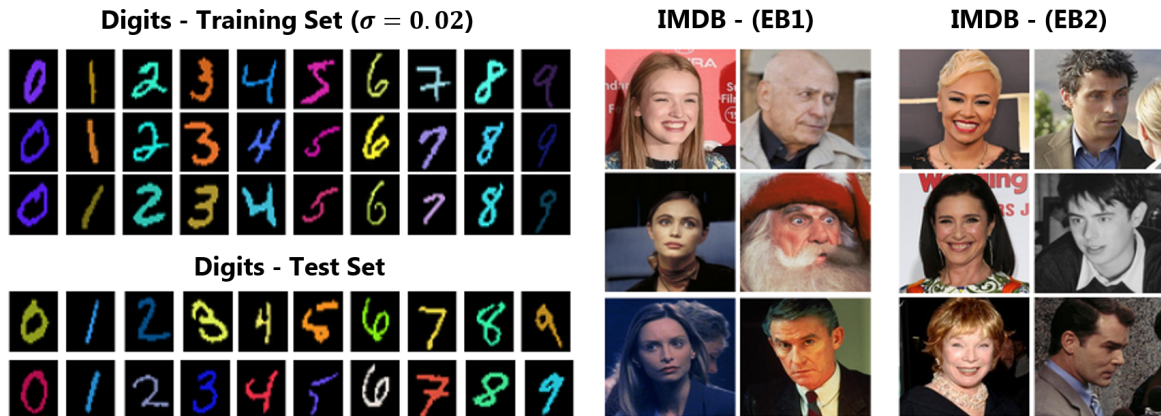


Figure 4.3 *Left*: digit examples for each class from training (here with  $\sigma = 0.02$ ) and test set. *Right*: Women and Men images from the two splits of the training set of the IMDB dataset.

vanilla gradient descent over Adam optimizer (55) eases training MINE (11) in most of our experiments.

## 4.5 Experiments

In the following, we show the effectiveness of models trained via Algorithm 3 in a series of benchmarks. First, we report results related to the setup proposed by Kim et al. (53) – learning to recognize color-biased digits without relying on color information. Next, we show that our proposed solution can scale to higher-capacity models and more difficult tasks, through the IMBD benchmark (4; 53), where the goal is classifying people gender without relying on the age bias. Finally, we show that our method can also be applied as it is to learn “fair” classifiers, by training models on the two standard benchmarks for algorithmic fairness (adu; ger).

### 4.5.1 Digit Recognition

**Experimental setup.** Following the setting defined by Kim et al. (53), we consider a digit classification task where each digit, originally from MNIST (61), shows an artificially induced color bias. More specifically, in the training set (with 60,000 samples), digit colors are drawn from Gaussian distributions, whose mean values are different for each class. In the test set (with 10,000 samples), digits show random colors. The benchmark is designed with seven different standard deviation values  $\sigma$  (equally spaced between 0.02 and 0.05): the lower the value, the more difficult the task, since the model can fit the training set by

recognizing colors instead of shapes, thus poorly generalizing (see Figure 4.3). To extract the color information (the attribute  $\mathbf{c}$ , recalling notation from Section 4.3), the maximum pixel value is encoded in a binary vector with 24-bit (8 bits per channel). Since the background is always black, the maximum value reflects the digit color.

Concerning the model, we exploit a convolutional neural network (60) with architecture *conv-pool-conv-pool-fc-fc-softmax*. The output of the second fully connected layer ( $\mathbf{z}$ ) is given in input to both the logit layer and MINE (Figure 5.3). The architecture of the statistics network  $T_\phi$  in MINE is a multi-layer perceptron (MLP) with 3 layers. More architectural details can be found in Appendix B. We compare models trained via Algorithm 3 with the solutions proposed by Kim et al. (53) and Alvi et al. (4), averaging across 3 runs and using accuracy as a metric. Before comparing against related work, we discuss how crucial hyper-parameters can be selected in our setting.

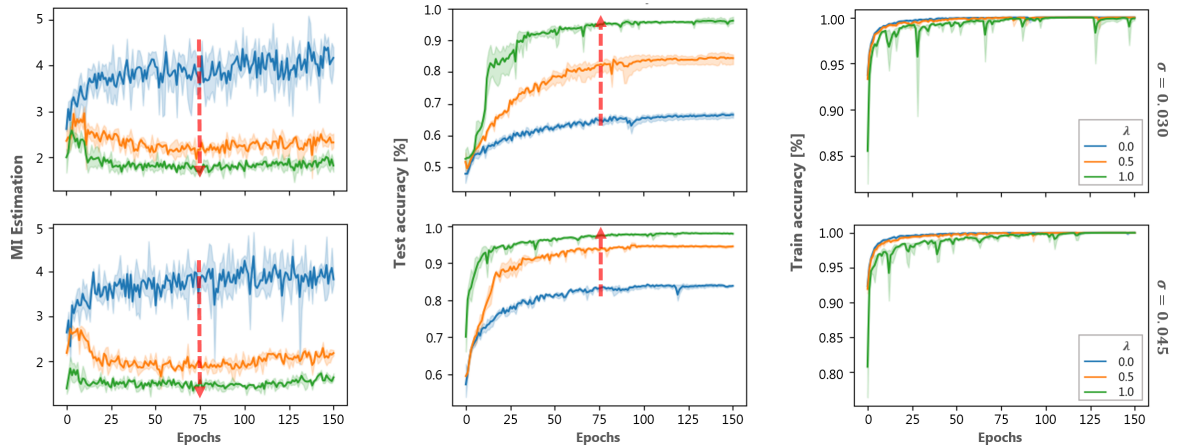


Figure 4.4 **Digit experiment – ablation study.** Evolution of mutual information estimation (left), test accuracy (middle) and training accuracy (right) for models trained on digits with  $\sigma = 0.03$  and  $\sigma = 0.045$  (top and bottom, respectively). Models are trained with Algorithm 3 with  $\lambda = 0.0$  (baseline, blue),  $\lambda = 0.5$  (orange) and  $\lambda = 1.0$  (green). Increasing the value of the hyper-parameter  $\lambda$  allows reducing the mutual information between the learned representation ( $Z$ ) and the attributes ( $C$ ). In turn, models better generalize to unbiased samples (test set). Further plots in Appendix B.

**Hyper-parameter choice.** We discuss in the following the model behavior as we modify  $\lambda$ , that governs the trade-off between learning a task and minimizing the mutual information between features and attributes.

Figure 4.4 reports the evolution of mutual information estimation (left), accuracy on test samples (middle) and accuracy on training sample (right) for models trained with  $\lambda = 0.0, 0.5, 1.0$  in blue, orange and green, respectively, for  $\sigma = 0.03, 0.045$  (top and bottom,

respectively). It can be observed that the mutual information between embeddings  $\mathbf{z}$  and color attributes  $\mathbf{c}$  can be reduced by increasing  $\lambda$ . Importantly, this results in a significantly higher accuracy on (unbiased) test samples. The importance of this result is twofold: on the one hand, it is a proof of concept of the intuition that lowering the mutual information does help generalizing to unbiased sources; on the other, it provides us a cross-validation strategy to pick a proper  $\lambda$  value (the one that allows minimizing the mutual information more efficiently). As can be observed in the plots on the right, the training procedure becomes more unstable when we increase  $\lambda$ . Therefore, in order to select the proper hyper-parameter, we can choose the highest  $\lambda$  value that allows the model fitting the data (*i.e.*, minimizing  $\mathcal{L}_{task}$ ) and reducing the mutual information (*i.e.*, minimizing  $\mathcal{L}_{ne}$ ).

Digit experiment					
Color std dev	Method				
( $\sigma$ )	ERM ( $\lambda = 0.0$ )	Alvi et al. (4)	Moyer et al. (81)	Kim et al. (53)	Ours ( $\lambda = 1.0$ )
<b>0.020</b>	0.476 $\pm$ 0.005	0.676	0.717	0.818	0.864 $\pm$ 0.052
<b>0.025</b>	0.542 $\pm$ 0.004	0.713	0.864	0.882	0.925 $\pm$ 0.020
<b>0.030</b>	0.542 $\pm$ 0.004	0.794	0.883	0.911	0.959 $\pm$ 0.008
<b>0.035</b>	0.720 $\pm$ 0.010	0.825	0.885	0.929	0.973 $\pm$ 0.003
<b>0.040</b>	0.785 $\pm$ 0.003	0.868	0.887	0.936	0.975 $\pm$ 0.001
<b>0.045</b>	0.838 $\pm$ 0.002	0.890	0.893	0.954	0.980 $\pm$ 0.001
<b>0.050</b>	0.870 $\pm$ 0.001	0.917	0.914	0.955	0.982 $\pm$ 0.001

Table 4.1 **Digit experiment – comparison with related work.** Experimental results on colored digit classification for different levels of variance ( $\sigma$ ) in the color distribution. The first row reports results related to models trained via standard Empirical Risk Minimization (ERM). Below we report results obtained by competitor methods (4; 53; 81). The last row reports results achieved with our method (with  $\lambda = 1.0$ ).

Another important hyper-parameter is the number of iterations used to train MINE (11) prior to each gradient update on the feature extractor ( $K$  in Algorithm 3). We observed that, the higher the number of iterations the better (we set  $K = 80$ ). This was expected, because the MI estimation becomes more reliable and therefore the removal of bias information is more accurate. The reader can refer to Figure B.3 in Appendix B for quantitative results.

**Comparison with related work.** We report in Table 4.1 the comparison between our method with  $\lambda = 1.0$  and related works (4; 53). We can observe consistently improved results in all splits (different  $\sigma$ 's). We emphasize that our method is more effective as the bias is more severe (small  $\sigma$ 's). It is also important to stress that other works (4; 53; 81) do not introduce any strategy to tune the hyper-parameters, whereas in this work the hyper-parameter search is efficiently resolved. Furthermore, the authors do not report any statistics around their

### IMDB experiment

Method	Train on EB1		Train on EB2	
	EB2	Test	EB1	Test
ERM ( $\lambda = 0.0$ )	$0.650 \pm 0.020$	$0.849 \pm 0.007$	$0.576 \pm 0.013$	$0.708 \pm 0.008$
Alvi et al. (4)	0.637 (53)	0.856 (53)	0.573 (53)	0.699 (53)
Kim et al. (53)	0.680	0.867	0.642	0.745
<i>Ours</i> ( $\lambda = 0.5$ )	$0.691 \pm 0.010$	$0.876 \pm 0.010$	$0.651 \pm 0.036$	$0.762 \pm 0.022$

Table 4.2 Experimental results from IMDB gender classification problem. The first row reports results obtained by setting  $\lambda = 0.0$  (ERM baseline). The last row reports results obtained with our method (*Ours*); Each column reports results associated with the indicated test set.

results (e.g., average and standard deviation across different runs), making a fair comparison difficult.

#### 4.5.2 IMDB: Removing the Age Bias

**Experimental setup.** Following related works (4; 53), we consider the IMDB dataset (94) as benchmark. It contains cropped images of celebrity faces with ground truth annotations related to gender and age. Alvi et al. (4) consider two subsets of the training set that are severely biased for what concerns age: the EB1 (“Extreme Bias”) split (36,003 samples) only contains images of women with an age in the range 0-30, and men who are older than 40; vice versa, the EB2 split (16,799 samples) only contains images of men with an age in the range 0-30, and women older than 40 (see Figure 4.3). The test set (22,468 samples) contains faces without any restrictions on age/gender (uniformly sampled). The goal here is learning an age-agnostic model, to overcome the bias present in the dataset.

Following previous work (4; 53), we encode the age attribute (our biased attribute,  $\mathbf{c}$ ) using bins of 5 years, via one-hot encoding. We use a ResNet-50 (43) model pre-trained on ImageNet (28) as classifier, modified with a 128-dimensional fully connected layer before the logit layer. This narrower embedding serves as our  $\mathbf{z}$ , and the reduced dimension eases the estimation of the mutual information, while not causing any detrimental effect in terms of accuracy. For each split (EB1 and EB2), we train the model through Algorithm 3 and evaluate it on the test set and on the split not used for training. We followed the same procedure detailed in Section 4.5.1 to choose the hyper-parameter  $\lambda$ ; we set  $K = 40$ . We compare our

### Fairness experiment

Method	Adult dataset		German dataset	
	Acc $\uparrow$	EO $\downarrow$	Acc $\uparrow$	EO $\downarrow$
FERM (30)	0.81	0.01	$0.73 \pm 0.04$	$0.05 \pm 0.03$
NN (77)	0.84	0.14	$0.74 \pm 0.04$	$0.47 \pm 0.19$
NN + $\chi$ (77)	0.83	0.03	$0.73 \pm 0.03$	$0.25 \pm 0.14$
LAFTR (74)	0.84	0.10	–	–
<i>Ours</i> ( $\lambda = 0.5$ )	$0.85 \pm 0.01$	$0.03 \pm 0.02$	$0.74 \pm 0.04$	$0.06 \pm 0.05$

Table 4.3 **Fairness experiments** – We compare against results on the two datasets as reported in (30; 74; 77). For accuracy, the higher the better. For EO, the lower the better (*i.e.*, the “fairer”). Our results were averaged across 10 different runs.

results with the ones published by related works (4; 53), using accuracy as a metric. We limited the training sets to only 2,000 samples: this choice was due to the fact that with the whole training sets we could observe baselines ( $\lambda = 0.0$ ) significantly higher than published results (53), whereas they are comparable for models trained on a subset.

**Results.** Table 4.2 reports our results. In all our experiments, we observe accuracy improvements with respect to the baseline ( $\lambda = 0.0$ ). In general, training on one split and testing on the other is more challenging than testing on the (neutral) test set, as confirmed by the baseline results (ERM, first row). In all the different protocols, our method (last row) has superior performance than Alvi et al. (4), and comparable or superior performance with Kim et al. (53).

These results confirm that our method can effectively remove biased, detrimental information even when modeling more complex data with higher-capacity models. In this case though, the improvements are more limited than the ones we showed in the digit experiment. One of the reasons might be that age and gender cannot be decoupled as efficiently as shape and color. In other words, removing age information may not necessarily increase accuracy.

### 4.5.3 Learning Fair Representations

**Experimental setup.** We explored the potentiality of our method in the context of algorithmic fairness with the popular UCI datasets Adult (adu) and German (ger). Both datasets contains tabular data with categorical and continuous attributes: Adult has more than 48,000



US adult Census data samples and the goal is to predict whether the person has a annual salary  $> 50K$ ; German is composed of 1,000 samples of bank customer descriptions and the binary, ground truth label is the risk degree associated with a customer, either good or bad. The goal is to learn a model to solve tasks with the constraint of removing sensitive information about gender in Adult and customer age in German. This problem is different with respect to the previous ones: here the invariance towards sensitive attribute does not imply a better generalization on the test set as it happens with, , digit recognition. The removal of the protected attribute is done for the sake of learning a fair representation (30; 57; 125; 128).

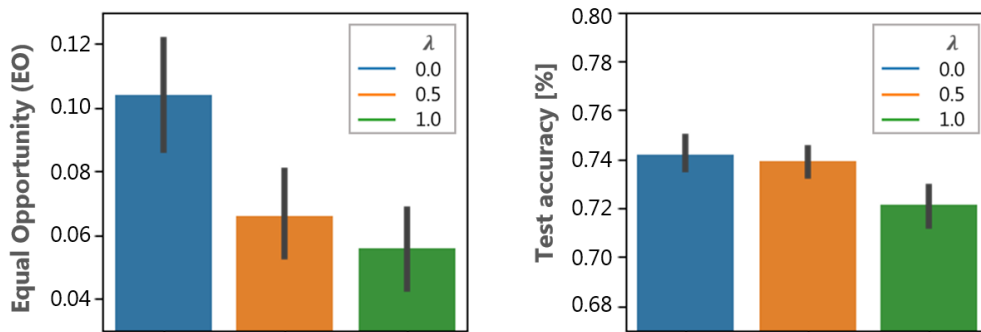


Figure 4.5 The two considered metrics vary as we modify the hyper-parameter  $\lambda$  on the German dataset. EO (Left) is significantly reduced as we set higher values of  $\lambda$ . Vice versa, test accuracy (Right) is only slightly affected.

