Working With What You've Got: Leveraging Mislabeled Datasets And Improving Imperfect Pretrained Models

PhD Dissertation

Walter Gerych

Worcester Polytechnic Institute

Data Science Program

September 2023

Abstract

Resources such as OpenML and HuggingFace have made large datasets and powerful pretrained models more accessible than ever for deep learning practitioners and researchers. However, the large-scale datasets typically used to train deep learning systems are often plagued by *noisy* labels, where the label associated with some datapoint may be incorrect. Likewise, many pretrained models exhibit biased outputs and lack the full range of functionalities desired by the end users. In this dissertation, I study four topics related to data and model quality issues.

Extending the Capabilities of Learning in Positive-Unlabeled Noisy Label Settings: In the first two tasks, I focus on the understudied *Positive Unlabeled* (PU) setting for noisy labels. In a PU dataset *some* of the positive instances are labeled, while the remaining positives and the negative instances are not distinguished form each other. This can be thought of as one-sided label noise, where some positive instances have their label flipped to the negative class. This label quality issue is common across datasets for many tasks and domains, from computer vision to biomedical data. For instance, computer vision datasets for object detection often provide annotations in the form of a list of objects that are in a given image. Despite its importance, there are limitations to the current work in the Positive Unlabeled setting. Specifically, existing methods typically assume a binary classification setting and that there is no sample selection bias in determining which instances are labeled. In my dissertation, I set out to addresses these shortcomings.

`Task 1: Extending Positive Unlabeled Learning To Multi-Label Data.` In this task, I extend methods for learning from Positive Unlabeled data, which typically are limited to only *binary classification*, to also work with *multi-label* data and *multi-label* classifiers. To do this, I formalize a novel unbiased risk to train models that are unbiased on the distribution of clean data given only noisy PU data. Experimental results on common multi-label datasets show that our method is significantly more accurate in predicting the correct label-set than alternative approaches, especially as the level of PU label noise is increased.

`Task 2: Modeling Biased PU Sample Selection.` Here, I study PU learning under the more realistic scenario of a *biased* sampling strategy that leads to unrepresentative labels. In contrast, previous PU works almost exclusively assumed unbiased labeling mechanisms. In this task, I analyze when it is theoretically possible to perform identifiable PU learning. I then propose two strategies to do so under a set of reasonable assumptions. The results indicate that our approaches nearly always approximate the true labeling likelihood and significantly outperform existing methods on a suite of common benchmark datasets.

Extending the Applicability of Pretrained Generative Models. The last two tasks focus on extending the usability of pretrained generative models. Specifically, I study the task of *debiasing* generative models as well as adding the ability of conditional generation to unconditional pretrained models. Due to the prohibitive costs associated with finetuning or training generative models along with access to *clean* unbiased data often being limited, I focus on finetuning-free *unsupervised* solution strategies.

`Task 3: Debiasing Pretrained Generative Models Without Retraining.` In this task, I focus on *debiasing* pretrained generative models. To do this, I formalize the concept of a *semantically uniform distribution*: a data distribution which places an equal amount of mass on each possible value of a *semantic attribute*, such as race or gender. I propose a principled approach for re-sampling from the generator's latent space in order to yield a semantically uniform output distribution, which allows us to debias the generator without retraining the model. Experimental analysis on multiple types of generative models (GANs, VAEs, and DDIMs) shows that our approach reduces the bias of the generative model's output significantly more than existing approaches on a variety of common image datasets.

`Task 4: Converting Unconditional Pretrained Generators into Conditional Models Without Retraining Or Supervision.` Lastly, I propose an approach for converting *unconditional* generative model, which generate data distributions but do not allow the user to choose which class to sample from, into *conditional* models that can be made to sample from specific classes. I achieve this by identifying and removing regions of the latent space that correspond to low-density regions in the output space, and then clustering the remaining regions. Each cluster in the new latent space can be shown to correspond to a semantically meaningful sub-manifold in the output space; e.g., each sub-manifold corresponds to a particular class. Using Gaussian Mixture models fit on each of these clusters, we can then selectively sample from a given sub-manifold in the output space in order to generate a sample from a desired mode. Experimental results indicate that the clusters found using our approach are significantly correlated with unique classes in the data space.

Contents

Acknowledgments

I was supported by many people throughout my time at WPI. I would like to deeply thank everyone who provided guidance, friendship, and help throughout this process.

First, I am incredibly grateful to my amazing advisors, Elke Rundensteiner and Emmanuel Agu. Prof. Rundensteiner's level of dedication to her students is humbling and an inspiration, and I am proud and fortunate that I have had the opportunity to learn from her. Prof. Agu has likewise provided support to me throughout these years, teaching me many valuable lessons and guiding me to become a much stronger researcher than I otherwise would be. A special thank you to Oren Mangoubi and Adam Kalai for agreeing to serve on my dissertation committee. I know their time is very valuable, and I am grateful for their willingness to spend some to help me on way to a PhD.

I am thankful to all of my fellow students in the DAISY research lab and the WASH research group for their feedback and support over the years. Special thanks to my friends and collaborators Thomas Hartvigsen, Luke Buiquicchio, Hamid Mansoor, Jidapa Thadajarassiri, Nick Josselyn, and Peter VanNostrand. Without your support my research would not have been possible, and without your friendships this journey would have been far less enjoyable.

I am also eternally grateful for my parents. Thank you for instilling a curiosity and love of learning within me, and for nurturing that throughout my life. Lastly, I will be forever thankful for the love and support my partner has given to me over all these years.

I also thank WPI for TA support, and the DARPA WASH grant for RA support to enable me to further my education.

This material is based on research sponsored by DARPA under agreement number FA8750-18-2-0077. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

1 Introduction

1.1 Motivation

There are now more easily accessible resources for building machine and deep learning empowered systems than ever before, fueled by the proliferation of open source and easily accessible datasets and models [1, 2, 3, 4, 5, 6]. Researchers and practitioners can find datasets fitting almost any task they may be interested in, from object detection [7] to protein sequencing [8], that they can freely use to develop their own machine and deep learning models [1, 2]. Likewise, resources like HuggingFace [3] and Open ML [5] provide both pretrained deep learning models as well as model architectures that can be retrained or fine-tuned on new data. The ability to leverage these pretrained models is highly valuable to many, as modern deep learning systems typically require computational resources far beyond what is accessible to individuals in all but the largest organizations [9]. However, there are both ethical and practical limitations to using these available resources in their current form [10, 11].

The datasets used to train models usually consist of pairs of data instances and labels; e.g., a dataset may contain images as well as labels in the form of a list of the objects that appear in the image. Machine and deep learning models are then trained to *map* between data instances and labels. For example, a model can learn to classify the objects in an image or to synthesize an image that matches a given label set [12]. Unfortunately, it is known that many datasets - including common benchmark datasets [10, 13, 14, 15] - often contain flaws such as *missing* or *inaccurate labels* and *biased* data [11, 16, 10, 17]. For example, the labels associated with a given image in a dataset may be an incomplete description of the total objects in the image. Moreover, these labeling issues are known to be pervasive in widely used, popular datasets are known to contain significant labeling issues.

Also, certain demographics may be more heavily effected by labeling issues than others. The performance of models trained using flawed labels can be significantly negatively impacted [17].

Likewise, *pretrained* machine learning and deep learning models frequently produce biased outputs [10, 18, 19, 20, 21]. As modern deep learning systems typically require vast amounts of data to train [22], hand crafting large scale datasets at the required scale becomes impractical. Thus the common approach for deep learning developers is to utilize huge and mostly uncurated datasets rather than carefully crafted training sets [23]. Bias and other data quality issues lead to biased and flawed models [23], such as healthcare models under-performing on minority groups [24] or image generators over-representing white males [21]. Aside from the issue of biased outputs, pretrained models may simply not have every desired functionality. For instance, many publicly available generative models (models that create synthetic data) are *unconditional* - e.g., they do not offer the user control over what *class* the generated data belongs to - in spite of us often having the need to generate instances belonging to some specific class [25]. For example, a pretrained generative model may be able to synthesize pictures of people's faces, but the user may not be able to specify the race of the individual.

In this dissertation, I study four problems relating to the issues of dataset quality and pretrained model limitations. Specifically, in the context of the former, I propose two approaches for learning from a dataset with one-sided label noise (”Positive Unlabeled Data” [26, 27, 28]). While in the context of the latter, I propose one approach for debiasing pretrained generative models and another approach for converting unconditional generative models into conditional models.

The following two subsections provide a high level description of the two broad directions of this dissertation; namely, (1) learning from flawed datasets and (2) enhancing imperfect pretrained models, respectively. They begin with a brief description of the existing literature which focuses on the broader topic, before narrowing the focus to the two subfields that I focus on in particular: learning from Positive Unlabeled (PU) data, and enhancing existing pretrained generative models.

1.2 Correcting Noisy Datasets

1.2.1 Background

In general, datasets can be *flawed* in many different ways. Datasets of multivariate or time series data can have *missing* or *incomplete* observations in the sense of certain features or timesteps missing from the available data [29, 30]. Datasets can be *biased* when they inadequately represent certain modes or data corresponding to different demographics (e.g., racial minorities are unfaithfully or under-represented) [31, 32]. Additionally, a dataset can also be biased in a more abstract sense when the statistical distribution of the training set does not match the distribution of data that the model is to be applied to [33, 34]. Then we say that there is a *distribution shift* between the training data and the target data [35, 36, 37]. A dataset may also contain *noisy observations* [38, 39, 40]; for instance, certain data points may have incorrect or misleading feature values.

In datasets with *label noise*, class labels associated with certain instances are incorrect [41, 42, 43]; e.g., a picture of a "dog" might be labeled as a picture of a "cat". ***Positive Unlabeled*** **(PU) datasets are a special case of noisy datasets [26, 28], which I propose to to investigate in this dissertation.** In PU data, only a subset of the *positive instances* of a class are labeled. A *positive instance* of a class is a datapoint that should be associated with a class; e.g. for images if the class is "dog" than a positive instance of the class would be a picture of a dog. If only a subset of the positive instances are labeled, then in our example some images of dogs in the dataset would not be given a label of "dog". Further, in PU no instances of the *negative class* are labeled. For example, there is no label explicitly indicating that a dog is *not* in the image.

In cleanly labeled data, a lack of an explicit label for a class indicates that the instance should not be associated with that task. In other words, no label for a given class indicates that the class is *negative* for that instance. However, in PU data, an absence of a label for a class could mean that the class is negative, or it could mean that the positive label is simply missing. PU data can thus be thought of as data with one-sided label noise: instances of the positive class may have their label "flipped" from *positive* to *negative* (*missing*).

1.2.2 State-of-Art

While the issue of noisy labels is far from being a solved problem, there is a broad range of literature on machine and deep learning methods to correct for or work around the issues caused by label noise [42, 44, 43, 45, 46, 41]. Methods for training models given noisy labels often aim to make models robust to label noise [41]. Alternatively, many methods propose to learn the *flip probabilities* [42] - the likelihood of an instance being labeled class i when it should belong to class j. Knowledge of these flip probabilities can be utilized to construct losses that result in a *risk* that matches the risk on a "clean" perfectly labeled data in expectation [43]. In other words, a model trained to minimize a noisy-label-corrected loss will also minimize the average loss or *error* that the classifier would have on clean data. Thus, a model that performs well on clean data can be trained even with noisy training data.

Positive Unlabeled data remains relatively less studied compared to the other general types of data issues, likely due to its more niche set of assumptions (e.g., one-sided label noise). However, correcting PU data is important for a wide range of domains: for instance, the PU setting applies to datasets used for knowledge base completion [47],

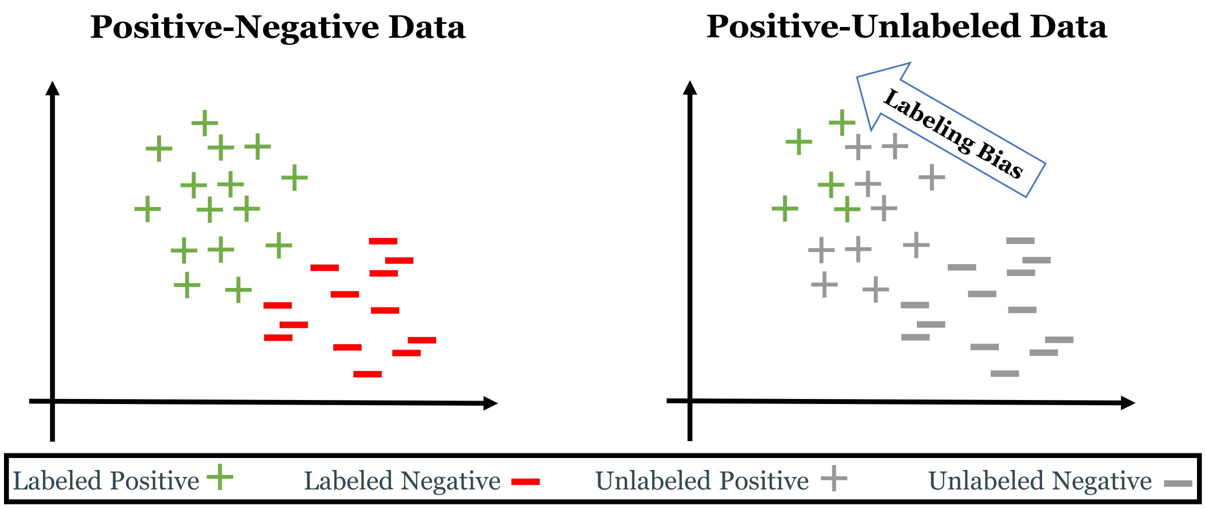


Figure 1: An example of PU data with a biased selection process.

bioinformatics [48], object detection [15], and on datasets consisting of self-reported activity logs [49, 50]. Existing PU methods typically assume that the labels are binary and the selection process for determining which (positive) instances are labeled and which go unlabeled is *unbiased*: the distribution of labeled positives and unlabeled positives are the same [26]. This is known as the *Selected Completely At Random (SCAR)* setting [26]. Under this assumption, probabilistic classifiers trained to predict the PU labels can be converted into classifiers that are accurate on the *true* class labels by only reweighting their outputs according to the proportion of unlabeled positives instances [27]. For instance, in the SCAR setting $p(\mathbf{y}=1|\mathbf{x}) = p(\mathbf{y}=1|\ell=1)p(\ell|\mathbf{x})$, where ℓ is a binary indicator variable corresponding to whether the instance is labeled. Thus, a probabilistic classifier that learns to predict $p(\ell|\mathbf{x})$ - which requires only PU data - can be used to obtain a likelihood of the true class $p(\mathbf{y}=1|\mathbf{x})$ when reweighted by $p(\mathbf{y}=1|\ell=1)$, which can also be approximated using PU data [51, 52]. Likewise, many standard losses can be reweighted according to $p(\mathbf{y}=1|\ell=1)$ in order to train unbiased classifiers on PU data [53].

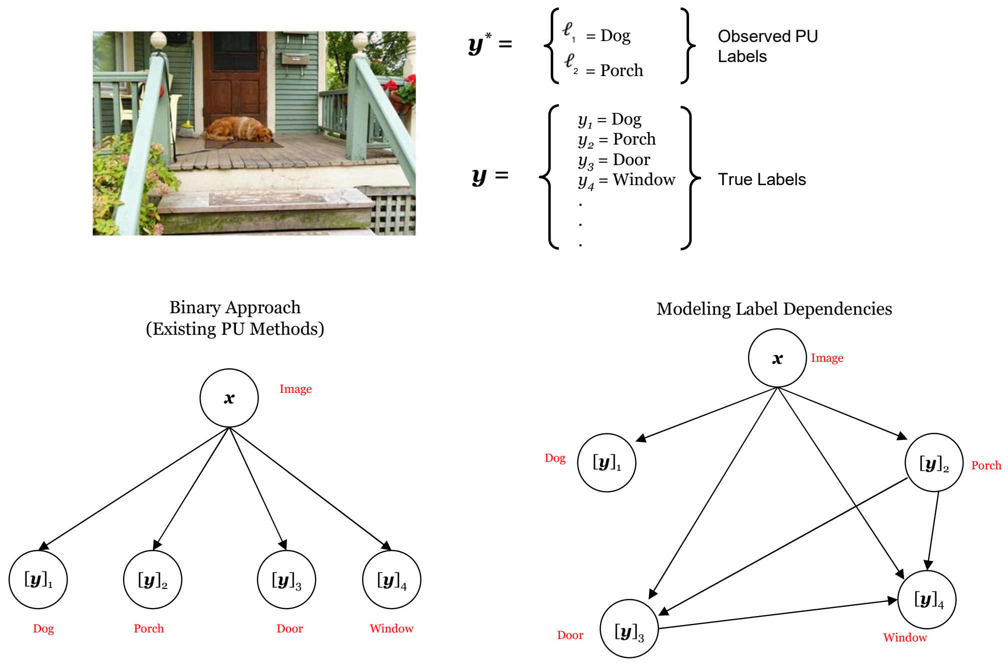


Figure 2: Existing PU methods treat each class as a separate binary classification problem. However, multi-label approaches that learn class inter-dependencies can increase performance.

1.2.3 Challenges.

Issues arise when the labeling process is *biased*; i.e., when $p(\mathbf{y} = 1|\ell = 1, \mathbf{x}) \neq p(\mathbf{y} = 1|\ell = 1)$ [26]. See Figure 1 for an example of this. In this case, referred to as the *Selected At Random (SAR)* setting [54], the property that $p(\mathbf{y} = 1|\mathbf{x}) = p(\mathbf{y} = 1|\ell = 1)p(\ell|\mathbf{x})$ no longer holds. In order to perform unbiased risk estimation and learn accurate class likelihoods under SAR, the labeling probability $e(x) = p(\mathbf{y} = 1|\ell = 1, \mathbf{x})$ - referred to as the *propensity score* [54] - must be known [26]. While prior work has proposed estimating $e(x)$ using an expectation-maximization algorithm [54], there has been little work on investigating when the propensity score is *identifiable*; i.e., when the score can be uniquely and exactly recovered given only biased PU data.

These two fundamental assumptions typically made by PU methods - unbiased sample selection and binary labels - limit their applicability to many real-world settings. For

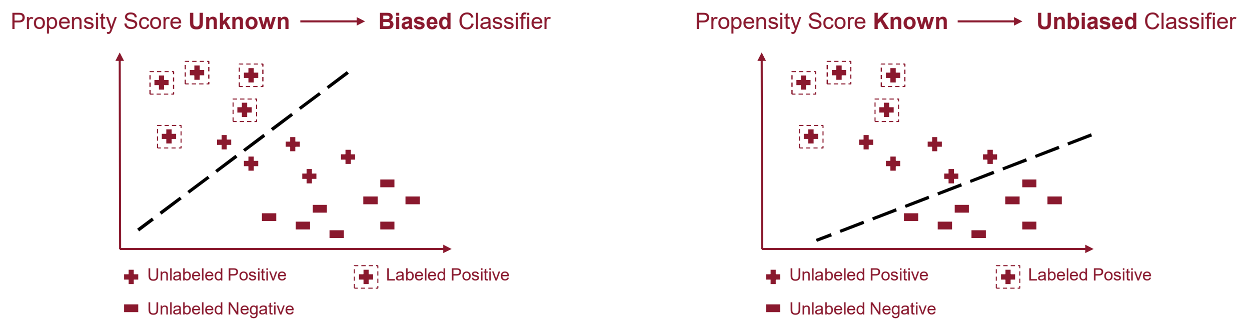

Propensity Score **Unknown** → **Biased** Classifier
Unlabeled Positive
Labeled Positive
Unlabeled Negative
Propensity Score **Known** → **Unbiased** Classifier
Unlabeled Positive
Labeled Positive
Unlabeled Negative

Figure 3: Knowledge of the Propensity Score allows us to train unbiased classifier given biased PU data. We thus study when it is possible to learn the Propensity Score in Task 2.

instance, tasks such as object detection [55] and diagnosing disease [56] are naturally multi-label; e.g. multiple objects can be present in the same image, and a person can suffer from multiple diseases simultaneously. Additionally, a biased sample selection is present in many scenarios; for instance, there is the well-known "healthy user bias" wherein healthy individuals are more likely to self-select for clinical trials than the general population [57]. Thus, individuals "labeled" with a positive outcome will not be representative of the distribution of positive outcomes amongst the general population. This dissertation focuses on addressing these remaining challenges in `Task 1` and `Task 2` (for which an overview is given below in Section 1.4).

1.3 Reusing and Repairing Pretrained Models Without Retraining.

1.3.1 Background

It has become common for practitioners and researchers to reuse existing pretrained models [58]. This is due largely to resources such as HuggingFace [3] making it easy to access pretrained deep learning models, as well as many modern deep learning systems being too prohibitively expensive for most individuals and smaller organizations to train on their own [9].

However, off-the-shelf models likely won't be *perfectly* suited to an end user's needs

- e.g., the users' data distribution will be different than the distribution that the model was trained on, and the target classes won't be exactly the ones that are needed. Additionally, as many pre-trained *generative* models are "foundation models" [59] that have been trained on a self-supervised but not task-specific loss, they do not offer a full range of control over their output. In particular, image-generating foundation models are often trained to reproduce the distribution of their training images, but may be *unconditional* in the sense that the user cannot easily choose which class or mode of data to sample from [25]. Moreover, many existing machine and deep learning systems are known to produce biased outputs that under-represent or yield worse performance for certain demographics such as racial and gender minorities [10, 18, 19, 20, 21].

1.3.2 State-of-Art

As model reuse has become more common, adapting and correcting pre-trained models has likewise become an active area of research [60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71]. Many works focus on *fine-tuning* pretrained models, which corresponds to updating some of the learned weights of the model [60, 61, 72]. This allows users to update existing models to perform more accurately on the user's desired target domain [63]. Fine-tuning is also used to debias pretrained models [67, 66, 68], along with data augmentation [73]. Transfer learning, wherein pretrained models are used to initialize or guide models trained on a new target domain [74], is also a common approach for leveraging pretrained models [75, 76, 77].

Methods for converting pretrained unconditional generative models into *conditional* models - such that the user can specify the class for which the model should generate data - typically utilize new, labeled training data as well as train new "helper" networks [25, 78, 79]. These "helper" networks take the form of *Hyper-Networks* [25] - models trained to predict new weights for the generative model - or *mapping networks* that are trained to adaptively sample from parts of the pretrained model's latent space [78, 79].

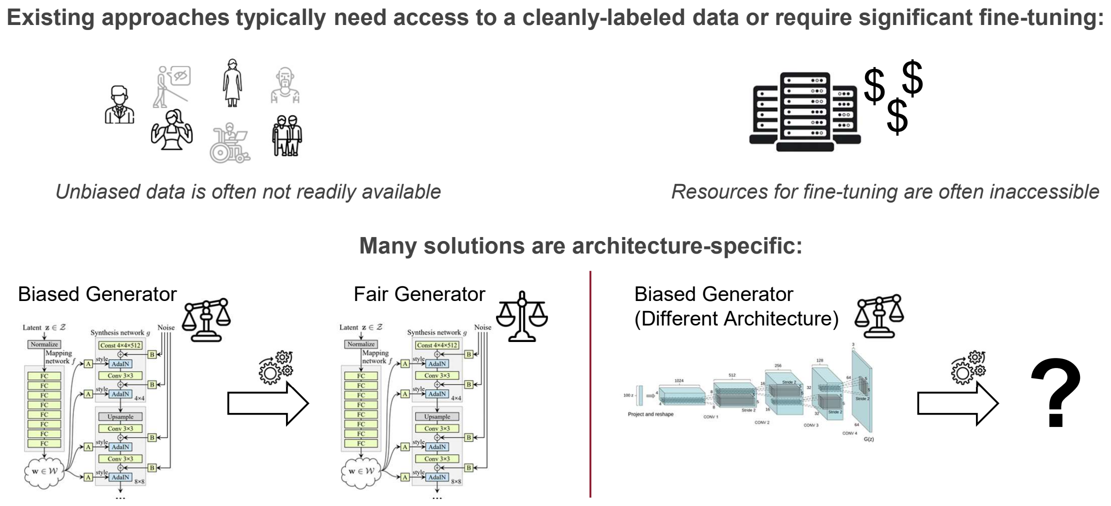

Existing approaches typically need access to a cleanly-labeled data or require significant fine-tuning:
Unbiased data is often not readily available
Resources for fine-tuning are often inaccessible
Many solutions are architecture-specific:
Biased Generator
Fair Generator
Biased Generator
(Different Architecture)
?

Figure 4: Limitations of existing methods for reusing and repairing pretrained models.

1.3.3 Challenges

The requirement for fine-tuning [67, 66, 68] or human feedback [80] limits the applicability of many existing techniques for debiasing pretrained models, as this often requires significant investments in terms of time and computational resources that may be beyond the capabilities of many smaller organizations. Further, techniques designed only for specific architectures [80] that do not generalize are not ideal, as new model types are continuously developed.

Furthermore, many methods for converting unconditional generative models into conditional models likewise require significant computational resources due to their need to train new components [25, 78, 79]. The requirement for labeled training data [25] and access to pretrained unbiased classifiers that already specialize on the classes of interest [78, 79] likewise limit the applicability of existing approaches.

This dissertation thus focuses on addressing these challenges in `Task 3` and `Task 4`, outlined below in Section 1.4.

1.4 Dissertation Tasks

In this dissertation, I tackle the following four tasks. The first two relate to learning from noisy training data, and the second two relate to reusing pretrained generative models.

Task 1: Extending Positive Unlabeled Learning To Multi-Label Data. In general, Positive Unlabeled techniques assume that there are only two classes: the positive class for which some instances are labeled and others are unlabeled, and the negative class that is completely unlabeled. In this task, I extend Positive Unlabeled learning to the *multi-label* setting, where multiple classes can apply to the same instance simultaneously. For example, when multiple objects are present in the same image. In this new setting, each class *may* be labeled for a particular instance if the class applies and is unlabeled otherwise.

First, I propose an unbiased risk minimization strategy that computes a loss equivalent in expectation to the loss a classifier would obtain on *clean* data, given only multi-label PU data. Next, I propose a system that leverages my proposed risk along with a model that can explicitly learn and leverage the dependencies and correlations *between* classes. This system is thus able to utilize knowledge of which classes co-occur, such as *dogs* and *Frisbee's* being correlated in images, in order to more accurately predict the full set of *true* classes associated with each instance, given only partially annotated training data. Further, we experimentally validate that our approach consistently outperforms existing approaches for correcting for missing labels on a variety of common benchmark multi-label datasets.

As many real-world multi-label datasets are naturally Positive Unlabeled while existing multi-label classifiers typically assume full and accurate label sets, this work provides a step forward on extending multi-label classifiers to more realistic data paradigms. **This work is published at CIKM 2022 [81].**

Task 2: Modeling Biased PU Sample Selection. In this task, I study when it is possible to build a model of the sample selection process for biased PU data. In the biased PU setting, some instances are more likely than others to be selected to be labeled. For instance, a fraudulent credit card charge for a large and atypical purchase is more likely to be "labeled" as fraud than a charge for a smaller amount at a location where the card holder usually shops. In this setting it is desirable to have a model that can provide the likelihood that a given positive instance is to be labeled, known as the *propensity score*. A model of this propensity score would allow us to correct for the biased sampling procedure, and train more accurate and robust classifiers on biased PU data.

However, it is not well-known *when* a propensity score model can be learned from biased PU data, or *how* to learn it under settings where it is theoretically possible. I first analyze data distribution assumptions that are commonly used by *unbiased* PU methods, and determine under which of these settings the propensity score can possibly be learned. I identify two sets of assumptions where the propensity score can be recovered, and propose methods for learning a model of the propensity score in these settings. In a series of experiments on a range of datasets and different biased labeling functions, I determine that our proposed approaches nearly always outperform PU methods that don't account for a biased sampling procedure. Additionally, we nearly always outperform the few biased PU methods recently proposed in the literature.

This work has practical importance as more accurate models of the labeling function, such as the ones I propose here, allow practitioners to train more accurate and less biased models given only biased and incompletely labeled data. **This work is published**

at AAAI 2022 [82].

Task 3: Debiasing Pretrained Generative Models Without Retraining. Existing pretrained generative models are known to produce biased outputs, such as image generators vastly over-representing white males. In this task I thus propose a method to *debias* pretrained models belonging to a common broad family of generative models; namely, models that map from a low-dimensional feature space to a higher-dimensional data space. This includes the common classes of generative models such as GANs and VAEs.

More specifically, I formalize a definition of what it means for a generative model to be unbiased by defining the concept of *uniform semantic distributions*. A generator with a *uniform semantic distribution* is one where the semantic attributes of its output have a uniform distribution; e.g., a semantically uniform generator produces an equal number of images of people from every race. I then propose strategies for reconditioning a generative model to produce a semantically uniform output. This approach learns a new distribution for the generator's input (latent) space, debiasing the model without requiring fine-tuning.

We prove that our resampling approach is optimal under the constraints of the problem setting, producing a distribution with minimal divergence from a uniform distribution over the semantic attribute space. Additionally, we outline the conditions for which reaching a perfectly fair distribution (i.e. a distribution with 0 divergence from the uniform distribution) is achievable. Through an experimental evaluation on a range of datasets and pretrained generative models, we show that our approach consistently produces a less biased output than existing methods for debiasing pretrained generative models.

This work provides a step forward on inexpensively debiasing generative models, which is a crucial problem to solve as generators are increasingly being used in high-stakes applications such as augmenting medical classifiers where it is imperative to have minimal bias. **This work was accepted to NeurIPS 2023 [83].**

Task 4: Converting Unconditional Pretrained Generators into Conditional Models Without Retraining or Supervision. Many pretrained generative models are *unconditional*, and do not allow the user to easily sample from particular classes or modes in the data space. However, being able to perform *conditional* generation that can selectively sample from different classes or modes is often an inherently desirable functionality. Thus, in this last task I propose an inexpensive method to convert unconditional generators into conditional models.

Existing approaches will typically require some combination of 1) additional model training, 2) access to classifiers for each mode, and 3) labeled data. In contrast, our approach does not require fine-tuning the generative model, training data, nor any supervision or signal indicating which class any generated instance belongs to. I achieve this by first leveraging the empirically observed property that for many high-dimensional datasets, each class exists on a separate sub-manifold (e.g., distinct and disconnected regions) of the data space, recently validated on image data by Brown et al. [84]. As generative models nearly always utilize continuous functions with a connected input space, their output will likewise be connected - and thus the generator must place a few samples *in-between* these sub-manifolds in the output space. We thus identify which regions in the generator's input space are mapped to low-probability regions in the output space, corresponding to the regions in-between the sub-manifolds. By removing these regions from the input space, we break the input space into distinct subsets such that each of these subsets corresponds to a different sub-manifold in the output space. We can then fit simple Gaussian mixture models on each of these regions in the input space - a process that does not require significant computational resources - and use these mixture models to selectively sample from the region corresponding to any desired output sub-manifold.

Our experimental evaluation indicates that these sub-manifolds are indeed strongly correlated with distinct classes, validating that our unsupervised approach indeed converts unconditional models into conditional generators. This approach can become increasingly useful as pretrained generative models are leveraged more and more in modern deep learning systems.

1.5 Organization of this Dissertation

The rest of this dissertation document is structured as follows:

- **Chapter 2: Extending Positive Unlabeled Learning To Multi-Label Data.**
  This chapter covers `Task 1`.
- **Chapter 3: Modeling Biased PU Sample Selection.**
  This chapter covers `Task 2`.
- **Chapter 4: Debiasing Pretrained Generative Models Without Retraining.**
  This chapter covers `Task 3`.
- **Chapter 5: Converting Unconditional Pretrained Generators into Conditional Models Without Retraining Or Supervision.**
  This chapter covers `Task 4`.
- **Chapter 6: Conclusion.**
  This chapter covers a summary of the contributions of this dissertation, along with promising directions for future work.
- **Chapter 7: List of Publications**
  A list of the papers I have completed, both as first author as well as co-authored

papers.

2 Extending Positive Unlabeled Learning To Multi-Label Data.

This work was published at CIKM 2022:

Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Emmanuel Agu, Elke Rundensteiner. "Robust Recurrent Classifier Chains for Multi-Label Learning with Missing Labels". CIKM 2022.

2.1 Motivation

Most multi-label classifiers make the strong assumption that all classes-of-interest of interest for each datapoint are labeled. For instance, if an image's labels are *vehicle* and *animal*, but not *person*, during learning we assume a *person* is not present in the image. In practice, however, an image's labels are often only a subset of its true classes [85]. Such *incomplete* labeling is caused by the high cost of labeling every possible class for each image and by the propensity of human annotators to miss objects in images [86]. As annotators typically *only* identify classes that *do* apply to an instance, and do not explicitly list every of the potentially hundreds or thousands of objects that do *not* apply, an incompletely-labeled dataset generally implies that unlabeled positives are not distinguished from true negatives. Thus, for each class some positive instances will be labeled while the remaining positives and all negatives go unlabeled. This exactly matches the the *Positive Unlabeled* (PU) setting. Specifically, for each class some positive instances will be labeled while the remaining positives and all negatives go unlabeled.

