ABSTRACT

Title of Dissertation: FOUNDATIONS OF
TRUSTWORTHY DEEP LEARNING:
FAIRNESS, ROBUSTNESS, AND EXPLAINABILITY

Vedant Nanda
Doctor of Philosophy, 2024

Dissertation Directed by: Professor John P. Dickerson
Department of Computer Science

Deep Learning (DL) models, especially with the rise of the so-called foundation models, are increasingly used in real-world applications either as autonomous systems (*e.g.* facial recognition), as decision aids (*e.g.* medical imaging, writing assistants), and even to generate novel content (*e.g.* chatbots, image generators). This naturally results in concerns about the trustworthiness of these systems, for example, do the models systematically perform worse for certain subgroups? Are the outputs of these models reliable under perturbations to the inputs? This thesis aims to strengthen the foundations of DL models, so they can be trusted in deployment. I will cover three important aspects of trust: fairness, robustness, and explainability. I will argue that we need to expand the scope of each of these aspects when applying them to DL models and carefully consider possible tradeoffs between these desirable but sometimes conflicting notions of trust.

Traditionally the fairness community has worked on mitigating biases in classical models such as Support Vector Machines (SVMs) and logistic regression. However, a lot of real-world

applications where bias shows up in a myriad of ways involve much more complicated DL models. In the first part, I will present two works that show how thinking about fairness for deep learning (DL) introduces new challenges, especially due to their overparametrized nature and susceptibility to adversarial attacks.

Robustness literature has focused largely on measuring the invariance of models to carefully constructed (adversarial attacks) or natural (distribution shifts) noise. In the second part, I will argue that to get truly robust models, we must focus on a more general notion of robustness: measuring the alignment of invariances of DL models with other models of perception such as humans. I will present two works that measure shared invariances between (1) DL models and humans, and (2) between DL models. Such measurements of robustness provide a measure of *relative robustness*, through which we can better understand the failure modes of DL models and work towards building truly robust systems.

Finally, in the third part, I will show how even a small subset of randomly chosen neurons from a pre-trained representation can transfer very well to downstream tasks. We call this phenomenon *diffused redundancy*, which we observe in a variety of pre-trained representations. This finding challenges existing beliefs in the explainability literature that claim individual neurons learn disjoint semantically meaningful concepts.

FOUNDATIONS OF TRUSTWORTHY DEEP LEARNING: FAIRNESS, ROBUSTNESS, AND EXPLAINABILITY

by

Vedant Nanda

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2024

Advisory Committee:
Professor John P. Dickerson, Chair/Advisor
Professor Krishna P. Gummadi, Co-Advisor
Professor Tom Goldstein
Professor Soheil Feizi
Professor Hernisa Kacorri

Dedication

Dedicated to my mom.

Acknowledgments

I would first like to thank my advisors, John and Krishna, for their unwavering support throughout my PhD. They believed in me through many paper rejections and encouraged me to keep pursuing crazy ideas. From John, I learned many technical and interpersonal skills, but one thing I cherish the most is how much he truly cares about his students. He was always approachable, for both research and life advice and never discouraged me from pursuing even the most far-fetched research ideas. He was present for all talks I gave during my PhD. This meant a lot to me, especially as an early-stage PhD student. I hope to be a mentor as empathetic as John to my future mentees. As an undergrad, I interned in Krishna's group and he was the person who convinced me to pursue a PhD. I owe a great deal of gratitude to Krishna for believing in me and admitting me to this unique PhD program between the University of Maryland and MPI-SWS. He taught me the importance of thinking clearly and asking the simple questions. I will always cherish our long discussions and debates which made me a better thinker and researcher.

I would like to thank my committee for their time and feedback on this thesis. I met Soheil Feizi and Tom Goldstein during my time at UMD and have had the pleasure of collaborating on a few papers early in my PhD with them. Brainstorming research ideas with them has been very useful in shaping how I see the field and has greatly influenced how I pick problems. I also thank Hernia Kacorri for agreeing to be the dean's representative for my thesis.

I was very fortunate to have collaborated with some amazing people during my PhD. I

worked on my first PhD project with Muhammad Bilal Zafar with whom I later interned for my first industry internship at AWS. I learned a great deal through our collaborations and his wisdom was especially useful in navigating a tough job market. In the third year of my PhD I met Adrian Weller and it has been a great pleasure to collaborate with him. I will always admire his directness and his ability to give concise but useful feedback. Through Adrian, I met Bradley Love whose inputs on our AAAI paper were invaluable and I learned a great deal about human perception from him. I would like to thank Mariya Toneva for serving on my area exam committee and allowing me to collaborate with her group. I am also grateful to all the student co-authors from papers that form this thesis, including Till Speicher, Samuel Dooley, Valeriia Cherepanova, Camila Kolling, Ayan Majumdar. I also enjoyed learning from other collaborations with Seyed Esmaeili, Pan Xu, Sharmila Duppala, Gabriele Merlin, Ruchit Rawal, Qinyuan Wu, Soumi Das, and other members of Socsys and Dickerson lab. I owe a great deal to the staff at both MPI-SWS and UMD, especially the office for helping with all the logistics and IT for always being proactive in making sure our cluster kept running, and for dealing with my constant requests for more GPUs.

PhD takes an emotional toll and it does take a village to keep going. Luckily, I had a great set of friends that made my journey quite enjoyable. You know who you are and I'm grateful to have you in my life. During my PhD, I met Melis Ilayda Bal whose presence makes the tough times easier.

Finally, this would not be possible without the support of my mom and my sister. We have been through all hardships together. My mom made sure I had all the right conditions to perform to the best of my abilities and always supported me in my endeavors (even as crazy as going for a PhD). I hope that by being the first person in my family to get a PhD, I can make them proud.

Table of Contents

Bibliography 191

List of Tables

List of Figures

Chapter 1: Introduction

My thesis touches on three pillars of trustworthy deep learning: fairness, robustness, and explainability, and the complicated interplay between these desirable, but sometimes conflicting goals. While these subfields have existed almost mutually exclusively for a few years now, my thesis introduces some of the first works that have proposed connections between these fields, identifying exciting open areas of research. The thesis is structured in three main parts, Chapters 2 & 3 focus on fairness, Chapters 4 & 5 on robustness, and Chapter 6 on explainability. This thesis is based on five first-authored papers I published during my PhD, two of which were co-first authored with my peers.

1.1 Fairness: Technical Challenges for Deep Learning

Technical Challenges for Training Fair Neural Networks; published at Responsible AI workshop at ICLR 2021. Joint first-author with Valeriia Cherepanova [12].

The fairness community has proposed many solutions to mitigate unfair behaviors by proposing constraints to the usual objective function during training. These are called in-processing mitigation techniques. Almost all in-processing techniques have been tested exclusively on classical ML models such as logistic regression, support vector machines (SVMs), and shallow multi-later perceptrons. In these cases, one can introduce affine constraints that model some

notion of fairness on the objective function of these models and then solve a constrained optimization problem in closed form, since these models are often convex. Here, we test fairness constraints for deep neural networks (DNNs) that are highly overparametrized and non-convex. This offers two challenges: firstly, it's not clear if constraints can work well for DNNs since the guarantees for constrained convex optimization with traditional models do not exist for highly non-convex models like DNNs. Secondly, even if fairness constraints worked, due to their highly fluid decision boundaries, DNNs are likely to be more susceptible to second-order effects, such as becoming fair along one subgroup at the cost of fairness along another subgroup, a phenomenon commonly known as *fairness gerrymandering*. In this work, we empirically investigate these questions and find that it is indeed challenging to get fairness constraints to generalize in deep learning. Training loss goes to zero and thus it seems like fairness has been achieved, however, this does not generalize to the test set. We also find that in rare cases where we do achieve fairness on the test set, it comes at a cost of increased unfairness alone another attribute, thus showing the susceptibility of DNNs to fairness gerrymandering. We cover this in more detail in Chapter 2.

1.2 Fairness: New Notions for Deep Learning

Fairness Through Robustness: Investigating Robustness Disparity in Deep Learning; published at FAccT 2021. Joint-first author with Samuel Dooley [13].

However, if fairness constraints worked in Chapter 2, would that be enough to maintain unbiased outcomes? In Chapter 3, we extend our analysis of challenges for fairness in deep learning by arguing that the susceptibility of DNNs to adversarial attacks makes fairness challenging in deep learning. Most fairness notions are one-shot, in that they only look at the outputs of the

model at a given snapshot in time. This becomes especially limiting in adversarial settings where data points can be misclassified by adding minimal noise to the inputs. We introduce the notion of *robustness bias* that takes into account how disparate levels of robustness in subpopulations can create an opportunity for downstream unfairness. Robustness bias measures how robust are various data points in the subgroup under a given perturbation budget. We then propose metrics to quantify this bias where we consider points farther away from the decision boundary to be more robust. We show how this form of bias can manifest itself in existing deep learning models in unintuitive ways. We show that robustness bias is a complicated property of the data distribution, the model's inductive bias, and its pretraining, and is thus an important metric to consider when auditing deep learning models in safety-critical applications.

1.3 Robustness: Shared Invariances between Machines and Humans

Do Invariances in Deep Neural Networks Align with Human Perception?; published at AAAI 2023 [14].

There is a large body of work studying robustness in DL models. Most of these works propose some class of perturbations and evaluate whether DL models remain invariant to these perturbations. Some common examples of perturbations include carefully crafted adversarial samples, which are often found by imposing a budget on the amount of perturbation and then solving a constrained optimization in the input space to maximize the loss. Another line of robustness research looks more naturally occurring perturbations, *e.g.* adding weather-related abnormalities like snow to an otherwise "clean" image. After obtaining these clean and corresponding corrupted inputs, robustness is defined as the difference in some metric (*e.g.* accuracy) on the

clean and perturbed test set. In other words, the goal of robustness research has been to measure and improve the *invariance* of DL models to "meaningless" perturbations. A key, but implicit assumption in these works is that the definition of "meaningless" is defined relative to human perception. That is, the understanding is that adding some small noise (natural or otherwise) doesn't change human perception of the input and thus the models should also not change their output when presented with this noise. In Chapter 4, we take a step back and ask the robustness question in the opposite direction. We evaluate whether invariances learned by DL models are reasonable invariances for humans. This is a more general measure of robustness and goes beyond the existing perturbation-based robustness benchmarks. We present a way to scalably and reliably perform such an evaluation. Using our evaluation pipeline we extensively study how different DL pipeline components (loss function, architecture, data augmentation, training paradigm) contribute to learning well-aligned invariances.

1.4 Robustness: Shared Invariances between Machines

Measuring Representational Robustness of Neural Networks Through Shared Invariances; published at ICML 2022 [15].

Building on work in Chapter 4, in Chapter 5 we remove human perception from the robustness evaluation pipeline and propose a measure of the robustness of a DNN relative to another DNN. Prior works on comparing DNNs use measures of representation similarity. These measures quantify the similarity of representations on a given dataset. However, these measures do not tell us how this similarity changes outside the given data. In this Chapter, we ask the "robustness version" of the representation similarity question: how similarly will two representations respond

to perturbations of a given set of inputs? This essentially boils down to measuring shared invariances between DNNs. We propose a measure that can reliably measure such shared invariance between any desired representation of DNNs. We call this measure STIR – Similarity Through Inverted Representations and use this measure to uncover many intriguing properties of representations learned by DNNs. For example, we find that two DNNs that only differ in their random initializations, while having very similar test accuracies, actually have only moderate levels of shared invariance, showing that they can differ massively in their predictions outside the given data distribution. We also analyze the effects of architectures, loss functions, training datasets, and layer depth on shared invariances. More broadly, we present STIR as a tool to better understand deep learning through the lens of robustness and to reliably audit DL-based systems for their behaviors when presented with data that does not resemble their training distribution.

1.5 Explainability: Diffused Redundancy in Representations

Diffused Redundancy in Pre-trained Representations; published at NeurIPS 2023 [16].

In Chapter 6, we look deeper into the nature of learned representations. With an increasing shift towards so-called foundation models, the standard paradigm to solve any downstream task (*e.g.* image classification) is to first get the representation of the inputs of the desired task from a pre-trained model (often trained on large-scale data with unsupervised / weakly supervised learning) and then either finetune this model or train a lightweight model on top of this representation for the downstream task. Increased reliance on pre-trained representations makes understanding the nature of these representations essential to solving any downstream task reliably. Many works in the explainability community try to understand pre-trained representations through a

mechanistic lens. Such works have shown how certain portions of DNNs learn semantically meaningful concepts that then interact to form complex abstractions of the input, that are ultimately useful in solving a variety of downstream tasks. For example, prior work has successfully isolated neurons that are almost always highly activated when presented with some concept (*e.g.* stripes) and has found *circuits* through which these individual neurons interact to form more complicated neurons. These works suggest that different parts of DNNs encode distinct information.

In our work, we uncover a phenomenon we call *diffused redundancy* which shows that even a small randomly chosen subset of neurons from a pre-trained representation can achieve comparable performance as the entire representation on a range of downstream tasks. This shows that relevant features are smeared all over the representation, contrary to popular belief about distinct parts learning distinct features. We show that random samples of neurons capture similar variance in downstream data as projections onto the top principal components and are thus effective at solving these downstream tasks. While using random subsets of neurons could be an effective strategy for efficient transfer learning, we draw caution to how this can lead to systematically lower accuracies for certain subgroups, leading to possible fairness issues. More broadly, our work challenges existing beliefs about the nature of learned representations and shows that there is room to grow our fundamental understanding of the kind of features that make pre-trained representations very effective at solving a variety of downstream tasks.

Chapter 2: Fairness: Technical Challenges for Deep Learning

> Experience is what you get when
> you didn't get what you wanted.
>
> *Randy Pausch*

As machine learning algorithms have been widely deployed across applications, many concerns have been raised over the fairness of their predictions, especially in high-stakes settings, such as facial recognition and medical imaging. To respond to these concerns, the community has proposed and formalized various notions of fairness as well as methods for rectifying unfair behavior. While fairness constraints have been studied extensively for classical machine learning models (SVMs, Logistic Regression), the effectiveness of methods for imposing fairness on deep neural networks is unclear. In the first part of this chapter, we observe that these large models overfit to fairness objectives, and produce a range of unintended and undesirable consequences. We conduct our experiments on both facial recognition and automated medical diagnosis datasets using state-of-the-art architectures.

2.1 Introduction

Machine learning systems are increasingly deployed in settings with major economic and social impacts. In such situations, differences in model behaviors between different social groups

may result in disparities in how these groups are treated [17, 18]. For this reason, it is crucial to understand the bias of machine learning systems, and to develop tools that prevent algorithmic discrimination against protected groups.

Much effort has been devoted to understanding and correcting biases in classical machine learning models (*e.g.* SVMs, Logistic Regression *etc.*). Overfitting is not a pernicious issue for classical models, and so fairness constraints that are imposed at train time often generalize to (unseen) test-time data. For overparameterized neural networks – as is often the case with modern deep neural networks [19] – our tools for understanding and controlling model bias are far less effective, in large part because of the difficulties created by overfitting. At train time, neural networks interpolate the data and achieve perfect accuracy on all sub-groups, thus making it impossible to obtain meaningful measures of bias on training data. When constraints are imposed using a sequestered dataset, the network may still overfit to constraints. Furthermore, the extremely fluid decision boundaries of neural networks open the possibility for complex forms of *fairness gerrymandering* [20, 21], in which the decision boundaries of the model are moved to achieve a fairness constraint, while at the same time creating unintended consequences for important sub-groups that were not explicitly considered at train time. Since deep neural networks perform so much better than their linear counterparts on a wide range of tasks, it is important that we better understand the fairness properties of these complex systems.

This part of the thesis investigates whether currently available methods for training fair models are capable of controlling bias in Deep Neural Networks (DNNs). We find that train-time fairness interventions – including those that have been thoroughly tested for classical ML models – are not effective for DNNs. We further show that when these fairness interventions do seem to work, they often result in the undesired phenomenon of fairness gerrymandering. Our

contributions are as follows:

- We empirically test a range of existing methods for imposing fairness using constraints and penalties during training of DNNs. While methods of this type have been widely used and rigorously studied in the under-fitting regime (*i.e.* SVMs and linear models), we show that they fail in the overparameterized regime.

- We find that in some cases, fairness surrogates that work well for classical models do not work well for DNNs. In particular, equality of losses does not necessarily translate to equality of metrics used to evaluate performance (*e.g.* area under the curve).

- We also observe that specialized constraints designed for facial recognition often appear to work on training data, but fail on holdout classes. Facial recognition systems present a unique case because they operate differently during inference than during training.

- We consider adversarial methods for learning "fair features". In addition to discussing theoretical problems with these approaches, we observe that these methods are not effective at achieving fair outcomes in practice.

- We observe that fairness gerrymandering can be particularly problematic for DNNs because of their highly flexible decision boundaries. That is, parity along one sensitive attribute (*e.g.* sex) comes at the cost of increased disparity along another sensitive attribute (*e.g.* age).

We acknowledge that fairness constraints are not a universal solution to fair ML (also argued in many prior works, *e.g.*, see Chapter 3 of [22]). Indeed, ML algorithms are only a small part of the bigger automated decision-making system, and making these algorithms "fair"

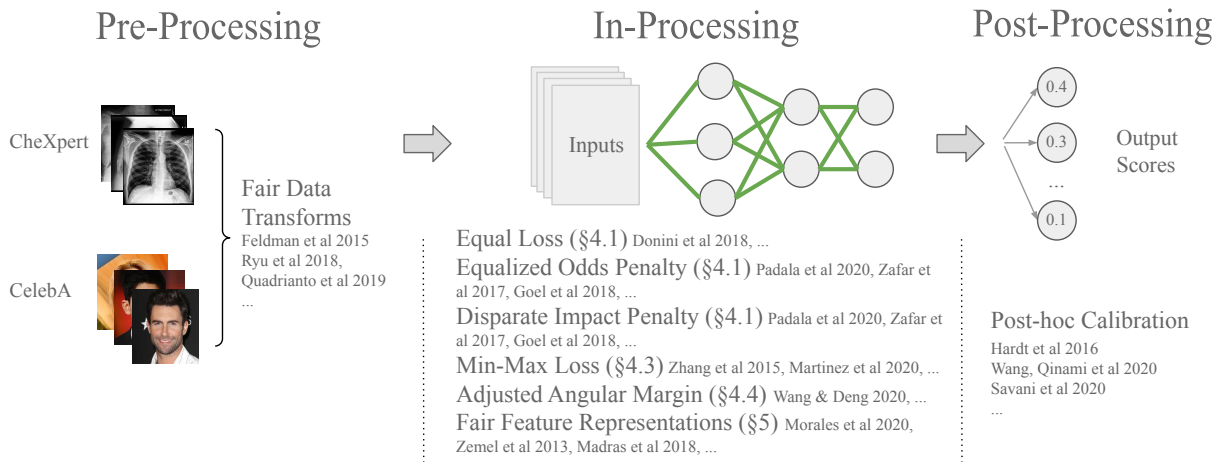


Figure 2.1: **[Brief Overview of Fairness in ML]** For the scope of this paper, we consider only in-processing techniques and apply them to deep neural networks. We show that the overparametrized nature of neural networks is one reason why current techniques fail.

is only solving a small part of what is a larger socio-technical problem. However, it is important to understand the limitations of algorithmic fairness solutions. In our work, we focus on analyzing the effectiveness of bias mitigation strategies in deep neural networks, and discuss the associated pitfalls.

2.2 Related work

There exist well documented cases of unfairness in key ML applications such as targeted ads [23–25], personalized recommendation systems [26,27], credit scoring [28], recidivism prediction [29], hiring [30], medical diagnosis [31,32], facial recognition [5,33,34], and others. This has resulted in a range of interdisciplinary work on understanding and mitigating bias in automated decision-making systems [13,35–39]. Existing work on mitigating algorithmic unfairness can be broadly put into three categories: pre-processing, in-processing and post-processing (see Figure 2.1). Works on pre-processing mostly focus on removing sensitive information from the data and building diverse and balanced datasets [40–43]. In-processing aims to change the training routine,

often via imposing constraints [43–50] or by changing the optimization routine [38, 51, 52]. Another in-processing strategy is to learn fair intermediate representations independent of sensitive attributes which lead to fairness on any downstream task [53–58]. There isn't a clear consensus on whether works along the lines of fair representation learning are pre-processing or in-processing (Zafar et al [46] categorize it as both in and pre). However, since most of these works rely on a learned model to perform the transformation of the dataset into a "fair" representation, for our purpose we consider these as in-processing. Post-processing techniques aim to change the inference mechanism to ensure fair outcomes [59–61]. In this paper, we limit our scope to understanding how in-processing techniques work for DNNs.

Our work is closely inspired by the works of Zafar et al [44] that proposed smooth surrogates for various statistical notions of fairness which could then be imposed as constraints on training. However, their proposed method was only evaluated on linear models (such as logistic regression and SVMs) where an optimal solution can be efficiently found using constrained optimization. Padala et al [49] extended this line of work and proposed the use of Neural Networks to empirically estimate measures of fairness by considering class logits to be proxies for class probabilities. They then applied these estimates of fairness as constraints to the training of neural networks using lagrangian multipliers. However, they only performed experiments on fully connected and shallow neural networks. We use their method of empirically estimating fairness measures and applying it as regularizers for training deep convolutional neural networks on image classification tasks. We find that their proposed approach does not give good fairness generalization in the highly overparametrized regime, such as the one we consider with our setup.

Most work on achieving fairness via in-processing methods in deep learning has been very tailored to a particular task like facial recognition [43] or achieves a very specific kind of fairness,

such as removing sensitive attribute information from a representation [62]. In addition to these tailored approaches, we also apply the more general approach of enforcing various definitions of fairness via regularization during training, which until now had been primarily tested on either linear models or very shallow deep neural networks.

Prior work has also suggested that algorithmic solutions for fairness are often hard to comprehend [63] or put into practice [64]. Additionally, industry practitioners believe that fairness issues in real-world ML systems could be more effectively tackled by systematically changing the broader system design, rather than solely focusing on algorithmic "debiasing" [65,66]. Our work aims to show the fragility of algorithmic fairness interventions for deep models, thus extending the scholarship on challenges for fairness in ML. We refer readers to [22] for a broad and nuanced discussion of the abilities and limitations of fair machine learning.

2.3 Investigating Fairness Constraints For Deep Learning

In this chapter, we investigate how different in-processing methods for mitigating algorithmic unfairness work on two different problems: facial recognition and medical image classification. We choose these two particular domains to illustrate broadly applicable pitfalls when applying train-time fairness interventions in deep learning. Moreover, while both facial recognition and medical image classification have seen unprecedented performance gains due to advances in deep learning, there have been well-documented cases of bias and unfairness in both of these domains [5,31–33]. Additionally, previous works have also outlined ethical and epistemic issues with facial recognition and algorithmic solutions to fairness in healthcare [67–69]. In this section, we describe our experimental setup which we use throughout the paper.

2.3.1 Face recognition

State-of-the-art facial recognition (FR) systems contain deep neural networks that extract facial features from **probe images** (new photos whose subject is identified by FR) and compare the resulting features to those corresponding to **gallery images** (references with known identities). Probe images are matched to the gallery image whose features lie closest with respect to the cosine similarity.

We train facial recognition models with ResNet-18 and ResNet-152 backbones using the popular CosFace head, designed to increase angular margins between identities [70]. All of our models are trained as classifiers using focal loss [71]. We split the Celeb-A dataset into train (9,177 identities) and test (1,000 identities) sets with non-overlapping identities. Therefore, at test time, the FR model is evaluated on people whose images it has not seen during training. We split a validation set from the training data consisting of 3 images from each identity with more than 6 images.

We measure the performance of FR systems with two methods. On training data we can use the classification head from training to report multi-label classification accuracy, while for validation and test sets, we rip off the classification head and report **rank-1** nearest neighbor (in feature space) accuracy as is mainstream in the facial recognition literature. We find that standard facial recognition models exhibit lower testing accuracy for females than for males, see Table 2.3. This accuracy gap does not result from unbalanced train data; in fact, female identities have more gallery images on average than male identities and 58% of identities in the data are female. Note that validation accuracy is lower than test accuracy in all of our experiments since the validation set contains 9 times as many classes and fewer images per class compared to the test set.

2.3.2 Medical image classification

We use CheXpert [72], a widely used and publicly available benchmark dataset for chest X-ray classification. This dataset consists of 224,316 chest radiographs annotated with a binary label indicating the presence of a given pathology. We consider the following 5 pathologies: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE). This yields a *multi-label* classification task, predicting which of the 5 pathologies are present given a chest x-ray image. We train models using weighted binary cross entropy loss, and we report performance via area under the curve (AUC) for each of the 5 tasks.

Our experiments only use images for which sex and age labels are available, yielding a total of 223,413 images. We randomly split this data in a 80:20 ratio to form the training and validation sets respectively. The dataset provides a test set with 234 images labeled by radiologists. The training set is primarily composed of males (60%), while validation and test sets are more balanced (55% males).

We use the highest ranked model on the CheXpert leaderboard[1] with a publicly available implementation.[2] We fine-tune a Densenet121 [73] pre-trained on ImageNet [74] for this task. Additional details about the model, data preprocessing, optimizer, and hyperparameters can be found in Appendix A.1.

[1] https://stanfordmlgroup.github.io/competitions/chexpert/
[2] https://github.com/jfhealthcare/Chexpert

2.4 Challenges of Imposing Fairness Notions

We consider the traditional fair machine learning setup consisting of a training dataset $\mathcal{D} = \{(x_i, a_i, y_i)\}_{i=1}^N$, where x_i are drawn independently from the input distribution $\mathcal{X}$, $a_i \in \mathcal{A}$ are sensitive features (such as race, gender *etc.*) and $y_i \in \mathcal{Y}$ are true labels. For simplicity's sake, assume that $\mathcal{A} = \{0, 1\}$. $\mathcal{Y}$ is binary in the medical imaging task and multi-class for facial recognition. We wish to train a model $f_\theta : \mathcal{X} \to \mathbb{R}$ which can predict an outcome $\hat{y}_i$ for a given x_i. Standard training procedures outside of the fair training regime minimize the average loss over training samples, $\hat{L}(f_\theta)$. We refer to this as the *baseline* training scheme in our experiments. We refer to average loss on data points where $a_i = 1$ as $\hat{L}^{a+}(f_\theta)$ and points where $a_i = 0$ as $\hat{L}^{a-}(f_\theta)$. Below, we recall fairness notions which we use in this work and also describe how we operationalize them.

Accuracy equality requires the classification system to have equal misclassification rates across sensitive groups [44–46].

$$\mathbb{P}(\hat{y} \neq y | a = 0) \approx \mathbb{P}(\hat{y} \neq y | a = 1) \tag{2.1}$$

Because accuracy is a discontinuous function of the model parameters, we use equal loss as a surrogate, and solve:

$$\min_\theta \left[\hat{L}(f_\theta) + \alpha \left| \hat{L}^{a+}(f_\theta) - \hat{L}^{a-}(f_\theta) \right| \right] \tag{2.2}$$

Equalized odds aims to equalize the true positive and false positive rates for a classifier (sometimes also referred to as *disparate mistreatment*) [59].

$$\mathbb{P}(\hat{y} = 1|a = 1, y = y) \approx \mathbb{P}(\hat{y} = 1|a = 0, y = y) \tag{2.3}$$

The Equalized Odds Penalty [49] aims to approximate equalized odds using logits as measures of probability. Thus, we minimize the following objective:

$$\min_{\theta} \left[\hat{L}(f_\theta) + \alpha(fpr + fnr) \right], \text{ where} \tag{2.4}$$

$$fpr = \left| \frac{\sum_i p_i(1 - y_i)a_i}{\sum_i a_i} - \frac{\sum_i p_i(1 - y_i)(1 - a_i)}{\sum_i (1 - a_i)} \right| \tag{2.5}$$

$$fnr = \left| \frac{\sum_i (1 - p_i)y_i a_i}{\sum_i a_i} - \frac{\sum_i (1 - p_i)y_i(1 - a_i)}{\sum_i (1 - a_i)} \right| \tag{2.6}$$

Here, p_i denotes a softmax output (binary prediction task).

Disparate impact is a widely adopted notion that requires any decision making process' outcomes to be independent of membership in a sensitive group [17, 29, 40, 75]:

$$\mathbb{P}(\hat{y} = 1|a = 1) \approx \mathbb{P}(\hat{y} = 1|a = 0) \tag{2.7}$$

The Disparate Impact Penalty [49] aims to approximate disparate impact through the objective,

$$\min_{\theta} \left[\hat{L}(f_\theta) + \alpha di \right], \text{ where} \tag{2.8}$$

$$di = -\min \left(\frac{\sum_i a_i p_i / \sum_i a_i}{\sum_i (1 - a_i)p_i / \sum_i (1 - a_i)}, \frac{\sum_i (1 - a_i)p_i / \sum_i (1 - a_i)}{\sum_i a_i p_i / \sum_i a_i} \right) \tag{2.9}$$

Max-Min fairness focuses on maximizing the performance for the most discriminated

against group, i.e. the group with lowest utility [37, 38, 51, 52, 76–78].

$$\max \min_{a \in \mathcal{A}} \mathbb{P}(\hat{y} = y|a) \tag{2.10}$$

To optimize models for Max-Min fairness, we minimize the loss for the sensitive group with maximum loss at the current iteration. That is, we perform the following optimization at each iteration:

$$\min_{\theta} \ \max \{\hat{L}^{a+}(f_\theta), \hat{L}^{a-}(f_\theta)\} \tag{2.11}$$

Since both equality of opportunity and disparate impact notions assume existence of a beneficial outcome, they are most useful for binary classification tasks, and we only use them in the medical image classification task.

2.4.1 Training with fairness regularization

One common approach for mitigating unfairness is through imposing fairness constraints or regularizers on the training objective. In this section, we describe the effectiveness of various regularization-based methods at improving the fairness of models trained for medical image classification and facial recognition.

CheXpert We implement three types of regularizers: equal loss, disparate impact, and equality of opportunity penalties, with the aim to achieve parity in performance (*i.e.* AUC scores) for both males and females. In all previous works that apply such constraints, the experiments are either performed on linear models (*e.g.* SVM in [47]) or on small neural networks (*e.g.* 2-layer network in [49]). Under such settings, it is reasonable to assume that one can reliably measure fairness

Table 2.1: **[CheXpert - training with fairness penalties]** Results for all 5 CheXpert tasks: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE). The regularizer is optimized on the training data, and α here denotes regularizer strength. We observe adding regularizers to DNNs on training data can actually hurt disparity on the test set. One notable exception (the equal loss regularizer), shows *fairness gerrymandering*. We also report the penalty that was explicitly optimized during training (lower is better) and show that even this penalty does not generalize.

Scheme	Task	Train				Test			
		M	F	Gap	Penalty	M	F	Gap	Penalty
Baseline	CD	0.905	0.902	0.004	0.006	0.712	0.691	0.046	0.016
	ED	0.857	0.844	0.013	0.015	0.905	0.849	0.057	0.022
	CO	0.832	0.834	0.004	0.004	0.824	0.760	0.069	0.163
	AT	0.716	0.709	0.008	0.002	0.804	0.756	0.065	0.004
	PE	0.873	0.880	0.007	0.013	0.851	0.923	0.072	0.433
	Avg	**0.837**	**0.834**	**0.007**	**0.008**	**0.819**	**0.796**	**0.062**	**0.127**
Eq. Loss Train $\alpha = 100$	CD	0.934	0.932	0.002	0.005	0.764	0.758	0.005	0.001
	ED	0.879	0.872	0.007	0.016	0.914	0.875	0.039	0.019
	CO	0.888	0.888	0.003	0.003	0.840	0.849	0.026	0.136
	AT	0.732	0.727	0.005	0.002	0.825	0.772	0.053	0.003
	PE	0.882	0.886	0.004	0.005	0.861	0.897	0.036	0.387
	Avg	**0.863**	**0.861**	**0.004**	**0.006**	**0.841**	**0.830**	**0.032**	**0.109**
Eq. Loss Train $\alpha = 1000$	CD	0.820	0.814	0.007	0.001	0.805	0.770	0.035	0.002
	ED	0.816	0.799	0.018	0.004	0.888	0.811	0.077	0.010
	CO	0.680	0.686	0.006	0.000	0.909	0.800	0.109	0.010
	AT	0.625	0.613	0.012	0.002	0.740	0.719	0.021	0.001
	PE	0.843	0.851	0.009	0.002	0.834	0.924	0.090	0.248
	Avg	**0.757**	**0.753**	**0.010**	**0.002**	**0.835**	**0.805**	**0.066**	**0.054**
Disp. Impact Penalty $\alpha = 100$	CD	0.925	0.921	0.005	−0.211	0.759	0.743	0.028	−0.206
	ED	0.882	0.872	0.010	−0.316	0.921	0.843	0.077	−0.305
	CO	0.865	0.867	0.003	−0.181	0.799	0.730	0.074	−0.161
	AT	0.753	0.744	0.011	−0.526	0.766	0.690	0.076	−0.510
	PE	0.891	0.898	0.006	−0.839	0.878	0.907	0.030	−0.781
	Avg	**0.863**	**0.860**	**0.007**	**-0.415**	**0.825**	**0.783**	**0.057**	**-0.393**
Disp. Impact Penalty $\alpha = 1000$	CD	0.922	0.919	0.004	−0.196	0.795	0.725	0.070	−0.184
	ED	0.877	0.864	0.013	−0.381	0.901	0.848	0.053	−0.367
	CO	0.834	0.837	0.003	−0.213	0.705	0.673	0.040	−0.206
	AT	0.744	0.733	0.012	−0.532	0.818	0.734	0.085	−0.532
	PE	0.885	0.889	0.005	−0.834	0.846	0.903	0.057	−0.806
	Avg	**0.852**	**0.848**	**0.007**	**-0.431**	**0.813**	**0.777**	**0.061**	**-0.419**
Eq. Odds Penalty $\alpha = 100$	CD	0.934	0.932	0.004	0.009	0.743	0.745	0.002	0.005
	ED	0.889	0.880	0.008	0.024	0.883	0.836	0.047	0.019
	CO	0.879	0.884	0.004	0.002	0.776	0.790	0.045	0.083
	AT	0.771	0.768	0.008	0.005	0.769	0.699	0.070	0.010
	PE	0.895	0.900	0.005	0.005	0.840	0.893	0.053	0.185
	Avg	**0.874**	**0.873**	**0.006**	**0.009**	**0.802**	**0.792**	**0.043**	**0.060**
Eq. Odds Penalty $\alpha = 1000$	CD	0.929	0.924	0.005	0.006	0.774	0.768	0.021	0.004
	ED	0.896	0.894	0.005	0.022	0.893	0.872	0.022	0.023
	CO	0.866	0.867	0.003	0.003	0.777	0.742	0.035	0.088
	AT	0.760	0.757	0.003	0.003	0.754	0.761	0.038	0.013
	PE	0.895	0.901	0.006	0.006	0.870	0.940	0.070	0.175
	Avg	**0.869**	**0.869**	**0.005**	**0.008**	**0.814**	**0.817**	**0.037**	**0.061**

notions on the train set and expect such fairness to generalize to an unseen test set. However, as we see in our experiments, this is seldom the case with DNNs, which are highly overparametrized and can easily fit the train data [19]. Hence, in theory, these regularizers will be ineffective since the regularizer's value will be extremely low on the train set purely as a result of overfitting.

We observe a similar trend in our empirical results reported in Table 2.1[3]. The model performs very well on the train set and thus appears to be fair, where fairness is measured by the difference in AUC value of males and females. However, when evaluated on the test set, *models trained with the regularizers can be even less fair than a baseline model*. There is one noticeable exception in Table 2.1; the equal loss regularizer is able to achieve better parity between AUC of males and females on the test set. However, we observe that this parity comes at the cost of increased disparity amongst age groups. This phenomenon is called *fairness gerrymandering*, which we discuss in detail in Section 2.7. All our models are trained using standard techniques to avoid overfitting (dropout and weight decay, more details in Appendix A.1).

Thus, we conclude that achieving fairness via imposing constraints on the training set is challenging for DNNs. Their overparametrized nature leads to overfitting on training data and thus preventing any generalization of fairness on the test set. Moreover, overparametrization leads to a fluid decision boundary, which is prone to fairness gerrymandering.

Additionally we also report (in Appendix A.2) (un)fairness as measured using the same metrics as used in fairness penalties during training over train, val and test sets and observe a similar trend as with fairness measured via difference in AUC scores. This suggests that the phenomenon of fairness not generalizing to the test set also happens when measuring the exact quantity that was optimized during training.

Face recognition We find that applying an equal loss penalty on the difference in losses, even with a small coefficient, leads to improved fairness on the train set, although with a large accuracy trade-off. In many cases, training accuracy on females even becomes higher than training accuracy on males. At the same time, the validation and test accuracy do not decrease significantly, and

[3]Our full slate of results can be found in Appendix A.2.

the accuracy gap remains close to the gap of the baseline model. Increasing the penalty size leads to a higher accuracy trade-off, yet this still does not nearly eliminate the bias on the validation and test sets. One possible explanation for such behavior is that the model overfits on the fairness objective to its training data. Additionally, since the validation and testing behavior of FR systems involves discarding the classification head and only using the feature extractor, one might guess that fairness on training data was embedded in the classification head but not the feature extractor. Thus, equality of losses might be a good proxy for equality of accuracies computed using the classification head but not for accuracies computed using k-nearest neighbors in feature space on validation and test data. To test this hypothesis, we additionally compute classification accuracy on the validation set and find that fairness is not appreciably improved even in this case, so we conclude that the problem is of the overfitting nature. The detailed results can be found in Table 2.2 and in Appendix A.2.

2.4.2 Imposing fairness constraints on a holdout set

If the only reason that fairness constraints on the training set are ineffective were that training accuracy is so high that models appear fair regardless of their test-time behavior, then one might be able to bypass this problem by imposing the fairness penalty on a holdout set instead.

However, as we see in our results in Tables 2.1, 2.2 and Appendix A.2, this approach also fails. For both medical image classification and facial recognition tasks, in all of the cases where fairness is imposed on a holdout set, the downstream fairness on the test set deteriorates. We posit that the model overfits the penalty on the validation set, which ultimately harms fairness on the test set.

Table 2.2: **[Face Recognition ResNet-18]** Performance of models trained with different training schemes designed for mitigating disparity in misclassification rates between males and females. All models have ResNet-18 backbone and CosFace head. The first column refers to the training scheme used, penalty indicates the size penalty coefficient. The train accuracy is computed in a classification manner, while validation and test accuracies are computed in the 1-nearest neighbors sense. The gap subcolumn refers to the difference between male and female accuracies.

Scheme	Penalty	Train			Validation			Test		
		Male	Female	Gap	Male	Female	Gap	Male	Female	Gap
Eq. Loss Train	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	$\alpha = 0.5$	72.7	75.1	-2.4	79.3	73.4	5.9	95	93.5	1.5
	$\alpha = 1$	28.2	29.5	-1.3	67.8	62	5.8	93	91.5	1.5
	$\alpha = 2$	0	0.1	-0.1	27	18.8	8.2	70.9	64.2	6.7
Eq. Loss Holdout	baseline	84	79.6	4.4	78.6	70.8	7.8	94.2	92.3	2.0
	$\alpha = 0.5$	59.3	46.9	12.4	75.2	66.5	8.7	94.5	92.5	2.0
	$\alpha = 1$	20.6	13	7.5	57.6	47.3	10.3	89.8	85.9	3.9
Min-Max	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	fair	74.3	75.9	-1.7	79.4	73.4	6	95.5	93.4	2.1
Random Labels Flipping	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	$p = 0.1$	92.4	91.8	0.6	81.3	75	6.3	95.4	93.2	2.1
	$p = 0.3$	68.2	86.6	-18.4	75.5	74.9	0.6	94.6	93.5	1.1
	$p = 0.5$	21.6	86	-64.4	67.3	73.3	-6	92.2	93	-0.7

Another observation from our results (Table in Appendix A.2) is that optimizing for the fairness penalties on the holdout validation set results in (slightly) higher AUC disparities on the validation set itself. This indicates that penalties such as equal loss, disparate impact and equalized odds might be bad proxies for the fairness metric we measure here, *i.e.* difference in AUCs.

2.4.3 Max-Min training

Face recognition For FR models, Min-Max training results in similar behavior as applying the equality of losses penalty with a small coefficient. In particular, the training losses across genders indeed converge to similar values, and the model becomes more accurate on female identities at

train time, but this result does not transfer to test data. In fact, the disparity in misclassification rates on the validation set improves marginally, while the test set accuracy gap deteriorates. Therefore, we again encounter an overfitting problem with fairness being achieved on the train set but not on unseen data.

CheXpert Table 2.1 displays the results for Max-Min training. We observe that such a training procedure does not improve fairness on the test set.

2.4.4 Adjusted angular margins for face recognition

As we mention in 2.3.1, facial recognition systems are trained using heads which increase the angular separation between classes. One way to improve fairness of the model with respect to gender is by using different angular margins during training and therefore promoting better feature discrimination for the minority class. [43] applied this idea to learn a strategy for finding the optimal margins during training for coping with racial bias in face verification.

To test this approach we train models with increased angular margin for females. To evaluate effectiveness of this method we follow [43] and measure mean intra- and inter-class angles in addition to accuracies. The intra-class angle refers to the mean angle between the average feature vector of an identity and feature vectors of images of the same person. Inter-class angle refers to the minimal angle between the average feature vector of an identity and average feature vectors of other identities. Intuitively, we would expect that increasing angular margin for females would decrease the intra-class angle and increase the inter-class angle for female identities.

Our results show that a model trained with increased angular margin for females achieves

Table 2.3: **[Face recognition]** Accuracy and intra- and inter-class angles measured for male and female images for ResNet-152 model trained with **adjusted angle margins**. The numbers are measured on validation and test sets. It can be seen that 'fair' models (trained with increased angle margin for females) improve fairness on validation set, but increase the accuracy gap on test set.

	Penalty	Acc M	Acc F	Acc Gap	Intra M	Intra F	Inter M	Inter F
Validation	baseline	88.1	81.4	6.8	33.9	36.2	68.5	68.2
	fair	85.7	85.2	0.5	34.6	28.5	68.2	70.8
Test	baseline	96.6	94.5	2.1	43.7	47.5	70.5	70.1
	fair	95.9	91	4.9	44.7	49.1	69.6	68.4

better validation accuracy, and intra- and inter-class separation for females. In fact, the accuracy gap on validation data drops from 6.8% to 0.5% for ResNet-152 model. However, these results do not transfer to test data which consists of photos of new identities. Ultimately, the accuracy and angle metrics worsen for female groups leading to increased misclassification rates across genders, see Table 2.3. These results indicate that adjusting angular margins for mitigating unfairness leads to an overfitting problem since fairness improves only on identities that appear in the training set.

2.5 Fair feature representations do not yield fair model behavior

Another strategy for mitigating unfairness is through learning fair representations that are not correlated with sensitive attributes in the hope that a classifier built on top of 'fair features' will yield fair predictions. A recent paper introduces SensitiveNets, a sensitive information removal network trained on top of a pre-trained feature extractor with an adversarial sensitive regularizer [62]. Intuitively, the method learns a projection of embeddings $\varphi(x)$ that minimizes the performance of a sensitive attribute classifier while maximizing the performance of a face

recognition system.

We apply this adversarial approach by training a sensitive information removal network for minimizing the facial recognition loss while simultaneously maximizing the probability of predicting a fixed gender class for all images. At the same time, the discriminator is trained for predicting the gender from $\varphi(x)$. Therefore, this can be formulated as a two-players game where the discriminator aims to predict the gender from features $\varphi(x)$, while the sensitive information removal network aims to output gender-independent features $\varphi(x)$ and confuse the discriminator. We find that when a network is trained without adversarial regularization, the discriminator predicts gender with 97% accuracy on the test set. Adversarial regularization with a sufficient penalty decreases the performance of a gender predictor to random, meaning that the resulting features $\varphi(x)$ are gender-independent.

The results show that when fair features $\varphi(x)$ are obtained through manipulating male images, *e.g.* when the adversarial regularizer forces the discriminator to label all images as females, the accuracy gap reduces at the expense of male accuracy. At the same time, when the adversarial regularizer manipulate female images, the model's accuracy gap only increases due to a drop in female accuracy. All results can be found in Table 2.4 and Appendix A.2. Thus, we conclude that adversarial training decreases accuracy of the FR system on images whose feature vectors were used in the regularizer, damaging performance on all other classes.

There are principled mathematical reasons why "fair features" are problematic. When the dataset is imbalanced, like Celeb-A which is majority women, the equilibrium strategy of a discriminator with no useful information is to always predict "female." In this case, the feature extractor, which has the goal of fooling the discriminator, can only do so by distorting the features of men to appear female – a strategy that disproportionately hurts accuracy of the male group.

Table 2.4: **[Face recognition]** Facial recognition and gender classification test accuracy for ResNet-152 model trained with a sensitive information removal network on top. Here, α denotes the magnitude of adversarial penalty and for sufficiently large α, the discriminator predicts a fixed gender for all images (in bold). Gender in the first column is the gender of images penalized during training.

	α	MALE	FEMALE	GAP	SENS ACC
PENALIZE FEMALES	$\alpha = 0$	95.9	93.3	2.6	97.0
	$\alpha = 1$	95.7	92.5	3.1	78.2
	$\alpha = 2$	95.1	91.0	4.1	**38.7**
	$\alpha = 3$	94.2	88.7	5.5	38.7
PENALIZE MALES	$\alpha = 0$	95.9	93.3	2.6	97.0
	$\alpha = 1$	95.4	93.2	2.2	90.5
	$\alpha = 2$	92.8	91.9	0.9	**61.3**
	$\alpha = 3$	90.2	90.1	0.1	61.3

Furthermore, in the hypothetical scenario where groups are balanced and the training process succeeds in creating features with no gender information, it becomes impossible to create a downstream classifier that assigns any label to men at a different rate than women. This is true because if such a classifier existed, it could be used to create a better-than-random gender classifier. In cases where the distribution of labels is different for men and women, this means that false positive and false negative rates must differ across genders.

2.6 A simple baseline for fairness: label flipping

Many of the methods for fairness that we test above sacrifice accuracy without any gains in fairness on the testing data. Sometimes, they sacrifice both fairness and accuracy. Label flipping is one way to navigate the trade-off by adjusting the accuracy of individual groups [79]. This is done by randomly flipping labels in the training data of the subgroup with superior accuracy. One might suppose that flipping labels will simply hurt performance, however we find that on facial

recognition, this simple method can actually remedy unfairness without harming performance.

Face recognition For the facial recognition task, our models achieve higher accuracy on male images than on female images. Therefore, to decrease the accuracy gap, we randomly flip labels of a portion of male images during training. We do this by swapping male identities only with other random male identities so that female accuracy is largely preserved. We try different proportions of flipped labels: $p = 0.1, 0.3, 0.5$. Surprisingly, flipping 30% of male labels increases female test accuracy by 0.6% thereby decreasing the gender gap from 2.1% to 1.6% for ResNet-152 model. Also, flipping half of the male labels only drops male test accuracy from 96.6% to 95.2% and results in a 0.3% accuracy gap on test data, see results in Appendix A.2.

CheXpert For CheXpert, we randomly flip the true label for $p = 0.01, 0.05,$ and 0.1 of samples from a particular sensitive group in each iteration. In general, we observe very unstable trends in AUC scores and disparities when using label flipping. For example, we observe that flipping 1% of male samples during training results in a major drop in AUC for males, averaged over all tasks (0.816 to 0.746) and a minor increase in AUC for females (0.781 to 0.784) resulting in increased disparity. However, with 5% flipped samples, we see that male AUC drops to a similar value (0.816 to 0.716) as the female AUC (0.781 to 0.705), thus leading to reduced disparity. A similar trend is seen when male samples are flipped. AUC values of models trained on randomly flipped data are consistently lesser than the baseline model. We thus conclude that random flipping might not be the best solution to achieving fairness in this setting, since it yields unreliable trends in fairness and reliably performs worse than the baseline model. Results can be found in Appendix A.2.

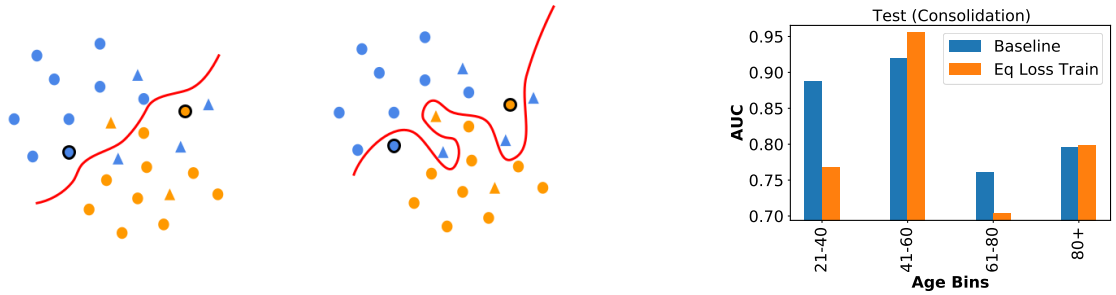


Figure 2.2: **Left:** an illustration of gerrymandering; color denotes label, shape and outline are sensitive attributes. Model on the right is more fair to shape but less fair to outline. **Right:** gerrymandering behavior on cheXpert. The equal loss regularizer (in orange) achieves better parity along one demographic (sex, see Table 2.1 by making predictions more disparate along another demographic (age). For a model to be fair across age groups, the bars should all be of the same height.

2.7 Fairness gerrymandering

Another unintended consequence of fairness interventions is fairness gerrymandering in which a model becomes more fair for one group but less fair to others [20, 21]. Similar/related images tend to clump together in feature space. For this reason, group-based fairness constraints that change the decision boundary are likely to induce label flips for entire groups of images with common features such as skin tone or age. Figure 2.2 (left) illustrates this phenomenon.

CheXpert In Table 2.1, we observed better parity for models trained with an equal loss penalty than baseline training. In this section, we take a closer look at how this affected disparities across another sensitive feature, age. Consolidation (CO) is the task for which disparity in AUC across males and females reduced the most. Figure 2.2 (right) shows that this reduction in disparity across males and females induced greater disparities across age groups, which is indicated by larger differences between subsequent age groups for the "fair" model. We see that fairness regularization with respect to gender leads to increased unfairness with respect to age.

Table 2.5: **[Fairness gerrymandering on BUPT dataset]** The first row shows accuracies obtained with baseline model. The second, third, and fourth blocks reflect performance of models trained with equal loss penalty, adjusted angle margin for females, and randomly flipped labels for males respectively. For the models trained for "gender-fairness", we report differences from baseline.

MODEL	AFRICAN	ASIAN	CAUCASIAN	INDIAN
BASELINE	92.8	90.4	93.7	94.5
EQ LOSS	90.9	89.4	91.5	93.6
DIFF	1.9	1.0	0.1	0.9
MARGINS	89.1	85.7	90.7	91.9
DIFF	3.7	**4.7**	3	3.5
FLIP	93.7	91.5	94	94.9
DIFF	-0.9	**-1.1**	-0.3	-0.4

Face recognition We investigate if face recognition models trained to be gender-fair suffer from gerrymandering. In particular, we consider models trained with the equal loss penalty, adjusted angle margins, and random label flipping. We evaluate our models on the race-labeled BUPT-Balancedface dataset [43] and find that models' performance changes disproportionately with respect to race when trained for "gender fairness". For example, the FR model trained to have equal loss across genders is significantly less accurate on people of African origin than the baseline model, while the accuracy on Caucasian faces remains almost the same. The system trained with adjusted angle margins becomes even less fair to Asians, the group most discriminated against by the baseline model, with the accuracy gap between that on images of Asian and Indian people being increased by 1.2%, see Table 2.5). We then compare these results to a model trained with random label flipping. Surprisingly, the model trained on corrupted data improves accuracy for all four races, but the biggest improvement occurs on images of Asian individuals.

2.8 Discussion

We empirically demonstrate the challenges of applying current train time algorithmic fairness interventions to DNNs. Due to overparameterization, DNNs can easily overfit the training data and thus give a false sense of fairness during training which does not generalize to the test set. In fact, we observe that adding fairness constraints via existing methods can even exacerbate unfairness on the test set. In cases where train-time fairness interventions are effective on the test set, we observe fairness gerrymandering. We posit that overparameterization makes the decision boundary of the learned neural network extremely fluid, and changing it to conform to fairness along a certain attribute can hurt fairness along another sensitive attribute. Additionally, we observe that using a holdout set to optimize fairness measures also does not yield fair outcomes on the test set, due to both overfitting and bad approximation by fairness surrogates. Our results outline some of the limitations of current train time interventions for fairness in deep learning. Evaluating other kinds of existing fairness interventions, such as pre-processing and post-processing for overparameterized models, as well as building better train time interventions are interesting avenues for future work.

