A Study into the Bias of SOTA Semi and Self-Supervised Models

Albert Tsao
Cornell University
Ithaca, NY
awt46@cornell.edu

Dhruv Sreenivas
Cornell University
Ithaca, NY
ds844@cornell.edu

Abstract

This report is a slight extension of work done by Wang et. al. [14][1], who suggested bias mitigation methods for supervised learning recongition tasks. We see if the methods developed can also be used in self-supervised and semi-supervised training regimes. We explore the bias encoded in a state-of-the-art semi-supervised computer vision algorithm, FixMatch [13], and a state-of-the-art self-supervised algorithm, SimCLR [3], and employ the same bias reduction techniques from Wang et. al. to see how much better they perform, particularly on the CIFAR-10 dataset [9](and its variants proposed by Wang et. al.) and the CelebA dataset [10].

1. Motivation

Deep computer vision systems have seen major success in recent years, in tasks from image classification and object recognition to 3D reconstruction. However, said systems have been shown to learn seemingly unrelated input features when learning to do said tasks, especially in those of classification, leading to inherent bias in the resulting model. Said bias can be detrimental to society, and there have been many recent cases of large models used by Google and Facebook being overtly racist or sexist due to these encoded feature biases. Recent work [14] has focused on bias mitigation for large, state-of-the-art computer vision models, such as ResNets [7], and recently proposed methods have been shown to work well on standard supervised learning tasks, where all the images and labels were available to train with. In many different real-world scenarios, getting a large number of image-label pairs is costly and sometimes intractable, and recent research has been focused on few-shot learning in particular for that reason alone. There has been a lot of recent research on self-supervised, contrastive and semi-supervised learning methods, which use significantly less labels than existing supervised learning regimes to do the same tasks with surprising effectiveness. The natural question that arises is: within a self-supervised or semi-supervised learning scheme, does more or less bias get encoded into the learned features of images in a dataset? A natural hypothesis is that less bias gets encoded due to the lack of labels, although one also has to consider that there may be inherent bias encoded due to the dataset itself. Our work focuses on testing this hypothesis to see its value, in a similar way that Wang et. al. [14] did.

2. Related Work

Bias has been a known problem in machine learning models but has only recently has bias in computer vision models been analyzed. Discussed below are a few of the current literature that inspired our project.

Buolamwini & Gebru [2] really laid the groundwork for research in bias in computer vision models. They design a more balanced dataset given images from two different facial analysis benchmarks, both of which individually had a lot of lighter-skinned faces in them (and thus being very unbalanced). Moreover, they showed that performace of then-SOTA facial recognition systems on their new balanced dataset is the worst on darker-skinned females (34.7% error) and the best on light-skinned males (0.8% error). This served as a warning to debias facial recognition algorithms (and general ML algorithms to some degree) before deployment in the real world.

Amini, Alexander et al. [1] attempt to debias facial recognition systems. In particular, they propose a novel algorithm for debiasing a dataset by first learning the latent variables of a class using a DB-VAE. Based on this latent structure, they use a semi-supervised method to learn a debiased classifier by increasing the probability rarer data will be selected for training. Their trained models were tested on the PPB dataset and had lower variance of classification accuracy, which they used as a proxy for bias, compared to non-debiased models.

Serna et. al [12] take a more general route when analyzing bias, looking first at a very general example (deep neural networks on MNIST) and then look toward facial

[1] https://github.com/princetonvisualai/DomainBiasMitigation

recognition systems, developing a new algorithm, Inside-Bias, which detects bias within models by using their learnt representations of images with a small training set (few-shot). However, one downside of their experiments was that they did only work with a dataset of 3 ethnic groups, which is not super representative of the real world.

Zhao et. al [15] show that datasets for tasks associated with multilabel object classification and visual semantic role labeling contain heavy gender bias. With this biased dataset, they also show that the models learned from this data are heavily biased. In order to identify bias, they compute the bias scores of differnet outputs with respect to various demographic vairables for a corpus/dataset. Then, they evaluate the bias amplification of a model. Their algorithm, Reducing Bias Amplification, adds corpus-level constraints to ensure outputs follow a desired distribution.