Unfortunately, PU methods generally assume a binary classification setting, and do not model the relationships between classes [26]. In contrast, leading multi-label classifier explicitly leverage label correlations yet assume completely labeled data [87, 88]. The focus of this work is thus to extend PU to work with a leading approach for multi-label

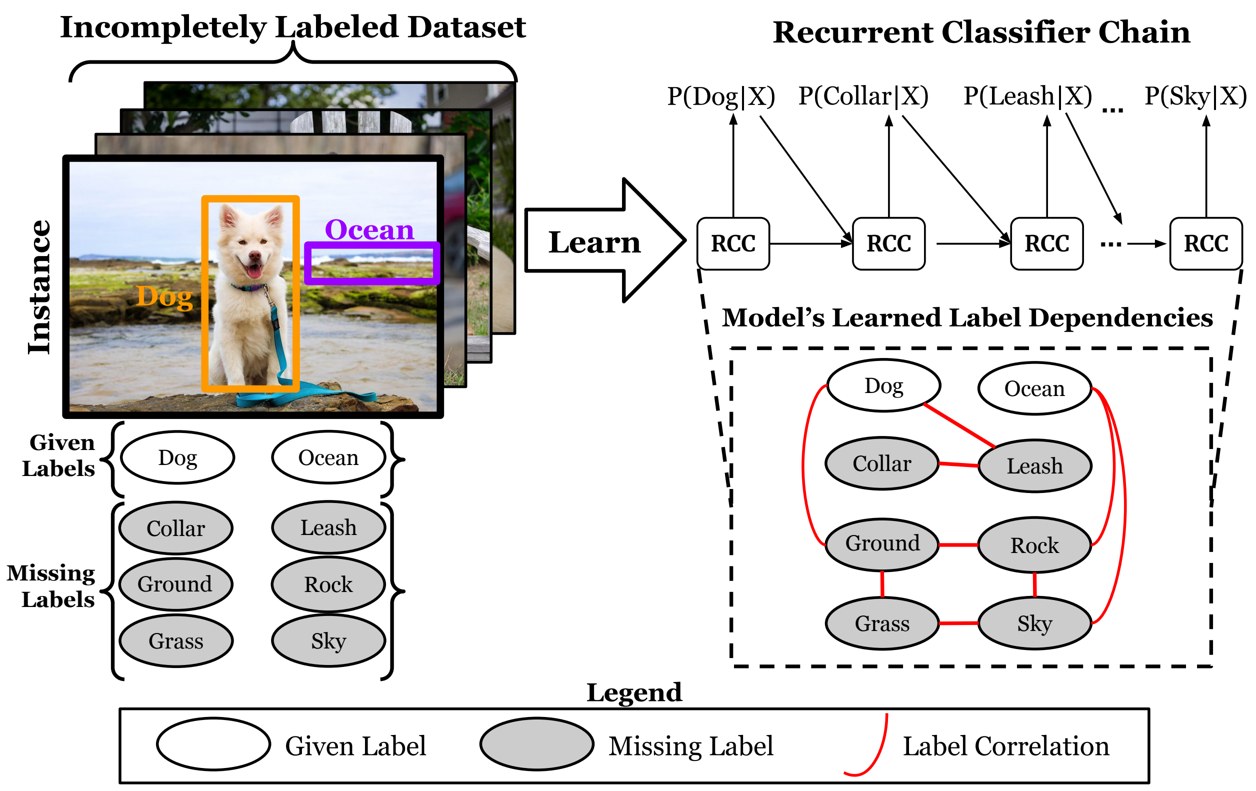


Figure 5: Goal: train a recurrent classifier chain to learn label dependencies on multi-label data with missing labels.

classification. Specifically, we aim to make the *Recurrent Classifier Chain* model robust to multi-label PU data. An example of this task is shown in Figure 5.

2.2 Related Work

Classifier Chains. Standard classifier chains (CCs) consist of a sequence of classifiers, each of which is trained to predict a single class while taking in observed data features as well as preceding class labels as input [88]. By conditioning each label prediction on those previously predicted labels, classifier chains succeed to learn joint label dependencies. While these classic methods use independent models for each predicted label [89], most recent works use recurrent neural networks (RNN) [87, 90, 91, 92]. Such *recurrent classifier chains* allow for parameter sharing between label predictions, often leading to better performance [87]. The most recent RCCs are order-free, meaning, they also learn the best label orderings [90, 91], which we adopt in this work. However, as all RCC methods rely on the strict assumption that all training data are labeled perfectly [87, 90, 91], they fail when trained on multi-label PU data.

Multi-Label Learning with Incomplete Labels. Multi-label classification from incompletely-labeled data is an active area of research [93, 94, 95, 96, 97, 98, 99]. The broad category of learning with incomplete labels encompasses several problem settings [100].

Semi-supervised multi-label learning (SS-ML) assumes that the training data comes in the form of a fully-labeled subset and an unlabeled subset [101, 102, 103, 104, 105]. This does not match our problem setting, where the set of labels *applied to an instance* may be incomplete.

Explicit Multi-Label learning with Missing Labels (Explicit MLML) differ from SS-ML by allowing each *instance* to be partially labeled [100]. They assume that for each class a given instance is given either a positive, negative, or explicitly *missing* label [106, 107, 108, 109, 85]. This means that they assume that explicit negative labels are given in addition to some missing labels. As we discussed in the introduction, assuming the availability of negative labels is often unrealistic.

Implicit Multi-Label learning with Missing Labels (Implicit MLML) Multi-label PU can be thought of as *implicit MLML*. Implicit MLML methods assume that for each class, an instance is either given a positive label for that class or else receives no label for the class [110, 111, 112, 99, 93, 113, 114]. In other words, no explicit negative labels are given and unlabeled classes could be either positive or negative. Note that implicit MLML methods can be applied to explicit MLML data by simply disregarding the negative labels. Alternatively, explicit MLML methods generally can't be applied to implicit MLML data as these methods may require explicit negative labels.

Existing implicit MLML methods optimize for metrics such as hamming accuracy and ranking loss [99], and are thus not suitable for training RCCs. This is because those metrics can be optimized for without learning the joint conditional interdependencies [115], in contrast to the motivation behind RCCs which is to learn such dependencies. Others require specific architecture choices [104] that make them incompatible with RCCs. *To-date, no method has been proposed that extends RCCs into the multi-label PU setting.*

Positive Unlabeled Learning. Positive Unlabeled (PU) learning is very closely related to implicit MLML. Like MLML, PU learning assumes that some positive labels are given while negative instances are not labeled [116]. Unlike MLML, PU methods are classically *binary* classification problems [117, 118], not multi-label. In this sense, implicit MLML can be seen as being synonymous with multi-label PU. PU learning is also an active area of research [119, 120, 121, 122, 123], with recent works showing that unbiased positive-negative risk minimization can be achieved in both the standard setting [124] and even when the labels are applied with a selection bias [54]. However, due to focusing on binary classification, classic PU methods are not applicable to RCCs. Methods such as RankPU [99] optimize for ranking loss, which can be optimized for without learning label dependencies [115] and is thus not appropriate for training RCCs.

2.3 Problem Definition

Formally, we define our problem as follows: Let $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^n$ be a dataset consisting of n pairs of input features $\boldsymbol{x}_i$ of an instance and its incomplete label sets $\boldsymbol{y}_i^*$. For the sake of readability, we drop the subscript i when referring to a particular instance, when none-ambigious. Let $\boldsymbol{y}^*$ be represented as a vector of labels, and $[\boldsymbol{y}^*]_k$ be the kth element of $\boldsymbol{y}^*$. As $\boldsymbol{y}^*$ is the vector representation of the *incomplete* label set, $[\boldsymbol{y}^*]_k = 1$ implies the kth class applies to the instance, while $[\boldsymbol{y}^*]_k = 0$ implies that the kth class can be either positive or negative—the true value is unknown. This implies that in our incompletely label setting, we have no explicit negative labels for any class. We assume that the labels are missing at random, as is standard [99]. This means that the probability that a true positive instance of a class k is labeled is some constant value c_k; i.e., $p([\boldsymbol{y}^*]_k = 1 \mid [\boldsymbol{y}]_k =$

$1) = c_k$, where $\boldsymbol{y}$ is the fully labeled ground truth vector such that $[\boldsymbol{y}]_k$ is the true value of the k-th class for this instance. Our goal is to train a classifier $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, where $\boldsymbol{y} \in \mathcal{Y}$ is the corresponding completely-labeled version of $\boldsymbol{y}^*$, given only observations from $\mathcal{X}$ and corresponding incomplete label vectors $\boldsymbol{y}^*$. Table 1 lists the meaning of the most important notation used in this work.

Symbol	Meaning
Lowercase bold symbol	A vector
Uppercase bold symbol	A matrix
$\boldsymbol{y}$	Ground-truth full label vector
$\boldsymbol{y}^*$	Incomplete label vector
$\hat{\boldsymbol{y}}$	Predicted full label set
$[(\cdot)]_k$	kth entry in vector $(\cdot)$
π_k	Class prior of kth class
$\boldsymbol{x}$	Input instance
L	Number of possible classes

Table 1: Notation for commonly used symbols.

2.3.1 Background on Recurrent Classifier Chains

RCCs model the conditional joint probability of the labels [87]. More formally, they model $p(\boldsymbol{y}|\boldsymbol{x}) = p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x})$. They accomplish this by factorizing the joint probability as follows:

$$p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x}) = p([\boldsymbol{y}]_1|\boldsymbol{x}) \prod_{i=2}^{L} p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x}), \tag{1}$$

where $[\boldsymbol{y}]_{<i} = ([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_{i-1})$. RCCs model the above as a recurrent neural network. The recurrent network reads in the feature attributes along with the observations of each class sequentially; i.e., at the ith step it reads in the observation for the ith class. It thus parameterizes $[\boldsymbol{y}]_{<i}$ as h_{i-1}, where h_{i-1} is the hidden state of the recurrent network at the $i-1th$ step. Likewise, it gives $p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x})$ as the output of a feed forward network conditioned on h_{i-1}.

As described, the RCC factorizes the classes in a predefined order; i.e., class 2 is predicted after class 1. However, recent RCC methods [90, 91] can predict the classes in an arbitrary order that differs instance-to-instance. Thus, they instead provide an alternative factorize to Equation 1 as:

$$p([\boldsymbol{y}]_1, [\boldsymbol{y}]_2, ..., [\boldsymbol{y}]_L|\boldsymbol{x}) = \prod_{i \in \mathcal{O}(\boldsymbol{x})}^{L} p([\boldsymbol{y}]_i|[\boldsymbol{y}]_{<i}, \boldsymbol{x}), \tag{2}$$

where $\mathcal{O}(\boldsymbol{x})$ is an ordered list of class indices specific to instance $\boldsymbol{x}$.

2.4 Challenges

We identify two core challenges in addressing the multi-label PU setting:

- *No negative labels.* The lack of negative labels and the ambiguity of unlabeled instances (i.e., being either positive or negative) makes the PU setting classically difficult [116].
- *Learnign class dependencies in the PU setting.* It is particularly challenging to learn label dependencies when labels can be missing. For instance, while class 1 and class 2 might be highly correlated, the label for class 2 might be missing from an instance and thus the inferred correlations are weaker. Learning these dependencies is clearly essential for multi-label learning.

2.5 Proposed Approach

2.5.1 Overview of Robust-RCC

In this work, we propose the Robust-RCC, the first RCC for multi-label PU data. The Robust-RCC learns the true conditional label dependencies from incompletely labeled data. Robust-RCC is composed of a featurization network and a recurrent network with the later optimized using a novel reformulation of multi-label risk which we derive in this work.

First, the R-RCC Featurization Network transforms an input instance to a latent vector representation. Second, the latent representation is then fed to a novel *R-RCC Classifier* (R-RCC Backbone), which learns the conditional distribution of the *ground truth* label vector given multi-label PU training data. We achieve this training the R-RCC Backbone using a novel multi-incomplete-label risk function (MILR), which reformulates the multi-label risk to be computable from incomplete labels. This is achieved using knowledge of the class priors, estimated by the R-RCC Prior Estimator from incompletely labeled data.

We first describe the novel risk function. Then, we describe the architecture of Robust-RCC in detail and show how the R-RCC Backbone can learn the order in which to predict classes even when given incomplete labels.

2.5.2 Reformulating the Multi-Label Risk.

In the traditional fully labeled setting, RCCs are trained by minimizing the Binary Cross Entropy (BCE) between the predicted label sets and the true label sets [87]. In other words, they aim to find the parameters θ^* of a model $f()$ that minimizes:

$$\theta^* = \arg\min_{\theta} \mathop{\mathbb{E}}_{\boldsymbol{x},\boldsymbol{y}\sim P_{X,Y}} \mathrm{BCE}(f_\theta(\boldsymbol{x}), \boldsymbol{y}), \tag{3}$$

where $P_{X,Y}$ is the joint probability of features and ground truth label vectors. However, we observe that Equation 3 cannot be calculated in the multi-label PU setting due to requiring the expectation over the completely-labeled instances $\boldsymbol{y}$, to which we have no access. Instead, we propose the following multi-incomplete-label risk that can be computed given only implicit multi-label PU data while still being minimized by the ground-truth label vector:

$$\begin{aligned} MILR = \sum_{k}^{L} \big[\pi_k \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^+} & L^+(f_\theta(\boldsymbol{x})[k]) \\ & +(\pi_k c_k - \pi_k) \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^+} L^-([f_\theta(\boldsymbol{x})]_k) \\ & +(1 - \pi_k c_k) \mathop{\mathbb{E}}_{[\boldsymbol{y}^*]_k^-} L^-([f_\theta(\boldsymbol{x})]_k) \big]. \end{aligned} \tag{4}$$

Here, π_k refers to the k-th class prior, $\pi_k = p([\boldsymbol{y}]_k = 1)$, $c_k = \frac{p([\boldsymbol{y}^*]_k=1)}{\pi_k}$, and $\mathbb{E}_{[\boldsymbol{y}^*]_k^{+/-}}$ refers to $\mathbb{E}_{\boldsymbol{x},[\boldsymbol{y}^*]_k \sim (X,[Y^*]_k=1/0)}$. L^- and L^+ refer to the components of the *decomposed* BCE loss, such that $L^{+/-}$ for a given class is the loss incurred by BCE for that class assuming that the ground truth is positive or negative, respectively. While Positive Unlabeled risk functions have been proposed in the setting of *binary* classification [125], this is the first general PU risk formulation for multi-label learning.

Equation 4 succeeds to optimize the multi-label BCE, requiring only expectations over positive instances ($\mathbb{E}_{[\boldsymbol{y}^*]_k^+}$) and unlabeled instances ($\mathbb{E}_{[\boldsymbol{y}^*]_k^-}$) per individual class. It is important to note that we can approximate these expectations from multi-label PU data, while expectations over negative instances would not be. However, Equation 4 is not useful if it were to produce biased label sets. Fortunately, we can establish that this is not the case, as stated in the theorem below.

Theorem 1. *The expected value of the MILR risk function computed from incompletely labeled data is equal in expectation to the expectation of multi-label binary cross entropy loss computed from the ground-truth full label vectors.*

Proof. Let $f_\theta(x)$ be the estimate of $\boldsymbol{y}$ for instance x such that f_θ is the output of a probabilistic RCC parameterized by θ and $[f_\theta(x)]_k$ is the RCC's estimate of $p([\boldsymbol{y}]_k|x)$. To train a standard RCC, we would minimize the BCE:

$$\theta^* = \arg\min_\theta \mathop{\mathbb{E}}_{x,\boldsymbol{y} \in (X,Y)} \text{BCE}(f_\theta(x), \boldsymbol{y}), \tag{5}$$

where BCE($f_\theta(x), \boldsymbol{y}$) is defined as

$$\begin{aligned}\text{BCE}(f_\theta(x), \boldsymbol{y}) =& \frac{1}{L}\sum_{k}^{L} -[\boldsymbol{y}]_k log([f_\theta(x)]_k) \\ &- (1 - [\boldsymbol{y}]_k) log(1 - [f_\theta(x)]_k).\end{aligned} \tag{6}$$

Inspired by binary Positive Unlabeled (PU) methods [125], we reformulate the risk in Equation 6 to be expressed in terms of only positive and unlabeled instances *in the multi-label setting*.

Let $L^+([f_\theta(x)]_k) = \log([f_\theta(x)]_k)$ and $L^-(]f_\theta(x)]_k) = \log(1 - [f_\theta(x)]_k)$. Then, because the expectation is a linear operator, Equation 5 can be rewritten as:

$$\begin{aligned}\theta^* = \arg\min_\theta \sum_{k}^{L} \mathop{\mathbb{E}}_{x,[\boldsymbol{y}]_k \in (X,[Y]_k)} &\big[L^+([f_\theta(x)]_k) \\ &+ L^-([f_\theta(x)]_k)\big].\end{aligned} \tag{7}$$

Let $\mathbb{E}_{[\boldsymbol{y}]_k^{+/-}}$ refer to $\mathbb{E}_{\boldsymbol{x},[\boldsymbol{y}]_k \sim (X,[Y]_k = 1/0)}$. Then, we split the expectation of the BCE into an expectation of positive and negative instances for each class:

$$\begin{aligned}\theta^* = \arg\min_\theta \sum_{k}^{L} \big[\pi_k &\mathop{\mathbb{E}}_{[Y]_k^+} L^+([f_\theta(x)]_k) \\ &+ (1 - \pi_k) \mathop{\mathbb{E}}_{[Y]_k^-} L^-([f_\theta(x)]_k)\big],\end{aligned} \tag{8}$$

where π_k refers to the k-th class prior, $\pi_k = p([\boldsymbol{y}]_k = 1)$. Next, note that the expectation of L^- over all instances of a given class can be rewritten as:

$$\begin{aligned}\mathop{\mathbb{E}}_{[Y]_k} L^-([f_\theta(x)]_k) =& \pi_k \mathop{\mathbb{E}}_{[Y]_k^+} L^-([f_\theta(x)]_k) \\ &+ (1 - \pi_k) \mathop{\mathbb{E}}_{[Y]_k^-} L^-([f_\theta(x)]_k)\end{aligned} \tag{9}$$

For each class, let $p(x)$ be the PDF of the features, and $p_{+/-/\ell/u}(x)$ be the PDF of the positive instance, negative instances, labeled positive instances, and unlabeled instances respectively. Then, for each class k, $x \sim \pi_k p_+(x) + (1-\pi_k)p_-(x) \sim \pi_k c_k p_\ell(x) + (1-\pi_k c_k)p_u(x)$ [116], where $c_k = \frac{p([\boldsymbol{y}^*]_k=1)}{\pi_k}$ (which can be explicitly calculated from the data given π_k). With that in mind, we can rewrite the expectation of L^- over the negative instances in terms of the expectation over all instances and the expectation over positive instances:

$$\begin{aligned}(1-\pi_k) \underset{[Y]_k^-}{\mathbb{E}} L^-([f_\theta(x)]_k) = &- \pi_k \underset{[Y]_k^+}{\mathbb{E}} L^-([f_\theta(x)]_k) \\ &+ \underset{[Y]_k}{\mathbb{E}} L^-([f_\theta(x)]_k).\end{aligned} \tag{10}$$

And likewise the unconditioned expectation over all instances can be written in terms of positive and unlabeled distributions as

$$\begin{aligned}\underset{[Y]_k}{\mathbb{E}} L^-([f_\theta(x)]_k) =& \pi_k c_k \underset{[Y]_k^+}{\mathbb{E}} L^-([f_\theta(x)]_k) \\ &+(1-\pi_k c_k) \underset{[Y]_k^{*-}}{\mathbb{E}} L^-([f_\theta(x)]_k),\end{aligned} \tag{11}$$

where $\mathbb{E}_{[Y]_k^{*-}} = \mathbb{E}_{x,[\boldsymbol{y}]_k \in (X,[Y^*]_k=0)}$.

We can replace the expectation over negative instances in Equation 8 with the right hand side of Equation 10 and the expectation over unconditioned instances with Equation 11 to arrive at our reformulated BCE loss function:

$$\begin{aligned}\theta^* = \arg\min_\theta \sum_k^L \big[&\pi_k \underset{[Y]_k^+}{\mathbb{E}} L^+([f_\theta(x)]_k) + \\ &(\pi_k c_k - \pi_k) \underset{[Y]_k^+}{\mathbb{E}} L^-([f_\theta(x)]_k) \\ &+(1-\pi_k c_k) \underset{[Y]_k^{*-}}{\mathbb{E}} L^-([f_\theta(x)]_k)\big].\end{aligned} \tag{12}$$

Lastly, as we assume that there is no selection bias in which classes are labeled for each instance, we can write the above expectation over positive instances of the true label set with positively labeled instances of the incomplete label set:

$$
\begin{aligned}
\theta^* = \arg\min_{\theta} \sum_{k}^{L} \big[\pi_k \mathop{\mathbb{E}}_{[Y^*]_k^+} L^+([f_\theta(x)]_k) + \\
(\pi_k c_k - \pi_k) \mathop{\mathbb{E}}_{[Y^*]_k^+} L^-([f_\theta(x)]_k) \\
+ (1 - \pi_k c_k) \mathop{\mathbb{E}}_{[Y^*]_k^-} L^-([f_\theta(x)]_k)\big].
\end{aligned} \tag{13}
$$

□

Theorem 1 implies that we can replace the BCE risk with the MILR risk (Equation 4) in order to train RCCs, without introducing bias into our predicted label sets.

Equation 4 requires us to compute the expectations of losses over feature-label pairs. During the training, since we have finite training data, we thus can replace Equation 4 with the *empirical MILR*, $\overline{MILR}$:

$$
\begin{aligned}
\overline{MILR}(f_\theta) = \sum_{k}^{L} \big[\frac{\pi_k}{n} \sum_{\boldsymbol{x}_i : [\boldsymbol{y}_i^*]_k = 1}^{n} L^+([f_\theta(\boldsymbol{x}_i)]_k) \\
+ \frac{(\pi_k c_k - \pi_k)}{n'} \sum_{\boldsymbol{x}_j : [\boldsymbol{y}_j^*]_k = 1}^{n'} L^-([f_\theta(\boldsymbol{x}_j)]_k) \\
+ \frac{(1 - \pi_k c_k)}{n''} \sum_{\boldsymbol{x}_m : [\boldsymbol{y}_m^*]_k = 0}^{n''} L^-([f_\theta(\boldsymbol{x}_m)]_k)\big]
\end{aligned} \tag{14}
$$

Through Equation 14, we succeed to present the very first risk function that can be used to train an RCC to model the true conditional class distribution even for multi-label PU data. Most importantly, Equation 14 does not require negative labels, yet it minimizes Equation 3.

2.5.3 Robust-RCC Architecture.

We next discuss the Robust-RCC's architecture. There are three main components of the Robust-RCC's architecture: the R-RCC Featurization Network, the R-RCC Backbone model, and the R-RCC Prior Estimator. These components are trained together to minimize Equation 14.

First, the R-RCC Featurization Network, $\mathcal{F}$, produces a latent vector representations of input instances x, namely, $\boldsymbol{v}' = \mathcal{F}(\boldsymbol{x})$. In this work, we use ResNet-18 [126] pre-trained on ImageNet to produce a featurized representation, as we focus on image datasets. However, many alternate options could equally be plugged in for this component based on the nature of the task at hand (i.e., a transformer for text data).

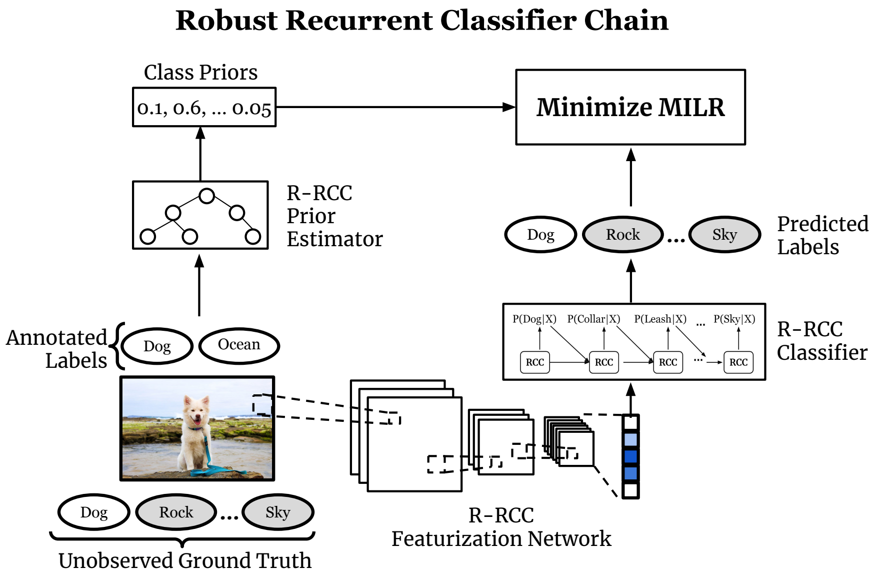


Figure 6: Architecture of Robust-RCC.

Second, the R-RCC Backbone is a recurrent network that takes in this latent representation and produces one class probability per step. At step t, the R-RCC Backbone outputs $[f_\theta(\boldsymbol{x})]_{c_t}$, such that $[f_\theta(\boldsymbol{x})]_{c_t} = p([\boldsymbol{y}]_{c_t} = 1|\boldsymbol{x})$ where c_t is the class predicted at step t. At each step, the input is $\boldsymbol{v} = \boldsymbol{v}' \oplus [f_\theta(\boldsymbol{x})]_{c_{t-1}}$, the concatenation of the feature representation $\boldsymbol{v}'$ with the previous class probability.

We use a *gated recurrent unit* (GRU) for the R-RCC Backbone, though in practice any recurrent network could be used. Thus the predicted class probabilities $[f_\theta(\boldsymbol{x})]_{c_t}$ are given as:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W_r v} + \boldsymbol{U_r h}_{t-1} + \boldsymbol{b}_r) \tag{15}$$

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z\boldsymbol{v} + \boldsymbol{U}_z\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{16}$$

$$\boldsymbol{s}_t = \Phi(\boldsymbol{W}_a\boldsymbol{v} + \boldsymbol{U}_a(\boldsymbol{r}_t \odot \boldsymbol{a}_t - 1) + \boldsymbol{b}_a) \tag{17}$$

$$\boldsymbol{a}_t = \boldsymbol{z}_t \odot \boldsymbol{a}_{t-1} + (1 - \boldsymbol{z}_t) \odot \boldsymbol{s}_t \tag{18}$$

$$[f_\theta(\boldsymbol{x})]_{c_t} = \sigma(\boldsymbol{W}_f\boldsymbol{a}_t + \boldsymbol{b}_f), \tag{19}$$

where $\boldsymbol{W}_{r,z,a}$ and $\boldsymbol{U}_{r,z,a}$ are the weight matrices of the GRU and $\boldsymbol{W}_f \in \mathbb{R}^{|h_t|\times 1}$ is the weight matrix of a feed-forward layer used to convert the hidden representation of the GRU into a class probability. σ is the sigmoid function and Φ is the hyperbolic tangent function.

The final component of the Robust-RCC is the R-RCC Prior Estimator, which estimates the frequency of each class: $\pi_k = p([\boldsymbol{y}]_k = 1)$ for $k = 1, ..., L$. This value is required by our risk function, as discussed in Section 2.5.2. If the class priors are known, they may be substituted here. Otherwise, we use TiCE [127], a leading prior estimation method, to estimate the class prior of each class. TiCE utilizes top-down decision tree induction to estimate the labeling frequency $c = p([\boldsymbol{y}^*]_k = 1|[\boldsymbol{y}]_k = 1)$ in subdomains of the data. Under the assumption that there is no bias in the labeling, subdomains with a higher ratio of labeled to unlabeled instances provide a better estimate of the labeling frequency. The class prior can be recovered form the labeling frequency by the simple conversion $\pi_k = p([\boldsymbol{y}^*]_k = 1)/c$ [116].

Order-free Classification. The order in which classes are predicted should be learned and not pre-determined, because $p(\boldsymbol{y}|\boldsymbol{x})$ can be factorized into any order of conditional probabilities. However, most RCCs are forced to predict labels in a pre-defined order (often frequent-to-rare or rare-to-frequent [87, 88]) despite the large impact of such selection on performance [88]. To overcome this, recent work has shown that classification can be improved with *order-free* approaches, where the RCC learns the order in which to predict classes based on the input. In our work, we use such an order-free approach inspired by [90]. We show experimentally that this choice is well justified. To predict a new class label at step t, our model chooses from the set of previously not yet predicted labels $\mathbb{C}'_t$:

$$c_t = \arg\max_{c' \in \mathbb{C}'_t} [p_t]_{c'} \tag{20}$$

$$f_{\theta_t} = f_{\theta_{t-1}} + [p_t]_{c_t} \tag{21}$$

$$\mathbb{C}'_t = \mathbb{C}'_{t-1} - \{c_t\}, \tag{22}$$

where c_t is the prediction of the class predicted at the t-th step, p_t is the predicted distribution over labels at step t, and $[p_t]_{c_t}$ is the marginal for that class.

2.6 Experiments

2.6.1 Datasets and Metrics.

We evaluate the Robust-RCC on the following three multi-label datasets using four metrics.

`PASCAL VOC 2007`[1] `[128]`: This standard multi-label image dataset consists of 9,963 natural images.

`Scene`[2] `[129]`: This dataset contains 2407 scenery images, each with up to six labels: *beach*, *sunset*, *fall foliage*, *field*, *mountain* and *urban*. Instead of using ResNet-18, these images have already been featurized into 294-dimensional vectors corresponding to the spatial color moments in the LUV space.

`Corel 5k`[3] `[130]`: This dataset is made up of 5,000 images taken from the Corel Photo Gallery.

[1]http://host.robots.ox.ac.uk/pascal/VOC/voc2007/, https://www.flickr.com/help/terms
[2]http://www.uco.es/kdis/mllresources/#SceneDesc, license: PDDL
[3]https://github.com/corel-5k-pytorch/corel-5k, license: Non-comercial use only

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.395**±**0.012**	0.000±0.000	0.068±0.022	0.089±0.082	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.557**±**0.018**	0.021±0.017	0.260±0.015	0.073±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.563**±**0.056**	0.154±0.035	0.347±0.012	0.041±0.026	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.599**±**0.036**	0.323±0.028	0.419±0.010	0.162±0.053	0.000±0.000	0.000±0.000	0.021±0.000
	50%	**0.575**±**0.028**	0.465±0.036	0.468±0.008	0.103±0.073	0.010±0.008	0.007±0.010	0.170±0.000
Hamming Loss	10%	**0.084**±**0.003**	0.127±0.000	0.118±0.003	0.181±0.029	0.127±0.000	0.127±0.000	0.127±0.000
	20%	**0.061**±**0.002**	0.124±0.002	0.093±0.003	0.199±0.043	0.127±0.000	0.127±0.000	0.127±0.000
	30%	**0.059**±**0.006**	0.106±0.005	0.081±0.001	0.207±0.033	0.127±0.000	0.127±0.000	0.127±0.000
	40%	**0.054**±**0.003**	0.083±0.005	0.071±0.001	0.152±0.030	0.127±0.000	0.127±0.000	0.124±0.006
	50%	**0.058**±**0.002**	0.065±0.004	0.065±0.001	0.177±0.028	0.125±0.001	0.126±0.001	0.106±0.010
Macro F1	10%	0.401±0.024	0.000±0.000	0.034±0.008	**0.589**±**0.050**	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.646**±**0.029**	0.013±0.013	0.169±0.013	0.587±0.074	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.670**±**0.046**	0.087±0.025	0.283±0.013	0.585±0.061	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.708**±**0.026**	0.274±0.047	0.374±0.019	0.638±0.053	0.000±0.000	0.000±0.000	0.034±0.067
	50%	**0.678**±**0.018**	0.489±0.046	0.446±0.015	0.619±0.048	0.006±0.006	0.006±0.006	0.243±0.161
Micro F1	10%	**0.626**±**0.016**	0.000±0.000	0.122±0.045	0.551±0.042	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.749**±**0.011**	0.038±0.031	0.427±0.030	0.545±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.765**±**0.019**	0.284±0.056	0.535±0.013	0.536±0.035	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.786**±**0.010**	0.511±0.041	0.617±0.010	0.607±0.044	0.000±0.000	0.000±0.000	0.041±0.094
	50%	**0.771**±**0.008**	0.659±0.032	0.665±0.007	0.576±0.039	0.023±0.010	0.025±0.010	0.293±0.142

Table 2: Performance of each method on the `Pascal VOC 2007` benchmark dataset.