Chapter 3: Fairness: New Notions for Deep Learning

> We can only see a short distance
> ahead, but we can see plenty there
> that needs to be done.
>
> *Alan Turing*

Even if fairness constraints in Chapter 2 worked as expected, we argue that traditional notions of fairness that are only based on models' outputs are not sufficient when the model is vulnerable to adversarial attacks, which is especially true for DL models. We show that in some cases, it may be easier for an attacker to target a particular subgroup, resulting in a form of *robustness bias*. We show that measuring robustness bias is a challenging task for DNNs and propose two methods to measure this form of bias. We then conduct an empirical study on state-of-the-art neural networks on commonly used real-world datasets such as CIFAR-10, CIFAR-100, Adience, and UTKFace and show that in almost all cases there are subgroups (in some cases based on sensitive attributes like race, gender, etc) which are less robust and are thus at a disadvantage. We argue that this kind of bias arises due to both the data distribution and the highly complex nature of the learned decision boundary in the case of DNNs, thus making mitigation of such biases a non-trivial task. Our results show that robustness bias is an important criterion to consider while auditing real-world systems that rely on DNNs for decision making.

More broadly, this chapter presents some of the first work to bridge fairness and robustness communities.

3.1 Introduction

Automated decision-making systems that are driven by data are being used in a variety of different real-world applications. In many cases, these systems make decisions on data points that represent humans (*e.g.*, targeted ads [23, 25], personalized recommendations [26, 27], hiring [30, 80], credit scoring [28], or recidivism prediction [81]). In such scenarios, there is often concern regarding the fairness of outcomes of the systems [17, 18]. This has resulted in a growing body of work from the nascent Fairness, Accountability, Transparency, and Ethics (FATE) community that—drawing on prior legal and philosophical doctrine—aims to define, measure, and (attempt to) mitigate manifestations of unfairness in automated systems [81–84].

Most of the initial work on fairness in machine learning considered notions that were one-shot and considered the model and data distribution to be static [17,53,81,85–87]. Recently, there has been more work exploring notions of fairness that are dynamic and consider the possibility that the world (*i.e.*, the model as well as data points) might change over time [37, 88–90]. Our proposed notion of robustness bias has subtle difference from existing one-shot and dynamic notions of fairness in that it requires each partition of the population be equally robust to imperceptible changes in the input (*e.g.*, noise, adversarial perturbations, etc).

We propose a simple and intuitive notion of *robustness bias* which requires subgroups of populations to be equally "robust." Robustness can be defined in multiple different ways [91–93]. We take a general definition which assigns points that are farther away from the decision

boundary higher robustness. Our key contributions are as follows:

- We define a simple, intuitive notion of ***robustness bias*** that requires all partitions of the dataset to be equally robust. We argue that such a notion is especially important when the decision-making system is a deep neural network (DNN) since these have been shown to be susceptible to various attacks [94, 95]. Importantly, our notion depends not only on the outcomes of the system, but also on the distribution of distances of data-points from the decision boundary, which in turn is a characteristic of *both* the data distribution and the learning process.

- We propose different methods to ***measure this form of bias***. Measuring the exact distance of a point from the decision boundary is a challenging task for deep neural networks which have a highly non-convex decision boundary. This makes the measurement of robustness bias a non-trivial task. In this chapter, we leverage the literature on adversarial machine learning and show that we can efficiently approximate robustness bias by using adversarial attacks and randomized smoothing to get estimates of a point's distance from the decision boundary.

- We do an in-depth analysis of *robustness bias* on popularly used datasets and models. Through ***extensive empirical evaluation*** we show that unfairness can exist due to different partitions of a dataset being at different levels of robustness for many state-of-the art models that are trained on common classification datasets. We argue that this form of unfairness can happen due to both the data distribution and the learning process and is an important criterion to consider when auditing models for fairness.

3.1.1 Related Work

Fairness in ML Models that learn from historic data have been shown to exhibit unfairness, *i.e.*, they disproportionately benefit or harm certain subgroups (often a sub-population that shares a common sensitive attribute such as race, gender etc) of the population [17, 28, 81]. This has resulted in a lot of work on quantifying, measuring and to some extent also mitigating unfairness [47,53,59,63,85–87,96–104]. Most of these works consider notions of fairness that are one-shot—that is, they do not consider how these systems would behave over time as the world (*i.e.*, the model and data distribution) evolves. Recently more works have taken into account the dynamic nature of these decision-making systems and consider fairness definitions and learning algorithms that fare well across multiple time steps [37,88–90]. We take inspiration from both the one-shot and dynamic notions but take a slightly different approach by requiring all subgroups of the population to be equally robust to minute changes in their features. These changes could either be random (*e.g.*natural noise in measurements) or carefully crafted adversarial noise. This is closely related to [88]'s effort-based notion of fairness; however, their notion has a very specific use case of societal scale models whereas our approach is more general and applicable to all kinds of models. Our work is also closely related to and inspired by Zafar et al.'s use of a regularized loss function which captures fairness notions and reduces disparity in outcomes [85]. There are major differences in both the *approach* and *application* between our work and that of Zafar et al's. Their disparate impact formulation aims to equalize the average distance of points to the decision boundary, $\mathbb{E}[d(x)]$; our approach, instead, aims to equalize the number of points that are "safe", *i.e.*$\mathbb{E}[\mathbb{I}\{d(x) > \tau\}]$ (see section 3.3 for a detailed description). Our proposed metric is preferable for applications of adversarial attack or noisy data, the focus of this chapter; whereas

the metric of Zafar et al is more applicable for an analysis of the consequence of a decision in a classification setting.

Robustness Deep Neural Networks (DNNs) have been shown to be susceptible to carefully crafted adversarial perturbations which—imperceptible to a human—result in a misclassification by the model [91–93]. In the context of this chapter, we use adversarial attacks to approximate the distance of a data point to the decision boundary. For this we use state-of-the-art white-box attacks proposed by [95] and [94]. Due to the many works on adversarial attacks, there have been many recent works on provable robustness to such attacks. The high-level goal of these works is to estimate a (tight) lower bound on the distance of a point from the decision boundary [105–107]. We leverage these methods to estimate distances from the decision boundary which helps assess robustness bias (defined formally in Section 3.3).

Fairness and Robustness Recent works have proposed poisoning attacks on fairness [108, 109]. [110] analyze why noise in features can cause disparity in error rates when learning a regression. We believe that our work is the very first to show that different subgroups of the population can have different levels of robustness which can lead to unfairness. We hope that this will lead to more work at the intersection of these two important sub-fields of ML.

3.2 Heterogeneous Robustness

In a classification setting, a learner is given data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ consisting of inputs $x_i \in \mathbb{R}^d$ and outputs $y_i \in \mathcal{C}$ which are labels in some set of classes $\mathcal{C} = \{c_1, \ldots, c_k\}$. These classes form a partition on the dataset such that $\mathcal{D} = \bigsqcup_{c \in \mathcal{C}} \{(x_i, y_i) \mid y_i = c_j\}$. The goal of learning in decision boundary-based optimization is to draw delineations between points in

feature space which sort the data into groups according to their class label. The learning generally tries to maximize the classification accuracy of the decision boundary choice. A learner chooses some loss function $\mathcal{L}$ to minimize on a training dataset, parameterized by parameters θ, while maximizing the classification accuracy on a test dataset.

Of course there are other aspects to classification problems that have recently become more salient in the machine learning community. Considerations about the fairness of classification decisions, for example, are one such way in which additional constraints are brought into a learner's optimization strategy. In these settings, the data $\mathcal{D} = \{(x_i, y_i, s_i)\}_{i=1}^{N}$ is imbued with some metadata which have a sensitive attribute $\mathcal{S} = \{s_1, \ldots, s_t\}$ associated with each point. Like the classes above, these sensitive attributes form a partition on the data such that $\mathcal{D} = \bigsqcup_{s \in \mathcal{S}} \{(x_i, y_i, s_i) \mid s_i = s\}$. Without loss of generality, we assume a single sensitive attribute. Generally speaking, learning with fairness in mind considers the output of a classifier based off

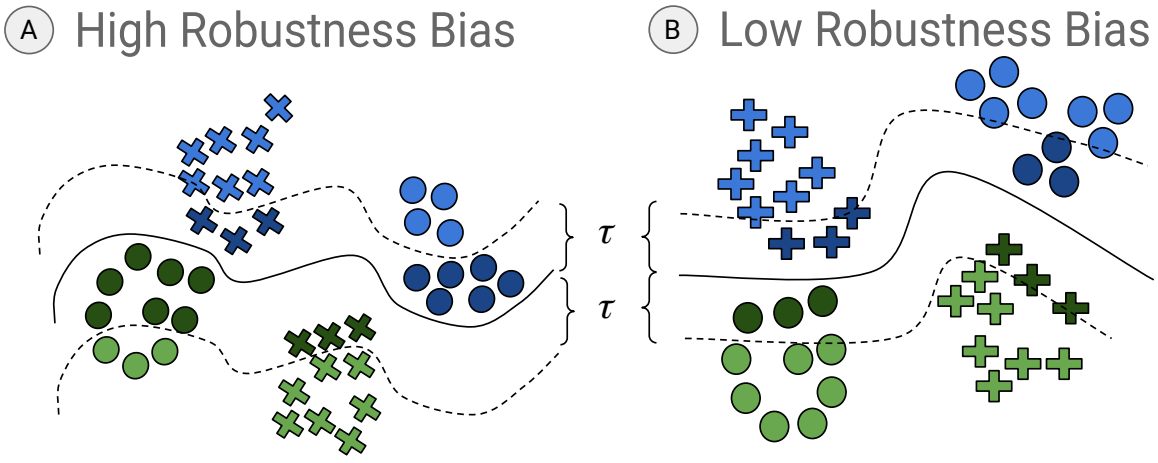
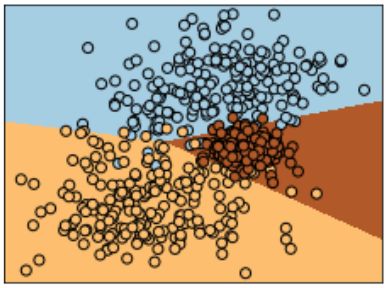
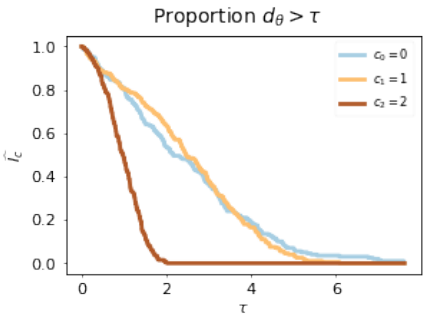


Figure 3.1: A toy example showing robustness bias. A.) the classifier (solid line) has 100% accuracy for blue and green points. However for a budget τ (dotted lines), 70% of points belonging to the "round" subclass (showed by dark blue and dark green) will get attacked while only 30% of points in the "cross" subclass will be attacked. This shows a clear bias against the "round" subclass which is less robust in this case. B.) shows a different classifier for the same data points also with 100% accuracy. However, in this case, with the same budget τ, 30% of both "round" and "cross" subclass will be attacked, thus being less biased.



((a)) Three-class classification problem for randomly generated data.



((b)) Proportion samples which are greater than τ away from a decision boundary.

Figure 3.2: An example of multinomial logistic regression.

of the partition of data by the sensitive attribute, where some objective behavior, like minimizing disparate impact or treatment [85], is integrated into the loss function or learning procedure to find the optimal parameters θ.

There is not a one-to-one correspondence between decision boundaries and classifier performance. For any given performance level on a test dataset, there are infinitely many decision boundaries which produce the same performance, see Figure 3.1. This raises the question: *if we consider all decision boundaries or model parameters which achieve a certain performance, how do we choose among them? What are the properties of a desirable, high-performing decision boundary?* As the community has discovered, one *undesirable* characteristic of a decision boundary is its proximity to data which might be susceptible to adversarial attack [91–93]. This provides intuition that we should prefer boundaries that are as far away as possible from example data [111, 112].

Let us look at how this plays out in a simple example. In multinomial logistic regression, the decision boundaries are well understood and can be written in closed form. This makes it easy for us to compute how close each point is to a decision boundary. Consider for example a dataset and learned classifier as in Figure 3.2(a). For this dataset, we observe that the brown class, as a

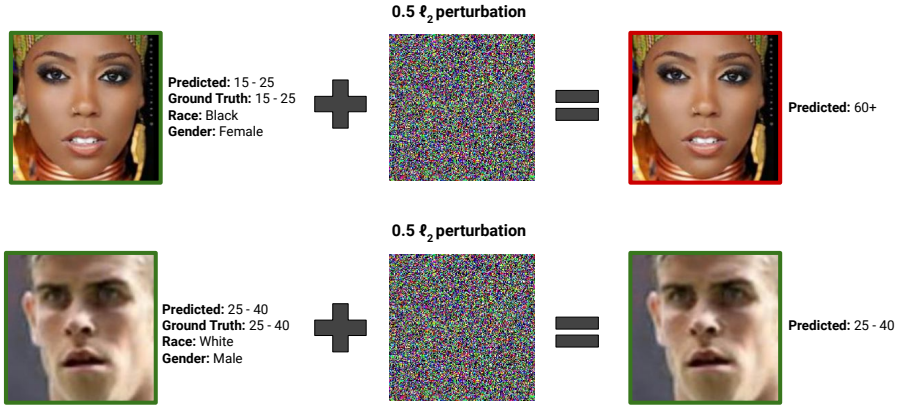


Figure 3.3: An example of robustness bias in the UTKFace dataset. A model trained to predict age group from faces is fooled for inputs belonging to certain subgroups (black and female in this example) for a given perturbation, but is robust for inputs belonging to other subgroups (white and male in this example) for the *same magnitude* of perturbation. We use the UTKFace dataset to make a broader point that robustness bias can cause harm. In the specific case of UTKFace (and similar datasets), the task definition of predicting age from faces itself is flawed, as has been noted in many previous studies [3–5].

whole, is closer to a decision boundary than the yellow or blue classes. We can quantify this by plotting the proportion of data that are greater than a distance τ away from a decision boundary, and then varying τ. Let $d_\theta(x)$ be the minimal distance between a point x and a decision boundary corresponding to parameters θ. For a given partition $\mathcal{P}$ of a dataset, $\mathcal{D}$, such that $\mathcal{D} = \bigsqcup_{P \in \mathcal{P}} P$, we define the function:

$$\widehat{I_P}(\tau) = \frac{|\{(x, y) \in P \mid d_\theta(x) > \tau, y = \hat{y}\}|}{|P|}$$

If each element of the partition is uniquely defined by an element, say a class label, c, or a sensitive attribute label, s, we equivalently will write $\widehat{I}_c(\tau)$ or $\widehat{I}_s(\tau)$ respectively. We plot this over a range of τ in Figure 3.2(b) for the toy classification problem in Figure 3.2(a). Observe that the function for the brown class decreases significantly faster than the other two classes, quantifying how much closer the brown class is to the decision boundary.

From a strictly classification accuracy point of view, the brown class being significantly closer to the decision boundary is not of concern; all three classes achieve similar classification accuracy. However, when we move away from this toy problem and into neural networks on real data, this difference between the classes could become a potential vulnerability to exploit, particularly when we consider adversarial examples.

3.3 Robustness Bias

Our goal is to understand how susceptible different classes are to perturbations (e.g., natural noise, adversarial perturbations). Ideally, no one class would be more susceptible than any other, but this may not be possible. We have observed that for the same dataset, there may be some classifiers which have differences between the distance of that partition to a decision boundary; and some which do not. There may also be one partition $\mathcal{P}$ which exhibits this discrepancy, and another partition $\mathcal{P}'$ which does not. Therefore, we make the following statement about robustness bias:

Definition 1. *A dataset $\mathcal{D}$ with a partition $\mathcal{P}$ and a classifier parameterized by θ exhibits* **robustness bias** *if there exists an element $P \in \mathcal{P}$ for which the elements of P are either significantly closer to (or significantly farther from) a decision boundary than elements not in P.*

A partition $\mathcal{P}$ may be based on sensitive attributes such as race, gender, or ethnicity—or other class labels. For example, given a classifier and dataset with sensitive attribute "race", we might say that a classifier exhibits robustness bias if, partitioning on that sensitive attribute, for some value of "race" the average distance of members of that particular racial value are substantially closer to the decision boundary than other members.

We might say that a dataset, partition, and classifier do not exhibit robustness bias if for all $P, P' \in \mathcal{P}$ and all $\tau > 0$

$$\begin{aligned} &\mathbb{P}_{(x,y)\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \in P, y = \hat{y}\} \approx \\ &\mathbb{P}_{(x,y)\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \in P', y = \hat{y}\}. \end{aligned} \tag{3.1}$$

Intuitively, this definition requires that for a given perturbation budget τ and a given partition P, one should not have any incentive to perturb data points from P over points that do not belong to P. Even when examining this criterion, we can see that this might be particularly hard to satisfy. Thus, we want to quantify the disparate susceptibility of each element of a partition to adversarial attack, i.e., how much farther or closer it is to a decision boundary when compared to all other points. We can do this with the following function for a dataset $\mathcal{D}$ with partition element $P \in \mathcal{P}$ and classifier parameterized by θ:

$$\begin{aligned} RB(P,\tau) = \mid &\mathbb{P}_{x\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \in P, y = \hat{y}\} - \\ &\mathbb{P}_{x\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \notin P, y = \hat{y}\} \mid \end{aligned} \tag{3.2}$$

Observe that $RB(P,\tau)$ is a large value if and only if the elements of P are much more (or less) adversarially robust than elements not in P. We can then quantify this for each element $P \in \mathcal{P}$—but a more pernicious variable to handle is τ. We propose to look at the area under the curve $\widehat{I_P}$ for all τ:

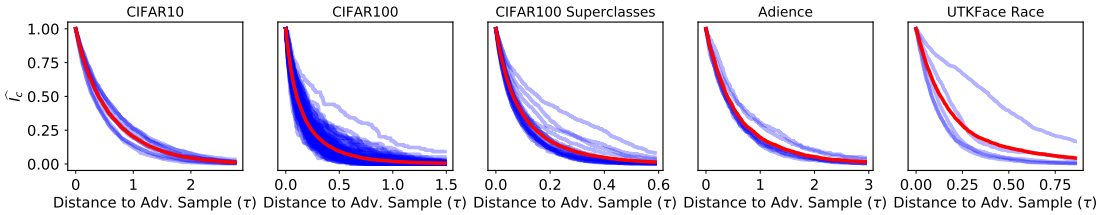


Figure 3.4: For each dataset, we plot $\widehat{I}_c(\tau)$ for each class c in each dataset. Each blue line represents one class. The red line represents the mean of the blue lines, i.e., $\sum_{c \in \mathcal{C}} \widehat{I}_c(\tau)$ for each τ.

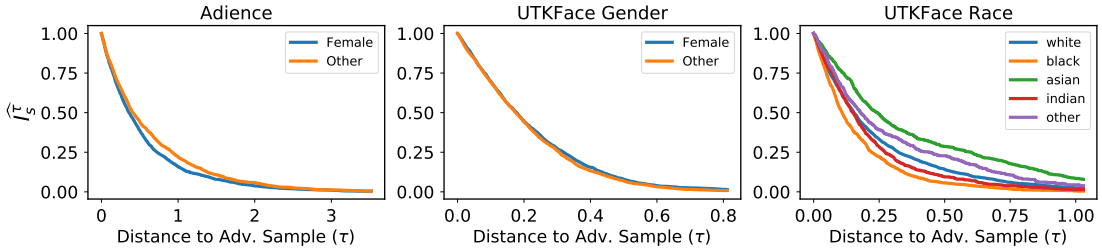


Figure 3.5: For each dataset, we plot $\widehat{I}_s^\tau$ for each sensitive attribute s in each dataset. Figures for other models can be found in Appendix B.1.

$$\sigma(P) = \frac{AUC(\widehat{I_P}) - AUC(\sum_{P' \neq P} \widehat{I_{P'}})}{AUC(\sum_{P' \neq P} \widehat{I_{P'}})} \tag{3.3}$$

Note that these notions take into account the distances of data points from the decision boundary and hence are orthogonal and complementary to other traditional notions of bias or fairness (*e.g.*, disparate impact/disparate mistreatment [85], etc). This means that having lower robustness bias does not necessarily come at the cost of fairness as measured by these notions. Consider the motivating example shown in Figure 3.1: the decision boundary on the right has lower robustness bias but preserves all other common notions (*e.g.* [53, 59, 86]) as both classifiers maintain 100% accuracy.

3.3.1 Real-world Implications: Degradation of Quality of Service

Deep neural networks are the core of many real-world applications, for example, facial recognition, object detection, etc. In such cases, perturbations in the input can occur due to multiple factors such as noise due to the environment or malicious intent by an adversary. Previous works have highlighted how harms can be caused due to the degradation in quality of service for certain sub-populations [3,65]. Figure 3.3 shows an example of inputs from the UTKFace dataset where an ℓ_2 perturbation of 0.5 could change the predicted label for an input with race "black" and gender "female" but an input with race "white" and gender "male" was robust to the same magnitude of perturbation. In such a case, the system worked better for a certain sub-group (white, male) thus resulting in unfairness. It is important to note that we use datasets such as Adience and UTKFace (described in detail in section 3.5) only to demonstrate the importance of having unbiased robustness. As noted in previous works, the very task of predicting age from a person's face is a flawed task definition with many ethical concerns [3–5].

3.4 Measuring Robustness Bias

Robustness bias as defined in the previous section requires a way to measure the distance between a point and the (closest) decision boundary. For deep neural networks in use today, a direct computation of $d_\theta(x)$ is not feasible due to their highly complicated and non-convex decision boundary. However, we show that we can leverage existing techniques from the literature on adversarial attacks to efficiently approximate $d_\theta(x)$. We describe these in more detail in this section.

3.4.1 Adversarial Attacks (Upper Bound)

For a given input and model, one can compute an *upper bound* on $d_\theta(x)$ by performing an optimization that alters the input image slightly so as to place the altered image into a different category than the original. Assume for a given data point x, we are able to compute an adversarial image $\tilde{x}$, then the distance between these two images provides an upper bound on distance to a decision boundary, i.e, $\|x - \tilde{x}\| \geq d_\theta(x)$.

We evaluate two adversarial attacks: DeepFool [95] and CarliniWagner's L2 attack [94]. We extend $\widehat{I_P}$ for DeepFool and CarliniWagner as

$$\widehat{I_P^{DF}} = \frac{|\{(x,y) \in P | \tau < \|x - \tilde{x}\|, y = \hat{y}\}|}{|P|} \tag{3.4}$$

and

$$\widehat{I_P^{CW}} = \frac{|\{(x,y) \in P | \tau < \|x - \tilde{x}\|, y = \hat{y}\}|}{|P|} \tag{3.5}$$

respectively. We use similar notation to define $\sigma^{DF}(P)$, and $\sigma^{CW}(P)$ (σ as defined in Eq 3.3). While these methods are guaranteed to yield upper bounds on $d_\theta(x)$, they need not yield similar behavior to $\widehat{I_P}$ or $\sigma(P)$. We perform an evaluation of this in Section 3.7.1.

3.4.2 Randomized Smoothing (Lower Bound)

Alternatively one can compute a lower bound on $d_\theta(x)$ using techniques from recent works on training provably robust classifiers [105, 106]. For each input, these methods calculate a radius in which the prediction of x will not change (i.e. the robustness certificate). In particular, we use the *randomized smoothing* method [105, 106] since it is scalable to large and deep neural

networks and leads to state-of-the-art in provable defenses. Randomized smoothing transforms the base classifier f to a new smooth classifier g by averaging the output of f over noisy versions of x. This new classifier g is more robust to perturbations while also having accuracy on par to the original classifier. It is also possible to calculate the radius δ_x (in the ℓ_2 distance) in which, with high probability, a given input's prediction remains the same for the smoothed classifier (i.e. $d_\theta(x) \geq \delta_x$). A given input x is then said to be provably robust, with high probability, for a δ_x ℓ_2-perturbation where δ_x is the robustness certificate of x.

For each point we use its δ_x, calculated using the method proposed by [106], as a proxy for $d_\theta(x)$. The magnitude of δ_x for an input is a measure of how robust an input is. Inputs with higher δ_x are more robust than inputs with smaller δ_x. Again, we extend $\widehat{I_P}$ for Randomized Smoothing as

$$\widehat{I_P^{RS}} = \frac{|\{(x, y) \in P | \tau < \delta_x, y = \hat{y}\}|}{|P|} \tag{3.6}$$

We use similar notation to define $\sigma^{RS}(P)$ (see Eq 3.3).

3.5 Empirical Evidence of Robustness Bias in the Wild

We hypothesize that there exist datasets and model architectures which exhibit robustness bias. To investigate this claim, we examine several image-based classification datasets and common model architectures.

Datasets and Model Architectures: We perform these tests of the datasets **CIFAR-10** [113], **CIFAR-100** [113] (using both 100 classes and 20 super classes), **Adience** [114], and **UTKFace** [115]. The first two are widely accepted benchmarks in image classification, while the latter two provide significant metadata about each image, permitting various partitions of the

data by final classes and sensitive attributes. More details can be found in Appendix B.1.

Our experiments were performed using PyTorch's torchvision module [116]. We first explore a simple **Multinomial Logistic Regression** model which could be fully analyzed with direct computation of the distance to the nearest decision boundary. For convolutional neural networks, we focus on **Alexnet** [117], **VGG19** [118], **ResNet50** [119], **DenseNet121** [73], and **Squeezenet1_0** [120] which are all available through torchvision. We use these models since they are widely used for a variety of tasks. We achieve performance that is comparable to state-of-the-art performance on these datasets for these models.[1] Additionally, we also train some other popularly used dataset specific architectures like a deep convolutional neural network (we call this **Deep CNN**)[2] and **PyramidNet** ($\alpha = 64$, depth=110, no bottleneck) [121] for CIFAR-10. We re-implemented Deep CNN in pytorch and used the publicly available repo to train PyramidNet[3]. We use another deep convolutional neural network (which we refer to as **Deep CNN CIFAR100**[4] and **PyramidNet** ($\alpha = 48$, depth=164, with bottleneck) for CIFAR-100 and CIFAR-100Super. For Adience and UTKFace we additionally take simple deep convolutional neural networks with multiple convolutional layers each of which is followed by a ReLu activation, dropout and maxpooling. As opposed to architectures from torchvision (which are pre-trained on ImageNet) these architectures are trained from scratch on the respective datasets. We refer to them as **UTK Classifier** and **Adience Classifier** respectively. These simple models serve two purposes: they form reasonable baselines for comparison with pre-trained ImageNet models finetuned on the respective datasets, and they allow us to analyze robustness bias when models are trained from scratch.

[1] See Appendix B.1 Table B.2 for model performances, additional quantitative results and supporting figures.
[2] `http://torch.ch/blog/2015/07/30/cifar.html`
[3] `https://github.com/dyhan0920/PyramidNet-PyTorch`
[4] https://github.com/aaron-xichen/pytorch-playground/blob/master/cifar/model.py

In sections 3.7 and 3.8 we audit these datasets and the listed models for robustness bias. In section 3.6, we train logistic regression on all the mentioned datasets and evaluate robustness bias using an exact computation. We then show in section 3.7 and 3.8 that robustness bias can be efficiently approximated using the techniques mentioned in 3.4.1 and 3.4.2 respectively for much more complicated models, which are often used in the real world. We also provide a thorough analysis of the types of robustness biases exhibited by some of the popularly used models on these datasets.

3.6 Exact Computation in a Simple Model: Multinomial Logistic Regression

We begin our analysis by studying the behavior of multinomial logistic regression. Admittedly, this is a simple model compared to modern deep-learning-based approaches; however, it enables is to explicitly compute the exact distance to a decision boundary, $d_\theta(x)$. We fit a regression to each of our vision datasets to their native classes and plot $\widehat{I}_c(\tau)$ for each dataset. Figure 3.4 shows the distributions of $\widehat{I}_c(\tau)$, from which we observe three main phenomena: (1) the general shape of the curves are similar for each dataset, (2) there are classes which are significant outliers from the other classes, and (3) the range of support of the τ for each dataset varies significantly. We discuss each of these individually.

First, we note that the shape of the curves for each dataset is qualitatively similar. Since the form of the decision boundaries in multinomial logistic regression are linear delineations in the input space, it is fair to assume that this similarity in shape in Figure 3.4 can be attributed to the nature of the classifier.

Second, there are classes c which indicate disparate treatment under $\widehat{I}_c(\tau)$. The treatment

disparities are most notable in UTKFace, the superclass version CIFAR-100, and regular CIFAR-100. This suggests that, when considering the dataset as a whole, these outlier classes are less susceptible to adversarial attacks than other classes. Further, in UTKFace, there are some classes that are considerably more susceptible to adversarial attack because a larger proportion of that class is closer to the decision boundaries.

We also observe that the median distance to the decision boundary can vary based on the dataset. The median distance to a decision boundary for each dataset is 0.40 for CIFAR-10; 0.10 for CIFAR-100; 0.06 for the superclass version of CIFAR-100; 0.38 for Adience; and 0.12 for UTKFace. This is no surprise as $d_\theta(x)$ depends both on the location of the data points (which are fixed and immovable in a learning environment) and the choice of architectures/parameters.

Finally, we consider another partition of the datasets. Above, we consider the partition of the dataset which occurs by the class labels. With the Adience and UTKFace datasets, we have an additional partition by sensitive attributes. Adience admits partitions based on gender; UTKFace admits partitions by gender and ethnicity. We note that Adience and UTKFace use categorical labels for these multidimensional and socially complex concepts. We know this to be reductive and serves to minimize the contextualization within which race and gender derive their meaning [5, 122]. Further, we acknowledge the systems and notions that were used to reify such data partitions and the subsequent implications and conclusions drawn therefrom. We use these socially and systemically-laden partitions to demonstrate that the functions we define, $\widehat{I_P}$ and σ depend upon how the data are divided for analysis. To that end, the function $\widehat{I_P}$ is visualized in Figure 3.5. We observe that the Adience dataset, which exhibited some adversarial robustness bias in the partition on $\mathcal{C}$ only exhibits minor adversarial robustness bias in the partition on $\mathcal{S}$ for the attribute 'Female'. On the other hand, UTKFace which had significant adversarial robustness

bias does exhibit the phenomenon for the sensitive attribute 'Black' but not for the sensitive attribute 'Female'.

This emphasizes that adversarial robustness bias is dependent upon the dataset and the partition. We will demonstrate later that it is also dependent on the choice of classifier. First, we talk about ways to approximate $d_\theta(x)$ for more complicated models.

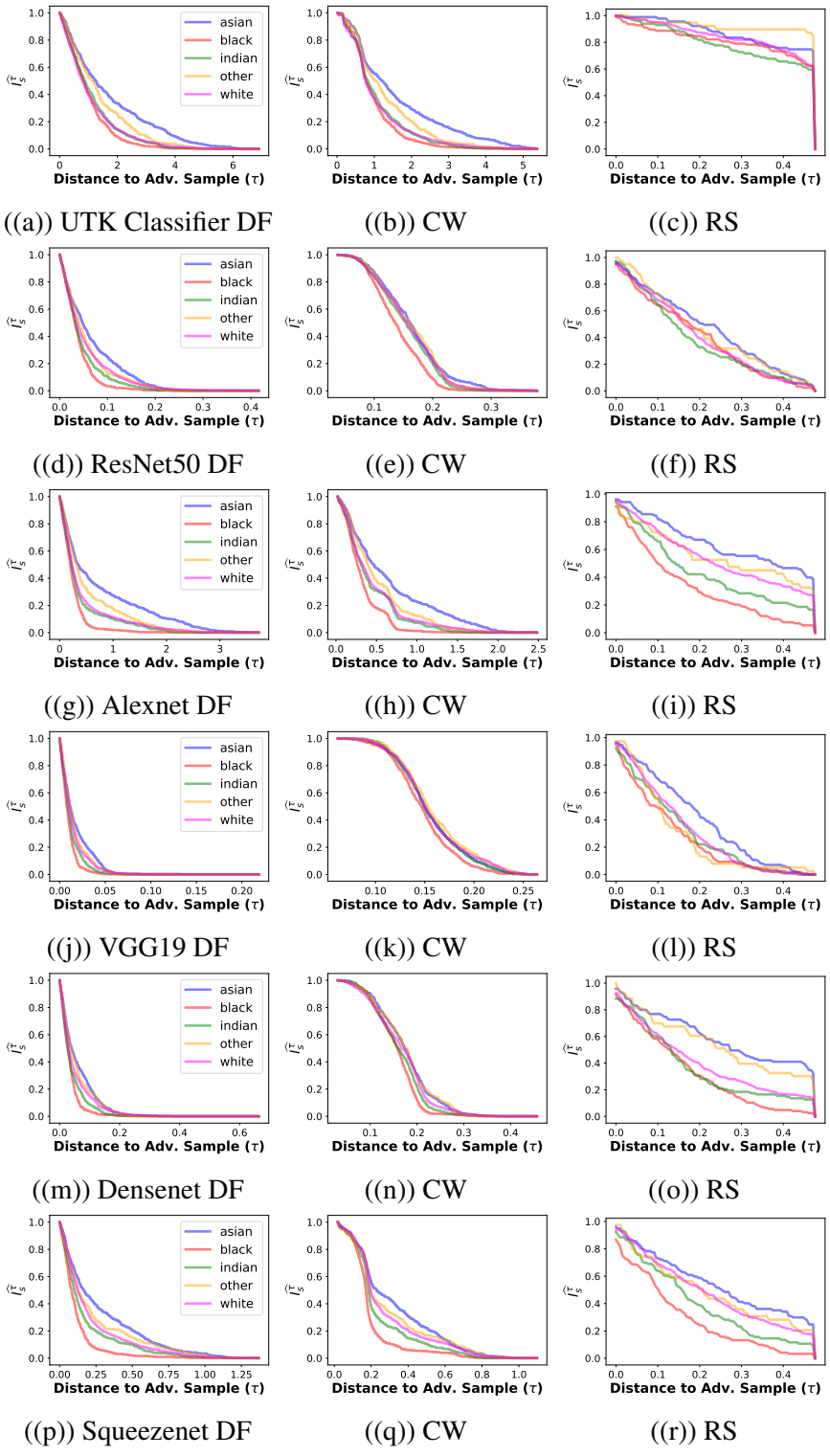
3.7 Evaluation of Robustness Bias using Adversarial Attacks

As described in Section 3.4.1, we argued that adversarial attacks can be used to obtain upper bounds on $d_\theta(x)$ which can then be used to measure robustness bias. In this section, we audit some popularly used models on datasets mentioned in Section 3.5 for robustness bias as measured using the approximation given by adversarial attacks.

3.7.1 Evaluation of $\widehat{I_P^{DF}}$ and $\widehat{I_P^{CW}}$

To compare the estimate of $d_\theta(x)$ by DeepFool and CarliniWagner, we first look at the signedness of $\sigma(P)$, $\sigma^{DF}(P)$, and $\sigma^{CW}(P)$. For a given partition P, $\sigma(P)$ captures the disparity in robustness between points in P relative to points not in P (see Eq 3.3). Considering all 151 possible partitions (based on class labels and sensitive attributes, where available) for all five datasets, both CarliniWagner and DeepFool agree with the signedness of the direct computation 125 times, i.e., $\mathbb{I}_P\left[\text{sign}(\sigma(P)) = \text{sign}(\sigma^{DF}(P))\right] = 125 = \mathbb{I}_P\left[\text{sign}(\sigma(P)) = \text{sign}(\sigma^{CW}(P))\right]$. Further, the mean difference between $\sigma(P)$ and $\sigma^{CW}(P)$ or $\sigma^{DF}(P)$, i.e., $(\sigma(P) - \sigma^{DF}(P))$, is 0.17 for DeepFool and 0.19 for CarliniWagner with variances of 0.07 and 0.06 respectively.

There is 83% agreement between the direct computation and the DeepFool and CarliniWagner



((a)) UTK Classifier DF ((b)) CW ((c)) RS

((d)) ResNet50 DF ((e)) CW ((f)) RS

((g)) Alexnet DF ((h)) CW ((i)) RS

((j)) VGG19 DF ((k)) CW ((l)) RS

((m)) Densenet DF ((n)) CW ((o)) RS

((p)) Squeezenet DF ((q)) CW ((r)) RS

Figure 3.6: UTKFace partitioned by race. We can see that across models, different populations are at different levels of robustness as calculated by different proxies (DeepFool on the left, CarliniWagner in the middle, and Randomized Smoothing on the right). This suggests that robustness bias is an important criterion to consider when auditing models for fairness.

estimates of $\widehat{I_P}$. This behavior provides evidence that adversarial attacks provide meaningful upper bounds on $d_\theta(x)$ in terms of the behavior of identifying instances of robustness bias.

3.7.2 Audit of Commonly Used Models

We now evaluate five commonly used convolutional neural networks (CNNs): Alexnet, VGG, ResNet, DenseNet, and Squeezenet. We trained these networks using PyTorch with standard stochastic gradient descent. We achieve comparable performance to documented state-of-the-art for these models on these datasets. A full table of performance on the test data is described in Table B.2 (Appendix). After training each model on each dataset, we generated adversarial examples using both methods and computed $\sigma(P)$ for each possible partition of the dataset. An example of the results for the UTKFace dataset can be seen in Figure 3.7 [5].

With evidence from Section 3.7.1 that DeepFool and CarliniWagner can approximate the robustness bias behavior of direct computations of d_θ, we first ask if there are any major differences between the two methods. *If DeepFool exhibits adversarial robustness bias for a dataset a model and a class, does CarliniWagner exhibit the same? and vice versa?* Since there are 5 different convolutional models, we have $151 \cdot 5 = 755$ different comparisons to make. Again, we first look at the signedness of $\sigma^{DF}(P)$ and $\sigma^{CW}(P)$ and we see that $\mathbb{I}_P\left[\text{sign}(\sigma^{DF}(P)) = \text{sign}(\sigma^{CW}(P))\right] = 708$. This means there is 94% agreement between DeepFool and CarliniWagner about the direction of the adversarial robustness bias.

To investigate if this behavior is exhibited earlier in the training cycle than at the final, fully-trained model, we compute $\sigma^{CW}(P)$ and $\sigma^{DF}(P)$ for the various models and datasets for trained models after 1 epoch and the middle epoch. For the first epoch, 637 of the 755 partitions

[5] Our full slate of approximation results are available in Appendix B.1

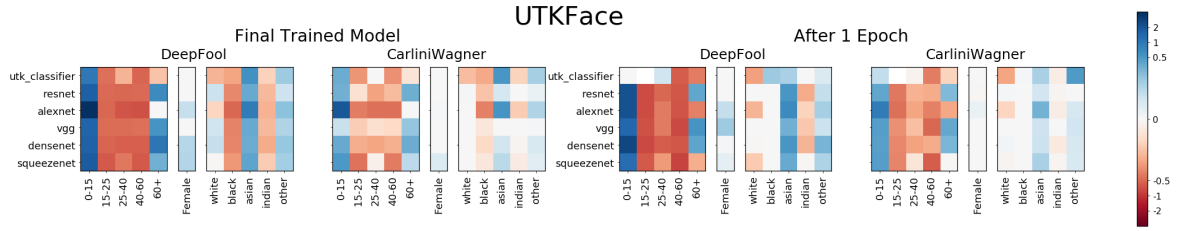


Figure 3.7: Depiction of σ_P^{DF} and σ_P^{CW} for the UTKFace dataset with partitions corresponding to the (1) class labels $\mathcal{C}$ and the, (2) gender, and (3) race/ethnicity. These values are reported for all five convolutional models both at the beginning of their training (after one epoch) and at the end. We observe that, largely, the signedness of the functions are consistent between the five models and also across the training cycle.

were internally consistent, i.e., the signedness of σ was the same in the first and last epoch, and 621 were internally consistent. We see that at the middle epoch, 671 of the 755 partitions were internally consistent for DeepFool and 665 were internally consistent for CarliniWagner. Unsurprisingly, this implies that as the training progresses, so does the behavior of the adversarial robustness bias. However, it is surprising that much more than 80% of the final behavior is determined after the first epoch, and there is a slight increase in agreement by the middle epoch.

We note that, of course, adversarial robustness bias is not necessarily an intrinsic value of a dataset; it may be exhibited by some models and not by others. However, in our studies, we see that the UTKFace dataset partition on Race/Ethnicity does appear to be significantly prone to adversarial attacks given its comparatively low $\sigma^{DF}(P)$ and $\sigma^{CW}(P)$ values across all models.

3.8 Evaluation of Robustness Bias using Randomized Smoothing

In Section 3.4.2, we argued that randomized smoothing can be used to obtain lower bounds on $d_\theta(x)$ which can then be used to measure robustness bias. In this section, we audit popular models on a variety of datasets (described in detail in Section 3.5) for robustness bias, as measured using the approximation given by randomized smoothing.

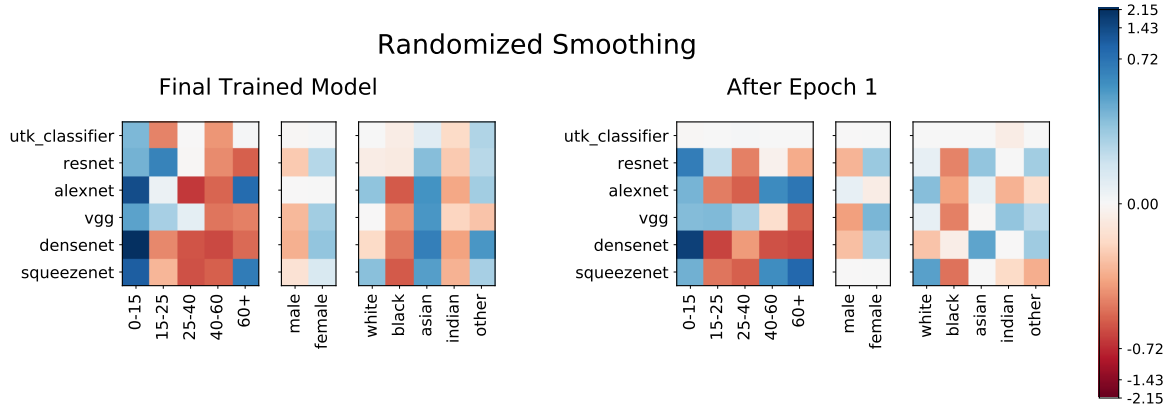


Figure 3.8: Depiction of σ_P^{RS} for the UTKFace dataset with partitions corresponding to the (1) class labels $\mathcal{C}$ and the, (2) gender, and (3) race/ethnicity. A more negative value indicates less robustness bias for the partition. Darker regions indicate a high robustness bias. We observe that the trend is largely consistent amongst models and also similar to the trend observed when using adversarial attacks to measure robustness bias (see Figure 3.7).

3.8.1 Evaluation of $\widehat{I_P^{RS}}$

To assess whether the estimate of $d_\theta(x)$ by randomized smoothing is an appropriate measure of robustness bias, we compare the signedness of $\sigma(P)$ and $\sigma^{RS}(P)$. When $\sigma(P)$ has a positive sign, higher magnitude indicates higher robustness of members of partition P as compared to members not included in that partition P; similarly, when $\sigma(P)$ is negatively signed, higher magnitude corresponds to lesser robustness for those members of partition P (see Eq 3.3). We may interpret shared signedness of both $\sigma(P)$ (where $d_\theta(x)$ is deterministic) and $\sigma^{RS}(P)$ (where $d_\theta(x)$ is measured by randomized smoothing as described in Section 3.4.2) as positive support for the $\widehat{I_P^{RS}}$ measure.

Similar to Section 3.7.1, we consider all possible 151 partitions across CIFAR-10, CIFAR-100, CIFAR-100Super, UTKFace and Adience. For each of these partitions, we compare $\sigma^{RS}(P)$ to the corresponding $\sigma(P)$. We find that their sign agrees 101 times, *i.e.*, $\mathbb{I}_P\left[\text{sign}(\sigma(P)) = \text{sign}(\sigma^{RS}(P))\right] = 101$, thus giving a 66.9% agreement. Furthermore, the mean difference between $\sigma(P)$ and

$\sigma^{RS}(P)$, *i.e.*, $(\sigma(P) - \sigma^{RS}(P))$ is 0.08 with a variance of 0.19.

This provides evidence that randomized smoothing can also provide a meaningful estimate on $d_\theta(x)$ in terms of measuring robustness bias.

3.8.2 Audit of Commonly Used Models

We now evaluate the same models and all the datasets for robustness bias as measured by randomized smoothing. Our comparison is analogous to the one performed in Section 3.7.2 using adversarial attacks. Figure 3.8 shows results for all models on the UTKFace dataset. Here we plot σ_P^{RS} for each partition of the dataset (on x-axis) and for each model (y-axis). A darker color in the heatmap indicates high robustness bias (darker red indicates that the partition is *less* robust than others, whereas a darker blue indicates that the partition is *more* robust). We can see that some partitions, for example, the partition based on class label "40-60" and the partition based on race "black" tend to be less robust in the final trained model, for all models (indicated by a red color across all models). Similarly, there are partitions that are more robust, for example, the partition based on class "0-15" and race "asian" end up being robust across different models (indicated by a blue color). Figure 3.6 takes a closer look at the distribution of distances for the UTKFace dataset when partitioned by race, showing that for different models different races can be more or less robust. Figures 3.6, 3.7 and 3.8 (we see similar trends for CIFAR-10, CIFAR-100, CIFAR-100Super and Adience which we report exhaustively in Appendix B.1) lead us to the following key conclusions:

Dependence on data distribution The presence of certain partitions that show similar robustness trends as discussed above (*e.g.*see final trained model in Figs 3.8 and 3.7, the partitions

by class "0-15" and race "asian" are more robust, whereas the class "40-60" and race "black" are less robust across *all models*) point to some intrinsic property of the data distribution that results in that partition being more (or less) robust regardless of the type decision boundary. Thus we conclude that robustness bias may depend in part on the data distribution of various sub-populations.

Dependence on model There are also certain partitions of the dataset (e.g., based on the classes "15-25" and "60+" as per Fig 3.8) that show varying levels of robustness across different models. Moreover, even partitions that have same sign of $\sigma^{RS}(P)$ across different models have very different values of $\sigma^{RS}(P)$. This is also evident from Fig 3.6 which shows that the distributions of $d_\theta(x)$ (as approximated by all our proposed methods) for different races can be very different for different models. Thus, we conclude that robustness bias is also dependent on the learned model.

Role of pre-training We now explore the role of pre-training on our measures of robustness bias. Specifically, we pre-train five of the six models (Resnet, Alexnet, VGG, Densenet, and Squeezenet) on ImageNet and then fine-tune on UTKFace. We also train a UTK classifier from scratch on UTKFace. Figures 3.8 and 3.7 shows robustness bias scores after the first epoch and in the final, fully-trained model. At epoch 1, we mostly see no robustness bias (indicated by close-to-zero values of $\sigma^{RS}(P)$) for UTK Classifier. This is because the model has barely trained by that first epoch and predictions are roughly equivalent to random guesses. In contrast, the other five models already have pre-trained ImageNet weights, and hence we see certain robustness biases that already exist in the model, even after the first epoch of training. Thus, we conclude that pre-trained models bring in biases due to the distributions of the data on which they were pre-trained and the resulting learned decision boundary after pre-training. We additionally see

that these biases can persist even after fine-tuning.

3.8.3 Comparison of Randomized Smoothing and Upper Bounds

We have now presented two ways of measuring robustness bias: via upper bounds and via randomized smoothing. While there are important distinctions between the two methods, it is worth comparing them. To do this, we compare the sign of the randomized smoothing method and the upper bounds as

$$\mathbb{1}_P \left[\text{sign}(\sigma^{RS}(P)) = \text{sign}(\sigma^{DF}(P))\right]$$

and

$$\mathbb{1}_P \left[\text{sign}(\sigma^{RS}(P)) = \text{sign}(\sigma^{CW}(P))\right].$$

We see that there is some evidence that the two methods agree. The Adience, UTKFace, and CIFAR-10 datasets have a strong agreement (at or above 75%) between the randomized smoothing for both types of upper bounds (DeepFool and CariliniWagner), while the CIFAR-100 dataset has a much weaker agreement (above but closer to 50%) and CIFAR-100Super has an approximately 66% agreement.

It is important to point out that it is not entirely appropriate to perform a comparison in this way. Recall that the upper bounds provide estimates of d_θ using a trained model. However, the randomized smoothing method estimates d_θ not directly with the trained model — instead, it first modifies (smooths) the model of interest and then performs an estimation. Since the upper bounds and randomized smoothing methods are so different in practice, there may be no truly appropriate way to compare the results therefrom. Therefore, too much credence should not be placed on the comparison of these two methods. Both methods indicate the existence of the

robustness bias phenomenon and can be useful in distinct settings.

We present the full results table in Appendix B.3.

3.9 An "Obvious" Mitigation Strategy

Having demonstrated the existence of this robustness bias phenomenon, it is natural to look ahead at common machine learning techniques to address it. In Appendix B.2, we have done just that by adding to the objective function a regularizer term which penalizes for large distances in the treatment of a minority and majority group. We write the empiric estimate of $RB(P, \tau)$ as $\tilde{RB}(P, \tau)$; a full derivation can be found in Appendix B.2.1. Formally,

$$\tilde{RB}(P, \tau) = \left| \frac{1}{\sum_{x \notin P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \notin P \\ y = \hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\} - \frac{1}{\sum_{x \in P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \in P \\ y = \hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\} \right| \tag{3.7}$$

Details of a full implementation of this loss function and the results can be found in Appendix B.2. Experimental results based on that implementation, reported in Appendix B.2.2, support the idea that regularization—a typical approach taken by the fairness in machine learning community—can reduce measures of robustness bias. However, we do believe that this type of experimentation belies the larger point of the present, largely descriptive, work: that robustness bias is a real and significant artifact of popular and commonly-used datasets and models. Surely there are ways to mitigate some of the effects or manifestations of this bias (as we show with our fairly standard regularization-based mitigation technique). However, we believe that any type of mitigation should be taken in concert with the contextualization of these technical systems in the social world, and thus leave "mitigation" research to, ideally, application-specific future work

involving both machine learning practitioners and the stakeholders of particular systems.

Chapter 4: Robustness: Shared Invariances between Machines and Humans

> If you're walking down the right path and you're willing to keep walking, eventually you'll make progress.
>
> *Barack Obama*

An evaluation criterion for safe and trustworthy deep learning is how well the invariances captured by representations of deep neural networks (DNNs) are shared with humans. We identify challenges in measuring these invariances. Prior works used gradient-based methods to generate *identically represented inputs* (IRIs), *i.e.*, inputs which have identical representations (on a given layer) of a neural network, and thus capture invariances of a given network. One necessary criterion for a network's invariances to align with human perception is for its IRIs look "similar" to humans. Prior works, however, have mixed takeaways; some argue that later layers of DNNs do not learn human-like invariances [123] yet others seem to indicate otherwise [124]. We argue that the loss function used to generate IRIs can heavily affect takeaways about invariances of the network and is the primary reason for these conflicting findings. We propose an *adversarial* regularizer on the IRI -generation loss that finds IRIs that make any model appear to have very little shared invariance with humans. Based on this evidence, we argue that there is scope for

improving models to have human-like invariances, and further, to have meaningful comparisons between models one should use IRIs generated using the *regularizer-free* loss. We then conduct an in-depth investigation of how different components (*e.g.* architectures, training losses, data augmentations) of the deep learning pipeline contribute to learning models that have good alignment with humans. We find that architectures with residual connections trained using a (self-supervised) contrastive loss with ℓ_p ball adversarial data augmentation tend to learn invariances that are most aligned with humans.

4.1 Introduction

The ability to train deep neural networks (DNNs) which learn useful features and representations is key for their widespread use [125, 126]. In domains where DNNs are used for tasks that previously required human intelligence (*e.g.* image classification) and where safety and trustworthiness are important considerations, it is helpful to assess the alignment of the learned representations with human perception. Such assessments can help in understanding and diagnosing issues such as lack of robustness to distribution shifts [127, 128], adversarial attacks [129, 130] or using undesirable features for a downstream task [5, 131–134].

One test of human-machine alignment is whether different images that map to identical internal network representations are also judged as identical by humans. To study alignment with human perception, prior works have used the approach of *representation inversion* [124]. The key idea is the following: given an input to a neural network, the approach first finds *identically represented inputs* (IRIs), *i.e.* inputs which have similar representations on some given layer(s) of the neural network. In the second step, the inputs that are perceived similarly by the neural

network are checked by humans for visual similarity. Thus, the approach relies on estimating whether a transformation of the inputs which is representation invariant to a neural network is also an invariant transformation to the human eye, *i.e.* it checks whether models and humans have shared or aligned invariances.

Prior works use gradient-based methods to generate IRIs for a given target input starting with a random seed input. These works revealed exciting insights: (a) Feather et al. studied representational invariance for different layers of DNNs trained over ImageNet data (using the standard cross-entropy loss). They showed that while later layer representations of DNNs do not share any invariances with human perception, the earlier layers are somewhat better aligned with human perception [123]. (b) Engstrom et al. found that, unlike standard DNNs, adversarially robust DNNs, *i.e.*, DNNs trained using adversarial training [1], learn representations that are well aligned with human perception, even in later layers [135]. This was also confirmed by other works [136, 137]. However, some of these findings are contradicted when differently regularized methods are used for generating IRIs, which show that even later layers of DNNs learn human-aligned representations [124, 138, 139].

We seek to make sense of these confusing earlier results and thereby to better understand alignment. We show that when we evaluate the alignment of DNNs' invariances and human perception using IRIs generated using different loss functions, we can arrive at very different conclusions. For example, Fig 4.1 shows how visual similarity of IRIs can vary massively across different categories of losses.