Last, but not least, Wang et. al. [14], which this work is heavily based off of, developed a benchmark as well as two methods that have proven useful for domain bias mitigation. In particular, one of said methods, *domain independent training*, was very simple and quite effective. Essentially, given N classes and D domains, a classifier is trained for each domain, and at the end, the predicted class would be the one that has the maximum possible summed or averaged probability over all domains d. We will cover domain independent training more in Section 4.2 of our report.

3. Datasets

Similar to Wang et. al. [14], we focus our attention most specifically on the CIFAR-10 dataset [9], and a little bit on the CelebA dataset [10]. This allowed us to most easily compare to the results gotten by Wang et. al., and moreover, it allowed us to stay away from the ImageNet dataset, which has been shown not to be a super effective benchmark anymore. Moreover, these specific datasets (we took a small subset of CelebA for our self-supervised learning experiments) were specifically small, so we could run experiments tractably with our limited compute resources.

3.1. CIFAR-10S

We ran experiments on the CIFAR-10S dataset for the most part, which is a domain-adapted extension of the original CIFAR-10 dataset, specifically with color and grayscale image domains. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes. There are 6000 images per class with 5000 for training and 1000 for testing. We tested on the p95.0 version of CIFAR-10S in which five classes are 95% grayscale and the other 5 are 95% color. For example, if a given class is chosen to be 95% grayscale, then the 5000 training images for that class are split 95% to 5% grayscale to color.

3.2. CelebA

The CelebA dataset is a dataset consisting of more than 200,000 238x178 celebrity images, each labeled with 40 attributes. Roughly 160k of these are training while 20k are validation and 20k are testing. The task this dataset is designed for is multi-label classification. Unlike CIFAR-10S, there are varying degrees of domain distribution unlike the exact 95-5 domain split in CIFAR-10S. Additionally, the attributes differ from each other in more ways than just a linear color-grayscale transformation. For the purpose of our experiment, we will follow the example of [14] and focus on the ”male” attributes as the domain. Thus, our models will perform multi-label classification on 39 labels.

4. Additional Motivation for Studying Bias

Wang et. al. in [14] provide additional motivation for studying bias through a simple experiment. We felt it was important enough to include in the report here. They train a standard ResNet-18 model for 10-way object classification. When they train this model on the skewed CIFAR-10S dataset and test on the colored CIFAR-10 images, they achieve $89.0 \pm 0.5\%$ accuracy. However, they trained the same model on an all-grayscale training set and testing on CIFAR-10 resulted in 93.0% accuracy. The authors postulate that the model trained on CIFAR-10S correlated presence of color and the object classes. Recall that the CIFAR-10S data has half of the classes heavily skewed towards colored images while the other half is skewed heavily towards grayscale images. When the model that was trained on CIFAR-10S sees the colored images from CIFAR-10, it infers that it is from one of the 5 color-skewed classes. Thus, this small experiment demonstrates the need for domain mitigation methods that can help models learn even in the presence of dataset bias.

5. Bias Mitigation Methods

As mentioned previously, our work is an extension upon [14]. Specifically, we apply two of the bias mitigation methods tested by them on our models. We discuss these two bias mitigation methods below.

5.1. Domain Discriminative Training

Domain discriminative training is based on [6] which focuses on *fairness through awareness*. In [6], they capture fairness by classifying individuals who are similar with respect to a task, as defined by a distance metric, similarly. For our experiments, we are using CIFAR-10S and CelebA, datasets in which we can explicitly split the training data into two different domains (i.e. gray vs color for CIFAR-10S). The high level idea would be to encode this domain knowledge and explicitly mitigate it during inference hopes of mitigating bias.

Consider a classification task with N classes. If there are D explicitly known domains, the simplest way to apply domain discriminative training would be to instead train a ND-way discriminative classifier. The target labels need to augmented using knowledge of the domain to permit training. For example, when doing domain discriminative training on CIFAR-10S, the number of domains was 2 (color vs. grayscale images) and the number of classes per domain is 10 (classes don't change in either case), so in total there were 20 "classes" to classify, which led to the domain discriminative classifier.