Following previous works (30; 74), we implemented the feature extractor as single-layer MLP. Additional details can be found in Appendix B. We evaluate accuracy and “equal opportunity” (EO) as comparison metrics, averaging across 10 different runs: Equal Opportunity measures the discrepancy between the True Positive(TP) rates of “protected” and “non-protected” populations,  $EO = |TP(\text{young}) - TP(\text{not young})|$ . The goal is to find a balance between reducing EO (*i.e.*, learning a fairer representation) without observing a too severe decrease in accuracy.

**Results.** In Figure 4.5, we show how the performance varies when increasing  $\lambda$  from 0 (standard Empirical Risk Minimization) to 1 for models trained on German. It can be observed that our method allows training fairer models (*i.e.*, reduced EO), while maintaining a good performance on test. For  $\lambda = 0.5$ , the fairness price is close to zero (, the accuracy does not decrease), while the fairness is substantially improved. We report the comparison with related works in Table 4.3, for both datasets. These results show that our method can be effectively used to tackle algorithmic fairness: we achieve a favorable fairness trade-off, matching or exceeding test accuracy while keeping EO lower than the competitor methods.

## 4.6 Conclusions

We propose a training procedure to learn representations that are not biased towards dataset-specific attributes in an alternative fashion to existing adversarial approaches. We leverage a neural estimator for the mutual information (11), devising a method that can be easily implemented in arbitrary architectures and provides a robust strategy for hyperparameter tuning. We show competitive results on benchmarks ranging from computer vision (4; 53; 81) to fair representation learning (30; 74; 77).

The main limitation of this line of approaches is that the dataset bias needs to be explicitly provided in order to be addressed (the attribute  $\mathbf{c}$  in our formulation). This naturally limits the applicability of our method to scenarios where this information is not available. We addressed this problem in Chapter 5.

# Chapter 5

## Learning to learn Unbiased Models from Biased data

### 5.1 Context

In classification tasks, it is widely recognized that deep learning architectures can learn large amount of data, reaching unprecedented outstanding performance. However, such models are also very sensitive to data, meaning that they are prone to errors with high confidence whenever test samples are drawn from a distribution different from that of the training set. One reason is that, in certain conditions, these models have problems to generalize well the classes considered as they likely memorize the training data rather than learning the salient characteristics of each category of examples. This behavior is especially evident when training data are biased, *i.e.*, samples include spurious correlations with class labels or, in other words, the trained model learns some “shortcuts” to classify data, so failing to generalize the class properly (10; 34). For example, a fish can be classified as such due to the presence of the blue sea in which fishes are typically depicted, and not for the actual fish structure and appearance, hence a model may likely fail in case the input image depicts a fish located on a brown market table. Such shortcuts are learnt since most of the samples are characterized by a bias (fishes in the sea) while only a few samples are unbiased (fishes in unusual contexts), which prevents from proper generalization.

When optimizing models under the presence of biased data, the ground-truth knowledge of the bias is typically beneficial. For instance, having an additional annotation regarding whether the fish is in the sea or not can be used to drive the optimization towards a data representation invariant to such attribute (See Figure 5.1(a)). Several methods approached the problem in this way and sought for a data representation invariant to a known factor

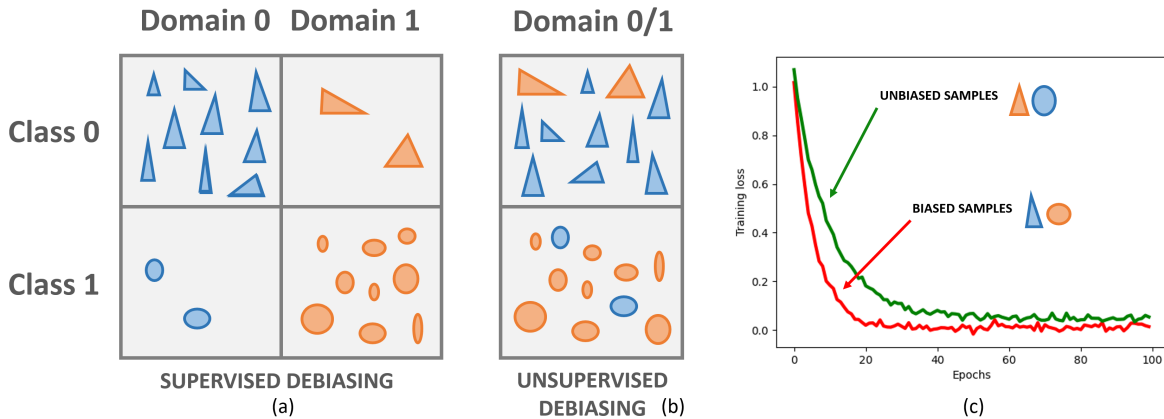


Figure 5.1 **Problem description.** (a) Biased dataset occurs when there is an imbalance regime regarding pairs (class, domain), where each class is observed mostly under one distribution, leaving other options under-represented. This results in trained models which do not generalize well. In the case of supervised debiasing case, one has additional annotations regarding the domain distribution. (b) In the unsupervised debiasing case, one has only access to the class labels. A possible approach to distinguish biased/unbiased samples is via pseudo-labeling. (c) The plots show that the loss for the biased samples are decreasing much faster than the loss for unbiased samples, proving that the former can be learnt more easily than the latter.

(4; 6; 24; 53; 66; 92; 99; 123): we term this problem *supervised debiasing*, *i.e.* the knowledge of the bias acts as an auxiliary data annotation that can be useful to consider in training in order to get invariance with respect to it. However, the hypothesis of having an additional label is unrealistic in most practical scenarios as it requires great effort during data annotation, and in some cases can even be impossible whenever the control of data gathering is unfeasible, hence the urge of methods that can generalize even without this additional supervision.

For these reasons, in this chapter we face the more challenging setting of the *unsupervised debiasing* problem: assuming that the ground-truth knowledge of the bias is not readily available, we attempt to (implicitly) infer this information while debiasing our model and achieving a successful generalization on the test set (See Figure 5.1(b)).

We devised a two-stage algorithm to tackle the unsupervised debiasing problem. First, we separate biased from unbiased samples through a pseudo-labeling approach. Second, equipped with such (noisy) pseudo-labels, we manage the problem of learning from this data using a Meta-Learning approach (inspired by Finn et al. (31)) to produce data representation that can accommodate both biased and unbiased samples. While existing methods (24; 85) focused primarily on increasing performance on unbiased samples, overlooking the need for keeping high accuracy on biased samples as well, we aim instead at achieving high accuracy over both types of data.

To this end, grounding on the intuition that meta-learning is a suitable approach to learn effectively different tasks, we propose to treat the learning from biased and unbiased data as different (meta-)training tasks, followed by a meta-validation step devoted to produce data representations which are suitable for both, aimed at better generalization. For the latter, we generate (augment) new data by linearly interpolating (129) biased and unbiased samples, so producing samples which are more “neutral” than the original biased and unbiased images, so reducing the contribution of spurious correlations in the prediction (See Fig. 5.1(c)) and overall regularizing the training. We can generate more “neutral” representations by mixing biased and unbiased samples even if they are not perfectly subdivided by the initial pseudo-labeling stage. In other words, the method is robust to some level of contamination between the estimated biased and unbiased subsets. Interestingly, this is a notable characteristic of our approach making it suitable in realistic scenarios. In fact, we do not need to perfectly identify the biased/unbiased samples, nor knowing or determining the bias affecting the data: the splitting performed by any pseudo-labeling algorithm can be managed by the subsequent meta-learning and data augmentation stage, to regularize the training.

We validate our method on several benchmarks that are both synthetic with controlled bias (colored MNIST and Corrupted CIFAR-10) and more realistic (Waterbirds and BAR), showing outstanding performance as compared with existing methods.

To recap, the contributions of our work are:

- We face the challenging unsupervised debiasing problem by introducing a two-stage approach that, after the initial coarse identification of the biased and unbiased samples, can modulate the contribution of each example during the model training by a meta-learning strategy.
- Specifically, our novel approach considers learning from biased and unbiased samples as separate meta-training tasks, while *generating* new data by augmentation, managed as a (meta-)validation task. By jointly optimizing the original meta-training and the meta-validation tasks, we inject a strong regularization in the training process, so compensating the imbalance problem between biased/unbiased samples by neutralizing the bias, and ultimately leading to more general representation learning.
- Our approach, validated on datasets with controlled bias and realistic benchmarks, showed to outperform state-of-the-art performance by a significant margin.

The rest of the chapter is organized as follows. In Section 2, we describe the related literature, highlighting the original aspects introduced. Section 3 reports our method, where we detail

our two-stage approach. Section 4 presents the results and a thorough ablation analysis. Section 5 wrap-ups the work and sketches future research directions.

## 5.2 Related works

Learning from biased data can be seen as a specific case of Out-Of-Distribution (OOD) domain generalization. This topic has been addressed with different methodologies, including meta-learning. Here, we briefly review the most related literature.

**Learning from biased data.** The problem of learning from biased data has been explored in past years in the supervised debiasing setting, i.e. when labels for the factor (bias) to be removed are available. Several methods approached the problem seeking an invariant data representation to a known factor. Such approaches rely on adversarial learning (4; 53; 128), variational inference (26; 72; 82), Information Theory (92), re-sampling strategies (66), or robust optimization (99). Invariant Risk Minimization (6) seeks an optimal representation which is invariant across domains.

Few recent works (8; 63; 68; 85) have addressed the unsupervised debiasing problem. (8) formalizes the *cross-bias* problem where malicious shortcuts exist, easing the fit of training data, whereas the same shortcuts result harmful at inference stage. The solution is learning a debiased model which is statistically independent from the one computed by a parallel computational stream that is guaranteed to be affected by the bias by design. In (85), the nature of the aforementioned “shortcuts” is analyzed in terms of fitting speed at training time. Nam et al. show that biased samples are learnt faster than the unbiased ones. The relative difficulty of each sample is cast into a weight that modulates its learning rate: in this way, at training time, it is given more importance to the few outlying samples that do not follow the shortcuts. To this end, an ensemble of networks is trained, similarly to (8). (63) tackles the problem via robust optimization, considering a worst case loss of a sub-population of the dataset (typically samples with the highest loss). In (54; 68), the training data is split in two subsets, relying on the predictions of a baseline model. In (68), the most difficult samples (likely those that do not follow shortcuts) are then upsampled. (54) identifies patches from the two splits and then swaps them in order to produce additional samples with which to train the debiased model.

Our work does not rely on an ensemble of networks to have a reference biased model. Instead, we perform a pseudo-labeling approach to split the dataset in two subsets and then treat them as two separate tasks to be learned via meta-learning. We also avoid data upsampling as in (68) and (66), whereas we pursue a data augmentation approach to combine

biased and unbiased samples. Inspired by Mixup (129), we mix factors which are peculiar of the bias regime (likely representing a shortcut to infer the class) with those that do not follow such rules. The newly generated samples are expected to break the spurious correlations that affect the original data.

**Meta-Learning for Out-Of-Distribution domain generalization.** A class of meta-learning methods based on bi-level optimization (*e.g.*, Model Agnostic Meta-Learning (31)), relies on an inner-loop stage optimizing model’s meta-parameters on source data, and an outer-loop stage that updates the model parameters on (meta-)validation data. This nested optimization which involves computing a gradient through a gradient, has been shown to be effective for a fast adaptation of the model to the validation data. The goal is learning from an (empirical) training task distribution so to generalize and learn faster (*i.e.*, with fewer samples) the validation task. Subsequently, other methods have tackled the problem of Domain generalization (DG) ((9; 65; 67), to cite a few), casting the problem of learning from multiple tasks to learning from multiple distributions/domains.

We adopt the same general scheme, however we face a considerably distinct problem: while in DG, different domains are fairly balanced, we deal with a severe data imbalance, that is, biased vs. unbiased, seen here as domains. This domain data imbalance is so dramatic that the model likely learns domain attributes to perform inference, hampering its generalization capabilities. This requires a tailored solution that we found effective through data augmentation, in order to attempt to reduce the imbalance problem. Moreover, differently from previous methods that rely on multiple source domains, we relax the hypothesis of having domain labels and adopt a pseudo-labeling approach to overcome this issue. Instead, we apply a pseudo-labeling method to discriminate the training set in two subsets that corresponds to different distributions. Additionally, since one needs a meta-validation set to train the outer loop, we produce synthetic validation data using data augmentation. In fact, since one needs a meta-validation set to train the outer loop, our solution is to produce it, by generating synthetic validation data using data augmentation. This resulted quite effective even if the two subsets are noisy, that is, even if they do not perfectly identify the real distributions.

## 5.3 Method

We consider supervised classification problems with a training set  $\mathcal{D}_{train} = \{x_k, y_k, d_k\}_{k=1}^N$ , where  $x_k$  are raw input data,  $y_k$  class labels and  $d_k$  domain labels. In the case of a biased dataset,  $\mathcal{D}_{train}$  has several classes  $y^i, i = 1, \dots, C$ , which are considered to be observed under

different domains  $d^j, j = 1, \dots, D$ ;  $D$  can be different from  $C$  but here, for clarity and without losing generality, we consider the case of  $D = C$ . When the majority of samples of a specific class  $y^i$  is observed under a single domain  $d^j$ , while other domains are under represented in the dataset, we say that the pair  $(y^i, d^j)$  is biased, *i.e.* there is a spurious correlation between class and domain.

We define  $\mathcal{D}_{bias}$  as the subset of training samples that exhibit spurious correlations and  $\mathcal{D}_{unbias}$  as the subset of samples with under represented pairs. Such subsets are highly imbalanced, *i.e.*  $|\mathcal{D}_{bias}| \gg |\mathcal{D}_{unbias}|$ . For instance, in a cats vs. dogs classification problem, most of the cats may be observed in an indoor home environment, while most of the dogs may be observed in outdoor scenes. For both classes, very few images are outside of the main distribution.

We aim to tackle the *unsupervised debiasing* problem, which means that we do not have access to domain labels  $d$  nor to other bias information, hence we consider a training set containing only input data and class label,  $\mathcal{D} = \{x_k, y_k\}_{k=1}^N$ . We want to train a neural network  $f_\theta$  on  $\mathcal{D}$ , with parameters  $\theta$ , to be deployed on test data  $\mathcal{D}_{test}$  not seen during training.  $\theta$  are usually found via Empirical Risk Minimization (ERM), *i.e.* minimizing the expected Cross-Entropy loss over the training data:

$$\min_{\theta} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim \mathcal{D}} \mathcal{L}(\mathcal{D}, f_\theta) \quad (5.1)$$

$$\mathcal{L}(\mathcal{D}, f_\theta) = \mathbf{y}^T \log(\sigma(f_\theta(\mathbf{x})))$$

where  $\sigma$  is the softmax function. In such scenario, when trained via ERM, a model focuses mostly on the more numerous biased samples, underfitting the unbiased ones: this results in a biased model that uses spurious correlation (*e.g.*, background) as a possible way to make inference, instead of correctly learning the class semantic. In general,  $\mathcal{D}_{test}$  follows a data distribution different from  $\mathcal{D}_{train}$ , *i.e.* the biased pairs may be not the majority of samples. Hence it is important to have a model that can be deployed on both biased and unbiased pairs.