We group existing IRI generation processes into two broad categories: *regularizer-free* (as in [123]), where the goal is to find an IRI without any additional constraints; and *human-leaning* (as in [124, 138, 139]), where the goal is to find an IRI that is also visually human-

comprehensible. Additionally, we propose and explore a new (third) broad category, *adversarial*, where the goal is to find an IRI that is visually (from a human perception perspective) far apart from the target input.

We find that compared to the regularizer-free IRI generation approach, the human-leaning IRI generation approach applies strong constraints on the kind of IRIs generated and thus limits the ability to freely explore the large space of possible IRIs . On the other hand, our proposed adversarial approach shows that in the worst case, all models have close to zero alignment, suggesting that there is scope for improvement in designing models that have human-like invariances (as shown in Fig 4.1 and Table 4.2). Based on this evidence, we argue that in order to have meaningful comparisons between models, one should measure alignment using the regularizer-free loss for IRI generation.

Many prior works do not formally define a measure that can quantify alignment with human perception beyond relying on visual inspection of the images by the authors [124, 135, 138, 139]. We show how alignment can be quantified reliably by designing simple visual perception tests that can be crowdsourced, *i.e.* used in human surveys. We also show how one can leverage widely used measures of perceptual distance [140] to automate our human surveys, which allows us to obtain insights at a scale not possible in previous works.

Next, inspired by the prior works that suggest that changes in the model training pipeline (as in training adversarially robust DNNs [135,136]) can lead to human-like invariant representations, we conduct an in-depth investigation to understand which parts of the deep learning pipeline are critical in helping DNNs better learn human-like invariances. We find that certain choices in the deep learning pipeline can significantly help learn representations that have human-like invariances. For example, we show that residual architectures (*e.g.*, ResNets [119]), when trained

with a self-supervised contrastive loss (*e.g.*, SimCLR [141]), using ℓ_2 ball adversarial data augmentations (*e.g.*, as in RoCL [142]); the learned representations – while typically having lower accuracies than their fully supervised counterparts – have a higher alignment of invariances with human perception. We highlight the following contributions:

- We show how different losses used for generating IRIs lead to different conclusions about a model's shared invariances with human perception, thus leading to seemingly contradictory findings in prior works.
- We propose an adversarial IRI generation loss, using which we show empirically that we can almost always discover invariances of DNNs that do not align with human perception, thus suggesting that there is scope to design better mechanisms to learn representations that are more aligned with human perception.
- We conduct an in-depth study of how loss functions, architectures, data augmentations and training paradigms lead to learning human-like shared invariances.

4.2 Measuring Shared Invariance with Human Perception

Measuring the extent to which invariances learned by DNNs are shared by humans is a two-step process. We first generate IRIs , *i.e.*, inputs that are mapped to identical representations by the DNN. IRIs give us an estimate about the invariances of the DNN. Then, we assess if these inputs are also considered identical by humans. More concretely, if invariances of a given DNN (g_{model}) are shared by humans (g_{human}) on a set of n d-dimensional samples $X \in \mathbb{R}^{n \times d}$, then:

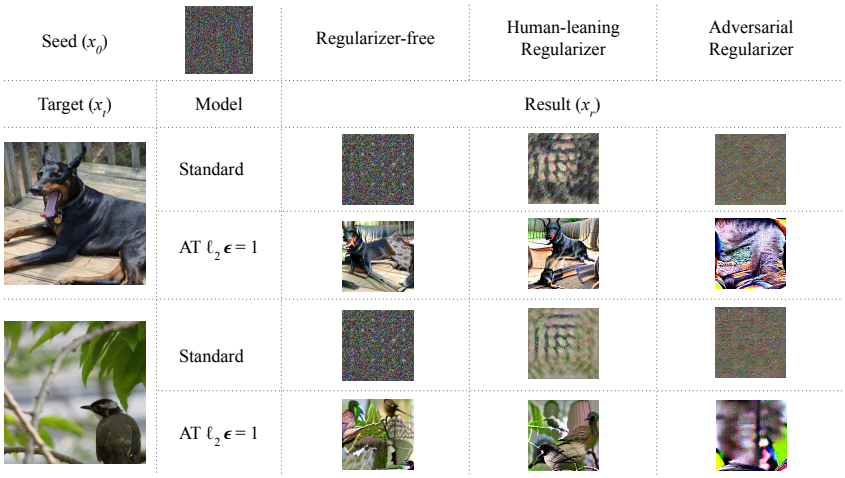


Figure 4.1: **[Representation Inversion for different kinds of $\mathcal{R}$; For ImageNet trained ResNet50]** For the standard ResNet50 (trained using cross-entropy loss), with regularizer-free and adversarial inversion, x_r looks perceptually much closer to x_0 than x_t, even though from the model's point of view, x_r and x_t are the same. However, with the human-leaning regularizer, we see that x_r contains some information like color patterns of x_t. For adversarially robust ResNet50 [1, 6] even though regularizer-free and human-leaning inversions look perceptually similar to x_t, for the adversarial regularizer even these models produce x_r that looks nothing like x_t. Images are generated by starting from x_0 and solving Eq 4.2 with different kinds of regularizers.

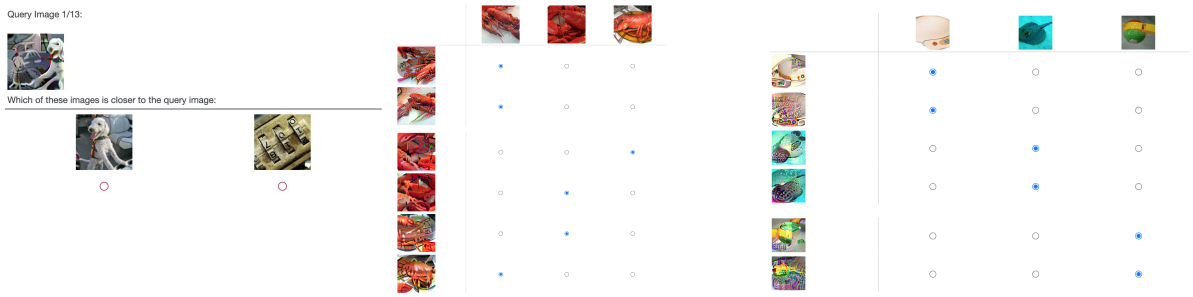
$$g_{\text{human}}(X^i) \approx g_{\text{human}}(X^j) \, \forall \, (X^i, X^j) \in \mathcal{S} \times \mathcal{S} \; ;$$

$$\mathcal{S} = \{X\} \cup \{X^i \mid g_{model}(X^i) \approx g_{model}(X)\}.$$

$\mathcal{S}$ denotes the IRIs for g_{model}. There are three major challenges here:

- Access to representations in the brain, *i.e.*, g_{human} is not available [1].
- Due to the highly non-linear nature of DNNs, $\mathcal{S}$ can be very hard to obtain.
- The fine-grained input space implies very many inputs n, making the choice of X hard.

[1]While some recent works have used fMRI recordings of brain activity [143] to interpret DNNs, these are very expensive to obtain and in this work, we assume we don't have access to representations in the human brain



((a)) 2AFC ((b)) Hard Clustering ((c)) Random Clustering

Figure 4.2: **[Survey Prompts for AMT workers]** In the 2AFC (left) setting we ask the annotator to choose which of the two images (x_t or x_0) is perceptually closer to the query image (x_r). In the clustering setting (center and right) we show 3 images from the ImageNet dataset (target images, x_t) in the columns and for each of these, we generate $x_{r_1} \in \mathcal{S}_{x_t}$ and $x_{r_2} \in \mathcal{S}_{x_t}$. Each of these is shown across the rows. The task here is to match each image on the row with the corresponding target image on the column.

We address each of these below. We also show how prior works that do not directly engage with these points can miss important issues in their conclusions about shared invariances of DNNs and humans.

4.2.1 Approximating g_{human}

Assuming we have a set of images with identical representations ($\mathcal{S}$; how we obtain this is discussed in Section 4.2.2), we must check if humans also perceive these images to be identical. The extent to which humans think this set of images is identical defines how aligned the invariances learned by the DNN are with human perception. In prior works, this has been done by either eyeballing IRIs [135] or by asking annotators to assign class labels to IRIs [123]; both approaches do not scale well. Additionally, assigning class labels to IRIs limits X to being samples from a data distribution containing human-recognizable images (*i.e.*, X cannot be sampled from any arbitrary distribution) with only a few annotations (*e.g.*, asking annotators to assign one class label out of 1000 ImageNet classes is not feasible). To address the issues of scalability and class labels,

we propose the following as a measure of alignment between DNN and human invariances:

$$\text{Alignment} = \frac{|\mathcal{A}|}{\sum_{x_t \in X} |\mathcal{S}_{x_t}|}, \text{where} \tag{4.1}$$

$$\mathcal{A} = \{x_{r_i} |\ ||g_{\text{human}}(x_t) - g_{\text{human}}(x_{r_i})|| <$$

$$||g_{\text{human}}(x_0) - g_{\text{human}}(x_{r_i})||\ \ \forall x_t \in X, x_{r_i} \in S_{x_t}\},$$

$$S_{x_t} = \{x_{r_i} | g_{\text{model}}(x_{r_i}) \approx g_{\text{model}}(x_t)\ \ \forall x_t \in X\},$$

where x_0 is the starting point for Eq 4.2 sampled from $\mathcal{N}(0, 1)$. In Section 4.2.4 we see how alignment is robust to the choice of x_0. By directly looking for perceptual similarity of IRIs (captured by $\mathcal{A}$), we get past the issue of assigning class labels to IRIs . The comparison used to generate $\mathcal{A}$ is referred to as the 2 alternative forced choice test (2AFC) which is commonly used to assess sensitivity of humans to stimuli [144]. In order to compute $\mathcal{A}$, we estimate perceptual distance $d(x_i, x_j) = ||g_{\text{human}}(x_i) - g_{\text{human}}(x_j)||$ between two inputs. Ideally, we would like to measure $d(x_i, x_j)$ by directly asking for human annotations, however, this approach is expensive and does not scale when we wish to evaluate many models. To address scalability, we use LPIPS [140] which is a commonly used measure for perceptual distance and thus can be used to approximate $d(x_i, x_j)$ [2]. While LPIPS is by no means a perfect approximation, it allows us to gain insights at a scale not possible in prior works.

To ensure the efficacy of LPIPS as a proxy for human judgments, we deploy two types of

[2]For all evaluations we report the average over 4 different backbones used to calculate LPIPS including the finetuned weights released by the authors. More details in Appendix C.1.2

surveys on Amazon Mechanical Turk (AMT) to also elicit human similarity judgments. Prompts for these surveys are shown in Fig. 4.2. We received approval from the Ethical Review Board of our institute for this survey. Each survey consists of 100 images plus some attention checks to ensure the validity of our responses. The survey was estimated to take 30 minutes (even though on average our annotators took less than 20 minutes), and we paid each worker 7.5 USD per survey.

Clustering In this setting, we ask humans to match the IRIs (x_{r_i}) on the row to the most perceptually similar image (x_t) on the column (each row can only be matched to one column). A prompt for this type of task is shown in Fig. 4.2(b) & 4.2(c). With these responses, we calculate a quantitative measure of alignment by measuring the fraction of x_{r_i} that were correctly matched to their respective x_t. For ImageNet, we observed that a random draw of three images (*e.g.*, Fig. 4.2(c)) can often be easy to match to based on how different the drawn images (x_t) are. Thus, we additionally construct a "hard" version of this task by ensuring that the three images are very "similar" (as shown in Fig. 4.2(b)). We leverage human annotations of ImageNet-HSJ [145] to draw these similar images. More details can be found in Appendix C.1.

2AFC This is the exact test used to generate $\mathcal{A}$. In this setting, we show the annotator a reconstructed image (x_r) and ask them to match it to one of the two images shown in the options. The images shown in the options are the seed (x_0, *i.e.*, starting value of x in Eq. 4.2) and the original image (x_t). Since x_r and x_t are IRIs for the model (by construction), alignment would imply humans also perceive x_r and x_t similarly. See Fig. 4.2(a) for an example of this type of survey.

4.2.2 Generating IRIs

Even if we assume a finite sampled set $X \sim \mathcal{D}$ (discussed in Section 4.2.3), there can be many samples in $\mathcal{S}$ due to the highly non-linear nature of DNNs. However, we draw on the insight that there is often some structure to the set of IRIs , that is heavily dependent on the IRI generation process. Prior work on understanding shared invariance between DNNs and humans [**?**, e.g.,]]engstrom2019adversarial, jenelle2019metamers has used representation inversion [124] to generate IRIs . However, IRIs generated this way depend heavily on the loss function used in representation inversion, as demonstrated by [138]. Fig. 4.1 shows how different loss functions can lead to very different looking IRIs . We group these losses previously used in the literature to generate IRIs into two broad types: ***regularizer-free*** (used by [123, 135]), and ***human-leaning*** (used by many works on interpretability of DNNs including [124, 138, 139, 146, 147]). We also explore a third kind of ***adversarial*** regularizer, that aims to generate *controversial stimuli* [148] between a DNN and a human.

Representation inversion is the task of starting with a random seed image x_0 to reconstruct a given image $x_t \in X$ from its representation $g(x_t)$ where $g(\cdot)$ is the trained DNN. The reconstructed image (x_r) is same as x_t from the DNN's point of view, *i.e.*, $g(x_t) \approx g(x_r)$. This is achieved by performing gradient descent on x_0 (in our experiments we use SGD with a learning rate of 0.1) to minimize a loss of the following general form:

$$\mathcal{L}_x = \frac{||g(x_t) - g(x)||_2}{||g(x_t)||_2} + \lambda * \mathcal{R}(x) \tag{4.2}$$

where λ is an appropriate scaling constant for regularizer $\mathcal{R}$. All of these reconstructions

induce representations in the DNN that are very similar to the given image (x_t), as measured using ℓ_2 norm. Depending on the choice of seed x_0 and the choice of $\mathcal{R}$, we get different reconstructions of x_t thus giving us a set of inputs $\{x_t, x_{r_1}, ..., x_{r_k}\}$ that are all mapped to similar representations by $g(\cdot)$. Doing this for all $x_t \in X$, we get the IRIs , $\mathcal{S} = \{X, X^{r_1}, ..., X^{r_k}\}$.

In practice we find that the seed x_0 does not have any significant impact on the measurement of shared invariance. However, the choice of $\mathcal{R}$ *does* significantly impact the invariance measurement (as also noted by [138]). We identify the following distinct categories of IRIs based on the choice of $\mathcal{R}$.

Regularizer-free. These methods do not use a regularizer, *i.e.*, $\mathcal{R}(x) = 0$.

human-leaning regularizer. This kind of regularizer purposefully puts constraints on x such that the reconstruction has some "meaningful" features. [124] use $\mathcal{R}(x) = TV(x) + ||x||_p$ where TV is the total variation in the image. Intuitively this penalizes high frequency features and smoothens the image to make it look more like natural images. [147] achieve a similar kind of high frequency penalization by blurring x before each optimization step. We combine both these frequency-based regularizers with pre-conditioning in the Fourier domain [138] and robustness to small transformations [147]. More details can be found in Appendix C.1.4. Intuitively a regularizer from this category generates IRIs that have been "biased" to look meaningful to humans.

Adversarial regularizer. We propose a new regularizer to generate IRIs while intentionally making them look *perceptually dissimilar* from the target, *i.e.*, $\mathcal{R} = -||g_{\text{human}}(x_t) - g_{\text{human}}(x)||$ (negative sign since we want to maximize perceptual distance between x and x_t). We leverage LPIPS (Learned Perceptual Image Patch Similarity) [140], a widely used *perceptual distance* measure, to approximate $||g_{\text{human}}(x_t) - g_{\text{human}}(x)||$. LPIPS uses initial layers of an ImageNet

trained model (finetuned on a dataset of human similarity judgments) to approximate perceptual distance between images which makes it differentiable and thus can be easily plugged into Eq. 4.2. Thus, the regularizer used is $\mathcal{R}(x) = -\text{LPIPS}(x, x_t)$. IRIs generated using this regularizer can be thought of as *controversial stimuli* [148] – they're similar from the DNN's perspective but distinct from a human's perspective.

4.2.3 Choice of inputs X

In order to overcome the challenge of choosing X, we assume X to be sampled from a given data distribution $\mathcal{D}$. In our experiments, we try out many different distributions, including the training data distribution and random noise distributions, and find that takeaways about alignment of models' invariances with humans do not depend heavily on the choice of $\mathcal{D}$. Some examples of X sampled from the data distribution and noise distributions (two random Gaussian distributions, $\mathcal{N}(0, 1)$ and $\mathcal{N}(0.5, 2)$), along with the corresponding IRIs are shown in Fig. C.1, Appendix C.1.3. Interestingly, the human-leaning regularizer, which explicitly tries to remove high-frequency features from x_r fails to reconstruct an x_t that itself consists of high-frequency features.

4.2.4 Evaluation and Takeaways

For each model, we randomly picked 100 images from the data distribution along with a seed image with random pixel values. For each of the 100 images, we do representation inversion using one regularizer each from *regularizer-free*, *human-leaning*, and *adversarial*.

Reliability of using LPIPS Table 4.1 shows the results for the surveys conducted with AMT

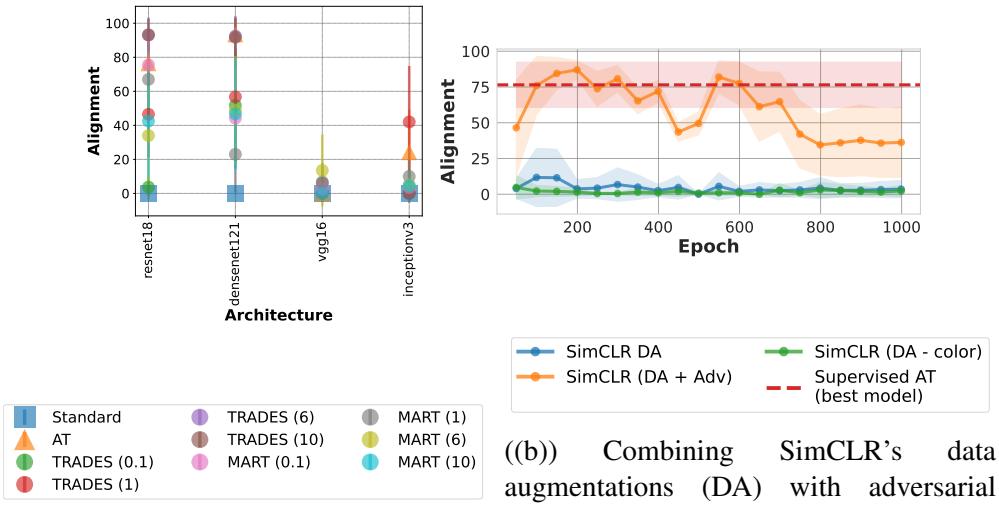
workers [3]. Each survey was completed by 3 workers. For a well-aligned model, the scores under `2AFC` and `Clustering` should be close to 1, while for a non-aligned model scores under `2AFC` should be close to 0, and scores under `Clustering` should be close to a random guess (*i.e.*, about 33%). We see that LPIPS (with different backbone nets, e.g., AlexNet, VGG) orders models similar to human annotators for both survey setups, thus showing that it's a reliable proxy.

Reliability of Human Annotators In Table 4.1, we make three major observations: 1) variance between different annotators is very low; 2) scores under `Human 2AFC` and `Human Clustering` order different models similarly; and finally, 3) even though accuracy drops for the "hard" version of ImageNet task, the relative ordering of models remains the same. These observations indicate that alignment can be reliably measured by generating IRIs and does not depend on bias in annotators. Note that AMT experiments were only performed on IRIs generated using the regularizer-free loss in Eq 4.2.

Impact of regularizer Table 4.2 shows the results of Alignment (Eq 4.1) for different regularizers for IRI generation. We evaluated multiple architectures of both standard and adversarially trained [1] CIFAR10 and ImageNet models. We find that under different types of regularizers, the alignment of models can look very different. We also see that adversarial regularizer makes aligment bad for almost all models, thus showing that for the worst pick of IRIs the alignment between learned invariances and human invariances has a lot of room for improvement. Conversely, the human-leaning regularizer overestimates the alignment.

Impact of X In the case of OOD targets (x_t) we see that humans are still able to faithfully judge similarity, yielding the same ranking of models as in-distribution targets. Some results for human judgments about similarity of IRIs for out-of-distribution samples are shown in Table C.2,

[3] This was conducted only using IRIs from regularizer-free inversion.



((a)) While adversarially robust models generally have high alignment, we see that different architectures. For example, VGG16 has very low levels of alignment despite being trained using robust training losses. Results on CIFAR100 and Imagenet in Appendix C.3.

((b)) Combining SimCLR's data augmentations (DA) with adversarial augmentations (Adv) leads to best alignment (in the early and mid epochs) – in some cases even surpassing the best supervised adversarially robust model. Results for more models and datasets in Appendix C.3.

Figure 4.3: Role of Loss Function in Alignment (left); Role of Training Paradigm in Alignment (right); ResNet18, CIFAR10

Appendix C.1.3. As seen in Fig C.1 (Appendix C.1.3), human-leaning regularizer does not work well for reconstructing noisy targets. This is because such regularizers explicitly remove high-frequency features from reconstructions [138] and thus struggle to meaningfully reconstruct targets that contain high-frequency features. Hence, all results in Table C.2, Appendix C.1.3 are reported on IRIs generated using regularizer-free loss.

Impact of x_0 We repeat some of the experiments with other starting points for Eq 4.2 and find that results are generally not sensitive to the choice of x_0. Results are included in Appendix C.1.5.

4.3 What Contributes to Learning Invariances Aligned with Humans

There have been many enhancements in the deep learning pipeline that have lead to remarkable generalization performance [118, 119, 141, 149–152]. In recent years there have been efforts to understand how invariances in representations learned by such networks align with those of humans [123,153,154]. However, how individual components of the deep learning pipeline affect the invariances learned is still not well understood. Prior works claim that adversarially robust models tend to learn representations with a "human prior" [135–137]. This leads to the question: how do other factors such as architecture, training paradigm, and data augmentation affect the invariances of representations?

We explore these questions in this section. All evaluations in this section are based on regularizer-free IRIs . We chose regularizer-free loss over the adversarial loss as the latter shows worst-case alignment for all models, which is not useful for understanding the effect of various factors in the deep learning pipeline (Appendix C.1.4 shows more results using the adversarial regularizer). Similarly, we preferred regularizer-free over human-leaning loss as the latter has a strong 'bias' enforced by the regularizer. While our approach generalizes to any layer, unless stated otherwise, all measurements of alignment are on the penultimate layer of the network.

4.3.1 Architectures and Loss Functions

We test the alignment of different DNNs trained using various loss functions – standard cross-entropy loss, adversarial training (AT), and variants of AT (TRADES [8], MART [9]). Both TRADES and MART have two loss terms – one each for clean and adversarial samples, which are balanced via a hyperparameter β. We report results for multiple values of β in Fig 4.3(a) and find

that the alignment of standard models (blue squares) is considerably worse than the robust ones (triangles and circles). However, the effect is also influenced by the choice of model architecture, *e.g.*, for CIFAR10, for all robust training losses, VGG16 has significantly lower alignment than other architectures.

4.3.2 Data Augmentation

Hand-crafted data augmentations are commonly used in deep learning pipelines. If adversarial training – which augments adversarial samples during training – generally leads to better aligned representations, then how do hand-crafted data augmentations affect invariances? For adversarially trained models, we try with and without the usual data augmentation (horizontal flip, color jitter, and rotation). Since standard models trained with usual data augmentation show poor alignment (Section 4.3.1), we try stronger data augmentation (a composition of random flip, color jitter, grayscale and gaussian blur, as used in SimCLR [141]) to see if hand-crafted data augmentations can improve alignment. Table C.3 Appendix C.3 shows how hand-crafted data augmentation can be crucial in learning aligned representations for some models (*e.g.*, adversarially trained ResNet18 benefits greatly from data augmentation). In other cases, data augmentation never hurts the alignment. We also see that standard models do not gain alignment even with stronger hand-crafted data augmentations. CIFAR100 and ImageNet results can be found in Table C.4 Appendix C.3 with similar takeaways.

4.3.3 Learning Paradigm

Since data augmentations (both adversarial and hand-crafted) along with residual architectures (like Resnet18) help alignment, self-supervised learning (SSL) models – which explicitly rely on data augmentations – should learn well aligned representations. This leads to a natural question: how do SSL models compare with the alignment of supervised models? SimCLR [141] is a widely used contrastive SSL method that learns 'meaningful' representations without using any labels. Recent works have built on SimCLR to also include adversarial data augmentations [142, 155]. We train both the standard version of SimCLR (using a composition of transforms, as suggested in [155]) and the one with adversarial augmentation on CIFAR10 and compare their alignment with the supervised counterparts. More training details are included in Appendix C.3. Additionally we also train SimCLR without the color distortion transforms – which were identified as key transforms by the authors [141] – to see how transforms that are crucial for generalization affect alignment. Fig 4.3(b) shows the results when comparing self-supervised and supervised learning. We see that SimCLR when trained with both hand-crafted and adversarial augmentations has the best alignment, even outperforming the best adversarially trained supervised model in initial and middle epochs of training. We also see that removing color-based augmentations (DA - color) does not have a significant impact on alignment, thus showing that certain DA can be crucial for generalization but not necessarily for alignment.

4.3.4 Summary

We find that three key components lead to good alignment: architectures with residual connections, adversarial data augmentation using ℓ_2 threat model, and a (self-supervised) contrastive

loss. We leave a more comprehensive study of the effects of these training parameters on alignment for future work.

4.4 Related Work

Robust Models Several methods have been introduced to make deep learning models robust against adversarial attacks [1, 8, 9, 156–161]. These works try to model a certain type of human invariance (small change to input that does not change human perception) and make the model also learn such an invariance. Our work, on the other hand, aims to evaluate what invariances have already been learned by a model and how they align with human perception.

Representation Similarity There has been a long-standing interest in comparing neural representations [7, 15, 162–167]. While these works are related to ours in that they compare two systems of cognition, they assume complete white-box access to both neural networks. In our work, we wish to compare a DNN and a human, with only black-box access to the latter.

DNNs and Human Perception Neural networks have been used to model many perceptual properties such as quality [168, 169] and closeness [140] in the image space. Recently there has been interest in measuring the alignment of human and neural network perception. Roads et al. do this by eliciting similarity judgments from humans on ImageNet inputs and comparing it with the outputs of neural nets [145]. Our work, however, explores alignment in the opposite direction, *i.e.*, we measure if inputs that a network seed the same are also the same for humans. [123, 135] are closest to our work as they also evaluate alignment from model to humans, however as discussed in Section 4.2, unlike our work, their approaches are not scalable, they do not discuss the effects of loss function used to generate IRIs , and they do not contribute to an understanding

of what components in the deep learning pipeline lead to learning human-like invariances.

4.5 Conclusion and Broader Impacts

Our work offers insights into how measures of alignment can vary based on different loss functions used to generate IRIs . We believe that when it is done carefully, measuring alignment is a useful model evaluation tool that provides insights beyond those offered by traditional metrics such as clean and robust accuracy, enabling better alignment of models with humans. We recognize that there are potentially worrying use cases against which we must be vigilant, such as taking advantage of alignment to advance work on deceiving humans. Human perception is complex, nuanced and discontinuous [170], which poses many challenges in measuring the alignment of DNNs with human perception [171]. In this work, we take a step toward defining and measuring the alignment of DNNs with human perception. Our proposed method is a necessary but not sufficient condition for alignment and, thus, must be used carefully and supplemented with other checks, including domain expertise. By presenting this method, we hope for better design, understanding, and auditing of DNNs.

Table 4.1: **[CIFAR10 and ImageNet Surveys To Confirm Efficacy of LPIPS]** We use LPIPS to simulate a human in both 2AFC and Clustering setups described in Section 4.2.1 and compare it with AMT worker's responses. A value close to 33% for clustering means random assignment and indicates no alignment. We see that LPIPS and humans rank models similarly in both 2AFC and clustering setups, thus showing that LPIPS is a reliable proxy for judging perceptual similarity of IRIs . These experiments were conducted on IRIs generated using regularizer-free loss in Eq 4.2. The variance reported for LPIPS is for different backbone networks that are available for LPIPS.

CIFAR10						
	Model	Human 2AFC	LPIPS 2AFC	Human Clustering		LPIPS Clustering
AT ℓ_2 $\epsilon = 1$	ResNet18	$96.00_{\pm 2.55}$	$87.25_{\pm 9.52}$	$97.48_{\pm 1.80}$		$88.13_{\pm 6.57}$
	VGG16	$38.83_{\pm 7.59}$	$4.00_{\pm 3.86}$	$55.39_{\pm 5.63}$		$46.09_{\pm 3.84}$
	InceptionV3	$82.00_{\pm 8.44}$	$54.12_{\pm 19.23}$	$84.47_{\pm 6.32}$		$74.87_{\pm 6.74}$
	DenseNet121	$98.67_{\pm 0.24}$	$91.75_{\pm 8.2}$	$97.64_{\pm 2.08}$		$91.92_{\pm 6.13}$
Standard	ResNet18	$0.17_{\pm 0.24}$	$0.0_{\pm 0.0}$	$38.55_{\pm 1.19}$		$35.35_{\pm 3.27}$
	VGG16	$0.17_{\pm 0.24}$	$0.0_{\pm 0.0}$	$33.84_{\pm 2.70}$		$32.58_{\pm 1.04}$
	InceptionV3	$0.17_{\pm 0.24}$	$0.38_{\pm 0.41}$	$38.38_{\pm 4.06}$		$36.62_{\pm 3.08}$
	DenseNet121	$9.83_{\pm 9.97}$	$0.12_{\pm 0.22}$	$42.42_{\pm 5.02}$		$37.12_{\pm 3.54}$
IMAGENET						
	Model	Human 2AFC	LPIPS 2AFC	Human Clustering	Human Clustering Hard	LPIPS Clustering
AT ℓ_2 $\epsilon = 3$	ResNet18	$93.17_{\pm 5.95}$	$53.37_{\pm 20.19}$	$96.00_{\pm 3.59}$	$87.75_{\pm 7.60}$	$65.28_{\pm 10.58}$
	ResNet50	$99.50_{\pm 0.00}$	$53.63_{\pm 20.64}$	$99.49_{\pm 0.71}$	$97.06_{\pm 3.47}$	$71.21_{\pm 9.93}$
	VGG16	$95.50_{\pm 2.12}$	$59.38_{\pm 21.48}$	$91.75_{\pm 5.22}$	$90.69_{\pm 3.13}$	$70.33_{\pm 9.78}$
Standard	ResNet18	$0.00_{\pm 0.00}$	$1.12_{\pm 1.67}$	$33.33_{\pm 0.00}$	-	$34.60_{\pm 0.56}$
	ResNet50	$5.33_{\pm 7.54}$	$0.38_{\pm 0.41}$	$38.38_{\pm 2.53}$	-	$35.35_{\pm 0.62}$
	VGG16	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$33.96_{\pm 2.00}$	-	$34.47_{\pm 1.49}$

Table 4.2: **[CIFAR10 and ImageNet Model Alignment Results for Different Regularizers]** For different regularizers, we see that ranking of models can look very different. For example, for Adversarially Trained (AT) Resnet18 vs InceptionV3 on CIFAR10, we see that reguarizer-free inversion leads to Resnet18 being significantly more aligned, but the trend is much less pronounced for the human-leaning regularizer. We also find that alignment can vary quite a bit between different architectures – all of which achieve similar clean and robust accuracies.

CIFAR10						
Training	Model	Alignment			Clean Acc.	Robust Acc.
		Reg.-Free	Human-Aligned	Adver-sarial		
AT $\ell_2, \epsilon = 1$	ResNet18	$63.25_{\pm 26.23}$	$79.00_{\pm 21.94}$	$0.33_{\pm 0.47}$	80.77	50.92
	VGG16	$0.25_{\pm 0.43}$	$41.41_{\pm 16.74}$	$1.00_{\pm 1.41}$	79.84	48.36
	InceptionV3	$23.25_{\pm 25.56}$	$64.75_{\pm 24.17}$	$3.00_{\pm 4.24}$	81.57	51.02
	DenseNet121	$82.75_{\pm 20.07}$	$86.25_{\pm 14.50}$	$1.33_{\pm 1.89}$	83.22	52.86
Standard	ResNet18	$0.00_{\pm 0.00}$	$21.09_{\pm 13.51}$	$1.33_{\pm 1.89}$	94.94	0.00
	VGG16	$0.00_{\pm 0.00}$	$21.88_{\pm 14.82}$	$0.00_{\pm 0.00}$	93.63	0.00
	InceptionV3	$0.00_{\pm 0.00}$	$21.88_{\pm 17.54}$	$0.33_{\pm 0.47}$	94.59	0.00
	DenseNet121	$0.00_{\pm 0.00}$	$26.56_{\pm 16.90}$	$0.00_{\pm 0.00}$	95.30	0.00
IMAGENET						
Training	Model	Alignment			Clean Acc.	Robust Acc.
		Reg.-Free	Human-Aligned	Adver-sarial		
AT $\ell_2, \epsilon = 3$	ResNet18	$42.00_{\pm 38.33}$	$46.75_{\pm 39.37}$	$0.33_{\pm 0.47}$	53.12	31.02
	ResNet50	$51.00_{\pm 34.89}$	$45.75_{\pm 37.39}$	$14.00_{\pm 3.74}$	62.83	38.84
	VGG16	$55.50_{\pm 34.14}$	$55.50_{\pm 38.29}$	$11.00_{\pm 3.74}$	56.79	34.46
Standard	ResNet18	$0.00_{\pm 0.00}$	$17.00_{\pm 28.30}$	$0.00_{\pm 0.00}$	69.76	0.01
	ResNet50	$0.00_{\pm 0.00}$	$16.25_{\pm 26.42}$	$0.00_{\pm 0.00}$	76.13	0.00
	VGG16	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	73.36	0.16

Chapter 5: Robustness: Shared Invariances between Machines

> It's not that I'm so smart, it's just
>
> that I stay with problems longer.
>
> ———————————————
>
> *Albert Einstein*

A major challenge in studying robustness in deep learning is defining the set of "meaningless" perturbations to which a given Neural Network (NN) should be invariant. Most work on robustness implicitly uses a human as the reference model to define such perturbations. Our work offers a new view on robustness by using another reference NN to define the set of perturbations a given NN should be invariant to, thus generalizing the reliance on a reference "human NN" to any NN. This makes measuring robustness equivalent to measuring the extent to which two NNs share invariances, for which we propose a measure called STIR . STIR re-purposes existing representation similarity measures to make them suitable for measuring shared invariances. Using our measure, we can gain insights into how shared invariances vary with changes in weight initialization, architecture, loss functions, and training dataset. Our implementation is available at: `https://github.com/nvedant07/STIR`.

5.1 Introduction

As deep neural networks are increasingly deployed in real-world scenarios, robustness of their automatically learned feature representations has emerged as a key desideratum. Prior works have defined many measures of robustness corresponding to different types of synthetically-generated or naturally-occurring perturbations that can be applied to the inputs and their distributions (e.g., adversarial inputs [91, 129, 172] or distributional shifts [128, 173–176]). The common property underlying the different robustness measures is that they all attempt to capture *the extent to which the learned representations remain invariant (i.e., unchanged) under some defined set of perturbations*.

In this chapter, we propose a new way to study, understand, and characterize robustness of neural networks. Our key insight is that the set of input perturbations against which a neural network's robustness is measured, can itself be defined by another *reference* neural network. Specifically, given a reference neural network, we first obtain a set of input perturbations that are imperceptible to the reference network (i.e., find inputs with invariant reference representations), and then check the extent to which representations of other neural networks are invariant to these perturbations. Our proposal allows us to measure relative invariance of two neural network representations and estimate the degree to which the two neural networks share representational invariance. Intuitively, our proposal generalizes the often unstated, but implicit assumption behind all interesting sets of perturbations used in robustness studies today: they are perturbations that are imperceptible to a particular reference neural network, the human brain.

Comparing the representational invariance of two neural networks is an important aspect of determining their representational (or perceptual) alignment. Assessing representational alignment

is crucial for a future society with interacting agents controlled by neural networks (e.g., cars driven by different deep learning systems). Additionally, the ability to measure relative invariance, and therefore robustness, of deep neural network representations can offer insights into interesting questions such as: when updating a model, to what extent are invariances preserved (which may be crucial to regulators for safety assurance)? How does representational invariance vary with the choice of network architectures, loss functions, random weight initialization, and datasets used in the training process?

Our work is inspired by and builds upon previous studies investigating similarity between deep neural network representations [7,163,164,177]. However, we find that existing representational similarity literature focuses *narrowly* on comparing two representations of data samples drawn from a specific input distribution, ignoring representations of data samples outside of the distribution or changes in representation caused by input perturbations for which one of the two representations remains invariant. As such, existing measures of similarity between neural network representations offer *no* insight into their robustness and consequently, their alignment. Nevertheless, we retain the compelling (axiomatic) properties of existing similarity measures, by re-purposing them to measure relative invariance. Specifically, for input perturbations imperceptible to the reference neural network, we quantify the invariance in representations of the other neural network using a popular similarity index called Centered Kernel Alignment (CKA) [7].

To summarize, our key contributions are as follows:

[leftmargin=*]We propose a measure of shared invariances between two representations that is based on the models which generated them. We show that our measure faithfully captures shared invariances between two models where existing measures of representation

similarity (such as CKA) do not work adequately. Our proposal repurposes existing representation similarity measures to measure shared invariance, thus preserving all the (desirable) axiomatic properties of these measures. Using our measure we are able to derive novel insights about the impact of weight initialization, architecture, loss and training dataset on the shared invariances between networks. Our initial results show that our measure is a promising evaluation tool to better understand deep learning. We find that typically the shared invariance between models reduces for later layers, however when trained using adversarial training, the same models end up with higher shared invariances even in later layers. We also see that models with residual connections tend to have high shared invariances among them than between other non-residual models.

5.2 Measuring Representational Robustness

Robustness is defined as "perform(ing) without failure under a wide range of conditions" [178]. Applying the definition in the context of learning models, we can disentangle two distinct requirements for a model to be considered robust: i) the model must produce correct outputs, *i.e.* have high accuracy, and ii) these outputs must be produced consistently for a diverse set of inputs, *i.e.* the outputs of the model must be *invariant* to irrelevant perturbations (changes) in the input. Extensive literature on robust learning [91, 129, 130] suggests that it is quite hard to train models that achieve high accuracy on standard benchmarking datasets and high invariance to irrelevant (adversarially-generated or naturally-occurring) perturbations simultaneously [1, 8, 179]. Reconciling correctness and invariance requirements remains a topic of active research.

Here, we investigate the robustness of neural network representations that are learned

in the process of generating outputs. Specifically, we attempt to quantify the invariance of learned neural network representations to *irrelevant* perturbations in the inputs. Intuitively, a high representational invariance is a necessary (though not sufficient) condition for a neural network model to be robust.

5.2.1 A Relative Invariance Framework

To quantify the invariance of a neural network's representations to irrelevant perturbations to inputs, we need to first define the set of such perturbations. Most of the existing works on robustness use humans as a reference model to define these perturbations. Thus all works on robustness are inherently (and often implicitly) *relative* to a human. We generalize the reliance on a humans as the reference model by assuming the reference model to be another neural network. We can then define the set of irrelevant input perturbations as those changes that do not cause any change in the reference model's representation (i.e., perception) of the inputs. Finally, we quantify how invariant a given neural network's representations are to these irrelevant input perturbations.

Our framework effectively quantifies invariance of one neural network's representation relative to another reference neural network. Our use of a reference neural network model is inspired by how human perception is used as a reference to determine which adversarial perturbations [91] or image corruptions [173, 174] or image transformations [175, 176] do not alter the perception of inputs and are, hence, considered irrelevant perturbations.

5.2.2 Problem Setting

Given a reference neural network $m_1 : \mathbb{R}^m \mapsto \mathbb{R}^{d1}$ and n samples ($X \in \mathbb{R}^{n \times m}$) from a given data distribution (), our goal is to define a measure of how invariant a given target network $m_2 : \mathbb{R}^m \mapsto \mathbb{R}^{d2}$ is to perturbations of samples X that are imperceptible by m_1, *i.e.* do not change their representations according to m_1.

Invariant input perturbations for a reference model (X') In our framework, the invariances that are desirable are determined with respect to a reference model, m_1. To this end, we introduce the notion of *Identically Represented Inputs (IRIs)*. For any given input data point x and a given reference model m_1, IRIs is the set of all data points that are mapped to the same representation as x by m_1. Formally,

$$IRI_{\text{strict}}(x; m_1) = \{x' \mid m_1(x') = m_1(x)\} \tag{5.1}$$

In other words, m_1 is *invariant* to and cannot perceive any difference between x and any $x' \in IRI\ (x; m_1)$. In practice, exact equality is hard to achieve, and thus we relax this formulation so that x and x' are *almost* indistinguishable for m_1, *i.e.*, for a small enough δ,

$$IRI_{\text{relax}}(x; m_1) = \{x' \mid \frac{||m_1(x') - m_1(x)||_2}{||m_1(x)||_2} \leq \delta\} \tag{5.2}$$

Going forward we use the relaxed formulation of IRIs and thus omit the subscript. For each of the n input points $x \in X$, if we pick a x' from $IRI\ (x; m_1)$, we get a corresponding batch of n samples X' such that $m_1(X') \approx m_1(X)$.

Measuring invariance of the target model on (X, X') Assuming we obtain X and X', our key idea is to capture the extent to which m_1 and m_2 share invarances, by measuring the degree to which representations assigned by m_2 to X and X' are *similar*. Specifically, we want to quantify the degree of similarity between two sets of n data representations $Y = m_2(X) \in \mathbb{R}^{n \times d2}$ and $Z = m_2(X') \in \mathbb{R}^{n \times d2}$. To this end, we can make use of any of the existing representation similarity measures (S_r), such as CKA [7], and variants of CCA [163, 164] as they're designed specifically to measure similarity between two sets of representations. This yields a *shared invariance measure,* S_i , which can now be formally defined as:

$$\mathrm{S_i}(m_2|m_1, X, X', \mathrm{S_r}) = \mathrm{S_r}(m_2(X), m_2(X')) \tag{5.3}$$

Note that while the traditional use of S_r , *i.e.*, $\mathrm{S_r}(m_1(X), m_2(X))$ does not measure shared invariance between m_1 and m_2 (we give concrete arguments for the same in Section 5.3.2), our proposed measure (Eq 5.3) shows how existing S_r measures can be repurposed to measure shared invariance.

It's important to note that our measure is a directional one since IRIs are defined relative to the reference model m_1. Other than the reference and target models m_1 and m_2, our measure takes given input points X and a representation similarity measure (S_r) as inputs. We discuss the concrete instantiations of $\mathrm{S_i}$ used in this work in Section 5.3.1. First, however, we describe how to construct X' given X.

5.2.3 Generating IRIs

Operationalizing Eq 5.3, requires the answer to a key question: **how to sample x' from an infinitely large set $IRI\ (x; m_1)$?** We argue that there are two key ways to do this: *arbitrarily* or *adversarially*. We can randomly choose a sample from $IRI\ (x; m_1)$, in which case we get *arbitrary* IRIs , or we can pick x' adversarially with respect to m_2, such that the representations $m_2(x')$ and $m_2(x)$ are farthest apart, in which case we get *adversarial* IRIs . We also show in Table 5.1 that takeaways about shared invariance can vary greatly depending on the choice of arbitrary or adversarial IRIs .

We leverage the key insight that m_1 can map multiple different inputs to the same representation (since m_1 is a highly non-linear deep neural network) which can be found using representation inversion [124]. For a given set of inputs typically drawn from the training distribution $X = [x_1...x_n]$ and a reference model m_1, we generate $X' = [x'_1...x'_n]$ that are all mapped to similar representations as X by m_1, *i.e.* $m_1(X) \approx m_1(X')$. Note that X and X', are, by construction IRIs . This is achieved by performing the following optimization for every $x \in X$:

$$\text{argmin}_{x'} \mathcal{L}(x'), \tag{5.4}$$

Which can be approximated using gradient descent by repeatedly performing the following update (where α is the step size):

$$x' = x' - \alpha * \nabla_{x'} \mathcal{L}. \tag{5.5}$$

A key consideration in solving this optimization is that we must start with some initial value of x', *i.e.* we must choose a seed x' from which to start the gradient descent. Different seeds can

lead to different solutions of x'. We find that in practice randomly picked seeds give fairly stable estimates[1].

Arbitrary IRI In order to simulate an arbitrary sample from $IRI\ (x; m_1)$, we use the following $\mathcal{L}$,

$$\mathcal{L}_{\text{arbitrary}}(x') = ||m_1(x') - m_1(x)||_2.$$

Since this scheme only optimizes for similar representation of x and x' on m_1 ($\forall x \in X$) and starts from a random initial value of x', it simulates a random sample from the (possibly infinitely) large set $IRI\ (x; m_1)$.

Adversarial IRIs We can alter the way we find x' ($\forall x \in X$) such that the resulting (X, X') are still, by definition IRIs but are optimized to generate very distinct outputs on m_2, the model for which we're measuring shared invariance. This can be achieved by using exactly the same procedure as in Eqs 5.4 and 5.5, except we now change $\mathcal{L}$ to:

$$\mathcal{L}_{adv} = ||m_1(x') - m_1(x)||_2 - ||m_2(x') - m_2(x)||_2.$$

Solving for x' using $\mathcal{L}_{adv}$ ensures that the inputs are still similarly represented as x on m_1 ($\forall x \in X$) and thus (X, X') are IRIs . However, this ensures that any measure of S_i on such IRIs will be a worst-case estimate. Such IRIs have been referred to as *controversial stimuli* [148] in existing literature.

[1]Since we deal with images we sample each pixel value as a random integer from $0 - 255$ with uniform probabilities.

Table 5.1: **[STIR faithfully estimates shared invariance]** Here the two ResNet18s in each column are trained on CIFAR10 with different random initializations, holding every other hyperparameter constant. 1.) For two such models trained using the vanilla crossentropy loss (left), interestingly, we find that STIR highlights a lack of shared invariance, whereas CKA overestimates this value; 2.) when both models are trained using adversarial training [1] (middle) STIR faithfully estimates high shared invariance; 3.) Finally STIR is able to show how having a directional measure can bring out the differences when comparing a model trained with vanilla loss and adv training (right), whereas CKA being unidirectional cannot derive these insights. All numbers are computed over 1000 random samples from CIFAR10 training set and averaged over 5 runs.

	RESNET18 VANILLA (m_1), RESNET18 VANILLA (m_2)		RESNET18 AT (m_1), RESNET18 AT (m_2)		RESNET18 AT (m_1), RESNET18 VANILLA (m_2)	
	$m_1 \vert m_2$	$m_2 \vert m_1$	$m_1 \vert m_2$	$m_2 \vert m_1$	$m_1 \vert m_2$	$m_2 \vert m_1$
STIR	$0.605_{\pm 0.013}$	$0.562_{\pm 0.023}$	$0.934_{\pm 0.003}$	$0.939_{\pm 0.002}$	$0.405_{\pm 0.020}$	$0.509_{\pm 0.011}$
STIR $_{adv}$	$0.085_{\pm 0.004}$	$0.064_{\pm 0.004}$	$0.096_{\pm 0.007}$	$0.078_{\pm 0.005}$	$0.054_{\pm 0.004}$	$0.070_{\pm 0.004}$
CKA	$0.967_{\pm 0.000}$		$0.937_{\pm 0.000}$		$0.536_{\pm 0.000}$	
ACC($m_1(X')$, $m_2(X')$)	$0.521_{\pm 0.061}$	$0.347_{\pm 0.029}$	$0.891_{\pm 0.007}$	$0.901_{\pm 0.002}$	$0.140_{\pm 0.028}$	$0.555_{\pm 0.012}$

5.3 Measuring Shared Invariances

5.3.1 STIR , an instantiation of $\mathrm{S_i}$

Using Eqs 5.4&5.5 we can generate k different (X, X'), by repeatedly sampling X k times from a given distribution (typically the training distribution of m_1, *e.g.* the train or test set). Now, we define STIR $(m_2|m_1, X, \mathrm{S_r})$ as follows:

$$\text{STIR } (m_2|m_1, X, \mathrm{S_r}) = \frac{1}{k} \sum_{X'} \mathrm{S_r}(m_2(X), m_2(X')). \tag{5.6}$$

Here, we find X' using representation inversion as described in Section 5.2.3. We call this measure **S**imilarity **T**hrough **I**nverted **R**epresentations — STIR .

When all X' are chosen in an adversarial manner (*i.e.*using $_{adv}$ as described in Section 5.2.3), we can estimate the worst case STIR as:

$$\text{STIR}_{adv}(m_2|m_1, X, \mathrm{S_r}) = \frac{1}{k} \sum_{X'_{adv}} \mathrm{S_r}(m_2(X), m_2(X')). \tag{5.7}$$

Both Equations 5.6 & 5.7 are parametrized by S_r . For our purpose, we use linear Centered Kernel Alignment (CKA) [7] as the similarity measure, *i.e.*S_r = Linear CKA . CKA has been adopted by the community as the standard measure for representation similarity and has been used by many subsequent works to derive important insights about deep learning [177, 180, 181]. CKA also has certain desirable axiomatic properties such as invariance to isotropic scaling and orthogonal transformations, which other methods such as SVCCA [163] and PWCCA [164] do not possess. Importantly, CKA is *not* invariant to every invertible linear transformation, which is not true of SVCCA or PWCCA . We refer to the paper [7] for an extensive discussion on why these are desirable properties for any similarity measure. While CKA, as proposed, can work with any kernel function, results show that Linear CKA works just as well as RBF CKA, so for simplicity we use Linear CKA for all our experiments. Definitions of the similarity metrics listed above can be found in Appendix D.1.

5.3.2 Why Existing Representation Similarity (S_r) Measures Cannot Measure Shared Invariance

While at first glance a representation similarity measure (RSM) such as CKA [7] or one of the others mentioned in section 5.3.1 might look appealing to measure shared invariance, these measures are in fact not suitable for this task, for several reasons.

First, **current RSMs are not designed for capturing invariance.** Their goal is to measure the degree of correlation between sets of points generated by two different models, that are transformed to be as aligned as possible under certain constraints. Measuring invariance, however, requires capturing the degree of difference between points generated *by the same model*, which should be the same.

Second, **RSMs don't interact with the model.** All existing RSMs are evaluated in a two-step process, by first obtaining collections of representations $Y = m_1(X), Z = m_2(X)$ from two models and then computing similarity based on those representations as $\mathrm{S_r}(Y, Z)$, without using the models further. From a causal perspective, an invariance is an intervention on a model's input that does not lead to a change in the model's output. However, a central result in the causality literature is that the effect of interventions cannot be determined from observational data alone [182]. Therefore, any metric that aims to make meaningful statements about model invariance needs to interact with the model to make interventions. STIR does this via the process of representation-inversion, however, existing RSMs are purely observational and therefore cannot properly determine invariances.

Third, **sharing invariance between two models is directional.** If model m_1 is constant, it will share all of model m_2's invariances, but not vice versa if m_2 is not constant itself. Existing

RSMs are not directional and therefore cannot express these relationships.

5.3.3 STIR Faithfully Measures Shared Invariance

Training two models (m_1 and m_2) with different random initializations (holding all other things like architecture, loss and other hyperparameters constant), leads to very "similar" representations on the penultimate layer, as measured by instantiations of S_r such as CKA [7]. We consider two variants of this experiment, where we train two ResNet18 models (on CIFAR10) from different random initializations (keeping everything else the same) with 1.) the standard crossentropy loss ($\mathrm{m}^{(\text{Vanilla})}{}_1$, $\mathrm{m}^{(\text{Vanilla})}{}_2$), and 2.) with adversarial training ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{AT})}{}_2$) [2]. Throughout the chapter, STIR is measured over CIFAR10 training samples with S_r = Linear CKA. Thus, to simplify the notation, we use STIR $(m_1|m_2)$ to mean STIR $(m_1|m_2, X \sim \text{CIFAR10}, \text{LinearCKA})$.

STIR provides insights into shared invariance where CKA fails. Both ($\mathrm{m}^{(\text{Vanilla})}{}_1$, $\mathrm{m}^{(\text{Vanilla})}{}_2$) and ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{AT})}{}_2$) achieve a high similarity score (as measured by CKA , on the penultimate layer of both models). However, we find that such a similarity measurement would be an overestimation of shared invariances between $\mathrm{m}^{(\text{Vanilla})}{}_1$ and $\mathrm{m}^{(\text{Vanilla})}{}_2$ which are much lower when measured as STIR $(\mathrm{m}^{(\text{Vanilla})}{}_1 \,|\, \mathrm{m}^{(\text{Vanilla})}{}_2)$ and STIR $(\mathrm{m}^{(\text{Vanilla})}{}_2 \,|\, \mathrm{m}^{(\text{Vanilla})}{}_1)$, as shown in Table 5.1. Two models trained using adversarial training ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{AT})}{}_2$) should intuitively have more shared invariances since these models were explicitly trained to be invariant to ℓ_2 perturbations. Indeed, we see that these two models have much higher values of STIR $(\mathrm{m}^{(\text{AT})}{}_1 \,|\, \mathrm{m}^{(\text{AT})}{}_2)$ and STIR $(\mathrm{m}^{(\text{AT})}{}_2 \,|\, \mathrm{m}^{(\text{AT})}{}_1)$.