Once training is complete, there are several ways to use the domain knowledge for inference. [14] utilize a solution introduced in [11] where the outputs of the ND-classifier are interpreted as probabilities. Let $P_{train}(\cdot)$ and $P_{test}(\cdot)$ denote the probabilities of an event under the training and testing distribution, respectively. We assume that our trained model can outputs network activations, s, that be used to compute $P_{train}(\cdot)$ by applying a softmax. From here, they derive the following formula for $P_{test}(y, d|x)$ where y is the target class, d is the domain, and x is the input image.

$$P_{test}(y, d|x) \propto P_{train}(y, d|x)\frac{P_{test}(y, d)}{P_{train}(y, d)}$$

To remove the dependence on the test label distribution, [14] unbiased $P_{test}(d|y)$ and uniform $P_{test}(y)$. Since $P_{test}(y, d) = P_{test}(d|y)P_{test}(y)$, these assumptions imply that $P_{test}(y, d) \propto 1$. Therefore, we have

$$P_{test}(y, d|x) \propto \frac{P_{train}(y, d|x)}{P_{train}(y, d)}$$

From here, [14] derive the three following methods for predicting the target class prediction, $\hat{y} = \arg\max_y P_{test}(y|x)$

$$\hat{y} = \arg\max_y \sum_d P_{test}(y, d|x) \propto \arg\max_y \sum_d \frac{P_{train}(y, d|x)}{P_{train}(y, d)}$$

$$\hat{y} = \arg\max_y \max_d P_{test}(y, d|x) \propto \arg\max_y \max_d \frac{P_{train}(y, d|x)}{P_{train}(y, d)}$$

$$\hat{y} = \arg\max_y \sum_d P_{test}(y, d|x) \propto \arg\max_y \sum_d P_{train}(y, d|x)$$

The first method is based on the Law of Total Probability. The second takes the max over the joint probability instead of the sum. The third is the same as the first but simply removes the weighting by $P_{train}(y, d)$. As we will see later during testing on CIFAR-10S, this third result performs the worst out of the 3 methods due to this extra removal of domain knowledge during inference.

5.2. Domain Independent Training

One worry about disciminative training is that the model will explicitly learn boundaries between the same class but across the different domains. Since the N-way class boundaries are most likely to be similar even across different domains, it seems unnecessary to learn this extra knowledge. Thus, [14] propose domain independent training which relies on training separate classifiers per domain. Instead of doing this as an ensemble model, the different classifiers will share the same feature representation.

For inference, if the domain $d^\star$ is known at test time, we can use the simple inference rule

$$\hat{y} = \arg\max_y P(y|d^\star, x)$$

Here, we obtain $P(y, d, x)$ by applying a sigmoid on the network activations, s. Then, we choose the classifier corresponding to domain d^* and take the argmax. However, this entirely ignores what the model learned about the class boundaries in the other domains which seems a bit wasteful. Thus, an alternative inference method is to sum up the network activations at the classifier layer. The inference formula is then

$$\hat{y} = \arg\max_y \sum_d s(y, d, x)$$

In simpler terms, we sum up the outputs for the same class across different domain classifiers to get N-scores and take the argmax to get our final prediction. Both methods mentioned are tested in CIFAR-10S.

For CelebA, the inference methods are slightly different due to the task being multi-label classification. Instead of just predicting one class, we want to output a distribution over the possible classes. These inference methods are

- $\hat{y} = P_{train}(y|d^*, x)$. This simply conditions on the known domain and uses the $N - way$ sigmoid from the corresponding classifier
- $\hat{y} = \max_d P_{train}(y|d, x)$. For each attribute, we take the max across the D classifiers to form our final prediction.
- $\hat{y} = \sum_d P_{train}(y|d, x)$. We sum the probabilities from the different domains
- $\hat{y} = \sum_d s(y, d, x)$. We sum the network activations from the different domain classifiers.

s here corresponds to the network activations and the probabilities are obtained by applying a sigmoid on the s. Informally

6. Architectures

In our project, our goal was to test the domain bias mitigation methods mentioned in section Sec. 5 on self-supervised and semi-supervised methods since [14] already test on a supervised ResNet model.

6.1. SimCLR

SimCLR [3] is a (simple) framework for contrastive learning of visual representations. It is a self-supervised learning approach to machine learning. It consists of four major components as shown in Fig. 1. These 4 steps are expanded upon below.