Our method tackles the unsupervised debiasing problem with a two-stage approach. In the first stage, we separate biased from unbiased samples through a pseudo-labeling algorithm. Equipped with such pseudo-labels, we train a model to produce a data representation that can accommodate both biased and unbiased samples. In the following, we detail the two main stages of our method.



### 5.3.1 Bias Identification

In this stage, our goal is to split the training set  $\mathcal{D}$  into two disjoint subsets  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$  that should resemble the actual, ground-truth  $\mathcal{D}_{bias}$  and  $\mathcal{D}_{unbias}$ . In (85), it is shown how the biased samples are learnt faster than the unbiased ones: the imbalanced nature of the dataset makes the model more prone to learn first the numerous biased samples and later those unbiased. This behaviour can be observed by looking at the loss function trends of the two subsets (See Fig. 5.1(c)). We exploit the fact that samples from  $\mathcal{D}_{bias}$  are easily learnt during training, to design a strategy for splitting the dataset. We train a neural network  $f_\phi$  via ERM until it reaches a training accuracy of  $\gamma$ , where  $\gamma$  is a hyper-parameter denoting the target accuracy. When the model reaches the desired accuracy level, the training stops and a forward pass of the entire training set is performed. Now, samples that are correctly predicted are assigned to  $\hat{\mathcal{D}}_{bias}$  while those not correctly predicted are assigned to  $\hat{\mathcal{D}}_{unbias}$ . More formally:

$$\begin{aligned}\hat{\mathcal{D}}_{bias}^\gamma &= \{(x, y) \in \mathcal{D} \mid \sigma(f_\phi^\gamma(x)) = y\} \\ \hat{\mathcal{D}}_{unbias}^\gamma &= \{(x, y) \in \mathcal{D} \mid \sigma(f_\phi^\gamma(x)) \neq y\}\end{aligned}\tag{5.2}$$

Using  $\gamma$  as hyper-parameter is convenient for two reasons. First, our setting of the amount of desired accuracy is dataset agnostic. This is different from prior work (68) that employs a similar strategy, but with the hyper-parameter controlling the number of epochs to train the model: in that case, the number of epochs are strictly dependent on the dataset that the model is trained on. Second, we can have a precise control of the amount of samples assigned to the two splits, *e.g.*  $\gamma = 0.85$  implies that 85% of training data are assigned to  $\hat{\mathcal{D}}_{bias}$  and 15% to  $\hat{\mathcal{D}}_{unbias}$ . In real use cases, we do not know the correct assignments of the samples to the splits, so we have to rely on a priori setting of this parameter.

### 5.3.2 Bias-invariant representation learning

Provided with pseudo-labels for the two estimated subsets  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$ , we deal with the problem of learning data representations that are not only good for the biased data but can generalize well to unbiased samples. We adopt a neural network  $f_\theta$ , trained from scratch, and we designed a bi-level optimization algorithm inspired by meta-learning to learn efficiently from such data.

**Inner loop step.** This is a meta-training step where we seek the best parameters  $\theta$  for the two subsets  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$  via gradient descent:

$$\theta^* = \theta - \eta \nabla_{\theta} [(1 - \gamma) \mathcal{L}(\hat{\mathcal{D}}_{bias}, f_{\theta}) + \gamma \mathcal{L}(\hat{\mathcal{D}}_{unbias}, f_{\theta})] \quad (5.3)$$

where  $\eta$  is the learning rate. In this step, the two splits of the training data are treated as two separate tasks: we scale the two loss functions with two coefficients to deal with data imbalance ( $|\hat{\mathcal{D}}_{bias}| \gg |\hat{\mathcal{D}}_{unbias}|$ ). To rebalance the contributions from the two splits, an obvious choice is to set weights inversely proportional to the cardinality of the two subsets, which is nothing else than the fixed and controllable hyper-parameter  $\gamma$ .

**Outer loop step.** Standard meta-learning usually optimizes for the meta-test task using the parameters found in the inner loop, relying on a (typically small and clean) validation set. Here, we get rid of this assumption since do not have access to any held-out nor clean data, therefore we opt for a data augmentation approach in order to provide unseen data to the model.

We seek a representation that can conciliate both biased and unbiased samples and at the same time prevent the model from overfitting the meta-training data (the two subsets  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$ ), which is a common problem in meta-learning. We take inspiration from Mixup (129) as a way to combine samples from the two subsets. Mixup provides a convex combination of both input samples and labels and it has demonstrated its efficacy as an effective regularizer. Specifically, we feed the model with samples resulting from the mix of examples from biased and unbiased data, aiming at likely breaking the shortcuts present in the dataset (see Fig. 5.2).

We construct  $\hat{\mathcal{D}}_{mix}$  by mixing samples of  $\hat{\mathcal{D}}_{bias}$ ,  $\hat{\mathcal{D}}_{unbias}$ , sampling the parameter  $\lambda \sim \text{Beta}(\alpha, \beta)$ :

$$\begin{aligned} x_{mix} &= \lambda \hat{x}_1 + (1 - \lambda) \hat{x}_2 \\ y_{mix} &= \lambda \hat{y}_1 + (1 - \lambda) \hat{y}_2 \\ (\hat{x}_1, \hat{y}_1) &\in \hat{\mathcal{D}}_{bias}, (\hat{x}_2, \hat{y}_2) \in \hat{\mathcal{D}}_{unbias} \end{aligned} \quad (5.4)$$

Computed the augmented samples  $x_{mix}, y_{mix}$ , the model is updated in the outer loop:

$$\mathcal{L} := \underbrace{(1 - \gamma) \mathcal{L}(\hat{\mathcal{D}}_{bias}, f_{\theta}) + \gamma \mathcal{L}(\hat{\mathcal{D}}_{unbias}, f_{\theta})}_{\text{Weighted ERM}} + \zeta \underbrace{\mathcal{L}(\hat{\mathcal{D}}_{mix}, f_{\theta^*})}_{\text{Regularizer}} \quad (5.5)$$

where  $\zeta$  is a hyper-parameter controlling the regularization. Note that the first two losses are evaluated on the current parameters configuration  $\theta$ , while the loss over the augmented samples is evaluated in the meta-state  $\theta^*$  (see Eq. 5.3). This implies that the model has to compute a gradient through a gradient, similarly to what happens in optimization-based

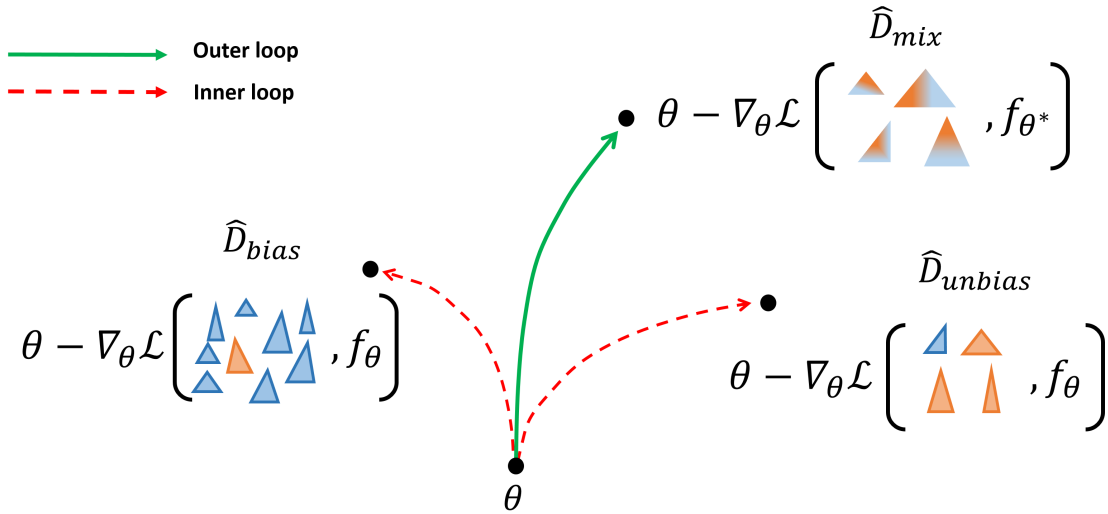


Figure 5.2 Starting from the current parameter configuration  $\theta$ , gradients on  $\mathcal{L}(\hat{\mathcal{D}}_{bias}, f_{\theta})$  and  $\mathcal{L}(\hat{\mathcal{D}}_{unbias}, f_{\theta})$  are evaluated to produce the new configuration  $\theta^*$ . The regularization step using mixed data aims at producing a contribution that decreases the loss function on  $\hat{\mathcal{D}}_{bias}$ ,  $\hat{\mathcal{D}}_{unbias}$ , and  $\hat{\mathcal{D}}_{mix}$ , simultaneously, the latter estimated over the configuration  $\theta^*$ . (Best viewed in color)

meta-learning methods. The hyperparameter  $\zeta$  controls the amount of regularization in the final loss: if  $\zeta = 0$ , the method corresponds to a (weighted) ERM in which the contributions of the losses on the two subsets are scaled by  $(1 - \gamma)$  and  $\gamma$ . When  $\zeta > 0$  the weighted ERM optimization trajectory is corrected by the regularization term. This corresponds to find parameters  $\theta$  that are good for both  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$ , but can also possibly reduce the loss value on the newly generated data samples  $\hat{\mathcal{D}}_{mix}$ . Accuracy is not so affected by the choice of the  $\zeta$  value: indeed, it increases as long as this  $\zeta$  assumes positive values up to reaching high performance quite steadily, after that the contribution of the regularization becomes too strong and accuracy decreases. We set the value of  $\zeta$  to a fixed value ( $= 10$ ) for all experiments. Further analysis on this parameter is reported in Appendix C. The complete method is summarized in Algorithm 3.

## 5.4 Experiments

We show the effectiveness of models trained by our method in a series of benchmarks, ranging from toy problems with synthetic biases to more realistic image classification tasks. We compare with methods that tackle the same bias problem in both supervised and unsupervised way.

**Algorithm 3** Learning to learn unbiased representations

- 
- 1: **Input:** Dataset  $\mathcal{D}$ , initialized weights  $\theta_0$ , learning rate  $\eta$ , hyper-parameters  $\zeta, \gamma, T$ .
  - 2: **Output:** learned weights  $\theta$
  - 3: **Initialize:**  $\theta \leftarrow \theta_0$
  - 4: **Identify**  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$  by pseudo-labeling (Eq. 5.2)
  - 5: **for**  $t = 1, \dots, T$  **do**
  - 6: Sample  $(\mathbf{x}_b, \mathbf{y}_b), (\mathbf{x}_u, \mathbf{y}_u)$  from  $\hat{\mathcal{D}}_{bias}, \hat{\mathcal{D}}_{unbias}$
  - 7: Compute  $\theta^*$  (Eq. 5.3) ▷ Inner loop step
  - 8: Sample  $(\hat{\mathbf{x}}_1, \hat{\mathbf{y}}_1), (\hat{\mathbf{x}}_2, \hat{\mathbf{y}}_2)$  from  $\hat{\mathcal{D}}_{bias}, \hat{\mathcal{D}}_{unbias}$
  - 9: Construct  $\hat{\mathcal{D}}_{mix}$  (Eq. 5.4) ▷ Produce augmented samples
  - 10: Update  $\theta$  (Eq. 5.5) ▷ Outer loop step
  - 11: **end for**
- 

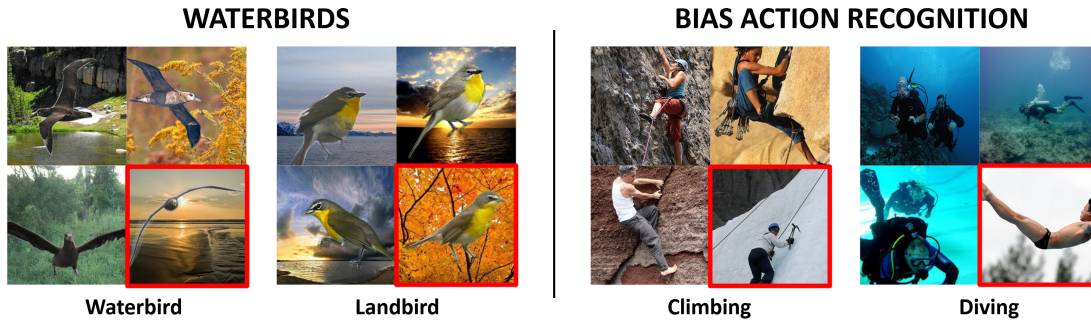


Figure 5.3 Examples of biased training data and unbiased data (with red boundary) from Waterbirds and BAR.

### 5.4.1 Benchmarks

**Synthetic bias datasets.** To control the bias in the data and for the sake of comparison, we adopt two benchmarks that have been employed by Nam et al. (85), namely Colored MNIST and Corrupted CIFAR-10<sup>1,2</sup>. The first is a modified version of the standard digit recognition dataset (62), in which colors are added in order to artificially induce a bias in the dataset. The dataset is made of 60,000 training RGB images and 10 classes. Each sample is colored with a color tone which is randomly sampled from a Gaussian distribution whose mean is specific for each class; in practice, each class in the training data is observed mostly under a certain color tone, while the test set has no specific correlation between classes and colors and is balanced. Corrupted CIFAR-10 is a modification of the original dataset (59) that has been introduced in (44). There are 50,000 training RGB images and 10 classes. The bias here stems from the fact that each image is corrupted with a specific noise (*e.g.*, Gaussian blur, salt and pepper noise, etc.). Specifically, each class has a privileged type of noise under which it

is observed during training (*e.g.*, most of car images are corrupted with motion blur). There are two versions of this dataset, namely Corrupted CIFAR-10<sup>1</sup> and Corrupted CIFAR-10<sup>2</sup>, with different types of noise.

**Realistic bias datasets.** We tested our method on real image datasets, Waterbirds and Bias Action Recognition (BAR). Waterbirds has been introduced by (99) and combines bird photos from the Caltech-UCSD Birds-200-2011 (CUB) dataset (126) with background images from the Places dataset (131). There are 4,795 training images and the goal is to distinguish two classes, namely *landbird* and *waterbird*. The bias here is represented by the background of the images: most landbirds are observed on a land background while most waterbirds are observed on a marine background. BAR has been introduced by (85) as a realistic benchmark to test model’s debiasing capabilities. It is constructed using several data sources and consists of 1,941 photos of people performing several actions, and the task is to distinguish them in 6 classes: Climbing, Diving, Fishing, Racing, Throwing and Vaulting. The bias arises from the context in which action photos are observed at training: for instance, climbing actions are performed in a dry mountain scenario at training time, whereas in the test set, they are set in a snowy environment. For details, readers can refer to the original paper (85).

## 5.4.2 Performances

We report the performance of our approach on the different benchmarks, starting from the Colored MNIST and corrupted CIFAR-10 since they have controlled bias for which we can better discuss the results. Since we deal with biased training data and balanced data in testing, we report both accuracies on the testing subset of unbiased samples only (those under-represented in the training data) as well as over the entire test set (biased + unbiased), to assess how much we lose on the biased samples. In fact, as we learn features having higher generalization capacity, spurious correlations are likely less exploited to classify biased examples, and this may cause a drop in performance on such samples.