These three datasets were chosen as they are standard image datasets that are naturally multi-label.

Feature Representations Each method used a feature representation of the images in each dataset. For the `Corel 5k`[130] and the `PASCAL VOC 2007`[128] datasets, we used a pretrained ResNet-18 [126] model to featurize the input images into 512-dimensional vectors. Specifically, we used the pretrained ResNet-18 model available in PyTorch, and extracted the feature representations from the final average pooling layer. The `Scene`[129] dataset was already featurized into 294-dimensional vectors corresponding to the spatial color moments in the LUV space, so we did not use the ResNet-18 model on this dataset and instead used these pre-computed features.

We use **four standard multi-label metrics**: *subset accuracy*, *hamming loss*, *macro F1*, and *micro F1*. The subset accuracy is of particular interest to us, as optimizing for this metric means that the model must learn the dependencies between labels [115]. We report on the top 10 labels for each dataset.

2.6.2 Compared Methods.

We compare Robust-RCC against the following state-of-the-art methods for learning with incomplete labels:

Positive Unlabeled Classifier Chains (PU CC) [89, 117]. We train a classifier chain using the standard PU method modification technique [117], as proposed in [113].

SMiLE [98]: This recently-proposed method for learning from incomplete labels uses a graphical model to learn correlations between classes. It optimizes its predictions to preserve the learned class correlations. Unlike our method, *SMiLE* learns the *unconditional* class relations rather than the *conditional* relations.

RankPU [99]: Similar to Robust-RCC, *RankPU* extends Positive Unlabeled learning to the multi-label setting. However, *RankPU* is designed to optimize for ranking algorithms and is not applicable to RCCs.

CleanLab [131]: This identifies instances that are likely mislabeled and removes them before training a classifier. As *CleanLab* is designed to work with any probabilistic classifier, we compare against two versions: one using an RCC as classifier, and the other using the ensemble-based classifier chain.

Recurrent Classifier Chain [87]: We compare against an RCC that treats all unlabeled instances as true negatives. This is the standard approach for maximizing subset accuracy in the fully-labeled setting. We expect other methods to outperform this approach as it does not naturally account for the incompletely labeled nature of the data.

2.6.3 Implementation Details.

Base RCC Architecture. Robust-RCC, RankPU[99], the Positive Unlabeled Classifier Chain (PUCC) [89, 117], and the Recurrent Classifier Chain (RCC) [87] were each implemented in PyTorch [132]. The recurrent methods (Robust-RCC and RCC) consisted of a 1-layer GRU with a hidden space size of 100. Additionally, each recurrent method had a 1-layer feed forward network to map from the 100 dimensional latent space into prediction prob-

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.308**±0.095	0.000±0.000	0.060±0.059	0.113±0.054	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.424**±0.116	0.000±0.000	0.068±0.062	0.181±0.125	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.474**±0.115	0.001±0.001	0.109±0.071	0.148±0.074	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.509**±0.087	0.010±0.005	0.078±0.066	0.123±0.027	0.000±0.000	0.000±0.000	0.048±0.078
	50%	**0.415** ±0.039	0.021±0.008	0.099±0.056	0.142±0.024	0.134±0.072	0.010±0.004	0.230±0.062
Hamming Loss	10%	**0.149**±0.016	0.181±0.000	0.347±0.124	0.264±0.035	0.181±0.000	0.181±0.000	0.181±0.000
	20%	**0.140**±0.008	0.181±0.000	0.347±0.083	0.226±0.057	0.181±0.000	0.181±0.000	0.181±0.000
	30%	**0.124**±0.016	0.181±0.000	0.305±0.111	0.235±0.033	0.181±0.000	0.181±0.000	0.181±0.000
	40%	**0.128**±0.013	0.180±0.002	0.374±0.115	0.249±0.023	0.181±0.000	0.181±0.000	0.173±0.013
	50%	**0.127**±0.008	0.178±0.001	0.314±0.081	0.233±0.027	0.158±0.022	0.179±0.001	0.143±0.012
Macro F1	10%	0.305±0.061	0.000±0.000	0.261±0.030	**0.575**±0.028	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.605±0.056	0.000±0.000	0.274±0.016	**0.606**±0.046	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.646**±0.057	0.003±0.003	0.310±0.035	0.606±0.027	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.698**±0.017	0.009±0.017	0.308±0.029	0.600±0.020	0.000±0.000	0.000±0.000	0.079±0.128
	50%	**0.683**±0.025	0.037±0.012	0.322±0.029	0.611±0.032	0.233±0.100	0.174±0.060	0.347±0.084
Micro F1	10%	0.344±0.072	0.000±0.000	0.267±0.023	**0.538**±0.040	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.635**±0.033	0.000±0.000	0.278±0.015	0.585±0.053	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.667**±0.027	0.002±0.003	0.307±0.037	0.581±0.030	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.702**±0.009	0.009±0.017	0.305±0.024	0.573±0.021	0.000±0.000	0.000±0.000	0.086±0.141
	50%	**0.692**±0.011	0.038±0.014	0.318±0.028	0.585±0.023	0.241±0.020	0.175±0.042	0.371±0.087

Table 3: Performance of each method on the `Scene` benchmark dataset.

abilities.

Base Feed Forward Network Architecture. The non-recurrent methods (PUCC and RankPU) consisted of a feed-forward network that mapped from the feature space to a 100 dimensional latent space, replacing the GRU of the recurrent methods. These methods likewise had an additional feed-forward layer to map from the latent space into prediction probabilities.

CleanLab Implementation. The CleanLab methods (CleanLab CC and CleanLab RCC) [131] used the above feed-forward and GRU models respectively. We used the publicly available code for CleanLab in order to identify and remove the unlabeled positives prior to training the classifier components.

SMiLE Implementation. We did not implement our own version of SMiLE [98], as the authors had made the code for this method publicly available. We used their code[4] and the parameter settings used in their paper, although we modified the neighbor parameter

[4]https://github.com/Jopepato/SMiLE

Metric	Percent Labeled	Approaches						
		R-RCC (ours)	PU CC	SMiLE	RankPU	CleanLab RCC	CleanLab CC	RCC
Subset Accuracy	10%	**0.139±0.042**	0.000±0.000	0.005±0.005	0.021±0.018	0.000±0.000	0.000±0.000	0.000±0.000
	20%	**0.247±0.023**	0.001±0.002	0.032±0.015	0.049±0.037	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.293±0.046**	0.020±0.019	0.077±0.012	0.045±0.013	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.321±0.019**	0.077±0.022	0.098±0.010	0.031±0.032	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.304±0.048**	0.158±0.036	0.133±0.016	0.031±0.037	0.003±0.004	0.006±0.011	0.014±0.022
Hamming Loss	10%	**0.137±0.008**	0.156± 0.000	0.155±0.000	0.259±0.028	0.156±0.000	0.156±0.000	0.156±0.000
	20%	**0.122±0.005**	0.156± 0.000	0.152±0.001	0.237±0.044	0.156±0.000	0.156±0.000	0.156±0.000
	30%	**0.119±0.009**	0.152± 0.002	0.148±0.001	0.246±0.016	0.156±0.000	0.156±0.000	0.156±0.000
	40%	**0.118±0.005**	0.140± 0.003	0.143±0.001	0.270±0.069	0.156±0.000	0.156±0.000	0.156±0.000
	50%	**0.120±0.009**	0.124± 0.006	0.138±0.002	0.243±0.037	0.155±0.001	0.154±0.003	0.153±0.003
Macro F1	10%	0.313±0.075	0.000±0.000	0.009±0.011	**0.508**±0.033	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.0493±**0.040**	0.004±0.008	0.051±0.019	**0.521**±0.038	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.542±0.044**	0.035±0.016	0.109±0.017	0.520±0.029	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.582±0.046**	0.136±0.020	0.149±0.014	0.512±0.043	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.585±0.012**	0.274±0.040	0.199±0.015	0.536±0.044	0.030±0.016	0.035± 0.020	0.068±0.063
Micro F1	10%	0.331±0.093	0.000±0.000	0.006±0.006	**0.494**±0.021	0.000±0.000	0.000±0.000	0.000±0.000
	20%	0.493±0.030	0.002±0.004	0.044±0.016	**0.526**±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	30%	**0.567±0.036**	0.048±0.033	0.107±0.014	0.521±0.015	0.000±0.000	0.000±0.000	0.000±0.000
	40%	**0.596±0.037**	0.187±0.035	0.155±0.015	0.505±0.049	0.000±0.000	0.000±0.000	0.000±0.000
	50%	**0.607±0.008**	0.350±0.059	0.217±0.020	0.523±0.037	0.016±0.007	0.021±0.020	0.051±0.050

Table 4: Performance of each method on the `Corel 5k` benchmark dataset.

to 400. This is higher than the number used in their paper, and was modified as the default value produced 0 subset accuracy for nearly all runs. We found the value of 400 for the neighbor hyperparameter to produce optimal results for this method.

Training Hyperparameters. For each method, we used a batch size of 128 and a learning rate of 0.001. We used the Adam optimizer [133] and PyTorch's exponential learning rate scheduler with gamma set to 0.99. Each method was trained until convergence for 200 epochs.

Experiments were performed on a computing cluster, using a Intel(R) Xeon(R) Platinum 263 8160 CPU @ 2.10GHz CPU, an NVIDIA Tesla V100 SXM2 GPU, and 128 GB of RAM.

2.6.4 Classification with Incompletely Labeled Data.

We first demonstrate that the Robust-RCC classifies implicit MLML data more accurately than the five state-of-the-art alternatives. To do this, inspired by the approach taken by

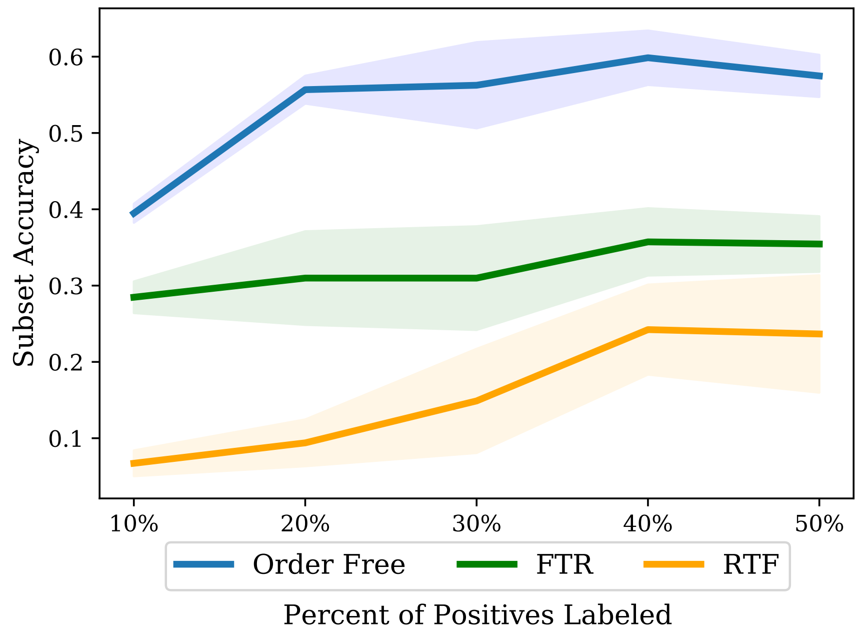


Figure 7: Order-free compared to FTR and RTF Robust-RCC on the `PASCAL VOC` dataset. The shaded region is the 95% confidence interval.

many other incomplete labeling experiments [134, 99, 98, 117, 127, 125], we remove various amounts of labels such that for each dataset only 10% to 50% of the positive instances are labeled. Positive instances from each class thus had the same labeling probability. The results for the `PASCAL VOC`, `Scene`, and `Corel 5k` datasets are shown in Tables 2, 3, and 4, respectively. Notably, the Robust-RCC routinely outperforms all other methods for the important subset accuracy metric. This is expected, as recurrent classifier chains are designed to learn the label dependencies required to optimize subset accuracy.

Interestingly, the Robust-RCC also nearly always outperforms the others on *macro* and *micro F1* metrics. Micro F1 is a measure of classification performance for each label *individually*, and not the label set as a whole. However, if the subset accuracy is very high, then it is expected to be high because a high number of exactly-right label sets implies a high number of individually-correct class predictions. The macro F1 is a measure of how well classification performed per-class and having a very high subset accuracy likewise implies a good macro F1 score by a similar argument. Of note is that the performances of the Robust-RCC and the other PU methods do not monotonically increase as the percentage of labeled data increases. This looks surprising at first, but fits the observations previously made about PU methods [135]. Namely, it has been shown they perform better on incompletely-labeled data, as unlabeled points can act to regularize the model [135].

The next-best performing methods are SMiLE [98] and RankPU, depending on the metric. SMiLE is perhaps the most similar method to the Robust-RCC in intent, as it aims to enforce class correlations during training. However, SMiLE enforces the *unconditioned* marginal class correlations rather than the joint probability of the labels conditioned on the input. This difference gives the Robust-RCC an edge in classification performance. RankPU is similar to the Robust-RCC in that it is a Positive-Unlabeled method, although it does not *explicitly* encourage label correlations. RankPU outperforms the Robust-RCC in macro and micro F1 score in a few cases and generally has a high score for these metrics. Despite this, RankPU has a *low* subset accuracy score. This fits with prior work that shows that a high score on multi-label metrics such as the F-1 scores does *not* imply that label correlations are being learned, as learning the label correlations would imply a high subset accuracy [115]. Other than the baseline RCC model, the *CleanLab* classifiers are the worst performing, likely because *CleanLab drops* instances from the training data that it identifies as unlabeled positives. This significantly decreases the amount of training data when a large proportion of class instances are unlabeled, as is the case in this experiment. This indicates that merely *dropping* likely unlabeled positive instances is not a viable solution in this setting due to the fact that nearly every instance will likely have some class unlabeled.

2.6.5 Ablation Study: Order Free Component

To understand the effect of the order-free classification on the Robust-RCC's performance, we also perform an ablation study comparing the Robust-RCC with two common label prediction orders, frequent-to-rare (FTR) and rare-to-frequent (RTF), on `PASCAL VOC`

2007. As expected, Figure 7 shows that order-free outperforms these preset orderings. Second-best is frequent-to-rare, indicating that it may be better for the Robust-RCC to predict the "easier" classes before the "harder" classes in most cases. Additionally, the order-free predictions have lower variance than the ordered predictions. This implies that allowing learnable orderings does indeed mitigate the challenge of error propagation during label prediction.

2.7 Chapter Summary

In this work, we introduce Robust-RCC, the first approach for training RCCs given multi-label data with missing labels. To achieve this, we introduce the multi-incomplete-label risk (MILR), a novel formulation of the multi-label risk that we prove can safely be computed from incompletely labeled data. With MILR, we succeed to train a recurrent classifier chain to match the distribution of the true fully-labeled data, despite access to only incomplete labels. Using three multi-label datasets, we conclusively demonstrate that our approach outperforms all major state-of-the-art alternatives on four common metrics. Our approach takes a large step forward for multi-label classification by RCCs to be applicable even in domains where fully labeled data is not available.

3 Modeling Biased PU Sample Selection.

This work was accepted to AAAI 2022:

Walter Gerych, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. "Recovering The Propensity Score From Biased Positive Unlabeled Data". AAAI 2022.

3.1 Motivation

Standard Positive Unlabeled learning assumes that a typically-unknown and complex labeling mechanism decides *which* positive instances are labeled. This mechanism is usually an imperfect human annotator with inherent and unobserved biases. The vast majority of recent methods for learning from PU data model this labeling mechanism by assuming that all positive instances are equally likely to be labeled [116]. This is overly simplistic and disregards all *biases* in the labeling mechanism, which naturally lead to certain data points being labeled. For instance, this assumption ignores the fact that individuals with health insurance (who are more likely to visit doctors) are more likely to be be diagnosed than individuals without health insurance. The biased labeling mechanism can also be much more socially innocuous: objects in the foreground of an image are more likely to be labeled than objects in the background.

The key idea of our work is to recover the true complex and biased labeling mechanism by identifying the likelihood that a given positive instance is labeled. Recovering this labeling mechanism is essential for biased PU learning, as success would allow us to train a classifier that distinguish between the positive and negative classes given *only* biased positive and unlabeled data [54]. Additionally, learning the biased labeling mechanism - referred to as the *propensity score* - allows us to recover the posterior of the positive class, which we can integrate over to obtain the class prior. This is important, as knowing the class prior allows us to compute performance metrics for standard positive-negative classification even when the test set includes only positive and unlabeled instances [116]. Lastly, knowing the labeling mechanism gives insight into *why* certain instances were labeled while others were not, which can be important for explainability tasks.

3.2 Related Work

Positive Unlabeled data has been researched for well over a decade [136]. The overwhelming majority of PU works focus on the case where the positively labeled samples are an *unbiased* sample of the true positive distribution [137, 138], and do not address the *biased* PU setting that is the focus of our work. These works range from performing classification [139, 140] to recovering the positive class prior [141, 142, 143].

Recent Positive Unlabeled work has begun to focus on the biased setting, where the labeled positives are a biased sample of the true positives [119, 54, 144, 145]. Many make the assumption that the probability of labeling a positive instances follows the order of the class probabilities [146, 147, 148], which is similar to but slightly more general than the assumption made by our *Probabilistic Gap Scenario* method. However, unlike our work these methods can not and do not recover the labeling mechanism (propensity score), and generally focus on making accurate *binary* classification decisions. A few other works relax this label ordering assumption [144, 149], but likewise do not learn the labeling mechanism and thus do not address the task that is the focus of this work.

The work that is most closely related to ours is the *SAR-EM* method of [54]. This paper focuses on our goal of recovering the propensity score. *SAR-EM* employs an expectation-maximization algorithm to jointly find the true class posterior and propensity score by maximizing the probability of the observed data. However, as we show in the following section, there are an infinite number of possible propensity scores over a wide range of values that perfectly explain the observed data, but are far from the true propensity score.

The method proposed in [119] differs from the other biased PU methods, as it is de-

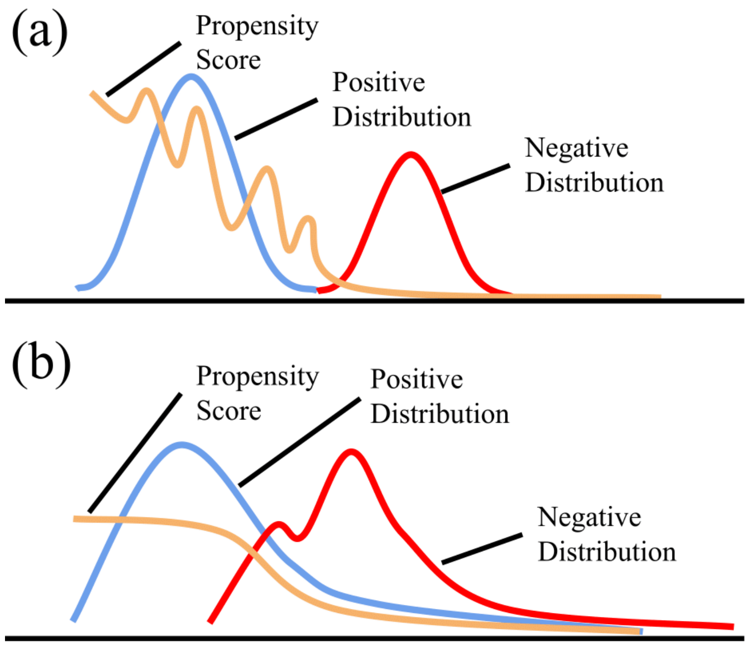


Figure 8: (a) Local Certainty Scenario. Non-overlapping class distributions and arbitrary propensity score. (b) Probabilistic Gap Scenario. Allows for non-overlapping classes, requires propensity score to follow ordering of $p(y = 1|x)$.

signed for recovering the class prior from biased PU data rather than performing classification. This approach assumes that there exists clusters in the distribution of the positive instances, such that the labeling probability is constant *per cluster*. Although not discussed in [119], as this method finds the prior of each cluster, it can be used to recover this constant propensity score per cluster. This differs from our work by assuming that the propensity score is constant per cluster, and thus the propensity score can only take on a few discrete values over the whole data space (limited by the number of clusters).

3.3 Problem Definition

The goal of Positive-Unlabeled (PU) learning is to map features $x \in \mathcal{X}$ into classes $\mathcal{Y} = \{0, 1\}$ given only positive and unlabeled examples. If for a given x the corresponding $y \in \mathcal{Y}$ is 1 then the class is "positive", otherwise the class is "negative". We assume there is a joint distribution $p(x, y, \ell)$, such that $y \in \{0, 1\}$ is the class and $\ell \in \{0, 1\}$ is the *label in-*

dicator. If $\ell = 1$, then the instance is labeled ($y = 1$). If $\ell = 0$, then the instance is unlabeled (and either $y = 1$ or $y = 0$). The class y is unobserved; thus, it is not straightforward to estimate the *class posterior* $p(y = 1|x)$, while the *label posterior* $p(\ell = 1|x)$ can be estimated by training a *non-traditional classifier* [139] to predict the probability of ℓ (which is observed) given x.

We assume the common *single training set scenario* (also known as the *censoring scenario*), in which a sample of data is collected from the joint distribution $p(x, y)$. When an instance is from the positive class, it is labeled with probability $p(\ell = 1|x, y = 1)$, and is unlabeled ($\ell = 0$) otherwise. The alternative PU assumption is the *case-control scenario*, in which the unlabeled data is drawn from the marginal $p(x)$ and another sample of labeled data is drawn from $p(\ell = 1|x, y = 1)$. We describe our method in the single training set scenario as it more commonly used in PU learning [116]. However, like most other PU techniques, it is straightforward to convert between the two.

Recently, a method for directly modeling $p(\ell = 1|x, y = 1)$, known as the *propensity score* and referred to by the symbol e, has been proposed [54]. Theorem 2 shows the value of modeling the propensity score.

Theorem 2. *Let $\hat{y}$ be the predicted posterior probability of y. Then, $\mathbb{E}[R_{prop}(\hat{Y}|E, L)] = R(Y|X)$, where R is the standard empirical positive-negative risk of the predictions $\hat{y} \in \hat{Y}$ and R_{prop} is the propensity weighted risk defined as*

$$\begin{aligned} R_{prop}(\hat{Y}|E, L) =& \frac{1}{n}\sum_{i=1}^{n} \ell_i \left(\frac{1}{e_i}\delta_1(\hat{y}_1) + (1 - \frac{1}{e_i})\delta_0(\hat{y}_i) \right) \\ &+ (1 - \ell_i)\delta_0(\hat{y}_i), \end{aligned} \tag{23}$$

such that $\delta_1(\hat{y}_i)$ is the cost of predicting $\hat{y}$ assuming that y_i is a true positive, and $\delta_0(\hat{y}_i)$ is the cost assuming y_i is a true negative, L is the set of class labels in the dataset and E is the set of corresponding propensity scores.

This theorem, proven by [54], tells us that if the propensity score is known, we can train a classifier to predict the true class y using risk minimization, given only biased positive and unlabeled data. Moreover, this allows us to train a probabilistic classifier to model the class posterior $p(y|x)$, which is useful for uncertainty analysis and for obtaining an estimate for the class prior $p(y)$, which is required to calculate several standard PU classification evaluation metrics [138, 116]. Note that this is in contrast to most other biased PU methods, which attempt to only make accurate binary predictions for y, rather than modeling the class posterior [148, 147]. Furthermore, the propensity score is a model of the complex labeling mechanism that decided *which* positive instances were labeled and thus provides information on *why* certain instances were selected to be labeled while others were not.

The focus of this work is thus to develop approaches that yield an identifiable propensity score.

3.4 Challenges

It is thus crucial to identify scenarios in which the propensity score *is* identifiable, due to its importance for biased PU learning. To yield identifiability, additional assumptions must be made on either the data distribution (specifically, the likelihoods of the positive and negative class) or on the propensity score itself. We thus propose two different estimation procedures: one that makes stronger assumptions on the positive and negative likelihoods but allows for a flexible propensity score, and another that makes stronger assumptions of the propensity score but allows for weaker likelihood assumptions.

3.5 Proposed Approach

3.5.1 Determining When The Propensity Score Is Identifiable

Developing approaches that yield an identifiable propensity score requires understanding when it is even possible for the propensity score to be identifiable. A natural starting place is to consider the four different standard data assumptions commonly in PU literature [116]: Local Certainty/Separable Classes (Bayes Error of 0 between positive and negative distributions), Positive Subdomain (there is some region A of the feature space determined by partial attribute assignment such that the Bayes error is 0), Positive Function (there is some region A of the feature space determined by an arbitrary function for which the Bayes error is 0), Irreducibility (the negative distribution is not a mixture containing the positive distribution).

Theorem 3. *Let propensity score e be an arbitrary function of x, $e : \mathcal{X} \rightarrow (0, 1]$. Let the PU assumption hold (y is unobserved, ℓ and x are observed). Then, e is non-identifiable under the Positive Subdomain, Positive Function, and Irreducibly scenarios.*

Theorem 3, proven in the Appendix, shows that a general propensity score is not identifiable in any of the standard PU data assumptions other than the Probabilistic Gap scenario. Thus, we provide an identifiable propensity score estimation procedure in this setting in the following Local Certainty Propensity Estimation section.

Note that Theorem 3 only holds for a general propensity score. If we make stronger assumptions on the propensity score, we can make less restrictive data assumptions. Thus, we provide an estimation strategy for an identifiable propensity score in the Positive Function assumption in the Probabilistic Gap Propensity Estimation section, in which we assume a linear functional form for the propensity score.

An overview of the difference of assumptions of these two scenarios is shown in Figure 8.

3.5.2 Local Certainty Propensity Estimation

We first describe a method to recover the true propensity score in the Local Certainty scenario. In this setting, we assume the relationship between the observed features and the true class is a deterministic function $f : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the feature space and $\mathcal{Y} = \{0, 1\}$ (where 1 is for the positive class and 0 is for the negative), while allowing the propensity score to be an arbitrary function $e : supp(p(x|y = 1)) \rightarrow \{q \in \mathbb{R} | 0 < q \leq 1\}$;

i.e., an arbitrary function from the feature space of positive instances to a probability between 0 and 1. Note that the propensity score is only defined for regions of the feature space where x may be positive, as in PU learning negative instances are never labeled. The deterministic f is a valid assumption when the observed features are sufficient for uniquely determining the class of each instance. For instance, the features observed for an image (i.e. the pixels) are sufficient for determining which objects are in the image.

The first step to identifying the propensity score under local certainty is to express it in terms of the positive distribution and labeled distribution:

$$\begin{aligned}
e =& p(\ell = 1|x, y = 1) && (24)\\
=& \frac{p(\ell = 1|x, y = 1)p(x, y = 1)}{p(x, y = 1)} && (25)\\
=& \frac{p(\ell = 1, x, y = 1)}{p(x, y = 1)} && (26)\\
=& \frac{p(\ell = 1, x)}{p(x, y = 1)} && (27)\\
=& \frac{p(\ell = 1)p(x|\ell = 1)}{p(y = 1)p(x|y = 1)} && (28)
\end{aligned}$$

While we can take samples from $p(x|\ell = 1)$, we cannot approximate the above density ratio because we cannot sample from $p(x|y = 1)$ and no existing prior estimation method is applicable for estimating $p(y = 1)$ in our biased PU data[5]. We thus propose to replace the $p(x|y = 1)$ in the denominator of the last line with $p(x)/p(y = 1)$.

At a glance, this seems like an arbitrary and poor approximation of $p(x|y = 1)$. However, note that $p(x) = p(y = 1)p(x|y = 1)+(1-p(y = 1))p(x|y = 0)$. Further, under the local certainty assumption there is no overlap between the positive and negative assumptions so for a positive instance x, $p(x|y = 0) = 0$ as there is no ambiguity of class belongingness for any observation. Thus, for positive instance x, $p(x) = p(y = 1)p(x|y = 1) + 0$ and so $p(x|y = 1) = p(y = 1)^{-1}p(x)$. This means that making this substitution in the denominator

[5]More precisely, there is no previously proposed method for calculating this class prior from biased PU data *under the local certainty assumption*.

of Equation 28 *will produce the exactly correct propensity score for all positive instances*. Let e^* then be the value when we swap into the denominator of Equation 28:

$$e^* = \frac{p(\ell = 1)p(x|\ell = 1)}{p(x)}. \tag{29}$$

Note that the true propensity score, given in Equation 28, is undefined for all negative instances under the Local Certainty Scenario, as the denominator will equal 0 when x is outside of the support of the positive class (which all negative instances will be under the deterministic mapping assumption). However, e^* will be 0 for all negative instances, as $p(x|\ell = 1) = 0$ when x is outside of the support of the positive class by the Positive Unlabeled assumption (i.e., no negative instances are labeled). Additionally, note that e^* will *only* equal 0 for negative instances because e is strictly greater than 0 for all positive instances. Thus,

$$e = \begin{cases} e^* & e^* \neq 0 \\ Undefined & e^* = 0 \end{cases} \tag{30}$$

The candidate propensity score e^* can be estimated from the observed PU data because both $p(\ell = 1)$ and $p(x|\ell = 1)/p(x)$ can be calculated from the observed data. $p(\ell = 1)$ can be easily calculated by simply calculating the ratio of labeled vs unlabeled data. The ratio of $p(x|\ell = 1)$ to $p(x)$ can be estimated using *density ratio estimation* [150, 151, 152] by taking samples from biased positives for $p(x|\ell = 1)$ and samples from the unlabeled instances for $p(x)$. Alternatively, this ratio can be calculated as the posterior probability of the label *indicator*. Thus, the output of a flexible probabilistic classifier trained to distinguish *labeled* vs. unlabeled, rather than positive vs. negative, can be used to obtain an estimate for e^*.

Equation 30 is the theoretically accurate value of the propensity score. However, in practice it is useful to have an estimate of e that is defined for all points when using e to train a classifier to determine the true class y. In this case, we propose directly using e^* as the estimated value of the propensity score e for all points. This substitution introduces no bias when used to train such a classifier:

Theorem 4. *Let $\hat{e}$ be an estimate for propensity score e. Then,*

$$bias(R_{prop}(\hat{Y}|E, L)) = \frac{1}{n}\sum_{i=1}^{n} y_i(1 - \frac{e_i}{\hat{e}_i})(\delta_1(\hat{y}) - \delta_0(\hat{y}))$$

This is shown in [54]. Theorem 4 implies that estimated propensity scores of negative instances do not introduce any bias when they are used to estimate $p(y = 1|x)$. Therefore defining the propensity score for negative instances to be 0 (or in fact any value) will not affect the downstream posterior estimates.

Thus, Equation 29 is a good estimate of the propensity score e in the case where the class of each instance is deterministic given the observed features. However, as discussed in our Introduction, for certain tasks the true class can not be determined with 100% certainty. In this scenario, the derivation that leads us to Equation 29 no longer applies, and so e^* may no longer be a good estimate of the propensity score. For this reason, we propose an additional estimation method for e that applies to the probabilistic problem setting, described below.

3.5.3 Probabilistic Gap Propensity Estimation

We now describe how we recover the true propensity score in the probabilistic setting, which we refer to as the *Probabilistic Gap Scenario*. We assume that there is a probabilistic function $f : \mathcal{X} \rightarrow \mathcal{Y}$, such that $f(x) = 1$ with probability $g(x)$, where $g : \mathcal{X} \rightarrow \{q \in \mathbb{R}|$ s.t. $0 < q < 1\}$. Additionally, we assume that The first assumption we now make is that there exists some region of the support of the positive class for which $p(y = 1|x) = 1$; i.e., there is some region without overlap, even if in general there is uncertainty for the class assignment. Informally, one can imagine this being the case for instances very far away from the majority of the negative instances. We note that this corresponds to the common Positive Function data assumption made in prior PU works [116] and described in our Preliminaries section. Unfortunately, as stated in Theorem 3, the propensity score is unidentifiable in this scenario when there are no additional assumptions made on the propensity score. Thus, we identify assumptions on the functional form of the propensity score in order to yield identifiability.

To this end, we base our propensity score assumption of of the common *invariance of order* assumption made by methods for biased PU learning [146, 147, 148]. This corresponds to $p(\ell = 1|x)$ following the order of $p(y = 1|x)$; i.e., if $p(y = 1|x_1) > p(y = 1|x_2)$ then $p(\ell = 1|x_1) = p(\ell = 1|x_2)$. However, this ordering assumption is slightly too weak to yield identifiability.