1. Given an input image, data augmentations are performed on the image to output two correlated views of the same example, $\tilde{x_i}$ and $\tilde{x_j}$. In the original SimCLR paper [3], they use random cropping (with flipping and resizing), color distortion, and Gaussian blurring.

2. A base encoder (i.e. ResNet) is used as a feature extractor to obtain the features $h_i = f(\tilde{x_i}) = \text{ResNet}(\tilde{x_i})$.

3. A projection head $g(\cdot)$, often modeled using a small neural network, maps the features to the space, $z_i = g(h_i)$, where we can apply the contrastive lost in the next step.

4. Given a set $\{\tilde{x_k}\}$ and a pair of positive examples $\tilde{x_i}$, $\tilde{x_j}$, the task is to identify $\tilde{x_j}$ from $\{\tilde{x_k}\}_{k\neq i}$ given $\tilde{x_i}$. For a positive pair of examples, this loss, called the NT-Xent loss, is defined as

$$\ell_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq i]} \exp(sim(z_i, z_k)/\tau)}$$

   where $sim(x, y) = \frac{x^T y}{\|x\|\|y\|}$ is the cosine similarity loss, and τ is a temperature parameter. Using this loss, we can compute a similar matrix. SimCLR is often trained with very large batch sizes such as 8192 due to the nature of the loss. Once the self-supervised model is trained, the feature extractor can be taken out of the network and finetuned for use in other downstream tasks such as classification (as we will show in our experiments).

6.2. FixMatch

FixMatch [13] is a state-of-the-art semi-supervised learning algorithm proposed by researchers at Google Brain, which incorporates a couple of standard techniques in semi-supervised learning: *pseudo-labeling*, where we assign "fake" labels to unlabeled images based on the confidence of the model on said images, and *consistency regularization*, which is the premise that weakly augmented and strongly augmented versions of the same image should be mapped to the same label. This is done by maximizing the $\arg\max$ of the label distribution of a weakly augmented version $\alpha(I)$ and a strongly augmented image $\mathcal{A}(I)$ of some unlabeled image I. More formally, this is done via the unlabeled loss function

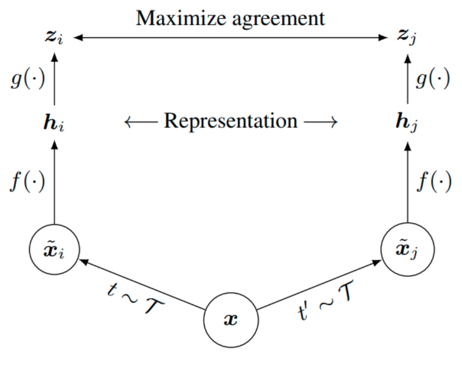

Figure 1. SimCLR pipeline

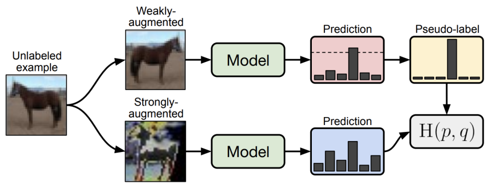

Figure 2. FixMatch pipeline

$$L(I) = \mathbb{1}(\max(p_m(y|\alpha(I))) > \tau) \cdot H(\arg\max(p_m(y|\alpha(I))), p_m(y|\mathcal{A}(I))). \tag{1}$$

where H is the cross-entropy loss. When the model is confident about a certain label (meaning the maximum label probability exceeds the threshold τ) of the weakly augmented image, then we aim to minimize the entropy between the $\arg\max$/delta distribution on that label and the model label distribution on a strongly augmented image $\mathcal{A}(I)$. In this way, the model learns to classify weakly and strongly augmented images with the same label, and by minimizing the entropy we make the model more confident in the predictions made on the strongly augmented images. An illustration of the FixMatch pipeline is given in Fig. 2.