For Colored MNIST, our network  $f_{\theta}$  is an MLP with 3 hidden layers with 100 neurons each. We used ResNet-18 (43) as a backbone for Corrupted CIFAR-10 and BAR, and ResNet-50 as backbone for Waterbirds. We remove the last layer from such backbones, adding a 2-layer MLP head on top of it. ResNet is pre-trained on ImageNet (28). The meta-parameter  $\theta^*$  is computed only for the last two fully connected layers while the backbone is trained with only the contribution of the weighted ERM in Eq. 5.5 ( $\zeta = 0$ ). We set the learning rate  $\eta = 0.001$  for all datasets with batch size of 256 on synthetic biased data and 128 for realistic bias data. We used Adam (55) as optimizer. All the experiments comply the same

Dataset	Bias ratio	ERM	REPAIR (66)	Group-DRO (99)	LfF (85)	Ours, $\zeta = 0$	Ours, $\zeta = 10$
Colored-MNIST	95%	77.6 ± 0.44	82.5 ± 0.59	84.5 ± 0.46	85.3 ± 0.94	82.3 ± 0.99	<b>89.3 ± 1.02</b>
	98%	62.3 ± 1.47	72.9 ± 1.47	76.3 ± 1.53	80.5 ± 0.45	73.8 ± 0.87	<b>83.4 ± 0.97</b>
	99%	50.3 ± 0.16	67.3 ± 1.69	71.3 ± 1.76	74.0 ± 2.21	68.3 ± 0.98	<b>81.6 ± 0.96</b>
	99.5%	35.3 ± 0.13	56.4 ± 3.74	59.7 ± 2.73	63.4 ± 1.97	57.1 ± 1.05	<b>72.2 ± 0.87</b>
Corrupted CIFAR-10 <sup>1</sup>	95%	45.2 ± 0.22	48.7 ± 0.71	53.1 ± 0.53	59.9 ± 0.16	54.3 ± 1.40	<b>63.3 ± 1.11</b>
	98%	30.2 ± 0.77	37.9 ± 0.22	40.2 ± 0.23	49.4 ± 0.78	44.4 ± 0.90	<b>56.2 ± 0.89</b>
	99%	22.7 ± 0.97	32.4 ± 0.35	32.1 ± 0.83	41.4 ± 2.34	33.4 ± 0.91	<b>50.5 ± 0.98</b>
	99.5%	17.9 ± 0.86	26.3 ± 1.06	29.3 ± 0.11	31.7 ± 1.18	26.1 ± 0.94	<b>43.3 ± 0.97</b>
Corrupted CIFAR-10 <sup>2</sup>	95%	41.3 ± 0.46	54.1 ± 1.01	57.9 ± 0.31	58.6 ± 1.18	53.8 ± 1.21	<b>62.5 ± 0.91</b>
	98%	28.3 ± 0.77	44.2 ± 0.84	46.1 ± 1.11	48.7 ± 1.68	43.2 ± 0.96	<b>55.2 ± 0.98</b>
	99%	20.7 ± 0.81	38.4 ± 0.26	39.6 ± 1.04	41.3 ± 2.08	37.0 ± 0.99	<b>49.8 ± 1.01</b>
	99.5%	17.4 ± 0.85	31.0 ± 0.42	34.2 ± 0.74	34.1 ± 2.39	30.6 ± 0.89	<b>43.6 ± 1.32</b>

Table 5.1 **Accuracy on whole test set.** Accuracy (in %) evaluated on biased + unbiased test samples for different bias ratios. Best performance in bold.

Dataset	Bias ratio	ERM	REPAIR (66)	Group-DRO (99)	LfF (85)	Ours, $\zeta = 0$	Ours, $\zeta = 10$
Colored-MNIST	95%	75.2 ± 0.87	83.3 ± 1.23	83.1 ± 0.81	85.8 ± 0.66	82.1 ± 0.88	<b>89.2 ± 1.09</b>
	98%	58.1 ± 0.56	73.4 ± 0.79	74.3 ± 1.09	80.7 ± 0.56	73.3 ± 0.73	<b>83.4 ± 0.85</b>
	99%	44.8 ± 0.84	68.3 ± 0.75	69.6 ± 0.63	74.2 ± 1.94	67.6 ± 0.92	<b>81.6 ± 0.79</b>
	99.5%	28.1 ± 0.45	57.3 ± 0.61	57.1 ± 0.78	63.5 ± 1.94	56.8 ± 0.79	<b>72.1 ± 0.94</b>
Corrupted CIFAR-10 <sup>1</sup>	95%	39.4 ± 0.75	50.0 ± 0.89	49.0 ± 0.48	59.6 ± 0.03	54.3 ± 0.89	<b>63.3 ± 1.10</b>
	98%	22.6 ± 0.45	38.9 ± 0.64	35.1 ± 0.92	48.7 ± 0.70	44.1 ± 0.83	<b>56.1 ± 0.92</b>
	99%	14.2 ± 0.91	33.0 ± 0.57	28.0 ± 0.68	39.5 ± 2.56	32.3 ± 0.84	<b>49.6 ± 0.85</b>
	99.5%	10.5 ± 0.28	26.5 ± 0.46	24.4 ± 0.48	28.6 ± 1.25	25.6 ± 0.91	<b>42.1 ± 0.88</b>
Corrupted CIFAR-10 <sup>2</sup>	95%	34.9 ± 0.84	54.5 ± 1.04	54.6 ± 0.61	58.6 ± 1.04	53.6 ± 0.86	<b>62.3 ± 1.04</b>
	98%	20.5 ± 0.64	44.6 ± 0.83	42.7 ± 0.77	48.9 ± 1.61	43.8 ± 0.84	<b>55.5 ± 0.98</b>
	99%	12.1 ± 0.75	38.8 ± 0.75	37.1 ± 1.22	40.8 ± 2.06	36.4 ± 0.93	<b>49.7 ± 0.94</b>
	99.5%	10.0 ± 0.84	31.4 ± 0.53	30.9 ± 0.89	32.0 ± 2.51	29.8 ± 0.91	<b>43.0 ± 0.85</b>

Table 5.2 **Results on unbiased test samples.** Accuracy (in %) evaluated only on the unbiased samples for different bias ratios. Best performance in bold.

evaluation protocol used in the competing methods for a fair comparison. Implementation details are reported in Appendix C.

**Results for synthetic bias datasets.** Tables 5.1 and 5.2 present the performance on synthetic biased datasets, reporting the overall average accuracy and the one for unbiased samples only, respectively. We compare against two baselines, a model trained by Empirical Risk Minimization (ERM) and our method with  $\zeta = 0$ , which cancels out the contribution of the regularization brought by the outer loop step in Eq. 5.5. This second baseline only weighs the contributions of the two splits found via pseudo-labeling. We also compare our approach with several former methods to learn unbiased representations, either using annotation for the bias or not. For the methods requiring explicit knowledge of the bias, we consider REPAIR

(66), which does sample upweighting, and Group-DRO (99), which tackles the problem using robust optimization. We finally report the performance of Learning from Failure (LfF) (85), which learns a debiased model without exploiting the labeling of the bias.

We consider different ratios of the bias (ranging from 95% up to 99.5%) as in (85). This ratio indicates the actual percentage of the dataset belonging to  $\mathcal{D}_{bias}$  and  $\mathcal{D}_{unbias}$ . This ratio is also linked to the parameter  $\gamma$  used in the pseudo-labeling stage, but in actual use cases it is not known. Hence, in all experiments, we made an arbitrary, largely loose choice for it, and fix the hyper-parameter  $\gamma = 0.85$ , *i.e.* we consider 85% of the training data as biased, and then assigned to  $\hat{\mathcal{D}}_{bias}$ , and the remaining 15% to  $\hat{\mathcal{D}}_{unbias}$ . Since  $\gamma$  is a sensitive parameter, we provide an ablation analysis in which we show how the performance changes as  $\gamma$  varies (see Section 5.4.3 below). We set  $\zeta = 10$  throughout all the experiments: in Appendix C, we report an ablation about this parameter.

We observe consistent better results with respect to the competitors, for all datasets and all possible bias ratios. Interestingly, the difference from the baselines increases as the dataset is more biased (higher bias ratio): this indicates that our method is more effective as the bias is more severe. We note that these performances are reached starting from a very coarse split of the data (85/15%), while the actual biased/unbiased sets are much more corrupted (from 95/5% to 99.5/0.5%): this robustness towards the pseudo-labeling subdivision is particularly useful in actual scenarios where the bias ratio is unknown. We report an ablation study in this regard in Section 5.4.3. The weighted ERM ( $\zeta = 0$ ) is already a strong baseline that surpasses, in some cases, former debiasing methods. Please, note that for both the unbiased samples and, in average, over the whole test set, the improvement is significant by a large margin. This shows that our method is not only better at generalizing over unbiased samples, but also maintains high accuracy on the biased set.

**Results on the realistic biased datasets.** In these trials, we still compare against the ERM baseline and Group DRO, as supervised method as before, and four unsupervised algorithms, LfF (85), CVaR DRO (63), ReBias (8), and JTT (68). Performances are reported in Table 5.3. For these datasets, we remind that we do not have the full control of the bias ratios. Specifically, in BAR we do not know exactly the biased/unbiased samples and, differently from Colored MNIST and Corrupted CIFAR-10, which have a balanced test set, Waterbirds test set is also imbalanced. In these cases, it is also important not only to cope with unbiased samples, but also to maintain accuracy on biased data. Hence, we aim here at finding a good trade-off between generalizing to unbiased samples while keeping high performance on biased data as well: that is why performances in Table 5.3 are reported as accuracies over both the entire test set (avg) and the unbiased samples for Waterbirds, and over the whole

Method	Bias supervision	Waterbirds		BAR
		Acc. avg	Acc. unbiased	Acc. avg
ERM	No	97.3%	72.6%	53.5%
CVaR DRO (63)	No	96.0%	75.9%	-
LfF (85)	No	91.2%	78.0%	62.9%
ReBias (8)	No	-	-	59.7%
JTT (68)	No	93.3%	86.7%	-
Ours ( $\zeta = 10$ )	No	94.1%	86.9%	64.3%
Group DRO (99)	Yes	93.5%	91.4%	-

Table 5.3 Performance on the whole (avg) and unbiased only test set: comparisons with baseline, unsupervised and supervised methods (see text).

test set only (avg) for BAR. For Waterbirds, we note that we score favorably with respect to other unsupervised methods for unbiased sample subset: we reach comparable performance with JTT, but we outperform it on the whole test set. In other words, we are able to learn bias invariant representations without giving up accuracy on the biased samples. ERM and CVaR DRO outperform ours as per the avg accuracy, but their accuracy drastically drops when considering unbiased samples only. We show also competitive performance against the supervised method Group DRO: without using any bias supervision, our method surpasses its average test accuracy even if the accuracy on biased data results lower (owing to the supervision in this case). Concerning the BAR dataset, since there is no ground-truth for the bias we report only the average accuracy over the whole test set: our method outperforms all other competitors by a considerable margin.

### 5.4.3 Ablation Study

We conducted an ablation analysis using Corrupted CIFAR-10<sup>1</sup> (bias ratio= 95%), to assess the contribution of each step characterizing our approach.

First, we test the robustness of the classification performance towards the choice of the hyper-parameter  $\gamma$  that governs the amount of data that we assign to the pseudo-labeled subsets (See Figure 5.4). We observe that by varying  $\gamma$  from 80% to 95%, the final accuracy



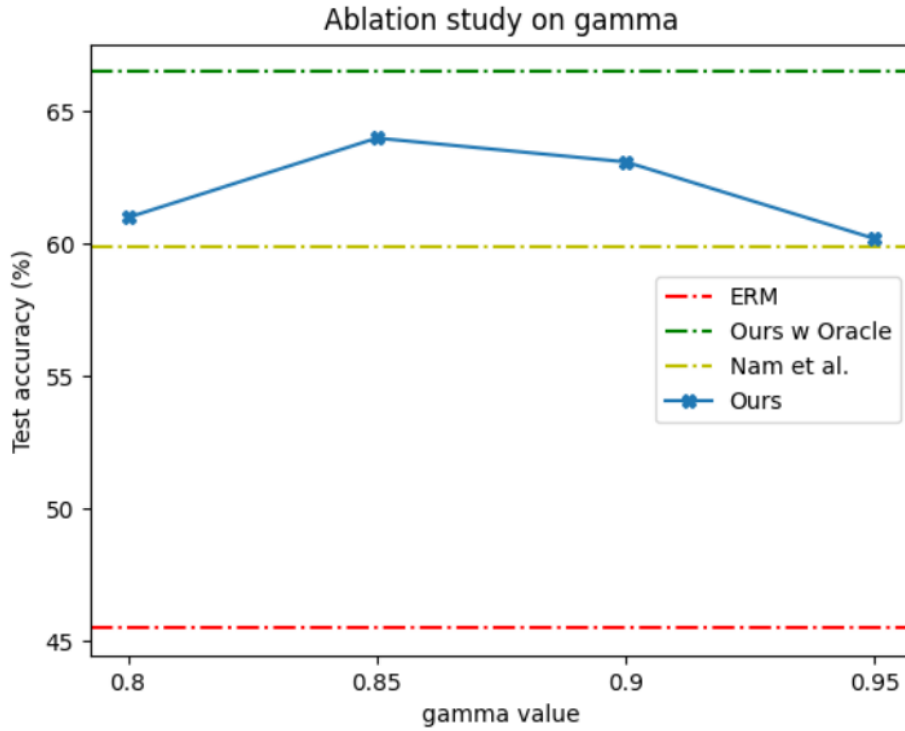


Figure 5.4 Test accuracies at different values of  $\gamma$  (from 0.8 to 0.95). We compare with ERM baseline, Nam et al. (85), and our method using the ground-truth bias knowledge as an oracle ( $\hat{\mathcal{D}}_{bias} = \mathcal{D}_{bias}$  and  $\hat{\mathcal{D}}_{unbias} = \mathcal{D}_{unbias}$ ).

does not change sensibly, meaning that the initial training of the network  $f_\phi$  is not a critical step since the biased training samples can be learnt faster than the unbiased ones.

We also assessed the quality of the splitting obtained, the influence of the pseudo-labeling (Eq. 5.2) on the final test accuracy. Considering F1-score (as a measure integrating Precision and Recall) as metric to evaluate the quality of the splitting, we estimate the test accuracies in the ideal case of perfect subdivision between biased and unbiased samples (Oracle,  $F1 = 1$ ), by applying our approach ( $F1 = 0.64$ ), and in case of random split ( $F1 = 0.37$ ). We noted that passing from the oracle conditions (best) to the random split (worst), accuracy drops of 4% in case of our method, and of 10% for the random split, showing a certain robustness to a coarse initial biased/unbiased sample subdivision.

Finally, we tested different strategies to perform data augmentation in the outer loop step. We combined samples from  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$  (Eq. 5.4). In Table 5.4 we report the results when sampling  $\hat{x}_1, \hat{x}_2$  from different combinations of the subsets. Mixing both samples from  $\hat{\mathcal{D}}_{bias}$  overfits the biased data and results in the worst accuracy, while mixing both samples from  $\hat{\mathcal{D}}_{unbias}$  increases the generalization over unbiased samples, but provides suboptimal results, especially for the biased subset. Samples from  $\hat{\mathcal{D}}_{bias}$  mixed with  $\hat{\mathcal{D}}_{unbias}$  corresponds

Set 1	Set 2	Bias ratio							
		95%		98%		99%		99.5%	
		Acc. all	Acc. unbiased	Acc. all	Acc. unbiased	Acc. all	Acc. unbiased	Acc. all	Acc. unbiased
No augmentation		58.8%	55.3%	46.1%	41.5%	40.0%	34.8%	33.6%	27.1%
$\hat{\mathcal{D}}_{bias}$	$\hat{\mathcal{D}}_{bias}$	35.2%	29.7%	34.0%	28.4%	32.9%	26.7%	32.0%	27.5%
$\hat{\mathcal{D}}_{unbias}$	$\hat{\mathcal{D}}_{unbias}$	60.2%	63.1%	54.1%	55.3%	48.4%	48.7%	40.4%	42.5%
$\hat{\mathcal{D}}_{bias}$	$\hat{\mathcal{D}}_{unbias}$	<b>63.8%</b>	<b>63.3%</b>	<b>56.4%</b>	<b>55.9%</b>	<b>50.9%</b>	<b>49.4%</b>	<b>43.1%</b>	<b>42.7%</b>

Table 5.4 **Ablation analysis on the augmentation strategies.** We report the accuracy resulting from different augmentation strategies and no augmentation, by varying the bias ratio. Our strategy results the winner over all the other mixing policies.

to our policy, which provides the best performance. We also report the baseline case in which no augmentation is performed (first row), *i.e.*  $x_{mix}, y_{mix}$  are just drawn from  $\mathcal{D}$ , whose results are significantly distant from our proposal. Additional details of these ablation analyses are reported in Appendix C.