Lemma 1. *Let the Positive Function scenario and invariance of order assumption hold. Then, the propensity score is not identifiable*

Proof. We show that e is non-identifiable by constructing multiple valid propensity score and class posterior pairs (both of which are latent variables that determine each other, as $p(y = 1|x)\cdot e = p(\ell = 1|x)$). Let the class posterior hypothesis be $p(y = 1|x) = p(\ell = 1|x)^{1/N}$, and let the corresponding propensity score $p(\ell = 1|y = 1, x) = p(\ell = 1|x)^{(N-1)/N}$. In this case, for any positive integer N greater than 1, $p(\ell = 1|x)$ will follow the order of $p(y = 1|x)$, and the propensity score/class posterior pair are valid. Thus, the propensity score is not identifiable under the invariance of order assumption. □

We thus modify the invariance of order assumption slightly by assuming that the propensity score is a linear function of the class posterior; i.e., $e = k \cdot p(y = 1|x)$. Intuitively, this corresponds to assuming that positive instances that have less class ambiguity, or are more typical of the positive class, have a higher likelihood of being labeled. This is a reasonable assumption for many real-world applications. For instance, consider a human annotator who is tasked with labeling the objects in an image. Objects that are obscured, blurred, in the background, or otherwise difficult to identify will be less likely to be labeled than objects clearly in the foreground of the image. Alternatively, think of the case where a credit card company is flagging charges as fraudulent or not fraudulent.

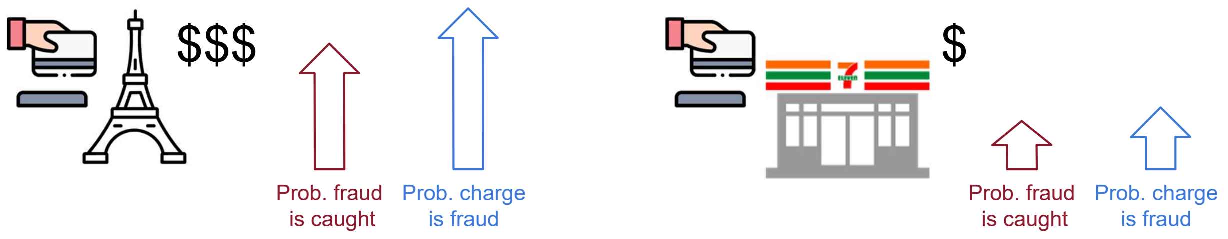


Figure 9: An example of the Scaled Propensity setting.

The likelihood that a charge is flagged (labeled) is higher if the charge is more unusual (higher likelihood of actually being fraud). An example of this is shown in Figure 9.

Let x_i be an instance in the feature space where $p(y_i = 1|x_i) = 1$, as we assume exists according to Positive Function. In this case, the corresponding e_i would be equal to k, as $e_i = k \cdot p(y = 1|x_i) = k \cdot 1$. This implies that $p(\ell_i = 1|x_i)$ would likewise equal k, as $p(\ell_i = 1|x_i) = e_i \cdot p(y_i = 1|x_i) = k \cdot 1$.

Moreover, this implies that $p(\ell_i = 1|x_i) = Sup_{x^* \sim \mathcal{X}}[p(\ell = 1|x^*)]$. Therefore, we can obtain the value of k by modeling the posterior probability of the *label indicator* ℓ, and then finding the value that maximizes this probability. Thus, if $h_\ell(x)$ is a model of $p(\ell = 1|x)$,

$$k = Sup_{x^* \sim \mathcal{X}}[h_\ell(x)]. \tag{31}$$

Next, using Equation 31 we can e from h_ℓ:

$$\begin{aligned}
e \cdot p(y = 1|x) =& h_\ell(x) \\
\frac{e^2}{k} =& h_\ell(x) \\
e =& \sqrt{k \cdot h_\ell(x)} \\
e =& \sqrt{Sup_{x^* \sim \mathcal{X}}[h_\ell(x)] \cdot h_\ell(x)}
\end{aligned}$$

Therefore, we can obtain an estimate of the propensity score in this setting by first training a model of the posterior of the *label indicator*, such that the propensity score is the

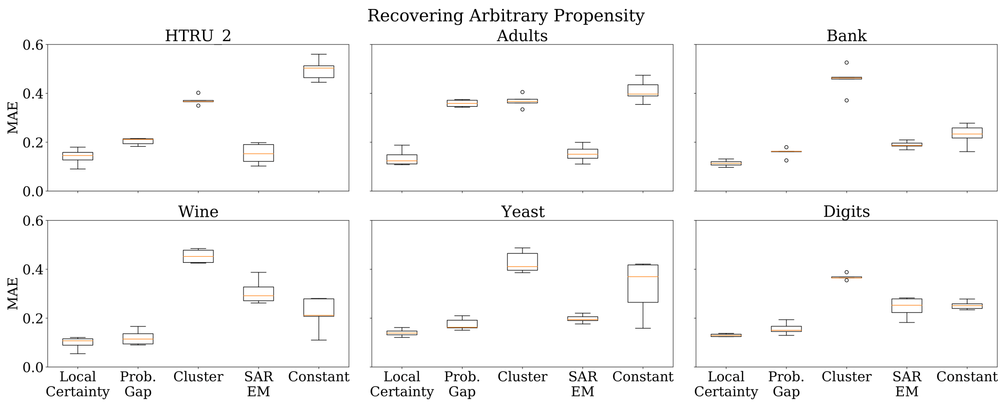


Figure 10: Results for recovering the propensity score for the arbitrary propensity setting. Our Local Certainty method significantly outperforms state-of-the-art on 4/6 datasets, and is never itself outperformed.

root of this posterior probability scaled by a constant that is also obtained from the label posterior.

3.6 Experiments

We study the effectiveness of our Local Certainty and Probabilistic Gap methods on several benchmark datasets. We use a Gaussian Process Classifier [153] to model the label indicator posterior necessary for each method. As implied, we calculate Equation 29 by using the posterior of the label indicator rather than more-complex density ratio estimates. Additional experiments are available in the supplementary materials.

3.6.1 Experimental Setup

Compared methods. We compare against a method that assumes constant labeling as a baseline. Specifically, we employ the state-of-the-art class prior estimation TiCE [141] to find an estimate of $p(y = 1)$, and then obtain an estimate of the propensity score by obtaining $p(\ell = 1|y = 1)$ from the estimated $p(y = 1)$. We call this baseline *Constant*, as it assumes that the propensity score is constant for all positive instances. We also compare against the two existing PU methods that estimate the propensity score: SAR EM [54] and the method proposed in [119] which we refer to as *Cluster*. We use the publicly-available code for TiCE[6] and SAR EM[7] and implement *Cluster* ourselves.

Datasets. We use several standard benchmark datasets from the UCI Machine Learning Repository [154]: `Yeast` [155], `Bank` [154], `Wine` [156], `HTRU_2` [157], `Occupancy` [158], and `Adults` [159].

We likewise use two real-world datasets: `Yelp Reviews` [160] and `PASCAL VOC 2007` [161].

These datasets were chosen to have a wide variety in dataset domains, size, and cardinality. Multi-class datasets were converted into binary classification problems when necessary as was done by [119]. Random 70/30 train/test splits were used for each dataset. Each experiment was repeated 10 times in order to obtain confidence intervals. Additional dataset details are available in the supplemental materials.

3.6.2 Recovering the Propensity Score

We evaluate the ability of each method to recover the propensity scores in two settings: One in which the true propensity score is an arbitrary complex function and there is little to no overlap between classes (the Local Certainty Scenario) and the setting where the Probabilistic Gap assumptions are met (classes overlap and the propensity score is a scaled version of the class posterior). We refer to these as *Arbitrary Propensity* and *Scaled Propensity*, respectively.

Arbitrary Propensity. In this experiment, we decide which positives are labeled according to an arbitrarily complex propensity score. Specifically, we cluster the distances of the positive instances from the mean into 20 bins or clusters using k-means (such that the clustering is on the *distances*, not the positions of the points in the feature space). Each bin is assigned a random propensity score between 0.1 and 0.9. Ten trials are run per dataset and bins are randomly assigned for each. This creates a very complex propensity function

[6]`https://dtaid.cs.kuleuven.be/software/tice`
[7]`https://dtai.cs.kuleuven.be/drupal/software/sar`

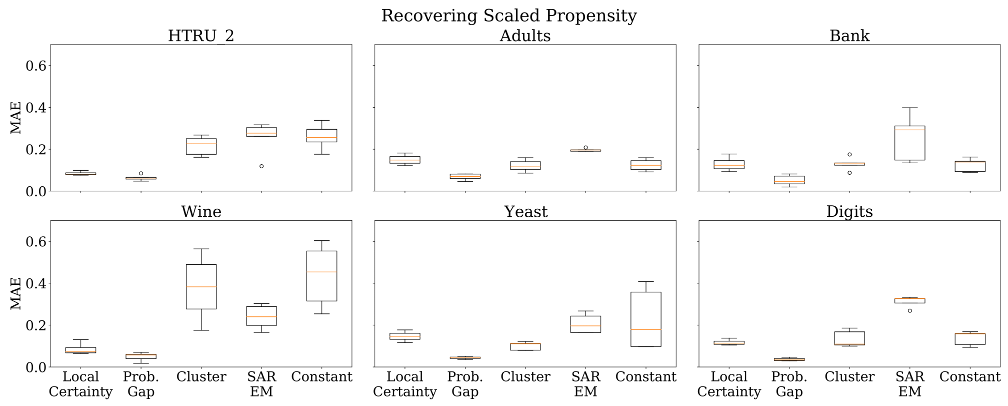


Figure 11: Results for recovering the propensity score for the scaled propensity setting. Our Probabilistic Gap method always significantly outperforms the state-of-the-art.

to test the ability of the Local Certainty method to recover the propensity score without assuming a specific functional form. Additional details on the experimental setup are available in the supplemental materials. Results of these experiments on each dataset are shown in Figure 10, which reports the MAE between the estimated propensity scores and the true propensity score.

As Figure 10 shows, our Local Certainty method significantly outperforms all other methods for all but one dataset where our method ties SAR EM. As expected, the Probabilistic Gap method does not perform particularly well (usually in third place), as the assumptions of this method are not met in this problem setting. SAR EM is generally in second place, which is again expected as, unlike the other non-Local Certainty methods, it does not require a particular functional form for the propensity score. However, SAR EM is likely to converge to an incorrect estimate of the propensity score even when the classes are separable. Our method corrects for this case and so consistently outperforms SAR EM.

Note that these datasets were not modified to enforce the class separability assumption of the Local Certainty scenario. However, previous work has shown that classifiers have been able to achieve close to 100% accuracy on these datasets, indicating that there is already naturally little or no class overlap.

Dataset	Prop. Func.	LC (Ours)	PG (Ours)	Cluster	SE	Constant
`PASCAL VOC`	Arbitrary	**0.13**+/-0.03	0.31+/-0.04	0.45+/-0.04	0.20+/-0.04	0.30+/-0.13
`PASCAL VOC`	Scaled	0.10+/-0.05	**0.09**+/-0.05	0.84+/-0.08	0.12+/-0.08	0.74+/-0.08
`Yelp`	Arbitrary	**0.15**+/-0.02	0.18+/-0.02	0.29+/-0.04	0.21+/-0.07	0.25+/-0.03
`Yelp`	Scaled	0.08+/-0.04	**0.04**+/-0.02	0.28+/-0.03	0.09+/-0.03	0.12+/-0.06

Table 5: Mean absolute error between true and recovered propensity score for each method on two real-world datasets.

Scaled Propensity. We next evaluate the performance of each method when the Probabilistic Gap assumptions are met: the classes overlap and the propensity score is a constant multiple of the class posterior (such that an instance more typical of the positive class is more likely to be labeled). To this end, we introduce class overlap to these benchmark datasets. This is achieved by applying Borderline SMOTE [162] to generate samples along the boundary of the positive and negative class, such that negative samples were generated on the positive side and vice versa. We apply this and ensure a roughly 30% class overlap for each dataset.

The ground truth propensity score in this setting is determined by first training a probabilistic classifier logistic regression model) to find the posterior of the positive class. Then, the propensity score was determined as the posterior model multiplied by a constant k, where k was randomly sampled from 0.3 to 0.8. k was re-sampled for each run, for ten runs per dataset.

Our findings are shown in Figure 11. Our Probabilistic Gap method significantly outperforms all other methods on each dataset, indicating that this approach does indeed more accurately recover the propensity score in this scenario.

Interestingly, we observe that our Local Certainty method is 2nd best, meaning that the Local Certainty method is robust to its class separability assumption being broken.

Real-World Datasets. The previous experiments were conducted on benchmark datasets that were modified to meet our assumptions. To show the robustness of our approach and utility for real-world datasets, we perform experiments on two additional real-world

Dataset	HTRU 2	Adult	Bank	Wine	Yeast	Digits
LC (Ours)	**0.04**+/-0.00	**0.35**+/-0.02	**0.06**+/-0.01	**0.24**+/-0.02	**0.44**+/-0.00	**0.21**+/-0.01
PG (Ours)	0.10+/-0.00	0.40+/-0.00	0.22+/-0.01	0.36+/-0.02	0.47+/-0.00	0.33+/-0.01
Cluster	0.05+/-0.00	0.37+/-0.00	0.09+/-0.00	0.48+/-0.01	0.46+/-0.00	0.23+/-0.00
SE	0.10+/-0.05	0.37+/-0.01	0.09+/-0.01	0.47+/-0.05	0.46+/-0.01	0.33+/-0.03
Constant	0.43+/-0.00	0.70+/-0.01	0.17+/-0.01	0.34+/-0.02	0.46+/-0.00	0.29+/-0.01

Table 6: Classification error for arbitrary propensity score scenario

Dataset	HTRU 2	Adult	Bank	Wine	Yeast	Digits
LC (Ours)	0.14+/-0.01	0.75+/-0.01	0.38+/-0.03	0.26+/-0.01	0.45+/-0.01	0.51+/-0.02
PG (Ours)	0.05+/-0.00	**0.25**+/-0.03	**0.30**+/-0.01	**0.25**+/-0.02	**0.41**+/-0.01	**0.27**+/-0.01
Cluster	**0.04**+/-0.00	0.27+/-0.00	0.33+/-0.00	0.78+/-0.01	0.45+/-0.00	0.51+/-0.01
SE	0.14+/-0.08	0.26+/-0.01	0.35+/-0.01	0.51+/-0.06	0.44+/-0.01	0.49+/-0.03
Constant	0.43+/-0.00	0.46+/-0.03	0.39+/-0.01	0.34+/-0.03	0.46+/-0.00	0.54+/-0.01

Table 7: Classification error for scaled propensity score scenario

datasets: `PASCAL-VOC 2007` (featurized using a pre-trained Resnet-18 model [126]) and the `Yelp` Reviews dataset. The data was not modified to meet our data assumptions (i.e.,we did not ensure separable classes, or add new data using SMOTE). For each dataset, we performed experiments with either the arbitrary propensity score or the linear propensity score, as was done with the other datasets in our paper. Table 5 shows the results of these experiments. Our Local Certainty method still outperforms all other methods on both real-world datasets for the arbitrary propensity score experiment, and performs second only to our other method (Probabilistic Gap) for the linear propensity score (as expected). This shows that our methods still outperform the state-of-the-art even on complex, real-world datasets.

3.6.3 Utility of Recovered Propensity Scores: Using Propensity Sores For Classification

As discussed in the Preliminaries section, the propensity score can be used for classification. We thus illustrate the utility of our estimated propensity scores by comparing the class posteriors obtained from our estimated propensity scores to those obtained form the state-of-the-art propensity estimation methods. We achieve this by training a downstream classifier for each dataset using the propensity-weighted risk (Equation 1). We utilize the propensity scores obtained in the the "Recovering the Propensity Score" experiments; thus, the dataset preparation and details for those experiments hold true for this experiment as well.

The results, shown in Table 6 and Table 7, demonstrate that our Local Certainty method produces a downstream classifier with the lowest (best) error for the Arbitrary Propensity setting (Table 6), and our Probabilistic Gap method produces the best classifier in 5/6 of the datasets in the Scaled Propensity setting (Table 7). This shows that our methods nearly always result in the training of a more accurate classifier than the state-of-the-art propensity-recovering PU methods.

3.7 Chapter Summary

Building a model of the propensity score is a challenging task:

- *No ground truth.* It is generally unrealistic to assume access to a labeled target value for the propensity score; we are typically given only a PU dataset, with no target variable indicating how likely each instance was to be labeled. The propensity score must thus be indirectly inferred.
- *Non-identifiability.* In general, there are an infinite number of potential propensity score functions that would generate the PU data distribution. We must first understand when it is even possible to recover the propensity score, before developing approaches to approximate it.

4 Debiasing Pretrained Generative Models Without Retraining.

This work was accepted to NeurIPS 2023:

Walter Gerych, Kevin Hickey, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Elke Rundensteiner, Emmanuel Agu. "Debiasing Pretrained Generative Models by Uniformly Sampling Semantic Attributes". NeurIPS 2023.

4.1 Motivation

Background. Generative models have become a cornerstone of modern machine learning, allowing for the synthesis of realistic data for many domains, including images [163, 164], audio [165], and text [166, 167]. However, even leading generative models often reproduce the biases present in their training data [19, 20], such as image generation models strongly over-representing white males [21]. As generative models are increasingly used for data augmentation to train downstream models [168] in domains from scientific to medical fields [169, 170], biased synthesized data could lead to results that are skewed or inaccurate. This can exacerbate existing issues such as facial recognition models performing significantly worse on non-white individuals [171] or healthcare models being much less accurate for certain minority groups [24]. Further, with the rapid growth of generative models in commercial applications, the potential financial, legal and ethical costs of biased outputs are significant. Thus, it is imperative to develop methods that mitigate bias in generative models to ensure that their outputs are fair and equitable by generating a roughly equal number of samples of each group. We call such outputs *semantically uniform*, and the attribute that they are uniform over - such as gender or race - the *semantic attribute*.

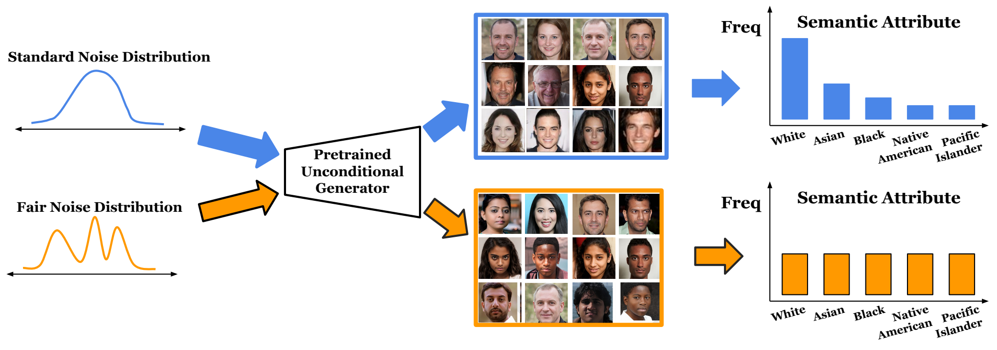


Figure 12: A generator conditioned on a fair noise distribution yields outputs that are uniform over a semantic attribute (i.e. race).

4.2 Related Work

Existing methods for addressing bias in generative models often train a new model from scratch [172, 80, 173], though this is computationally expensive, requires significant labeled data, and wastes resources already previously spent training the existing (biased) generative model. Latent attribute editing methods modify the samples in the latent space of generative models to produce controlled changes in the output, and could potentially be used to correct for bias [174]. However, this requires making limiting assumptions such as that semantic attributes correspond to *linear* directions in the latent space [174, 175, 176, 177, 178, 179]. While some recent advances have been made in uniformly sampling the output space of a pretrained model [180, 181], they can fail to yield uniform samples over *semantic attributes* [180]. For example, they will make an image generator's output uniform over a manifold in the pixel space, but may still overproduce images of white individuals as the output won't be uniform over the *semantic attribute of race*.

4.3 Problem Definition

Assume we are given a pretrained generative model G_θ that maps a latent space $\mathcal{Z}$ to a feature space $\mathcal{X}$, and a pretrained classifier C_ϕ that maps $\mathcal{X}$ to a semantic attribute space $\mathcal{Y}$, where $\mathcal{Y}$ is the set of *group* (*class*) labels. Thus, $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_N\}$, where Y_i is the label of the ith group. When not otherwise ambiguous, we will refer to group Y_i as group

i. Let $\mathbf{y}$ and $\hat{\mathbf{y}}$ be the random variables indicating the true semantic attribute label and the predicted label from C_ϕ respectively. Let E be the prediction-conditional error rates of C_ϕ, with E being a left stochastic matrix such that $E_{i,j} = P(\mathbf{y} = i|\hat{\mathbf{y}} = j)$. Let C_ϕ be a better-than-random classifier, i.e., for N groups, $P(\mathbf{y} = i|\hat{\mathbf{y}} = i) > \frac{1}{N}$ for all groups i. Thus, E is by definition a diagonally dominant matrix. As it is typical for developers to report the error rates of their classifiers, we assume that E is known. While these error rates are typically reported for the classifier's training distribution, we can correct E to apply to the generated distribution under a label shift assumption [182]. See the Appendix for details on this and for the proofs of the upcoming theorems. Lastly, let C' denote an ideal *perfect* classifier that always correctly predicts $\mathbf{y}$ with zero error. C' is hypothetical and unavailable to us.

Our goal here is to sample from a Fair Noise Distribution, defined as follows:

Definition 1 (Fair Noise Distribution)**.** *For a generative model $G_\theta : \mathcal{Z} \to \mathcal{X}$, a distribution $\mathbb{Q}$ over $\mathcal{Z}$ is a **Fair Noise Distribution** with respect to $\mathcal{Y}$ if for $\mathbf{z} \sim \mathbb{Q}$, $C'(G_\theta(\mathbf{z})) \sim Unif(\mathcal{Y})$, where $Unif(\mathcal{Y})$ is the uniform distribution over the groups in $\mathcal{Y}$.*

Intuitively, a distribution $\mathbb{Q}$ is a Fair Noise Distribution if G_θ produces samples that are uniform over the semantic space $\mathcal{Y}$ when conditioned on $\mathbb{Q}$. For example, if G_θ synthesizes images of people and $\mathcal{Y}$ is the set of races, then $\mathbb{Q}$ is a Fair Noise Distribution if conditioning on it makes G_θ produce a roughly equal number of images of people from each race.

More realistically, we want to find a $\mathbb{Q}$ that is easy to sample from and produces reasonable variance from the generator. That is, it does not yield only one unique example of each group, which would otherwise make the problem trivial. Notably, we assume that we do *not* have any real training data (i.e., no real samples from $\mathcal{X}$). Also, we do not aim to retrain G_θ, i.e., not change the parameters θ of G_θ. Additionally, we do not make any assumptions on the differentiability of G_θ or C_ϕ.

See Figure 12 for an example of our desired output.

4.4 Challenges

Our task of producing samples that are semantically uniform has three major hurdles:

1. *Expensive retraining:* It is often prohibitively expensive to retrain large generative models in terms of computational resources and time.
2. *Inaccessible training data:* The data used to train a released generator is typically unavailable; either due to being proprietary or simply too large of a volume for most practitioners to utilize. Thus, we do not have adequate data available to tune the generator or classifier.
3. *Inaccurate classifier:* As we do not have any samples of real data available, we must rely on the possibly imperfect semantic attribute classifier C_ϕ to provide labels. However, since classifiers may produce inaccurate predictions - especially on underrepresented groups of $\mathcal{Y}$, this can cause incorrect estimations of the number of instances for each group.

4.5 Proposed Approach

Our approach for sampling from a Fair Noise Distribution $\mathbb{Q}$ hinges on training a distribution mapping function M_ω such that $M_\omega(\mathbf{z}) \sim f\mathbb{Q}$, where $\mathbf{z}$ is a draw from the original conditioning distribution of G_θ. The functional form of M_ω has many options; for instance, M_ω can be a GAN generator [183], a VAE [184], a DDPM [185], or a normalizing flow model [186, 187]. No matter which form is chosen, we will need a dataset of samples drawn from $\mathbb{Q}$ to train M_ω.

4.5.1 Collecting Fair Samples Using Imperfect Classifiers

A naive method for collecting a dataset of samples distributed according to a Fair Noise Distribution $\mathbb{Q}$ is given in Algorithm 1. The basic approach is to continuously sample $\mathbf{z}$ from the noise distribution, collect the generator's output for each draw, and use the classifier to determine the value of the semantic attribute corresponding to each noise

Algorithm 1 Naively collect data from $\mathbb{Q}$

```
1: procedure NAIVE_Q_DATASET($G_\theta, C_\phi, S$)    ▷ Colect from $\mathbb{Q}$ by trusting $C_\phi$
2:     Input: Pretrained generator $G_\theta$ and classifier $C_\phi$, number of samples $S$ for each attribute
3:     Output: Dataset with $S$ samples for each $y \in \mathcal{Y}$
4:     $Q\_dataset \leftarrow dict()$    ▷ Initialize empty dictionary for the dataset
5:     for $group \in \mathcal{Y}$ do
6:         $Q\_dataset[group] = [\,]$    ▷ Initialize every group as empty array
7:     end for
8:     while $\exists\, key \in Q\_dataset.keys$ s.t. $length(Q\_dataset[key]) < S$ do
9:         $z \sim \mathbb{P}_{noise}$    ▷ Sample noise
10:        $group = C_\phi(G_\theta(z))$    ▷ Get the predicted group of $G_\theta(z)$
11:        if $length(Q\_dataset[class]) < S$ then
12:            $Q\_dataset[class].append(z)$    ▷ If ¡ $S$ samples for that class, add $z$ to the dataset
13:        end if
14:    end while
15:    return $Q\_dataset$    ▷ Return dataset with $S$ samples for each group in $\mathcal{Y}$
16: end procedure
```

draw. As a result, samples of $\mathbf{z}$ corresponding to each group are saved to dataset $\mathcal{D}_\mathbb{Q}$ until $\mathcal{D}_\mathbb{Q}$ has S number of samples for each $y \in \mathcal{Y}$. This procedure results in samples of noise that once passed through G_θ will be uniform across the semantic attribute space *according to C_ϕ*; i.e., for $\mathbf{z} \sim \mathcal{D}_\mathbb{Q}$, $C_\phi(G_\theta(\mathbf{z})) \sim Unif(\mathcal{Y})$. However, this will only yield samples from a Fair Noise Distribution in the case where C_ϕ is a perfect classifier (such that E is the identity matrix). If C_ϕ is an imperfect classifier then the distribution may not be truly fair. For instance, if C_ϕ often incorrectly predicts group j as group i, then instances with attributes matching group i may be more prevalent than those for group j.

Fortunately, we can utilize knowledge from the prediction-conditional error rates E to sample a dataset of noise that will yield more semantically uniform generated instances despite the noisy predictions of C_ϕ. To achieve this, we use a weighted sample of datapoints predicted for each group. Let $\mathbb{P}_{z|C_\phi=i}$ be a distribution over $\mathcal{Z}$ such that the imperfect classifier's prediction of generated samples arising from this distribution are all of group i; $C_\phi(G_\theta(\mathbf{z})) = i$ for $\mathbf{z} \sim \mathbb{P}_{z|C_\phi=i}$. In addition, let $\mathbb{Q}^\lambda = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \mathbb{P}_{z|C_\phi=i}$, such that $\lambda_i > 0 \;\forall\; i$ and $\sum_{i=1}^{|\mathcal{Y}|} \lambda_i = 1$.

We can in many instances find values of $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{Y}|}\}$ such that $\mathbb{Q}^\lambda$ is a Fair

Noise Distribution, as stated in Lemma 2:

Lemma 2. *Let $\mathbf{1}^{|\mathcal{Y}|}$ be the vector of length $|\mathcal{Y}|$ such that every element is 1. Let $cone(E) = \{\sum a_i E_{:,i} | a_i \in \mathbb{R}_{\geq 0}\}$ be the finite convex cone generated by the columns of the prediction-conditional error matrix. If $\mathbf{1}^{|\mathcal{Y}|} \in cone(E)$, then $\exists\ \lambda$ such that $\mathbb{Q}^{\lambda}$ is a Fair Noise Distribution.*

Additionally, we can show that in the case where $\mathbf{y}$ is binary and C_ϕ is better-than-random, we can *always* find λ such that samples drawn from $\mathbb{Q}^{\lambda}$ will definitely yield generated samples that are uniform over the semantic attributes, as shown by the following lemma.

Lemma 3. *Let $|\mathcal{Y}| = 2$ and let C_ϕ be better-than-random, such that the diagonal elements of E are each greater than $\frac{1}{2}$. Then, $\exists\ \lambda$ such that $\mathbb{Q}^{\lambda}$ is a Fair Noise Distribution.*

The proof of Lemma 3 comes from showing that $\langle 1, 1\rangle$ is always in the finite convex cone generated by C_ϕ's prediction-conditional error matrix, with Lemma 2 implying that this property guarantees that $\mathbb{Q}^{\lambda}$ will yield samples that produce semantically uniform generated instances.

Even in the cases where we cannot guarantee that there exists an ideal λ, we can still find values for λ that will yield a distribution that is *as close as possible* to a uniform distribution over the semantic space, with the difference from uniformity measured by KL divergence. For this, let us define a Minimally-Unfair Noise Distribution.

Definition 2 (Minimally-Unfair Noise Distribution)**.** *For a generative model $G_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, a distribution $\mathbb{Q}^{\lambda} = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \mathbb{P}_{z|C_\phi=i}$, $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{Y}|}\}$, $\lambda_i \in \mathbb{R}_{\geq 0}$, $\sum_{i=1}^{|\mathcal{Y}|} \lambda_i = 1$, is a **Minimally-Unfair Noise Distribution** over $\mathcal{Z}$ with respect to $\mathcal{Y}$ if for $\mathbf{z} \sim \mathbb{Q}^{\lambda}$, $\lambda = \underset{\lambda}{\operatorname{argmin}}\, KL\{C'(G_\theta(\mathbf{z}))||Unif(\mathcal{Y})\}$.*

Intuitively, $\mathbb{Q}^{\lambda}$ is a Minimally-Unfair Noise Distribution if the values of λ yield generated samples with minimal divergence from a semantically uniform distribution, under the constraint that $\mathbb{Q}^{\lambda}$ is a convex combination of the $\mathbb{P}_{z|C_\phi=i}$'s. This constraint is required so that we can sample from $\mathbb{Q}^\lambda$ using a weighted sampling technique based off of our imperfect classifier C_ϕ.

Fortunately, we can easily find the values of λ that will yield a Minimally-Unfair Noise Distribution. Before deriving the procedure for this, let us define the distribution $\mathbb{P}_E^\lambda$ as the normalized weighted sum of the columns of E, where each column i is weighted according to a corresponding λ_i:

Definition 3 ($\mathbb{P}_E^\lambda$)**.** *Define $\mathbb{P}_E^\lambda = \sum_{i=1}^{|\mathcal{Y}|} \lambda_i E_{:,i}$ as a distribution over $\mathcal{Y}$ determined by prediction-conditional error matrix E and $\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_{|\mathcal{Y}|}\}$, $\lambda_i \in \mathbb{R}_{\geq 0}$, $\sum_{i=1}^{|\mathcal{Y}|} \lambda_i = 1$.*

Next, we note that finding the values for λ that maximize the entropy of $\mathbb{P}_E^\lambda$ is equivalent to finding the λ for which the divergence between $\mathbb{P}_E^\lambda$ and $Unif(\mathcal{Y})$ is minimized. This is stated formally in the following Proposition.

Proposition 1. *If $\lambda^* = \underset{\lambda}{\operatorname{argmax}}\, H(\mathbb{P}_E^\lambda)$ where H is entropy, then for the same λ^* it is true that $\lambda^* = \underset{\lambda}{\operatorname{argmin}}\, KL\{\mathbb{P}_E^\lambda || Unif(\mathcal{Y})\}$.*

Before stating the theorem that directly implies a strategy for learning λ for which $\mathbb{Q}^\lambda$ is Minimally-Unfair, all that is left to do is to link $\mathbb{P}_E^\lambda$ with the distribution of $C'(G_\theta(\mathbf{z}))$:

Proposition 2. *If $\mathbf{z} \sim \mathbb{Q}^\lambda$, then it is true that $C'(G_\theta(\mathbf{z})) \sim \mathbb{P}_E^\lambda$.*

Proposition 2 states that for a given λ the distribution of the perfect classifier C' given samples of the generator conditioned on draws from $\mathbb{Q}^\lambda$ will be distributed according to $\mathbb{P}_E^\lambda$. Now, we state Theorem 5 which directly implies our sampling strategy.