Sanity Checks for STIR For high values of STIR , intuitively we would expect the representations of the model we're evaluating (m_2) to have similar representations on IRIs , *i.e.* $m_2(X) \approx m_2(X')$. Since IRIs , by construction, have similar representation on the reference

[2] trained using ℓ_2 threat model with $\epsilon = 1.0$, see [1] for more details

model (m_1), we expect the predictions of m_2 on X' (pred($m_2(X')$) to agree with predictions of m_1 on X' (pred($m_1(X')$). Similarly, for low STIR values, we'd expect less agreement between pred($m_1(X')$) and pred($m_2(X')$). We see that this relationship holds as lower STIR values for $\mathrm{m}^{(\text{Vanilla})}{}_1$ and $\mathrm{m}^{(\text{Vanilla})}{}_2$ also correspond to less agreement in their predictions on X' and higher STIR values for $\mathrm{m}^{(\text{AT})}{}_1$ and $\mathrm{m}^{(\text{AT})}{}_2$ correspond to higher agreement in their predictions (as shown in the right of each row of Table 5.1). To further corroborate that the estimate of shared invariance given by STIR is justified in being lower for ($\mathrm{m}^{(\text{Vanilla})}{}_1$, $\mathrm{m}^{(\text{Vanilla})}{}_2$) than ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{AT})}{}_2$), we generate *controversial stimuli* [148] for both pairs of models. Details of how to generate these can be found in the Appendix D.3. We find that indeed it's significantly easier to generate controversial stimuli for ($\mathrm{m}^{(\text{Vanilla})}{}_1$, $\mathrm{m}^{(\text{Vanilla})}{}_2$) than for ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{AT})}{}_2$) (see Appendix D.3 for the results).

STIR $_{adv}$ shows that in the worst case, there are almost no shared invariances. When measuring shared invariance in the worst case (using STIR $_{adv}$, Eq 5.7), we see that even in the case of $\mathrm{m}^{(\text{AT})}{}_1$ and $\mathrm{m}^{(\text{AT})}{}_2$ the shared invariance drops close to 0. This is shown in the second row of Table 5.1. Thus, for the rest of the chapter we focus on STIR measured using arbitrary IRIs , since STIR $_{adv}$ gives a very pessimistic estimate of shared invariance and thus is not useful in comparing models.

STIR brings out nuance through directionality. When comparing across training types, *e.g.* ($\mathrm{m}^{(\text{AT})}{}_1$, $\mathrm{m}^{(\text{Vanilla})}{}_2$), we find that directionality of STIR is able to show that invariances of $\mathrm{m}^{(\text{Vanilla})}{}_2$ are not well captured by $\mathrm{m}^{(\text{AT})}{}_1$ (indicated by the low value of STIR $(\mathrm{m}^{(\text{Vanilla})}{}_1 \,|\, \mathrm{m}^{(\text{AT})}{}_2)$). However, STIR $(\mathrm{m}^{(\text{AT})}{}_2 \,|\, \mathrm{m}^{(\text{Vanilla})}{}_1)$ is much higher. We posit that models trained using AT have a "superior" set of invariances and do not possess "bad" invariances that models trained using the vanilla loss exhibit. Thus, when evaluating IRIs from a Vanilla model on a model trained using

AT – one can expect lower shared invariance than in the other direction. STIR can capture these nuances in comparison between models because of its directionality. Measures of representation similarity (S_r) like CKA do not offer any such insights, since they're not directional.

5.4 Using STIR to Analyze Model Updates

One motivation for a shared invariance measure like STIR , as mentioned in Section 5.1, is to monitor how different models "align" with each other. This can then be used to analyze if updates to a model lead to the preservation of invariances learned before the update. We first show that STIR can capture relative differences between model invariances and then use STIR to analyze a simulated model update scenario where we incrementally add more training data.

5.4.1 STIR Captures Relative Robustness

We demonstrate that STIR can capture differences between models of varying degrees of robustness. We know that increasing adversarial robustness increases invariance to ℓ_p ball perturbations. Thus, by construction, if a model m_1 has higher adversarial robustness than another model m_2, then invariances of m_1 should be "superior" to those of m_2 and hence we should see STIR $(m_2|m_1) >$ STIR $(m_1|m_2)$. To empirically test this, we construct three models with varying degrees of adversarial robustness: a model trained using the vanilla crossentropy loss (0%), a model trained using adversarial training (AT) with the usual 10 iterations used to solve the inner maximization of AT [1] (51.5%), and a model trained using AT but with only 1 iteration in the inner maximization loop, which gives adversarial robustness somewhere in the middle (34.6%). Table 5.2 shows comparison between these three models and confirms that

Table 5.2: `ITERS` is the number of iterations in the inner loop of Adversarial Training (AT). Each cell shows STIR $(m_j|m_i)$ where m_j is the model on the column and m_i is the model on the row. CKA numbers are shown in (). The three models have robust accuracies (measured under $\ell_2, \epsilon = 1$) such that $AT, IT = 10 > AT, IT = 1 >$ Vanilla.

	AT IT=10	AT IT=1	VANILLA
AT IT=10 ROB: $51.5_{\pm 0.6}$ CLEAN: $80.4_{\pm 0.4}$	—	$0.93 \pm_{0.01}$ (0.89 ± 0.02)	$0.51 \pm_{0.01}$ (0.56 ± 0.01)
AT IT=1 ROB: $34.6_{\pm 0.6}$ CLEAN: $87.2_{\pm 2.1}$	$0.86 \pm_{0.02}$ (0.89 ± 0.02)	—	$0.52 \pm_{0.01}$ (0.64 ± 0.00)
VANILLA ROB: $0.0_{\pm 0.0}$ CLEAN: $95.0_{\pm 0.1}$	$0.41 \pm_{0.02}$ (0.56 ± 0.01)	$0.34 \pm_{0.05}$ (0.64 ± 0.00)	—

STIR $(m_2|m_1) >$ STIR $(m_1|m_2)$ if adversarial robustness $m_1 > m_2$ (measured here by robust accuracy).

5.4.2 Updating Models With More Data

Models deployed in the real world are continuously updated with more data. In such cases, it may be crucial to understand how a model update at a given timestep shares invariance with the model at the previous timestep. To simulate such a scenario, we train a ResNet18 on CIFAR10 where at each timestep, we add $5k$ training samples, *i.e.*, at timesteps $t = 0, 1, ..., T$, we train the model on $5k, 10k, ..., 45k$ samples (we keep $5k$ samples for a holdout validation set). At each timestep, we train for 100 epochs. Figure 5.1 shows how STIR $(m_t|m_{t-1})$ (and STIR $(m_{t-1}|m_t)$) changes as we progressively add more data. Here m_t is the model at a given timestep and m_{t-1} is the model on the previous timestep. For both AT and Vanilla loss we see STIR $(m_t|m_{t-1})$ and

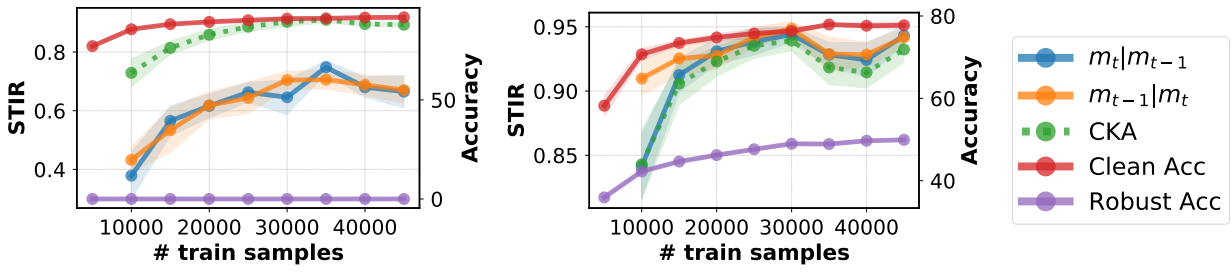
STIR $(m_{t-1}|m_t)$ increase as we add more data. We also see that after a certain amount of data the STIR scores plateau – thus indicating that adding more data has diminishing returns for shared invariance.

5.5 Evaluating the Impact of Design Choices on Shared Invariance using STIR

A lot of research effort has been dedicated to finding architectures, training schemes, and datasets that produce more correct (*i.e.* accurate) models. However, the effect that these design choices have on the relative invariance of models is still not properly understood. In this section, we leverage STIR to investigate the effect that the various choices in the training pipeline have on shared invariances between models. All evaluations of STIR in this section are performed using the CIFAR10 dataset. See Appendix D.2 for additional details.

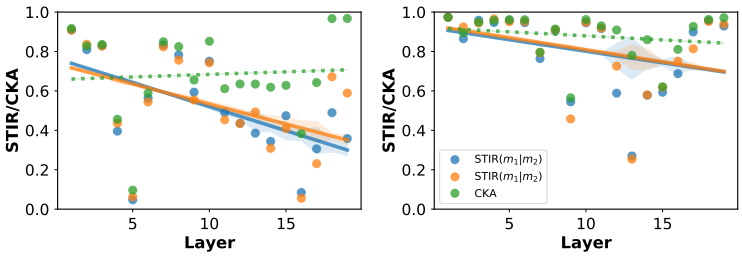
5.5.1 Role of Different Random Initialization, Different Training Datasets

Random Initializations [183] find that the same architecture trained from two different random initializations should converge to highly similar representations, *i.e.*layer k in both models has high similarity. We find that this is not necessarily the case for shared invariances. Fig. 5.2



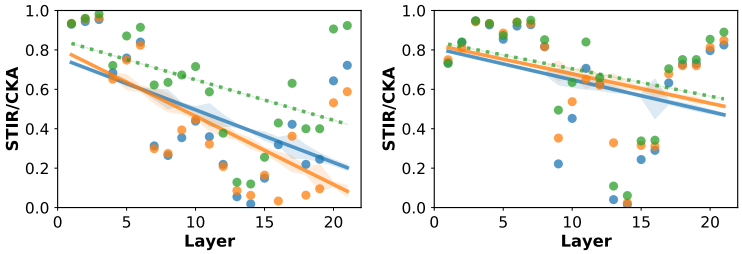
((a)) Standard Training ((b)) Adv Training

Figure 5.1: Adding more training data for (x-axis) leads to a monotonic increase in STIR between the model at a given timestep (m_t) and the previous timestep (m_{t-1}) (in both directions).



((a)) 2 ResNet18 with different init, Vanilla

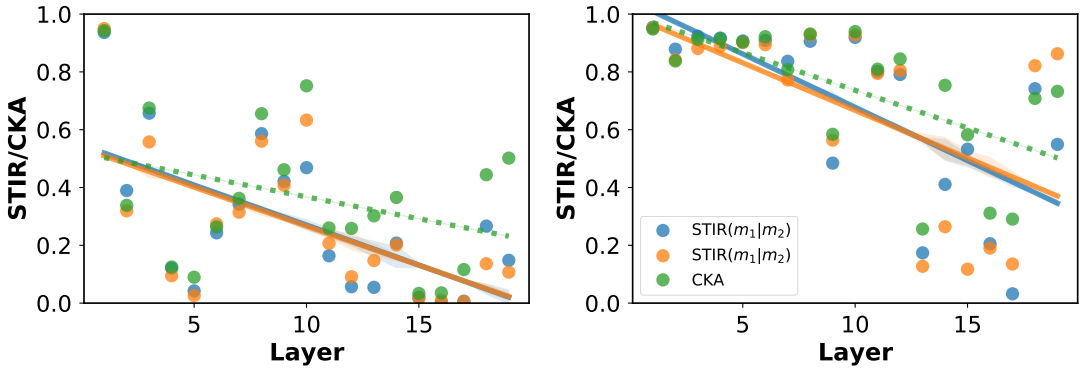
((b)) 2 ResNet18 with different init, AT



((c)) 2 VGG16 with different init, Vanilla

((d)) 2 VGG16 with different init, AT

Figure 5.2: **[Role of Random Initialization; CIFAR10]** For models trained using the Vanilla crossentropy loss, we find that only initial layers have high shared invariances. However, when the training procedure explicitly introduces invariances (*e.g.*adversarial training), then all layers converge to having similarly high shared invariances. Similar trends also hold for deeper variants of ResNet and VGG (results in Appendix D.4).



((a)) m_1 = ResNet18 on CIFAR10 w Vanilla Loss
m_2 = ResNet18 on CIFAR100 w Vanilla Loss

((b)) m_1 = ResNet18 on CIFAR10 w AT
m_2 = ResNet18 on CIFAR100 w AT

Figure 5.3: **[Different Datasets; ResNet18 on CIFAR10 and CIFAR100]** Similar to the finding of [7] we find that across datasets, initial layers tend to have more shared invariances. However, we also find that shared invariances increase substantially for all layers with adversarial training (AT).

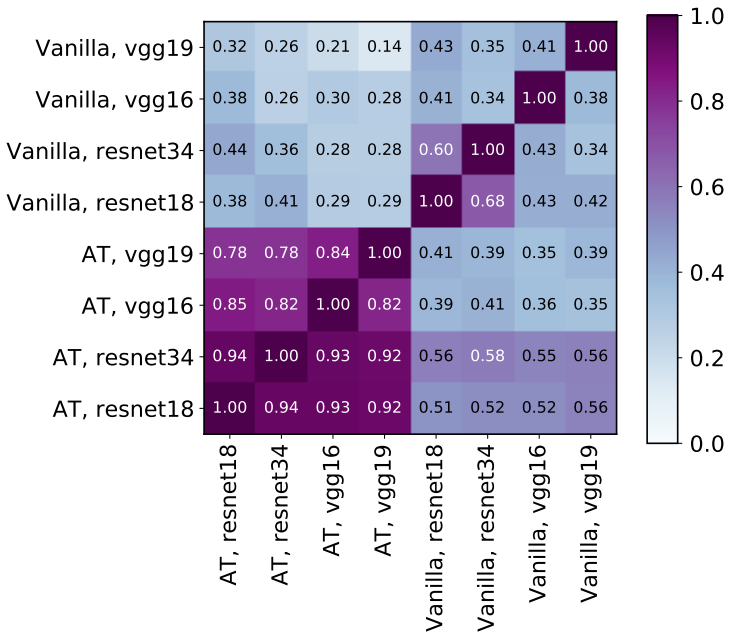
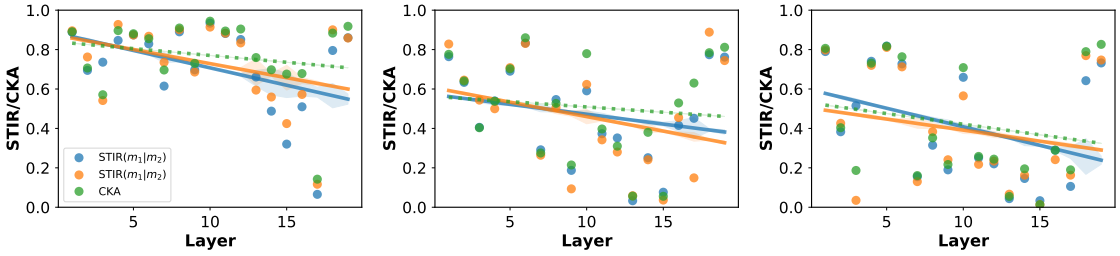


Figure 5.4: **[Different Architectures, Penultimate Layer Shared Invariances; CIFAR10]** We find that in general ResNets have high shared invariances when trained using the same loss, but this shared invariance drops across training types. We also see that with AT, even models with different architectures converge to high shared invariances.



((a)) AT (m_1) TRADES (m_2) ((b)) AT (m_1) MART (m_2) ((c)) TRADES (m_1) vs MART (m_2)

Figure 5.5: **[Different Types of Adversarially Robust Training; ResNet18 on CIFAR10]** Comparing same model trained using 3 different adversarial training variants: AT [1], TRADES [8] and MART [9]. While these methods are geared towards the same goal of achieving invariance to ℓ_p perturbations, we find that other than TRADES and AT, these methods in fact do not have very high shared invariance (as opposed to similarity between two ResNet18s trained using AT, which have much higher values of shared invariance as shown in Fig. 5.2(b)). This shows that these methods achieve resistance to ℓ_p ball attacks in very different ways.

shows results for two ResNet18 (different random initialization) and two VGG16 (different random initialization) models trained on CIFAR10 using the vanilla crossentropy loss and adversarial training. We see that models trained with vanilla loss (Fig. 5.2(a) & 5.2(c)), later layers have

lower shared invariances than initial layers, indicated by a negative slope of lines of best fit. A similar result showing high variance between two classifiers trained with vanilla loss that differ only in their random initialization has been observed for NLP models as well [184], albeit for post-hoc explanations. This suggests a possibly interesting connection between shared invariance and post-hoc explanations which we leave for future work. However, with adversarial training, both initial and final layers converge to (almost) similarly high levels of shared invariances (indicated by flatter lines of fit in Fig. 5.2(b) & 5.2(d)). Thus, we conclude that when training from different random initializations, training procedures that explicitly introduce invariance (such as adversarial training) make each layer of two differently initialized models converge to similar shared invariances.

Datasets For the same architecture (ResNet18) trained on different datasets (CIFAR10 and CIFAR100), we evaluate the shared invariances between each of the corresponding layers. We find that in general, initial layers tend to have higher shared invariances, as indicated by the negative slope of the line of best fit in Fig. 5.3. Interestingly, [7] also had a similar observation when measuring similarity for models trained on different datasets. Additionally, we see that shared invariance increases substantially (for all layers) when these models are trained using adversarial training, similar to the random initialization case, even though the training here is performed on different datasets. We see similar trends for other architectures too (results in Appendix D.4).

5.5.2 Different Architectures, Penultimate Layer

We train ResNet18, ResNet34, VGG16 and VGG19 with two different random seeds and using both the vanilla loss and adversarial training (AT). We then compute the shared invariances across all these configurations of models (for the penultimate layer of each model) as shown in Fig. 5.4.

We find that in general, architectures with residual connections (ResNet18 and ResNet34) have high shared invariances amongst themselves, as indicated by high values of STIR amongst ResNets (for both vanilla and AT).

5.5.3 Different Adversarial Training Methods

As observed in Fig. 5.4, models trained with AT generally have higher values of STIR amongst them than models trained using the vanilla loss. This raises a natural question: does high STIR between models hold for any kind of training that makes models robust to ℓ_p ball perturbations?

To test this, we train a ResNet18 on CIFAR10 using 3 distinct training losses, all of which have been shown to be highly robust to ℓ_p ball perturbations: Adversarial Training (AT) [1], TRADES [8] and MART [9]. TRADES and MART contain an additional hyperparameter β that is used to trade-off between accuracy and adversarial robustness. We use $\beta = 0.1$ for our experiments, and additional details can be found in Appendix D.2. All 3 of these ResNets achieve similar clean and robust accuracy.

Surprisingly, we find that models trained using these methods only have mild levels of shared invariance, as indicated by lines of best fit around 0.5 (see Fig. 5.5). This is in contrast to the case of two ResNet18s trained using AT (see Fig. 5.2(b)) which leads to very high shared

invariance. This shows that the differences shown in Fig. 5.5 are not due to stochasticity in training but rather due to the training methods themselves. Thus, while all of these methods achieve the same goal of robustness in an ℓ_p ball, they achieve so in very different ways. We see similar trends for other architectures too (results in Appendix D.4).

In summary, we find that shared invariance between models tends to decrease with increasing layer depth and adversarial training significantly increases the degree of shared invariance. Models with architectures using residual connections exhibit a higher degree of shared invariance, whereas different methods of adversarial training do not necessarily lead to models with the same shared invariances.

5.6 Related Work

Comparing Representations. A number of papers have proposed methods for measuring the similarity of representations in deep neural networks [7, 162–166]. Our invariance measure leverages CKA [7], however, we discuss in Sections 5.3.2 and 5.3.3 why the existing metrics themselves are unsuitable for measuring shared invariances between models. Recent work has identified a lack of consistency between existing similarity measures and proposed a measure that is consistent [167]. However, their proposed measure also measures similarity and does not take into account the invariances. It can thus be used in conjunction with our proposed measure, but not replace it for measuring invariance. There has been work on measuring shared invariance between NN representations and (black box) humans [185,186], however, we consider a different problem of shared invariances between two NNs.

Understanding Representations. A lot of work on scrutinizing representations has been geared

towards improving the interpretability of neural networks [124,138,187–189]. We're particularly inspired by representation inversion [124,189] which is a key component of our proposed measure. However, while the goal of all of these works is to be able to make a neural network more interpretable to humans, *i.e.* to enable qualitative judgments about the network's behavior, our goal is to instead measure the degree of shared invariance between any two models, *i.e.* to make quantitative statements. Recent work has used model stitching to compare representations [190], but it is focused on better understanding of learned representations, rather than measuring robustness. [191] define disentanglement in representation learning by leveraging the concept of invariance. Their work, however, only provides a definition of disentanglement and no measure for the degree of invariance.

Robustness. Our work takes inspiration from many works on adversarial [1, 91, 134, 140], natural [173, 174] and distributional [127, 128, 192] robustness. However, in all these works, the reference model is (implicitly) assumed to be a human and the goal is to make a neural network follow the invariances of a human. In our work, we explicitly characterize the reference model, which can be another neural network, that allows us to unify all the different notions of robustness.

5.7 Conclusion

We proposed a directional measure of shared invariance between representations that takes into account the invariances of the model that generated these representations. We showed how our measure faithfully estimates shared invariances where existing representation similarity methods may fail. Furthermore, we showed how our measure can be used to derive interesting

insights about deep learning. It will be interesting to explore this direction further in future work, revisiting earlier analysis based on previous representation similarity approaches [177, 180]. Another interesting avenue to explore is how our measure can be used during training to encourage one model to follow similar invariances to another. This could be helpful to update a neural network in safety-critical applications by ensuring that the new network maintains the invariances of the original model.

Chapter 6: Explainability: Diffused Redundancy

I like nonsense; it wakes up the
brain cells.

Dr. Seuss

Representations learned by pre-training a neural network on a large dataset are increasingly used successfully to perform a variety of downstream tasks. In this work, we take a closer look at how features are encoded in such pre-trained representations. We find that learned representations in a given layer exhibit a degree of *diffuse redundancy*, *i.e.*, any randomly chosen subset of neurons in the layer that is larger than a threshold size shares a large degree of similarity with the full layer and is able to perform similarly as the whole layer on a variety of downstream tasks. For example, a linear probe trained on 20% of randomly picked neurons from the penultimate layer of a ResNet50 pre-trained on ImageNet1k achieves an accuracy within 5% of a linear probe trained on the full layer of neurons for downstream CIFAR10 classification. We conduct experiments on different neural architectures (including CNNs and Transformers) pre-trained on both ImageNet1k and ImageNet21k and evaluate a variety of downstream tasks taken from the VTAB benchmark. We find that the loss & dataset used during pre-training largely govern the degree of diffuse redundancy and the "critical mass" of neurons needed often depends on the downstream task, suggesting that there is a task-inherent redundancy-performance Pareto

frontier. Our findings shed light on the nature of representations learned by pre-trained deep neural networks and suggest that entire layers might not be necessary to perform many downstream tasks. We investigate the potential for exploiting this redundancy to achieve efficient generalization for downstream tasks and also draw caution to certain possible unintended consequences. Our code is available at `https://github.com/nvedant07/diffused-redundancy`.

6.1 Introduction

Over the years, many architectures have been proposed (such as [118, 119, 193]) that achieve competitive accuracies on many benchmarks [74]. A key reason for the success of these models is their ability to learn useful representations of data [126]. While these models continue to get better, understanding the properties of underlying learned representations continues to be a challenge.

Prior works have attempted to understand representations learned by deep neural networks through the lens of mutual information between the representations, inputs, and outputs [194] and hypothesize that neural networks perform well because of a "compression" phase where mutual information between inputs and representations decreases. Additionally, recent works have found that many neurons are *polysemantic*, *i.e.*, one neuron can encode multiple "concepts" [139, 195], and that one can then train sparse linear models on such concepts to do "explainable" classification [196]. However, it is not well understood if or how extracted features are concentrated or spread across the entire representation.

While the length of the feature vectors extracted from state-of-the-art networks [1] can vary

[1] Extracted features for the purpose of this chapter refers to the representation recorded on the penultimate layer, but the larger concept applies to any layer

Table 6.1: Different model architectures with varying penultimate layer lengths trained on ImageNet1k. WRN50-2 stands for WideResNet50-2. Implementation of architectures is taken from `timm` [2]. Diffused redundancy here measures what fractions of neurons (randomly picked) can be discarded to achieve within $\delta = 90\%$ performance of the full layer.

Model	Feature Length	ImageNet1k Top-1 Accuracy	Diffused Redundancy for $\delta = 0.9$			
			CIFAR10	CIFAR100	Flowers	Oxford-IIIT-Pets
ViT S-16	384	64.82%	0.70	0.50	0.50	0.80
ViT S-32	384	55.73%	0.70	0.50	0.50	0.70
ResNet18	512	69.23%	0.80	0.50	0.50	0.90
ResNet50	2048	80.07%	0.90	0.50	0.20	0.90
WRN50-2	2048	77.00%	0.95	0.80	0.50	0.95
VGG16	4096	73.36%	0.95	0.80	0.80	0.95

greatly, their accuracy on downstream tasks are not correlated to the size of the representation (see Table 6.1), but rather depend mostly on the inductive biases and training recipes [197, 198]. In all cases, the size of extracted feature vector (*i.e.* number of neurons) is orders of magnitude less than the dimensions of the input and thus allows for efficient transfer to many downstream tasks [125, 199–201]. We show that even using a *random* subset of these extracted neurons is enough to achieve downstream transfer accuracy close to that achieved by the full layer, thus showing that learned representations exhibit a degree of *diffused redundancy* (Table 6.1).

Early works in perception suggest that there are many redundant neurons in the human visual cortex [202] and some later works argued that a similar redundancy in artificial neural networks should help in faster convergence [203]. In this chapter, we revisit redundancy in the context of modern DNN architectures that are trained on large-scale datasets. In particular, we propose the **diffused redundancy** hypothesis and systematically measure its prevalence across different pre-training datasets, losses, model architectures, and downstream tasks. We also show

how this kind of redundancy can be exploited to obtain desirable properties such as generalization performance but at the same time draw caution to certain drawbacks of such an approach in increasing disparity in inter-class performance. We highlight the following contributions:

- We present the diffused redundancy hypothesis which states that learned representations exhibit redundancy that is diffused throughout the layer, *i.e.*, many *random subsets* (of sufficient size) of neurons can perform as well as the entire layer. Our work aims to better understand the nature of representations learned by DNNs.

- We present an initial analysis of why diffused redundancy exists in deep neural networks, we show that a randomly chosen subset of neurons of size k performs as well as the projection of the entire layer on the first k principal components. Intuitively this means that in DNNs' representations, *many* random subsets of size k roughly capture all the variation in data that is possible with k dimensions (since PCA represents directions of maximum variance).

- We propose a measure of diffused redundancy and systematically test our hypothesis across various architectures, pre-training datasets & losses, and downstream tasks.

  - We find that diffused redundancy is significantly impacted by pre-training datasets & loss and downstream datasets.

  - We find that models that are explicitly trained such that particular parts of the full representation perform as well as the full layer, *i.e.*, these models have *structured redundancy* (*e.g.* [10]), also exhibit a significant amount of diffused redundancy. Further, we also evaluate models trained with regularization that decorrelates activation

of neurons and again find that these regularizations surprisingly *do not* affect diffused redundancy. These results suggest that this phenomenon is perhaps inevitable when DNNs have a wide enough final layer.

  - We quantify the degree of diffused redundancy as a function of the number of neurons in a given layer. As we reduce the dimension of the extracted feature vector and re-train the model, the degree of diffused redundancy decreases significantly, implying that diffused redundancy only appears when the layer is wide enough to accommodate redundancy.

- Finally we draw caution to some potential undesirable side-effects of exploiting diffused redundancy for efficient transfer learning that have implications for fairness.

6.1.1 Related Work

Closest to our work is that of Dalvi et al. [204] who also investigate neuron redundancy but in the context of pre-trained language models. They analyze two language models and find that they can achieve good downstream performance with a significantly smaller subset of neurons. However, there are two key differences to our work. First, their analysis of neuron redundancy uses neurons from all layers (by concatenating each layer), whereas we show that such redundancy exists even at the level of a single (penultimate) layer. Second, and perhaps more importantly, they use feature selection to choose the subset of neurons, whereas we show that features are diffused throughout and that even a *random* pick of neurons suffices. Our work also differs by analyzing vision models (instead of language models) and using a diverse set of 30 pre-trained models (as opposed to testing only two models) which allows us to better understand

the causes of such redundancy.

Efficient Representation Learning These works aim to learn representations which are "slim", with the goal of efficient deployment on edge devices [205–207]. Recently proposed paradigm of *Matryoshka Representation Learning* [10] aims to learn nested representations where one can perform downstream tasks with only a small portion of the representation. The goal of such representations is to allow quick, adaptive deployment without having to perform multiple, often expensive, forward passes. These works could be seen as inducing *structured redundancy* on the learned representations, where pre-specified parts of the representation are made to perform similar to the full representation. Our work, instead, aims to look at *diffused redundancy* that arises naturally in the training of DNNs. We carefully highlight the tradeoffs involved in exploiting this redundancy.

Pruning and Compression Many prior works focus on pruning weights [208–214] and how it can lead to sparse neural networks with many weights turned off. Our focus, however, is on understanding redundancy at the neuron level, without changing the weights. Work on structured pruning is more closely related to our work [215, 216], however, a key focus of these works is to prune channels/filters from convolution layers. Our work is more focused on understanding the nature of learned features and is more broadly applicable to all kinds of layers and models. We additionally focus on randomly pruning neurons, whereas structured pruning methods perform magnitude or feature-selection-based pruning.

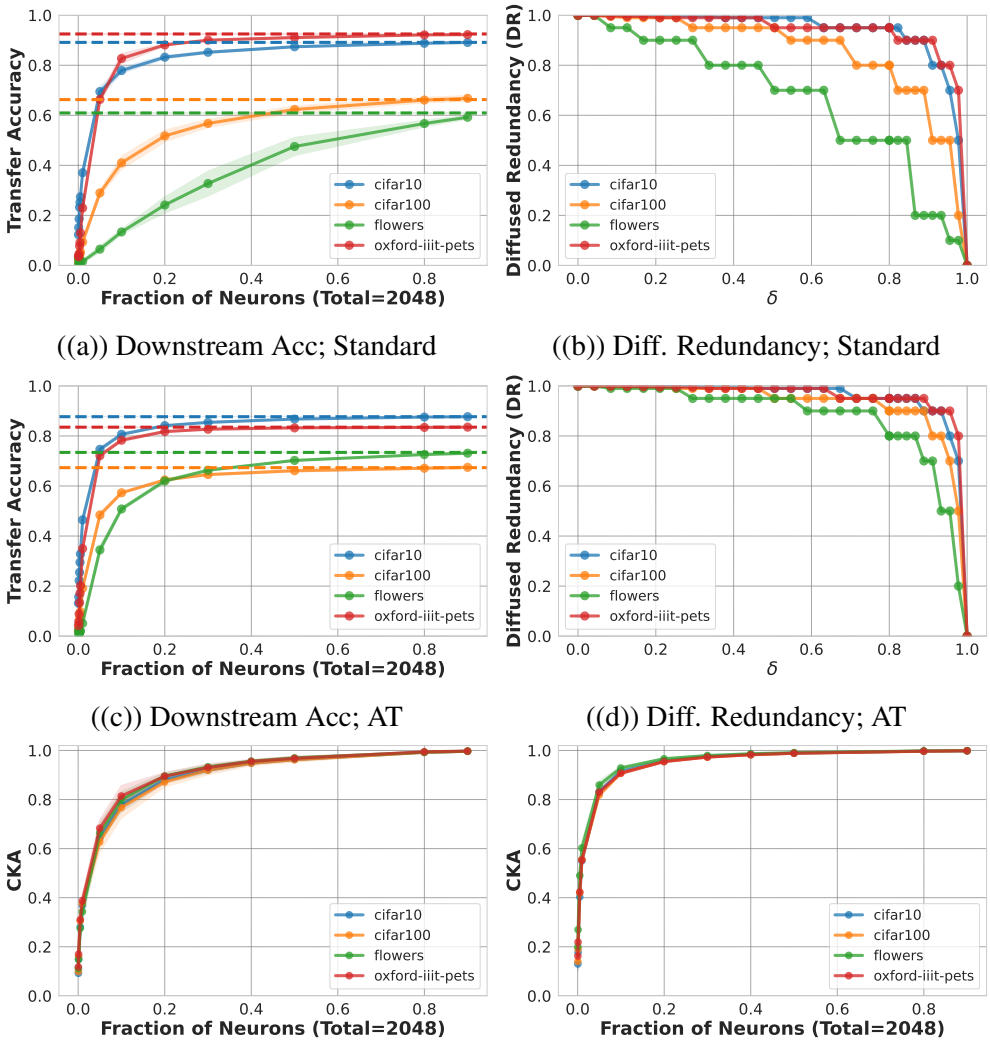
Explainability/Interpretability Many works aim to understand learned representations with the goal of better explainability [124, 138, 139, 187, 195, 217–219]. Two works in this space are especially related to our work: sparse linear layers [196] which show that one can train sparse linear layers on top of extracted features from DNNs; and concept bottleneck models [220] which

explicitly introduce a layer in which each neuron corresponds to a meaningful semantic concept. Both these works explicitly optimize for small/sparse layers, whereas our work shows that similar "small" layers already exist in pre-trained networks, and in fact, can be found simply with random sampling.

Understanding Deep Learning A related concept is that of instrinsic dimensionality of DNN landscapes [221]. Similar to our work, intrinsic dimensionality also requires dropping random parameters (weights) of the network. We, however, are concerned with dropping individual neurons. Other works on understanding deep learning [194, 222] have also looked at the learned features, however, none of these works analyze the redundancy at the neuron level. Another related phenomenon to diffused redundancy is that of neural collapse [223], which states that representations of the penultimate layer "collapse" to K points (where K = number of classes in the pre-training dataset). This implies that for perfectly collapsed representations we need to store just enough information in the final layer activations to be able to represent K different points. We interestingly show that this information is spread throughout the layer with a significant degree of redundancy.

6.2 The Diffused Redundancy Phenomenon

Prior observations about a *compression* phase [194] and neural collapse [223] suggest that the representations need not store a lot of information about the input. These findings imply that all neurons in a learned representation might not be necessary to capture all the information in a particular layer. Extending these observations, we propose the *diffused redundancy* hypothesis:



((a)) Downstream Acc; Standard

((b)) Diff. Redundancy; Standard

((c)) Downstream Acc; AT

((d)) Diff. Redundancy; AT

((e)) CKA; Standard

((f)) CKA; AT

Figure 6.1: **[Testing For Diffused Redundancy in ResNet50 Pre-trained on ImageNet1k]** Top: transfer accuracies + Diffused Redundancy (DR) measure (Eq 6.1) on different downstream datasets, dotted horizontal line shows accuracy obtained using the full layer. We see that accuracy obtained using parts of representation varies greatly with pre-training loss (much more diffused redundancy in Adversarially Trained (AT) ResNet), but also depends on the downstream dataset. Bottom: comparing CKA between a randomly chosen fraction of neurons to the whole layer. Here we evaluate CKA on samples from different datasets and find that similarity of a subset of layer rapidly increases, reaching a similarity of greater than 90% on the adversarially trained ResNet with only 10% randomly chosen neurons. Values are averaged over 5 random picks and error bars show std. dev.

Learned features are diffused throughout a given layer with redundancy such that there exist many randomly chosen subsets of neurons that can achieve similar performance to the whole layer for a variety of downstream tasks.

Note that our hypothesis has two related but distinct parts to it: 1) redundancy in learned features, and 2) diffusion of this redundancy throughout the extracted feature vector. *Redundancy* refers to features being replicated in parts of the representation so that one can perform downstream tasks with parts of representation as well as with the full representation. *Diffusion* refers to this redundancy being spread all over the feature vector (as opposed to being structured), *i.e.*, *many* random subsets (of sufficient size) of the feature vector perform equally well.

In order to evaluate the *redundancy* part of the diffused redundancy hypothesis we use two tasks: 1) representation similarity between randomly chosen subsets of a representation with the whole representation, and 2) transfer accuracy on out-of-distribution datasets (using a linear probe) of randomly chosen subsets of the representation compared to the whole representation. To estimate *diffusion*, we run each check for redundancy over multiple random seeds and plot the standard deviation over these runs.

Representation Similarity of Part vs Whole Centered Kernel Alignment (CKA) is a widely used representation similarity measure and takes in two representations of n data points $Z \in \mathbb{R}^{n \times d_1}$ and $Y \in \mathbb{R}^{n \times d_2}$ and gives a similarity score between 0 and 1 [7]. Intuitively, CKA (with linear kernel, see Appendix E.1 for details about CKA) measures if the two representations rank the n points similarly (where similarity is based on cosine distances). For a given neural network g and n samples drawn from a given data distribution, *i.e.*, $X \sim \mathcal{D}$, let $g(X)$ be the (penultimate) layer representation. If m is a boolean vector representing a subset of neurons in $g(X)$, then we aim to measure CKA($m \odot g(X), g(X)$) to estimate how much redundancy exists in the layer. If indeed CKA($m \odot g(X), g(X)$) is high (*i.e.* close to 1) then it's a strong indication that the diffused redundancy hypothesis holds.

Downstream Transfer Performance of Part vs Whole A commonly used paradigm to measure

the quality of learned representations is to measure their performance on a variety of downstream tasks [199, 224]. Here, we attach a linear layer (h) on top of the extracted features of a network (g) to do classification. This layer is then trained using the training dataset of the particular task (keeping g frozen). If features were to be diffused redundantly then accuracy obtained using $h \circ g$, *i.e.*linear layer attached to the entire feature vector, should be roughly the same as $h' \circ (m \odot g)$; where m is a boolean vector representing a subset of neurons extracted by g, and h & h' are independently trained linear probes.

For both tasks, *i.e.* representation similarity and downstream transfer performance, we evaluate on CIFAR10/100 [113], Oxford-IIIT-Pets [225] and Flowers [226] datasets, from the VTAB benchmark [224]. Training and pre-processing details are included in Appendix E.2.

Measure of Diffused Redundancy In order to rigorously test our hypothesis, we define a measure of diffused redundancy (DR) for a given model (g) with $\mathcal{M}$ being a set of all possible boolean vectors of size $|g|$, *i.e.* size of the representation extracted from g. Each vector $m \in \mathcal{M}$ represents a possible subset of neurons from the entire layer. This measure is defined on a particular task (T) as follows:

$$DR(g, T, \delta) = 1 - \frac{\min\ f\ \text{, s.t. } \frac{1}{|\mathcal{M}_f|} \sum_{m \in \mathcal{M}_f} \frac{T(m \odot g)}{T(g)} \geq \delta}{|g|}, \tag{6.1}$$

$$\mathcal{M}_f = \left\{ m \in \mathcal{M} | \sum_i m_i = f\ ;\ m \in \{0,1\}^{|g|} \right\}$$

Here $T(.)$ denotes the performance of the model inside $()$ for the particular task and δ is a user-defined tolerance level. For the task of representation similarity $T(m \odot g)$ is CKA between a subset of neurons denoted by $m \odot g$ and g, and $T(g)$ is always 1, since it denotes CKA between g and g. For downstream transfer performance, $T(m \odot g)$ is the test accuracy obtained by training

a linear probe on the portion of representation denoted by $m \odot g$ and $T(g)$ is the test accuracy obtained using the full representation. For $\delta = 1$, this measure tells what fraction of neurons could be discarded to exactly match the performance of the entire set of neurons. A higher value of DR denotes that only a few random neurons were needed to match the task performance of the full set of neurons, and thus indicates higher redundancy. Since $\mathcal{M}$ contains an exponential number of vectors ($2^{|g|}$), precisely estimating this quantity is hard. Thus, we first choose a few f (number of neurons to be chosen) to define subsets of $\mathcal{M}$. Then for each $\mathcal{M}_f$ we randomly select 5 samples.

6.2.1 Prevalence of Diffused Redundancy in Pre-Trained Models

Figure 6.1 checks for diffused redundancy in the penultimate layer representation of two types of ResNet50 pre-trained on ImageNet1k: one using the standard cross-entropy loss and another trained using adversarial training [1] (with ℓ_2 threat model and $\epsilon = 3$) [2]. We check for diffused redundancy using both tasks of representation similarity and downstream transfer performance.

Redundancy This is indicated along the x-axis of Fig 6.1, *i.e.*, redundancy is shown when some small subset of the full set of neurons can achieve almost as good performance as the full set of neurons. When looking at downstream task performance (Figs 6.1(a)&6.1(c)), in order to obtain performance within some $\delta\%$ of the full layer accuracy (dotted lines), the fraction of neurons that can be discarded are task-dependent, *e.g.* across both training types we see that flowers (102 classes) and CIFAR100 (100 classes) require more fraction of neurons than CIFAR10 (10

[2] We also report results for $\ell_\infty, \epsilon = 4/255$ in Appendix E.4 and generally find that the ℓ_2 models exhibit higher diffused redundancy. We choose to study adversarially robust models since they've been shown to transfer better [6].

classes) and oxford-iiit-pets (37 classes), perhaps because both these tasks have more classes. Additionally, across all datasets, the model trained with adversarial training exhibits more diffused redundancy than the one trained with standard loss (Fig 6.1(d)& 6.1(b) respectively), meaning we can discard far more neurons for the adversarially trained model to reach close to the full layer accuracy. Interestingly when looking at CKA between part of the feature vector with the full extracted vector (Figs 6.1(e)&6.1(f)), we do not see a significant difference in trends when evaluating CKA on samples from different datasets. However, we still see that we can achieve a given level of CKA with far fewer fraction of neurons in the adversarially trained ResNet50 as compared to the usually trained ResNet50.

***Diffused* Redundancy** This is indicated by small error bars in Figs 6.1(a)&6.1(c)&6.1(e)&6.1(f). If redundancy were instead very structured, then different random picks of neurons would have high variance, however, the error bars here are very low, showing that performance is very stable across different random picks, thus indicating that redundancy is diffused throughout the layer.

While both tasks of downstream transfer and CKA between part and whole indicate higher diffused redundancy for the adversarially trained model, we see that downstream transfer performance can differ substantially based on the dataset (while CKA remains fairly stable across the same datasets), indicating that downstream performance turns out to be a "harder" test for diffused redundancy. Thus, in the rest of the chapter, we examine diffused redundancy through the lens of downstream transfer performance and include CKA results in Appendix E.1.

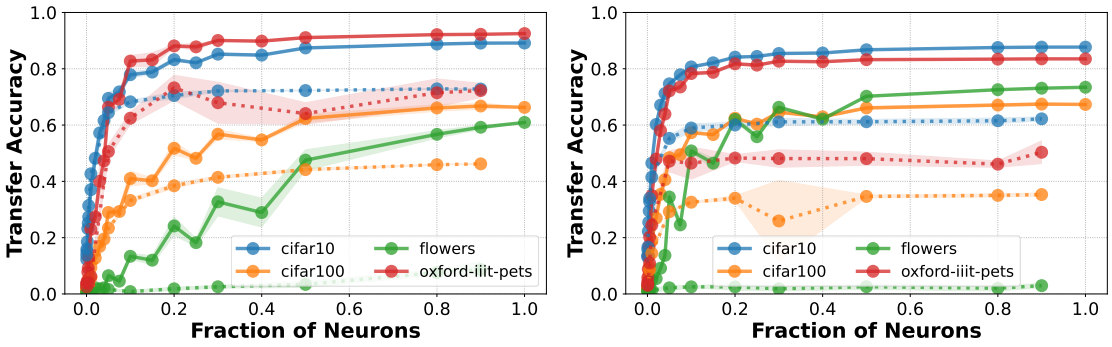


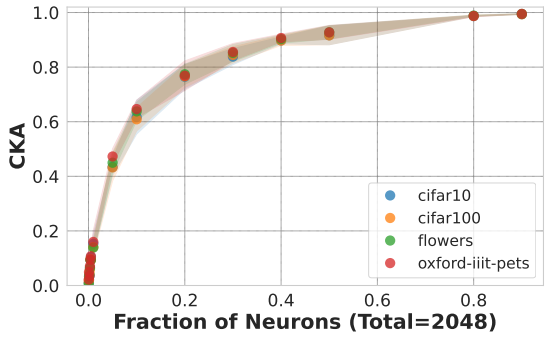
Figure 6.2: **[Random Projection (dotted) vs Randomly Choosing Neurons (solid)]** Standard (top) vs Adversarial (bottom) Training. We find that diffused redundancy performs significantly better than random projections. This indicates that the "constrained" projection offered by diffused redundancy is crucial in achieving good downstream performance.

6.2.2 Understanding Why *Many* Random Subsets Work

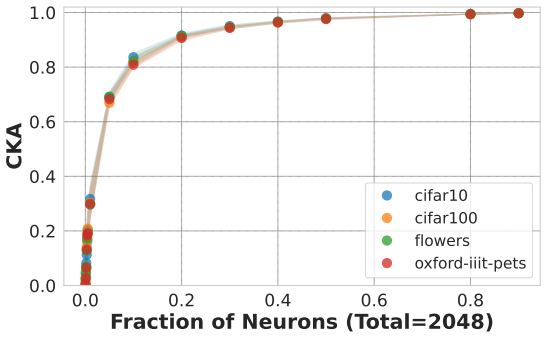
Many prior works explicitly train models to have "small" representations (*e.g.* [10, 205–207] with the goal of efficient downstream learning. These works show that, when explicitly optimized, networks can perform downstream classification with fewer neurons than typically used in state-of-the-art architectures. We show, however, that such subsets already exist in models that are *not* explicitly trained for this goal, and in fact, one doesn't even have to try hard to find this subset; it can be *randomly* chosen. Later in section 6.3.3 we compare some of these efficient representation learning methods to randomly chosen subsets and carefully analyze the tradeoffs involved. Here, however, we seek to better understand why there exist so many randomly selected subsets that work just as well as the whole layer.

High CKA between two random subsets of the same size. First, we use representation similarity (CKA) [7] to get a sense of how two random picks of neurons (of the same size) relate to each other. Concretely, we calculate CKA between two random picks of $k\%$ neurons in the penultimate layer (averaged over 10 such randomly picked pairs) on samples taken from different

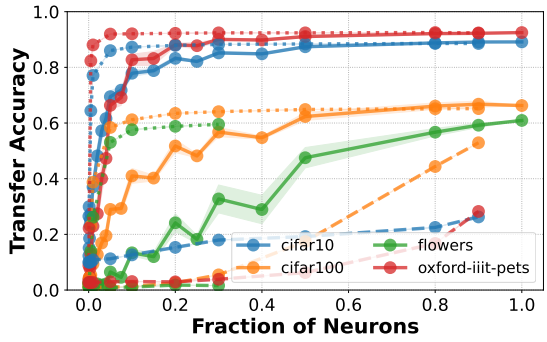
datasets. Fig 6.3(a) & 6.3(b) show CKA results averaged over these different picks of pairs of subsets of the full set of neurons. We see that after picking a certain threshold, *i.e.* for a large enough value of k, the similarity between any two randomly picked pairs of heads is fairly high. For example, for the adversarially trained ResNet50 (Fig 6.3(b)), we observe that any 10% of neurons picked from the penultimate layer are highly similar (CKA of about 0.8), with very low error bars. A similar value of CKA is obtained with 20% of neurons for the standard ResNet50 model. These results indicate that, given a sufficient size, picking any random subset of that size



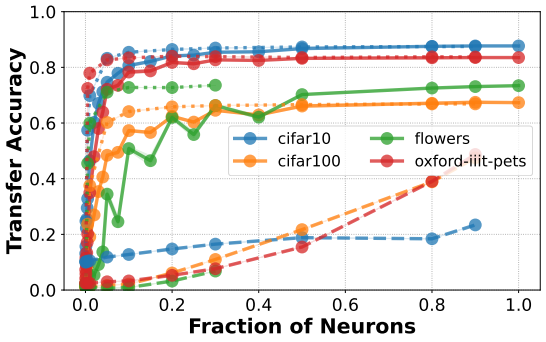
((a)) Standard Training; CKA between two random subsets



((b)) Adversarial Training; CKA between two random subsets



((c)) Standard Training; PCA top (dotted) and bottom (dashed)



((d)) Adversarial Training; PCA top-k (dotted) and bottom-k (dashed)

Figure 6.3: **[Why Many Random Subsets Work]** (6.3(a)& 6.3(b)) Similarity between any two randomly picked sets of $k\%$ neurons becomes fairly high (for a "critical mass" of $k\%$), thus showing that any random pick beyond this threshold is likely to perform similarly; (6.3(c)& 6.3(d)) Performance of a linear probe trained for a downstream task on randomly chosen neurons (solid lines) closely follows that of a linear probe trained on a projection on top k principal components (dotted lines) for a sufficiently large value of k.

has very similar representations and thus provides an initial intuition for why almost *any* random subset, with high probability, could work equally well.

Downstream performance on k randomly chosen neurons closely follows performance on top k principal components. While high representation similarity is a necessary condition for two representations to have similar downstream performance, it's not a sufficient one. As seen earlier in Fig 6.1, similar values of CKA can still show different downstream accuracy. Thus, to further understand why any random pick of neurons achieves similar accuracy, we compare a randomly chosen subset of neurons with a projection on the same number of top k principal components. Fig 6.3(c)& 6.3(d) shows that the performance of a linear probe trained on a random subset of neurons initially lags behind the top PCA dimensions for small values of $k(< 20\%)$, but for higher values ($> 20\%$) closely follows the performance on a probe trained on the projection on the same number of top principal components. This indicates that a sufficiently sized random subset of size k (roughly) captures the maximum variance in data that can be captured in k dimensions (upper bound by top k principal components since they are designed to capture maximum variance). As a sanity check, we also show the results for the bottom k principal components (directions of least variance) and see that randomly chosen neurons perform significantly better.

Random projections into lower dimensions do not perform well on downstream tasks. The presence of diffused redundancy in a particular layer indicates that a dimension lower than the layer's size suffices to perform many downstream tasks. However, randomly sampling neurons (as we do in our experiments on diffused redundancy) can be seen as one very particular way of reducing dimension that is restricted by what the network has learned. This raises a natural question, would our observation extend to less restricted ways of reducing dimensions? To

test this, we compare diffused redundancy *i.e.*, a linear probe trained on a random sample of k neurons to a linear probe trained on a random projection of the layer's activations. To project a d dimensional layer into a lower dimension (k) we multiply it by a (normalized) randomly sampled matrix from a normal distribution $\mathbb{R}^{d \times k}$. We report our results over 5 seeds in Fig 6.2 and find that diffused redundancy significantly outperforms random projection.

6.3 Factors Influencing The Degree of Diffused Redundancy

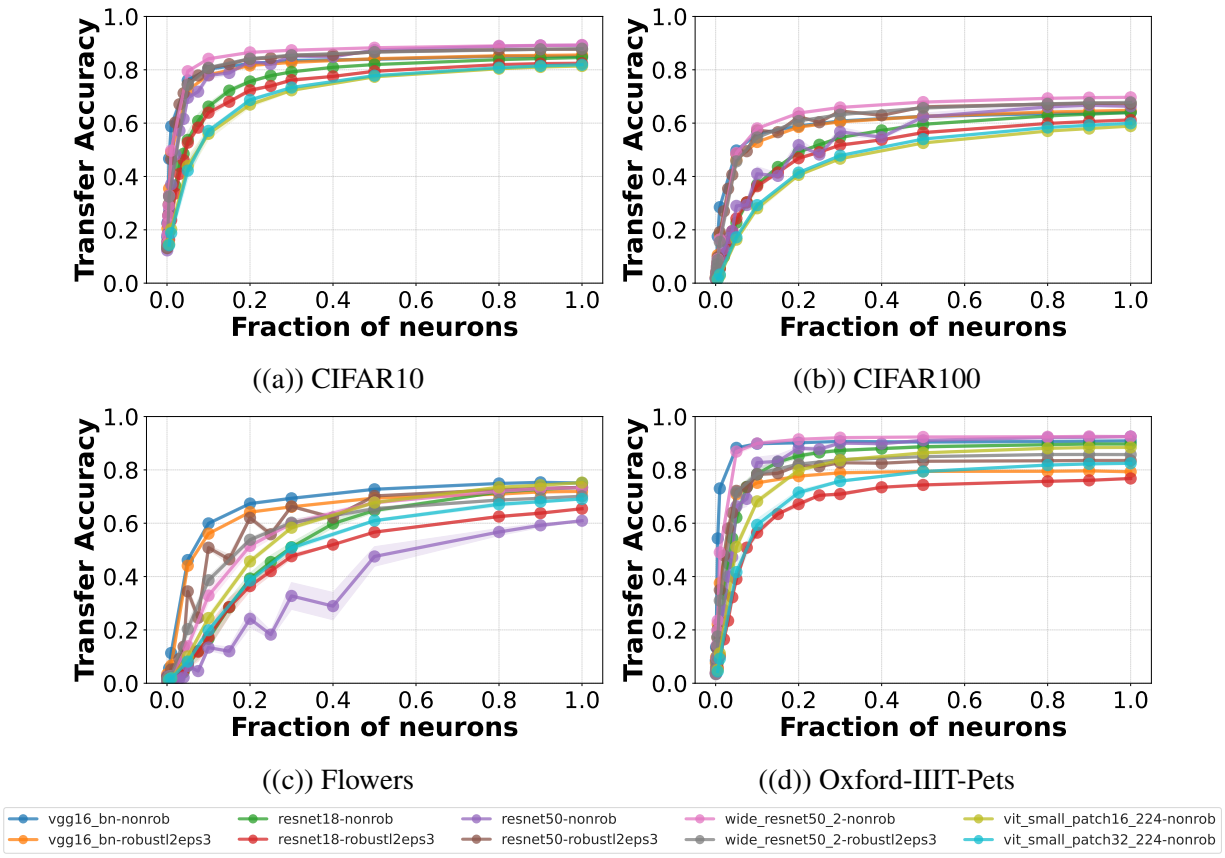


Figure 6.4: **[Comparisons Across Architectures For Downstream Task Accuracy]** All models shown here are pre-trained on ImageNet1k. We see that diffused redundancy exists across architectures, and the trend observed in Figure 6.1(c) & 6.1(a) regarding adversarially trained models also holds here as models' curves that are more "inside" are the ones trained with standard loss.