## 5.5 Conclusions

We proposed a novel solution for the problem of unsupervised debiasing using a meta-learning strategy. After having subdivided by a pseudo-labeling method the training dataset into two subsets of biased and unbiased samples, we treated them as tasks to be learned with a bi-level optimization algorithm. The key idea is the mixing of the two subsets to provide the model with unseen data that can break the spurious correlations between data and class labels. As future directions, we point out two main problems. First, designing more refined strategies to perform the pseudo-labeling stage to fill the gap with the ground-truth bias label. Second, finding novel ways of combining biased and unbiased samples to allow the model to better generalize.

# Chapter 6

## Conclusions

In this thesis we address issues related to the dataset bias problem and propose novel algorithmic solutions to face them. This is an important problem as machine learning models are based on correlations observed in data: when such correlations are spurious, then models learn wrong patterns that do not encode the data semantics, thus leading to poor generalization. Another important issue arises when spurious correlations represent prejudices present in the data (e.g. towards a specific ethnicity or gender): This may lead to models which reflect societal bias towards specific groups, hence discriminating them when deployed. As machine learning technologies are becoming more and more pervasive into services and products that are used on a daily basis, it becomes crucial to have automatic decisions to be taken in a possible “neutral” way, i.e. in an unbiased way.

In Chapter 3 we pursue adaptation to a target unlabeled distribution, relying on pseudo-labels to cope with the absence of annotations for the target domain. We devise an iterative method that computes pseudo-labels for unlabeled samples and then train a Generative Adversarial Network to generate new data points according to such noisy labels. Repeating this procedure in an iterative way, allows to refine the pseudo-labels, leading to improved accuracy on the target domain.

We address the learning from biased data problem, where we deal with spurious correlations present in the training set. Such spurious correlations may severely compromise the model’s generalization capability, therefore needs to be explicitly addressed. In Chapter 4, we present a method to remove unwanted information from the data representation in order to better generalize. In this case the attribute to be removed has to be given as a ground truth label: we seek for statistical independence between the data representation and the bias attribute, by relying on a differentiable estimator for the Mutual Information. We show how

---

the method is general and can be successfully applied to tackle computer vision classification problems and algorithmic fairness.

In Chapter 5 we consider the more challenging scenario of unsupervised debiasing, where no ground truth annotations for the bias are available. We propose a two-stage method: first, we attempt to identify, within the training data, the samples affected by bias and those which are not. Second, we treat such subsets as different tasks and learn from them in a meta-learning fashion. We propose to produce a synthetic validation set, by linearly interpolating biased and unbiased samples. This solution has proved to be effective in practice and we show how the method can learn debiased models even if the pseudo-labels are noisy.

All the proposed methods try to alleviate the risk of incorporating biases into trained models. Although considerable improvements have been made in tackling the problem, some problems are not yet addressed. We highlight here the main limitations of the proposed methods and sketch possible future research directions. A major limitation for methods presented in Chapters 3 and 5 is the lack of theoretical guarantees in the pseudo-labels assignment process: most of the claims are in fact based on empirical evidences. We partially address this limitation in Chapter 5 by providing an ablation analysis regarding the pseudo-labeling stage, thus showing how the devised method is robust towards noise in the pseudo-labels. A related point to be investigated is the splitting of data using hard (pseudo-)labels, that results in two non overlapping subsets, biased and unbiased: although the assumption of a hard division between the two subsets holds true for synthetic data (i.e. Colored MNIST, Corrupted CIFAR), it might be too rigid for real world dataset, in which boundaries between biased and unbiased are more fuzzy. In this respect a possible improvement can be brought by a soft assignment to the two subsets. An important problem that has not been tackled yet by the existing literature, is the case in which multiple biases affect the data: here, more than one spurious correlations are present in the data and the model has to handle them. Finally, the research in the debiasing field has so far adopted only small datasets as benchmarks. The reason is to try to have datasets for which it is possible to have a ground truth annotation regarding task labels and bias labels in order to assess the model's performance. Nevertheless existing methods should be tested on large scale real world datasets to prove their ability to be used in practical scenarios.

# References

- [adu] Adult census data set. <https://archive.ics.uci.edu/ml/datasets/adult>.
- [kim] Code for "learning not to learn: Training deep neural networks with biased data". <https://github.com/feidfoe/learning-not-to-learn>.
- [ger] Statlog (german credit data) data set. [https://archive.ics.uci.edu/ml/datasets/Statlog+\(German+Credit+Data\)](https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)).
- [4] Alvi, M. S., Zisserman, A., and Nellåker, C. (2018). Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. In *European Conference on Computer Vision (ECCV) Workshops*.
- [5] Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- [6] Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. (2020). Invariant risk minimization.
- [7] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia. PMLR.
- [8] Bahng, H., Chun, S., Yun, S., Choo, J., and Oh, S. J. (2020). Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, pages 528–539.
- [9] Balaji, Y., Sankaranarayanan, S., and Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- [10] Beery, S., Horn, G. V., and Perona, P. (2018). Recognition in terra incognita. *CoRR*, abs/1807.04975.
- [11] Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual information neural estimation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 531–540, Stockholm, Sweden. PMLR.

- [12] Ben-Tal, A., Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*.
- [13] Berthelot, D., Schumm, T., and Metz, L. (2017). BEGAN: boundary equilibrium generative adversarial networks. *CoRR*, abs/1703.10717.
- [14] Beutel, A., Chen, J., Zhao, Z., and Chi, E. H. (2017). Data decisions and theoretical implications when adversarially learning fair representations. *CoRR*, abs/1707.00075.
- [15] Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, EMNLP '06*, pages 120–128, Stroudsburg, PA, USA. Association for Computational Linguistics.
- [16] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2016a). Un-supervised pixel-level domain adaptation with generative adversarial networks. *CoRR*, abs/1612.05424.
- [17] Bousmalis, K., Silberman, N., Dohan, D., Erhan, D., and Krishnan, D. (2017). Un-supervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [18] Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016b). Domain separation networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 343–351. Curran Associates, Inc.
- [19] Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.
- [20] Byrd, J. and Lipton, Z. C. (2018). Weighted risk minimization & deep learning. *CoRR*, abs/1812.03372.
- [21] Cavazza, J., Morerio, P., and Murino, V. (2019). Scalable and compact 3d action recognition with approximated rbf kernel machines. *Pattern Recognition*, 93:25 – 35.
- [22] Cavazza, J., Zunino, A., San Biagio, M., and Murino, V. (2016). Kernelized covariance for action recognition. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 408–413. IEEE.
- [23] Chopra, S., Balakrishnan, S., and Gopalan, R. (2013). Dlid: Deep learning for domain adaptation by interpolating between domains.
- [24] Clark, C., Yatskar, M., and Zettlemoyer, L. (2019). Don’t take the easy way out: Ensemble based methods for avoiding known dataset biases.
- [25] Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, 20:273–297.
- [26] Creager, E., Madras, D., Jacobsen, J., Weis, M. A., Swersky, K., Pitassi, T., and Zemel, R. S. (2019). Flexibly fair representation learning by disentanglement. *CoRR*, abs/1906.02589.

- [27] Damodaran, B. B., Kellenberger, B., Flamary, R., Tuia, D., and Courty, N. (2018). Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. *CoRR*, abs/1803.10081.
- [28] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.
- [29] Denker, J. S., Gardner, W. R., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., Baird, H. S., and Guyon, I. (1989). Advances in neural information processing systems 1. chapter Neural Network Recognizer for Hand-written Zip Code Digits, pages 323–331.
- [30] Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. S., and Pontil, M. (2018). Empirical risk minimization under fairness constraints. In *Advances in Neural Information Processing Systems*, pages 2791–2801.
- [31] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.
- [32] Ganin, Y. and Lempitsky, V. S. (2015). Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1180–1189.
- [33] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1).
- [34] Geirhos, R., Jacobsen, J., Michaelis, C., Zemel, R. S., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *CoRR*, abs/2004.07780.
- [35] Ghifary, M., Kleijn, W. B., Zhang, M., and Balduzzi, D. (2015). Domain generalization for object recognition with multi-task autoencoders. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 2551–2559.
- [36] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.
- [37] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.
- [38] Goodfellow, I. J. (2017). NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160.
- [39] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773.

- [40] Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In Jain, S., Simon, H. U., and Tomita, E., editors, *Algorithmic Learning Theory*, pages 63–77, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [41] Haeusser, P., Frerix, T., Mordvintsev, A., and Cremers, D. (2017). Associative domain adaptation. In *International Conference on Computer Vision (ICCV)*.
- [42] Häusser, P., Frerix, T., Mordvintsev, A., and Cremers, D. (2017). Associative domain adaptation. *CoRR*, abs/1708.00938.
- [43] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.
- [44] Hendrycks, D. and Dietterich, T. G. (2019). Benchmarking neural network robustness to common corruptions and perturbations. *CoRR*, abs/1903.12261.
- [45] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6626–6637. Curran Associates, Inc.
- [46] Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018). CyCADA: Cycle-consistent adversarial domain adaptation. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998, Stockholmsmässan, Stockholm Sweden. PMLR.
- [47] Hospedales, T. M., Antoniou, A., Micaelli, P., and Storkey, A. J. (2020). Meta-learning in neural networks: A survey. *CoRR*, abs/2004.05439.
- [48] hyun Lee, D. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)*.
- [49] III, H. D. and Marcu, D. (2011). Domain adaptation for statistical classifiers. *CoRR*, abs/1109.6341.
- [50] Kang, G., Zheng, L., Yan, Y., and Yang, Y. (2018). Deep adversarial attention alignment for unsupervised domain adaptation: the benefit of target expectation maximization. *CoRR*, abs/1801.10068.
- [51] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.
- [52] Khosla, A., Zhou, T., Malisiewicz, T., Efros, A., and Torralba, A. (2012). Undoing the damage of dataset bias. In *European Conference on Computer Vision (ECCV)*, Florence, Italy.



- [53] Kim, B., Kim, H., Kim, K., Kim, S., and Kim, J. (2019). Learning not to learn: Training deep neural networks with biased data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9012–9020.
- [54] Kim, E., Lee, J., and Choo, J. (2021). Biaswap: Removing dataset bias with bias-tailored swapping augmentation. *CoRR*, abs/2108.10008.
- [55] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.
- [56] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.
- [57] Kleinberg, J., Mullainathan, S., and Raghavan, M. (2016). Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*.
- [58] Kodali, N., Abernethy, J. D., Hays, J., and Kira, Z. (2017). How to train your DRAGAN. *CoRR*, abs/1705.07215.
- [59] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images.
- [60] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1(4):541–551.
- [61] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.
- [62] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.
- [63] Levy, D., Carmon, Y., Duchi, J. C., and Sidford, A. (2020). Large-scale methods for distributionally robust optimization.
- [64] Li, D., Yang, Y., Song, Y., and Hospedales, T. M. (2017a). Deeper, broader and artier domain generalization. *The IEEE International Conference on Computer Vision (ICCV)*.
- [65] Li, D., Yang, Y., Song, Y., and Hospedales, T. M. (2017b). Learning to generalize: Meta-learning for domain generalization. *CoRR*, abs/1710.03463.
- [66] Li, Y. and Vasconcelos, N. (2019). Repair: Removing representation bias by dataset resampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9572–9581.
- [67] Li, Y., Yang, Y., Zhou, W., and Hospedales, T. (2019). Feature-critic networks for heterogeneous domain generalisation. In *The Thirty-sixth International Conference on Machine Learning*.
- [68] Liu, E. Z., Haghgoo, B., Chen, A. S., Raghunathan, A., Koh, P. W., Sagawa, S., Liang, P., and Finn, C. (2021). Just train twice: Improving group robustness without training group information. *CoRR*, abs/2107.09044.
- [69] Liu, M., Breuel, T. M., and Kautz, J. (2017). Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848.

- [70] Liu, M.-Y. and Tuzel, O. (2016). Coupled generative adversarial networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 469–477. Curran Associates, Inc.
- [71] Long, M., Cao, Y., Wang, J., and Jordan, M. I. (2015). Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML’15, page 97–105.
- [72] Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. (2015). The Variational Fair Autoencoder. *arXiv e-prints*, page arXiv:1511.00830.
- [73] Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2017). Are gans created equal? A large-scale study. *CoRR*, abs/1711.10337.
- [74] Madras, D., Creager, E., Pitassi, T., and Zemel, R. S. (2018). Learning adversarially fair and transferable representations. *CoRR*, abs/1802.06309.
- [75] Mancini, M., Porzi, L., Bulò, S. R., Caputo, B., and Ricci, E. (2018). Boosting domain adaptation by discovering latent domains. *CoRR*, abs/1805.01386.
- [76] Mao, X., Li, Q., Xie, H., Lau, R. Y. K., and Wang, Z. (2016). Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076.
- [77] Mary, J., Calauzenes, C., and El Karoui, N. (2019). Fairness-aware learning for continuous attributes and treatments. In *International Conference on Machine Learning*, pages 4382–4391.
- [78] Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *CoRR*, abs/1411.1784.
- [79] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.
- [80] Morerio, P., Cavazza, J., and Murino, V. (2018). Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *International Conference on Learning Representations*.
- [81] Moyer, D., Gao, S., Brekelmans, R., Galstyan, A., and Ver Steeg, G. (2018a). Invariant representations without adversarial training. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 9084–9093. Curran Associates, Inc.
- [82] Moyer, D., Gao, S., Brekelmans, R., Steeg, G. V., and Galstyan, A. (2018b). Evading the adversary in invariant representation. *CoRR*, abs/1805.09458.
- [83] Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 10–18, Atlanta, Georgia, USA. PMLR.
- [84] Murez, Z., Kolouri, S., Kriegman, D. J., Ramamoorthi, R., and Kim, K. (2017). Image to image translation for domain adaptation. *CoRR*, abs/1712.00479.

- [85] Nam, J., Cha, H., Ahn, S., Lee, J., and Shin, J. (2020). Learning from failure: Training debiased classifier from biased classifier. *Advances on Neural Information Processing systems (NeurIPS)*.
- [86] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.
- [87] Pearson, K. (1992). *On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can be Reasonably Supposed to have Arisen from Random Sampling*, pages 11–28. Springer New York, New York, NY.
- [88] Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., and Saenko, K. (2017). Visda: The visual domain adaptation challenge. *CoRR*, abs/1710.06924.
- [89] Pinheiro, P. O. (2017). Unsupervised domain adaptation with similarity learning. *CoRR*, abs/1711.08995.
- [90] Poole, B., Ozair, S., Oord, A. v. d., Alemi, A. A., and Tucker, G. (2019). On variational bounds of mutual information.
- [91] Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [92] Ragonesi, R., Volpi, R., Cavazza, J., and Murino, V. (2020). Learning unbiased representations via mutual information backpropagation.
- [93] Rolnick, D., Veit, A., Belongie, S. J., and Shavit, N. (2017). Deep learning is robust to massive label noise. *CoRR*, abs/1705.10694.
- [94] Rothe, R., Timofte, R., and Gool, L. V. (2016). Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision (IJCV)*.
- [95] Rozantsev, A., Salzmann, M., and Fua, P. (2016). Beyond sharing weights for deep domain adaptation. *CoRR*, abs/1603.06432.
- [96] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA.
- [97] Russo, P., Carlucci, F. M., Tommasi, T., and Caputo, B. (2018). From source to target and back: Symmetric bi-directional adaptive gan. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [98] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV’10*, pages 213–226, Berlin, Heidelberg. Springer-Verlag.