Theorem 5. *If $\lambda^* = \underset{\lambda}{\operatorname{argmax}}\, H(\mathbb{P}_E^\lambda)$, then $\mathbb{Q}^{\lambda *}$ is a Minimally-Unfair Noise Distribution. When $C_\phi = C'$ or $\mathbf{1}^{|\mathcal{Y}|} \in cone(E)$, then $\mathbb{Q}^{\lambda *}$ is also a Fair Noise Distribution.*

Theorem 5 follows from the preceding Propositions. To see how Theorem 5 implies a strategy for sampling from a Minimally-Unfair Noise Distribution, recall that $\mathbb{Q}^\lambda$ is defined as the mixture of distributions of each $\mathbb{P}_{z|C_\phi=i}$ for i from $1 \rightarrow |\mathcal{Y}|$ with λ_i as the ith mixture weight, where $\mathbb{P}_{z|C_\phi=i}$ is the distribution of $\mathbf{z}$'s that yield generated samples that

Algorithm 2 Collecting a dataset of samples from $\mathbb{Q}^\lambda$

```
1: procedure WEIGHTED_Q_DATASET($\{\mathcal{D}_{z|C_\phi=i}\}_{i=1}^{|\mathcal{Y}|}, S$)
2:     Input: $\mathcal{D}_{z|C_\phi=i}$ for $i$ from $1 \rightarrow |\mathcal{Y}|$: Dataset of noise samples that produce generated instances which $C_\phi$ classifies as group $i$; $S$: number of instances we wish to sample from $\mathbb{Q}^\lambda$
3:     Output: Dataset with $S$ samples from $\mathbb{Q}^\lambda$
4:     $\lambda \leftarrow argmaxH(\mathbb{P}_E^\lambda)$
5:     $Q\_dataset = []$                    ▷ Initialize the $\mathbb{Q}^\lambda$ as an empty array
6:     for $m \leftarrow 1$ to $S$ do
7:         $r \leftarrow random\_int(\lambda)$    ▷ get random int (1 to $|\mathcal{Y}|$) with probability proportional to $\lambda$
8:         $z \sim \mathcal{D}_{z|C_\phi=r}$
9:         $Q\_dataset.append(z)$
10:    end for
11:    return $Q\_dataset$                  ▷ Return dataset with $S$ samples from $\mathbb{Q}^\lambda$
12: end procedure
```

the noisy classifier predicts as group i. The above theorem states that the mixture of these distributions with mixture weights λ that produce maximum entropy for $\mathbb{P}_E^\lambda$ will yield a mixture distribution $\mathbb{Q}^\lambda$ that is worse-case Minimally-Unfair and, when possible, will be a perfect Fair Noise Distribution. Thus, to sample from $\mathbb{Q}^\lambda$ all that is needed is to 1) construct datasets $\mathcal{D}_{z|C_\phi=i}$ for i from $1 \rightarrow |\mathcal{Y}|$ such that if $\mathbf{z} \in \mathcal{D}_{z|C_\phi=i}$ then $C_\phi(G_\theta(\mathbf{z})) = i$ (i.e., these datasets serve as proxies for $\mathbb{P}_{z|C_\phi=i}$); 2) find λ such that $H(\mathbb{P}_E^\lambda)$ is maximized; and 3) perform a weighted sampling from each $\mathcal{D}_{z|C_\phi=i}$ proportional to its corresponding λ_i. Note that the datasets $\mathcal{D}_{z|C_\phi=i}$ can be formed by an approach similar to Algorithm 1. The algorithm returns a dictionary with keys being the elements of $\mathcal{Y}$, and the values of each key returned from that procedure form the corresponding required datasets (i.e., $\mathcal{D}_{z|C_\phi=i}$). A pseudo-code implementation of our approach for constructing a dataset of noise samples from a Minimally-Unfair Noise Distribution $\mathbb{Q}^\lambda$ is given in Algorithm 2.

4.5.2 Training the Distribution Mapper

Let $\mathcal{D}_{\mathbb{Q}^\lambda}$ be a dataset of noise samples from the space $\mathcal{Z}$ distributed according to $\mathbb{Q}^\lambda$, such that $\mathcal{D}_{\mathbb{Q}^\lambda}$ is obtained as described in the previous subsection (i.e., $\mathcal{D}_{\mathbb{Q}^\lambda}$ comes from Algorithm 1 in the case where the error rates E are unknown or C_ϕ can be assumed to be an ideal classifier, or from Algorithm 2 otherwise).

Now, we train a *distribution mapper* $M_\omega : \mathcal{Z} \rightarrow \mathcal{Z}$ such that if $\mathbf{v} \sim \mathbb{P}_{noise}$ then $M_\omega(\mathbf{v}) \sim \mathbb{Q}^\lambda$, where $\mathbb{P}_{noise}$ is an easy-to-sample-from noise distribution such as a multivariate Gaussian, or whatever noise distribution was used as input when training G_θ. As previously stated, the functional form of M_ω and the procedure used to train it is flexible. There is a wide class of generative models and training strategies that could be employed [188]. We chose to model M_ω as a GAN generator and find the parameters of M_ω using adversarial training. Thus, we train M_ω as such:

$$\max_\rho \min_\omega L(M_\omega, F_\rho) = \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_{\mathbb{Q}^\lambda}}\Big[\log\big(F_\rho(\mathbf{z})\big)\Big] - \mathbb{E}_{\mathbf{v}\sim\mathbb{P}_{noise}}\Big[\log\big(1 - F_\rho(M_\omega(\mathbf{v}))\big)\Big],$$

where F_ρ is a discriminator network.

Note that $\mathbf{z}$ is a multivariate vector, where the ordering of the elements of the vector are arbitrary (i.e., there is likely no spacial relationship in $\mathbf{z}$). Thus, the inductive bias of convolutional filters used in many leading GANs [189, 163, 190] is not appropriate in this case. Rather, we suggest to train the distribution mapper using a GAN architecture designed for tabular data; notably, CTGAN which has shown to perform well on data where a sequential or spacial inductive bias is not appropriate [191].

After training M_ω we can sample an arbitrary amount of samples from $\mathbb{Q}$ without applying Algorithm 1 or Algorithm 2, as $M_\omega(\mathbf{v}) \sim \mathbb{Q}^\lambda$ for $\mathbf{v} \sim \mathbb{P}_{noise}$. Importantly, $C'(G_\theta(M_\omega(\mathbf{v}))) \sim Unif(\mathcal{Y})$ when $\mathbb{Q}^\lambda$ is a Fair Noise Distribution, and will otherwise have minimum divergence from $Unif(\mathcal{Y})$ under the constraints given in Definition 2. Notably, we achieve this with *no samples of real data, an imperfect classifier, and without fine-tuning* G_θ. Instead, the only training needed is for M_ω, which in general should require much less complexity (and thus much less cost) than G_θ. Figure 13 shows the Distribution Mapper paired with the pretrained generator G_θ.

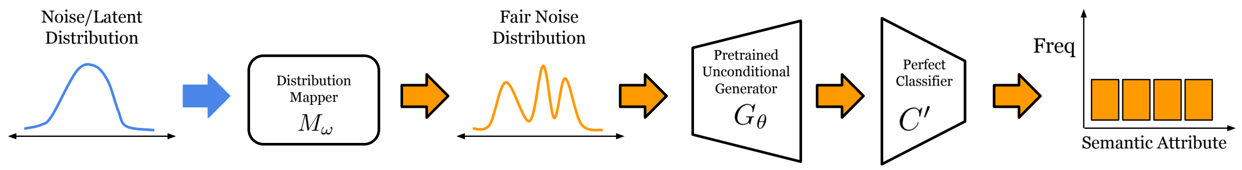

Noise/Latent Distribution
Distribution Mapper M_ω
Fair Noise Distribution
Pretrained Unconditional Generator G_θ
Perfect Classifier C'
Freq
Semantic Attribute

Figure 13: Distribution Mapper M_ω turns a standard noise distribution into a Fair Noise Distribution.

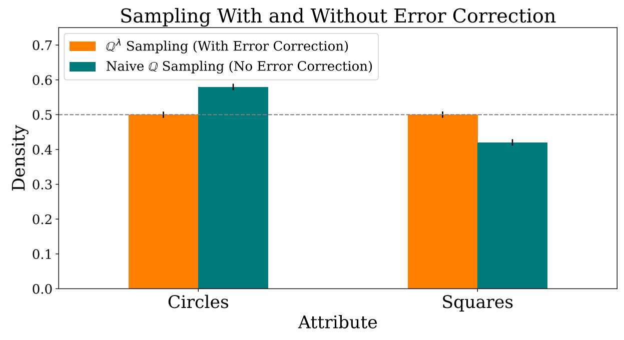
4.6 Experiments

We now experimentally evaluate our approach. Details such as hyperparameter choice and architectures are available in the appendix. Code for our method is available in the supplemental material.

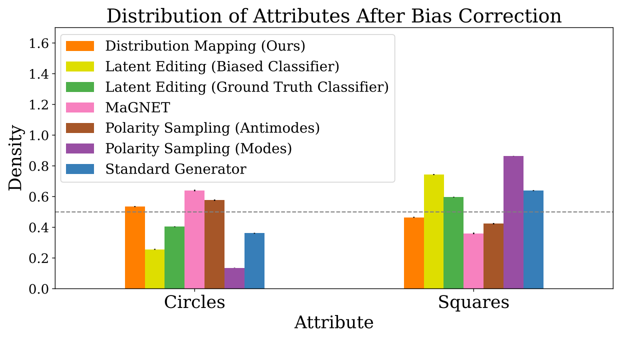
4.6.1 Compared Methods

Latent Editing (2019). We apply the commonly used latent editing [174, 175, 176, 177, 178, 179] approach to the task of constructing a dataset with an equal number of instances from each group in $\mathcal{Y}$. While some work uses non-linear directions when editing [192], using linear directions has been shown to work as well in practice [174, 179]. Thus, for each group $y \in \mathcal{Y}$, we fit a linear classifier K_y on noise samples from $\mathbb{P}_{noise}$, with the goal of separating noise instances that are mapped to group y from those mapped to all other groups. After training these classifiers, we can sample a noise instance belonging to a group with the following procedure: First, sample $z \sim \mathbb{P}_{noise}$; if $K_y(z) = y$ then return z, otherwise perform latent editing on z until $K_y(z_{new}) = y$. The editing approach is given by $z_{new} = z_{previous} + \epsilon \cdot \eta_y$, where η_y is the normal vector to K_y's decision boundary and ϵ is a step size. We set $\epsilon = 0.1$ in our experiments.

MaGNET (2022). This method was recently proposed to uniformly sample the manifold of pretrained generative models [180], using a sampling strategy that leverages the Jacobian of the generative model. The authors argue that while this does not guarantee an equal distribution for each group, *MaGNET* should increase the frequency of dis-



(a) Effect of bias correction; 0.5 (dashed line) is ideal.



(b) Comparison of methods; 0.5 (dashed line) is ideal.

Figure 14: a) Effects of applying our bias correction when constructing Fair Noise dataset; b) comparison of methods on the Shapes dataset.

advantaged groups by sampling more often from low-density regions of the generative manifold.

Polarity Sampling (2022). An extension of *MaGNET*, *Polarity Sampling* allows for more controlled sampling over the generative manifold [181]. Sampling is controlled by a parameter ρ where as ρ goes to $-\infty$ modes are sampled from increasingly often, antimodes are sampled from more as $\rho \to \infty$, and the original generative distribution is sampled from for $\rho = 0$. We thus compare against two *Polarity Sampling* settings; *Polarity Mode Sampling* where $\rho = -2$, and *Polarity Antimode Sampling* where $\rho = 2$. By a similar argument used for *MaGNET*, sampling from antimodes may result in minority groups being more represented.

Standard Generator. We compare against unmodified pretrained generators used as initially intended. This is a baseline which other methods should outperform.

4.6.2 Uniformly Sampling From Shapes Dataset

We first evaluate our approach on a dataset of synthetic images. Each image is of either a circle or a square, where the shape has a random size, color, and position in the image. We use the object's shape as the semantic attribute; thus, $\mathcal{Y} = \{\text{`}\texttt{Circle}\text{'}, \text{`}\texttt{Square}\text{'}\}$. This dataset was first used by Jing et al. [193]. We utilize the VAE Jing et al. compared against in the same work as our generator.

Testing Classifier Bias Correction Approach. We demonstrate the utility of correcting for a biased classifier when constructing a dataset of Fair Noise samples. To that end, we compare the distribution of each group when we trust the biased classifier C_ϕ (Algorithm 1) to the distribution found when using our proposed biased correction approach (Algorithm 2). We use a linear classifier trained to distinguish images of squares from circles as our biased classifier C_ϕ. This classifier is implemented using Scikit-learn's LinearSVC classifier [194]. As a ground truth, we utilize a deep convolutional network classifier that achieves approximately perfect accuracy on the task of distinguishing squares from circles. Thus, this network acts as C'.

Figure 14 a) shows the distribution of shapes in the Fair Noise Dataset with error correction (Algorithm 2) and without applying our bias correction approach (Algorithm 1). As $|\mathcal{Y}| = 2$ for this experiment, the best case is when the density of each group is 0.5. Clearly, our bias correction approach yields a dataset that is much closer to being uniform over the semantic attribute. This implies the need to correct for inaccurate classifiers when constructing Fair Noise Datasets.

Comparative Study on Shapes Dataset. We next test our proposed Distribution Mapping approach (Section 4.5.2) against the compared methods. For our approach, we train M_ω as a CTGAN [191] trained on the corrected Fair Noise Distribution obtained in the previous experiment. As the compared *Latent Editing* method also requires a classifier as a ground truth for training each of its linear models K_y, we compare against two versions of the method: 1) *Latent Editing (Biased Classifier)* using the biased C_θ as a ground truth, and 2) *Latent Editing (Perfect Classifier)* which uses the ground-truth, effectively perfect convolutional network discussed above. Note that from our problem definition, this perfect classifier would usually be unavailable to us.

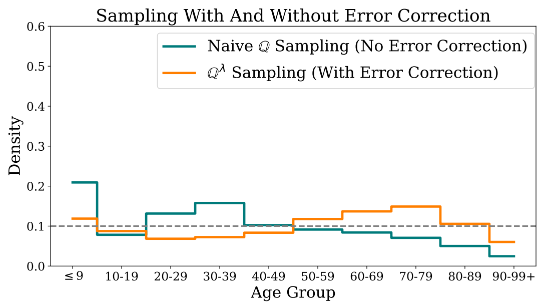
As Figure 14 b) shows, our approach clearly results in the most uniform distribution out of all compared methods. Additionally, the *Latent Editing* approach performs much worse when using the biased classifier. As expected, *Polarity Antimode Sampling* does indeed increase the frequency of the minority class (*Circles*). Interestingly, *MaGNET* also increases the frequency of the minority class, but is roughly as far from uniform as is the distribution of the original generator; *MaGNET* flips the distribution and under-represents what was previously the majority class.

4.6.3 Age Bias in Face Image Generator

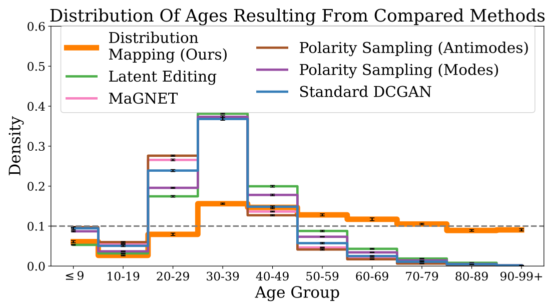
In this next experiment, we evaluate our proposed method's ability to debias a generative model that produces images of people's faces. Specifically, the generative model is a DCGAN [190] that we pretrain on a grayscale version of the UTKFace dataset [195]. For this experiment, the semantic attribute is `Age`. The age of each individual in UTKFace is given as a label. As our approach assumes the semantic attribute $\mathcal{Y}$ is discrete, we bin the ages in increments of 10 years such that $\mathcal{Y} = \{`\leq \texttt{9}', `\texttt{10-19}', \ldots, `\texttt{90-99+}'\}$. We train a deep convolutional network as the ground truth classifier (used for evaluation), and use a corresponding classifier with a quarter of the feature maps as the biased classifier.

Evaluating Bias Correction Approach With Large Number of Classes. We repeat the bias correction experiment we performed on the Shapes data again for the UTKFace Generator. While the previous experiment required only two classes, we now have ten classes (one for each age bin). Despite this increase, Figure 15 a) shows that performing the bias correction we propose in Algorithm 2 yields a much more uniform Fair Noise training dataset than results from trusting the biased classifier.

Comparative Study on Reducing Age Bias. We evaluate all methods on the UTKFace Generator, with the goal of making the output images uniform over `Age` (i.e., generate an equal number of images of people belonging to each age group). Figure 15 b) clearly shows that our approach (orange line) produces a much more uniform distribution over ages than the compared method, though it does generate images for the `10-19` bin too infrequently. While approaches such as *Polarity Antimode Sampling* and *MaGNET* are some-



(a) Effect of bias correction; 0.1 (dashed line) is ideal.



(b) Comparison of methods; 0.1 (dashed line) is ideal.

Figure 15: a) Our biased correction approach yields better Fair Noise Distributions; b) comparison of methods on correcting for age bias in the UTKFace DCGAN.

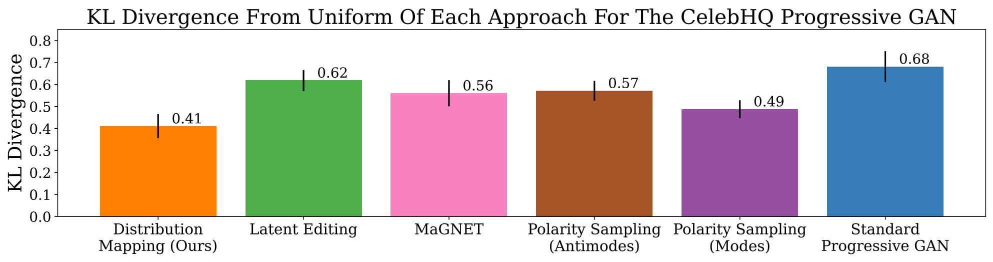


Figure 16: KL Divergence between the distribution over the semantic space for the output of each method (*lower is better*).

what less biased away from generating images of young people, they fail to generate many samples for older individuals. Note that for the *Latent Editing* approach the we fit a linear regressor (guided by the ground truth) on the latent space rather than a classifier, as age is more naturally a continuous attribute.

4.6.4 Uniformly Sampling Over Race in Progressive GAN

We apply our approach to debias a pretrained Progressive GAN [163]; specifically, the PyTorch [196] version of the `'celebAHQ-256'` model [8]. We use the `Race` of the individual in each image as the semantic attribute, and use the MTCNN classifier from the DeepFace [197] package to classify race. Matching the classes available in DeepFace, our semantic attribute space is $\mathcal{Y} = \{$`'Asian'`, `'Indian'`, `'White'`, `'Middle Eastern'`, `'Latino Hispanic'`, `'Black'`$\}$.

[8]see https://pytorch.org/hub/facebookresearch_pytorch-gan-zoo_pgan/

The Progressive GAN strongly favors generating white individuals, likely because its training images were of predominantly white celebrities. Here, we consider the classifier to be accurate.

Comparative Study of Debiasing Progressive GAN. For each method, we report the KL divergence between a uniform distribution and the classifier's output on the samples generated from the method in Figure 16. Note that unlike in the previous experiments, for this metric *lower is better*. Our approach has the lowest KL divergence, indicating it produces a more equal number of images of people from each race than any compared method. We note that despite this we still observe an over-representation of images of white individuals in the samples produced by our (and all other) approaches (see Appendix). This indicates that most regions of the latent space are likely associated with semantic attribute, and unless the distribution mapper very closely matches its Fair Noise training distribution, white persons will still be over-represented. Still, our approach results in a distribution that is most fair out of all compared methods. As we observed previously, sampling antimodes with *Polarity Sampling* produces next-best results; likely because it explicitly draws from low-probability regions of the latent space, which correspond to non-white individuals.

4.7 Chapter Summary

Impact. The goal of this work, reconditioning generative models to not reproduce the biased distribution of their training set but rather produce one that treats each group equitably, is driven by the desire to mitigate the effects that systemic biases have on generative models increasingly used in real world applications. This has the potential for beneficial societal impact by counteracting the harmful effects of such systemic biases that lead these models to be unfair to underrepresented groups. However, the potential negative impacts of advancing generative modeling should not go unconsidered. Such models have already been used for some harmful applications such as Deep Fakes [198].

It is important to stress that while our approach aims to mitigate the effects of bias in existing models, it is not a fix-all nor an excuse to train models on knowingly biased data. When possible, it is essential to collect fair and equitable training datasets, and to take measures to ensure that the models we train are fair *without* the need for post-hoc corrections.

Limitations. Currently, our approach assumes that the semantic space $\mathcal{Y}$ is discrete. We plan to extend this work to handle continuous attributes, such as skin tone, in future work. Additionally, while the conditional distribution $P(\mathbf{x}|\mathbf{y})$ should be roughly equal before and after applying our method if the classifier C_ϕ is accurate, in the case of a biased classifier the conditional distribution may change. Correcting for potential distribution shift is likewise future work.

Conclusion. This is the first method to correct a pretrained biased generative model (i.e., one that strongly favors generating images of white people over all other races) given only the generator and a potentially-biased classifier. We propose a sampling strategy to construct a fair training set using the biased classifier in a way that corrects for its bias, and a Distribution Mapping module that uses this training set to learn how to sample noise instances that produce fair outputs when used as input to the generative model. Notably, we are able to debias the generative model without retraining the model or utilizing any real data. Our results indicate that our approach produces outputs that are much fairer than existing methods. This work may inspire more research on Distribution Mapping techniques to recondition generative models by transforming their standard latent distributions into distributions that yield more favorable behavior.

5 Converting Unconditional Pretrained Generators into Conditional Models Without Retraining Or Supervision.

This work is under review at ICLR 2023:

Walter Gerych, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. "Latent Shattering: Turning Unconditional Pretrained Generators Into Conditional Models By Imposing Latent Structure". Under review.

5.1 Motivation

Motivation Generative models are increasingly being adopted in a wide range of domains due to their impressive abilities to synthesize realistic data for many modalities [188]. Many popular classes of generative models, such as Generative Adversarial Networks (GANs) [183] and Variational Auto-Encoders (VAEs) [184], utilize a *generator*[9] that maps from a *lower*-dimensional *latent space* $\mathbb{Z}$ to a *higher*-dimensional *data space* $\mathbb{X}$. Though some pretrained GAN and VAE models allow for conditional generation, many are *unconditional* [25]; the user does not have control over which *class* is sampled from. For instance, a user can use an unconditional generator trained on a dataset of pictures on animals to synthesize random images of animals, but can not specify that the image be a picture of a "dog". Due to the expense and difficulty associated with training these generative models [199] it would be beneficial to have the ability to convert pretrained unconditional models into conditional models *without* having to retrain the generator or any other deep network. Likewise, as in general the specific modes or classes a generator is capable of producing may be unknown, an *unsupervised* approach is also desirable. In this work we thus develop a finetuning-free, unsupervised method for converting pre-

[9]In auto-encoder based models, the generator corresponds to the decoder network

trained unconditional generators into conditional models.

The key idea of our work hinges on the fact that common generative models utilize *continuous* functions with a connected latent space, and are thus unable to produce perfectly distinct sub-manifolds in their output space [200]. Assuming the true data generation is concentrated on a set of sub-manifolds, the generator should place most mass on these sub-manifolds but *must* also place a small amount of mass *between* the manifolds [201]. We posit that by separating out the low-probability regions from high-probability regions, it will be possible to adaptively sample from the modes or classes the generator is capable of producing.

5.2 Related Work

Existing work into performing class-specific generation given pretrained unconditional models typically require 1) classifiers or similar *energy functions* that provide feedback on which class each point in the latent space $\mathbb{Z}$ corresponds to when passed through the generator [78, 79]; or 2) labeled data and training additional deep networks to guide the generator's output [25]. However, access to additional networks that can provide this class-specific feedback as well as access to training data and sufficient compute resources is not always a practical assumption.

Converting unconditional models to conditional models. Methods such as PromptGen [78] assume the availability of energy functions - such as pretrained classifiers - that can be utilized to learn a distribution of a certain class or attribute in the latent space. Latent Constraints likewise requires an auxilary classifier or user-provided reword function, and also requires training an additional GAN-like model on the latent space [79]. This approach requires training additional invertable neural networks to sample from each of these learned distributions. HyperGAN [25] converts unconditional generative models into conditional models by training a Hyper Net on an auxiliary labeled training dataset containing instances of each class of interest.

Disjoint manifold learning. It is well known that typical generative models that are trained to map a connected $\mathbb{Z}$ to a disjoint data manifold are incapable of producing a disjoint distribution in their output space [201, 202, 200]. Approaches such as Partition-Guided GANs [203], Disconnected Manifold Learning [201], and MG-GAN [204] address this issue by training multiple generators. Another standard approach is to utilize an unconnected latent space when training the generative model [205, 206, 207]. However, these approaches are applicable when training *new* generative models and do not address the problem of sampling from different modes given an *existing* generator. On the other hand, truncation approaches [208, 200, 209] *reject* certain sampled latent codes, which can allow a pretrained generative to sample from a disconnected latent space. Similarly, Latent Reweighting [210] and Polarity Sampling [181] aim to reweight the latent distribution to sample latent codes that correspond to modes in the output space. While these approaches can be applied to pretrained generators, they do not typically allow for control over *which* mode is being sampled from at a given time.

5.3 Problem Definition

Let $G : \mathbb{Z} \to \mathbb{R}^N$, $\mathbb{Z} \subset \mathbb{R}^M$ with $M < N$, be a pretrained generative model such that G is a smooth, continuous function. Let $\mathbb{Z}$ be a simply connected manifold. Let $G(\mathbb{Z})$ be a Riemannian manifold with intrinsic dimensionality M, and let q_G be the "generated distribution" defined as a pushforward distribution $q_G = G_{\#}p_{\mathbf{z}}$, where $p_{\mathbf{z}}$ is the prior "noise" distribution over $\mathbb{Z}$. $p_{\mathbf{z}}$ is generally a Gaussian or uniform distribution [188], but we do not require this to be the case.

Further, assume that G was trained to minimize a divergence between $q_G = G_{\#}p_{\mathbf{x}}$ and $p_{\mathbf{x}}$, where $p_{\mathbf{x}}$ is the target distribution of real data. Let $\mathbb{X} \subset \mathbb{R}^N$ be a subset of $\mathbb{R}^N$ where the real data lies. Let $\mathbb{X}$ consist of k distinct, non-overlapping sub-manifolds: $\bigsqcup_{i=1}^{k} \mathbb{M}_i = \mathbb{X}$, and $\mathbb{M}_i \cap \mathbb{M}_j = \emptyset \, \forall i \neq j$.

Our goal is to find a latent space $\mathbb{Z}^* \subset \mathbb{Z}$ and a corresponding distribution $q_{\mathbf{v}}$ with the following properties: 1) $q_{\mathbf{v}}$ has minimal divergence from $q_{\mathbf{z}}$; 2) the latent space $\mathbb{Z}^* = \bigsqcup_{i=1}^{k} \mathbb{V}_i$, $\mathbb{V}_i \cap \mathbb{V}_j = \emptyset \, \forall i \neq j$ consists of k distinct non-overlapping manifolds $\mathbb{V}_i$; 3) if latent code $v_i \in \mathbb{V}_i$ then $G(v_i) \in \mathbb{M}_i$.

To develop our method, we make use of a small set of assumptions that we outline in detail below:

Assumption 1. *G is a homeomorphism from $\mathbb{Z}$ to $\mathbb{R}^N$. Thus, the pullback distribution $(G^* q_G)(G(\boldsymbol{z}))$ is well-defined for all $\boldsymbol{z} \in \mathbb{Z}$ and $G(\boldsymbol{z})$ is unique for all $\boldsymbol{z} \in \mathbb{Z}$, and $\{G(\boldsymbol{z}) \mid \boldsymbol{z} \in \mathbb{Z}\}$*

Assumption 1 requires G to be a one-to-one mapping from the latent space to the data space, such that each generated instance has a unique latent code. This assumption is commonly used by other works [181]. Like them, we do not practically require that this exactly holds for every instance and we do not need to explicitly define a left-inverse; injectivity is required for theoretical analysis, and in practice our approach will be effective so long as it holds for almost all of the input space.

Assumption 2. *There exists a positive real valued number ρ such that $p_G(G(\boldsymbol{z})) > \rho$ if $G(\boldsymbol{z}) \in supp(p_{\mathbf{x}})$, and $p_G(G(\boldsymbol{z})) \leq \rho$ if $G(\boldsymbol{z}) \notin supp(p_{\mathbf{x}})$ otherwise.*

In essence, Assumption 2 states that the density of the generated distribution is higher when on-manifold with regards to the real data $\mathbb{X}$, and lower when off-manifold. For example, realistic synthetic data points have higher density than unrealistic synthetic points.

Assumption 3. *For each submanifold in the data space, $\mathbb{M}_i \in \mathbb{X}$, $\exists \, \boldsymbol{z} \in \mathbb{Z}$ such that $G(\boldsymbol{z}) \in \mathbb{M}_i$.*

Assumption 3 states that the generator places at least some mass on each mode or submanifold of $\mathbb{X}$. For example, if the real data consists of 10 distinct classes, the generator G would be capable of producing some instances of each class.

5.4 Challenges

Unsupervised and finetuning-free class-conditional generation is a difficult task:

- *Unknown Classes and Modes.* In general, it may be unknown what classes a generator is even capable of producing. While this should be a subset of the classes of the data it was trained on, the training classes may be unknown. For instance, the model could have been trained on an unlabeled data or the training data could be proprietary.

- *Uninformative Latent Space.* A natural proxy for class labels is to identify the different modes or sub-manifolds in the data space. This is motivated by recent work which has provided evidence that high-dimensional data such as images often lay on a union of disjoint sub-manifolds [84], where these sub-manifolds correspond to semantically meaningful classes. For example, in MNIST each digit lies on a different sub-manifold in the image space [84]. Thus, a potential solution could involve identifying which regions of the latent space $\mathbb{Z}$ correspond to each of these sub-manifolds, and then adaptively sample from these regions. However, this is challenging as the distribution over the latent space of leading generative models is typically enforced as a prior and is not data-dependent [183, 184]; e.g. the distribution is Gaussian or uniform, and there is no reason for a fluctuation in density or other variation that aligns with any of the sub-manifolds in the data space.

5.5 Proposed Approach

In this section, we describe our approach to transform the "unstructured" single-manifold latent space into k distinct sub-manifolds, where each of the k submanifolds contains latent codes that are mapped to only one single submanifold in the target data space $\mathbb{X}$. Section 5.5.1 describes this approach of finding these k submanifolds in $\mathbb{Z}$. Thereafter, we describe our proposed approach for selectively sampling from each of these submanifolds

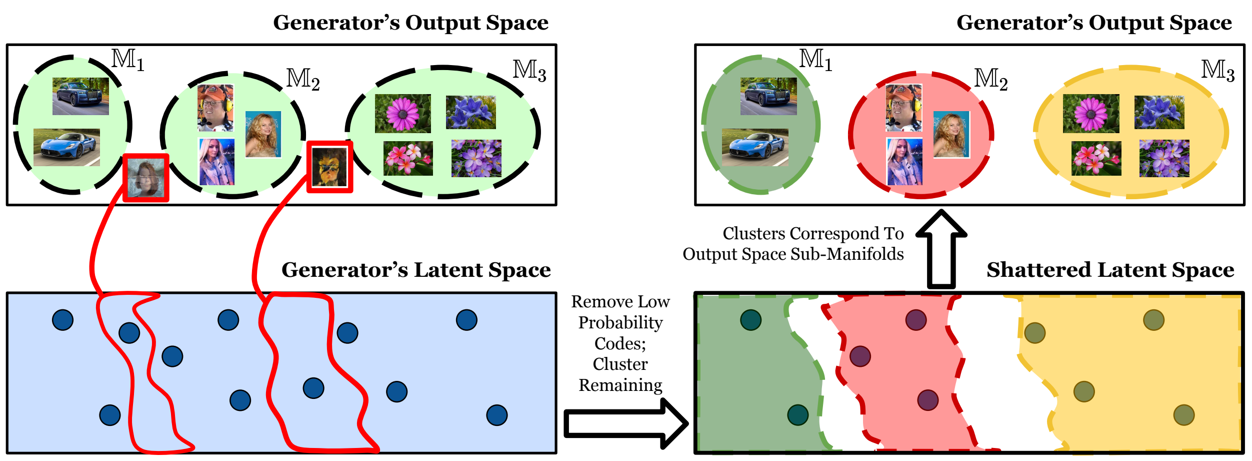


Figure 17: *Latent Shattering*. We remove latent codes that correspond to low-probability (*off-manifold*) samples in the output space, and then cluster the higher-probability latent codes. This results in a disjoint, structured latent space.

in Section 5.5.2.