As far as the image augmentations went in practice, weak augmentations α were sequential horizontal flips followed by random crops on the image, whereas strong augmentations $\mathcal{A}$ are these same flips and crops followed by more complicated augmentation techniques from the family of AutoAugment augmentations [4], followed by Cutout [5]. Using the above, we compute the loss over a batch as $\ell_\mu = \frac{1}{\mu M} \sum_{b=1}^{M} L(I_b)$ and also compute a supervised loss, $\ell_s = \frac{1}{B} \sum_{b=1}^{B} H(p_b, p_m(y|a(x_b)))$. Then, we simply aim to minimize the loss

$$\ell_s + \lambda_u \ell_u$$

where λ_u is a fixed scalar hyperparameter.

7. Experimental Setup

7.1. Data Setup

CIFAR-10S As mentioned previously, we used the p95.0 version of CIFAR-10S. For testing the baseline (no bias mitigation method applied), we used the same training labels as the CIFAR-10 training labels. However, the domain discriminative and domain discriminative independent models have a 20-d output so the original training labels were augmented to work with those models. The exact details of how these labels were created can be found at `https://github.com/princetonvisualai/DomainBiasMitigation`. For testing, all models were tested on both the original CIFAR-10S dataset as well as a grayscale version of CIFAR-10S

CelebA Since CelebA contains images that are of much larger size than CIFAR-10 and also contains over 3 times as many images, we chose to downsample the CelebA images to 32x32 by first cropping the center square 178x178 and then resizing the cropped image. We sampled 50k images from the training set of CelebA to match the size of CIFAR-10 and kept the size of the validation and test set the same since inference does not take much time.

7.2. SimCLR

Our SimCLR implementation was based heavily on a publicly available implementation on GitHub `https://github.com/leftthomas/SimCLR`. Specifically, the encoder network is ResNet50 with the linear and max pooling layers removed. Thus, the output size of the encoder is a 2048-d vector. Since ResNet50 was originally built for ImageNet but the implementation is using CIFAR-10, the conv1 layer kernel size is set to 3, the stride is set to 1, and the padding is set to 1. The projection head was a MLP consisting of a linear layer, batch normzliation, ReLu, then a final linear layer to get the final 128-d output.

In the original SimCLR paper [3], the augmentations used were random cropping, color distortion, and Gaussian blurring. However, the implementation we based ours on did not use Gaussian blurring. Specifically, during training, we applied the following transformations

- Random Resized Crop to size 32
- Random Horizontal flip with probability 0.5
- Random Color Jitter with probability 0.8. The parameters for brightness,contrast, and saturation were set to 0.4 and hue was set to 0.1
- Normalization

During testing, we only applied normalization.
CIFAR-10S For CIFAR-10S, we trained using the p95.0 version of the dataset. This training was done using Google Colab Pro with a Tesla P100 GPU with 16GB RAM. Since we could only use 1 GPU, we could not set as large of a batch size as in the original SimCLR implementation. Since SimCLR benefits from larger batch size, this resulted in decreased performance as shown in the results. For training, we set a batch size of 256, temperature parameter of 0.5, feature dimension of 128, and trained for 128 epochs. We also used Adam, as opposed to the LARS optimizer from the original implementation, as the optimizer with learning rate 10^{-3} and weight decay 10^{-6}. This choice was again based on the implementation we based ours off of. Additionally, several other implementations we found also used Adam due to its simplicity as LARS isn't directly implemented in PyTorch. Once SimCLR was done, we moved onto the finetuning stage. We took the ResNet encoder from the SimCLR architecture and finetuned it with Adam for 250 epochs. We fine-tuned the following 3 models

- Baseline model where a final linear layer is added to the end of the encoder that takes the 2048-d output of the encoder down to 10-d. We trained with cross entropy loss.
- Domain discriminative model where the linear layer instead goes to 20-d output. Again, the loss is cross entropy loss
- Domain independent model where the architecture is the same as the Domain Discriminative model but the loss is now a modified version of the negative log likelihood loss used in [14]

For these 3 models, we tested finetuning with the encoder frozen and unfrozen. When the encoder was unfrozen, its learning rate was set to 1e-5. In all cases, the learning rate of the final linear layer was set to 1e-3. For testing, the metric we compute is accuracy.
Celeb-A For Celeb-A, the training for SimCLR itself was identical to CIFAR-10S since the training set had the same number of images and had identically sized images. Thus, the hyperparameters for training SimCLR with CelebA were the same as during CIFAR-10S Next, we fine-tuned the ResNet50 encoder using Adam for 250 epochs in the following 2 models.