- [99] Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. (2019). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *CoRR*, abs/1911.08731.
- [100] Saito, K., Ushiku, Y., and Harada, T. (2017). Asymmetric tri-training for unsupervised domain adaptation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 2988–2997.
- [101] Saito, K., Ushiku, Y., Harada, T., and Saenko, K. (2018a). Adversarial dropout regularization. In *International Conference on Learning Representations*.
- [102] Saito, K., Watanabe, K., Ushiku, Y., and Harada, T. (2018b). Maximum classifier discrepancy for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [103] Sankaranarayanan, S., Balaji, Y., Castillo, C. D., and Chellappa, R. (2018). Generate to adapt: Aligning domains using generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [104] Sener, O., Song, H. O., Saxena, A., and Savarese, S. (2016). Learning transferrable representations for unsupervised domain adaptation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2110–2118. Curran Associates, Inc.
- [105] Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., and Sarawagi, S. (2018). Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*.
- [106] Shannon, C. E. (1948). A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423.
- [107] Shmelkov, K., Schmid, C., and Alahari, K. (2018). How good is my gan? In *The European Conference on Computer Vision (ECCV)*.
- [108] Shu, R., Bui, H. H., Narui, H., and Ermon, S. (2018). A dirt-t approach to unsupervised domain adaptation.
- [109] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1).
- [110] Sugiyama, M., Liu, S., du Plessis, M. C., Yamanaka, M., Yamada, M., Suzuki, T., and Kanamori, T. (2013). Direct divergence approximation between probability distributions and its applications in machine learning. *JCSE*, 7(2):99–111.
- [111] Sugiyama, M., Suzuki, T., and Kanamori, T. (2012). Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044.
- [112] Sun, B. and Saenko, K. (2016). Deep CORAL: correlation alignment for deep domain adaptation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pages 443–450.

- [113] Taigman, Y., Polyak, A., and Wolf, L. (2017). Unsupervised cross-domain image generation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.
- [114] Thekumparampil, K. K., Khetan, A., Lin, Z., and Oh, S. (2018). Robustness of conditional gans to noisy labels. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 10271–10282. Curran Associates, Inc.
- [115] Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. *CoRR*, abs/1503.02406.
- [116] Torralba, A. and Efros, A. A. (2011). Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '11*, pages 1521–1528, Washington, DC, USA. IEEE Computer Society.
- [117] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [118] Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., and Darrell, T. (2014). Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474.
- [119] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.
- [120] Volpi, R., Morerio, P., Savarese, S., and Murino, V. (2018). Adversarial feature augmentation for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [121] Volpi, R. and Murino, V. (2019). Model vulnerability to distributional shifts over image transformation sets.
- [122] Volpi\*, R., Namkoong\*, H., Sener, O., Duchi, J., Murino, V., and Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. In *Advanced in Neural Information Processing Systems (NIPS)* 32.
- [123] Wang, H., He, Z., Lipton, Z. C., and Xing, E. P. (2019). Learning robust representations by projecting superficial statistics out. *The International Conference on Learning Representations (ICLR)*.
- [124] Wang, J., Lan, C., Liu, C., Ouyang, Y., and Qin, T. (2021). Generalizing to unseen domains: A survey on domain generalization. *CoRR*, abs/2103.03097.
- [125] Wang, T., Zhao, J., Yatskar, M., Chang, K., and Ordonez, V. (2018). Adversarial removal of gender from deep image representations. *CoRR*, abs/1811.08489.
- [126] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

- [127] Xu, Z., Li, W., Niu, L., and Xu, D. (2014). Exploiting low-rank structure from latent domains for domain generalization. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 628–643.
- [128] Zhang, B. H., Lemoine, B., and Mitchell, M. (2018). Mitigating unwanted biases with adversarial learning. *CoRR*, abs/1801.07593.
- [129] Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412.
- [130] Zhang, Z. and Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *CoRR*, abs/1805.07836.
- [131] Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., and Torralba, A. (2018). Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1452–1464.
- [132] Zhou, K., Yang, Y., Hospedales, T. M., and Xiang, T. (2020). Deep domain-adversarial image generation for domain generalisation. *CoRR*, abs/2003.06054.
- [133] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*.
- [134] Zunino, A., Cavazza, J., Volpi, R., Morerio, P., Cavallo, A., Becchio, C., and Murino, V. (2019). Predicting intentions from motion: The subject-adversarial adaptation approach. *International Journal of Computer Vision*.

# Appendix A

## Supplementary material for Chapter 3

### A.1 Architectures and Hyperparameters

We provide here a detailed description of the networks used for our experiments.

Figure A.1, Figure A.2 and Figure A.3 depict the architectures used for  $C$  (classifier),  $G$  (GAN’s generator) and  $D$  (GAN’s discriminator), respectively, in the different benchmark experiments.

We report in the following the hyperparameters associated with the same experiments. We use Adam optimizer (55) in all the experiments, and set the learning rate to train pre-train  $C$  on data from the source distribution to  $3 \cdot 10^{-4}$ . For the cGAN pre-training, we set the learning rate for training both  $G$  and  $D$  to  $10^{-5}$ . When running Algorithm 1, we set  $\eta = 10^{-5}$  and  $\delta = 5 \cdot 10^{-5}$ .

Architectural choices, as well as hyperparameter tuning, were carried out with the goal of making GANs converge.

### A.2 Are deeper architectures more resistant against shift noise?

In (93) the authors provide empirical evidence that deeper models (e.g. Residual Networks (43)) are more robust against uniform label noise than shallow architectures. We investigated whether such resilience of deep models arises also with shift noise. Our experiments led us to exclude such hypothesis. We considered the split MNIST  $\rightarrow$  SVHN, where shift noise is very significant, and repeated the experiment of Table 2: we trained different ResNets (from scratch) with different depths on target samples corrupted by shift noise; we observe that

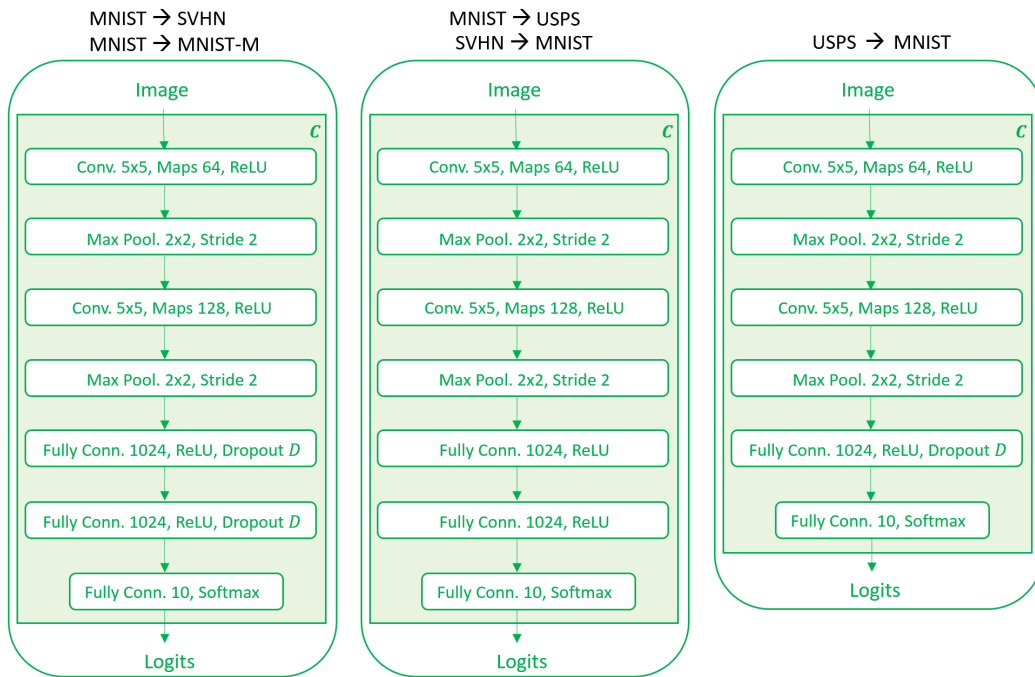


Figure A.1 Architectures for the classifier  $C$  (see Figure 4 in Chapter 3).

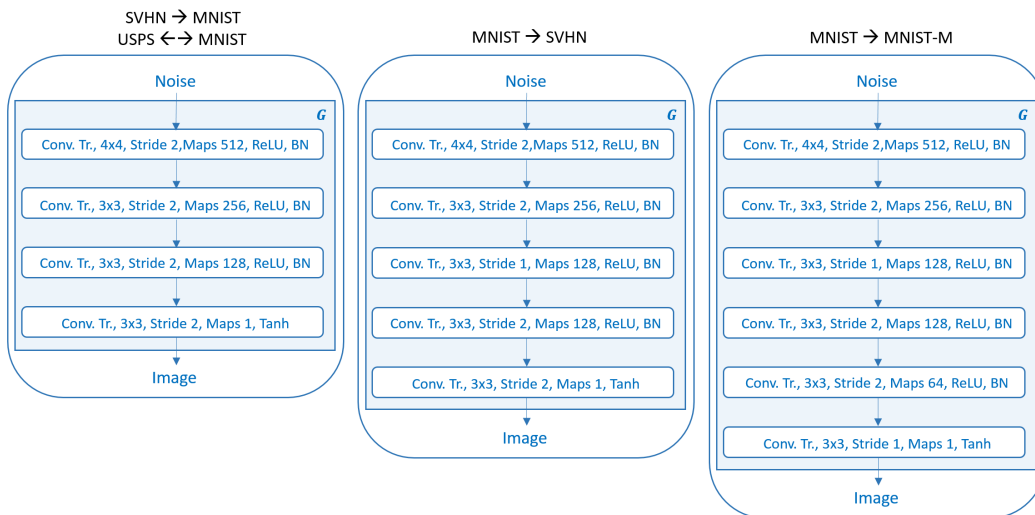


Figure A.2 Architectures for the generator  $G$  (see Figure 4 in Chapter 3).



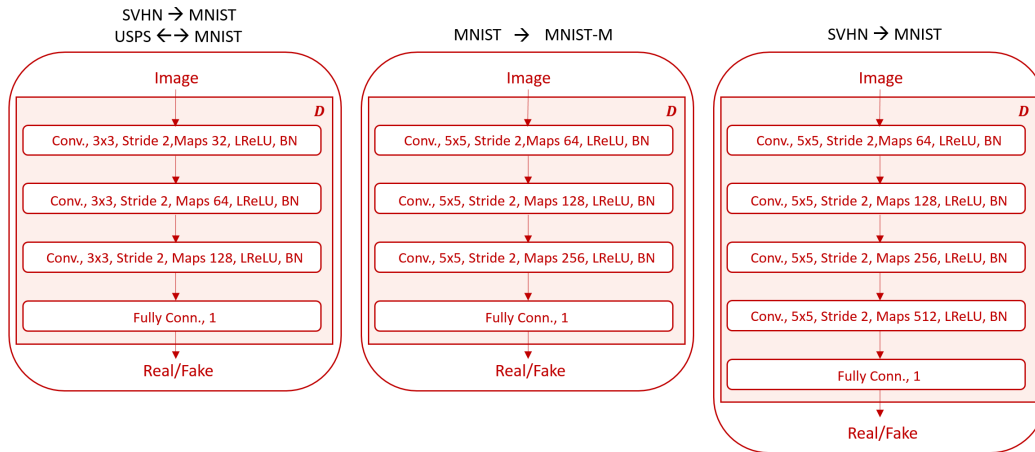


Figure A.3 Architectures for the discriminator  $D$  (see Figure 4 in Chapter 3).

despite improved capacity of the models, they overfit the noisy labelled samples and are not able to reduce  $\delta_A$ . (Figure A.4).

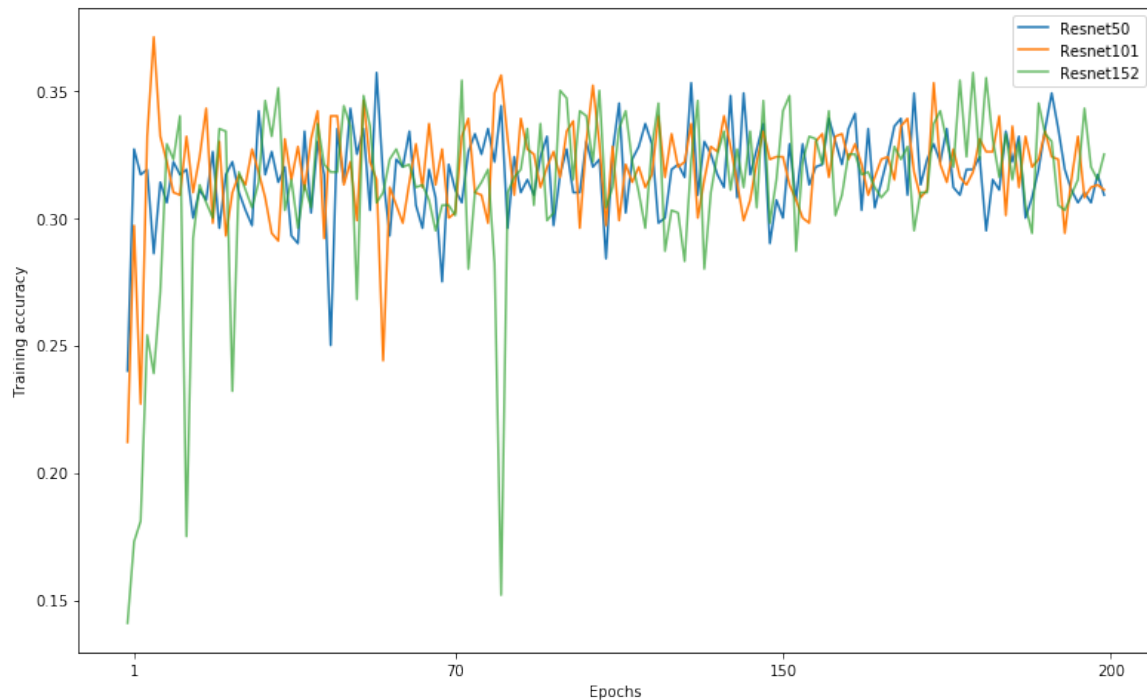


Figure A.4 Training on the shift noise: we evaluate the accuracy on *clean training set* at the end of each epoch. Despite ResNet models are deeper and more resistant to uniform noise (93), they are not robust against shift noise. Indeed, accuracy on the *noisy training set* reaches about 100% pointing out that the models overfit noise.

### A.3 Generated images

We report in Figures A.5, A.6, A.7, A.8 samples generated by  $G$  after the training procedure defined by Algorithm 1, for the splits SVHN  $\rightarrow$  MNIST, MNIST  $\rightarrow$  SVHN, MNIST  $\rightarrow$  MNIST-M, MNIST  $\rightarrow$  USPS and USPS  $\rightarrow$  MNIST, respectively. For each experiment, we randomly generated 20 samples associated with the different classes and reported them in the Figures, where each row is related to a different class.