5.5.1 Latent Shattering: Dividing Latent Space Into Meaningful Submanifolds

Note that as G is a continuous function and the latent space $\mathbb{Z}$ is simply connected, then $\mathbb{G} = G(\mathbb{Z})$ will likewise be simple connected [200]. Thus, the generated data will by default lie on a single generated manifold in $\mathbb{R}^N$. However, as G was trained to match the distribution of real data (which lies on k submanifolds), q_G should place more mass on the submanifolds in $\mathbb{X}$ and lower mass between submanifolds (see Assumption 2). This leads us to define the property of a generated manifold being k-ρ-*disconnected*:

Definition 4 (k-ρ-disconnected)**.** *For a generative model $G : \mathbb{Z} \to \mathbb{R}^n$, let $\mathbb{G}$ be the support of the generated distribution q_G. Let $\mathbb{G}_{\setminus\rho} \subset \mathbb{G}$ be a subset of the support such that $\mathbb{G}_{\setminus\rho} = \{\boldsymbol{x} \in \mathbb{G} | q_G(\boldsymbol{x}) \geq \rho\}$. Then, we say that G is k-ρ-**disconnected** if $\mathbb{G}_{\setminus\rho}$ consists of k disjoint submanifolds; i.e., $\mathbb{G}_{\setminus\rho} = \bigsqcup_{i=1}^{k} \mathbb{S}_i$ and $\mathbb{S}_i \cap \mathbb{S}_j = \emptyset \; \forall \; i \neq j$.*

Definition 4 means that we say the manifold of generated data is k-ρ-disconnected if removing points where q_G has a density less than ρ results in a set $\mathbb{G}_{\setminus\rho}$ that consists of k disjoint submanifolds.

Next, we define the density of the generated data after removing the regions with density $< \rho$ as the ρ-shattered density $q_{G\setminus\rho}$.

Definition 5 (ρ-shattered density $q_{G\setminus\rho}$)**.** *For a generative model $G : \mathbb{Z} \to \mathbb{R}^n$, let $\mathbb{G}_{\setminus\rho} = \{\boldsymbol{x} \in \mathbb{G} | q_G(\boldsymbol{x}) \geq \rho\}$. The density of the ρ-**shattered density** $q_{G\setminus\rho}$ is the density function defined over $\mathbb{G}_{\setminus\rho}$ such that $q_{G\setminus\rho}(\boldsymbol{x}) = \big(1 - \int_{\mathbb{G}-\mathbb{G}_{\setminus\rho}} q_G(\boldsymbol{x}')d\boldsymbol{x}'\big)^{-1} q_G(\boldsymbol{x})$.*

Definition 5 simply defines a distribution over the disjoint generated space $\mathbb{G}_{\setminus\rho}$ as equivalent to the original generated density q_G with only a few slight modifications: 1) the density is set to 0 where $q_G < \rho$; and 2) the remaining non-zero density is scaled by a constant factor to account for the mass that was “removed”.

We can begin to see where these two definitions are useful when we consider Lemma 4.

Lemma 4. *If $\mathbb{X}$ consist of k disjoint submanifolds $\mathbb{M}_1$ to $\mathbb{M}_k$, then $\exists\rho$ such that G is k-ρ-disconnected.*

The above lemma follows from Assumptions 1 - 3. This means that if G was trained to match a real-data distribution over k submanifolds, then there should exist a density value for the generated distribution that distinguishes on-manifold instances from off-manifold instances. Put differently, there is a value ρ such that the generated density is greater than ρ for all generated data that lies on any of the real-data manifolds and is less than ρ otherwise.

In effect, the preceding statements have lead up to the insight that we can remove low-probability generated instances (where low-probability is defined in reference to a scalar value ρ) such that the remaining generated data will lie on k distinct submanifolds. However, this does not tell us anything about the struture of our latent space - which was our initial goal. To that end, we now introduce the following important theorem underlying our proposed method.

Theorem 6. *Let G be k-ρ-disconnected by Definition 4. Then the support of the pullback distribution $(G^* q_{G\setminus\rho})(G(\boldsymbol{z}))$, $\mathbb{Z}_{\setminus\rho} = \{\boldsymbol{z} \in \mathbb{Z} \mid q_{G\setminus\rho}(G(\boldsymbol{z})) > 0\}$ consists of k disjoint submanifolds.*

Theorem 6 states that if removing generated instances with a density less than ρ not only makes the generated distribution lie on *k disjoint submanifolds in the data space*, but also results in a *latent space with k distinct submanifolds.*

The following lemma and theorem now establish the *relationship* between the submanifolds in $\mathbb{Z}$ to the submanifolds in the data space.

Lemma 5. *For each submanifold $\mathbb{S}_i \in \mathbb{G}_{\backslash\rho}$, $\exists\ \mathbb{V}_j \in \mathbb{Z}_{\backslash\rho}$ such that $G(\boldsymbol{z}) \in \mathbb{S}_i\ \forall\ \boldsymbol{z} \in \mathbb{V}_j$. Likewise, for each submanifold $\mathbb{V}_j \in \mathbb{Z}_{\backslash\rho}\ \exists\ \mathbb{S} \in \mathbb{G}_{\backslash\rho}$ such that $G(\boldsymbol{z}) \in \mathbb{S}_i\ \forall\ \boldsymbol{z} \in \mathbb{V}_j$.*

Theorem 7. *Let ρ be a scalar such that $q_G(G(\boldsymbol{z})) > \rho\ \forall G(\boldsymbol{z}) \in \mathbb{X}$ and $q_G(G(\boldsymbol{z})) \leq \rho\ \forall G(\boldsymbol{z}) \notin \mathbb{X}$. Then, for each submanifold $\mathbb{M}_i \in \mathbb{X}\ \exists \mathbb{V}_j \in \mathbb{Z}_{\backslash\rho}$ such that every point in $\mathbb{V}_j$ is mapped to a point in $\mathbb{M}_i$. Additionally, for every submanifold $\mathbb{V}_j \in \mathbb{Z}_{\backslash\rho}$, every point in $\mathbb{V}_j$ maps to the same submanifold $\mathbb{M}_i \in \mathbb{X}$.*

Theorem 7 follows directly from the preceding Theorem and Lemmas. In a nutshell, the above analysis shows that by removing points from the latent space that map to low-density points in the data space, *we can split the latent space of our pretrained generator G into k disjoint submanifolds (or clusters)*, where *each submanifold in the latent space corresponds to one of the submanifolds in the data space*. In other words, we now have achieved our important goal of introducing structure onto the prior latent space by removing data from certain regions of $\mathbb{Z}$.

The above findings refer to the concept of removing latent codes that are mapped to low-probability regions in the data space. Of course, this requires us to be able to identify which codes are mapped to low-probability regions. The following equation for the density of $G(\mathbf{z})$ allows us to do just that:

$$q_G(\boldsymbol{z}) = \frac{p_{\mathbf{z}}(\boldsymbol{z})}{\sqrt{det(\boldsymbol{J}_G^\top(\boldsymbol{z})\boldsymbol{J}_G(\boldsymbol{z}))}}, \tag{32}$$

where $\boldsymbol{J}_G(\boldsymbol{z}) = \frac{\partial G}{\partial \boldsymbol{z}}$ is the Jacobian of G evaluated at $\boldsymbol{z}$. Equation 32 follows from the pushforward distribution between Riemannian manifolds and was is similarly utilized by recent works [180]. Here, we utilize the expression in Equation 32 to identify points with a generated density less than ρ.

It is known that for high-dimensional data, a Gaussian distribution is well approximated by a uniform distribution over a sphere [211]. Thus, for $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, the generated density is approximately proportional to the following for the majority of realizations of $\mathbf{z}$:

$$q_G(\mathbf{z}) \underset{\sim}{\propto} \frac{1}{\sqrt{det(\boldsymbol{J}_G^\top(\mathbf{z})\boldsymbol{J}_G(\mathbf{z}))}}. \tag{33}$$

Equation 33 also holds for the case where $\mathbf{z}$ is from a uniform distribution.

Our above results imply a strategy for converting the original latent space $\mathbb{Z}$ of the pretrained generator G into a structured latent space $\mathbb{Z}^*$. In short, first, we collect a large sample of D points $\{\boldsymbol{z}_i\}_{i=1:D}$ from $p_{\mathbf{z}}$. Second, we compute the density of each point according to Equation 32. Lastly, we remove each $\boldsymbol{z}_i$ where $q_G(\boldsymbol{z}_i) < \rho$, where ρ is a hyperparameter that we chose.

A pseudocode algorithm for this process is given in Algorithm 3.

Algorithm 3 Obtaining dataset from structured latent space $\mathbb{Z}^*$

```
Require: Pretrained generator $G$, sample size $D$, hyperparameter $\rho$
Ensure: Dataset of samples $\mathcal{Z}^*$ from structured latent space $\mathbb{Z}^*$
1: Initialize empty set $\mathcal{Z}^*$
2: Sample $\{\boldsymbol{z}_i\}_{i=1:D}$ from $p_{\mathbf{z}}$
3: for $i = 1$ to $D$ do
4:     Compute density $q_G(\boldsymbol{z}_i)$ using Equation 33
5:     if $q_G(\boldsymbol{z}_i) \geq \rho$ then
6:         Add $\boldsymbol{z}_i$ to $\mathcal{Z}^*$
7:     end if
8: end for
9: return $\mathcal{Z}^*$
```

5.5.2 Sampling From The Shattered Latent Space

We now need a strategy of performing online sampling from $\mathbb{Z}^*$ or from some specific submanifold $\mathbb{V}_i \subset \mathbb{Z}^*$, so as to perform controllable generation.

To that end, we first propose to cluster the new latent space $\mathbb{Z}^*$ so that we can identify the k submanifolds it consists of. This can be achieved using one of the existing clustering algorithms. In our experiments we utilize k-means sampling due to its simplicity, and find it is sufficient to achieve our goal. Though more sophisticated approaches could equally be deployed, and they may yield even more robust clustering performance.

After clustering, the fitted clusterer C maps each point $\boldsymbol{z} \in \mathbb{Z}^*$ to a cluster label $\ell \in \{1, 2, \ldots, k\}$. Then, we propose to fit a *mapper* function $K : \{1, 2, \ldots, k\} \times \mathbb{J} \to \mathbb{Z}^*$ that allows us to sample from any of the k desired submanifolds in $\mathbb{Z}^*$. Specifically, the mapper K takes in the index i of the cluster we want to sample from along with a noise code $\boldsymbol{j}$ and maps to a point $\boldsymbol{z} \in \mathbb{Z}^*$ such that $\boldsymbol{z} \in \mathbb{V}_i$.

As we do not want to modify the distribution of latent codes for each cluster, we parameterize K to encourage $K_i {}_{\#} p_{\mathbf{j}} = p_{\mathbb{Z}^*_{C=i}}$ for each cluster i. Specifically, we parameterize $K(i, \cdot)$ as a collection of Gaussian mixture models (GMM) fit to samples from $\mathbb{V}_i$. We utilizes GMMs as they are very computationally inexpensive. At the end, we can selectively sample from the ith sub-manifold by sampling from the GMM that was fit to the ith cluster in the latent space.

5.6 Experiments

We now experimentally evaluate our approach. Detailed settings of our experiments, such as hyperparameter choice and architectures, are available in the appendix.

5.6.1 Compared Methods

Original Generator. We fit k-means clustering on samples from the generator′ original latent distribution $p_{\mathbf{z}}$. This acts as a baseline; the clusters found here should not meaning-fully correspond to cluster or classes in the output space.

Polarity Sampling. This approach allows for sampling from modes or minima of the generated distribution through reweighting the original latent distribution [181]. Polarity Sampling is determined using a parameter ρ where as ρ goes to $-\infty$ modes are sampled from increasingly often, and antimodes are sampled from more as $\rho \to \infty$, with the original generative distribution corresponding to $\rho = 0$. As we aim to sample from the on-manifold region of the output space, we set ρ to a negative value such as to sample from modes rather than low-probability regions. Specifically we set $\rho = -0.1$.

JBT Sampling. This approach, proposed in [200], reweights the latent distribution according to the Frobenius norm of the generator's jacobian. Whereas our method reweights the latent distribution according to the *volume* change of the generator's transformation, JBT's approach corresponds to rescaling proportional to the length of the diagonal of the *diagonal* of the unit box after mapping through the generator. The diagonal provides less meaningful feedback on the change in density induced by the generator than our approach, which directly models the volume change.

For each method, we evaluate the quality of the clusters found through applying k-means on large samples from their induced latent distributions. We also fit a GMM on each cluster for each method and report metrics for sampling from clusters in this manner, reported as "{*method*} + *GMM*".

5.6.2 Datasets.

We compare all methods on three datasets: `MNIST`, `FashionMNIST`, and `Faces + Flowers`. The `Faces + Flowers` dataset is constructed by mixing together the `CelebA` and `iNaturalist Flowers` datasets. We do this as the manifold of images of human faces and the manifold of images of flowers should be disjoint.

For each dataset, we pretrain a DCGAN [190] to match the data distribution. We use a latent dimension of 64 for each dataset. We likewise pretrain a convolutional classifier for

	Metric	Original Generator	Polarity Sampling	JBT Sampling	LASH (Ours)
MNIST	Homogeneity	0.0374 ± 0.0044	0.1657 ± 0.0069	0.3786 ± 0.0147	**0.4105 ± 0.0118**
	Completeness	0.0375 ± 0.0044	0.1775 ± 0.0077	0.3793 ± 0.0147	**0.4120 ± 0.0119**
	V Measure	0.0374 ± 0.0044	0.1714 ± 0.0073	0.3789 ± 0.0147	**0.4113 ± 0.0118**
	Adjusted MI	0.0198 ± 0.0045	0.1555 ± 0.0074	0.3675 ± 0.0150	**0.4004 ± 0.0121**
Fashion MNIST	Homogeneity	0.0336 ± 0.0051	0.1723 ± 0.0128	0.1965 ± 0.0090	**0.2297 ± 0.0091**
	Completeness	0.0337 ± 0.0051	0.1761 ± 0.0127	0.1974 ± 0.0091	**0.2312 ± 0.0092**
	V Measure	0.0336 ± 0.0051	0.1742 ± 0.0128	0.1970 ± 0.0091	**0.2305 ± 0.0091**
	Adjusted MI	0.0159 ± 0.0052	0.1588 ± 0.0130	0.1822 ± 0.0092	**0.2163 ± 0.0093**
Face +Flowers	Homogeneity	0.0020 ± 0.0016	**0.1290 ± 0.0065**	0.1027 ± 0.0074	0.1275 ± 0.0121
	Completeness	0.0032 ± 0.0025	0.1804 ± 0.0087	0.1716 ± 0.0113	**0.2211 ± 0.0170**
	V Measure	0.0025 ± 0.0019	0.1504 ± 0.0074	0.1285 ± 0.0089	**0.1617 ± 0.0140**
	Adjusted MI	0.0020 ± 0.0019	0.1501 ± 0.0074	0.1281 ± 0.0089	**0.1613 ± 0.0140**

Table 8: Comparative study of the cluster quality found from each method. Higher numbers are better for each metric. Best performance is bolded.

each dataset, to predict the classes in each. This classifier utilizes an architecture equivalent to the discriminator in DCGAN, but with a final layer size equal to the number of classes. This classifier is used only for evaluating the clusters found from each compared method.

5.6.3 Comparing Clusters on Samples Directly From The Induced Latent Spaces

In this first experiment we analyze the quality of the clusters found when applying k-means clustering to the latent space induced by each method. For each method, we sample 20,000 latent codes from their induced latent space. We then cluster the latent codes using k-means clustering. Note that we do *not* fit a mapper function K to sample from these clusters in this experiment; we instead perform our analysis directly on the latent codes and their cluster assignment.

For each cluster, we generate the corresponding output samples by passing the instances in each cluster through the pretrained generator. We then use the pretrained classifier to provide class labels for each instance in each cluster. Ideally, each cluster will be

	Metric	Original Generator + GMM	Polarity Sampling + GMM	JBT Sampling + GMM	LASH (Ours) + GMM
`MNIST`	Homogeneity	0.0364 ± 0.0049	0.1494 ± 0.0058	0.3507 ± 0.0142	**0.3781** ± **0.0145**
	Completeness	0.0366 ± 0.0049	0.1584 ± 0.0061	0.3519 ± 0.0143	**0.3801** ± **0.0146**
	V Measure	0.0365 ± 0.0049	0.1538 ± 0.0059	0.3513 ± 0.0142	**0.3791** ± **0.0145**
	Adjusted MI	0.0188 ± 0.0050	0.1377 ± 0.0061	0.3394 ± 0.0145	**0.3677** ± **0.0148**
`Fashion MNIST`	Homogeneity	0.0340 ± 0.0045	0.1649 ± 0.0089	0.1800 ± 0.0124	**0.2107** ± **0.0082**
	Completeness	0.0341 ± 0.0045	0.1677 ± 0.0095	0.1810 ± 0.0125	**0.2122** ± **0.0083**
	V Measure	0.0341 ± 0.0045	0.1663 ± 0.0092	0.1805 ± 0.0125	**0.2115** ± **0.0083**
	Adjusted MI	0.0163 ± 0.0046	0.1509 ± 0.0094	0.1654 ± 0.0127	**0.1970** ± **0.0084**
`Face +Flowers`	Homogeneity	0.0031 ± 0.0014	0.1196 ± 0.0073	0.0932 ± 0.0051	**0.1268** ± **0.0096**
	Completeness	0.0048 ± 0.0021	0.1684 ± 0.0093	0.1533 ± 0.0091	**0.2145** ± **0.0116**
	V Measure	0.0037 ± 0.0016	0.1398 ± 0.0081	0.1159 ± 0.0064	**0.1593** ± **0.0107**
	Adjusted MI	0.0033 ± 0.0016	0.1395 ± 0.0082	0.1155 ± 0.0064	**0.1589** ± **0.0107**

Table 9: Comparative study of the sample quality from the mapper function K. Higher numbers are better for each metric. Best performance is bolded.

strongly correlated with one and only one class - under the assumption that each class lies on a distinct submanifold in the data space.

To measure the correlation between cluster assignments and class labels, we utilize four metrics: *Homogeneity*, *Completeness*, *V Measure* [212], and *Adjusted Mutual Information (MI)* [213].*Homogeneity* is maximized when each cluster contains instances from only a single class, while *Completeness* is maximized if all instances of a given class are given the same cluster assignment. *V Measure* is their harmonic mean, and is thus maximized when there is a perfect correspondence between cluster assignment and class label. *Adjusted MI* measures the mutual information between cluster assignments and class labels, normalized to account for the MI generally increasing even for random assignments when the number of clusters grows.

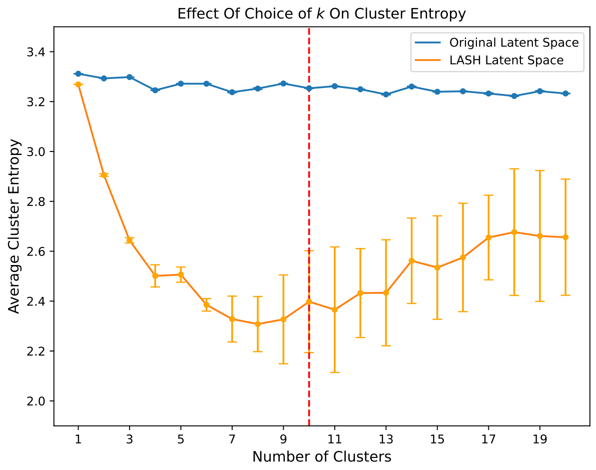
Results are shown in Table 8. First, we note that each method is an order of magnitude higher than the baseless method of applying clusters to the Original Generator's unstructured latent space. Importantly, our method routinely outperforms all other compared methods across all metrics. The only case where our method is not the best performer (and is instead second best) is for the *Homogeneity* score on the `Face + Flowers` dataset, for which Polarity Sampling slightly outperforms LASH. However, LASH significantly outperforms Polarity Sampling on the *Completeness* metric for this setting. This indicates that while each cluster found after applying Polarity Sampling contains mostly instances from single classes, not every class is strongly correlated with some cluster. A likely explanation for this is that polarity sampling over-samples the mode (the face manifold in this case), and thus both clusters it finds mostly contain face images but the flower manifold does not have a cluster it is as strongly associated with.

5.6.4 Comparing Quality of Mixture Models Fit On The Induced Latent Spaces

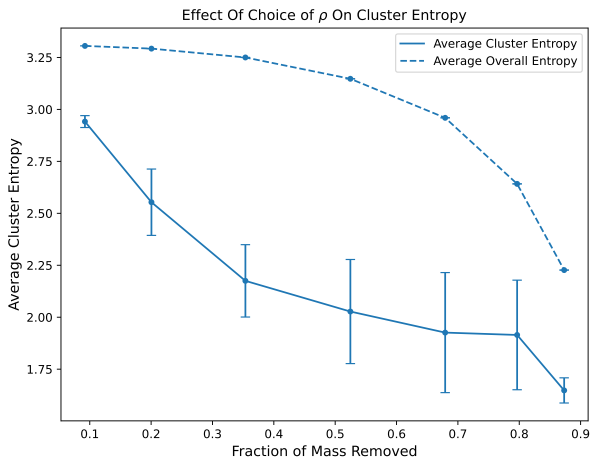
In this experiment, we evaluate our proposed approach of using a Gaussian Mixture Model (GMM) to easily sample from the clusters found in our shattered latent space. To this end, we take the clusters identified in the preceding experiment and fit a GMM on each cluster. We utilize a two-component GMM for each cluster. We then sample instances from each GMM, and compare the correlation between GMM samples and classes in the output space. If the the clusters are correlated with classes or submanifolds in the output space (which is our goal), then the GMM samples should likewise be correlated with these classes if they well-model the clusters. To compare our approach with the state-of-the-art, we likewise fit GMMs on the latent space induced by each compared approach. We utilize the same metrics as in our previous experiment above. Results are shown in Table 9. We see that LASH once again routinely outperforms all other compared methods. Importantly, we note that the performance of each method using the GMM to obtain samples is close to the corresponding performance in the last experiment, where the analysis was performed directly on the latent space without using any mapper network. This indicates that the GMMs are able to well-model the submanifolds in the latent space of each method, implying that using a GMM to cheaply and quickly sample from any desired latent cluster is a promising approach that has only minimal performance cost.

5.6.5 Parameter Study

We analysis the impact of our two important hyperparameters on LASH's performance: the choice of k for k-means clustering, and the amount of mass we chose to remove (analogous to our choice of ρ). We utilize `MNIST` for this experiment. Figure 18a shows the average cluster entropy of LASH's clusters over a range of choices for number of clusters k. If each cluster is strongly associated with some class, the entropy should be low. We see that this is the case so long as the number of clusters is not very significantly over or under estimated. Figure 18b shows both the average cluster entropy, as well as the average total entropy across clusters, as a function of the amount of mass removed. We see that the average cluster entropy decreases significantly as more mass is removed, indicating each cluster becomes more strongly correlated with a class. However, the average clas entropy likewise decreases for the distribution of classes *across* clusters - indicating that as more mass is removed, we start to lose more intances of certain classes causing an increase in class imbalance. We see that removing around 20% of the mass results in minimal loss of overall diversity, while having the desired property of a low average cluster entropy.



(a) Parameter study for number of clusters k



(b) Parameter study for amount of mass removed.

Figure 18: Parameter study for LASH.

5.7 Chapter Summary

In this work we have proposed *LASH*, a novel approach for converting pretrained unconditional generators into conditional generative models. Our approach is based on the idea that by removing latent codes that are mapped to low-probability regions between submanifolds in the data space *shatters* the latent space into disjoint submanifolds, which we can then cluster and adaptively sample from by utilizing helper mapper functions. Notably, we do not require retrianing the generative model or any new deep network. LASH can be easily applied to any pushforward generative model that maps from a lower dimensional latent space with a known distribution into a high-dimensional data space. Our experimental analysis shows that the clusters found using LASH are highly correlated with classes in the output space.

6 Conclusion

6.1 Summary of Contributions

In this dissertation I studied two tasks relating to noisy datasets in the Positive Unlabeled setting, as well as two tasks on repairing and extending pretrained generative models.

Task 1: *I extended Positive Unlabeled learning to the multi-label setting*. I proposed a principled multi-label PU risk formalization and training strategy that can be applied to a wide range of classifiers and data modalities, enabling practitioners and researchers to perform multi-label learning on PU datasets. *This has the potential to be useful to a wide range of deep learning researchers and practitioners.* The datasets commonly used to train deep learning models often contain missing labels, and the approach proposed in Task 1 offers a plug-and-play optimization function to learn accurate and unbiased classifiers given such partially labeled data.

Task 2: *I developed methods to identify and correct for labeling bias in biased Positive Unlabeled data.* I performed a theoretical analysis to determine when identifiable biased PU learning is possible, both in the case where no assumptions are made on the sampling process and in the case where a specific functional form is assumed for the labeling process. Additionally, I proposed methods for performing identifiable PU learning in these settings. *This work can allow practitioners to train less biased and more accurate classifiers, given biased data*. This applies to many domains, as the datasets utilized to build deep learning models often exhibit labeling bias. This work also should be of interest to theoreticians studying biased PU data, as it narrows down the requirements for when the labeling mechanism can be accurately learned.

Task 3: *I proposed an efficient approach for debiasing pretrained generative models*. One contribution of this task is the formalization of *semantically uniform distributions*, which are distributions that place an equal amount of mass on each demographic or semantic attribute of interest. Further, we theoretically prove when it is possible to condition a pretrained generative model to yield a semantically uniform distribution, given only the pretrained generator and an auxiliary (even potentially biased) attribute classifier. *This method has broad applicability, as the leading generative models are known to be biased*. Producing less biased, more representative data is important for both ethical and legal reasons. Due to the low computational cost of our approach, this should be of particular interest to smaller institutions and research groups that can not afford to fine-tune existing models.

Task 4: *I develop a method to turn an unconditional generative model into a conditional model in a completely unsupervised manner, without requiring any retraining.* I propose a method to impose *structure* on the generator's latent space by identifying regions in this space that are mapped to low-probability regions of the model's output distribution. By removing these regions from the latent space, the remaining data will fall on sub-manifolds that each correspond to a semantically meaningful sub-manifold in the output space. Lastly, I show that it is possible to adaptively sample from each of these sub-manifolds by utilizing only a simple Gaussian Mixture Model. *This work should aid researchers developers working with generative models that are trained on unlabeled or partially labeled data.* As our approach requires no classifier feedback, it offers a method for *discovering* what classes a generative model is even capable of producing.

6.2 Future Work

There are several fruitful directions for future work that build off of the ideas presented in this dissertation.

- **Biased Multi-Label PU Learning.** In `Task 1`, I propose a method for performing PU learning in the multi-labeled setting. However, this work assumes that the labeling mechanism that selects which instances of each class are labeled is unbiased. A natural next step is to combine the work in `Task 2`, which deals with biased PU learning in the binary setting, with the multi-label approach performed in `Task 1`.

- **Debiasing Pretrained Model Given Knowledge Of the Biased Labeling Mechanism.** In `Task 2`, we propose methods for creating a model of the biased labeling mechanism itself. One direction for future work is to investigate whether such a model could be used to *debias* a model - either generative or discriminative - that was trained on data labeled according to that biased mechanism. In effect, this would combine ideas from `Task 2` and `Task 3`.

- **Debiasing Continuous Attributes of Generative Models.** The work I did in `Task 3` applies to debiasing binary attributes. A natural next step is to extend this work to handle continuous semantic attributes.

- **Converting Unconditional Diffusion Models Into Conditional Models.** `Task 4` proposes an approach for converting *push forward* generative models that map from a lower dimensional space into a higher dimensional space, such as GANs and VAEs, into conditional models. However, this does not apply to Diffusion models as they map from a space with an equal or higher intrinsic dimension as the data space has. Reformulating this approach to work for models like Diffusion would extend its practicality, as Diffusion models are becoming increasingly prevalent as the state-of-the-art image generators.

- **Debiasing Language Models.** `Task 3`'s debiasing approach applies to methods that utilize a single latent code of fixed dimensionality. While this is common for most generative models that produce images or tabular data, it does not typically apply to language models. A natural extension of this work is to study ways of debiasing models with multiple, varying-dimension latent codes to make our work applicable to language models.

7 List of Publications

7.1 Papers Included In This Dissertation

1. **Walter Gerych**, Tom Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Robust Recurrent Classifier Chains for Multi-Label Learning with Missing Labels.* CIKM, 2022.

2. **Walter Gerych**, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Recovering The Propensity Score From Biased Positive Unlabeled Data.* AAAI, 2022.

3. **Walter Gerych**, Kevin Hickey, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Elke Rundensteiner, Emmanuel Agu. *Debiasing Pretrained Generative Models by Uniformly Sampling Semantic Attributes.* NeurIPS, 2023.

4. **Walter Gerych**, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *Latent Shattering: Turning Unconditional Pretrained Generators Into Conditional Models By Imposing Latent Structure.* Under Review.

7.2 Published First-Author Works Not Included In This Dissertation

1. **Walter Gerych**, Thomas Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Positive Unlabeled Learning with a Sequential Selection Bias.* SDM, 2022.

2. **Walter Gerych**, Tom Hartvigsen, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Recurrent Bayesian Classifier Chains for Exact Multi-Label Classification.* NeurIPS, 2021.

3. **Walter Gerych**, Harrison Kim, Joshua DeOliveira, MaryClare Martin, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Hamid Mansoor, Emmanuel Agu, Elke Rundensteiner. *GAN For Generating User-Specific Human Activity Data From An*

Incomplete Training Corpus. IEEE Big Data 4th Special Session on HealthCare Data, 2021.

4. **Walter Gerych**, Jessica Bader, Declan Nelson, Thalia Chai-Zhang, Luke Buquicchio, Abdulaziz Alajaji, Kevin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *Local Geometry Preserving Deep Networks For Featurizing High-Dimensional Datasets.* IEEE ICMLA, 2021.

5. Caitlin Kuhlman*, **Walter Gerych* (Joint First Author)**, Elke A. Rundensteiner. *Measuring Group Advantage: A Comparative Study of Fair Ranking Metrics.* AIES, 2021.

6. **Walter Gerych**, Luke Buquicchio, Kavin Chandrasekaran, Abdulaziz Alajaji, Hamid Mansoor, Aidan Murphy, Emmanuel Agu, Elke Rundensteiner. *BurstPU: Classification of Weakly Labeled Datasets with Sequential Bias.* IEEE Big Data, 2020.

7. **Walter Gerych**, Emmanuel Agu, Elke Rundensteiner. *Classifying Depression in Imbalanced Datasets Using an Autoencoder- Based Anomaly Detection Approach.* ICSC, 2019.

7.3 In-Submission First-Author Works Not Included In This Dissertation

1. **Walter Gerych**, Kevin Hickey, Thomas Hartvigsen, Luke Buquicchio, Abdulaziz Alajaji, Kavin Chandrasekaran, Hamid Mansoor, Emmanuel Agu, Elke Rundensteiner. *Support-Alignment For Stable GAN Training.* Under Review.

2. **Walter Gerych**, Yara Rizk, Vatche Isahagian, Vinod Muthusamy, Evelyn Duesterwald, Praveen Venkateswaran. *Who Knows The Answer? Finding the Best Model And Prompt For Each Query Using Confidence-Based Search.* Under Review.

7.4 Published Co-Authored Papers

1. Abdulaziz Alajaji, Walter Gerych, Luke Buquicchio, Kavin Chandrasekaran, Hamid Mansoor, Emmanuel Agu, Elke Rundensteiner. *Domain Adaptation Methods for Lab-*

to-Field Human Context Recognition. MDPI Sensors, 2023.

2. Hamid Mansoor, Walter Gerych, Abdulaziz Alajaji, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner, Angela Incollingo Rodriguez. *INPHOVIS: Interactive visual analytics for smartphone-based digital phenotyping.* Visual Informatics, 2023.

3. Abdulaziz Alajaji, Walter Gerych, Kavin Chandrasekaran, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *Adversarial Human Context Recognition: Evasion Attacks and Defenses.* COMPSAC, 2023.

4. Jidapa Thadajarassiri, Thomas Hartvigsen, **Walter Gerych**, Xiangnan Kong, Elke Rundensteiner. *Knowledge Amalgamation for Multi-Label Classification via Label Dependency Transfer.* AAAI, 2023.

5. Joshua DeOliveira, Walter Gerych, Aruzhan Koshkarova, Elke Rundensteiner, Emmanuel Agu. *HAR-CTGAN: A Mobile Sensor Data Generation Tool for Human Activity Recognition.* Big Data, 2022.

6. Thomas Hartvigsen, **Walter Gerych**, Jidapa Thadajarassiri, Xiangnan Kong, Elke Rundensteiner. *Early Classification of Irregular Time Series* CIKM, 2022.