In order to better understand the phenomenon of diffused redundancy we analyze 30 different

pre-trained models, with different architectures, pre-training datasets and losses. We then evaluate each model for transfer accuracy on 4 datasets mentioned in Section 6.2. While all results in this section are on the penultimate layer, we also report results on intermediate layers in Appendix E.5 and find similar trends of diffused redundancy. All details for reproducibility can be found in Appendix E.2.

Architectures We consider VGG16 [118], ResNet18, ResNet50, WideResNet50-2 [119], ViT-S16 & ViT-S32 [193]. Additionally, we consider ResNet50 with varying widths of the final layer (denoted by `ResNet50_ffx` where x denotes the number of neurons in the final layer).

Upstream Datasets ImageNet-1k & ImageNet-21k [74].

Upstream Losses Standard cross-entropy, adversarial training (ℓ_2 threat model, $\epsilon = 3$ and ℓ_∞ threat model, $\epsilon = 4/255$ [3]) [1], MRL Loss [10], DeCov [227], and varying strengths of dropout [228].

Downstream Datasets CIFAR10/1000, Oxford-IIIT-Pets, and Flowers, same as Section 6.2. Additionally, we also report performance on harder datasets such as ImageNetv2 [127] and Places365 [229] in Appendix E.6. For all analyses in this section, we also report corresponding approximations for DR (Eq 6.1) in Appendix E.4.

6.3.1 Effects of Architecture, Upstream Loss, Upstream Datasets, and Downstream Datasets

Extending the analysis in Section 6.2, we evaluate the diffused redundancy hypothesis on other architectures. Fig 6.4 shows transfer performance for different architectures. All architectures shown in Fig 6.4 are trained on ImageNet1k. We find that our takeaways from Section 6.2 also

[3] Results for ℓ_∞ robust models can be found in Appendix E.4

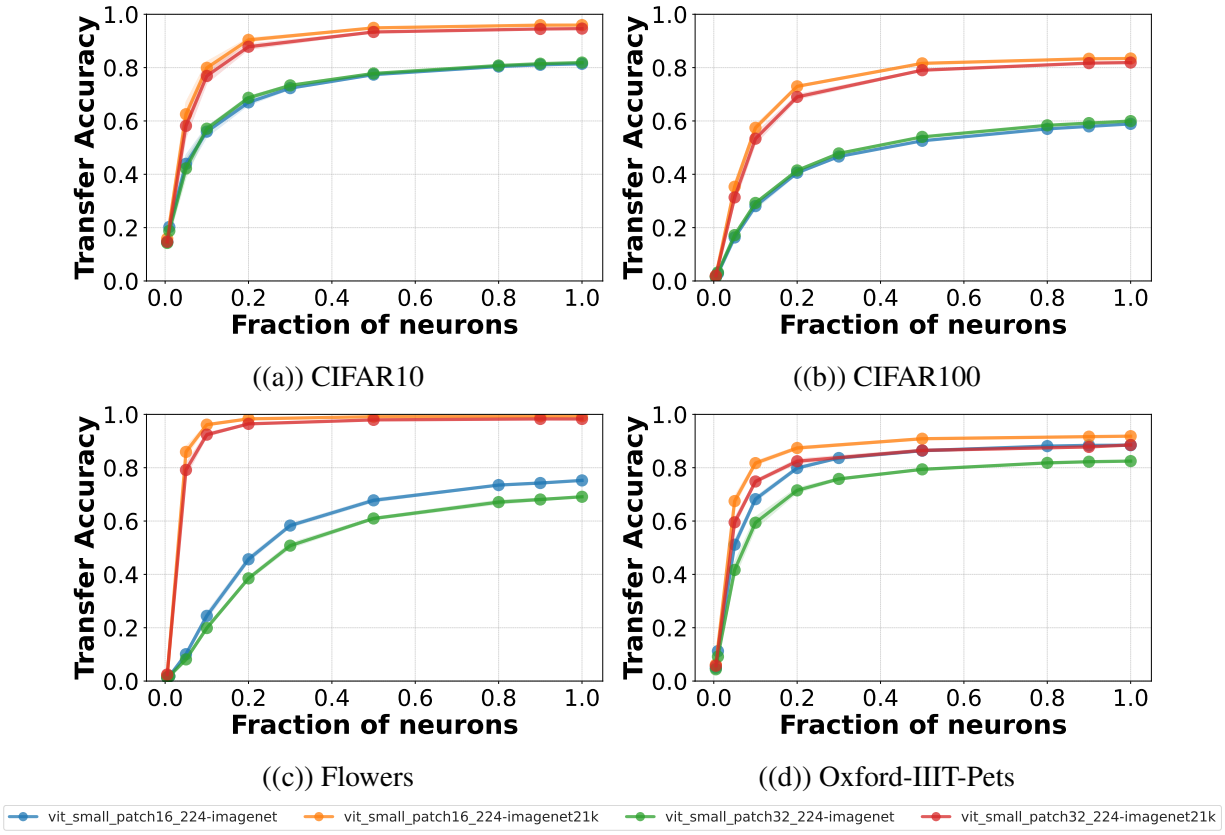


Figure 6.5: **[Comparison Across Upstream Datasets]** We see that degree of diffused redundancy depends a great deal on the upstream training dataset, in particular, models trained on ImageNet21k exhibit a higher degree of diffused redundancy, although the differences in the degree of diffused redundancy are downstream task dependent

extend to other architectures.

Fig 6.5 compares two instances each of ViT-S16 and ViT-S32, one trained on a bigger upstream dataset (ImageNet21k) and another on a smaller dataset (ImageNet1k)

Note that the nature of all curves in both Figs 6.4&6.5 highly depends on downstream datasets. This is also consistent with the initial observation of Section 6.2 about diffused redundancy being downstream dataset dependent. Additionally, we also report results on ImageNetV2 and Places365 in Appendix E.6 showing that diffused redundancy also holds on "harder" datasets.

6.3.2 Diffused Redundancy as a Function of Layer Width

We take the usual ResNet50 with a penultimate layer consisting of 2048 neurons and compare it with variants that are pre-trained with a much smaller penultimate layer, these are denoted by `ResNet50_ffx` where x (< 2048) is the number of neurons in the penultimate layer. Fig 6.6 shows how diffused redundancy slowly fades away as we squeeze the layer to be smaller. In fact, for `ResNet50_ff8`, we see that across all datasets we need $> 90\%$ of the full layer to achieve performance close to the full layer. This shows that diffused redundancy only appears in DNNs when the layer is sufficiently wide to encode redundancy.

6.3.3 Comparison With Methods That Optimize For Lesser Neurons

Matryoshka Representation Learning (MRL) is a recently proposed paradigm that learns nested representations such that the first $k, 2k, 4k, ..., N$ (where N = size of the full layer) dimensions of the extracted feature vector are all explicitly made to be good at minimizing upstream loss, with the intuition of learning coarse-to-fine representations. This ensures that one can flexibly use these smaller parts of the representation for downstream tasks. MRL, thus, ensures that redundancy shows up in learned representations in a *structured* way, *i.e.*, we know the first $k, 2k, ...$ neurons can be picked and used for downstream tasks and should perform reasonably.

Here we investigate two questions regarding Matryoshka representations: 1) do these representations also exhibit the phenomenon of diffused redundancy? *i.e.* if we were to ignore the structure imposed by MRL-type training and instead just pick random neurons from all over the layer, do we still get reasonable performance?, and 2) how do they compare to representations learned by other kinds of losses?

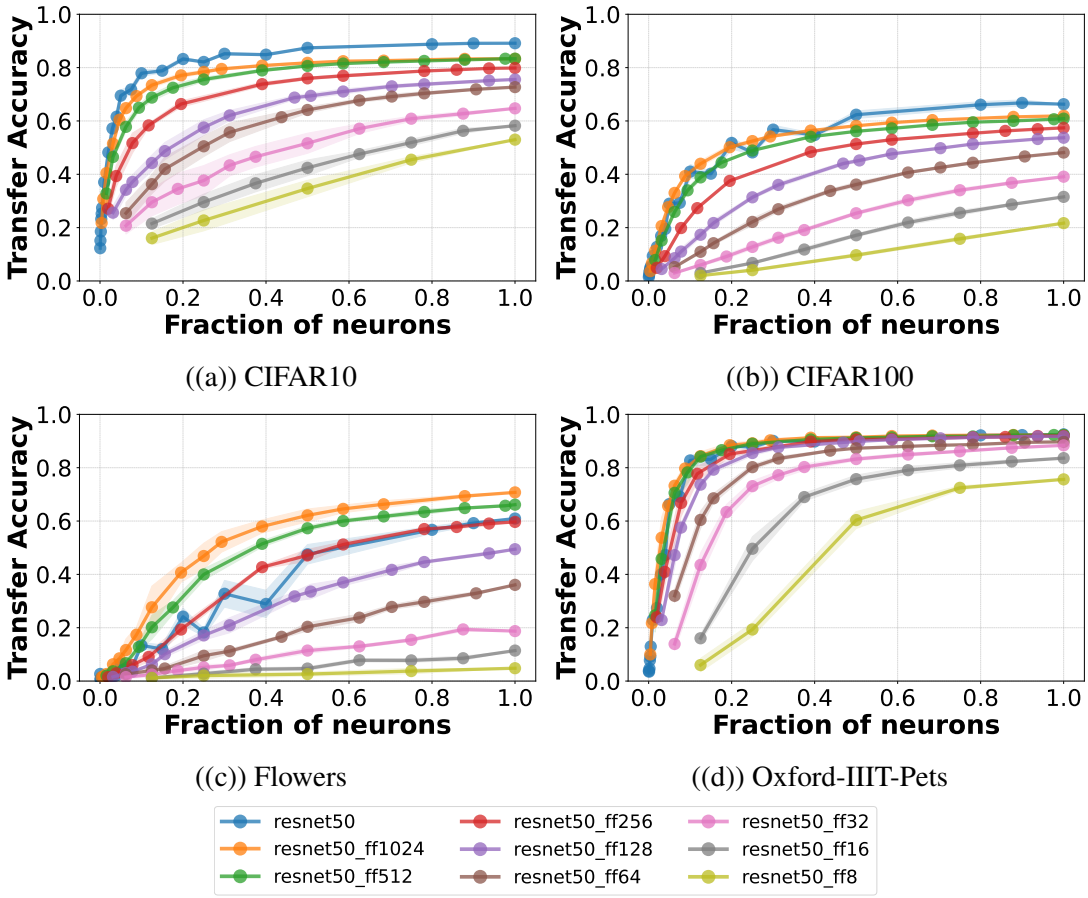


Figure 6.6: **[Diffused Redundancy as Function of Layer Width]** As we make the length of the layer smaller, the degree of redundancy becomes lesser. For `ResNet50_ff8`, *i.e.*ResNet50 with only 8 neurons in the final layer, we see that we need almost 90% of neurons to achieve similar accuracy as the full layer.

Figure 6.7 investigates these questions by comparing ResNet50 representations learned using MRL loss to other losses. `resnet50_mrl_nonrob_first` (red line) denotes a ResNet50 trained using MRL loss and evaluated on parts of the representation that were optimized to have low upstream loss (*i.e.* first $k, 2k, ...N$ neurons, here $k = 8$ and $N = 2048$) and `resnet50_mrl_nonrob_random` (green line) refers to the same model with the same number of neurons chosen for evaluation, except they're chosen at random from the entire layer.

First, we interestingly see that even the ResNet50 trained with MRL loss exhibits diffused redundancy (denoted by green line spiking very quickly for most datasets in Fig 6.7), despite

having been trained to only have structured redundancy. Based on this observation, we conjecture that diffused redundancy is a natural consequence of having a wide layer. Second, we see that ResNet50 trained on MRL indeed does better in the low neuron regime across datasets (red line on the extreme left part of the plots in Fig 6.7), but other models quickly catch up as we pick more neurons, thus indicating that major efficiency benefits of MRL-type models are best realized when using an extremely low number of neurons, else one can obtain similar downstream performances by simply picking random samples from existing pre-trained models.

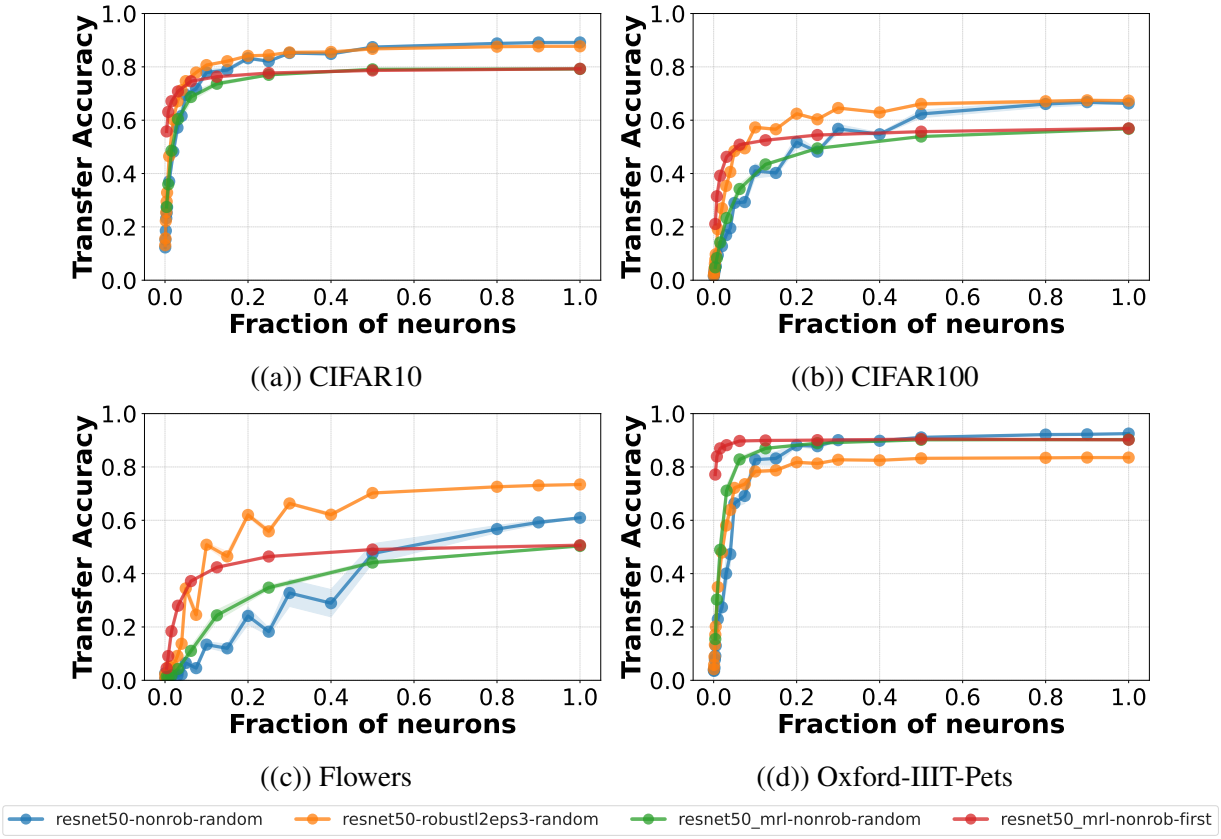


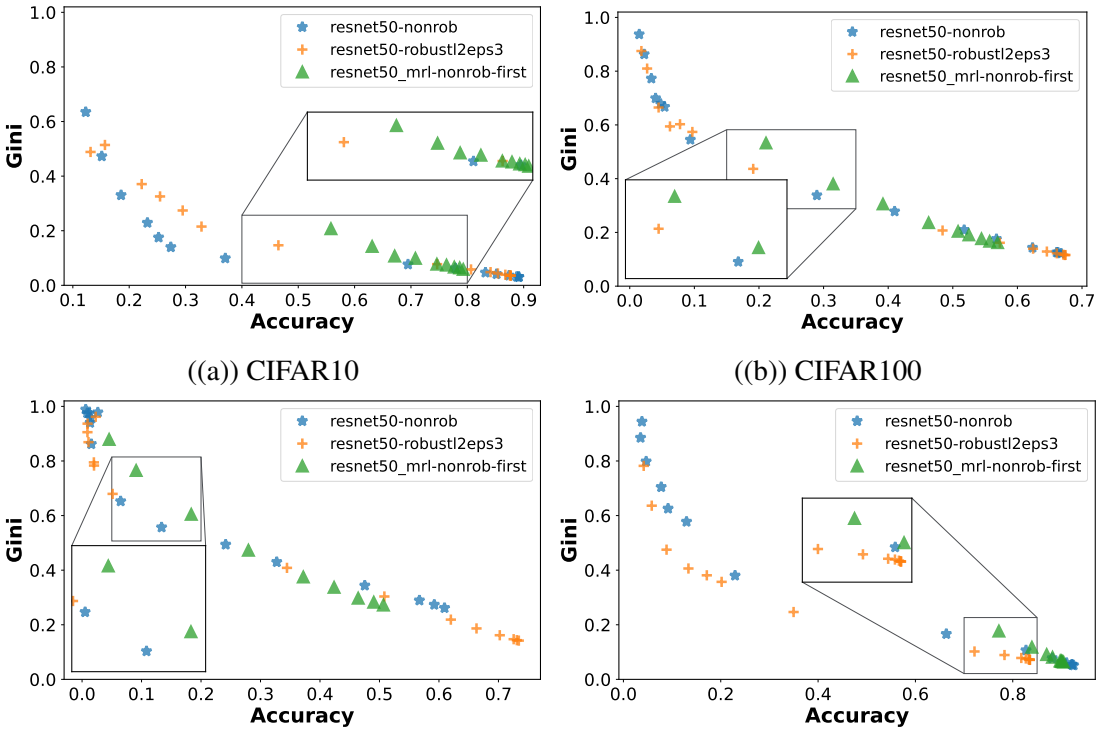
Figure 6.7: **[Comparison of Diffused Redundancy in MRL vs other losses]** Here we compare ResNet50 trained using multiple losses including MRL [10]. `nonrob` indicates that the model was trained with the standard crossentropy loss while `robustl2eps3` indicates adversarial training with ℓ_2 threat model and $\epsilon = 3$. Red line shows results for part of the representation explicitly optimized in MRL, whereas the green line shows results for parts that are picked randomly from the same representation. Even the MRL model shows a significant amount of diffused redundancy despite being explicitly trained to instead have structured redundancy.

6.3.4 Methods That Prevent Co-adaptation of Neurons Also Exhibit Diffused Redundancy

We consider two additional modifications to upstream pre-training: we add regularization to the usual cross-entropy loss that decorrelates neurons in the feature representation using DeCov [227] and Dropout [228]. Dropout does this implicitly by randomly dropping neurons during training and DeCov does this explicitly by putting a loss on the activations of a given layer. We pre-train different ResNet50 on ImageNet1k by varying strengths of dropout ranging from 0.1 all the way up to 0.8 and also by separately adding the DeCov regularizer (regularization strength = $1e-4$) to give 9 additional pre-trained models. Intuitively these methods should lead to *increased* diffused redundancy since by design these methods force the model to disperse similar information in different parts of a layer to perform the same task downstream task. Interestingly, we see that diffused redundancy in these models is almost completely independent of the regularization strength. This suggests that diffused redundancy might be an inevitable property of DNNs when the layer has sufficient width. Due to space constraints, we include these results in Appendix E.7.

6.4 Possible Fairness-Efficiency Tradeoffs in Efficient Downstream Transfer

One natural use case for diffused redundancy is efficient transfer to downstream datasets. As defined in Eq 6.1 and as also seen in Sections 6.2& 6.3, dropping neurons comes at a small cost (δ in Eq 6.1) in performance as compared to the full set of neurons. Here we take a deeper look into this drop in overall performance and investigate how it is distributed across classes. If the drop affects only a few classes, then dropping neurons – although efficient for downstream tasks – could have implications for fairness, which is not only of concern to ML researchers and



((a)) CIFAR10

((b)) CIFAR100

((c)) Flowers

((d)) Oxford-IIIT-Pets

Figure 6.8: **[Gini Coefficient of Class-Wise Accuracies as we Drop Neurons]** Higher value of Gini coefficient indicates higher inequality [11]. We see that for all models gini coefficients become higher as the accuracy reduces (as a result of dropping neurons). Additionally, in some regions (highlighted in the plots), the model explicitly optimized for efficient transfer (`resnet50_mrl`) can give rise to higher gini values, resulting in a more unequal spread of accuracy over classes.

practitioners [59, 65, 230], but also to lawyers [231] and policymakers [232].

We compare the spread of accuracies across classes using inequality indices, which are commonly used in economics to study income inequality [233, 234] and have also recently been adopted in the fair ML literature [235]. We use gini index [11] and coefficient of variation [236] to quantify the spread of performance across classes. For a perfect spread, both gini and coefficient of variation are 0, and higher values indicate higher inequality.

Figure 6.8 compares the gini index for various models at varying levels of accuracy (note that accuracy monotonically increases with more neurons, hence the right most point for a model represents the model with all neurons). We make two observations: across all datasets and all

models we find that a loss in accuracy (compared to the full layer) comes at the cost of a few classes, as opposed to being smeared throughout classes, as indicated by high gini values on the left of each plot. Additionally, we observe that the model trained using MRL loss tends to have slightly higher gini values in the regions where the drop in accuracy is slightly higher (highlighted on the plots). To ensure that this trend is not simply due to lower accuracy, we investigate the error distributions across classes (Appendix E.3) and find that predictions become more homogeneous as we drop more neurons. Similar trends are also observed with coeff. of variation as shown in Appendix E.3. These results draw caution to the potential unintended side-effects of exploiting diffused redundancy and suggest that there could be a possible fairness-efficiency tradeoff involved.

Connections to Robustness-Fairness Tradeoffs There are well-established tradeoffs between robustness and fairness in both standard [13] and adversarial training [237, 238]. One major difference between these works and our setup is that the task they train on is also the task they test on. We instead operate in the transfer learning setup where we train a linear probe on the representation of a pre-trained model. While it's not immediately clear whether the discrepancy between class accuracies observed on the pre-training task (*e.g.* as in [237]) also leads to an inter-class accuracy discrepancy on downstream tasks when dropping neurons, it would be very interesting future work to establish more formal connections between the robustness-fairness tradeoff and its effects on the fairness-efficiency tradeoff presented in this chapter.

6.5 Conclusion

We introduce the diffused redundancy hypothesis and analyze a wide range of models with different upstream training datasets, losses, and architectures. We carefully analyze the causes of such redundancy and find that upstream training (both loss and datasets) plays a crucial role and that this redundancy also depends on the downstream dataset. One direct practical consequence of our observation is increased efficiency for downstream training times which can have many positive impacts in terms of reduced energy costs [239] which is crucial in moving towards "green" AI [240]. We, however, also draw caution to the potential pitfalls of such efficiency gains , which might hurt the accuracy of certain classes more than others, thus having direct consequences for fairness.

Chapter 7: Discussion and Future Research Agenda

While my thesis took some steps to better understand and strengthen the foundations of trustworthy deep learning, there is still a lot of work to be done. With the rapid advancements in foundation models, *i.e.*models trained in extremely large datasets with large amounts of compute, there are ever-evolving challenges to ensure that we can keep these models safe and can trust them in deployment. I see three key promising directions for future work that I have highlighted below.

7.1 Safe Generative AI via Controlled Inference

Foundation models [241] are changing the landscape of machine learning, and more broadly the field of AI. I am particularly excited about the shift towards generative models, which have started to show remarkable capabilities with scale. Going forward a large part of our information diets will be controlled in part by content from generative models. Thus, it's extremely important to ensure that we can control how data is generated from these models. For example, image-to-text models have shown great image generation capabilities [242]. However, they continue to portray and sometimes even amplify harmful stereotypes [243]. Problematic stereotypes have also been reported for many modern LLMs [244]. Understanding and debugging the failure modes of such models is extremely challenging, primarily because their pre-training data scale is massive and often uncurated, and in many cases, private. Even if these models represent their

pre-training data distribution at inference, likely, this data distribution – which is mostly scraped from the internet – is not representative of our reality.

A promising direction, that can be broadly applied to a variety of safety problems in generative AI (such as avoiding hate speech, stereotypes, offensive content *etc.*) is *controlled generation*. The current interface that the user has into these models is the prompt. However, with controlled generation, my goal is to give the end user more control in guiding the model's outputs. This has two aspects to it: (1) how do we guide the model at inference time towards a user-specified distribution, and (2) how do we model a user's preferred output distribution? Thus, this is an interdisciplinary challenge involving both machine learning and human computer interaction expertise. For example, when generating pictures of a computer scientist from a text-to-image model, can we provide a lever for the end user to specify the skin color distribution of the outputs? Such a lever would not only be useful for end users but would also indicate that these models have a broad spectrum of plausible outputs and a given instance of a model only represents one plausible set of outputs.

7.2 Efficient Finetuning of Large Models

In the current age of foundation models, working with models containing more than a billion parameters is common. GPT-3, released in 2020, already included 175 billion parameters, and the latest versions of these enterprise models likely contain even more parameters. Before these billion parameter models became ubiquitous, common wisdom was that all model parameters could fit entirely on one GPU, and if one wanted to utilize multiple GPUs, one would replicate the entire model on different GPUs and train in data-parallel (*i.e.* each GPU sees its slice of

data and gradients are averaged before backpropagation) which speeds up training. However, in this new age of ever-larger models, one must also split the model parameters across different devices, since GPU memory is expensive and has not grown at the same rate as model size. This sharding, however, comes at a significant overhead of communication between different devices, since during training these devices must communicate activations, parameters, gradients, and optimizer states multiple times to just run one training iteration. This overhead can be overcome by using an expensive, high-bandwidth interface to connect different GPUs and nodes. This prohibits researchers with limited resources from being able to do full-feature finetuning of such large models.

I wish to build on my work on diffused redundancy [16] to reduce the communication overhead in the distributed finetuning of large models (both vision and language). I wish to investigate if this insight can be used to transfer the activations of only a few neurons when communicated during distributed training, thus greatly reducing overhead in communicating neuron activations. Recent works in this space have also proposed system-level optimizations to reduce the communication overhead of optimizer states [245]. Ultimately, my goal is to enable researchers with modest resources to also do full-feature finetuning of state-of-the-art models, since I believe increasing the ability of researchers to experiment with foundation models is a key aspect of building models responsibly.

7.3 Scaling Laws for Domain-Specific Models

One reason for excitement around large models is that these tend to work great for a variety of tasks and domains, even ones that these models were not explicitly trained for. Such models

are often trained on huge amounts of data, all mixed during pre-training. Prior works have found scaling laws for such settings, being able to reliably predict how one should scale their model size based on the amount of data. Many other domain experts (*e.g.* healthcare, law, archaeology *etc.*) are also interested in leveraging foundation models for their fields of study. The goal of domain-specific foundation models is to train a model (say LLM) to mimic or assist an expert in that domain. The model is still a foundation model, in that it can achieve a variety of "general" tasks in that domain. For example, a legal LLM should be able to perform a variety of legal tasks such as feature extraction from court cases, predict judgments from court transcripts, and parse and answer questions from legal research papers *etc.*.

Scaling laws for domain-specific models are less known. Questions such as: what data mixtures to use for pre-training such models and what amount of compute to spend per token of data are open questions with no clear answers. In collaboration with domain experts, I wish to better understand the requirements for building competitive and useful domain-specific foundation models.

Appendix A: Chapter 2, Appendix

A.1 Experimental Details

All code for this chapter was written in Python and PyTorch.

A.1.1 Medical Image Classification - CheXpert

Table A.1 shows the demographic composition of CheXpert for all the splits.

Data Pre-Processing

CheXpert contains chest x-ray images of different patients from different angles and thus does not have a fixed resolution. We resize each image to 256x256. We use the same pre-processing pipeline found in `https://github.com/jfhealthcare/Chexpert`. We add a Gaussian blur with $\sigma = 3$ and mean normalize each image with a mean of 128.0 and standard deviation of 64.0.

Data Augmentation

We augment data by duplicating each image with a random affine transformation with rotations between -15 and 15 degrees, a vertical and horizontal translation of 0.05, scaling between 0.95 and 1.05. We fill the areas outside the transformed region with gray color (128 RGB value). Pre-processing steps are applied to each augmented image.

Optimizer and Hyperparameters

We use a batch size of 56 for all our experiments. For some of the label flip experiments, we needed to increase the batch size to 200 so as to find a non-zero number of samples to flip. Each model was trained for 20 epochs. We used the Adam optimizer with a learning rate of 0.0001 and learning rate drops by a factor of 0.1 every epoch. This is the same as done by the authors of `https://github.com/jfhealthcare/Chexpert`. For the experiments where we added a fairness regularizer (Eq 2.2, 2.4, 2.8), we need to choose an additional hyperparameter α. We try a range of reasonable values and report results for the best α in Chapter 2. We also show results for other α values in Appendix A.2, however, we observe that higher values can interfere with the usual training objective.

Hardware

All experiments for CheXpert were run on a machine with 2 NVIDIA Tesla V100 GPUs. It took about 12-14 hours to fully train one CheXpert model. Face Recognition models were trained on Nvidia GeForce RTX 2080Ti GPUs, training one ResNet-18 model took approximately 10 hours, while training one ResNet-152 model took approximately 15 hours.

Table A.1: Distribution of males and females across CheXpert splits.

	TRAIN	VALIDATION	TEST
MALE	61%	53%	55%
FEMALE	39%	47%	45%

A.1.2 Face Recognition

Training routine

We train facial recognition systems with ResNet-18 and ResNet-152 backbones and CosFace head using Focal Loss for 120 epochs with a batch size of 512. We utilize the SGD optimizer with a momentum of 0.9, weight decay of 5e-4 and learning rate of 0.1, and we drop the learning rate by a factor of 10 at epochs 35, 65 and 95. All images used for training contain aligned faces re-scaled to 112×112. During training, we use random horizontal flip data augmentation.

For training routines, we modify the code from publicly available github repository face.evoLVe.PyTorch [1].

Training with fairness constraints

For facial recognition, we only apply an equal loss penalty, that is the absolute value of the difference between focal losses computed on male and female images in a batch. When regularization is imposed on training data, the same images are used to compute the classification loss and fairness penalty. When regularization is imposed on a holdout set, 10% of training images are kept for enforcing the fairness penalty and are not used in the recognition objective.

Adjusted Angular Margins

Below, we provide the loss for CosFace:

$$L_C = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{i=1,i\neq y}^{n} e^{s\cos(\theta_{j,i})}},$$

where x_i is the feature vector from the i-th sample from class $y^{(i)}$. W denotes the weight matrix of the last layer. Then, W_j is the j-th column of W and θ_{ij} denotes the angle between W_i and x_j. A fixed parameter $m \geq 0$ controls the magnitude of the cosine margin.

When trained regularly, $m = 0.35$ is used for both female and male images. When angular

[1] https://github.com/ZhaoJ9014/face.evoLVe.PyTorch

margin is adjusted for females, $m = 0.75$ is used for females and $m = 0.35$ for males.

Sensitive Information Removal Network

We denote the parameters of the sensitive information removal network (SIRN) as w and parameters of the sensitive classifier on top of it as w_s. SIRN takes as an input pre-trained embedding x_i of an image i from identity y_i and sensitive group s_i (gender) and outputs its projection $\varphi(x)$. The modified embedding is then fed into the CosFace head which outputs the logits for identities and into the sensitive head that outputs logits for genders. SIRN consists of 4 linear layers with ReLU-nonlinearities and the sensitive head consists of 3 linear layers with ReLU-nonlinearities. Let L_{FR} denote the focal loss for the facial recognition task and L_S denote the cross-entropy loss for the sensitive attribute classification task.

Then, the optimization objective for the problem is

$$\min_{w} \frac{1}{N} \sum_{i} L_{FR}(\varphi(x_i), y_i) + \alpha \log(1 + |0.9 - P_s(s|\varphi(x_i))|)$$

$$\min_{w_s} \frac{1}{N} \sum_{i} L_S(\varphi(x_i), s_i),$$

where s is a fixed sensitive group (fixed gender), and α is the magnitude of the adversarial regularization term. $P_s(s|\varphi(x_i)$ denotes the sensitive head logit corresponding to gender s. Therefore, the first objective minimizes the facial recognition loss while simultaneously maximizing the probability of predicting a fixed gender class for all images. At the same time, the second objective minimizes the classification loss of the sensitive attribute classifier.

A.2 Additional Results

Face Recognition Tables A.2 and A.3 show results for Face Recognition tasks. We also report results for additional hyperparameter values here.

CheXpert Tables A.4, A.5, A.6, and A.7 show results for all CheXpert tasks. We also report results for additional hyperparameter values here. Figure A.1 illustrates training curves for facial recognition models.

Table A.2: **[Face Recognition ResNet-18]** Performance of models trained with different training schemes designed for mitigating disparity in misclassification rates between males and females. All models have ResNet-18 backbone and CosFace head. The first column refers to the training scheme used, penalty indicates the size penalty coefficient. The train accuracy is computed in a classification manner, while validation and test accuracies are computed in the 1-nearest neighbors sense. The gap subcolumn refers to the difference between male and female accuracies. For the Fair Features training scheme, "gender" refers to the subgroup of images used in adversarial regularization during training.

Scheme	Penalty	Train			Validation			Test		
		Male	Female	Gap	Male	Female	Gap	Male	Female	Gap
Eq. Loss penalty on train set	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	$\alpha = 0.5$	72.7	75.1	-2.4	79.3	73.4	5.9	95	93.5	1.5
	$\alpha = 1$	28.2	29.5	-1.3	67.8	62	5.8	93	91.5	1.5
	$\alpha = 2$	0	0.1	-0.1	27	18.8	8.2	70.9	64.2	6.7
Eq. Loss penalty on holdout set	baseline	84	79.6	4.4	78.6	70.8	7.8	94.2	92.3	2.0
	$\alpha = 0.5$	59.3	46.9	12.4	75.2	66.5	8.7	94.5	92.5	2.0
	$\alpha = 1$	20.6	13	7.5	57.6	47.3	10.3	89.8	85.9	3.9
Random Labels Flipping	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	$p = 0.1$	92.4	91.8	0.6	81.3	75	6.3	95.4	93.2	2.1
	$p = 0.3$	68.2	86.6	-18.4	75.5	74.9	0.6	94.6	93.5	1.1
	$p = 0.5$	21.6	86	-64.4	67.3	73.3	-6	92.2	93	-0.7
Adjusted Margins	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	fair	94.2	90.2	3.9	81	77.6	3.3	94.4	91.9	2.6
Min-Max	baseline	97.6	94.4	3.2	82.4	74.6	7.9	94.8	92.9	1.9
	fair	74.3	75.9	-1.7	79.4	73.4	6	95.5	93.4	2.1
Fair Features (Females)	baseline	98.8	97	1.8	81.5	74.5	7	93.7	91.2	2.6
	$\alpha = 1$	98.6	96.8	1.9	81.4	73.9	7.5	93.7	90.9	2.8
	$\alpha = 2$	98.4	96.3	2.1	81.2	72.9	8.3	93.7	90.6	3.1
	$\alpha = 3$	96.2	93.2	3	79.9	70.5	9.4	93.3	88.9	4.4
Fair Features (Males)	baseline	98.8	97	1.8	81.5	74.5	7	93.7	91.2	2.6
	$\alpha = 1$	98.7	96.8	1.9	81.1	74.2	6.9	93.6	91.2	2.4
	$\alpha = 2$	98.7	96.8	1.9	81.1	74.2	6.9	93.6	91.2	2.4
	$\alpha = 3$	90.3	86.6	3.7	75.3	71.9	3.4	89.5	89.9	-0.3

Table A.3: **[Face Recognition ResNet-152]** Performance of models trained with different training schemes designed for mitigating disparity in misclassification rates between males and females. All models have ResNet-152 backbone and CosFace head. The first column refers to the training scheme used, penalty indicates the size penalty coefficient. The train accuracy is computed in a classification manner, while validation and test accuracies are computed in the 1-nearest neighbors sense. The gap subcolumn refers to the difference between male and female accuracies. For the Fair Features training scheme, gender refers to the subgroup of images used in adversarial regularization during training.

Scheme	Penalty	Train			Validation			Test		
		Male	Female	Gap	Male	Female	Gap	Male	Female	Gap
Eq. Loss penalty on train set	baseline	99.6	99	0.6	88.1	81.4	6.8	96.6	94.5	2.1
	$\alpha = 0.5$	84.7	87.4	-2.7	86.4	80.9	5.5	97.2	95.3	1.9
	$\alpha = 1$	38.8	40.6	-1.8	76.4	70.4	6	95.4	94.2	1.2
	$\alpha = 2$	0	0	0	24.8	17.4	7.4	69.1	61.3	7.8
Random Labels Flip	baseline	99.6	99	0.6	88.1	81.4	6.8	96.6	94.5	2.1
	$p = 0.3$	80.9	94.9	-13.9	83.8	81.3	2.4	96.8	95.1	1.6
	$p = 0.5$	31.5	93.2	-61.7	77.8	80.8	-3	95.2	94.9	0.3
Adjusted Margins	baseline	99.6	99	0.6	88.1	81.4	6.8	96.6	94.5	2.1
	fair	99.1	98.4	0.7	85.7	85.2	0.5	95.9	91	4.9
Fair Features (Females)	baseline	99.8	99.6	0.2	87.3	80.6	6.7	95.9	93.3	2.6
	$\alpha = 1$	99.7	99.5	0.2	86.9	79.5	7.4	95.7	92.5	3.1
	$\alpha = 2$	99.4	98.9	0.5	86	77.1	8.9	95.1	91	4.1
	$\alpha = 3$	96.8	96	0.8	85.2	74.1	11.1	94.2	88.7	5.5
Fair Features (Males)	baseline	99.8	99.6	0.2	87.3	80.6	6.7	95.9	93.3	2.6
	$\alpha = 1$	99.7	99.5	0.2	86.3	80.2	6.1	95.4	93.2	2.2
	$\alpha = 2$	97.8	97.3	0.6	82.5	78.4	4	92.8	91.9	0.9
	$\alpha = 3$	91.9	90.1	1.8	79.4	75.7	3.7	90.2	90.1	0.1

Table A.4: **[CheXpert - training with fairness penalties]** Results for all 5 CheXpert tasks: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE). The regularizer is optimized on the training data, and α here denotes the coefficient of the regularizer. Results in Table 2.1 (main paper) are for $\alpha = 100$ on the equal loss penalty and $\alpha = 1000$ on the equal odds and disparate impact penalties. Additionally here we also report the penalty that was explicitly optimized during training (lower is better) and show that even this penalty does not generalize.

Scheme	Task	Train				Test			
		M	F	Gap	Penalty	M	F	Gap	Penalty
Baseline	CD	$0.905_{\pm 0.033}$	$0.902_{\pm 0.036}$	$0.004_{\pm 0.002}$	$0.006_{\pm 0.003}$	$0.712_{\pm 0.032}$	$0.691_{\pm 0.017}$	$0.046_{\pm 0.023}$	$0.016_{\pm 0.017}$
	ED	$0.857_{\pm 0.028}$	$0.844_{\pm 0.035}$	$0.013_{\pm 0.007}$	$0.015_{\pm 0.009}$	$0.905_{\pm 0.012}$	$0.849_{\pm 0.048}$	$0.057_{\pm 0.053}$	$0.022_{\pm 0.008}$
	CO	$0.832_{\pm 0.062}$	$0.834_{\pm 0.065}$	$0.004_{\pm 0.003}$	$0.004_{\pm 0.002}$	$0.824_{\pm 0.051}$	$0.760_{\pm 0.068}$	$0.069_{\pm 0.063}$	$0.163_{\pm 0.108}$
	AT	$0.716_{\pm 0.058}$	$0.709_{\pm 0.064}$	$0.008_{\pm 0.003}$	$0.002_{\pm 0.001}$	$0.804_{\pm 0.056}$	$0.756_{\pm 0.110}$	$0.065_{\pm 0.070}$	$0.004_{\pm 0.003}$
	PE	$0.873_{\pm 0.021}$	$0.880_{\pm 0.021}$	$0.007_{\pm 0.002}$	$0.013_{\pm 0.006}$	$0.851_{\pm 0.029}$	$0.923_{\pm 0.014}$	$0.072_{\pm 0.023}$	$0.433_{\pm 0.167}$
	Avg	**0.837**	**0.834**	**0.007**	**0.008**	**0.819**	**0.796**	**0.062**	**0.127**
Eq. Loss Train $\alpha = 100$	CD	$0.934_{\pm 0.028}$	$0.932_{\pm 0.029}$	$0.002_{\pm 0.002}$	$0.005_{\pm 0.002}$	$0.764_{\pm 0.040}$	$0.758_{\pm 0.044}$	$0.005_{\pm 0.004}$	$0.001_{\pm 0.001}$
	ED	$0.879_{\pm 0.020}$	$0.872_{\pm 0.020}$	$0.007_{\pm 0.000}$	$0.016_{\pm 0.001}$	$0.914_{\pm 0.012}$	$0.875_{\pm 0.004}$	$0.039_{\pm 0.007}$	$0.019_{\pm 0.001}$
	CO	$0.888_{\pm 0.074}$	$0.888_{\pm 0.072}$	$0.003_{\pm 0.000}$	$0.003_{\pm 0.002}$	$0.840_{\pm 0.005}$	$0.849_{\pm 0.031}$	$0.026_{\pm 0.009}$	$0.136_{\pm 0.069}$
	AT	$0.732_{\pm 0.021}$	$0.727_{\pm 0.026}$	$0.005_{\pm 0.005}$	$0.002_{\pm 0.001}$	$0.825_{\pm 0.038}$	$0.772_{\pm 0.083}$	$0.053_{\pm 0.045}$	$0.003_{\pm 0.002}$
	PE	$0.882_{\pm 0.012}$	$0.886_{\pm 0.013}$	$0.004_{\pm 0.002}$	$0.005_{\pm 0.001}$	$0.861_{\pm 0.025}$	$0.897_{\pm 0.048}$	$0.036_{\pm 0.023}$	$0.387_{\pm 0.002}$
	Avg	**0.863**	**0.861**	**0.004**	**0.006**	**0.841**	**0.830**	**0.032**	**0.109**
Eq. Loss Train $\alpha = 1000$	CD	$0.820_{\pm 0.018}$	$0.814_{\pm 0.025}$	$0.007_{\pm 0.005}$	$0.001_{\pm 0.001}$	$0.805_{\pm 0.027}$	$0.770_{\pm 0.017}$	$0.035_{\pm 0.011}$	$0.002_{\pm 0.001}$
	ED	$0.816_{\pm 0.005}$	$0.799_{\pm 0.000}$	$0.018_{\pm 0.006}$	$0.004_{\pm 0.003}$	$0.888_{\pm 0.027}$	$0.811_{\pm 0.007}$	$0.077_{\pm 0.033}$	$0.010_{\pm 0.004}$
	CO	$0.680_{\pm 0.014}$	$0.686_{\pm 0.013}$	$0.006_{\pm 0.001}$	$0.000_{\pm 0.000}$	$0.909_{\pm 0.002}$	$0.800_{\pm 0.019}$	$0.109_{\pm 0.022}$	$0.010_{\pm 0.001}$
	AT	$0.625_{\pm 0.057}$	$0.613_{\pm 0.062}$	$0.012_{\pm 0.005}$	$0.002_{\pm 0.002}$	$0.740_{\pm 0.167}$	$0.719_{\pm 0.168}$	$0.021_{\pm 0.001}$	$0.001_{\pm 0.000}$
	PE	$0.843_{\pm 0.022}$	$0.851_{\pm 0.020}$	$0.009_{\pm 0.002}$	$0.002_{\pm 0.001}$	$0.834_{\pm 0.039}$	$0.924_{\pm 0.026}$	$0.090_{\pm 0.013}$	$0.248_{\pm 0.109}$
	Avg	**0.757**	**0.753**	**0.010**	**0.002**	**0.835**	**0.805**	**0.066**	**0.054**
Disp. Impact Penalty $\alpha = 100$	CD	$0.925_{\pm .045}$	$0.921_{\pm .048}$	$0.005_{\pm .002}$	$-0.211_{\pm .146}$	$0.759_{\pm .020}$	$0.743_{\pm .010}$	$0.028_{\pm .020}$	$-0.206_{\pm .147}$
	ED	$0.882_{\pm .047}$	$0.872_{\pm .052}$	$0.010_{\pm .005}$	$-0.316_{\pm .194}$	$0.921_{\pm .031}$	$0.843_{\pm .036}$	$0.077_{\pm .063}$	$-0.305_{\pm .200}$
	CO	$0.865_{\pm .084}$	$0.867_{\pm .083}$	$0.003_{\pm .002}$	$-0.181_{\pm .084}$	$0.799_{\pm .081}$	$0.730_{\pm .156}$	$0.074_{\pm .085}$	$-0.161_{\pm .095}$
	AT	$0.753_{\pm .067}$	$0.744_{\pm .076}$	$0.011_{\pm .008}$	$-0.526_{\pm .154}$	$0.766_{\pm .037}$	$0.690_{\pm .070}$	$0.076_{\pm .035}$	$-0.510_{\pm .171}$
	PE	$0.891_{\pm .026}$	$0.898_{\pm .027}$	$0.006_{\pm .002}$	$-0.839_{\pm .180}$	$0.878_{\pm .012}$	$0.907_{\pm .014}$	$0.030_{\pm .016}$	$-0.781_{\pm .248}$
	Avg	**0.863**	**0.860**	**0.007**	**-0.415**	**0.825**	**0.783**	**0.057**	**-0.393**
Disp. Impact Penalty $\alpha = 1000$	CD	$0.922_{\pm .047}$	$0.919_{\pm .050}$	$0.004_{\pm .003}$	$-0.196_{\pm .125}$	$0.795_{\pm .031}$	$0.725_{\pm .041}$	$0.070_{\pm .010}$	$-0.184_{\pm .126}$
	ED	$0.877_{\pm .051}$	$0.864_{\pm .056}$	$0.013_{\pm .006}$	$-0.381_{\pm .062}$	$0.901_{\pm .009}$	$0.848_{\pm .013}$	$0.053_{\pm .016}$	$-0.367_{\pm .079}$
	CO	$0.834_{\pm .107}$	$0.837_{\pm .105}$	$0.003_{\pm .002}$	$-0.213_{\pm .140}$	$0.705_{\pm .116}$	$0.673_{\pm .085}$	$0.040_{\pm .047}$	$-0.206_{\pm .140}$
	AT	$0.744_{\pm .059}$	$0.733_{\pm .057}$	$0.012_{\pm .001}$	$-0.532_{\pm .111}$	$0.818_{\pm .049}$	$0.734_{\pm .052}$	$0.085_{\pm .015}$	$-0.532_{\pm .110}$
	PE	$0.885_{\pm .013}$	$0.889_{\pm .011}$	$0.005_{\pm .002}$	$-0.834_{\pm .086}$	$0.846_{\pm .019}$	$0.903_{\pm .008}$	$0.057_{\pm .011}$	$-0.806_{\pm .114}$
	Avg	**0.852**	**0.848**	**0.007**	**-0.431**	**0.813**	**0.777**	**0.061**	**-0.419**
Eq. Odds Penalty $\alpha = 100$	CD	$0.934_{\pm 0.050}$	$0.932_{\pm 0.054}$	$0.004_{\pm 0.002}$	$0.009_{\pm 0.002}$	$0.743_{\pm 0.043}$	$0.745_{\pm 0.042}$	$0.002_{\pm 0.001}$	$0.005_{\pm 0.000}$
	ED	$0.889_{\pm 0.029}$	$0.880_{\pm 0.029}$	$0.008_{\pm 0.000}$	$0.024_{\pm 0.001}$	$0.883_{\pm 0.041}$	$0.836_{\pm 0.058}$	$0.047_{\pm 0.017}$	$0.019_{\pm 0.006}$
	CO	$0.879_{\pm 0.103}$	$0.884_{\pm 0.101}$	$0.004_{\pm 0.002}$	$0.002_{\pm 0.000}$	$0.776_{\pm 0.032}$	$0.790_{\pm 0.013}$	$0.045_{\pm 0.013}$	$0.083_{\pm 0.004}$
	AT	$0.771_{\pm 0.050}$	$0.768_{\pm 0.059}$	$0.008_{\pm 0.002}$	$0.005_{\pm 0.002}$	$0.769_{\pm 0.072}$	$0.699_{\pm 0.097}$	$0.070_{\pm 0.025}$	$0.010_{\pm 0.001}$
	PE	$0.895_{\pm 0.013}$	$0.900_{\pm 0.015}$	$0.005_{\pm 0.001}$	$0.005_{\pm 0.001}$	$0.840_{\pm 0.066}$	$0.893_{\pm 0.029}$	$0.053_{\pm 0.037}$	$0.185_{\pm 0.003}$
	Avg	**0.874**	**0.873**	**0.006**	**0.009**	**0.802**	**0.792**	**0.043**	**0.060**
Eq. Odds Penalty $\alpha = 1000$	CD	$0.929_{\pm 0.053}$	$0.924_{\pm 0.057}$	$0.005_{\pm 0.004}$	$0.006_{\pm 0.001}$	$0.774_{\pm 0.015}$	$0.768_{\pm 0.036}$	$0.021_{\pm 0.005}$	$0.004_{\pm 0.001}$
	ED	$0.896_{\pm 0.037}$	$0.894_{\pm 0.042}$	$0.005_{\pm 0.002}$	$0.022_{\pm 0.007}$	$0.893_{\pm 0.010}$	$0.872_{\pm 0.007}$	$0.022_{\pm 0.003}$	$0.023_{\pm 0.016}$
	CO	$0.866_{\pm 0.112}$	$0.867_{\pm 0.109}$	$0.003_{\pm 0.002}$	$0.003_{\pm 0.000}$	$0.777_{\pm 0.109}$	$0.742_{\pm 0.090}$	$0.035_{\pm 0.020}$	$0.088_{\pm 0.000}$
	AT	$0.760_{\pm 0.059}$	$0.757_{\pm 0.063}$	$0.003_{\pm 0.003}$	$0.003_{\pm 0.002}$	$0.754_{\pm 0.120}$	$0.761_{\pm 0.083}$	$0.038_{\pm 0.007}$	$0.013_{\pm 0.002}$
	PE	$0.895_{\pm 0.020}$	$0.901_{\pm 0.020}$	$0.006_{\pm 0.000}$	$0.006_{\pm 0.001}$	$0.870_{\pm 0.024}$	$0.940_{\pm 0.007}$	$0.070_{\pm 0.031}$	$0.175_{\pm 0.007}$
	Avg	**0.869**	**0.869**	**0.005**	**0.008**	**0.814**	**0.817**	**0.037**	**0.061**

Table A.5: **[CheXpert - fairness penalties on the validation set]** Results for all 5 CheXpert tasks: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE).

Scheme	Task	Train			Validation			Test		
		M	F	Gap	M	F	Gap	M	F	Gap
Baseline	CD	.988	.990	.002	.826	.818	.007	.766	.739	.027
	ED	.953	.952	.000	.780	.777	.002	.885	.862	.024
	CO	.996	.996	.000	.681	.687	.006	.896	.808	.088
	AT	.879	.883	.004	.637	.631	.007	.637	.570	.068
	PE	.936	.942	.006	.841	.854	.013	.895	.925	.030
	Avg	**.950**	**.953**	**.002**	**.753**	**.753**	**.007**	**.816**	**.781**	**.047**
Eq. Loss Val	CD	.985	.987	.002	.827	.824	.002	.745	.669	.076
	ED	.931	.930	.001	.746	.735	.011	.885	.877	.008
	CO	.979	.984	.005	.663	.682	.019	.865	.680	.185
	AT	.863	.868	.005	.626	.629	.002	.759	.763	.004
	PE	.926	.932	.006	.839	.856	.017	.897	.933	.036
	Avg	**.937**	**.940**	**.004**	**.740**	**.745**	**.010**	**.830**	**.784**	**.062**
Eq. Odds Val	CD	.994	.996	.002	.776	.774	.002	.637	.715	.078
	ED	.936	.934	.003	.729	.721	.009	.858	.699	.159
	CO	.995	.995	.000	.670	.678	.008	.797	.705	.093
	AT	.867	.867	.001	.608	.595	.013	.671	.565	.106
	PE	.945	.953	.008	.830	.845	.015	.896	.917	.021
	Avg	**.947**	**.949**	**.003**	**.723**	**.722**	**.009**	**.772**	**.720**	**.091**
Disp. Impact Val	CD	.997	.997	.000	.795	.788	.007	.691	.708	.018
	ED	.972	.973	.001	.768	.760	.008	.853	.891	.038
	CO	.995	.996	.000	.641	.635	.006	.712	.640	.072
	AT	.874	.879	.005	.633	.626	.007	.786	.668	.118
	PE	.945	.951	.006	.830	.843	.013	.897	.921	.024
	Avg	**.957**	**.959**	**.003**	**.733**	**.730**	**.008**	**.788**	**.765**	**.054**

Table A.6: **[CheXpert - minmax training]** Results for all 5 CheXpert tasks: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE). We see that minmax training does not lead to increased fairness on the test set.