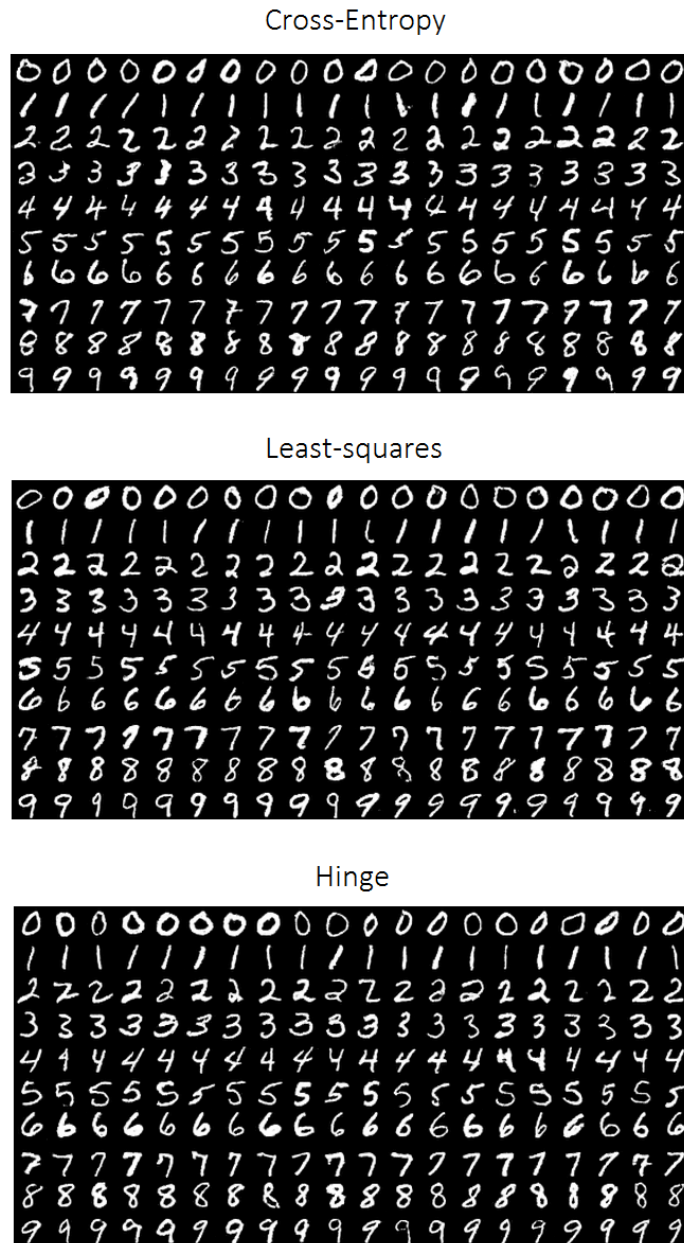


Figure A.5 MNIST samples generated by  $G$ , trained with Algorithm 1 (SVHN  $\rightarrow$  MNIST split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

Cross-Entropy



Least-squares



Hinge

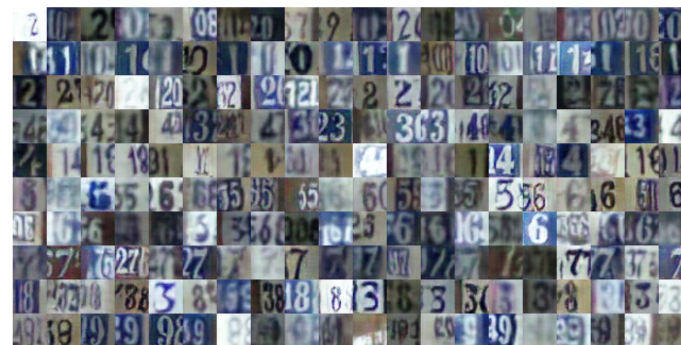


Figure A.6 SVHN samples generated by  $G$ , trained with Algorithm 1 (MNIST  $\rightarrow$  SVHN split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

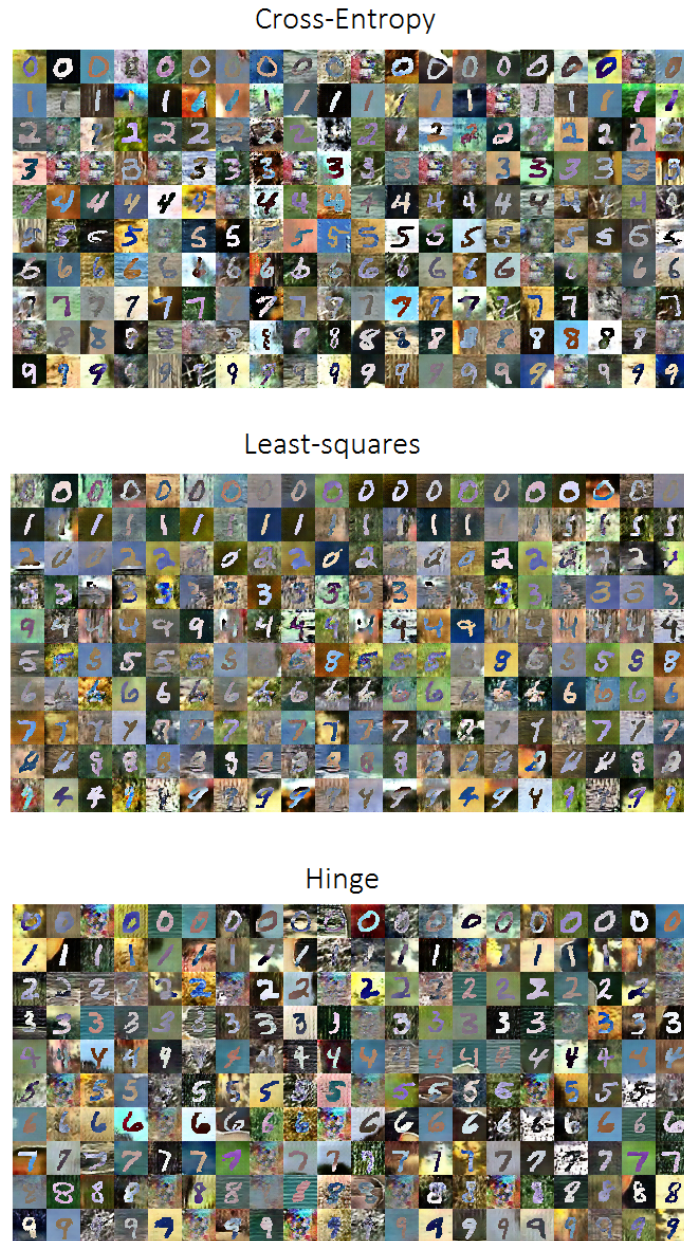


Figure A.7 MNIST-M samples generated by  $G$ , trained with Algorithm 1 (MNIST  $\rightarrow$  MNIST-M split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).



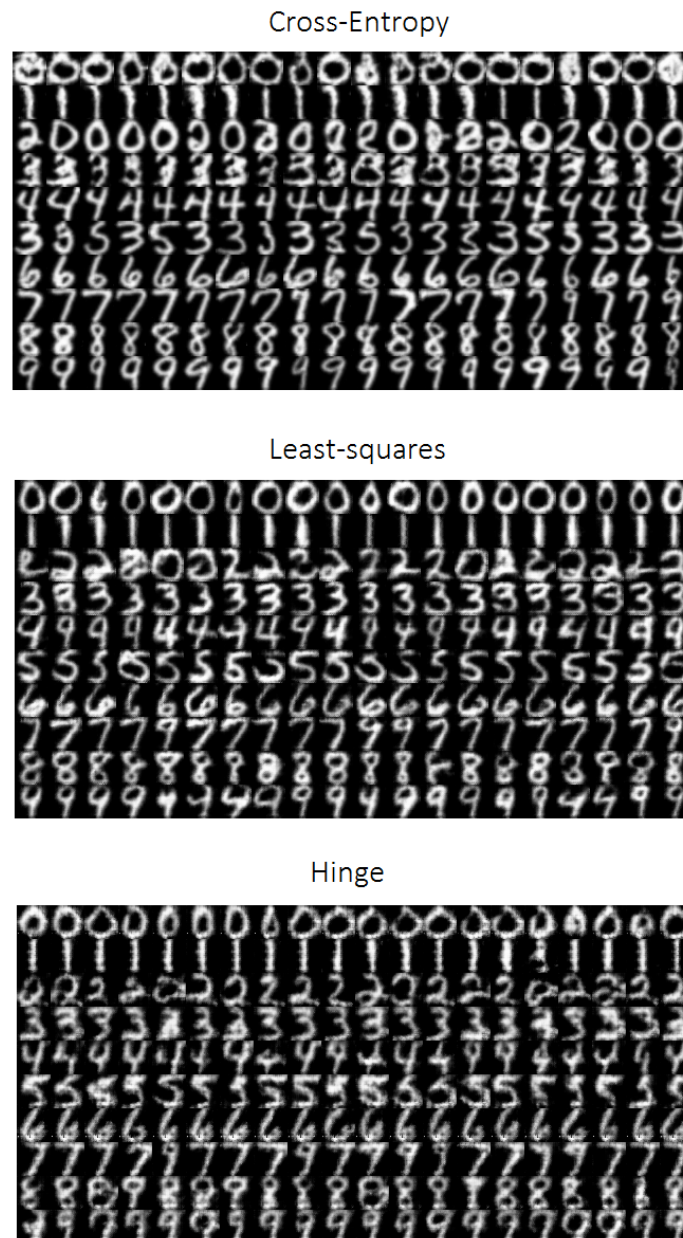


Figure A.8 USPS samples generated by  $G$ , trained with Algorithm 1 (MNIST  $\rightarrow$  USPS split). Each row is related to a different label code (from *top* to *bottom*, 0 to 9).

# Appendix B

## Supplementary material for Chapter 4

### B.1 Implementation Details

In the following paragraphs, we provide the implementation details. We carried out all of our experiments using TensorFlow <sup>1</sup>. Concerning the architectures used, please refer to Figure B.2.

To ease the discussion, we can divide the optimization problem presented in our work into the following two optimization problems:

$$\min_{\theta, \psi} \mathcal{L}_{task} + \lambda \mathcal{L}_{ne} \quad (\text{B.1})$$

$$\max_{\phi} \mathcal{L}_{ne} \quad (\text{B.2})$$

where the learning rates associated to Eq.(B.1) and Eq.(B.2) are  $\alpha$  and  $\eta$ , respectively. We use the same notation of Algorithm 2.

**Digit experiment.** We train our models for 150 epochs, using mini-batches of size 1024. The learning rates  $\alpha$  and  $\eta$  are both set to  $10^{-4}$ . We use Adam (55) as optimizer for Eq.(B.1) and (B.2). For each gradient update to optimize Eq.(B.1) with respect to  $\theta, \psi$ , we update MINE parameters 80 times ( $K = 80$ ). That is, we perform 80 update steps to optimize Eq.(B.2), as to better train MINE (see Section B.2 for a detailed discussion around this choice).

---

<sup>1</sup><https://www.tensorflow.org/>

**IMDB experiment.** For both training splits (EB1 and EB2) we restrict the training set to 2000 samples (for each run we sampled different random images). This choice is motivated by the fact that using the whole training sets we observed higher baselines results than the ones published in previous art (53). We trained each model for 6 epochs with mini-batch size set to 24. The learning rate  $\alpha$  is set to  $10^{-5}$ ; the learning rate  $\eta$  is set to  $10^{-1}$ . We use Adam (55) as optimizer for Eq.(B.1) and vanilla gradient descent for Eq.(B.2). We found a number  $K = 20$  of MINE iterations to be sufficient in order to estimate the mutual information throughout training.

**German experiment.** We adopted the same settings as previous art that uses this benchmark (81). The 1,000 data samples available are split in 70% training and 30% test (randomly picked in each run). The model is trained for 500 epochs with mini-batch size set to 64. The learning rate  $\alpha$  is set to  $10^{-5}$ ; the learning rate  $\eta$  is set to  $10^{-1}$ . We use Adam (55) as optimizer for Eq.(B.1), and vanilla gradient descent for Eq.(B.2). We set to a number of MINE iterations  $K = 30$ .

**Adult experiment.** We adopted the same settings as Madras et al.(74). The model is trained for 1,000 epochs with mini-batch size set to 64. The learning rates  $\alpha$  and  $\eta$  are set to  $10^{-3}$ . We use Adam (55) as optimizer for both Eq.(B.1) and Eq.(B.2). We set to a number of MINE iterations  $K = 30$ .

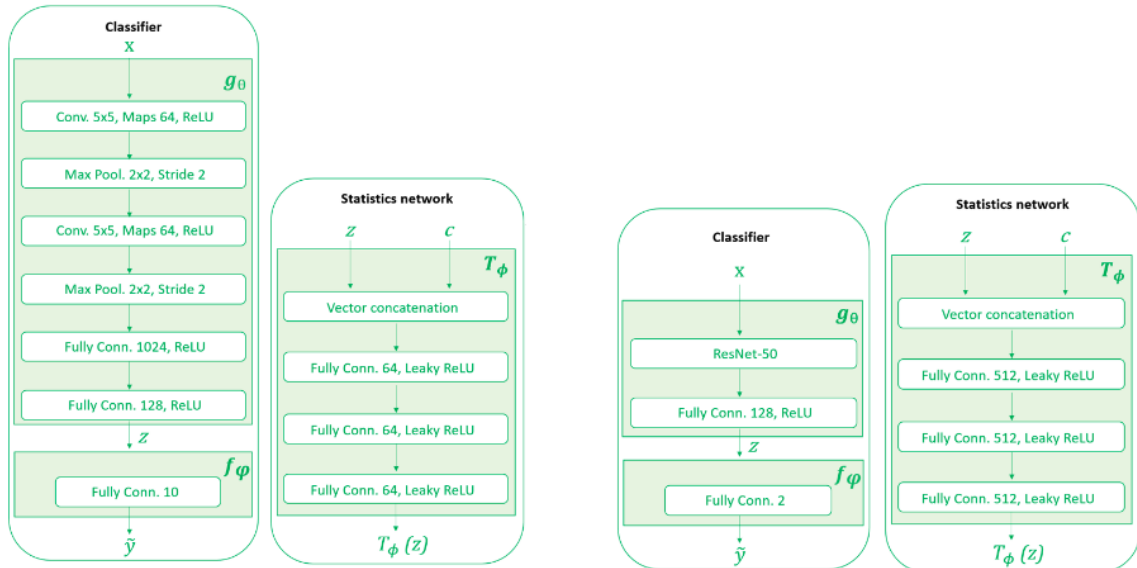


Figure B.1 Description of the architectures (classifiers and statistics networks) for the experiments on Digits (left), IMDB (right).



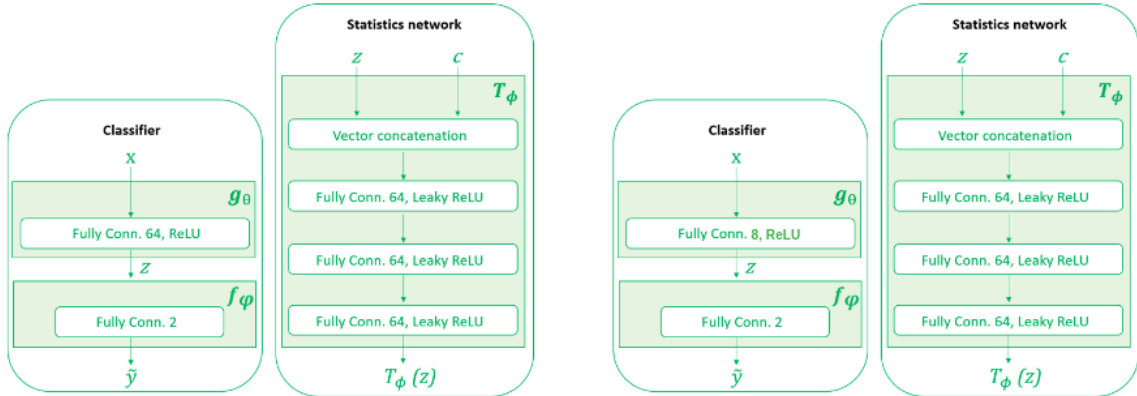


Figure B.2 Description of the architectures (classifiers and statistics networks) for the experiments on German (*left*), Adult (*right*).