7. ML Tlachac, **Walter Gerych**, Kratika Agrawal, Nicholas Jurovich, Benjamin Litterer, Saitheeraj Thatigotla, Jidapa Thadajarassiri, Elke Rundensteiner. *Conditional Sequence Generative Adversarial Networks for Text Generation.* Big Data, 2022.

8. Luke Buquicchio, **Walter Gerych**, Kavin Chandrasekaran, Abdulaziz Alajaji, Hamid Mansoor, Thomas Hartvigsen, Elke Rundensteiner, Emmanuel Agu. *Variational Open Set Recognition.* IEEE Big Data, 2021.

9. Luke Buquicchio, **Walter Gerych**, Abdulaziz Alajaji, Kavin Chandrasekaran, Hamid Mansoor, Emmanuel Agu, and Elke Rundensteiner, *Few-Shot Classification for Human Context Recognition Using Smartphone Data Traces.* IEEE ICMLA, 2021.

10. Hamid Mansoor, **Walter Gerych**, Abdulaziz Alajaji, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *Visual Analytics of Smartphone-Sensed Human Behavior and Health.* IEEE Computer Graphics and Applications, 2021.

11. Abdulaziz Alajaji, **Walter Gerych**, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *Smartphone Health Biomarkers: Positive Unlabeled Learning of In-the-Wild Contexts.* Pervasive Computing, 2021.

12. Hamid Mansoor, **Walter Gerych**, Abdulaziz Alajaji, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *PLEADES: Population Level Observation of Smartphone Sensed Symptoms for In-the-wild Data using Clustering.* VISIGRAPP, 2021.

13. Kavin Chandrasekaran, **Walter Gerych**, Luke Buquicchio, Abdulaziz Alajaji, Emmanuel Agu, Elke Rundensteiner. *CARTMAN: Complex Activity Recognition Using Topic Models for Feature Generation from Wearable Sensor Data.* SMARTCOMP, 2021.

14. Hamid Mansoor, **Walter Gerych**, Luke Buquicchio, Abdulaziz Alajaji, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *INTOSIS: Interactive Observation of Smartphone Inferred Symptoms for In-The-Wild Data.* IEEE Big Data, 2020.

15. Abdulaziz Alajaji, **Walter Gerych**, Kavin Chandrasekaran, Luke Buquicchio, Emmanuel Agu, Elke Rundensteiner. *DeepContext: Parameterized Compatibility-Based Attention CNN for Human Context Recognition.* ICSC, 2020.

16. Hamid Mansoor, **Walter Gerych**, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *ARGUS: Interactive Visual Analytics Framework for the Discovery of Disruptions in Bio-Behavioral Rhythms.* EuroVis (Short Papers), 2020.

17. Hamid Mansoor, **Walter Gerych**, Luke Buquicchio, Kavin Chandrasekaran, Emmanuel Agu, Elke Rundensteiner. *COMEX: Identifying Mislabeled Human Behavioral Context Data Using Visual Analytics.* COMPSAC, 2019.

References

[1] M. Kelly, R. Longjohn, and K. Nottingham, "UCI Machine Learning Repository." [Online]. Available: https://archive.ics.uci.edu/citation

[2] X. Chen, "Awesome Public Datasets," Sep. 2023, original-date: 2014-11-20T06:20:50Z. [Online]. Available: https://github.com/awesomedata/awesome-public-datasets

[3] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," Jul. 2020, arXiv:1910.03771 [cs]. [Online]. Available: http://arxiv.org/abs/1910.03771

[4] J. Y. Koh, "Model Zoo - Deep learning code and pretrained models for transfer learning, educational purposes, and more." [Online]. Available: https://modelzoo.co/

[5] "OpenML." [Online]. Available: https://www.openml.org/

[6] "Model Zoo - Deep learning code and pretrained models for transfer learning, educational purposes, and more." [Online]. Available: https://modelzoo.co/

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[8] M. AlQuraishi, "ProteinNet: a standardized data set for machine learning of protein structure," *BMC Bioinformatics*, vol. 20, no. 1, p. 311, Jun. 2019. [Online]. Available: https://doi.org/10.1186/s12859-019-2932-0

[9] B. Cottier, "Trends in the dollar training cost of machine learning systems," Jan. 2023. [Online]. Available: https://epochai.org/blog/trends-in-the-dollar-training-cost-of-machine-learning-systems

[10] A. J. Larrazabal, N. Nieto, V. Peterson, D. H. Milone, and E. Ferrante, "Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis," *Proceedings of the National Academy of Sciences*, vol. 117, no. 23, pp. 12 592–12 594, Jun. 2020, publisher: Proceedings of the National Academy of Sciences. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.1919012117

[11] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A Survey on Bias and Fairness in Machine Learning," *ACM Computing Surveys*, vol. 54, no. 6, pp. 115:1–115:35, Jul. 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3457607

[12] G. James, D. Witten, T. Hastie, R. Tibshirani *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[13] S. Yun, S. J. Oh, B. Heo, D. Han, J. Choe, and S. Chun, "Re-labeling ImageNet: from Single to Multi-Labels, from Global to Localized Labels," Jul. 2021, arXiv:2101.05022 [cs]. [Online]. Available: http://arxiv.org/abs/2101.05022

[14] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy, "The Pile: An 800GB Dataset of Diverse Text for Language Modeling," in *Preprint at https://arxiv.org/abs/2101.00027*. Preprint at https://arxiv.org/abs/2101.00027, Dec. 2020.

[15] Y. Yang, "Object Detection as a Positive-Unlabeled Problem," in *The 31st British Machine Vision Virtual Conference*, 2020.

[16] B. Velichkovska, D. Denkovski, H. Gjoreski, M. Kalendar, and V. Osmani, "A Survey of Bias in Healthcare: Pitfalls of Using Biased Datasets and Applications," in *Artificial Intelligence Application in Networks and Systems*, ser. Lecture Notes in Networks and Systems, R. Silhavy and P. Silhavy, Eds. Cham: Springer International Publishing, 2023, pp. 570–584.

[17] J. M. Johnson and T. M. Khoshgoftaar, "A Survey on Classifying Big Data with Label Noise," *Journal of Data and Information Quality*, vol. 14, no. 4, pp. 1–43, Dec. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3492546

[18] J. Buolamwini and T. Gebru, "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification," in *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*. PMLR, Jan. 2018, pp. 77–91, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v81/buolamwini18a.html

[19] H. Guo, L. Zhu, and K. Huang, "Are GANs Biased? Evaluating GAN-Generated Facial Images via Crowdsourcing," in *NeurIPS 2022 Workshop on Human Evaluation of Generative Models.*, 2022.

[20] V. H. Maluleke, N. Thakkar, T. Brooks, E. Weber, T. Darrell, A. A. Efros, A. Kanazawa, and D. Guillory, "Studying Bias in GANs through the Lens of Race," Sep. 2022, arXiv:2209.02836 [cs]. [Online]. Available: http://arxiv.org/abs/2209.02836

[21] A. S. Luccioni, C. Akiki, M. Mitchell, and Y. Jernite, "Stable Bias: Analyzing Societal Representations in Diffusion Models," Mar. 2023, arXiv:2303.11408 [cs]. [Online]. Available: http://arxiv.org/abs/2303.11408

[22] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. M. A. Patwary, Y. Yang, and Y. Zhou, "Deep Learning Scaling is

Predictable, Empirically," Dec. 2017, arXiv:1712.00409 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1712.00409

[23] R. Schwartz, A. Vassilev, K. Greene, L. Perine, A. Burt, and P. Hall, "Towards a standard for identifying and managing bias in artificial intelligence," National Institute of Standards and Technology (U.S.), Gaithersburg, MD, Tech. Rep. NIST SP 1270, Mar. 2022. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1270.pdf

[24] H. Ledford, "Millions of black people affected by racial bias in health-care algorithms," *Nature*, vol. 574, no. 7780, pp. 608–609, Oct. 2019. [Online]. Available: https://www.nature.com/articles/d41586-019-03228-6

[25] H. Laria, Y. Wang, J. Van De Weijer, and B. Raducanu, "Transferring Unconditional to Conditional GANs with Hyper-Modulation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. New Orleans, LA, USA: IEEE, Jun. 2022, pp. 3839–3848. [Online]. Available: https://ieeexplore.ieee.org/document/9857039/

[26] J. Bekker and J. Davis, "Learning from positive and unlabeled data: a survey," *Machine Language*, vol. 109, no. 4, pp. 719–760, Apr. 2020. [Online]. Available: https://doi.org/10.1007/s10994-020-05877-5

[27] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 213–220. [Online]. Available: https://dl.acm.org/doi/10.1145/1401890.1401920

[28] B. Liu, Y. Dai, X. Li, W. Lee, and P. Yu, “Building text classifiers using positive and unlabeled examples,” in *Third IEEE International Conference on Data Mining*, Nov. 2003, pp. 179–186.

[29] B. Gabrys, “Learning with Missing or Incomplete Data,” in *Image Analysis and Processing – ICIAP 2009*, ser. Lecture Notes in Computer Science, P. Foggia, C. Sansone, and M. Vento, Eds. Berlin, Heidelberg: Springer, 2009, pp. 1–4.

[30] T. Katsuki and T. Osogami, “Regression with Sensor Data Containing Incomplete Observations,” in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 15 911–15 927, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v202/katsuki23a.html

[31] G. Kleinberg, M. J. Diaz, S. Batchu, and B. Lucke-Wold, “Racial underrepresentation in dermatological datasets leads to biased machine learning models and inequitable healthcare,” *Journal of biomed research*, vol. 3, no. 1, pp. 42–47, 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9815490/

[32] R. Mahadev and A. Chakravarti, “Understanding Gender and Racial Disparities in Image Recognition Models,” Jul. 2021, arXiv:2107.09211 [cs]. [Online]. Available: http://arxiv.org/abs/2107.09211

[33] S. Zhioua and R. Binkytė, “Shedding light on underrepresentation and Sampling Bias in machine learning,” Jun. 2023, arXiv:2306.05068 [cs]. [Online]. Available: http://arxiv.org/abs/2306.05068

[34] P. Bertail, S. Clémençon, Y. Guyonvarch, and N. Noiry, “Learning from Biased Data: A Semi-Parametric Approach,” in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 803–812, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v139/bertail21a.html

[35] K. Zhang, B. Scholkopf, K. Muandet, and Z. Wang, "Domain Adaptation under Target and Conditional Shift," in *International Conference on Learning Representations*. International Conference on Learning Representations, Jun. 2013. [Online]. Available: https://www.semanticscholar.org/paper/Domain-Adaptation-under-Target-and-Conditional-Zhang-Scholkopf/66c3d69f94c90a884d3f6b5367813d51708f6ded

[36] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring Robustness to Natural Distribution Shifts in Image Classification," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 18 583–18 599. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/d8330f857a17c53d217014ee776bfd50-Abstract.html

[37] J. Liang, R. He, and T. Tan, "A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts," Mar. 2023, arXiv:2303.15361 [cs]. [Online]. Available: http://arxiv.org/abs/2303.15361

[38] C.-M. Teng, "Correcting Noisy Data," in *Proceedings of the Sixteenth International Conference on Machine Learning*, ser. ICML '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jun. 1999, pp. 239–248.

[39] K. Poulinakis, D. Drikakis, I. W. Kokkinakis, and S. M. Spottswood, "Machine-Learning Methods on Noisy and Sparse Data," *Mathematics*, vol. 11, no. 1, p. 236, Jan. 2023, number: 1 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2227-7390/11/1/236

[40] C. L. Giles, S. Lawrence, and A. C. Tsoi, "Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference," *Machine Learning*, vol. 44, no. 1, pp. 161–183, Jul. 2001. [Online]. Available: https://doi.org/10.1023/A:1010884214864

[41] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from Noisy Labels with Deep Neural Networks: A Survey," Mar. 2022, arXiv:2007.08199 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2007.08199

[42] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari, "Learning with Noisy Labels," in *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://papers.nips.cc/paper_files/paper/2013/hash/3871bd64012152bfb53fdf04b401193f-Abstract.html

[43] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making Deep Neural Networks Robust to Label Noise: A Loss Correction Approach," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 2233–2241. [Online]. Available: http://ieeexplore.ieee.org/document/8099723/

[44] C. Scott, "A Rate of Convergence for Mixture Proportion Estimation, with Application to Learning from Noisy Labels," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Feb. 2015, pp. 838–846, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v38/scott15.html

[45] X. Li, T. Liu, B. Han, G. Niu, and M. Sugiyama, "Provably End-to-end Label-Noise Learning without Anchor Points," Oct. 2021, arXiv:2102.02400 [cs]. [Online]. Available: http://arxiv.org/abs/2102.02400

[46] T. Liu and D. Tao, "Classification with Noisy Labels by Importance Reweighting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 3, pp. 447–461, Mar. 2016, arXiv:1411.7718 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1411.7718

[47] K. Zupanc and J. Davis, "Estimating Rule Quality for Knowledge Base Completion with the Relationship between Coverage Assumption," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. Lyon, France: ACM Press, 2018, pp. 1073–1081. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3178876.3186006

[48] F. Li, S. Dong, A. Leier, M. Han, X. Guo, J. Xu, X. Wang, S. Pan, C. Jia, Y. Zhang, G. I. Webb, L. J. M. Coin, C. Li, and J. Song, "Positive-unlabeled learning in bioinformatics and computational biology: a brief review," *Briefings in Bioinformatics*, vol. 23, no. 1, p. bbab461, Jan. 2022.

[49] A. Alajaji, W. Gerych, L. Buquicchio, K. Chandrasekaran, H. Mansoor, E. Agu, and E. A. Rundensteiner, "Smartphone Health Biomarkers: Positive Unlabeled Learning of In-the-Wild Contexts," *IEEE Pervasive Computing*, vol. 20, no. 1, pp. 50–61, Jan. 2021, conference Name: IEEE Pervasive Computing.

[50] W. Gerych, L. Buquicchio, K. Chandrasekaran, A. Alajaji, H. Mansoor, A. Murphy, E. Rundensteiner, and E. Agu, "BurstPU: Classification of Weakly Labeled Datasets with Sequential Bias," in *2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 147–154.

[51] M. Christoffel, G. Niu, and M. Sugiyama, "Class-prior Estimation for Learning from Positive and Unlabeled Data," in *Asian Conference on Machine Learning*. PMLR, Feb. 2016, pp. 221–236, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v45/Christoffel15.html

[52] J. Bekker and J. Davis, "Estimating the Class Prior in Positive and Unlabeled Data Through Decision Tree Induction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/11715

[53] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-Unlabeled Learning with Non-Negative Risk Estimator," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/7cce53cf90577442771720a370c3c723-Abstract.html

[54] J. Bekker, P. Robberechts, and J. Davis, "Beyond the Selected Completely at Random Assumption for Learning from Positive and Unlabeled Data," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science, U. Brefeld, E. Fromont, A. Hotho, A. Knobbe, M. Maathuis, and C. Robardet, Eds. Cham: Springer International Publishing, 2020, pp. 71–85.

[55] T. Gong, B. Liu, Q. Chu, and N. Yu, "Using multi-label classification to improve object detection," *Neurocomputing*, vol. 370, pp. 174–185, Dec. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231219312585

[56] R. Li, W. Liu, Y. Lin, H. Zhao, and C. Zhang, "An Ensemble Multilabel Classification for Disease Risk Prediction," *Journal of Healthcare Engineering*, vol. 2017, p. 8051673, 2017.

[57] W. H. Shrank, A. R. Patrick, and M. Alan Brookhart, "Healthy User and Related Biases in Observational Studies of Preventive Interventions: A Primer for Physicians," *Journal of General Internal Medicine*, vol. 26, no. 5, pp. 546–550, May 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3077477/

[58] W. Jiang, N. Synovic, M. Hyattt, T. Schorlemmer, R. Sethi, Y.-H. Lu, G. Thiruvathukal, and J. Davis, "An Empirical Study of Pre-Trained Model Reuse in the Hugging Face Deep Learning Model Registry," *IEEE/ACM*

International Conference on Software Engineering, May 2023. [Online]. Available: https://ecommons.luc.edu/cs_facpubs/319

[59] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, “On the Opportunities and Risks of Foundation Models,” Jul. 2022, arXiv:2108.07258 [cs]. [Online]. Available: http://arxiv.org/abs/2108.07258

[60] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li, and M. Sun, “Parameter-efficient fine-tuning of large-scale pre-trained language models,” *Nature Machine Intelligence*, vol. 5, no. 3, pp. 220–235, Mar. 2023, number: 3 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s42256-023-00626-4

[61] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations*. International Conference on Learning Representations, Oct. 2021. [Online]. Available: https://openreview.net/forum?id=nZeVKeeFYf9

[62] T. Alshalali and D. Josyula, "Fine-Tuning of Pre-Trained Deep Learning Models with Extreme Learning Machine," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2018, pp. 469–473.

[63] B. Chu, V. Madhavan, O. Beijbom, J. Hoffman, and T. Darrell, "Best Practices for Fine-Tuning Visual Classifiers to New Domains," in *Computer Vision – ECCV 2016 Workshops*, ser. Lecture Notes in Computer Science, G. Hua and H. Jégou, Eds. Cham: Springer International Publishing, 2016, pp. 435–442.

[64] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, conference Name: Proceedings of the IEEE.

[65] G. Y. Park, S. Lee, S. W. Lee, and J. C. Ye, "Training Debiased Subnetworks with Contrastive Weight Pruning," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vancouver, BC, Canada: IEEE, Jun. 2023, pp. 7929–7938. [Online]. Available: https://ieeexplore.ieee.org/document/10204608/

[66] R. Marcinkevics, E. Ozkan, and J. E. Vogt, "Debiasing Deep Chest X-Ray Classifiers using Intra- and Post-processing Methods," in *Proceedings of the 7th Machine Learning for Healthcare Conference*. PMLR, Dec. 2022, pp. 504–536, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v182/marcinkevics22a.html

[67] M. Gira, R. Zhang, and K. Lee, "Debiasing Pre-Trained Language Models via Efficient Fine-Tuning," in *Proceedings of the Second Workshop on Language*

Technology for Equality, Diversity and Inclusion. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 59–69. [Online]. Available: https://aclanthology.org/2022.ltedi-1.8

[68] M. Kaneko and D. Bollegala, "Debiasing Pre-trained Contextualised Embeddings," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1256–1266. [Online]. Available: https://aclanthology.org/2021.eacl-main.107

[69] S. Luo, X. Wang, G. Fang, Y. Hu, D. Tao, and M. Song, "Knowledge amalgamation from heterogeneous networks by common feature learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. Macao, China: AAAI Press, Aug. 2019, pp. 3087–3093.

[70] J. Ye, Y. Ji, X. Wang, X. Gao, and M. Song, "Data-Free Knowledge Amalgamation via Group-Stack Dual-GAN," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 12 513–12 522, iSSN: 2575-7075.

[71] J. Thadajarassiri, T. Hartvigsen, W. Gerych, X. Kong, and E. Rundensteiner, "Knowledge Amalgamation for Multi-Label Classification via Label Dependency Transfer," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 8, pp. 9980–9988, Jun. 2023, number: 8. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26190

[72] J. Zhu, H. Ma, J. Chen, and J. Yuan, "DomainStudio: Fine-Tuning Diffusion Models for Domain-Driven Image Generation using Limited Data," Aug. 2023, arXiv:2306.14153 [cs]. [Online]. Available: http://arxiv.org/abs/2306.14153

[73] N. Meade, E. Poole-Dayan, and S. Reddy, "An Empirical Survey of the Effectiveness of Debiasing Techniques for Pre-trained Language Models," in *Proceedings of the*

60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1878–1898. [Online]. Available: https://aclanthology.org/2022.acl-long.132

[74] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A Comprehensive Survey on Transfer Learning," Jun. 2020, arXiv:1911.02685 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1911.02685

[75] Y.-X. Ding, X.-Z. Wu, K. Zhou, and Z.-H. Zhou, "Pre-Trained Model Reusability Evaluation for Small-Data Transfer Learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 37 389–37 400, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/f308b5f207348484552997c536375654-Abstract-Conference.html

[76] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman, "Intermediate-Task Transfer Learning with Pretrained Language Models: When and Why Does It Work?" in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5231–5247. [Online]. Available: https://aclanthology.org/2020.acl-main.467

[77] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio, "Transfusion: Understanding Transfer Learning for Medical Imaging," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/eb1e78328c46506b46a4ac4a1e378b91-Abstract.html

[78] C. H. Wu, S. Motamed, S. Srivastava, and F. D. De la Torre, "Generative Visual Prompt: Unifying Distributional Control of Pre-Trained Generative Models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 422–22 437, Dec.

2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/8cb1c53863b290ee09b94d17f16ef355-Abstract-Conference.html

[79] J. Engel, M. Hoffman, and A. Roberts, "Latent Constraints: Learning to Generate Conditionally from Unconditional Generative Models," in *International Conference on Learning Representations*, Feb. 2018. [Online]. Available: https://openreview.net/forum?id=Sy8XvGb0-

[80] X. Shao, K. Stelzner, and K. Kersting, "Right for the Right Latent Factors: Debiasing Generative Models via Disentanglement," Feb. 2022, arXiv:2202.00391 [cs]. [Online]. Available: http://arxiv.org/abs/2202.00391

[81] W. Gerych, T. Hartvigsen, L. Buquicchio, E. Agu, and E. Rundensteiner, "Robust recurrent classifier chains for multi-label learning with missing labels," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 582–591.

[82] ——, "Recovering the propensity score from biased positive unlabeled data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 6694–6702.

[83] W. Gerych, K. Hickey, L. Buquicchio, E. Rundensteiner, and E. Agu, "Debiasing pre-trained generative models by uniformly sampling semantic attributes," in *Advances In Neural Information Processing Systems (NeurIPS)*, 2023.

[84] B. C. A. Brown, A. L. Caterini, B. L. Ross, J. C. Cresswell, and G. Loaiza-Ganem, "Verifying the Union of Manifolds Hypothesis for Image Data," Mar. 2023, arXiv:2207.02862 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2207.02862

[85] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji, "Multi-label learning with missing labels for image annotation and facial action unit recognition," *Pattern Recognition*, 2015.

[86] J. Deng, O. Russakovsky, J. Krause, M. S. Bernstein, A. Berg, and L. Fei-Fei, "Scalable multi-label annotation," in *SIGCHI*, 2014.

[87] J. Nam, E. L. Mencía, H. J. Kim, and J. Fürnkranz, "Maximizing subset accuracy with recurrent neural networks in multi-label classification," in *NeurIPS*, 2017.

[88] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains: A review and perspectives," *arXiv*, 2019.

[89] ——, "Classifier chains for multi-label classification," in *ECML PKDD*, 2009.

[90] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. F. Wang, "Order-free rnn with visual attention for multi-label classification," in *AAAI*, 2018.

[91] C.-P. Tsai and H.-Y. Lee, "Order-free learning alleviating exposure bias in multi-label classification," in *AAAI*, 2020.

[92] W. Gerych, T. Hartvigsen, L. Buquicchio, E. Agu, and E. Rundensteiner, "Recurrent bayesian classifier chains for exact multi-label classification," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[93] X. Kong, Z. Wu, L.-J. Li, R. Zhang, P. S. Yu, H. Wu, and W. Fan, "Large-scale multi-label learning with incomplete label assignments," in *SDM*, 2014.

[94] L. Wang, Y. Liu, C. Qin, G. Sun, and Y. Fu, "Dual relation semi-supervised multi-label learning." in *AAAI*, 2020.

[95] A. H. Akbarnejad and M. S. Baghshah, "An efficient semi-supervised multi-label classifier capable of handling missing labels," *IEEE TKDE*, 2019.

[96] M. Xu, Y.-F. Li, and Z.-H. Zhou, "Robust multi-label learning with pro loss," *IEEE TKDE*, 2019.

[97] H.-M. Chu, C.-K. Yeh, and Y.-C. Frank Wang, "Deep generative models for weakly-supervised multi-label classification," in *ECCV*, 2018.

[98] Q. Tan, Y. Yu, G. Yu, and J. Wang, "Semi-supervised multi-label classification using incomplete label information," *Neurocomputing*, 2017.

[99] A. Kanehira and T. Harada, "Multi-label ranking from positive and unlabeled data," in *CVPR*, 2016, pp. 5138–5146.

[100] W. Liu, H. Wang, X. Shen, and I. Tsang, "The emerging trends of multi-label learning," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[101] Y. Liu, R. Jin, and L. Yang, "Semi-supervised multi-label learning by constrained non-negative matrix factorization," in *AAAi*, vol. 6, 2006, pp. 421–426.

[102] X. Kong, M. K. Ng, and Z.-H. Zhou, "Transductive multilabel learning via label set propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 704–719, 2011.

[103] H. Wang, Z. Li, J. Huang, P. Hui, W. Liu, T. Hu, and G. Chen, "Collaboration based multi-label propagation for fraud detection," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 2477–2483.

[104] Y. Zhu, J. T. Kwok, and Z.-H. Zhou, "Multi-label learning with global and local label correlation," *IEEE TKDE*, 2017.

[105] L. Sun, S. Feng, G. Lyu, and C. Lang, "Robust semi-supervised multi-label learning by triple low-rank regularization," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2019, pp. 269–280.

[106] B. Wu, Z. Liu, S. Wang, B.-G. Hu, and Q. Ji, "Multi-label learning with missing labels," in *2014 22nd International Conference on Pattern Recognition*. IEEE, 2014.

[107] M. Xu, R. Jin, and Z.-H. Zhou, "Speedup matrix completion with side information: Application to multi-label learning," *Advances in neural information processing systems*, 2013.

[108] C.-K. Yeh, W.-C. Wu, W.-J. Ko, and Y.-C. F. Wang, "Learning deep latent space for multi-label classification," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[109] B. Wu, F. Jia, W. Liu, B. Ghanem, and S. Lyu, "Multi-label learning with missing labels using mixed dependency graphs," *International Journal of Computer Vision*, 2018.

[110] Y.-Y. Sun, Y. Zhang, and Z.-H. Zhou, "Multi-label learning with weak label," in *Twenty-fourth AAAI conference on artificial intelligence*, 2010.

[111] H. Yang, J. T. Zhou, and J. Cai, "Improving multi-label learning with missing labels by structured semantic correlations," in *European conference on computer vision*. Springer, 2016.

[112] H.-F. Yu, H.-Y. Huang, I. Dhillon, and C.-J. Lin, "A unified algorithm for one-cass structured matrix factorization with side information," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[113] P. Teisseyre, "Classifier chains for positive unlabelled multi-label learning," *Knowledge-Based Systems*, 2021.

[114] E. Cole, O. Mac Aodha, T. Lorieul, P. Perona, D. Morris, and N. Jojic, "Multi-label learning from single positive labels," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[115] W. Cheng, E. Hüllermeier, and K. J. Dembczynski, "Bayes optimal multilabel classification via probabilistic classifier chains," in *ICML*, 2010.

[116] J. Bekker and J. Davis, “Learning from positive and unlabeled data: A survey,” *Machine Learning*, vol. 109, no. 4, pp. 719–760, 2020.

[117] C. Elkan and K. Noto, “Learning classifiers from only positive and unlabeled data,” in *ACM SIGKDD*, 2008.

[118] G. Su, W. Chen, and M. Xu, “Positive-unlabeled learning from imbalanced data,” in *Proceedings of the 30th International Joint Conference on Artificial Intelligence, Virtual Event*, 2021.

[119] S. Jain, J. Delano, H. Sharma, and P. Radivojac, “Class prior estimation with biased positives and unlabeled examples,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4255–4263.

[120] W. Gerych, L. Buquicchio, K. Chandrasekaran, A. Alajaji, H. Mansoor, A. Murphy, E. Rundensteiner, and E. Agu, “Burstpu: Classification of weakly labeled datasets with sequential bias,” in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020.

[121] E. Sansone, F. G. De Natale, and Z.-H. Zhou, “Efficient training for positive unlabeled learning,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[122] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang, “Positive-unlabeled learning in streaming networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016.

[123] Y. Zhou, J. Xu, J. Wu, Z. Taghavi, E. Korpeoglu, K. Achan, and J. He, “Pure: Positive-unlabeled recommendation with generative adversarial network,” in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[124] R. Kiryo, G. Niu, M. C. Du Plessis, and M. Sugiyama, “Positive-unlabeled learning with non-negative risk estimator,” *Advances in neural information processing systems*, 2017.

[125] M. Du Plessis, G. Niu, and M. Sugiyama, “Convex formulation for learning from positive and unlabeled data,” in *ICML*, 2015.

[126] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[127] J. Bekker and J. Davis, “Estimating the class prior in positive and unlabeled data through decision tree induction,” in *AAAI*, 2018.

[128] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.”

[129] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, “Learning multi-label scene classification,” *Pattern Recognition*, 2004.

[130] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth, “Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary,” in *ECCV*, 2002.

[131] C. G. Northcutt, L. Jiang, and I. L. Chuang, “Confident learning: Estimating uncertainty in dataset labels,” *arXiv*, 2019.

[132] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *NeurIPS*, 2019.

[133] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv*, 2014.

[134] S. S. Bucak, R. Jin, and A. K. Jain, "Multi-label learning with incomplete class assignments," in *CVPR*, 2011.

[135] G. Niu, M. C. du Plessis, T. Sakai, Y. Ma, and M. Sugiyama, "Theoretical comparisons of positive-unlabeled learning against positive-negative learning," in *NeurIPS*, 2016.

[136] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Third IEEE International Conference on Data Mining*. IEEE, 2003, pp. 179–186.

[137] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *Advances In Neural Information Processing Systems (NeurIPS)*, 2017, pp. 1675–1685.

[138] S. Jain, M. White, and P. Radivojac, "Recovering true classifier performance in positive-unlabeled learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, S. P. Singh and S. Markovitch, Eds. AAAI Press, 2017, pp. 2066–2072.

[139] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 213–220.

[140] T. Guo, C. Xu, J. Huang, Y. Wang, B. Shi, C. Xu, and D. Tao, "On positive-unlabeled classification in gan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8385–8393.

[141] J. Bekker and J. Davis, "Estimating the class prior in positive and unlabeled data through decision tree induction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[142] S. Jain, M. White, and P. Radivojac, "Estimating the class prior and posterior from noisy positives and unlabeled data," in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 2685–2693.

[143] D. Zeiberg, S. Jain, and P. Radivojac, "Fast nonparametric estimation of class proportions in the positive-unlabeled classification setting," in *AAAI*, 2020.

[144] Z. Hammoudeh and D. Lowd, "Learning from positive and unlabeled data with arbitrary positive shift," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 13 088–13 099.

[145] W. Gerych, L. Buquicchio, K. Chandrasekaran, A. Alajaji, H. Mansoor, A. Murphy, E. A. Rundensteiner, and E. O. Agu, "Burstpu: Classification of weakly labeled datasets with sequential bias," *2020 IEEE International Conference on Big Data (Big Data)*, pp. 147–154, 2020.

[146] F. He, T. Liu, G. I. Webb, and D. Tao, "Instance-dependent PU learning by bayesian optimal relabeling," *CoRR*, vol. abs/1808.02180, 2018.

[147] N. Youngs, D. E. Shasha, and R. Bonneau, "Positive-unlabeled learning in the face of labeling bias," in *IEEE International Conference on Data Mining Workshop, ICDMW 2015, Atlantic City, NJ, USA, November 14-17, 2015*. IEEE Computer Society, 2015, pp. 639–645. [Online]. Available: https://doi.org/10.1109/ICDMW.2015.207

[148] M. Kato, T. Teshima, and J. Honda, "Learning from positive and unlabeled data with a selection bias," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[149] B. Na, H. Kim, K. Song, W. Joo, Y. Kim, and I. Moon, "Deep generative positive-unlabeled learning under selection bias," in *CIKM '20: The 29th ACM International*

Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020. ACM, 2020, pp. 1155–1164.

[150] B. Rhodes, K. Xu, and M. U. Gutmann, "Telescoping density-ratio estimation," *arXiv preprint arXiv:2006.12204*, 2020.

[151] M. Sugiyama, T. Suzuki, and T. Kanamori, *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[152] M. Sugiyama, "Density ratio estimation: A new versatile tool for machine learning," in *Asian Conference on Machine Learning*. Springer, 2009, pp. 6–9.

[153] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[154] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[155] P. Horton and K. Nakai, "A probabilistic classification system for predicting the cellular localization sites of proteins." in *Ismb*, vol. 4, 1996, pp. 109–115.

[156] S. Aeberhard, D. Coomans, and O. De Vel, "Comparative analysis of statistical pattern recognition methods in high dimensional settings," *Pattern Recognition*, vol. 27, no. 8, pp. 1065–1077, 1994.

[157] R. J. Lyon, B. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles, "Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach," *Monthly Notices of the Royal Astronomical Society*, vol. 459, no. 1, pp. 1104–1123, 2016.