Scheme	Task	Train			Validation			Test		
		M	F	Gap	M	F	Gap	M	F	Gap
Baseline	CD	.988	.990	.002	.826	.818	.007	.766	.739	.027
	ED	.953	.952	.000	.780	.777	.002	.885	.862	.024
	CO	.996	.996	.000	.681	.687	.006	.896	.808	.088
	AT	.879	.883	.004	.637	.631	.007	.637	.570	.068
	PE	.936	.942	.006	.841	.854	.013	.895	.925	.030
	Avg	**.950**	**.953**	**.002**	**.753**	**.753**	**.007**	**.816**	**.781**	**.047**
Min-Max Loss	CD	.992	.981	.011	.824	.820	.004	.771	.789	.018
	ED	.963	.904	.058	.774	.774	.000	.895	.838	.057
	CO	.988	.976	.012	.694	.713	.019	.886	.821	.065
	AT	.901	.771	.130	.657	.652	.004	.758	.826	.068
	PE	.946	.896	.050	.842	.852	.010	.891	.927	.036
	Avg	**.958**	**.906**	**.052**	**.758**	**.762**	**.008**	**.840**	**.840**	**.049**

Table A.7: **[CheXpert - random label flipping]** Results for all 5 CheXpert tasks: Cardiomegaly (CA), Edema (ED), Consolidation (CO), Atelectasis (AT) and Pleural Effusion (PE). We see that randomly flipping the true labels of some fraction of a certain group has no clear trend and thus, while it may seem like a simple method to ensure fairness, it requires a lot of trial and error (*e.g.* setting the right fraction to flip, which subgroup to flip *etc.*). There is also a decline in performance (AUC scores drop as compared to the baseline) for all of the label flip experiments.

Scheme	Task	Train			Validation			Test		
		M	F	Gap	M	F	Gap	M	F	Gap
Baseline	CD	.988	.990	.002	.826	.818	.007	.766	.739	.027
	ED	.953	.952	.000	.780	.777	.002	.885	.862	.024
	CO	.996	.996	.000	.681	.687	.006	.896	.808	.088
	AT	.879	.883	.004	.637	.631	.007	.637	.570	.068
	PE	.936	.942	.006	.841	.854	.013	.895	.925	.030
	Avg	**.950**	**.953**	**.002**	**.753**	**.753**	**.007**	**.816**	**.781**	**.047**
Random Flip $p = 0.1$ Male Flipped	CD	.980	.943	.037	.814	.794	.020	.743	.790	.047
	ED	.930	.882	.048	.756	.750	.006	.892	.774	.118
	CO	.967	.897	.070	.656	.636	.020	.600	.667	.067
	AT	.853	.797	.056	.622	.594	.028	.668	.612	.056
	PE	.925	.914	.011	.841	.850	.009	.832	.916	.084
	Avg	**.931**	**.887**	**.044**	**.738**	**.725**	**.017**	**.747**	**.752**	**.074**
Random Flip $p = 0.1$ Female Flipped	CD	.902	.965	.063	.738	.778	.039	.711	.646	.065
	ED	.890	.919	.029	.735	.744	.009	.908	.827	.081
	CO	.836	.908	.072	.580	.643	.064	.685	.668	.017
	AT	.777	.825	.048	.610	.606	.004	.687	.758	.071
	PE	.877	.909	.032	.807	.833	.026	.776	.927	.150
	Avg	**.856**	**.905**	**.049**	**.694**	**.721**	**.028**	**.754**	**.765**	**.077**
Random Flip $p = 0.05$ Male Flipped	CD	.989	.961	.028	.790	.762	.028	.716	.634	.082
	ED	.950	.921	.030	.757	.744	.013	.889	.742	.147
	CO	.971	.941	.030	.684	.681	.003	.876	.758	.118
	AT	.854	.827	.027	.652	.632	.020	.689	.675	.014
	PE	.917	.910	.007	.829	.841	.013	.842	.906	.063
	Avg	**.936**	**.912**	**.024**	**.742**	**.732**	**.015**	**.802**	**.743**	**.085**
Random Flip $p = 0.05$ Female Flipped	CD	.956	.979	.024	.802	.811	.009	.729	.726	.003
	ED	.906	.924	.017	.730	.737	.007	.869	.733	.137
	CO	.897	.952	.055	.622	.635	.013	.659	.624	.035
	AT	.760	.784	.024	.592	.592	.001	.676	.639	.037
	PE	.872	.883	.011	.787	.799	.012	.795	.765	.030
	Avg	**.878**	**.904**	**.026**	**.707**	**.715**	**.008**	**.745**	**.697**	**.048**
Random Flip $p = 0.01$ Male Flipped	CD	.984	.981	.003	.786	.776	.010	.782	.683	.099
	ED	.949	.941	.008	.765	.756	.009	.856	.779	.076
	CO	.974	.972	.002	.647	.643	.003	.792	.794	.002
	AT	.821	.814	.007	.621	.608	.013	.770	.770	.000
	PE	.915	.917	.002	.817	.820	.003	.834	.916	.082
	Avg	**.929**	**.925**	**.004**	**.727**	**.720**	**.008**	**.807**	**.788**	**.052**

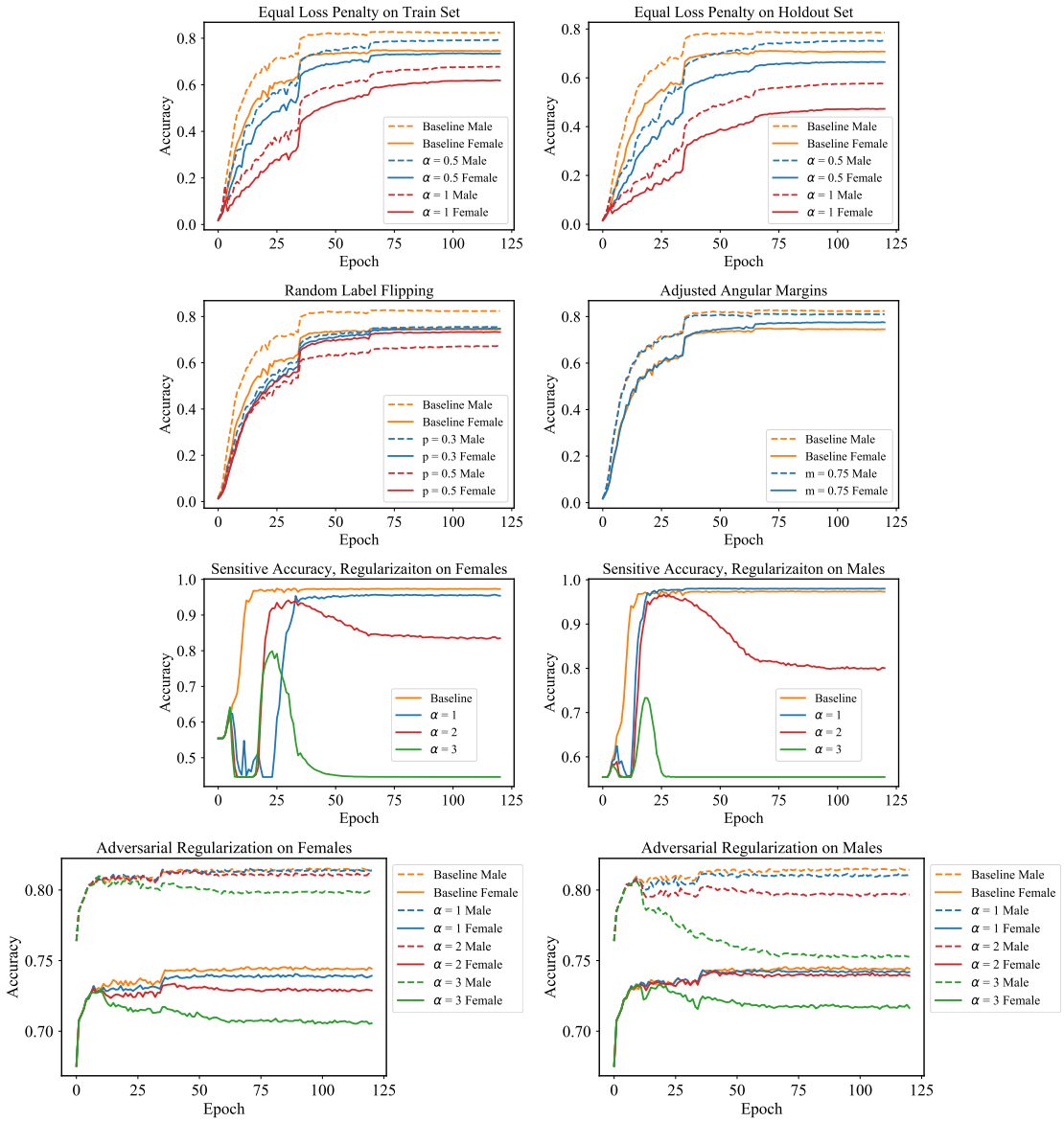


Figure A.1: **[Training Curves for Facial Recognition]** First row: Validation accuracy of facial recognition models trained with the equal loss penalty imposed on train (1) and holdout data (2). Second row: Validation accuracy of models trained with random label flipping (3) and adjusted angular margins (4). Third row: Accuracy of the sensitive classifier predicting gender for an adversarial penalty imposed on females (5) and males (6). Fourth row: Validation accuracy of models trained with adversarial regularization imposed on females (7) and males (8). Solid lines denote accuracy of the model for females and dashed lines denote accuracy for males. All curves are for ResNet-18 based models.

Appendix B: Chapter 3 Appendix

B.1 Additional Model and Dataset Details

CIFAR-10, CIFAR-100, CIFAR100Super These are standard deep learning benchmark datasets. Both CIFAR-10 and CIFAR-100 contain $60,000$ images in total which are split into $50,000$ train and $10,000$ test images. The task is to classify a given image. Images are mean normalized with mean and std of $(0.5, 0.5, 0.5)$.

UTKFace Contains images of a people labeled with race, gender and age. We split the dataset into a random $80:20$ train:test split to get $4,742$ test and $18,966$ train samples. We bin the age into 5 age groups and convert this into a 5-class classification problem. Images are normalized with mean and std of 0 and 1 respectively.

Adience Contains images of people labeled with gender and age group. The task is to classify a given image into one of 8 age groups. We split the dataset into a random 80:20 train:test split to get $14,007$ train and $3,445$ test samples. Images are normalized with mean and std of $(0.485, 0.456, 0.406)$ and $(0.229, 0.224, 0.225)$ respectively.

Accuracy of models trained on these datasets can be found in Table B.2. Figures B.1, B.3, B.5, and B.7 show the measures of robustness bias as approximated by CarliniWagner (σ_P^{CW}) and DeepFool (σ_P^{DF}) across different partitions of all the datasets.

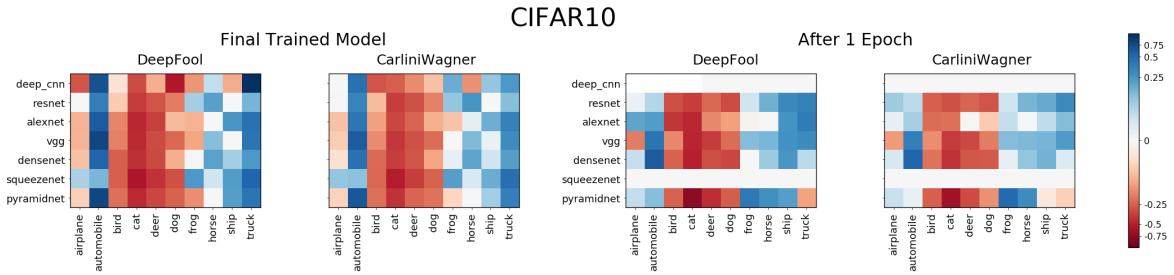


Figure B.1: Depiction of σ_P^{DF} and σ_P^{CW} for the CIFAR10 dataset with partitions corresponding to the class labels $\mathcal{C}$. These values are reported for all five convolutional models both at the beginning of their training (after one epoch) and at the end. We observe that, largely, the signedness of the functions are consistent between the five models and also across the training cycle.

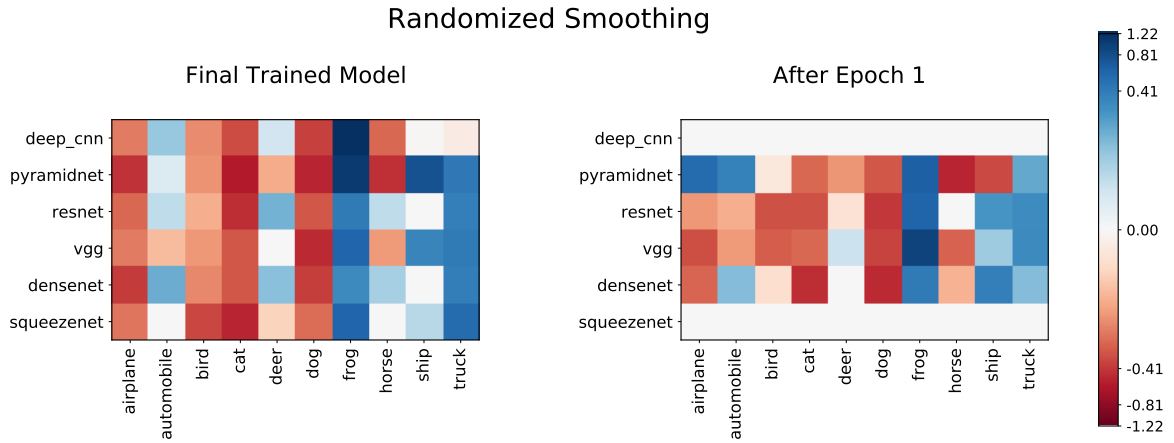


Figure B.2: Depiction of σ_P^{RS} for the CIFAR10 dataset with partitions corresponding to the class labels $\mathcal{C}$.

B.2 Regularization Results

In this section, we first derive the regularization term introduced in section 3.9 in Chapter 3. Then using this regularization term we re-train models to see if we can mitigate robustness bias. As we saw in Figures B.8 and B.7 in Chapter 3, Adience barely showed any robustness bias when partitioned on the sensitive attribute (gender), so in this section we only focus on CIFAR-10 and UTKFace (partitioned by race). We show an in-depth analysis Resnet50, VGG19 and Deep CNN on CIFAR-10 and Resnet50, VGG19 and UTK Classifier on UTKFace.

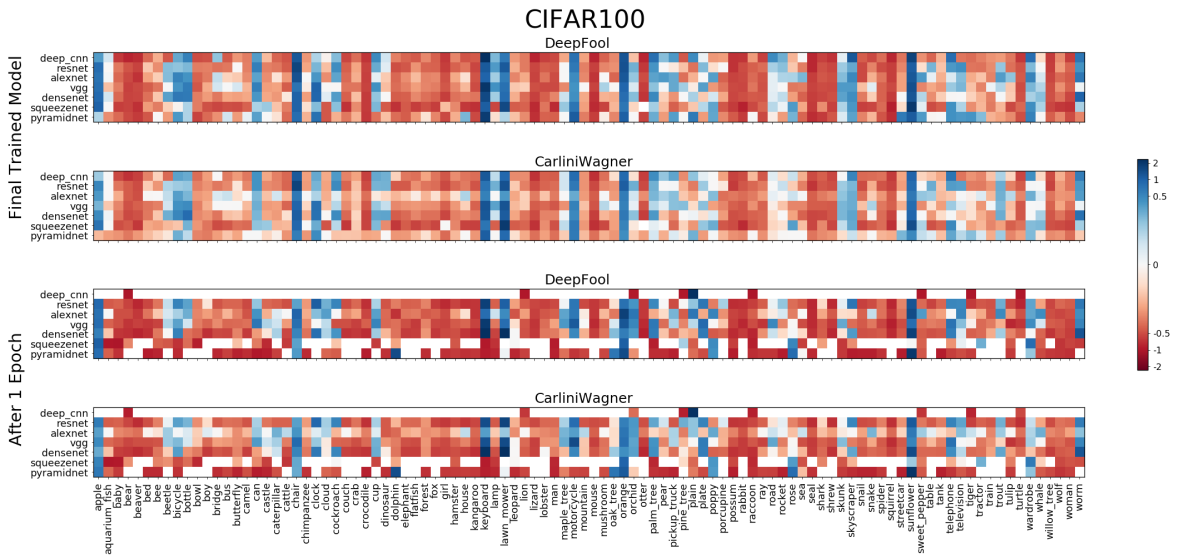


Figure B.3: Depiction of σ_P^{DF} and σ_P^{CW} for the CIFAR100 dataset with partitions corresponding to the class labels $\mathcal{C}$. These values are reported for all five convolutional models both at the beginning of their training (after one epoch) and at the end. We observe that, largely, the signedness of the functions are consistent between the five models and also across the training cycle.

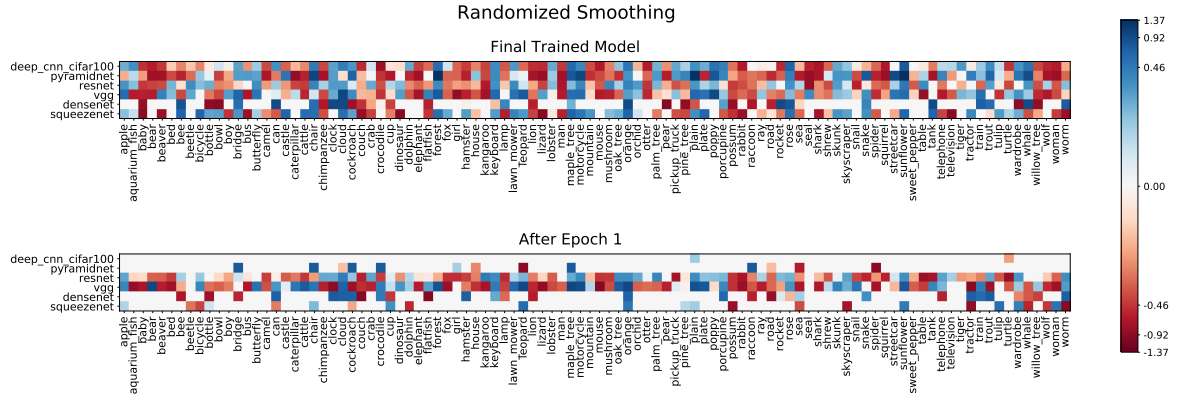


Figure B.4: Depiction of σ_P^{RS} for the CIFAR100 dataset with partitions corresponding to the class labels $\mathcal{C}$.

B.2.1 Formal Derivation of the Regularization Term

In this section, we provide a step-by-step derivation of the regularization term used in Section 3.9 in Chapter 3. Recall the traditional Empirical Risk Minimization objective, ERM $:=$ $l_{cls}(f_\theta(X), Y)$, where l_{cls} is cross entropy loss. Now we wish to model our measure of fairness

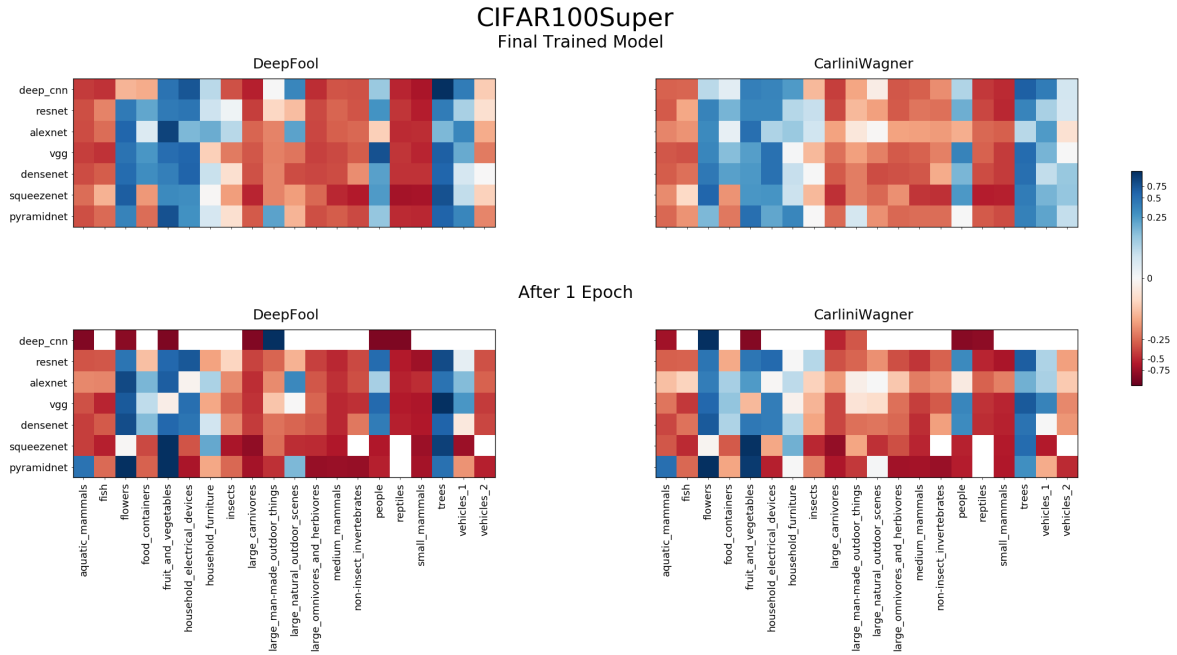


Figure B.5: Depiction of σ_P^{DF} and σ_P^{CW} for the CIFAR100super dataset with partitions corresponding to the class labels $\mathcal{C}$. These values are reported for all five convolutional models both at the beginning of their training (after one epoch) and at the end. We observe that, largely, the signedness of the functions are consistent between the five models and also across the training cycle.

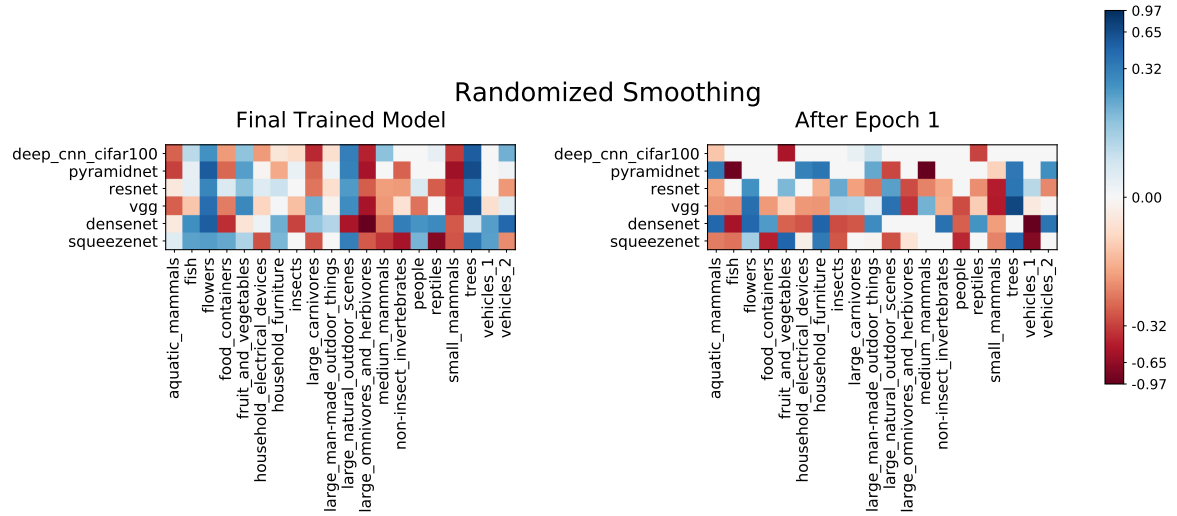


Figure B.6: Depiction of σ_P^{RS} for the CIFAR100super dataset with partitions corresponding to the class labels $\mathcal{C}$.

(§3.3) and minimize for it alongside ERM. To model our measure, we first evaluate the following

cumulative distribution functions:

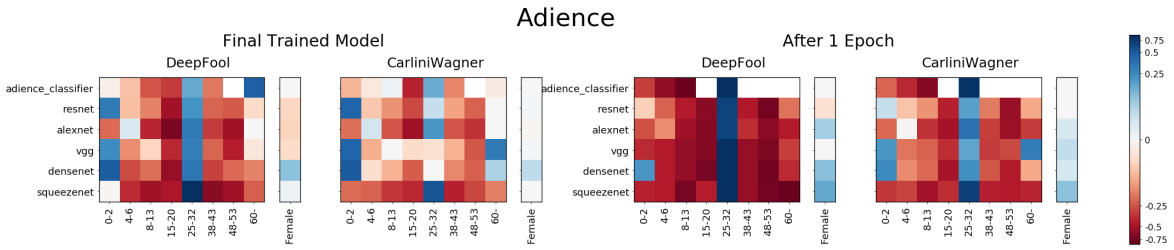


Figure B.7: Depiction of σ_P^{DF} and σ_P^{CW} for the Adience dataset with partitions corresponding to the (1) class labels $\mathcal{C}$ and the and (2) gender. These values are reported for all five convolutional models both at the beginning of their training (after one epoch) and at the end. We observe that, largely, the signedness of the functions are consistent between the five models and also across the training cycle.

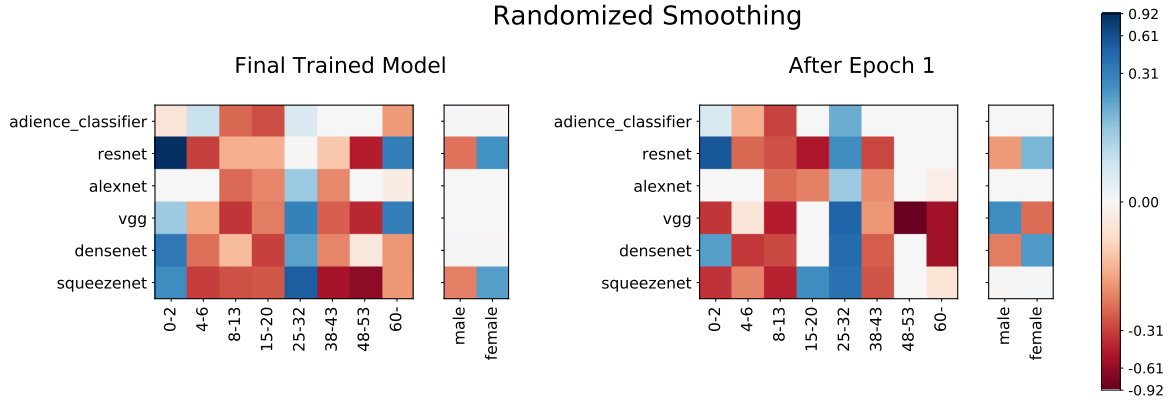


Figure B.8: Depiction of σ_P^{RS} for the Adience dataset with partitions corresponding to the (1) class labels $\mathcal{C}$ and the and (2) gender.

$$\mathbb{P}_{x\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \in P, y = \hat{y}\} =$$

$$\frac{1}{\sum_{x\notin P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x\notin P \\ y=\hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\}$$

$$\mathbb{P}_{x\in\mathcal{D}}\{d_\theta(x) > \tau \mid x \notin P, y = \hat{y}\} =$$

$$\frac{1}{\sum_{x\in P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x\in P \\ y=\hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\}$$

This gives us the empirical estimate of the robustness bias term $RB(P, \tau)$, parameterized

by partition P and threshold τ, defined as Equation B.1 below.

$$\tilde{RB}(P,\tau) = \Big| \frac{1}{\sum_{x \notin P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \notin P \\ y=\hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\} - \frac{1}{\sum_{x \in P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \in P \\ y=\hat{y}}} \mathbb{1}\{d_\theta(x) > \tau\} \Big| \tag{B.1}$$

Now, for $\tilde{RB}$ as defined in Equation B.1 to be computed and used, for example, during training, we must approximate a closed form expression of d_θ. To formulate this, we take inspiration from the way adversarial inputs are created using DeepFool [95]. Just like in DeepFool, we also approximate distance from f_θ considering f_θ to be linear (even though it may be a highly non-linear Deep Neural Network). Thus, we get,

$$d_\theta(x) = \left| \frac{f_\theta(x)}{||\nabla_x f_\theta(x)||} \right|. \tag{B.2}$$

By combining Equations B.1 and B.2, we recover $\tilde{RB}$, a computable estimate of the robustness bias term RB, as follows.

$$\tilde{RB}(P,\tau) = \Big| \frac{1}{\sum_{x \notin P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \notin P \\ y=\hat{y}}} \mathbb{1}\{| \frac{f_\theta(x)}{||\nabla_x f_\theta(x)||} | > \tau\} - \frac{1}{\sum_{x \in P} \mathbb{1}\{y = \hat{y}\}} \sum_{\substack{x \in P \\ y=\hat{y}}} \mathbb{1}\{| \frac{f_\theta(x)}{||\nabla_x f_\theta(x)||} | > \tau\} \Big| \tag{B.3}$$

Finally, given scalar α, we minimize for the new objective function, *AdvERM*, as follows:

$$AdvERM := l_{cls}(f_\theta(X), Y) + \alpha \cdot \tilde{RB}(P, \tau).$$

B.2.2 Additional Regularization Results

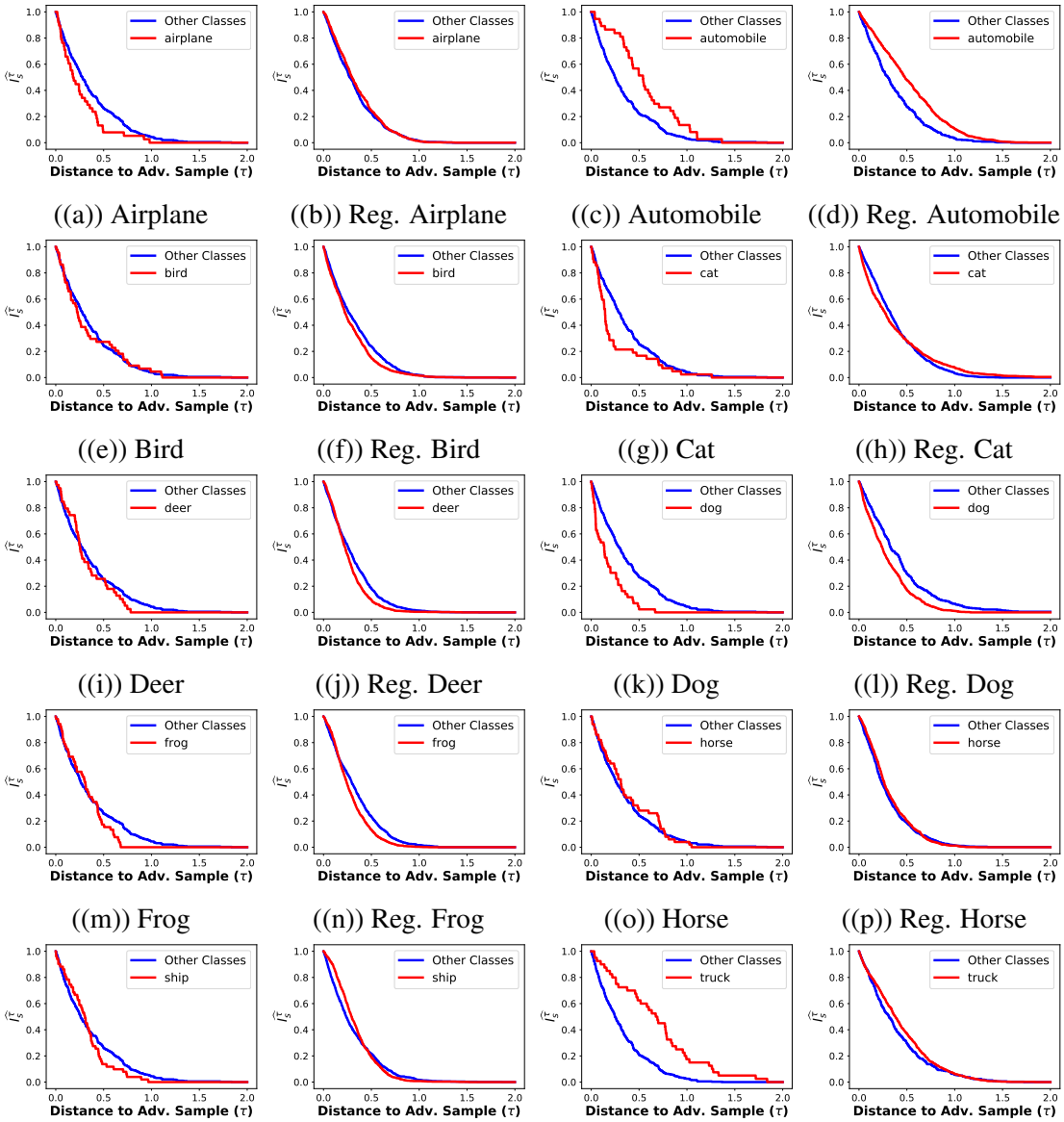
Figures B.9, B.10, B.11, B.12, B.13, B.14 shows how models trained with our proposed regularization term show lesser robustness bias. Figures B.9, B.10, and B.11 correspond to CIFAR-10, while Figures B.12, B.13, and B.14 correspond to UTKFace. For example, for Deep CNN trained on CIFAR-10, for the partition "cat", we see that the distribution of distances become less disparate for the regularized model (Figure B.9(h)) as compared to the original non-regularized model (Figure B.9(g)). This trend persists across models and datasets. We provide a PyTorch implementation of the proposed regularization term in our accompanying code.

B.3 Comparison of Approximation(s) using Randomized Smoothing and Adversarial Attacks

We present the percent agreement in the signedness of σ^{RS} with σ^{DF} and the signedness of σ^{RS} with σ^{CW}. These results are presented in Table B.1.

Table B.1: Percentage of agreement with σ^{RS} and σ^{DF} and σ^{CW}

	DeepFool	CarliniWagner
Adience	83.33	79.63
UTKFace	74.24	72.73
CIFAR10	75.00	76.67
CIFAR100	53.83	54.00
CIFAR100super	65.83	66.67



((a)) Airplane ((b)) Reg. Airplane ((c)) Automobile ((d)) Reg. Automobile

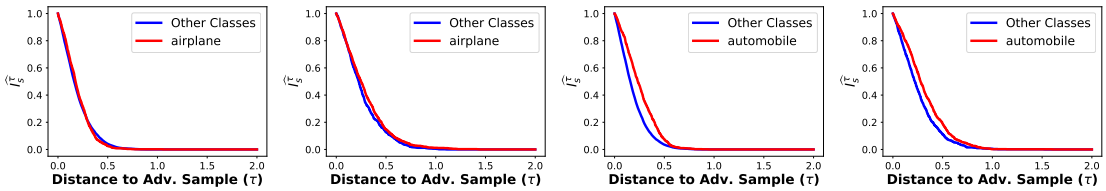
((e)) Bird ((f)) Reg. Bird ((g)) Cat ((h)) Reg. Cat

((i)) Deer ((j)) Reg. Deer ((k)) Dog ((l)) Reg. Dog

((m)) Frog ((n)) Reg. Frog ((o)) Horse ((p)) Reg. Horse

((q)) Ship ((r)) Reg. Ship ((s)) Truck ((t)) Reg. Truck

Figure B.9: [Regularization] CIFAR10 - Deep CNN

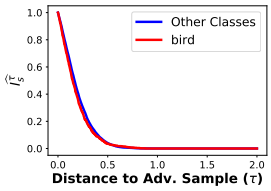


((a)) Airplane

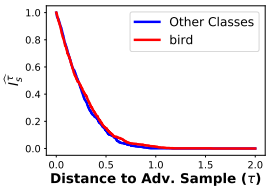
((b)) Reg. Airplane

((c)) Automobile

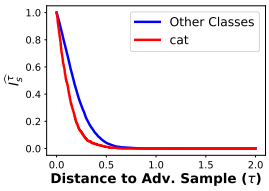
((d)) Reg. Automobile



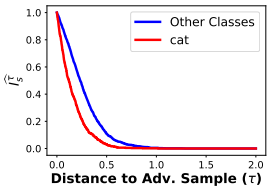
((e)) Bird



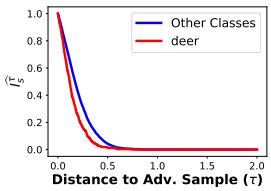
((f)) Reg. Bird



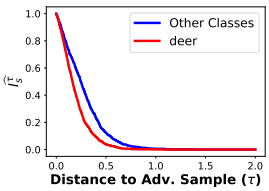
((g)) Cat



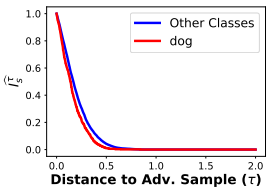
((h)) Reg. Cat



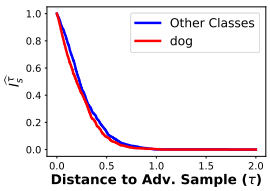
((i)) Deer



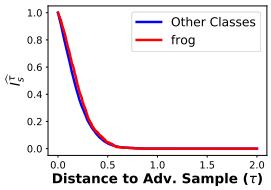
((j)) Reg. Deer



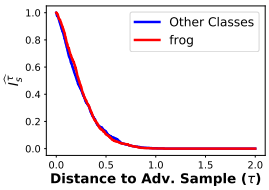
((k)) Dog



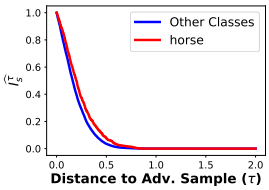
((l)) Reg. Dog



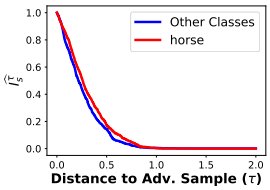
((m)) Frog



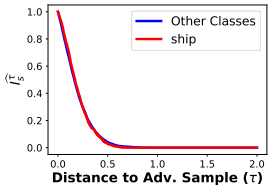
((n)) Reg. Frog



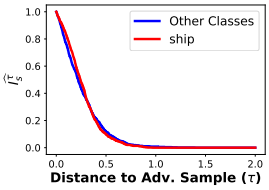
((o)) Horse



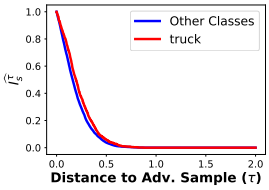
((p)) Reg. Horse



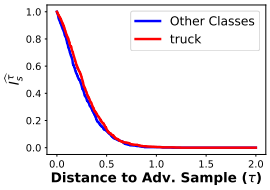
((q)) Ship



((r)) Reg. Ship

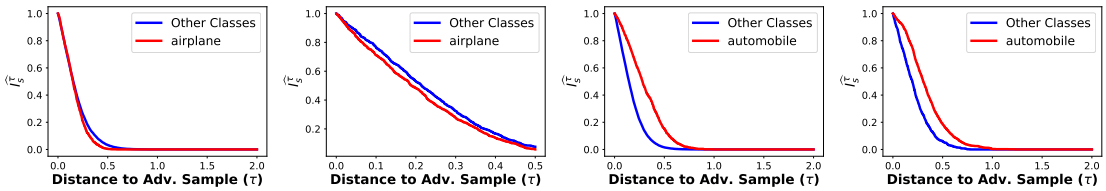


((s)) Truck



((t)) Reg. Truck

Figure B.10: [Regularization] CIFAR10 - Resnet50

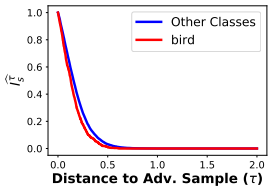


((a)) Airplane

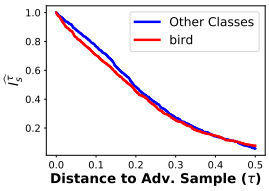
((b)) Reg. Airplane

((c)) Automobile

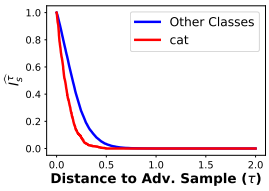
((d)) Reg. Automobile



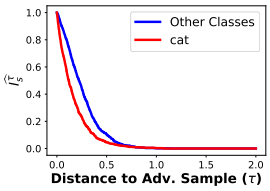
((e)) Bird



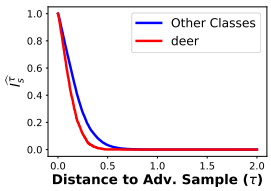
((f)) Reg. Bird



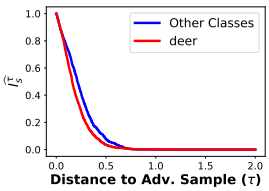
((g)) Cat



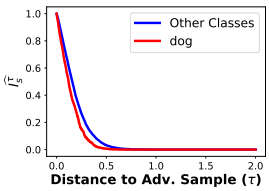
((h)) Reg. Cat



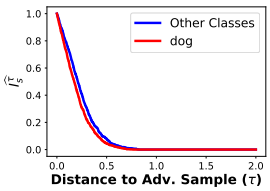
((i)) Deer



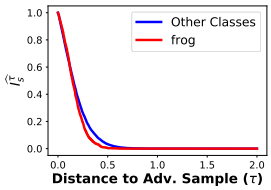
((j)) Reg. Deer



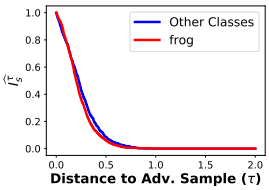
((k)) Dog



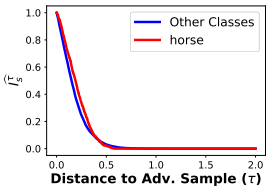
((l)) Reg. Dog



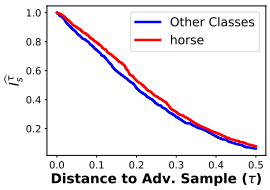
((m)) Frog



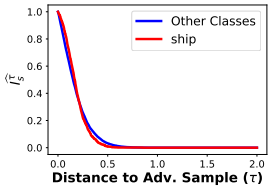
((n)) Reg. Frog



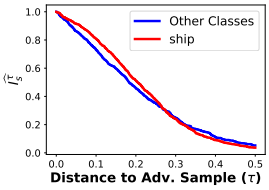
((o)) Horse



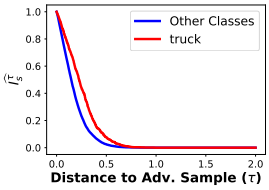
((p)) Reg. Horse



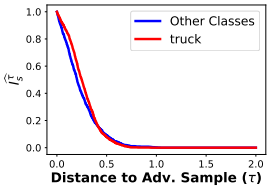
((q)) Ship



((r)) Reg. Ship

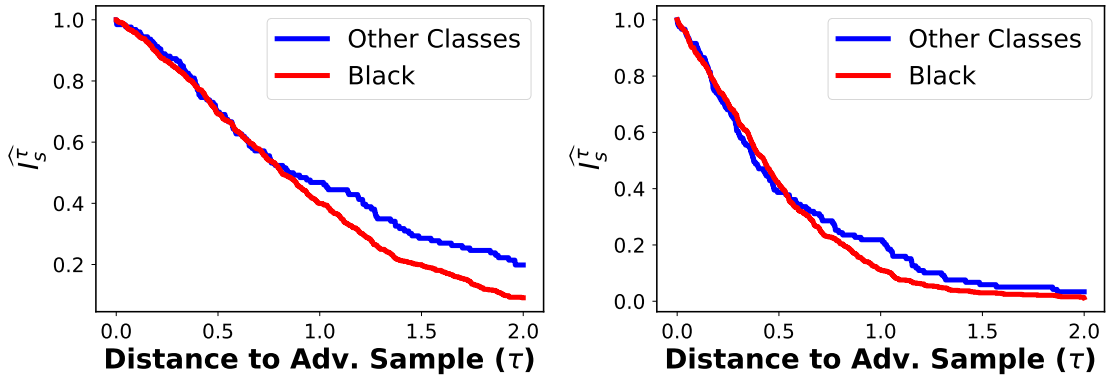


((s)) Truck



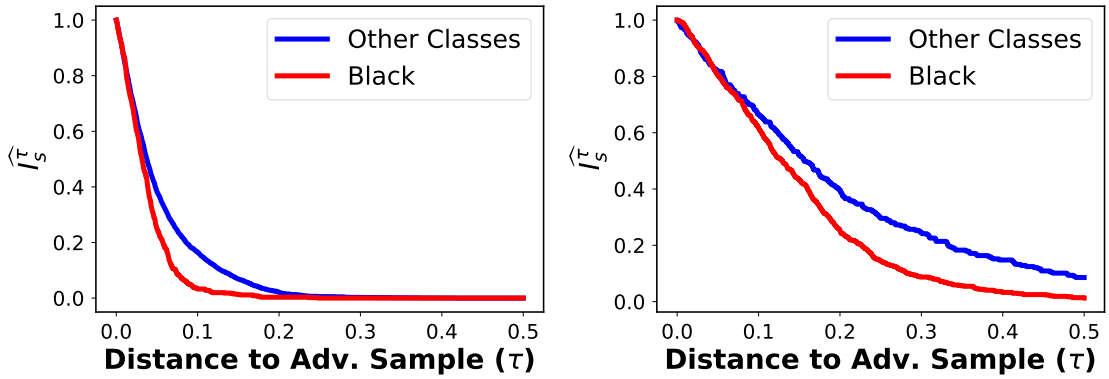
((t)) Reg. Truck

Figure B.11: [Regularization] CIFAR10 - VGG19



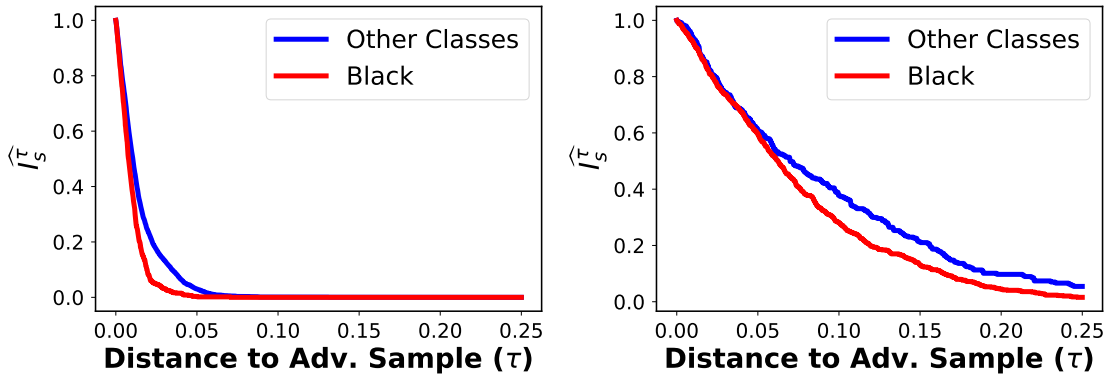
((a)) Partitioned by Race ((b)) Regularized for Race

Figure B.12: [Regularization] UTKFace partitioned by race - UTK Classifier.



((a)) Partitioned by Race ((b)) Regularized for Race

Figure B.13: [Regularization] UTKFace partitioned by race - Resnet50.



((a)) Partitioned by Race ((b)) Regularized for Race

Figure B.14: [Regularization] UTKFace partitioned by race - VGG.

Table B.2: Test data performance of all models on different datasets.

	Deep CNN	PyramidNet	Adience Classifier	UTK Classifier	Resnet50	Alexnet	VGG	Densenet	Squeeze-net
Adience	-	-	48.80	-	49.75	46.04	51.41	50.80	49.49
UTKFace	-	-	-	66.25	69.82	68.09	69.89	69.15	70.73
CIFAR10	86.97	86.92	-	-	83.26	92.08	89.53	85.17	76.97
CIFAR100	59.60	56.42	-	-	55.81	71.31	64.39	61.05	40.36
CIFAR100super	71.78	67.55	-	-	67.27	80.7	76.06	71.22	55.16

Appendix C: Chapter 4, Appendix

C.1 Measuring Human Alignment via Representation Inversion

C.1.1 Measuring Human Perception Similarity

We recruited AMT workers with completion rate $\geq 95\%$ and who spoke English. To further ensure that the workers understood the task, we added attention checks. For 2AFC task, this meant making the query image as the same image as one of the images in the option. In clustering setting, this meant making the image on the row same as one of the images in the columns. All the workers who took our survey passed the attention checks.

We estimated a completion time of about 30 minutes for each survey and thus paid each worker $7.5\$$. We allotted 60 minutes per survey, so workers are not forced to rush through the survey. Most of the workers were able to complete the task in less than 20 minutes. Our study was approved by the Ethical Review Board of our institute.

ImageNet Clustering Hard In order to create the hard ImageNet clustering task we use the human annotations of similarity between ImageNet images collected by ImageNet-HSJ authors [145]. This contains a matrix (M) of similarity scores for each image in ImageNet validation set, where M_{ij} is an indication for similarity between i^{th} and j^{th} images. For each image i (randomly picked), we sample two more images that are the most similar to i as per M_i. This creates a

Table C.1: **[Adversarial IRIs]** We observe that using the adversarial regularizer (described in Section 4.2.2) makes alignment for all models look bad. AT = Adversarial Training, DA = Data Augmentations. For details about standard SimCLR DA and DA without color, see Section C.3.

CIFAR10		
	$x_0 \sim \mathcal{N}(0, 1)$	$x_0 \sim \mathcal{N}(0, 0.01)$
MODEL	ALIGNMENT ON ADVERSARIAL IRIS	
SUPERVISED RESNET18, STANDARD	$1.33_{\pm 1.89}$	$0.00_{\pm 0.00}$
SUPERVISED VGG16, STANDARD	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
SUPERVISED INCEPTIONV3, STANDARD	$0.33_{\pm 0.47}$	$0.00_{\pm 0.00}$
SUPERVISED DENSENET121, STANDARD	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
SUPERVISED RESNET18, AT $\epsilon(\ell_2) = 1$	$0.33_{\pm 0.47}$	$1.00_{\pm 0.00}$
SUPERVISED VGG16, AT $\epsilon(\ell_2) = 1$	$1.00_{\pm 1.41}$	$1.00_{\pm 0.00}$
SUPERVISED INCEPTIONV3, AT $\epsilon(\ell_2) = 1$	$3.00_{\pm 4.24}$	$0.00_{\pm 0.00}$
SUPERVISED DENSENET121, AT $\epsilon(\ell_2) = 1$	$1.33_{\pm 1.89}$	$1.00_{\pm 0.00}$
SUPERVISED RESNET18, SIMCLR DA	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
SIMCLR RESNET18, STANDARD DA	$0.00_{\pm 0.00}$	$1.00_{\pm 0.00}$
SIMCLR RESNET18, DA WITHOUT COLOR	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
SIMCLR RESNET18, STANDARD + ADV DA $\epsilon(\ell_2) = 1$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
IMAGENET		
	$x_0 \sim \mathcal{N}(0, 1)$	$x_0 \sim \mathcal{N}(0, 0.01)$
MODEL	ALIGNMENT ON ADVERSARIAL IRIS	
SUPERVISED RESNET18, STANDARD	$0.00_{\pm 0.00}$	$3.00_{\pm 0.00}$
SUPERVISED RESNET50, STANDARD	$0.00_{\pm 0.00}$	$3.50_{\pm 0.50}$
SUPERVISED VGG16, STANDARD	$0.00_{\pm 0.00}$	$3.00_{\pm 2.00}$
SUPERVISED RESNET18, AT $\epsilon(\ell_2) = 3$	$0.33_{\pm 0.47}$	$2.50_{\pm 1.50}$
SUPERVISED RESNET50, AT $\epsilon(\ell_2) = 3$	$14.00_{\pm 3.74}$	$3.50_{\pm 0.50}$
SUPERVISED VGG16, AT $\epsilon(\ell_2) = 3$	$11.00_{\pm 3.74}$	$4.50_{\pm 1.50}$

task that is much harder to perform for human annotators since the images on the columns look perceptually very similar (See Fig 4.2(b) for an example).