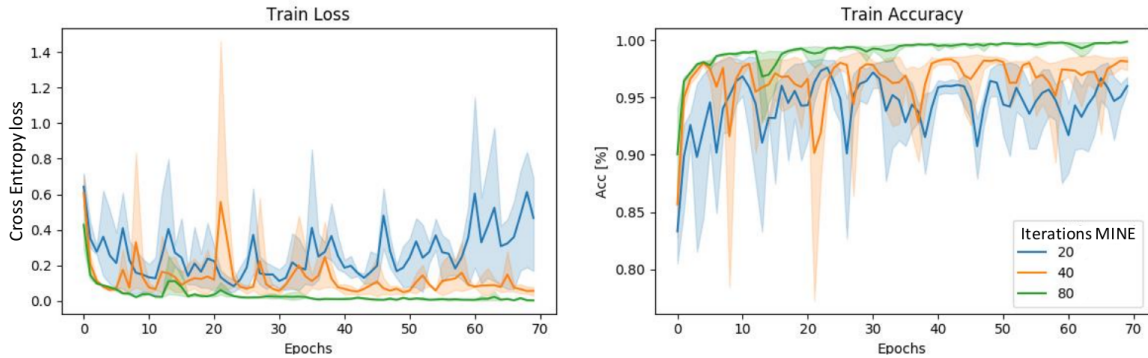


Figure B.3 Training (cross-entropy) loss (*left*) and training accuracy (*right*) with  $\lambda = 1.0$  for different number of iterations of MINE ( $K$ ) on the digit recognition task (setting  $\sigma = 0.02$ ). An increased number of iterations ( $K = 20, 40, 80$  in *blue, orange* and *green*, respectively) has the effect of stabilizing the training procedure, it allows the model minimizing the loss function and fitting the training data. The charts report the average of 3 runs.

## B.2 Discussion on the Hyper-Parameters

In this section, we discuss the hyper-parameters that we adopted throughout the experiments reported in this work.

**Choice of the number of iterations to update MINE.** We found that increasing the number of iterations to estimate  $I(Z, C)$  stabilizes the overall training procedure, as shown in Figure B.3. As our intuition behind this fact, we posit that the better the estimation of the mutual information through MINE is, the more precise and effective the gradients  $\nabla_{\theta} \mathcal{L}_{ne}$  are. The only drawback we observed is the increased computational cost, since the time increases linearly with the number of iterations employed to estimate the mutual information.

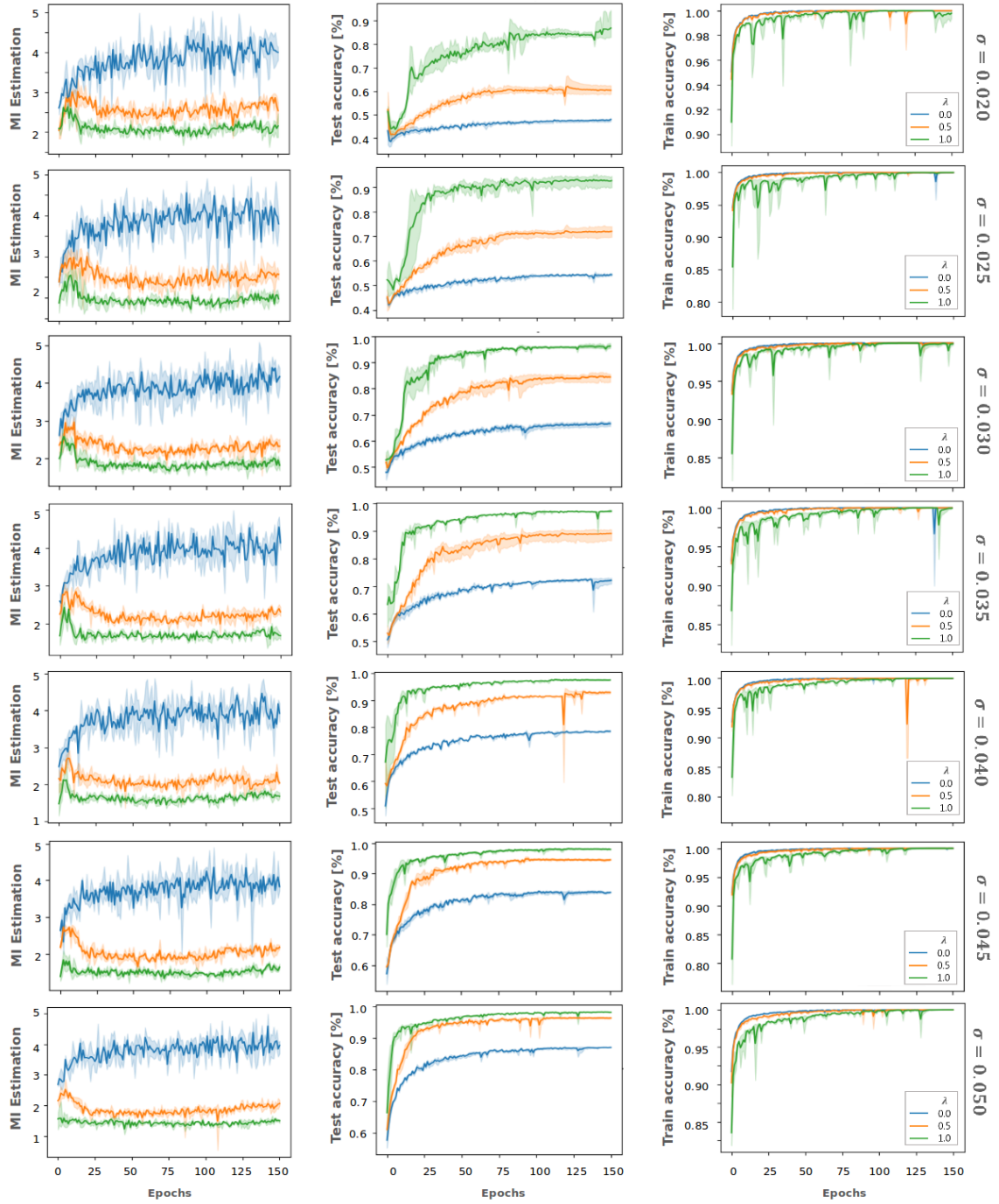


Figure B.4 Values for mutual information (left column), test accuracy (middle column) and train accuracy (right column). We accounted for the different color, modelled by different  $\sigma$  (check Section 4.5 of Chapter 4), and here represented by different rows. It is visible how a decrease in the (estimated) mutual information correlates with an improved performance.

**Choice of the hyper-parameter  $\lambda$ .** The hyper-parameter  $\lambda$  regulates the trade-off between minimizing the task loss and reducing the mutual information between the biased attribute and the learned representation in Eq.(B.1). In Section 4.5 of Chapter 4, we describe how to properly tune it. We report in Figure B.4 the complete version of the analysis reported in the manuscript for the Digit experiment. We report the evolution of mutual information, test accuracy and training accuracy for different values of the hyper-parameter  $\lambda$ .

# Appendix C

## Supplementary material for Chapter 5

This appendix reports some further details about the proposed method and the results obtained, and presents in particular an extended ablation analysis. Specifically, we present:

- the method’s performances by varying the parameter  $\zeta$  controlling the weight of the regularization term;
- how the quality of the subdivision of the data in biased/unbiased samples performed by the initial pseudo-labeling stage affects the final accuracy;
- implementation details related to all our experiments.

To have the full control of the experimental conditions, most of these trials is carried out on the synthetic biased dataset Corrupted CIFAR-10<sup>1</sup>, if not differently specified.

### C.1 Ablation study on $\zeta$

We set different values of the parameter  $\zeta$ , i.e.,  $[0, 1, 10, 100, 1000]$  on the synthetic biased dataset Corrupted CIFAR-10<sup>1</sup> with bias ratio= 95%, and we show in Figure C.1 the accuracy on the unbiased and on the full set of samples. We can note that there is a large range of  $\zeta$  values,  $1 < \zeta < 100$ , in which the accuracy is reaching high values in both cases (please, note the logarithmic scale in the  $x$  axis). This empirically shows that  $\zeta$  is not so a sensitive parameter with respect to the proposed strategy, and for this reason, we fix  $\zeta = 10$  in all our experiments.

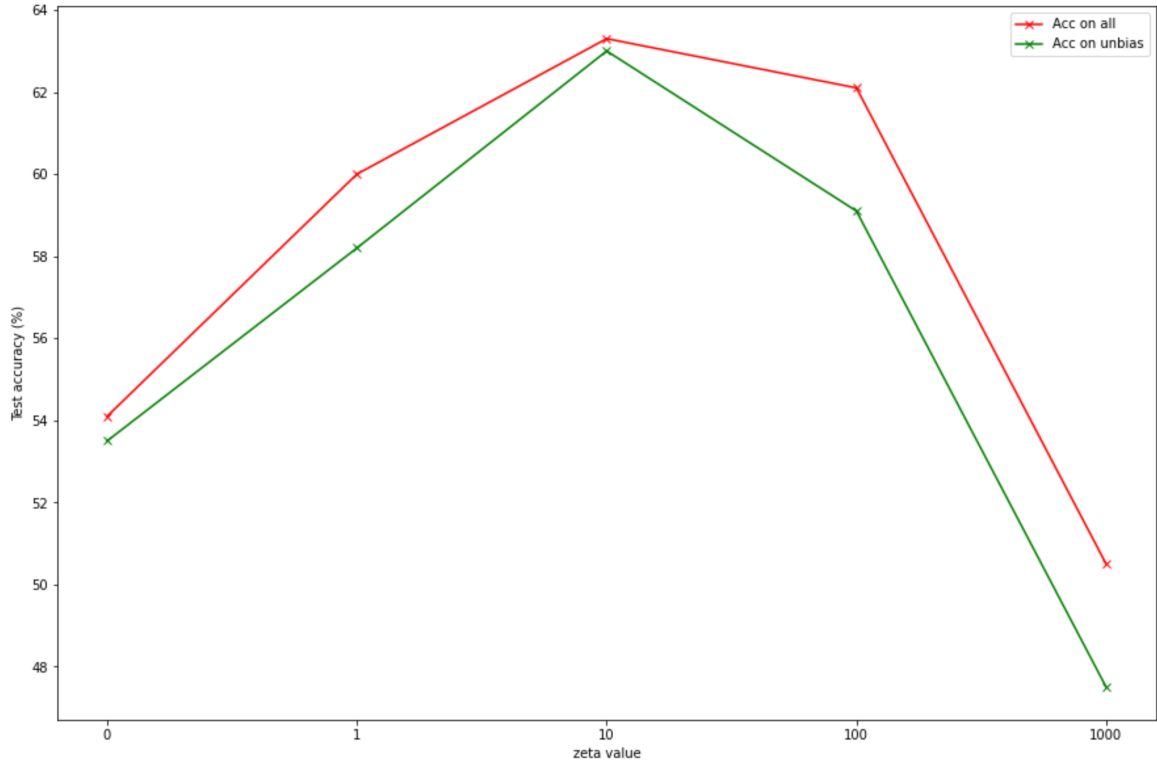


Figure C.1 Test accuracy for unbiased samples (green) and full set of samples (red) for different values of  $\zeta$ . The  $x$  axis is in logarithmic scale. For  $\zeta = 0.0$  we have the weighted ERM of Eq. 5.

Bias Identification			Oracle		Random split	
Bias ratio	F1 score	Test Acc.(%)	F1 score	Test Acc.(%)	F1 score	Test Acc.(%)
95%	0.65	63.3	1.0	66.3	0.37	50.3
98%	0.62	56.2	1.0	59.4	0.34	40.4
99%	0.58	50.5	1.0	54.7	0.33	29.7
99.5%	0.54	43.3	1.0	49.0	0.32	21.5

Table C.1 **Ablation study on bias identification.** F1 score of  $\hat{\mathcal{D}}_{bias}$ ,  $\hat{\mathcal{D}}_{unbias}$  compared to ground truth annotations  $\mathcal{D}_{bias}$ ,  $\mathcal{D}_{unbias}$  obtained with our Bias Identification strategy (Eq.2), using an oracle and with subsets generated randomly. We report the final test accuracy on the whole test set for the three different cases.

## C.2 Ablation study on bias identification

We investigated the performance of the initial pseudo-labeling stage and evaluate it using the F1-score metric. We evaluate how the estimated subsets  $\hat{\mathcal{D}}_{bias}$  and  $\hat{\mathcal{D}}_{unbias}$  are close to the ground-truth splits  $\mathcal{D}_{bias}$  and  $\mathcal{D}_{unbias}$ , respectively, and consequently, how this subdivision

influences the final test accuracy (which include bias + unbiased samples). We conducted our analysis on CIFAR-10<sup>1</sup>, setting  $\gamma = 0.85$ , as done for all the experiments presented in Chapter 5. The results of this analysis are reported in Table C.1. We can notice that although the splits obtained with our initial pseudo-labeling method are not outstanding, reaching  $F1 = 0.65$  in the easiest case (bias ratio = 95%), this has a minor effect on the final accuracies. Specifically, with respect to the “oracle” case ( $F1 = 1$ ), we experience a drop ranging from 3% to 6% (the latter in the hardest case, bias ratio = 99.5%). This shows how the method is robust to the quality of the estimated pseudo-labeled subsets. Instead, in the case of random splits, when pseudo-labels are generated randomly from a discrete uniform distribution,  $F1$  score (averaged over 5 runs) is significantly lower, ranging from 0.37 to 0.32 as long as bias ratio increases, and this significantly affects the final accuracies, which drop from 16% to about 27% with respect to the oracle. Finally, we also want to highlight that we still outperform our ERM baseline and, in some cases, state-of-the-art methods specifically designed for debiasing (See Table 5.1).

### C.3 Implementation details

We report here some additional implementation details regarding the conditions in which we performed our experiments. We followed prior works (68; 85; 99) in order to get results comparable with those obtained by state-of-the-art methods.

**Colored MNIST.** We used an MLP with 3 hidden layers with 100 neurons each as  $f_\theta$ . We set learning rate  $\eta = 0.001$ , batch size = 256, hyperparameters  $\gamma = 0.85$ ,  $\zeta = 10$  and trained for  $K = 100$  epochs. We used Adam (55) as optimizer.

**Corrupted CIFAR-10<sup>1,2</sup>.** We used the Pytorch implementation of ResNet-18 (43) with pre-training on ImageNet (28). We removed the last layer and added a 2-layer MLP with 256 neurons in the hidden layer on top of the backbone. We set learning rate  $\eta = 0.001$ , batch size = 256, hyperparameters  $\gamma = 0.85$ ,  $\zeta = 10$  and trained for  $K = 100$  epochs. We used Adam as optimizer, and random crops as data augmentation as in (85). We compute  $\theta^*$  only for the MLP-head parameters; in other words, the backbone is not involved in the meta-learning process, but is trained only with the Weighted ERM contribute of Eq. 5.5.

**Waterbirds.** We used pre-trained ResNet-50 (following (68)). We removed the last layer and added a 2-layer MLP with 256 neurons in the hidden layer on top of the backbone. We set learning rate  $\eta = 0.001$ , batch size = 128, hyper-parameters  $\gamma = 0.85$ ,  $\zeta = 10$ , and we trained for  $K = 100$  epochs. We used Adam as optimizer (with weight decay = 0.0001) and no data augmentation. We compute  $\theta^*$  only for the MLP-head parameters.

**BAR.** We used pre-trained ResNet-18 from which we removed the last layer and added a 2-layers MLP with 256 neurons in the hidden layer on top of the backbone. We set learning rate  $\eta = 0.001$ , batch size= 128, hyper-parameters  $\gamma = 0.85$ ,  $\zeta = 10$ , and we trained for  $K = 100$  epochs. We used Adam as optimizer (with weight decay= 0.0001) and random resized crops as data augmentation as in (85). We compute  $\theta^*$  only for the MLP-head parameters as we do for Corrupted CIFAR-10.