[158] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and co2 measurements using statistical learning models," *Energy and Buildings*, vol. 112, pp. 28–39, 2016.

[159] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Kdd*, vol. 96, 1996, pp. 202–207.

[160] H. Sajnani, V. Saini, K. Kumar, E. Gabrielova, P. Choudary, and C. Lopes, "Classifying yelp reviews into relevant categories," 2012.

[161] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge 2007 (voc2007) results http://www. pascal-network. org/challenges," in *VOC/voc2007/workshop/index. html*, 2007.

[162] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[163] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Frowing Of GANs For Improved Quality, Stability, And Variation," in *International Conference on Learning Representations*, 2018.

[164] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," Apr. 2022, arXiv:2112.10752 [cs]. [Online]. Available: http://arxiv.org/abs/2112.10752

[165] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative Adversarial Networks for Conditional Waveform Synthesis," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/6804c9bca0a615bdb9374d00a9fcba59-Abstract.html

[166] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong, "DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models," in *International Conference on Learning*

Representations, Feb. 2023. [Online]. Available: https://openreview.net/forum?id=jQj-_rLVXsj

[167] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language Models are Few-Shot Learners," Jul. 2020, arXiv:2005.14165 [cs]. [Online]. Available: http://arxiv.org/abs/2005.14165

[168] K. Cong and M. Zhou, "Face Dataset Augmentation with Generative Adversarial Network," *Journal of Physics: Conference Series*, vol. 2218, no. 1, p. 012035, Mar. 2022, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2218/1/012035

[169] M. Luo, J. Cao, X. Ma, X. Zhang, and R. He, "FA-GAN: Face Augmentation GAN for Deformation-Invariant Face Recognition," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2341–2355, 2021, conference Name: IEEE Transactions on Information Forensics and Security.

[170] S. Festag, J. Denzler, and C. Spreckelsen, "Generative adversarial networks for biomedical time series forecasting and imputation," *Journal of Biomedical Informatics*, vol. 129, p. 104058, May 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046422000740

[171] M. Orcutt, "Are Face Recognition Systems Accurate? Depends on Your Race." 2016. [Online]. Available: https://www.technologyreview.com/2016/07/06/158971/are-face-recognition-systems-accurate-depends-on-your-race/

[172] M. Mehrab Tanjim, R. Sinha, K. K. Singh, S. Mahadevan, D. Arbour, M. Sinha, and G. W. Cottrell, "Generating and Controlling Diversity in Image Search," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 3908–3916. [Online]. Available: https://ieeexplore.ieee.org/document/9706969/

[173] S. Bhat, J. Jiang, O. Pooladzandi, and G. Pottie, "De-Biasing Generative Models using Counterfactual Methods," Feb. 2023, arXiv:2207.01575 [cs]. [Online]. Available: http://arxiv.org/abs/2207.01575

[174] A. Jahanian*, L. Chai*, and P. Isola, "On the "steerability" of generative adversarial networks," in *International Conference on Learning Representations*, vol. International Conference on Learning Representations. International Conference on Learning Representations, Dec. 2019. [Online]. Available: https://openreview.net/forum?id=HylsTT4FvB

[175] Y. Shen, J. Gu, X. Tang, and B. Zhou, "Interpreting the Latent Space of GANs for Semantic Face Editing," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 9240–9249. [Online]. Available: https://ieeexplore.ieee.org/document/9157070/

[176] E. Denton, B. Hutchinson, M. Mitchell, T. Gebru, and A. Zaldivar, "Image Counterfactual Sensitivity Analysis for Detecting Unintended Bias," Oct. 2020, arXiv:1906.06439 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1906.06439

[177] H. Rangwani, K. R. Mopuri, and R. V. Babu, "Class balancing GAN with a classifier in the loop," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*. PMLR, Dec. 2021, pp. 1618–1627, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v161/rangwani21a.html

[178] Y. Shen, C. Yang, X. Tang, and B. Zhou, "InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs," Oct. 2020, arXiv:2005.09635 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2005.09635

[179] P. Doubinsky, N. Audebert, M. Crucianu, and H. Le Borgne, "Multi-attribute balanced sampling for disentangled GAN controls," *Pattern Recognition Letters*, vol. 162, pp. 56–62, Oct. 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167865522002501

[180] A. I. Humayun, R. Balestriero, and R. Baraniuk, "MaGNET: Uniform Sampling from Deep Generative Network Manifolds Without Retraining," in *International Conference on Learning Representations*, Jan. 2022. [Online]. Available: https://openreview.net/forum?id=r5qumLiYwf9

[181] ——, "Polarity Sampling: Quality and Diversity Control of Pre-Trained Generative Networks via Singular Values," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, Jun. 2022, pp. 10 631–10 640. [Online]. Available: https://ieeexplore.ieee.org/document/9879198/

[182] T. Sipka, M. Sulc, and J. Matas, "The Hitchhiker's Guide to Prior-Shift Adaptation," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 2031–2039. [Online]. Available: https://ieeexplore.ieee.org/document/9706940/

[183] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

[184] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[185] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

[186] D. Rezende and S. Mohamed, "Variational Inference with Normalizing Flows," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 1530–1538, iSSN: 1938-7228. [Online]. Available: https://proceedings.mlr.press/v37/rezende15.html

[187] L. Dinh, D. Krueger, and Y. Bengio, "NICE: Non-linear Independent Components Estimation," Apr. 2015, arXiv:1410.8516 [cs]. [Online]. Available: http://arxiv.org/abs/1410.8516

[188] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, Nov. 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9555209/

[189] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-Free Generative Adversarial Networks," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 852–863. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/076ccd93ad68be51f23707988e934906-Abstract.html

[190] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Jan. 2016, arXiv:1511.06434 [cs]. [Online]. Available: http://arxiv.org/abs/1511.06434

[191] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling Tabular data using Conditional GAN," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/hash/254ed7d2de3b23ab10936522dd547b78-Abstract.html

[192] T. Aoshima and T. Matsubara, "Deep Curvilinear Editing: Commutative and Nonlinear Image Manipulation for Pretrained Deep Generative Model," Mar. 2023, arXiv:2211.14573 [cs]. [Online]. Available: http://arxiv.org/abs/2211.14573

[193] L. Jing, J. Zbontar, and y. lecun, "Implicit Rank-Minimizing Autoencoder," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 14 736–14 746. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/a9078e8653368c9c291ae2f8b74012e7-Abstract.html

[194] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[195] Z. Zhang, Y. Song, and H. Qi, "Age Progression/Regression by Conditional Adversarial Autoencoder," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 4352–4360, iSSN: 1063-6919.

[196] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS 2017 Autodiff Workshop*. NIPS 2017 Autodiff Workshop, 2017.

[197] S. I. Serengil and A. Ozpinar, "HyperExtended LightFace: A Facial Attribute Analysis Framework," in *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, Oct. 2021, pp. 1–4, iSSN: 2409-2983.

[198] Y. Mirsky and W. Lee, "The Creation and Detection of Deepfakes: A Survey," *ACM Computing Surveys*, vol. 54, no. 1, pp. 1–41, Jan. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3425780

[199] H. Chen, "Challenges and Corresponding Solutions of Generative Adversarial Networks (GANs): A Survey Study," *Journal of Physics: Conference Series*, vol. 1827, no. 1, p. 012066, Mar. 2021, publisher: IOP Publishing. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1827/1/012066

[200] U. Tanielian, T. Issenhuth, E. Dohmatob, and J. Mary, "Learning disconnected manifolds: a no GAN's land," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 9418–9427, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v119/tanielian20a.html

[201] M. Khayatkhoei, M. K. Singh, and A. Elgammal, "Disconnected Manifold Learning for Generative Adversarial Networks," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/hash/2b346a0aa375a07f5a90a344a61416c4-Abstract.html

[202] A. Salmona, V. D. Bortoli, J. Delon, and A. Desolneux, "Can Push-forward Generative Models Fit Multimodal Distributions?" in *Advances in Neural Information*

Processing Systems. Advances in Neural Information Processing Systems, May 2022. [Online]. Available: https://openreview.net/forum?id=Tsy9WCO_fK1

[203] M. Armandpour, A. Sadeghian, C. Li, and M. Zhou, "Partition-Guided GANs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, vol. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 5099–5109. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/html/Armandpour_Partition-Guided_GANs_CVPR_2021_paper.html

[204] P. Dendorfer, S. Elflein, and L. Leal-Taixe, "MG-GAN: A Multi-Generator Model Preventing Out-of-Distribution Samples in Pedestrian Trajectory Prediction," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 13 138–13 147. [Online]. Available: https://ieeexplore.ieee.org/document/9711160/

[205] S. Liu, T. Wang, D. Bau, J.-Y. Zhu, and A. Torralba, "Diverse Image Generation via Self-Conditioned GANs," Feb. 2022, arXiv:2006.10728 [cs]. [Online]. Available: http://arxiv.org/abs/2006.10728

[206] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "ClusterGAN: Latent Space Clustering in Generative Adversarial Networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 4610–4617, Jul. 2019, number: 01. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4385

[207] S. Gurumurthy, R. K. Sarvadevabhatla, and R. V. Babu, "DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 4941–4949. [Online]. Available: http://ieeexplore.ieee.org/document/8100008/

[208] O. Katzir, V. Perepelook, D. Lischinski, and D. Cohen-Or, "Multi-level Latent Space Structuring for Generative Control," Feb. 2022, arXiv:2202.05910 [cs]. [Online]. Available: http://arxiv.org/abs/2202.05910

[209] S. Azadi, C. Olsson, T. Darrell, I. Goodfellow, and A. Odena, "Discriminator Rejection Sampling," *ArXiv*, Sep. 2018. [Online]. Available: https://www.semanticscholar.org/paper/866aa9bcb15cf4a23a0afed515fa2f6b93f91d11

[210] T. Issenhuth, U. Tanielian, D. Picard, and J. Mary, "Latent reweighting, an almost free improvement for GANs," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2022, pp. 3574–3583. [Online]. Available: https://ieeexplore.ieee.org/document/9706934/

[211] R. Vershynin, "High-Dimensional Probability," *Cambridge University Press*, 2018.

[212] A. Rosenberg and J. Hirschberg, "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 410–420. [Online]. Available: https://aclanthology.org/D07-1043

[213] N. X. Vinh, J. Epps, and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *Journal of Machine Learning Research*, vol. 11, no. 95, pp. 2837–2854, 2010. [Online]. Available: http://jmlr.org/papers/v11/vinh10a.html

[214] S. Garg, Y. Wu, S. Balakrishnan, and Z. Lipton, "A Unified View of Label Shift Estimation," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 3290–

3300. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/219e052492f4008818b8adb6366c7ed6-Abstract.html

[215] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

A Appendix

A.1 Appendix for Task 3

A.1.1 Calculating E for The Generated Distribution

The error rates reported for a classifier C_ϕ are typically reported on the distribution on the distribution of fit's training data, $\mathbb{P}_{training}$. However, the distribution $\mathbb{P}_{G_\theta}$ of the generative model

G_θ may differ from the training distribution. Additionally, rather than reporting $P(\mathbf{y}|\hat{\mathbf{y}})$, often times the error rates are given in a confusion matrix $C_{\hat{\mathbf{y}}|\mathbf{y}}$ where $C_{\hat{\mathbf{y}}|\mathbf{y}}[i,j] = P(\hat{\mathbf{y}}|\mathbf{y})$. Thankfully, we can construct the error rate matrix E for the generative distribution $\mathbb{P}_{G_\theta}$ under the simplifying assumption that the difference between $\mathbb{P}_{G_\theta}$ and $\mathbb{P}_{training}$ can be explained as a label shift [214, 182].

By Bayes' Theorem, we know that

$$P(\mathbf{y}|\hat{\mathbf{y}}) = P(\hat{\mathbf{y}}|\mathbf{y})\frac{P(\mathbf{y})}{P(\hat{\mathbf{y}})}.$$

Under the label shift assumption, $P(\hat{\mathbf{y}}|\mathbf{y})$ stays the same between $\mathbb{P}_{training}$ and $\mathbb{P}_{G_\theta}$. Additionally, $P(\mathbf{y})$ can be calculated for $\mathbb{P}_{G_\theta}$ under label shift [214, 182]. Lastly, $P(\hat{\mathbf{y}})$ can be approximated for $\mathbb{P}_{G_\theta}$ by finding the proportion predicted for each class on a large sample from the generative model. Thus, E can be calculated as:

$$E = C_{\hat{\mathbf{y}}|\mathbf{y}}\frac{P_{G_\theta}(\mathbf{y})}{P_{G_\theta}(\hat{\mathbf{y}})}.$$

A.1.2 Distribution of Races Generated By Progressive GAN

We show the two best performing methods' distributions on Progressive GAN, along with the distribution of the unmodified ProgressiveGAN, over the `Race` attribute.

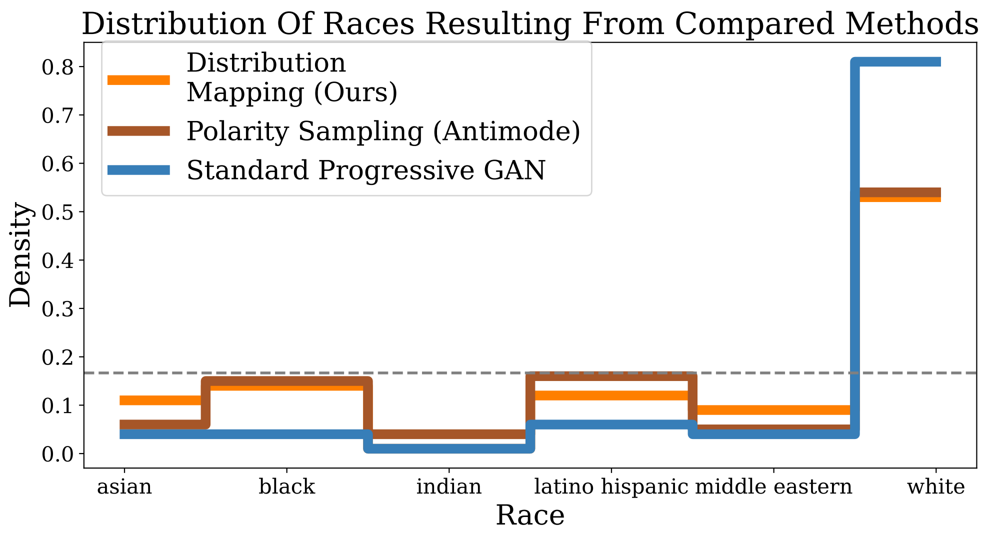


Figure 19: Distribution of our approach, Polarity Antimode Sampling (next best), and the standard generator.

A.1.3 Implementation Details

Ground Truth Shape Classifier

```
----------------------------------------------------------------
      Layer (type)               Output Shape
================================================================
            Conv2d-1        [-1, 32, 16, 16]
              ReLU-2        [-1, 32, 16, 16]
            Conv2d-3          [-1, 64, 8, 8]
              ReLU-4          [-1, 64, 8, 8]
            Conv2d-5         [-1, 128, 4, 4]
              ReLU-6         [-1, 128, 4, 4]
            Conv2d-7         [-1, 256, 2, 2]
              ReLU-8         [-1, 256, 2, 2]
            Conv2d-9           [-1, 2, 1, 1]
================================================================
```

Encoder for Shapes VAE

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
            Conv2d-1           [-1, 32, 16, 16]
              ReLU-2           [-1, 32, 16, 16]
            Conv2d-3             [-1, 64, 8, 8]
              ReLU-4             [-1, 64, 8, 8]
            Conv2d-5            [-1, 128, 4, 4]
              ReLU-6            [-1, 128, 4, 4]
            Conv2d-7            [-1, 256, 2, 2]
              ReLU-8            [-1, 256, 2, 2]
            Conv2d-9           [-1, code_dim, 1, 1]
================================================================
```

Decoder for Shapes VAE

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
   ConvTranspose2d-1            [-1, 256, 2, 2]
             ReLU-2            [-1, 256, 2, 2]
   ConvTranspose2d-3            [-1, 128, 8, 8]
             ReLU-4            [-1, 128, 8, 8]
   ConvTranspose2d-5          [-1, 64, 16, 16]
             ReLU-6          [-1, 64, 16, 16]
   ConvTranspose2d-7          [-1, 32, 32, 32]
             ReLU-8          [-1, 32, 32, 32]
   ConvTranspose2d-9           [-1, 3, 64, 64]
        Sigmoid-10           [-1, 3, 64, 64]
```

```
================================================================
```

Biased Age Classifier (Note: Target value was normalized age, made binary after)

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
            Conv2d-1            [-1, 2, 32, 32]
       BatchNorm2d-2            [-1, 2, 32, 32]
         LeakyReLU-3            [-1, 2, 32, 32]
           Dropout-4            [-1, 2, 32, 32]
            Conv2d-5            [-1, 4, 16, 16]
       BatchNorm2d-6            [-1, 4, 16, 16]
         LeakyReLU-7            [-1, 4, 16, 16]
           Dropout-8            [-1, 4, 16, 16]
            Conv2d-9              [-1, 8, 8, 8]
      BatchNorm2d-10              [-1, 8, 8, 8]
        LeakyReLU-11              [-1, 8, 8, 8]
          Dropout-12              [-1, 8, 8, 8]
          Flatten-13                  [-1, 512]
           Linear-14                   [-1, 64]
        LeakyReLU-15                   [-1, 64]
           Linear-16                    [-1, 1]
          Sigmoid-17                    [-1, 1]
================================================================
```

Ground Truth Age Classifier (Note: Target value was normalized age; made binary after)

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
            Conv2d-1            [-1, 8, 32, 32]
       BatchNorm2d-2            [-1, 8, 32, 32]
         LeakyReLU-3            [-1, 8, 32, 32]
           Dropout-4            [-1, 8, 32, 32]
            Conv2d-5           [-1, 16, 16, 16]
       BatchNorm2d-6           [-1, 16, 16, 16]
         LeakyReLU-7           [-1, 16, 16, 16]
           Dropout-8           [-1, 16, 16, 16]
            Conv2d-9             [-1, 32, 8, 8]
      BatchNorm2d-10             [-1, 32, 8, 8]
        LeakyReLU-11             [-1, 32, 8, 8]
          Dropout-12             [-1, 32, 8, 8]
          Flatten-13                 [-1, 2048]
           Linear-14                   [-1, 64]
        LeakyReLU-15                   [-1, 64]
           Linear-16                    [-1, 1]
          Sigmoid-17                    [-1, 1]
================================================================
```

The distribution mapper used default architecture of SDV's CTGAN [10] version 0.6.0, except for in the ProgressiveGAN experiment where `embedding_dim =512, generator_dim =(512,512)` were passed as arguments.

For the networks we trained, we utilized the Adam optimizer [215] with learning rate between 0.002 and 0.0001.

The linear classifier utilized Scikit-Learn's LinearSVC (for latent editing) and Ridge-Classifier for the biased Shapes classifier.

[10]https://sdv.dev/SDV/user_guides/single_table/ctgan.html#how-to-modify-the-ctgan-hyperparameters

A.1.4 Proof of Lemma 1

Proof. First, note that if $1^{|\mathcal{Y}|} \in Cone(E)$, then likewise $\frac{1}{\mathcal{Y}}1^{|\mathcal{Y}|} \in Cone(E)$.

Let $\mathbf{z}' \sim \mathbb{P}_{z|C_\phi=i}$; i.e., z is a draw from the distribution of noise such that the classifiers prediction of the generated sample corresponding to z' is group i.

Let $(C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i}$ be the pushforward distribution of the perfect classifier C''s output when conditioned on the generator's output of draws from $\mathbb{P}_{z|C_\phi=i}$. Then, $(C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i} = [Pr(\mathbf{y} = 1|C_\theta = i), Pr(\mathbf{y} = 2|C_\theta = i), \ldots, Pr(\mathbf{y} = N|C_\theta = i)] = E_{:,i}$. Thus, $Cone(\{(C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i}, \ldots, (C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=|\mathcal{Y}|}\}) = Cone(E)$. Therefor, following from above, $\frac{1}{\mathcal{Y}}1^{|\mathcal{Y}|} \in Cone(\{(C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i}, \ldots, (C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=|\mathcal{Y}|}\})$. This means that $\exists \lambda_1, \lambda_2, \ldots, \lambda_{|\mathcal{Y}}$ s.t. $\lambda_1 (C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i} + \cdots + \lambda_{|\mathcal{Y}|} (C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=|\mathcal{Y}|} = [\frac{1}{|\mathcal{Y}|}, \ldots, \frac{1}{|\mathcal{Y}|}] = Unif \mathcal{Y}$. This is equivalent to saying that $C'(G_\theta(\mathbf{z})) \sim Unif(\mathcal{Y})$ for $\mathbf{z} \sim \sum_{i=1}^{|\mathcal{Y}|} \lambda_i \mathbb{P}_{z|C_\phi=i} = \mathbb{Q}^\lambda$. Thus, by definition $\mathbb{Q}^\lambda$ is a Fair Noise Distribution.

□

A.1.5 Proof of Lemma 2

Proof. Note that the sign of the coefficient of the cross product $E_{:,1} \times E_{:,2}$ is $P(\mathbf{y} = 1|\hat{\mathbf{y}} = 1)P(\mathbf{y} = 2|\hat{\mathbf{y}} = 2) - P(\mathbf{y} = 1|\hat{\mathbf{y}} = 2)P(\mathbf{y} = 2|\hat{\mathbf{y}} = 1)$. Also note that $E_{:,1} \times [0.5, 0.5]$ is $0.5P(\mathbf{y} = 1|\hat{\mathbf{y}} = 1) - 0.5P(\mathbf{y} = 2|\hat{\mathbf{y}} = 1)$.

Additionally, $P(\mathbf{y} = 1|\hat{\mathbf{y}} = 1)P(\mathbf{y} = 2|\hat{\mathbf{y}} = 2) > P(\mathbf{y} = 1|\hat{\mathbf{y}} = 1)0.5 > 0$, and $0 < P(\mathbf{y} = 1|\hat{\mathbf{y}} = 2)P(\mathbf{y} = 2|\hat{\mathbf{y}} = 1) < 0.5P(\mathbf{y} = 2|\hat{\mathbf{y}} = 1)$. Thus, the coefficient of $E_{:,1} \times E_{:,2}$ is greater than $E_{:,1} \times [0.5, 0.5]$, while there signs are equal. This implies that $[0.5, 0.5]$ is in between $E_{:,1}$ and $E_{:,2}$. Thus, $[0.5, 0.5] \in cone(E)$. The rest of the proof follows directly from Lemma 1. □

A.1.6 Proof of Proposition 1

Proof. Note that $\mathbb{P}_E^\lambda$ has density $[\sum_i \lambda_i Pr(\mathbf{y} = 1|\hat{\mathbf{y}} = i), \ldots, \sum_i \lambda_i Pr(\mathbf{y} = N|\hat{\mathbf{y}} = i)]$. For ease of notation let us refer to $\sum_i \lambda_i Pr(\mathbf{y} = m|\hat{\mathbf{y}} = i)$ as r_m^λ.

Then,

$$\begin{aligned}
KL\{\mathbb{P}_E^\lambda || Unif(\mathcal{Y})\} &= \sum_m r_m^\lambda \log\big(\frac{r_m^\lambda}{u}\big) \\
&= \sum_m \Big(r_m^\lambda \log(r_m^\lambda) - r_m^\lambda \log(\frac{1}{|\mathcal{Y}|})\Big) \\
&= \sum_m r_m^\lambda \log(r_m^\lambda) - \sum_m r_m^\lambda \log(\frac{1}{|\mathcal{Y}|})
\end{aligned}$$

Note that $\log\big(\frac{1}{N}\big)$ is constant for each term in the second summation. Thus,

$$\begin{aligned}
&= \sum_m r_m^\lambda \log(r_m^\lambda) - \log\big(\frac{1}{N}\big)\sum r_m^\lambda \\
&= \sum_m r_m^\lambda \log(r_m^\lambda) - \log\big(\frac{1}{N}\big),
\end{aligned}$$

As $\log\big(\frac{1}{N}\big)$ does not depend on r_m^λ,

$$\begin{aligned}
\operatorname*{argmin}_\lambda KL\{\mathbb{P}_E^\lambda || Unif(\mathcal{Y})\} &= \operatorname*{argmin}_\lambda \sum_m r_m^\lambda \log(r_m^\lambda) \\
&= \operatorname*{argmin}_\lambda -H(\mathbb{P}_E^\lambda) \\
&= \operatorname*{argmax}_\lambda H(\mathbb{P}_E^\lambda)
\end{aligned}$$

□

A.1.7 Proof of Proposition 2

Proof.

$$
\begin{aligned}
(C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i} =& [Pr(\mathbf{y}=1|C_\theta=i), Pr(\mathbf{y}=2|C_\theta=i), \ldots, Pr(\mathbf{y}=N|C_\theta=i)] \\
\implies \sum_i \lambda_i (C' \circ G_\theta)_* \mathbb{P}_{z|C_\phi=i} =& [\sum_i \lambda Pr(\mathbf{y}=1|C_\theta=i), \sum_i Pr(\mathbf{y}=2|C_\theta=i), \\
& \ldots, \sum_i \lambda Pr(\mathbf{y}=N|C_\theta=i)] \\
\implies \mathbb{P}_E = \mathbb{Q}^\lambda &
\end{aligned}
$$

□

A.1.8 Proof of Theorem 1

The first statement follows directly from Proposition 1 and Proposition 2.

If $C_\phi = C'$, then $\{\frac{E_{:,1}}{|E_{:,1}|}, \ldots, \frac{E_{:,|\mathcal{Y}|}}{|E_{:,|\mathcal{Y}|}|}\}$ forms a standard basis of $\mathbb{R}^{|\mathcal{Y}|}$, and therefor $1^{|\mathcal{Y}|}$ is is in $Cone(E)$. Thus, $\mathcal{Q}^{\lambda *}$ is a Fair Noise Distribution by Lemma 1.

A.2 Appendix For Task 4

A.2.1 Hyperparameters

Polarity Sampling. Polarity Sampling is determined using a parameter ρ where as ρ goes to $-\infty$ modes are sampled from increasingly often, and antimodes are sampled from more as $\rho \to \infty$, with the original generative distribution corresponding to $\rho = 0$. As we aim to sample from the on-manifold region of the output space, we set ρ to a negative value such as to sample from modes rather than low-probability regions. Specifically, we set $\rho = -0.1$.

Training Details. For MNIST and Fashion MNIST, we trained each network for 50 epoch with a batch size of 64. For Celeba + Flowers we trained for 50 epochs with a batch size of 16. For all datasets and models, we used the Adam optimizer [215] with learning rate set to 0.001 and betas=(0.5, 0.999). All models were implemented in PyTorch [196].

A.2.2 Network Structures

For MNIST and FashionMNIST, we utilized the following network structures:

Generator

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
   ConvTranspose2d-1          [-1, 256, 4, 4]
       BatchNorm2d-2          [-1, 256, 4, 4]
              ReLU-3          [-1, 256, 4, 4]
   ConvTranspose2d-4          [-1, 128, 8, 8]
       BatchNorm2d-5          [-1, 128, 8, 8]
              ReLU-6          [-1, 128, 8, 8]
   ConvTranspose2d-7           [-1, 64, 16, 16]
       BatchNorm2d-8           [-1, 64, 16, 16]
              ReLU-9           [-1, 64, 16, 16]
  ConvTranspose2d-10            [-1, 1, 32, 32]
             Tanh-11            [-1, 1, 32, 32]
================================================================
```

Discriminator

```
----------------------------------------------------------------
        Layer (type)               Output Shape
================================================================
            Conv2d-1           [-1, 64, 16, 16]
         LeakyReLU-2           [-1, 64, 16, 16]
```

```
            Conv2d-3          [-1, 128, 8, 8]
       BatchNorm2d-4          [-1, 128, 8, 8]
         LeakyReLU-5          [-1, 128, 8, 8]
            Conv2d-6          [-1, 256, 4, 4]
       BatchNorm2d-7          [-1, 256, 4, 4]
         LeakyReLU-8          [-1, 256, 4, 4]
            Conv2d-9            [-1, 1, 1, 1]
          Sigmoid-10            [-1, 1, 1, 1]
================================================================
```

Classifier (For Evaluation)

```
----------------------------------------------------------------
      Layer (type)               Output Shape
================================================================
            Conv2d-1          [-1, 16, 28, 28]
              ReLU-2          [-1, 16, 28, 28]
         MaxPool2d-3          [-1, 16, 14, 14]
            Conv2d-4          [-1, 32, 14, 14]
              ReLU-5          [-1, 32, 14, 14]
         MaxPool2d-6            [-1, 32, 7, 7]
            Linear-7                  [-1, 10]
================================================================
```

For CelebA + Flowers we utilized the following networks:

Generator

```
----------------------------------------------------------------
      Layer (type)               Output Shape
```

```
================================================================
   ConvTranspose2d-1         [-1, 512, 4, 4]
       BatchNorm2d-2         [-1, 512, 4, 4]
              ReLU-3         [-1, 512, 4, 4]
   ConvTranspose2d-4         [-1, 256, 8, 8]
       BatchNorm2d-5         [-1, 256, 8, 8]
              ReLU-6         [-1, 256, 8, 8]
   ConvTranspose2d-7       [-1, 128, 16, 16]
       BatchNorm2d-8       [-1, 128, 16, 16]
              ReLU-9       [-1, 128, 16, 16]
  ConvTranspose2d-10        [-1, 64, 32, 32]
      BatchNorm2d-11        [-1, 64, 32, 32]
             ReLU-12        [-1, 64, 32, 32]
  ConvTranspose2d-13         [-1, 3, 64, 64]
             Tanh-14         [-1, 3, 64, 64]
================================================================
```

Discriminator

```
----------------------------------------------------------------
      Layer (type)               Output Shape
================================================================
            Conv2d-1          [-1, 64, 32, 32]
         LeakyReLU-2          [-1, 64, 32, 32]
            Conv2d-3         [-1, 128, 16, 16]
         LeakyReLU-4         [-1, 128, 16, 16]
       BatchNorm2d-5         [-1, 128, 16, 16]
            Conv2d-6           [-1, 256, 8, 8]
         LeakyReLU-7           [-1, 256, 8, 8]
       BatchNorm2d-8           [-1, 256, 8, 8]
```

```
            Conv2d-9           [-1, 512, 4, 4]
        LeakyReLU-10           [-1, 512, 4, 4]
      BatchNorm2d-11           [-1, 512, 4, 4]
           Conv2d-12             [-1, 1, 1, 1]
          Sigmoid-13             [-1, 1, 1, 1]
================================================================
```

Classifier (For Evaluation)

```
----------------------------------------------------------------
      Layer (type)               Output Shape
================================================================
            Conv2d-1          [-1, 64, 32, 32]
         LeakyReLU-2          [-1, 64, 32, 32]
            Conv2d-3         [-1, 128, 16, 16]
         LeakyReLU-4         [-1, 128, 16, 16]
       BatchNorm2d-5         [-1, 128, 16, 16]
            Conv2d-6           [-1, 256, 8, 8]
         LeakyReLU-7           [-1, 256, 8, 8]
       BatchNorm2d-8           [-1, 256, 8, 8]
            Conv2d-9           [-1, 512, 4, 4]
        LeakyReLU-10           [-1, 512, 4, 4]
      BatchNorm2d-11           [-1, 512, 4, 4]
           Conv2d-12             [-1, 1, 1, 1]
          Sigmoid-13             [-1, 1, 1, 1]
================================================================
```

A.2.3 Proof of Lemma 4

Proof. Assume that G is not k-ρ-*disconnected*; then, $\mathbb{G}_{\setminus\rho}$ consists of h distinct sub-manifolds, where $h < k$ or $h > k$. If $h < k$, then there was a sub-manifold in the data space with region for which G did not place a density of at least ρ. However, this is not possible as it breaks the combination of Assumption 2 and Assumption 3. If $h > k$, then there is a subset outside of the support of p_{x} where G's density is greater than ρ. This also breaks Assumption 2. Thus, Lemma 4 is proven by contradiction. $\square$

A.2.4 Proof of Theorem 6

Proof. By Lemma 1 we know that $\mathbb{G}_{\setminus\rho}$ consists of k distinct sub-manifolds. Since by Assumption 1 G is a continuous injective function, it's domain must also consist of k distinct sub-manifolds. $\square$

A.2.5 Proof of Lemma 5

Proof. Due to G being a continuous function, each $\mathbb{V}_i$ must be mapped to a single sub-manifold in the output space. Further, each of the k output sub-manifolds requires there to be at least one sub-manifold in the latent space that is mapped to it. Since there are k sub-manifolds in each space, there must be a one-to-one correspondence between them. $\square$