C.1.2 Using LPIPS as a proxy for g_{human}

In order to ensure that LPIPS [140] is a reliable proxy to simulate human perception we measured if LPIPS could simulate human annotators on two perceptual similarity task setups: `2AFC` and `Clustering`, as described in Section 4.2.1. For `2AFC`, this meant using LPIPS to measure the distance between the query image and the two images shown in the options and then matching the query image to the one with lesser LPIPS distance. And similarly in `Clustering`, for each image on the row, we used LPIPS to measure its distance from each of the 3 images in the column and then matched it to the closest one. Our results (Table 4.1 in Chapter 4) show that this can serve as a good proxy for human perception similarity. For all our experiments, we report avergae over 4 different LPIPS backbones: ImageNet trained Alexnet & VGG16, and both of the Imagenet trained Alexnet & VGG16 finetuned by the authors for perceptual similarity `https://github.com/richzhang/PerceptualSimilarity`.

C.1.3 Role of Input Distribution

Fig C.1 shows some examples of inputs sampled from different gaussians (the lighter ones are sampled from $\mathcal{N}(0.5, 2)$ and darker ones from $\mathcal{N}(0, 1)$). We find that completing `2AFC` and `Clustering` tasks on inputs that look like noise to humans is a qualitatively harder task than when doing this on in-distribution target samples.

However, remarkably, we observe that humans are still able to bring out the differences

Table C.2: **[CIFAR10 and ImageNet In-Distr vs OOD Survey Results]** We observe that even on OOD samples that look like noise, humans can still bring out the relative differences between models, *e.g.*, densenet121 on CIFAR10 is still ranked best aligned model on targets sampled from both kinds of noise. Reduced accuracy of humans on noise shows that this identifying similarities between IRIs on OOD samples is a harder task than with in-distribution target samples.

CIFAR10			
	In-Dist.	Noise $\mathcal{N}(0,1)$	Noise $\mathcal{N}(0.5,2)$
Model	Human 2AFC	Human 2AFC	Human 2AFC
ResNet18	$96.00_{\pm 2.55}$	$31.053_{\pm 15.912}$	$64.386_{\pm 7.310}$
VGG16	$38.83_{\pm 7.59}$	$0.351_{\pm 0.496}$	$0.351_{\pm 0.496}$
InceptionV3	$82.00_{\pm 8.44}$	$28.772_{\pm 20.476}$	$2.105_{\pm 1.549}$
Densenet121	$98.67_{\pm 0.24}$	$60.702_{\pm 11.413}$	$68.947_{\pm 16.119}$
IMAGENET			
	In-Dist.	Noise $\mathcal{N}(0,1)$	Noise $\mathcal{N}(0.5,2)$
Model	Human 2AFC	Human 2AFC	Human 2AFC
ResNet18	$93.17_{\pm 5.95}$	$28.748_{\pm 34.491}$	$65.079_{\pm 5.616}$
ResNet50	$99.50_{\pm 0.00}$	$70.745_{\pm 7.409}$	$93.617_{\pm 9.027}$
VGG16	$95.50_{\pm 2.12}$	$29.806_{\pm 19.704}$	$46.914_{\pm 13.827}$

between different models, even when given (a harder) task of matching re-constructed noisy inputs. Table C.2 shows the results of surveys conducted with noisy target samples. It's worth noting that the accuracy of humans drop quite a bit from in-distribution targets, thus indicating that this is indeed a harder task.

C.1.4 Regularizers

For the human-aligned regularizers, we use the ones discussed in [138]. These fall into three broad categories: *frequency penalization*, *transformation robustness*, and *pre-conditioning*.

- **Frequency Penalization**: The goal is to explicitly penalize high-frequency features in the reconstruction (x_r). This is done by adding a regularizer of the form $\mathcal{R}(x) = TV(x) + ||x||_p$, where TV is the total variation and p = 1 [124]. A similar effect of frequency penalization can also be done by ensuring robustness of x_r to blurring, *i.e.*, $\mathcal{R}(x) = ||x - \text{StopGradient}(blur(x))||_2^2$ [147].

- **Transformation Robustness**: This ensures that x_r is such that the representation is same even if we slightly transform x_r. This is achieved by replacing x with $T(x)$ in Eq 4.2. We use T as a composition of color jitter, random scaling, and random rotation.

- **Pre-conditioning**: This involves taking gradient steps in the fourier domain, which decorrelates the pixels in x_r.

For our experiments we find that transformation robustness generates the best looking x_r and thus we report results under *human-learning regularizer* based on x_r generated using *transformation robustness* during representation inversion.

For adversarial regularizer, we report results in Table C.1 and find that such a regularizer can make almost all models look like they have bad alignment.

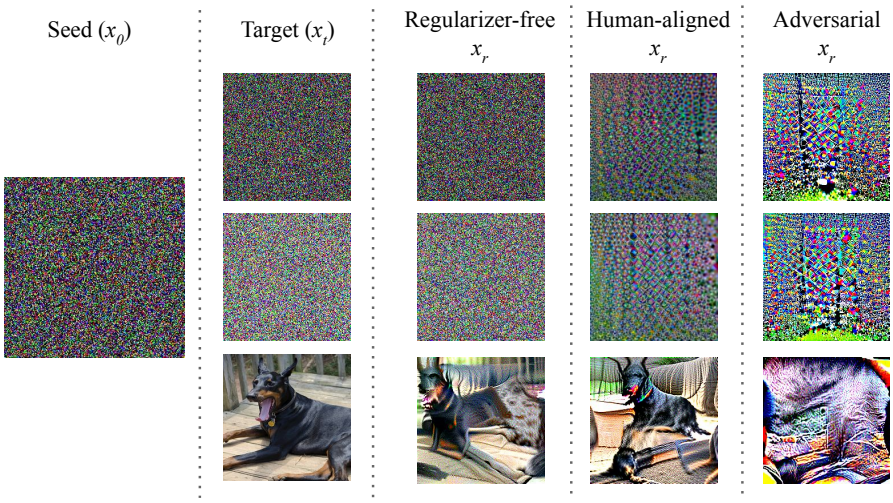


Figure C.1: **[In vs Out of Distribution Samples]** Examples of reconstructions of in-distribution (bottom row) sample vs out-of-distribution samples for an ImageNet trained ResNet50 using the three regularizers mentioned in Section 4.2.2. Here the OOD targets are sampled from two separate random gaussians $\mathcal{N}(0, 1)$ (top row) and $\mathcal{N}(0.5, 2)$ (second row). We see that similar to the in-distribution sample, regularizer-free and adversarial inversions result in x_r resembling and differing from x_t respectively. Interestingly, for human-aligned regularizer, which explicitly tries to remove high-frequency features from x_r fails to reconstruct an x_t that itself consists of high-frequency features.

C.1.5 Role of x_0

We additionally report results for the adversarial regularizer where IRIs were generated from a separate seed. While experiments in Chapter 4 reported for a seed sampled from $\mathcal{N}(0, 1)$, we report results here for a seed sampled from $\mathcal{N}(0, 0.01)$ in Table C.1 and find that regardless of seed, adversarial regularizer makes all models look bad.

C.2 Model, Code, Assets, and Compute Details

C.2.1 Code and Assets

In our code we make use of many open source libraries such as timm [2], pytorch [246], pytorch-lightning [247], numpy [248], robustness [249], matplotlib [250]. timm, pytorch-lightning

and have an Apache 2.0 license. Numpy has a BSD 3-Clause License. Robustness has an MIT license. PyTorch's license can be found here: `https://github.com/pytorch/pytorch/blob/master/LICENSE`, and matplotlib's here: `https://github.com/matplotlib/matplotlib/blob/main/LICENSE/LICENSE`. All these licenses allow free use, modification and distribution. We use publicly available academic datasets CIFAR10/100 [113] and ImageNet [74].

C.2.2 Models

Supervised We used VGG16, ResNet18, Densenet121 and InceptionV3 for experiments on CIFAR10 and CIFAR100. The "robust" version of these models were trained using adversarial training [1], with an ℓ_2, ϵ of 1. All these models were trained using the standard data augmentations(a composition of `RandomCrop`, `RandomHorizontalFlip`, `ColorJitter`, `RandomRotation`). For ImageNet, we used VGG16, ResNet18 and ResNet50 and the "robust" versions of these models were taken from [6] with an ℓ_2, ϵ of 3. ImageNet models used sightly different data augmentationsRandomHorizontalFlip ColorJitter Lighting

Self-supervised We used SimCLR [141] to train a ResNet18 backbone on CIFAR10 and CIFAR100. More details about different types of data augmentations in Section C.3.

ImageNet We used VGG16 (with batchnorm), ResNet18 and ResNet50 for ImageNet. The "robust" versions of these models were taken from [6], who trained these models using adversarial training with an ℓ_2 epsilon of 3.

Table C.3: **[CIFAR10 Models; Effect of Data Augmentation]** For certain models, *e.g.*, adversarially trained resnet18, data augmentation is crucial in learning aligned representations.

ADV. TRAINING				
	RESNET18	DENSENET121	VGG16	INCEPTIONV3
USUAL DATA AUG	$76.50_{\pm 15.91}$	$93.50_{\pm 9.60}$	$0.25_{\pm 0.43}$	$24.25_{\pm 25.17}$
NO DATA AUG	$30.00_{\pm 12.02}$	$93.75_{\pm 8.20}$	$1.00_{\pm 1.73}$	$12.25_{\pm 20.08}$
STANDARD				
	RESNET18	DENSENET121	VGG16	INCEPTIONV3
STRONG DATA AUG	$0.00_{\pm 0.00}$	$1.00_{\pm 1.73}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
USUAL DATA AUG	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$

C.2.3 Compute Details

We used our institute's GPU cluster to run all experiments. Since our experiments involve standard models and datasets, these can be run on any hardware supported by PyTorch. In our case, we used 5 machines with 2 V100 Nvidia Tesla GPUs (32GB each, volta architecture) and a Nvidia dgx machine with 8 Nvidia Tesla P100 GPUs (16GB each). We estimate a total of 500+ GPU hours.

C.3 What Contributes to Good Alignment

SimCLR training details We used data augmentations shown to work best by the authors (a composition of `RandomHorizontalFlip`, `ColorJitter`, `RandomGrayscale` and `GaussianBlur`, as implemented in the original codebase `https://github.com/google-research/simclr`).

We also train SimCLR models without the color augmentations (*i.e.* only `RandomHorizontalFlip` and `GaussianBlur`). Since color transforms were crucial for obtaining representations with good generalization performance, we wanted to analyze how removing augmentations crucial for generalization impacts alignment. Finally, we also train a variant of SimCLR with adversarial data augmentations, as proposed in some recent works [142, 155]. As opposed to traditional adversarial training, here we generate adversarial data augmentations for a model (g) by solving the following maximization for each input x:

$$\text{argmax}_{x'} ||g(x') - g(x)||_2 \;\; \text{st} ||x - x'||_2 \leq \epsilon$$

For our experiments $\epsilon = 1$ for CIFAR10 and $\epsilon = 3$ for ImageNet (similar to supervised models).

Architectures and Loss Function, CIFAR100 & ImageNet Fig C.3 shows results for CIFAR100 and ImageNet for standard and robust training. Similar to previous works, we find that robust models are better aligned with human perception. Interestingly, we find that the variance between different architectures that we observed for CIFAR10 does not exist for CIFAR100 and ImageNet, *i.e.*, regardless of architecture, robustly trained models are well aligned with human perception. Indicating that (unsurprisingly) training dataset also plays a major role in alignment.

SimCLR, CIFAR100 Fig C.2 shows alignment of different types of SimCLR models throughout training. We observe a similar trend as CIFAR10, where adversarial data augmentation improves alignment.

C.3.1 Data Augmentation, CIFAR10, CIFAR100 & ImageNet

Since re-training ImageNet models with adversarial training is very resource intensive, we train ImageNet models using Free Adversarial Training (Free AT) [251]. Free AT only has an implementation for ℓ_{inf} threat model, hence we train these models with ℓ_{inf}, $\epsilon = 4/255$ (for both with and without data augmentation). Table C.4 shows that similar to CIFAR10 (Table C.3), for some models, like ResNet18, data augmentation is crucial in learning aligned representations (despite being trained to be adversarially robust). For other models, data augmentation never hurts alignment (except InceptionV3 for CIFAR100).

Table C.4: **[CIFAR100 & ImageNet Models all trained to be adversarially robust; Effect of Data Augmentation]** Similar to CIFAR10, data augmentation is crucial for adversarially trained resnet18 to learn aligned representations.

CIFAR100				
	RESNET18	DENSENET121	VGG16	INCEPTIONV3
USUAL DATA AUG	$82.50_{\pm 20.85}$	$88.00_{\pm 14.51}$	$58.75_{\pm 32.15}$	$69.25_{\pm 26.39}$
NO DATA AUG	$81.50_{\pm 15.58}$	$89.75_{\pm 15.01}$	$46.25_{\pm 32.25}$	$84.75_{\pm 13.70}$

IMAGENET		
	RESNET18	RESNET50
USUAL DATA AUG	$13.00_{\pm 22.52}$	$0.00_{\pm 0.00}$
NO DATA AUG	$0.75_{\pm 1.30}$	$0.00_{\pm 0.00}$

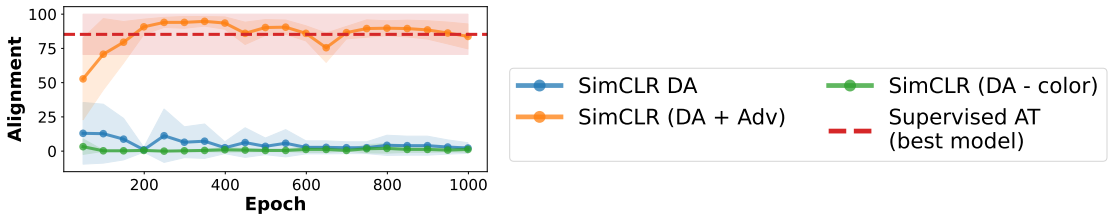


Figure C.2: ResNet18 backbone trained using SimCLR on CIFAR100.

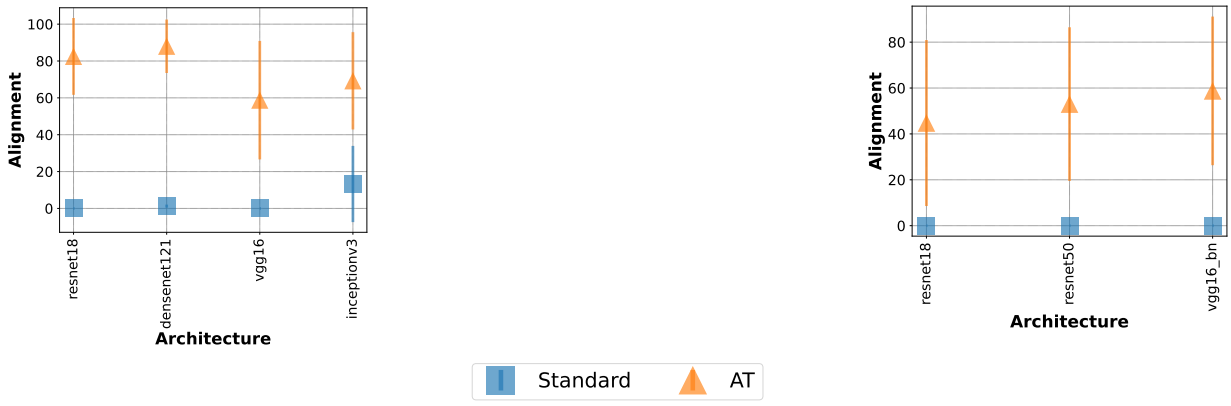


Figure C.3: Role of Loss Function in Alignment; CIFAR100 (left), and ImageNet (right)

Appendix D: Chapter 5, Appendix

D.1 Representation Similarity Measures

A recent line of work has studied measures for the similarity of representations in deep neural networks. Given two models m_1 and m_2 and a set Z of input points to both, all of these measures quantify the degree of representational similarity between two sets of representations $X = m_1(Z) \in \mathbb{R}^{n \times d_1}$ and $Y = m_2(Z) \in \mathbb{R}^{n \times d_2}$ as $\mathrm{S_r}(X, Y)$ and are defined as follows:

SVCCA [163] is computed via the following steps:

1. Compute X', Y' by pruning X, Y using SVD to only retain the first k_1 and k_2 principal components that are necessary to retain 99% of the variance, respectively.

2. Perform Canonical Correlation Analysis CCA (X', Y') yielding $k = \min(k_1, k_2)$ correlation coefficients $\rho_1, \ldots, \rho_k$.

3. Return $\mathrm{S_r}(X, Y) = \frac{1}{k} \sum_{i=1}^{k} \rho_i$

PWCCA [164] is computed via the following steps:

1. Perform Canonical Correlation Analysis CCA (X, Y) yielding $k = min(d_1, d_2)$ correlation coefficients $\rho_1, \ldots, \rho_k$ and CCA vectors $h_1, \ldots, h_k$

2. Compute weights $\alpha_i = \sum_{j=1}^{d_1} |\langle h_i, z_j \rangle|$, where the z_j's are the d_1 columns of X

3. Return $\mathrm{S_r}(X, Y) = \frac{\sum_{i=1}^{k} \alpha_i \rho_i}{\sum_{i=1}^{k} \alpha_i}$

(Linear) CKA [7] is computed via the following steps:

1. Compute similarity matrices $K = XX^T$, $L = YY^T$
2. Compute normalized versions $K' = HKH$, $L' = HLH$ of the similarity matrix using centering matrix $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$
3. Return CKA $(X, Y) = \frac{\text{HSIC}(K,L)}{\sqrt{\text{HSIC}(K,K)\text{HSIC}(L,L)}}$, where $\text{HSIC}(K, L) = \frac{1}{(n-1)^2}\text{flatten}(K') \cdot \text{flatten}(L')$

D.2 Model Architecture, Hardware, Training, and Other Details

We use ResNet18, ResNet34 [119], VGG16 and VGG19 [118] trained on CIFAR10/100 [113] using the standard crossentropy loss and other adversarially robust training methods such as AT, TRADES and MART. All of our experiments are performed on standard models and datasets that can fit on standard GPUs. We also attach our code to reproduce the numbers. For all purposes of adversarial training we use the ℓ_2 threat model with $\epsilon = 1.0$ (see [1] for details). Additionally TRADES and MART require another hyperparameter β (that balances adversarial robustness and clean accuracy) which we set to 1.0 for our experiments.

D.3 STIR Faithfully Measures Shared Invariance

To confirm that STIR is correct in assigning different scores to two (same) models trained from different initializations, we generate controversial stimuli [148] for all configurations in

Table D.1: $(\mathrm{m^{(Vanilla)}}_1, \mathrm{m^{(Vanilla)}}_2)$, $(\mathrm{m^{(AT)}}_1, \mathrm{m^{(AT)}}_2)$, and $(\mathrm{m^{(Vanilla)}}_1, \mathrm{m^{(AT)}}_2)$. Such stimuli are generated by solving the following optimization for a training data point x and any two models m_1 and m_2 :

$$\text{argmin}_{x_c} \; ||m_1(x) - m_1(x_c)||_2 - ||m_2(x) - m_2(x_c)||_2 \tag{D.1}$$

Here we assume $m_1(.)$ and $m_2(.)$ are penultimate layer representations of the respective models. This process generates x_c for every point x such that $||m_1(x) - m_1(x_c)||_2$ is low and $||m_2(x) - m_2(x_c)||_2$ is high. Since these x_c are "perceived" very differently by the two models (wrt the original x), they are called *controversial stimuli*. We use the empirical mean of the amount of perturbation needed (*i.e.*$\delta = \mathbb{E}_{x_c,x}[||x - x_c||_2]$) as a measure of ease of generating controversial stimuli [1]. For a pair of models (m_1 , m_2) where it's easy to generate such x_c, the shared invariance should be low. We see that this is indeed the case as indicated by numbers in Table D.1.

D.4 Experiments: Insights using STIR

D.4.1 Role of Different Random Initialization, Different Training Datasets

Fig D.1&D.2 shows results for different random initializations and different datasets respectively. We see similar trends hold for deeper models such as ResNet34 and VGG19.

D.4.2 Different Architectures, Penultimate Layer

Fig D.3 shows that trends for STIR hold across different choice of seeds. We also see that in contrast CKA assigns high value across the board and is thus not a faithful measure of shared

[1] High(er) values of δ indicate it was hard(er) to generate controversial stimuli

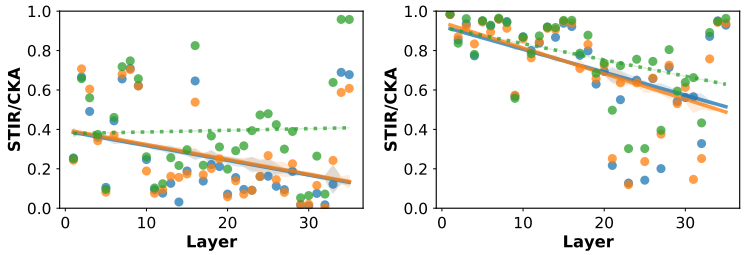
Table D.1: **[STIR faithfully estimates shared invariance]** Here the two ResNet18s in each column are trained on CIFAR10 with different random initializations, holding every other hyperparameter constant. 1.) For two such models trained using the vanilla crossentropy loss (left), interestingly, we find that STIR highlights a lack of shared invariance, whereas CKA overestimates this value; 2.) when both models are trained using adversarial training [1] (middle) STIR faithfully estimates high shared invariance; 3.) Finally STIR is able show how having a directional measure can bring out the differences when comparing a model trained with vanilla loss and adv training (right), whereas CKA being unidirectional cannot derive these insights. All numbers are computed over 1000 random samples from CIFAR10 training set and averaged over 5 runs. Additionally we show Δ between original inputs (X) and controversial stimuli (X') for a given configuration of m_1 and m_2. We see that controversial stimuli are "hard" to generate for higher STIR scores. Here hardness is indicated by the amount of ℓ_2 perturbation needed to generate controversial stimuli.

	ResNet18 Vanilla (m_1), ResNet18 Vanilla (m_2)		ResNet18 AT (m_1), ResNet18 AT (m_2)		ResNet18 AT (m_1), ResNet18 Vanilla (m_2)	
	$m_1 \vert m_2$	$m_2 \vert m_1$	$m_1 \vert m_2$	$m_2 \vert m_1$	$m_1 \vert m_2$	$m_2 \vert m_1$
STIR	$0.605_{\pm 0.013}$	$0.562_{\pm 0.023}$	$0.934_{\pm 0.003}$	$0.939_{\pm 0.002}$	$0.405_{\pm 0.020}$	$0.509_{\pm 0.011}$
STIR $_{adv}$	$0.085_{\pm 0.004}$	$0.064_{\pm 0.004}$	$0.096_{\pm 0.007}$	$0.078_{\pm 0.005}$	$0.054_{\pm 0.004}$	$0.070_{\pm 0.004}$
CKA	$0.967_{\pm 0.000}$		$0.937_{\pm 0.000}$		$0.536_{\pm 0.000}$	
Acc($m_1(X')$, $m_2(X')$)	$0.521_{\pm 0.061}$	$0.347_{\pm 0.029}$	$0.891_{\pm 0.007}$	$0.901_{\pm 0.002}$	$0.140_{\pm 0.028}$	$0.555_{\pm 0.012}$
Δ ($= \frac{1}{n} \sum \vert\vert X - X' \vert\vert_2$)	$8.588_{\pm 0.491}$	$8.570_{\pm 0.474}$	$17.106_{\pm 3.204}$	$16.210_{\pm 2.193}$	$19.270_{\pm 2.202}$	$7.368_{\pm 0.602}$

invariance.

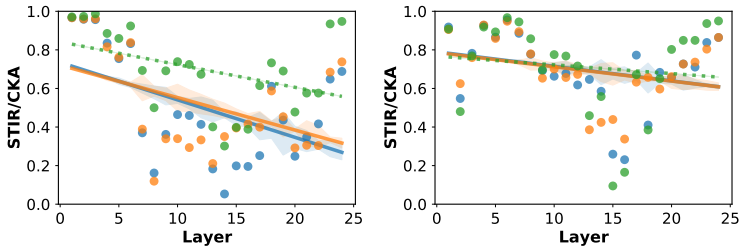
D.4.3 Different Adversarial Training Methods

Fig D.4 shows results on VGG16. These are much higher than values for ResNet18, thus showing that these methods achieve robustness differently across architectures. For AT and MART, we see lower shared invariances even for VGG16.s



((a)) 2 ResNet34 with different init, Vanilla

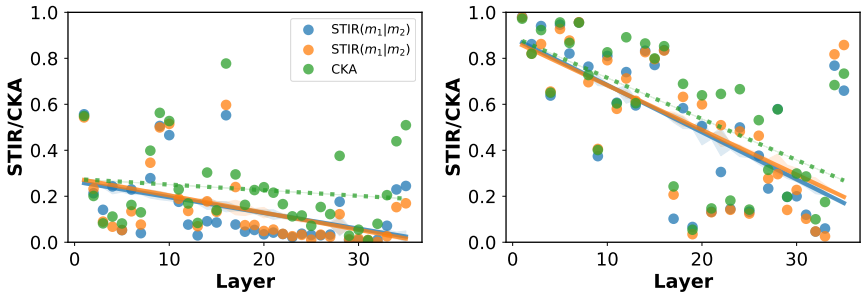
((b)) 2 ResNet34 with different init, AT



((c)) 2 VGG19 with different init, Vanilla

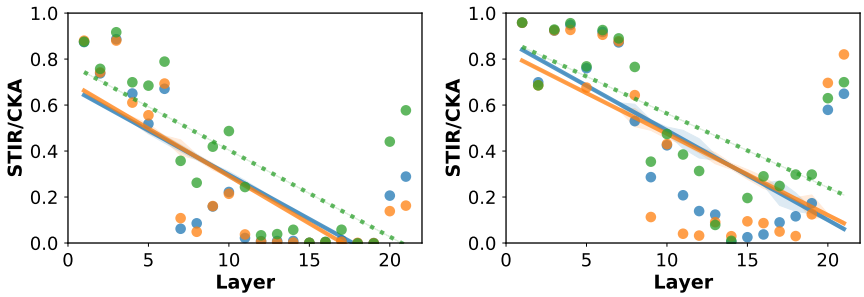
((d)) 2 VGG19 with different init, AT

Figure D.1: **[Role of Random Initialization; CIFAR10]** For models trained using the Vanilla crossentropy loss, we find that only initial layers have high shared invariances. However, when the training procedure explicitly introduces invariances (*e.g.*adversarial training), then all layers converge to having similarly high shared invariances. We see similar trends for ResNet34 and VGG19 here.



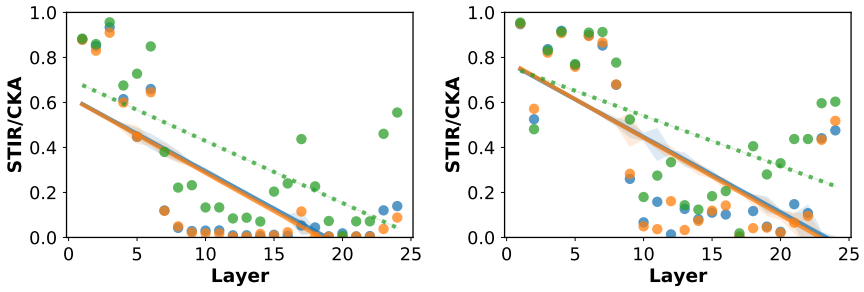
((a)) m_1 = ResNet34 on CIFAR10 w ERM m_2 = ResNet34 on CIFAR100 w ERM

((b)) m_1 = ResNet34 on CIFAR10 w AT m_2 = ResNet34 on CIFAR100 w AT



((c)) m_1 = VGG16 on CIFAR10 w ERM m_2 = VGG16 on CIFAR100 w ERM

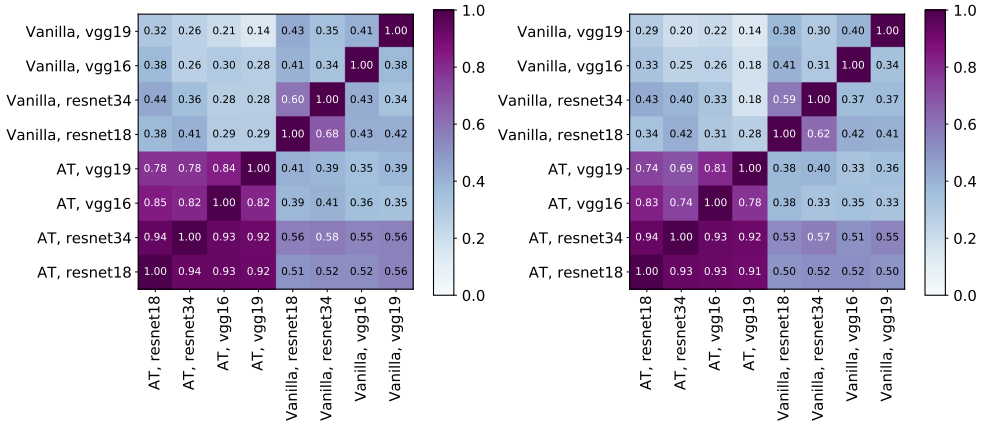
((d)) m_1 = VGG16 on CIFAR10 w AT m_2 = VGG16 on CIFAR100 w AT



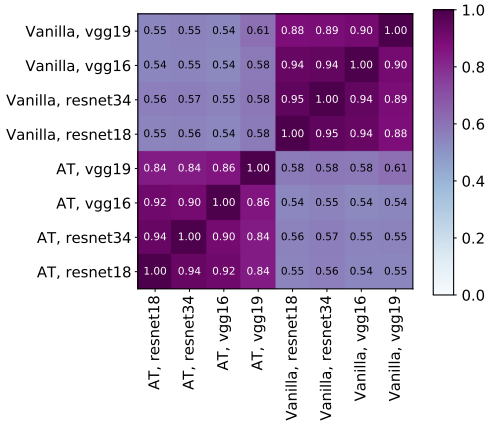
((e)) m_1 = VGG19 on CIFAR10 w ERM m_2 = VGG19 on CIFAR100 w ERM

((f)) m_1 = VGG19 on CIFAR10 w AT m_2 = VGG19 on CIFAR100 w AT

Figure D.2: **[Different Datasets; ResNet34, VGG16 and VGG19 on CIFAR10 and CIFAR100]** Similar to the finding of [7] we find that across datasets, initial layers tend to have more shared invariances. However, we also find that shared invariances increases substantially for all layers with adversarial training (AT).

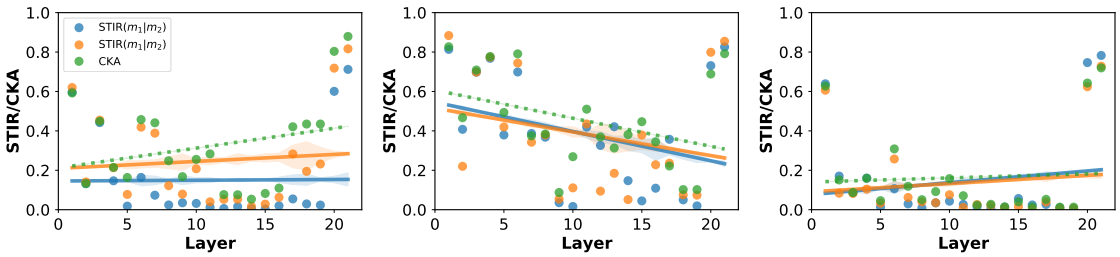


((a)) STIR - completely random seeds ((b)) STIR - seeds sampled from $\mathcal{N}(0, 1)$



((c)) CKA

Figure D.3: **[Different Architectures, Penultimate Layer Shared Invariances; CIFAR10]** We find that in general ResNets have high shared invariances when trained using the same loss, but this shared invariance drops across training types. We also see that with AT, even models with different architectures converge to high shared invariances. Here additionally we show comparison with CKA which does not provide any faithful estimate of shared robustness and is uniformly high for similarly trained models. We also see that this trend holds even when different seeds are used to generate X'.



((a)) AT (m_1) TRADES (m_2) ((b)) AT (m_1) MART (m_2) ((c)) TRADES (m_1) vs MART (m_2)

Figure D.4: **[Different Types of Adversarially Robust Training; VGG16 on CIFAR10]** We se much high levels of shared invariance for VGG16 than for ResNet18. Even for VGG16, AT and MART achieve ell_p ball robustness in different ways.

Appendix E: Chapter 6, Appendix

E.1 Measuring Diffused Redundancy

E.1.1 CKA Definition

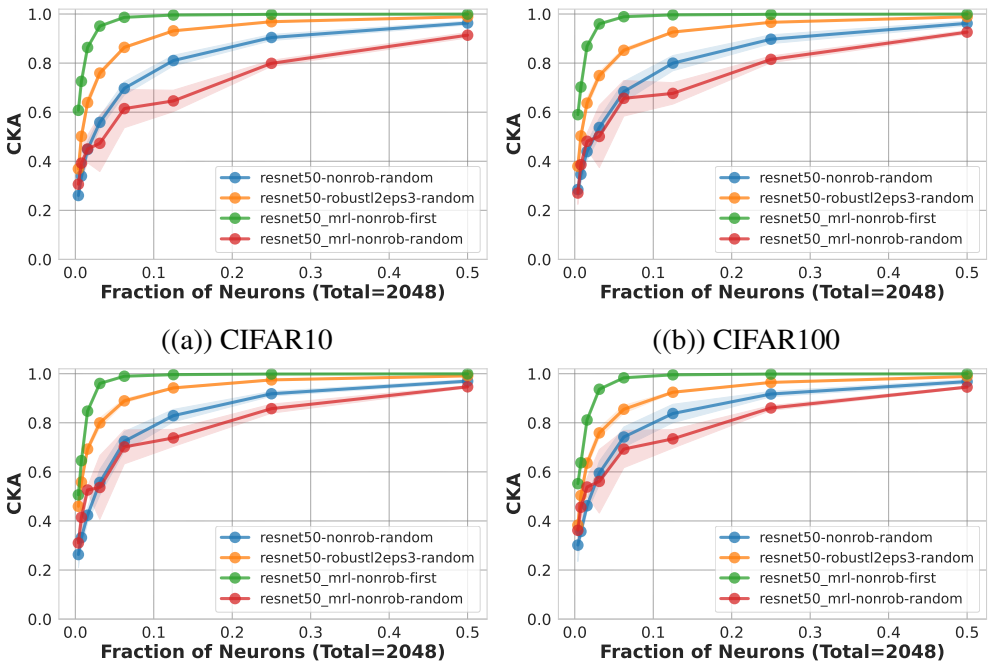
In all our evaluations we use CKA with a linear kernel [7] which essentially amounts to the following steps:

1. Take two representations $Y \in \mathbb{R}^{n \times d1}$ and $Z \in \mathbb{R}^{n \times d2}$
2. Compute dot product similarity within these representation, *i.e.*compute $K = YY^T$, $L = ZZ^T$
3. Normalize K and L to get $K' = HKH$, $L' = HLH$ where $H = I_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$
4. Return $\text{CKA}(Y, Z) = \frac{\text{HSIC}(K,L)}{\sqrt{\text{HSIC}(K,K)\text{HSIC}(L,L)}}$, where $\text{HSIC}(K, L) = \frac{1}{(n-1)^2}(\text{flatten}(K') \cdot \text{flatten}(L'))$

We use the publicly available implementation of [15], which provides an implementation that can be calculated over multiple mini-batches: `https://github.com/nvedant07/STIR`

E.1.2 Additional CKA results

Fig E.1 shows CKA comparison between randomly chosen parts of the layer and the full layer for different kinds of ResNet50. We observe that even ResNet50 trained with MRL loss shows a significant amount of diffused redundancy.



((a)) CIFAR10 ((b)) CIFAR100

((c)) Flowers ((d)) Oxford-IIIT-Pets

Figure E.1: **[Comparison of Diffused Redundancy in MRL vs other losses, through the lens of CKA]** We see a similar trend as reported in Fig 6.7 in Chapter 6, where even the MRL model shows a significant amount of diffused redundancy despite being explicitly trained to instead have structured redundancy. The amount of diffused redundancy however is much lesser than the resnets trained using the standard loss and adv. training as denoted by a much lower red line across all datasets.

E.2 Training and Pre-Processing Details for Reproducibility

Here we list the sources of weights for the various pre-trained models used in our experiments:

- ResNet18 trained on ImageNet1k using standard loss: taken from `timm` v0.6.1.

- ResNet18 trained on ImageNet1k with adv training: taken from Salman et al. [6]:
- ResNet50 trained on ImageNet1k using standard loss: taken from `timm` v0.6.1.
- ResNet50 trained on ImageNet1k with adv training: taken from Salman et al. [6]: `https://github.com/microsoft/robust-models-transfer`.
- ResNet50 trained on ImageNet1k using MRL and with different final layer widths (`resnet50_ffx`): taken from released weights of by Kusupati et al. [10]: `https://github.com/RAIVNLab/MRL`.
- WideResNet50-2 on ImageNet1k both standard and avd. training: taken from Salman et al. [6]: `https://github.com/microsoft/robust-models-transfer`.
- VGG16 trained on ImageNet1k with standard loss: taken from `timm` v0.6.1.
- VGG16 trained on ImageNet1k with adv training: taken from Salman et al. [6]: `https://github.com/microsoft/robust-models-transfer`.
- ViTS32 & ViTS16 trained on ImageNet21k & ImageNet1k: taken from weights released by Steiner et al. [198]: `https://github.com/google-research/vision_transformer`.

All linear probes trained on the representations of these models are trained using SGD with a learning rate of 0.1, momentum of 0.9, batch size of 256, weight decay of $1e-4$. The probe is trained for 50 epochs with a learning rate scheduler that decays the learning rate by 0.1 every 10 epochs. Scripts for training can also be found in the attached code.

For pre-processing, we re-size all inputs to 224x224 (size used for pre-training) and apply the usual composition of RandomHorizontalFlip, ColorJitter(brightness=0.25, contrast=0.25, saturation=0.25,

hue=0.25), RandomRotation(degrees=2). All inputs were mean normalized. For imagenet1k pre-trained models: mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225]. For imagenet21k pre-trained models: mean = [0.5,0.5,0.5], std = [0.5,0.5,0.5].

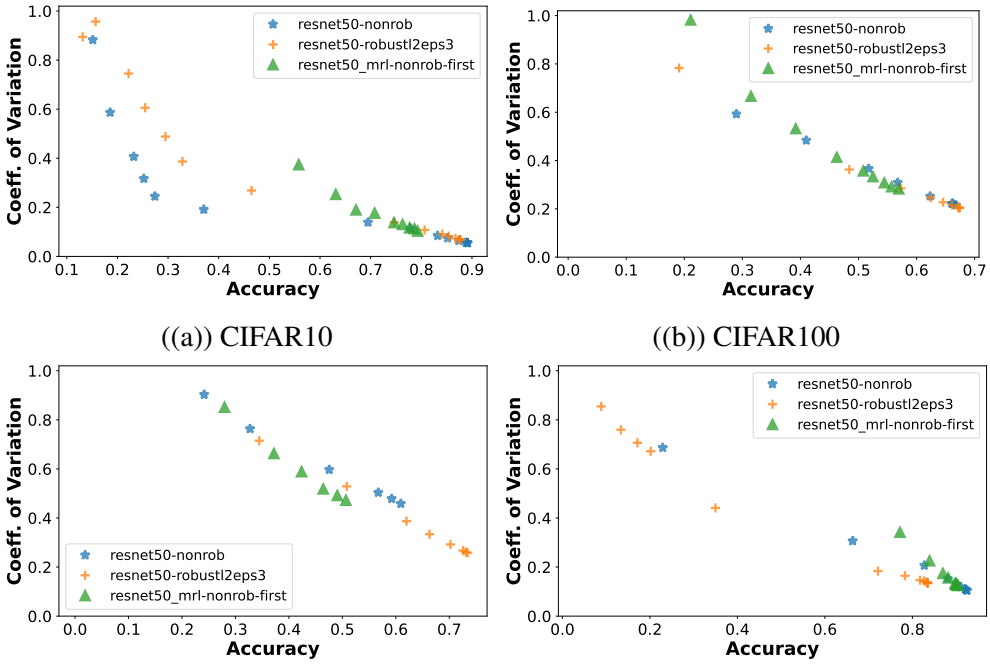
E.3 Deeper Analysis of Fairness-Efficiency Tradeoff in Section 6.4

Analyzing Error Distributions To ensure that the higher gini coefficient shown in Fig 6.8 as we drop more neurons is not merely an artifact of lower overall accuracy, we plot class-wise accuracies as we drop neurons (Figs E.3, E.4 & E.5). We find that for the entire layer, accuracy starts at an almost uniform distribution, and while overall accuracy deteriorates as we drop neurons, the drop comes at a larger cost for a few classes resulting in disparate inter-class accuracies.

Coeff of Variation for Measuring Inequality in Inter-Class Accuracy Fig E.2 shows results for the same analysis shown in Fig 6.8 of Chapter 6 and we find similar takeaways even when using the coefficient of variation as a measure of inequality.

E.4 Corresponding Diffused Redundancy Estimates For Analyses in Section 6.3 & ℓ_∞ Robust Model Results

Corresponding diffused redundancy (DR) ablations for Figures 6.4,6.5,6.7. These are shown in Figures E.7,E.8,E.9 respectively. This should allow for easy comparison of diffused redundancy (lines that are more outside have higher DR). For example, Figure E.8 clearly shows higher diffused redundancy in models trained on larger upstream datasets (here ImageNet21k) since these curves lie more on the outside of the same model's curves for ImageNet1k.



((a)) CIFAR10 ((b)) CIFAR100

((c)) Flowers ((d)) Oxford-IIIT-Pets

Figure E.2: **[Coefficient of Variation As We Drop Neurons]** We see a similar trend as reported in Fig 6.8 of Chapter 6 where inequality increases as we drop neurons for all models on all datasets.

Additionally, we show numbers on x-axis for Figure 6.4 in Figure E.10. Figures 6.5 and 6.7 compare models with same number of neurons in the final layer and hence trends shown with fraction on the x-axis will be exactly the same with absolute numbers on the x-axis. However, Figure E.10 allows a direct comparison of the performance of the same absolute number of neurons across different models.

We report results for ℓ_∞ robust models (with $\epsilon = 4/255$) in Fig E.6 and find that ℓ_2 model generally shows a greater degree of diffused redundancy.

E.5 Results on Intermediate Layers

We additionally ran our experiment on other intermediate layers and report the results in Fig E.11. We present results for a ResNet50 pretrained on ImageNet1k using the standard CrossEntropy loss. The intermediate layers considered are characterized as activations following each residual connection within distinct ResNet blocks. `layerX.Y.act3` means the Yth residual connection in the Xth ResNet block and act3 indicates that we're taking the value after the activation (ReLU) has been applied.

E.6 Results on Harder Downstream Tasks: ImageNetV2 and Places365

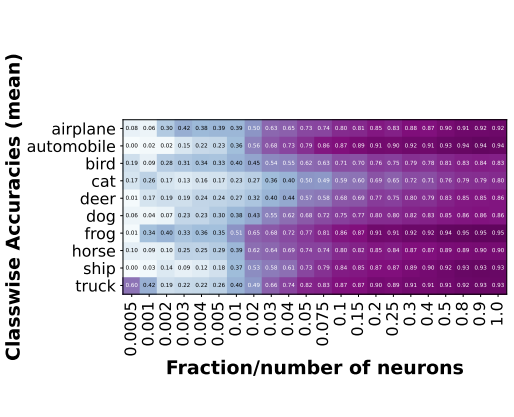
We report results on harder downstream tasks such as ImageNet1k, ImageNetV2, and Places365 in Figure E.12. We find that when randomly dropping neurons, the model is still able to generalize to ImageNet1k and Places365 with very few neurons, *i.e.*, the phenomena of diffused redundancy observed for smaller datasets, also holds for harder datasets. Interestingly we also observe that the accuracy gap between ImageNet1k and ImageNetV2 is maintained even as we drop neurons.

E.7 Effects of Explicitly Preventing Co-adaptation of Neurons: Analysis of Dropout and DeCov

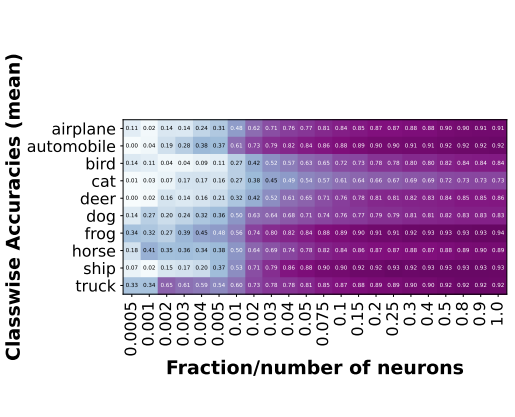
Regularizers such as dropout and DeCov, force different parts of the representations to not be correlated. Thus these regularizers can be seen as explicitly requiring different, compact parts of the representation to be self-contained for the downstream task. Thus, intuitively, such

methods should increase diffused redundancy. Here we investigate if our observation about diffused redundancy is influenced by such regularizers. We evaluate ResNet50 pre-trained on ImageNet1k with dropout in the penultimate layer ranging from a strength of 0.1 all the way to 0.8. We also train another ResNet50 model with the DeCov regularizer added to the usual crossentropy loss and put a weight of 0.0001 on the regularizer to ensure that its numerical range is similar to that of the cross entropy loss term.

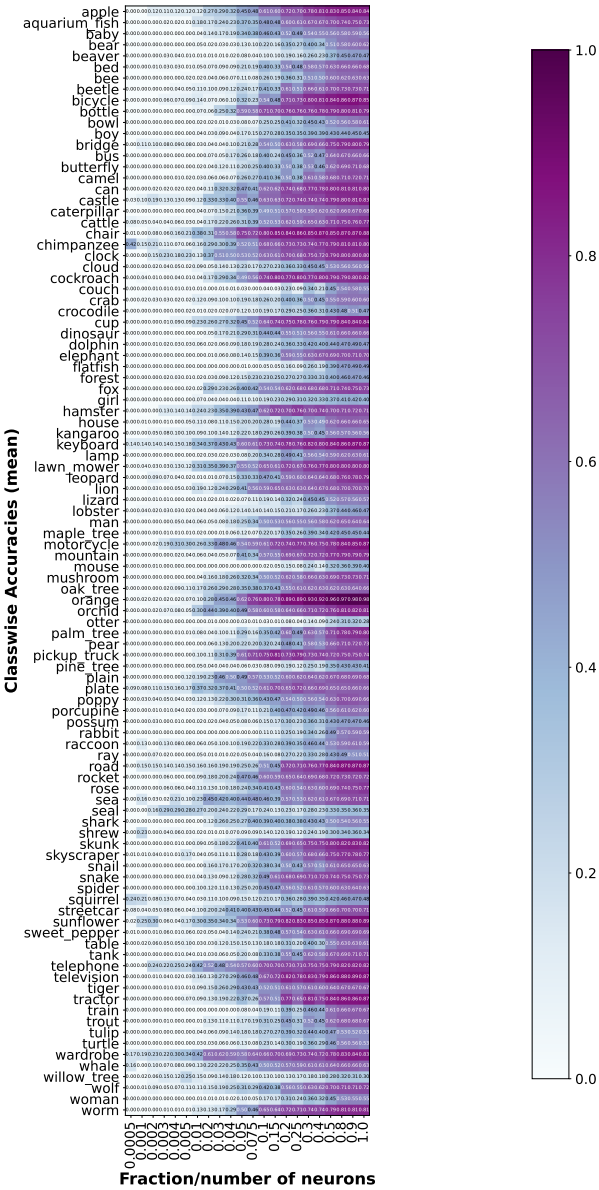
Results in Figures E.13 & E.14 suggest that such regularizers have *almost no effect* on diffused redundancy. Trends across datasets remain consistent regardless of the strength of dropout or the weight given to DeCov regularizer. This observation further adds to the evidence that diffused redundancy is likely to be a natural property of representations learned by DNNs.



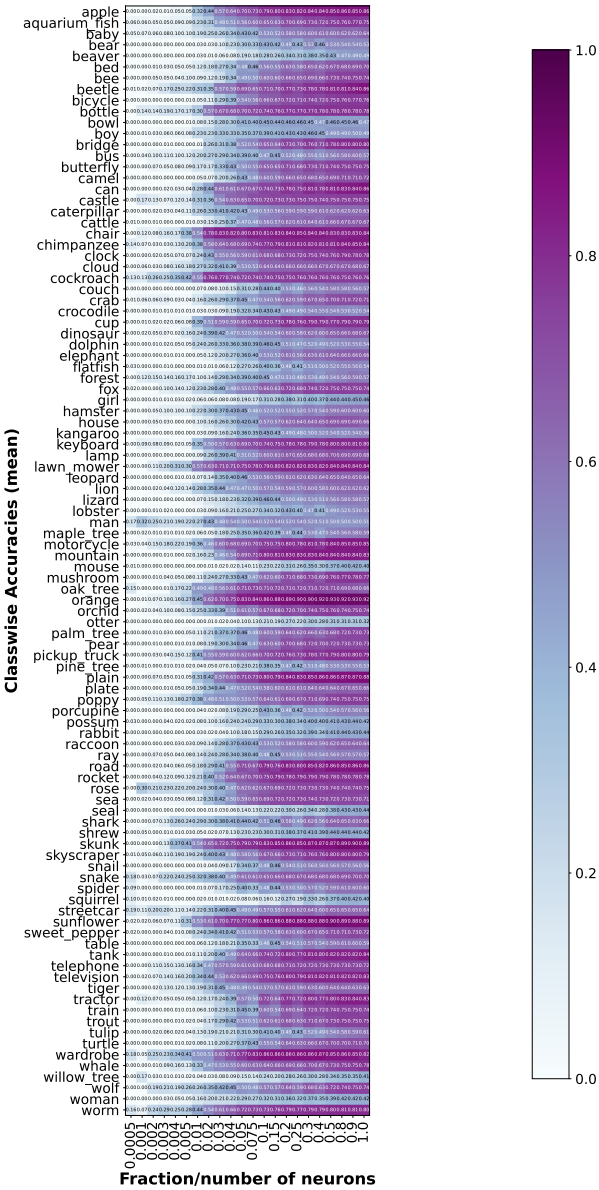
((a)) CIFAR10, Standard



((b)) CIFAR10, Adversarial Training

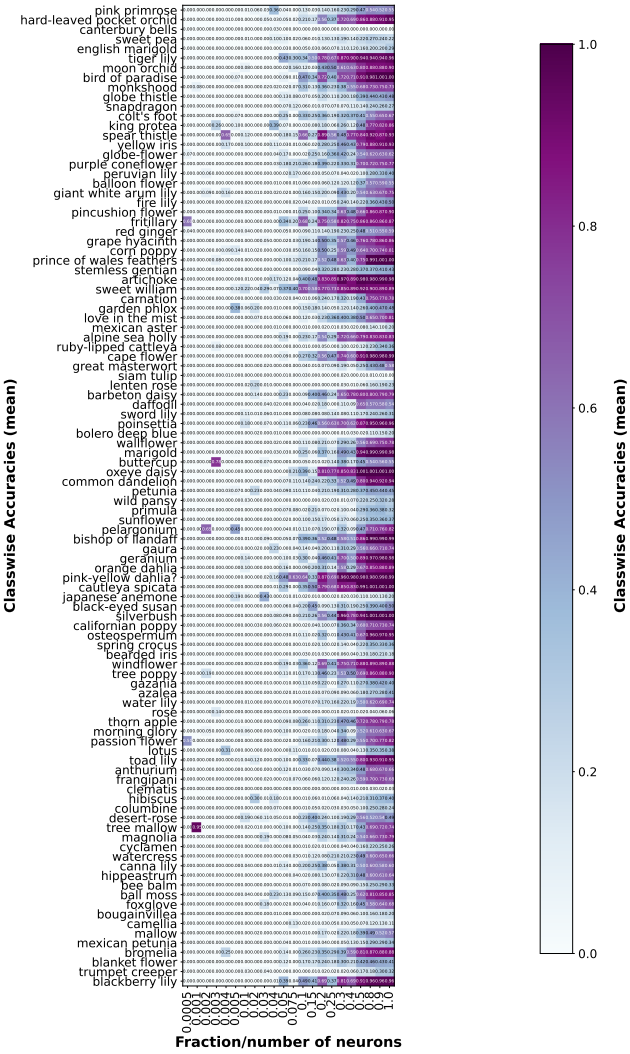


((c)) CIFAR100, Standard

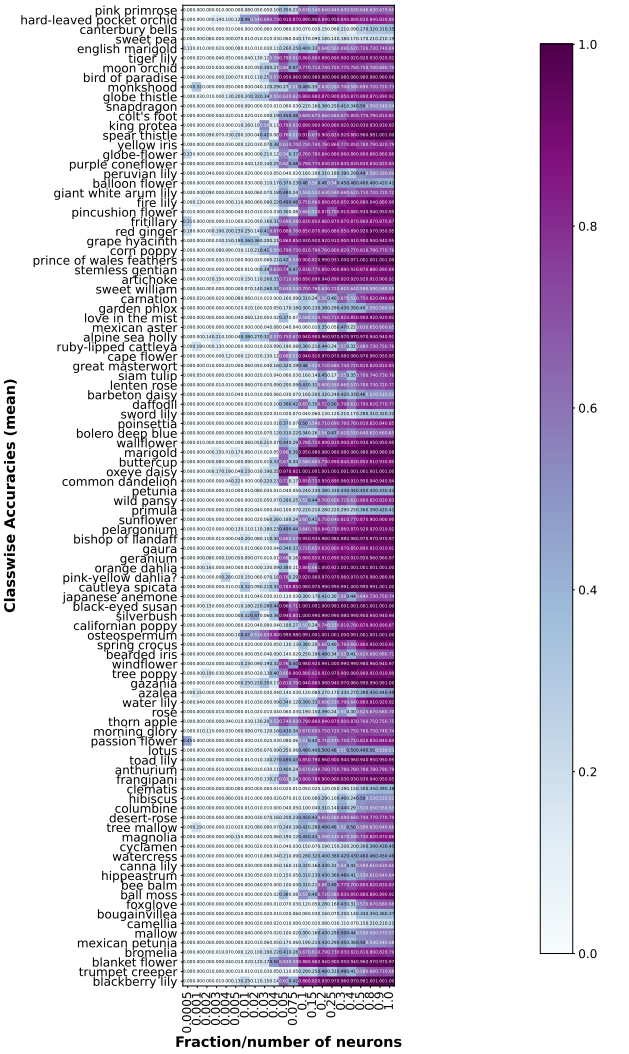


((d)) CIFAR100, Adversarial Training

Figure E.3: **[Error Distributions As A Function of Fraction of Neurons]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.

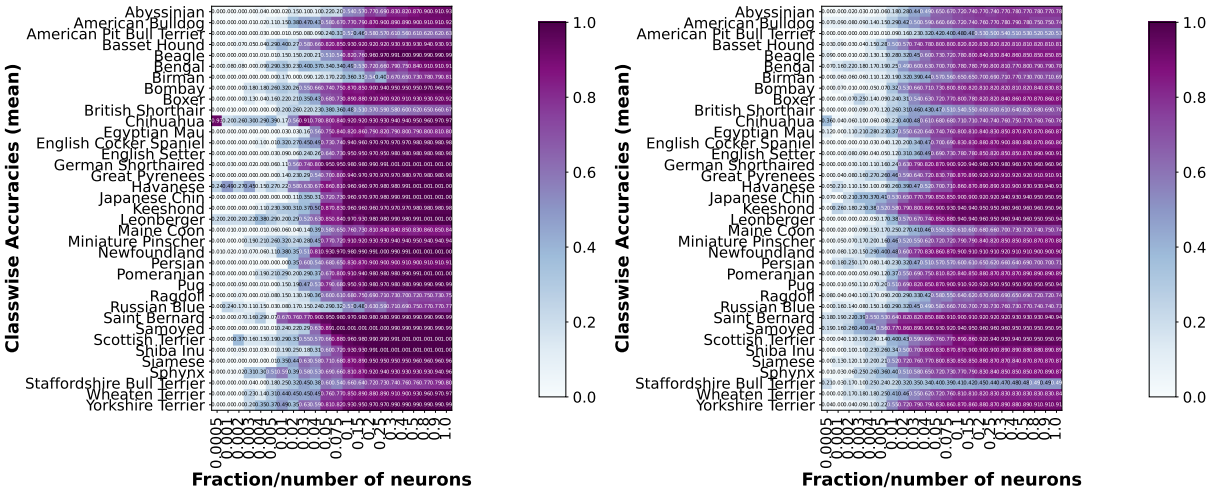


((a)) Flowers, Standard



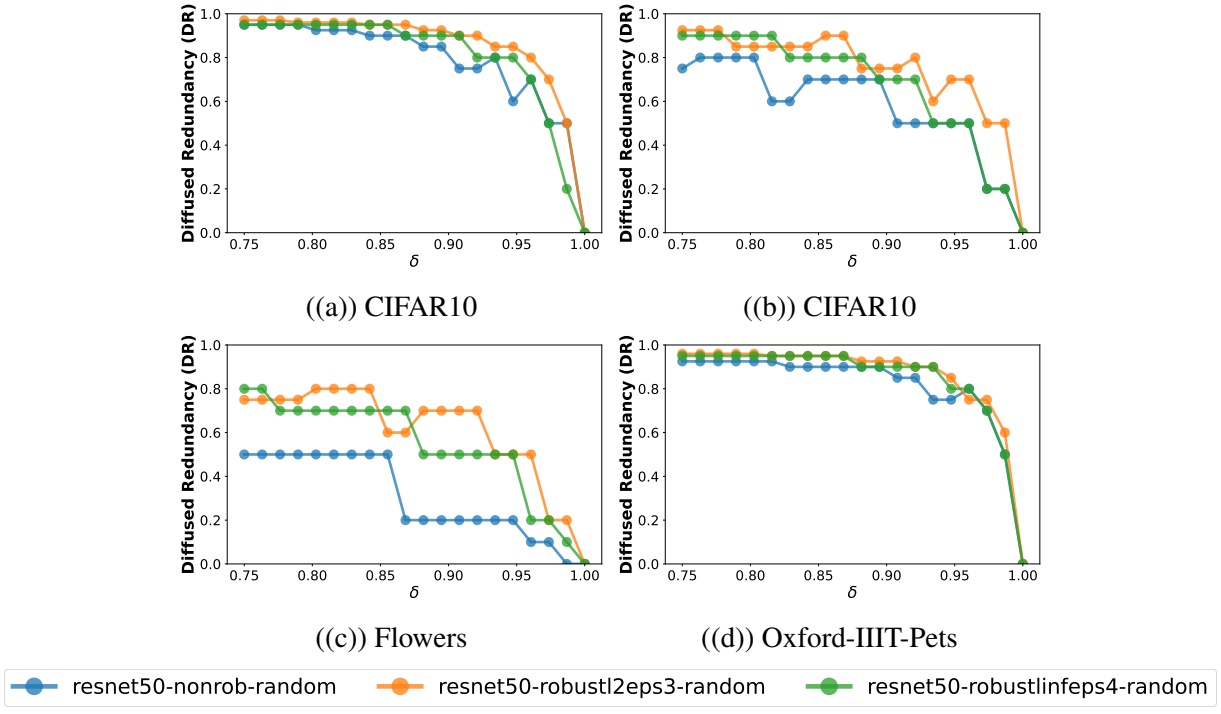
((b)) Flowers, Adversarial Training

Figure E.4: **[Error Distributions As A Function of Fraction of Neurons – continued]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.



((a)) Oxford-IIIT-Pets, Standard ((b)) Oxford-IIIT-Pets, Adversarial Training

Figure E.5: **[Error Distributions As A Function of Fraction of Neurons – continued]** We see that accuracy deteriorates as we drop neurons, however, this drop comes at a larger cost for a few classes and results in near homogenous predictions for the least number of neurons on the left.



((a)) CIFAR10 ((b)) CIFAR10

((c)) Flowers ((d)) Oxford-IIIT-Pets

Figure E.6: **[Results for ℓ_∞ robust model]** We show results for ResNet50 trained with 3 different losses: CrossEntropy (nonrob), Adversarial Training with ℓ_2 threat model (robl2eps3), and with the ℓ_∞ threat model (roblinfeps4). We see that the ℓ_2 threat model shows the most diffused redundancy. All models are trained on ImageNet1k.

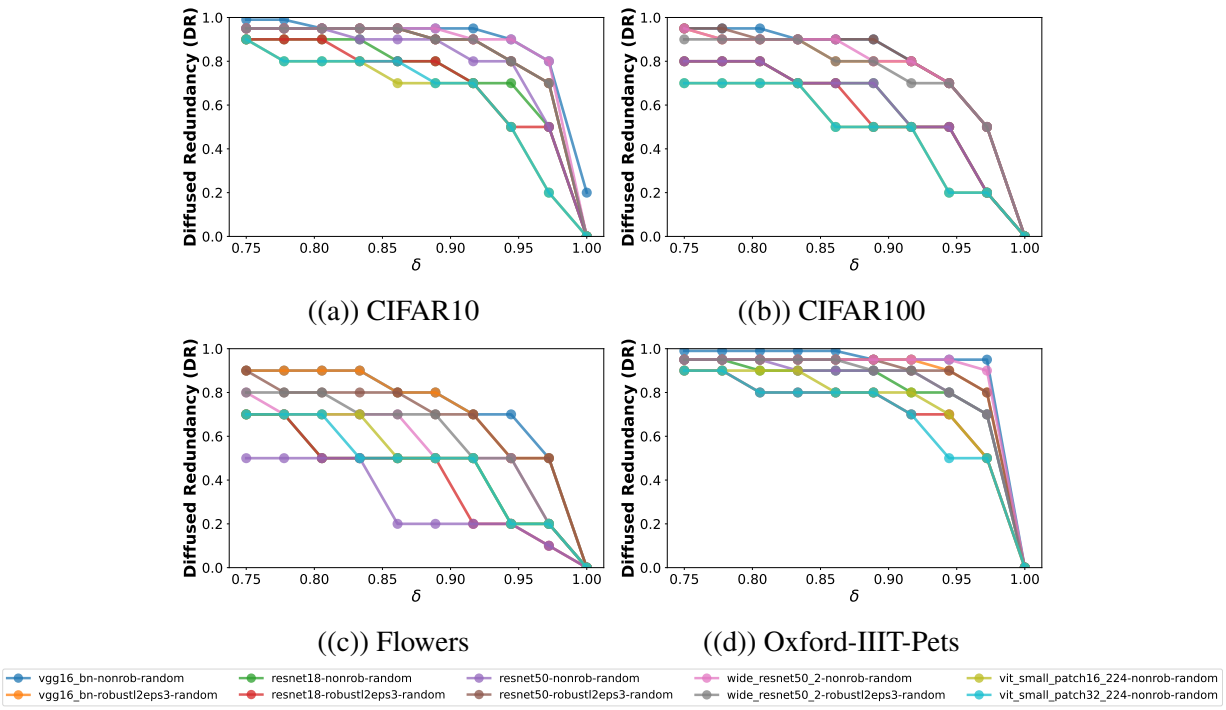


Figure E.7: **[Comparisons Across Architectures For Downstream Task Accuracy]** All models shown here are pre-trained on ImageNet1k. This Figure shows corresponding diffused redundancy values for Figure 6.4 different δ values. We see that diffused redundancy exists across architectures, and the trend observed in Figure 6.1(c)&6.1(a) regarding adversarially trained models also holds here as models curves that are more "inside" are the ones trained with standard loss.

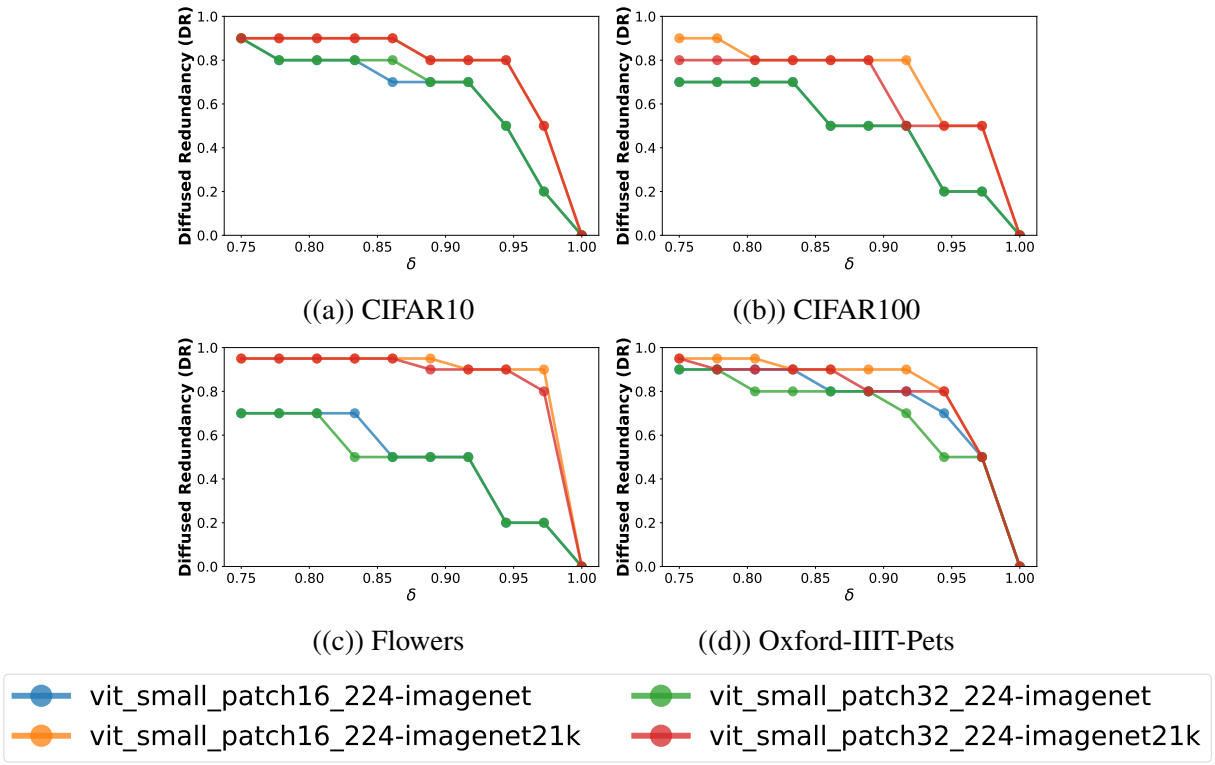


Figure E.8: **[Comparison Across Upstream Datasets]** We see that degree of diffused redundancy depends a great deal on the upstream training dataset, in particular models trained on ImageNet21k exhibit a higher degree of diffused redundanacy, although the differences in the degree of diffused redundanacy are downstream task dependent

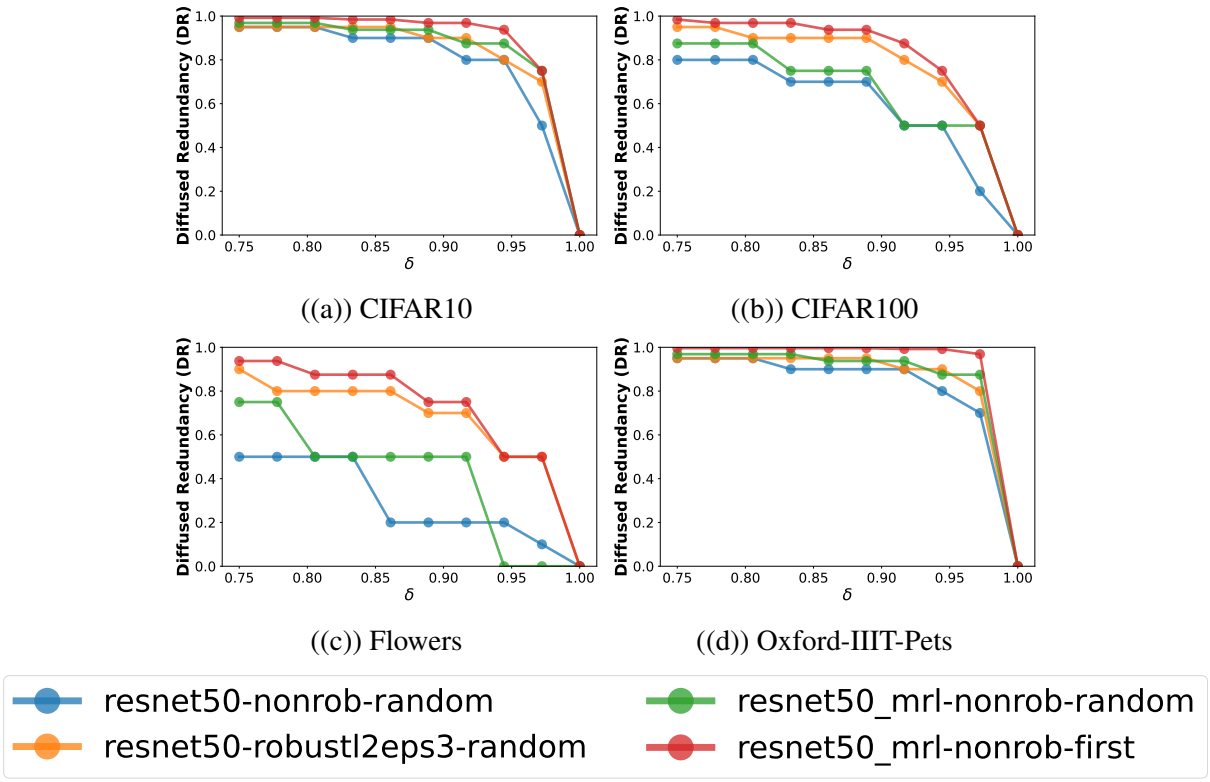
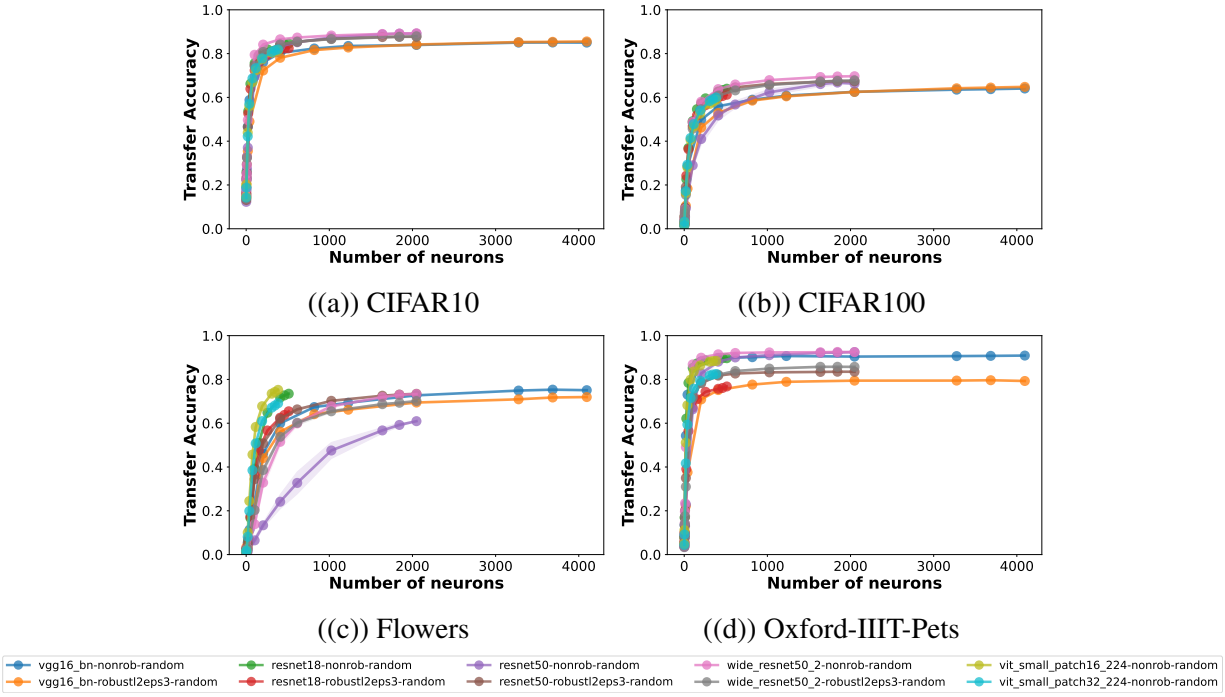


Figure E.9: **[Comparison of Diffused Redundancy in MRL vs other losses]** Here we compare ResNet50 trained using multiple losses including MRL [10]. Red line shows results for part of the representation explicitly optimized in MRL, whereas green line shows results for parts that are picked randomly from the same representation. Even the MRL model shows a significant amount of diffused redundancy despite being explicitly trained to instead have structured redundancy. This figure shows diffused redundancy (DR) for all plots in Figure 6.7.



((a)) CIFAR10 ((b)) CIFAR100

((c)) Flowers ((d)) Oxford-IIIT-Pets

Figure E.10: **[Comparisons Across Architectures For Downstream Task Accuracy]** This shows the same plots as Figure 6.4, except showing absolute number of neurons on the x-axis

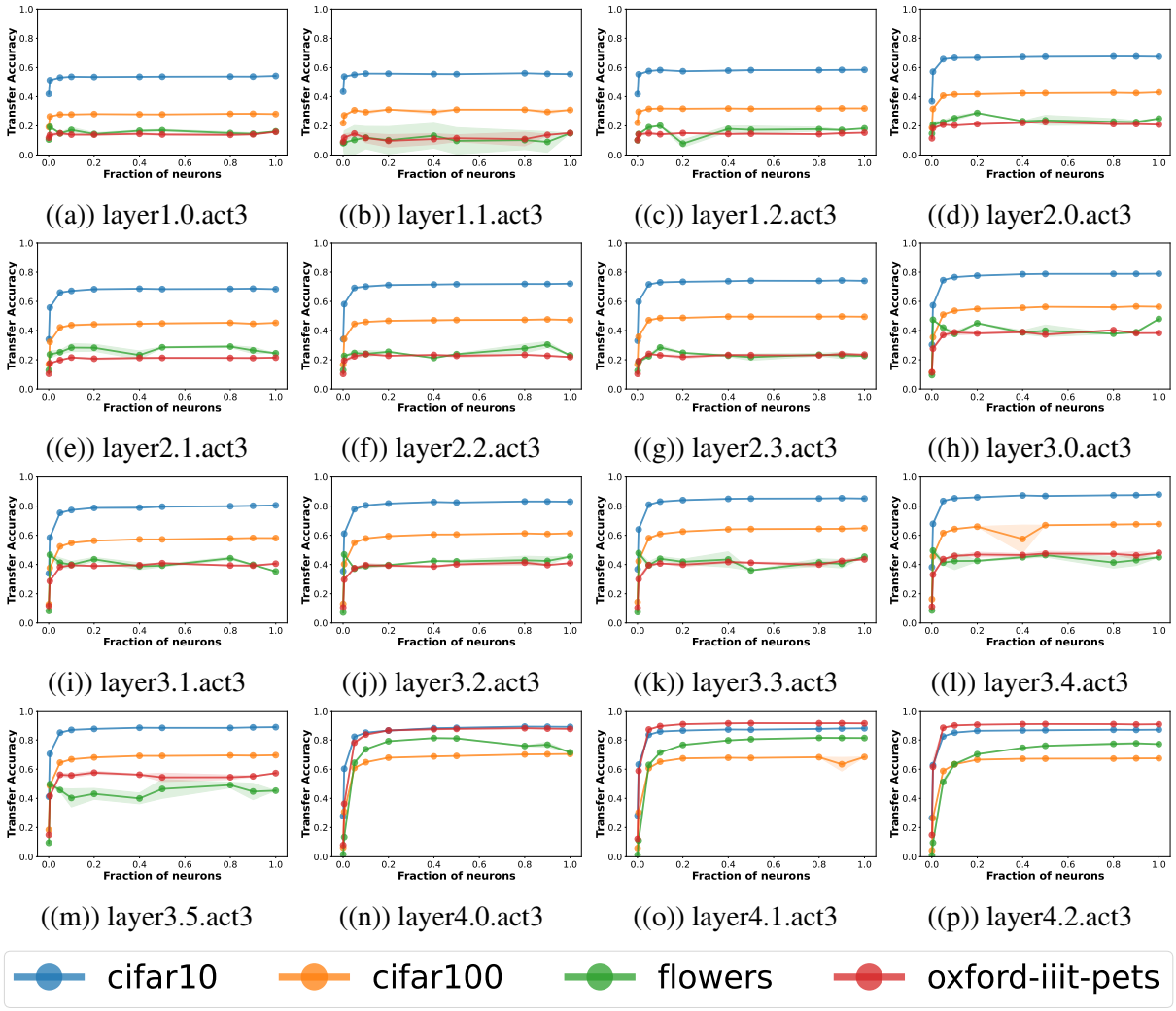
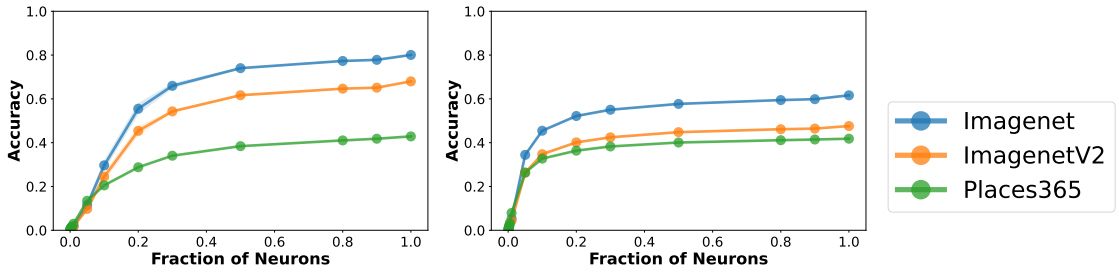
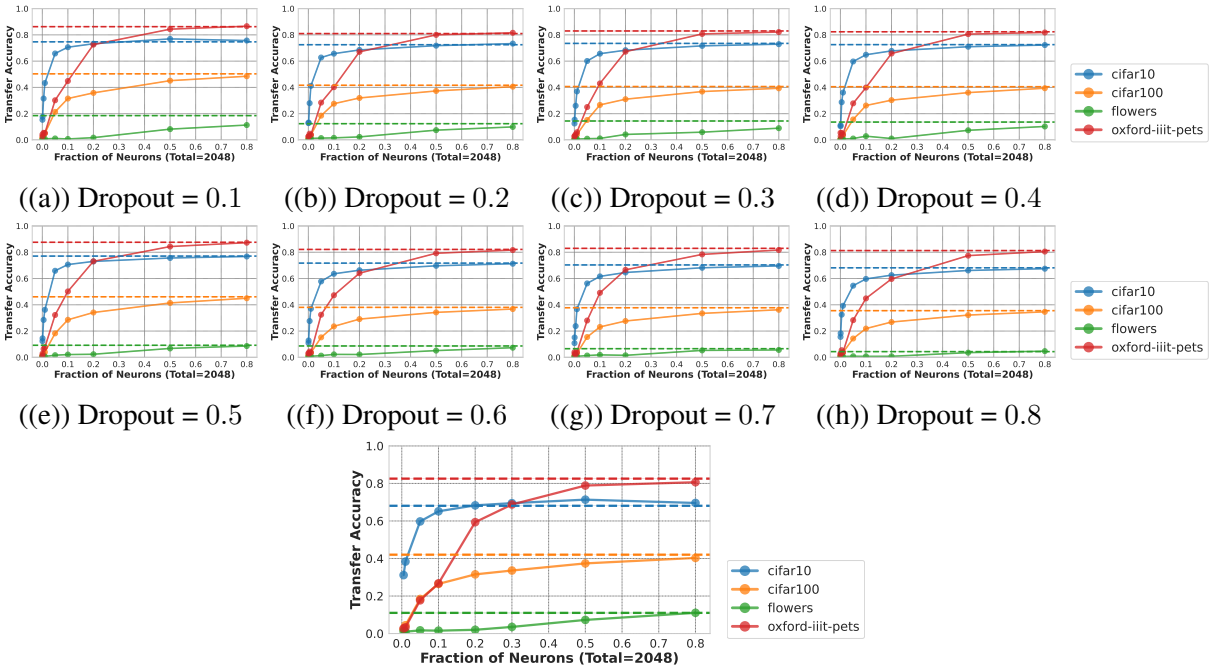


Figure E.11: **[Middle Layers; ResNet50, trained with CrossEntropy loss on ImageNet1k]** We see that as we go deeper in the network, accuracy progressively increases. We see even middle layers exhibit diffused redundancy, and accuracy plateaus very quickly for earlier layers. `layerX.Y.act3` refers to the Yth residual connection in the Xth ResNet block and act3 indicates that we're taking the value after the activation (ReLU) has been applied.



((a)) ResNet50 nonrobust ((b)) ResNet50 robust $\ell_2\epsilon = 3$

Figure E.12: **[Performance on ImageNet1k, ImageNetV2, and Places365]** We check for the performance of randomly chosen subsets of neurons on harder tasks like ImageNet1k, ImageNetV2, and Places365. We find that diffused redundancy holds for all these harder tasks as well. Additionally, we see that randomly dropping neurons still preserves the accuracy gap between ImageNet1k and ImageNetV2.

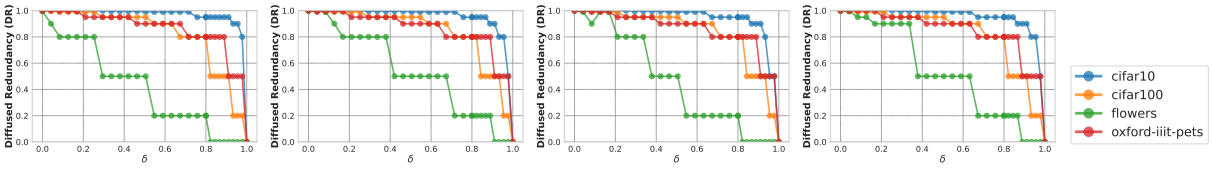


((a)) Dropout = 0.1 ((b)) Dropout = 0.2 ((c)) Dropout = 0.3 ((d)) Dropout = 0.4

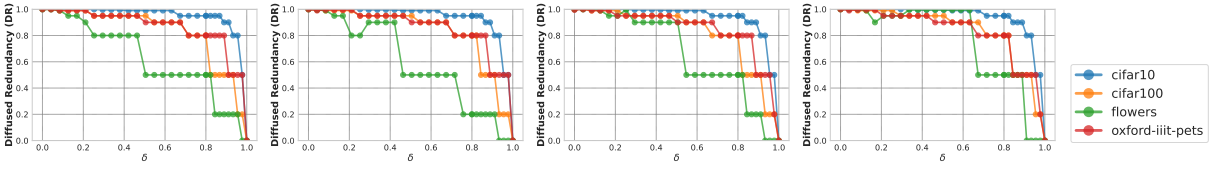
((e)) Dropout = 0.5 ((f)) Dropout = 0.6 ((g)) Dropout = 0.7 ((h)) Dropout = 0.8

((i)) DeCov Regularization (0.0001)

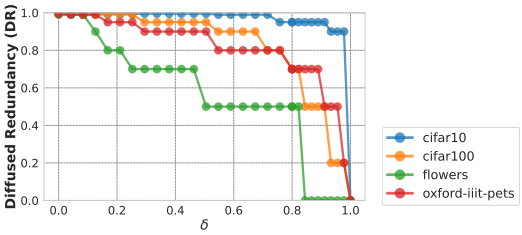
Figure E.13: **[Dropout and DeCov regularizer's effect on Diffused Redundancy]**



((a)) Dropout = 0.1 ((b)) Dropout = 0.2 ((c)) Dropout = 0.3 ((d)) Dropout = 0.4



((e)) Dropout = 0.5 ((f)) Dropout = 0.6 ((g)) Dropout = 0.7 ((h)) Dropout = 0.8



((i)) DeCov Regularization (0.0001)

Figure E.14: **[Dropout and DeCov regularizer's effect on Diffused Redundancy]** Same results as Figure E.13, but showing DR estimates (Eq 6.1). Lines that are more towards the right (*i.e.*more on the "outside") mean they exhibit more diffused redundancy.

Bibliography

[1] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.

[2] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[3] Henriette Cramer, Jenn Wortman Vaughan, Ken Holstein, Hanna Wallach, Jean Garcia-Gathright, Hal Daumé III, Miroslav Dudík, and Sravana Reddy. Challenges of incorporating algorithmic fairness into industry practice. *FAT* Tutorial*, 2019.

[4] Kate Crawford and Trevor Paglen. Excavating ai: The politics of images in machine learning training sets. 2019.

[5] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. pages 77–91, 2018.

[6] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? In *Advances in Neural Information Processing Systems*, 2020.

[7] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[8] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[9] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019.

[10] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. Matryoshka representations for adaptive deployment. *arXiv preprint arXiv:2205.13147*, 2022.

[11] Corrado Gini. Measurement of inequality of incomes. *The economic journal*, 31(121):124–126, 1921.

[12] Valeriia Cherepanova, Vedant Nanda, Micah Goldblum, John P Dickerson, and Tom Goldstein. Technical challenges for training fair neural networks. *arXiv preprint arXiv:2102.06764*, 2021.

[13] Vedant Nanda, Samuel Dooley, Sahil Singla, Soheil Feizi, and John P Dickerson. Fairness through robustness: Investigating robustness disparity in deep learning. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 466–477, 2021.

[14] Vedant Nanda, Ayan Majumdar, Camila Kolling, John P Dickerson, Krishna P Gummadi, Bradley C Love, and Adrian Weller. Do invariances in deep neural networks align with human perception? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 9277–9285, 2023.

[15] Vedant Nanda, Till Speicher, Camila Kolling, John P Dickerson, Krishna Gummadi, and Adrian Weller. Measuring representational robustness of neural networks through shared invariances. In *International Conference on Machine Learning*, pages 16368–16382. PMLR, 2022.

[16] Vedant Nanda, Till Speicher, John P Dickerson, Soheil Feizi, Krishna P Gummadi, and Adrian Weller. Diffused redundancy in pre-trained representations. *arXiv preprint arXiv:2306.00183*, 2023.

[17] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California Law Review*, 104:671–732, 2016.

[18] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: Testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 498–510, New York, NY, USA, 2017.

[19] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[20] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2564–2572. PMLR, 10–15 Jul 2018.

[21] Zachary C Lipton, Alexandra Chouldechova, and Julian McAuley. Does mitigating ml's impact disparity require treatment disparity? In *NIPS*, pages 8136–8146, 2018.

[22] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. `http://www.fairmlbook.org`.

[23] Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Nunes Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna P. Gummadi, Patrick Loiseau, and Alan Mislove. Potential for discrimination in online targeted advertising. In *FAT**, volume 81, pages 5–19, 2018.

[24] Muhammad Ali, Piotr Sapiezynski, Miranda Bogen, Aleksandra Korolova, Alan Mislove, and Aaron Rieke. Discrimination through optimization: How facebook's ad delivery can lead to biased outcomes. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

[25] Filipe N. Ribeiro, Koustuv Saha, Mahmoudreza Babaei, Lucas Henrique, Johnnatan Messias, Fabricio Benevenuto, Oana Goga, Krishna P. Gummadi, and Elissa M. Redmiles. On microtargeting socially divisive ads: A case study of russia-linked ad campaigns on facebook. In *FAT**, page 140–149, New York, NY, USA, 2019.

[26] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*, page 405–414, 2018.

[27] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *KDD*, 2018.

[28] Amir E. Khandani, Adlar J. Kim, and Andrew W. Lo. Consumer credit-risk models via machine-learning algorithms. *Journal of Banking & Finance*, 34(11):2767–2787, 2010.

[29] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, 2017.

[30] Candice Schumann, Jeffrey S. Foster, Nicholas Mattei, and John P. Dickerson. We need fairness and explainability in algorithmic hiring. In *AAMASConf*, page 1716–1720, 2020.

[31] Agostina J. Larrazabal, Nicolás Nieto, Victoria Peterson, Diego H. Milone, and Enzo Ferrante. Gender imbalance in medical imaging datasets produces biased classifiers for computer-aided diagnosis. *PNAS*, 117(23):12592–12594, 2020.

[32] Laleh Seyyed-Kalantari, Guanxiong Liu, Matthew McDermott, Irene Y. Chen, and Marzyeh Ghassemi. Chexclusion: Fairness gaps in deep chest x-ray classifiers. *arXiv preprint arXiv:2003.00827*, 2020.

[33] P J. Phillips Patrick J. Grother, George W. Quinn. Report on the evaluation of 2d still-image face recognition algorithms. *NIST Interagency/Internal Report (NISTIR)*, 2010.

[34] Mei L. Ngan and Patrick J. Grother. Face recognition vendor test (frvt) - performance of automated gender classification algorithms. *NIST Interagency/Internal Report (NISTIR)*, 2015.

[35] Reuben Binns. Fairness in machine learning: Lessons from political philosophy. *Proceedings of Machine Learning Research*, 81:1–11, 2017.

[36] Derek Leben. Normative principles for evaluating fairness in machine learning. In *AIES*, pages 86–92, 2020.

[37] Tatsunori Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. In *ICML*, volume 80, pages 1929–1938, 2018.

[38] Natalia Martinez, Martin Bertran, and Guillermo Sapiro. Minimax pareto fairness: A multi objective perspective. In *ICML*, volume 119, pages 6755–6764, 2020.

[39] Hoda Heidari, Vedant Nanda, and Krishna Gummadi. On the long-term impact of algorithmic decision policies: Effort unfairness and feature segregation through social learning. In *ICML*, volume 97, pages 2692–2701, 2019.

[40] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *KDD*, pages 259–268, 2015.

[41] Hee Jung Ryu, Hartwig Adam, and Margaret Mitchell. Inclusivefacenet: Improving face attribute detection with race and gender diversity. *arXiv preprint arXiv:1712.00193*, 2018.

[42] Novi Quadrianto, Viktoriia Sharmanska, and Oliver Thomas. Discovering fair representations in the data domain. In *CVPR*, pages 8227–8236. Computer Vision Foundation / IEEE, 2019.

[43] Mei Wang and Weihong Deng. Mitigating bias in face recognition using skewness-aware reinforcement learning. In *CVPR*, pages 9322–9331, 2020.

[44] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: Mechanisms for fair classification. In *AISTATS*, volume 54, pages 962–970. PMLR, 2017.

[45] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment & disparate impact. *WWW*, Apr 2017.

[46] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: A flexible approach for fair classification. *JMLR*, 20(75):1–42, 2019.

[47] Michele Donini, Luca Oneto, Shai Ben-David, John Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. In *NIPS*, page 2796–2806, 2018.

[48] Naman Goel, Mohammad Yaghini, and Boi Faltings. Non-discriminatory machine learning through convex fairness criteria. *AAAI*, 32(1), 2018.

[49] Manisha Padala and Sujit Gujar. Fnnc: Achieving fairness through neural networks. In *IJCAI-20*, pages 2277–2283, 7 2020.

[50] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *ICML*, volume 80, pages 60–69, 2018.

[51] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. Convergent algorithms for (relaxed) minimax fairness. *arXiv preprint arXiv:2011.03108*, 2020.

[52] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H. Chi. Fairness without demographics through adversarially reweighted learning. *arXiv preprint arXiv:2006.13114*, 2020.

[53] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS*, page 214–226, 2012.

[54] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *ICML*, number 3, pages 325–333, 17–19 Jun 2013.

[55] Harrison Edwards and Amos J. Storkey. Censoring representations with an adversary. In *ICLR*, 2016.

[56] David Madras, Elliot Creager, Toniann Pitassi, and Richard S. Zemel. Learning adversarially fair and transferable representations. In *ICML*, volume 80, pages 3381–3390, 2018.

[57] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*, 2017.

[58] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *ICCV*, pages 5310–5319, 2019.

[59] Moritz Hardt, Eric Price, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. In *NIPS*, volume 29, pages 3315–3323, 2016.

[60] Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation, 2020.

[61] Yash Savani, Colin White, and Naveen Sundar Govindarajulu. Post-hoc methods for debiasing neural networks. *arXiv preprint arXiv:2006.08564*, 2020.

[62] Aythami Morales, Julian Fierrez, Ruben Vera-Rodriguez, and Ruben Tolosana. Sensitivenets: Learning agnostic representations with application to face images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[63] Debjani Saha, Candice Schumann, Duncan C. McElfresh, John P. Dickerson, Michelle L Mazurek, and Michael Carl Tschantz. Measuring non-expert comprehension of machine learning fairness metrics. In *ICML*, 2020.

[64] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Allison Woodruff, Christine Luu, Pierre Kreitmann, Jonathan Bischof, and Ed H. Chi. Putting fairness principles into practice: Challenges, metrics, and improvements. In *AIES*, page 453–459, New York, NY, USA, 2019.

[65] Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé, Miro Dudik, and Hanna Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In *CHI*, page 1–16, New York, NY, USA, 2019.

[66] Michael A Madaio, Luke Stark, Jennifer Wortman Vaughan, and Hanna Wallach. Co-designing checklists to understand organizational challenges and opportunities around fairness in ai. In *CHI*, pages 1–14, 2020.

[67] Melissa McCradden, Mjaye Mazwi, Shalmali Joshi, and James A. Anderson. When your only tool is a hammer: Ethical limitations of algorithmic fairness solutions in healthcare machine learning. In *AIES*, page 109, New York, NY, USA, 2020.

[68] Inioluwa Deborah Raji, Timnit Gebru, Margaret Mitchell, Joy Buolamwini, Joonseok Lee, and Emily Denton. Saving face: Investigating the ethical concerns of facial recognition auditing. In *AIES*, page 145–151, New York, NY, USA, 2020.

[69] Mark Andrejevic and Neil Selwyn. Facial recognition technology in schools: critical questions and concerns. *Learning, Media and Technology*, 45(2):115–128, 2020.

[70] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, pages 5265–5274, 2018.

[71] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017.

[72] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jayne Seekins, David A. Mong, Safwan S. Halabi, Jesse K. Sandberg, Ricky Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *AAAI*, 33:590–597, Jul. 2019.

[73] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[75] T. Calders, F. Kamiran, and M. Pechenizkiy. Building classifiers with independency constraints. In *ICDM Workshops*, pages 13–18, 2009.

[76] John Rawls. *A Theory of Justice*. Harvard University Press, 1971.

[77] Chongjie Zhang and Julie A Shah. Fairness in multi-agent sequential decision-making. In *NIPS*, volume 27, pages 2636–2644, 2014.

[78] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. Agnostic federated learning. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4615–4625, 2019.

[79] Hongyan Chang, Ta Duy Nguyen, Sasi Kumar Murakonda, Ehsan Kazemi, and Reza Shokri. On adversarial bias and the robustness of fair machine learning. *arXiv preprint arXiv:2006.08669*, 2020.

[80] Candice Schumann, Samsara N. Counts, Jeffrey S. Foster, and John P. Dickerson. The diverse cohort selection problem. In *AAMASConf*, page 601–609, 2019.

[81] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data*, 5(2):153–163, 2017.

[82] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. pages 259–268, 2015.

[83] Derek Leben. Normative principles for evaluating fairness in machine learning. pages 86–92, 2020.

[84] Reuben Binns. Fairness in machine learning: Lessons from political philosophy. *Proceedings of Machine Learning Research*, 81:1–11, 2017.

[85] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez-Rodriguez, and Krishna P. Gummadi. Fairness constraints: A flexible approach for fair classification. *Journal of Machine Learning Research*, 20(75):1–42, 2019.

[86] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, Krishna P. Gummadi, and Adrian Weller. From parity to preference-based notions of fairness in classification. 2017.

[87] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. pages 325–333, 2013.

[88] Hoda Heidari, Vedant Nanda, and Krishna P. Gummadi. On the long-term impact of algorithmic decision policies: Effort unfairness and feature segregation through social learning. 2019.

[89] Hoda Heidari and Andreas Krause. Preventing disparate treatment in sequential decision making. 2018.

[90] Lydia T Liu, Sarah Dean, Esther Rolf, Max Simchowitz, and Moritz Hardt. Delayed impact of fair machine learning. 2018.

[91] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.

[92] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. 2015.

[93] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[94] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (S&P)*, pages 39–57, 2017.

[95] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. pages 2574–2582, 2016.

[96] Cynthia Dwork and Christina Ilvento. Fairness under composition. 2018.

[97] Nina Grgić-Hlača, Muhammad Bilal Zafar, Krishna P. Gummadi, and Adrian Weller. Beyond distributive fairness in algorithmic decision making: Feature selection for procedurally fair learning. 2018.

[98] Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. pages 2412–2420, 2019.

[99] Christina Wadsworth, Francesca Vera, and Chris Piech. Achieving fairness through adversarial learning: an application to recidivism prediction. *CoRR*, abs/1807.00199, 2018.

[100] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Advances in Neural Information Processing Systems 30*, NIPS'17, pages 3992–4001. 2017.

[101] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems 30*, pages 4066–4076. 2017.

[102] Niki Kilbertus, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. In *NeurIPS*, pages 656–666, 2017.

[103] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5680–5689. Curran Associates, Inc., 2017.

[104] Zeyu Wang, Klint Qinami, Yannis Karakozis, Kyle Genova, P. Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8916–8925, 2020.

[105] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. 2019.

[106] Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. pages 11292–11303. 2019.

[107] Sahil Singla and Soheil Feizi. Second-order provable defenses against adversarial attacks. 2020.

[108] David Solans, Battista Biggio, and Carlos Castillo. Poisoning attacks on algorithmic fairness, 2020.

[109] Ninareh Mehrabi, Muhammad Naveed, Fred Morstatter, and Aram Galstyan. Exacerbating algorithmic bias through fairness attacks, 2020.

[110] Fereshte Khani and Percy Liang. Noise induces loss discrepancy across groups for linear regression, 2019.

[111] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, June 1999.

[112] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. page 144–152, 1992.

[113] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009.

[114] Eran Eidinger, Roee Enbar, and Tal Hassner. Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170–2179, 2014.

[115] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[116] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8026–8037. 2019.

[117] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014.

[118] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 2015.

[119] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016.

[120] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size. *CoRR*, abs/1602.07360, 2016.

[121] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *IEEE CVPR*, 2017.

[122] Alex Hanna, Emily Denton, Andrew Smart, and Jamila Smith-Loud. Towards a critical race methodology in algorithmic fairness. pages 501–512, 2020.

[123] Jenelle Feather, Alex Durango, Ray Gonzalez, and Josh McDermott. Metamers of neural networks reveal divergence from human perceptual systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[124] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them, 2014.

[125] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[126] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[127] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019.

[128] Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, 2020.

[129] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[130] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

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

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

[133] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.

[134] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, 2019.

[135] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Adversarial robustness as a prior for learned representations, 2019.

[136] Simran Kaur, Jeremy M. Cohen, and Zachary C. Lipton. Are perceptually-aligned gradients a general property of robust classifiers? 2019.

[137] Shibani Santurkar, Andrew Ilyas, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Image synthesis with a single (robust) classifier. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[138] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[139] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in.

[140] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

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

[142] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *arXiv preprint arXiv:2006.07589*, 2020.

[143] Mariya Toneva and Leila Wehbe. *Interpreting and Improving Natural-Language Processing (in Machines) with Natural Language-Processing (in the Brain)*. 2019.

[144] Gustav Theodor Fechner. Elements of psychophysics, 1860. 1948.

[145] Brett D. Roads and Bradley C. Love. Enriching imagenet with human similarity judgments and psychological embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 3547–3557. Computer Vision Foundation / IEEE, 2021.

[146] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks, 2015.

[147] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015.

[148] Tal Golan, Prashant C. Raju, and Nikolaus Kriegeskorte. Controversial stimuli: Pitting neural networks against each other as models of human cognition. *Proceedings of the National Academy of Sciences*, 117(47):29330–29337, 2020.

[149] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[150] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

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

[152] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

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

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

[155] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 699–708, 2020.

[156] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[157] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519, 2017.

[158] Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[159] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[160] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

[161] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[162] Aarre Laakso and Garrison Cottrell. Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76, 2000.

[163] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6078–6087, 2017.

[164] Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 5732–5741, 2018.

[165] Liwei Wang, Lunjia Hu, Jiayuan Gu, Yue Wu, Zhiqiang Hu, Kun He, and John Hopcroft. Towards understanding learning representations: To what extent do different neural networks learn the same representation. *arXiv preprint arXiv:1810.11750*, 2018.

[166] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations?, 2016.

[167] Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity with statistical testing. *arXiv preprint arXiv:2108.01661*, 2021.

[168] Seyed Ali Amirshahi, Marius Pedersen, and Stella X Yu. Image quality assessment by comparing cnn features between images. *Journal of Imaging Science and Technology*, 60(6):60410–1, 2016.

[169] Fei Gao, Yi Wang, Panpeng Li, Min Tan, Jun Yu, and Yani Zhu. Deepsim: Deep similarity for image quality assessment. *Neurocomputing*, 257:104–114, 2017.

[170] Brian J Stankiewicz and John E Hummel. Categorical relations in shape perception. *Spatial vision*, 10(3):201–236, 1996.

[171] Olivia Guest and Bradley C Love. What the success of brain imaging implies about the neural code. *Elife*, 6:e21397, 2017.

[172] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer Berlin Heidelberg, 2013.

[173] Robert Geirhos, Carlos R. M. Temme, Jonas Rauber, Heiko H. Schütt, Matthias Bethge, and Felix A. Wichmann. Generalisation in humans and deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[174] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[175] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR, 2019.

[176] Alhussein Fawzi and Pascal Frossard. Manitest: Are classifiers really invariant? *CoRR*, abs/1507.06535, 2015.

[177] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020.

[178] Merriam-Webster. Robust. In *Merriam-Webster.com dictionary*. 2022.

[179] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

[180] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34, 2021.

[181] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*, 2020.

[182] Judea Pearl. *Causality*. Cambridge University Press, 2 edition, 2009.

[183] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

[184] Muhammad Bilal Zafar, Michele Donini, Dylan Slack, Cedric Archambeau, Sanjiv Das, and Krishnaram Kenthapadi. On the lack of robust interpretability of neural text classifiers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3730–3740, Online, August 2021. Association for Computational Linguistics.

[185] Jenelle Feather, Alex Durango, Ray Gonzalez, and Josh McDermott. Metamers of neural networks reveal divergence from human perceptual systems. *Advances in Neural Information Processing Systems*, 32, 2019.

[186] Vedant Nanda, Ayan Majumdar, Camila Kolling, John P. Dickerson, Krishna P. Gummadi, Bradley C. Love, and Adrian Weller. Do invariances in deep neural networks align with human perception? *CoRR*, abs/2111.14726, 2022.

[187] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[188] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29:658–666, 2016.

[189] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016.

[190] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. Revisiting model stitching to compare neural representations. *Advances in Neural Information Processing Systems*, 34:225–236, 2021.

[191] Irina Higgins, David Amos, David Pfau, Sébastien Racanière, Loïc Matthey, Danilo J. Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. *CoRR*, abs/1812.02230, 2018.

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

[193] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[194] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[195] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.

[196] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pages 11205–11216. PMLR, 2021.

[197] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. abs/2110.00476, 2021.

[198] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021.

[199] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020.

[200] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[201] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[202] Fred Attneave. Some informational aspects of visual perception. *Psychological review*, 61(3):183, 1954.

[203] Yoshio Izui and Alex Pentland. Analysis of neural networks with redundancy. *Neural Computation*, 2(2):226–238, 1990.

[204] Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4908–4926, Online, November 2020. Association for Computational Linguistics.

[205] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas S. Huang. Slimmable neural networks. abs/1812.08928, 2018.

[206] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1803–1811, 2019.

[207] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

[208] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[209] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[210] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[211] Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

[212] Asriel Levin, Todd Leen, and John Moody. Fast pruning using principal components. *Advances in neural information processing systems*, 6, 1993.

[213] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 30, 2017.

[214] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[215] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[216] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[217] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization, 2015.

[218] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

[219] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013.

[220] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[221] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*, 2018.

[222] Alessandro Achille and Stefano Soatto. On the emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 125:126–127, 2017.

[223] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[224] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[225] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[226] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[227] Michael Cogswell, Faruk Ahmed, Ross Girshick, Larry Zitnick, and Dhruv Batra. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*, 2015.

[228] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

[230] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rogriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. In *Artificial intelligence and statistics*, pages 962–970. PMLR, 2017.

[231] Songül Tolan, Marius Miron, Emilia Gómez, and Carlos Castillo. Why machine learning may lead to unfairness: Evidence from risk assessment for juvenile justice in catalonia. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law*, pages 83–92, 2019.

[232] Michael Veale, Max Van Kleek, and Reuben Binns. Fairness and accountability design needs for algorithmic support in high-stakes public sector decision-making. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–14, 2018.

[233] Fernando G De Maio. Income inequality measures. *Journal of Epidemiology & Community Health*, 61(10):849–852, 2007.

[234] Robert R Schutz. On the measurement of income inequality. *The American Economic Review*, 41(1):107–122, 1951.

[235] Till Speicher, Hoda Heidari, Nina Grgic-Hlaca, Krishna P Gummadi, Adish Singla, Adrian Weller, and Muhammad Bilal Zafar. A unified approach to quantifying algorithmic unfairness: Measuring individual &group unfairness via inequality indices. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2239–2248, 2018.

[236] Barbara S Lawrence. Perspective—the black box of organizational demography. *Organization science*, 8(1):1–22, 1997.

[237] Han Xu, Xiaorui Liu, Yaxin Li, Anil Jain, and Jiliang Tang. To be robust or to be fair: Towards fairness in adversarial training. In *International conference on machine learning*, pages 11492–11501. PMLR, 2021.

[238] Zeming Wei, Yifei Wang, Yiwen Guo, and Yisen Wang. Cfa: Class-wise calibrated fair adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8193–8201, 2023.

[239] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics, 2019.

[240] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

[241] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[242] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[243] Federico Bianchi, Pratyusha Kalluri, Esin Durmus, Faisal Ladhak, Myra Cheng, Debora Nozza, Tatsunori Hashimoto, Dan Jurafsky, James Zou, and Aylin Caliskan. Easily accessible text-to-image generation amplifies demographic stereotypes at large scale. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency*, pages 1493–1504, 2023.

[244] Abubakar Abid, Maheen Farooqi, and James Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

[245] Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pages 10118–10129. PMLR, 2021.

[246] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[247] William Falcon. Pytorch lightning. `https://github.com/PyTorchLightning/pytorch-lightning`, 2019.

[248] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[249] Logan Engstrom, Andrew Ilyas, Hadi Salman, Shibani Santurkar, and Dimitris Tsipras. Robustness (python library), 2019.

[250] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[251] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! *arXiv preprint arXiv:1904.12843*, 2019.