- Baseline model in which we added a linear layer, taking the 2048-d output of the encoder down to 39-d, followed by a ReLU and dropout layer. We did not explicitely have a final sigmoid layer. The loss used here was PyTorch's binary cross entropy with loss which combines a sigmoid layer and binary cross entropy loss.
- Domain independent model where the linear layer now goes to 78-d. The loss is a modified version of binary cross entropy loss used in [14].

Due to lack of time, we did not test the domain discriminative method with CelebA. Like with CIFAR-10S, we tested fine-tuning with both the encoder frozen and unfrozen. When the encoder was unfrozen, the learning rate was set to 5e-5. This was set higher than CIFAR-10S due to the increased difficulty of the task. The linear layer's learning rate was set to 1e-3 like in CIFAR-10S. For testing, the metric we compute is mAP.

7.3. FixMatch

For FixMatch, we used a publicly available PyTorch repository, linked here: `https://github.com/kekmodel/FixMatch-pytorch`. This took care of implementation for the most part. We used 1000 labeled examples from the CIFAR-10S dataset (significantly less than the 60K image-label pairs that are in the dataset itself), which limited the amount of supervision during training (which, as our hypothesis at the beginning of the report stated, should, in expectation, encode less bias into the resulting image features). We used a wide ResNet structure as the feature extractor (model depth of 28, width factor of 2, no drop rate). The other hyperparameters were kept the same as in the repo.

8. Experiments and Results

8.1. Original ResNet results

For comparison, we will include the relevant results from [14]. Note, however, that due to lack of compute, the results we got using SimCLR and FixMatch do not match the original paper's supervised models in the absolute, nor did we expect it to. We simply wanted to see if using these domain bias mitigation methods would result in similar or increased performance over the baseline.

8.1.1 CIFAR-10S

The relevant results for CIFAR-10S are shown in Tab. 1. All architectures are based on ResNet-18. As we can see, the domain independent method with test inference $\arg\max_y \sum_d s(y,d,x)$ performs the best over baseline. However, we can also see that the domain discrimative methods also perform well.

8.1.2 CelebA

The relevant results for CelebA are shown in Tab. 4. All architectures are based on ResNet-50. For this, we only show the baseline and domain independent models as we only tested those for SimCLR.

8.2. SimCLR

8.2.1 CIFAR-10S

The results for SimCLR are shown in Tab. 3. This table contains results for both finetuning with a frozen encoder and with an unfrozen encoder. Unsurprisingly, the models where the encoder was frozen perform much worse than the models with the unfrozen models.

As we mentioned earlier, our models in the absolute do not perform as well as the completely supervised training models tested in the original paper. SimCLR benefits from larger batch size but we were constrained to using a max batch size of 256 due to a lack of computing resources. However, the interesting thing to note is that in regardless of whether the encoder was frozen during training, both bias mitigation methods were able to yield significant improvements over baseline. This can most likely be attributed to the fact that we did finetuning for quite a large number of epochs. In the original paper, the authors saw the most improvement, 3-4%, with the domain independent method which also yielded the most improvement for our models.

For the frozen models, the domain independent model (with summed activation for test inference) gained 2.31% in color accuracy and 0.04% in gray accuracy. For the unfrozen models, the same domain independent model gained 1.6% color accuracy and 0.6% gray ccuracy. Interestingly, we saw a larger improvement when finetuning the model with the encoder frozen than when unfrozen. Since the baseline model with the frozen encoder was performing poorly, we believe there were larger gains to be made when switching to the domain independent model. When the encoder was unfrozen, the baseline performance was already very good so we the learning rate of 1e-5 for the encoder was too low for there to be significant improvements when switching to the domain independent or domain discriminative models over baseline.

Another significant result is how the difference in the color and gray test accuracies changed. For the frozen models, we saw that the difference significantly reduced going from the baseline to most of the other models which was expected. In the unfrozen case, the difference widens, albeit only slightly. We believe that since the feature extractor encoder was completely frozen during finetuning, there was less chance for bias to be reintroduced into the model during finetuning which has occurred in the unfrozen case. It is possible that finetuning when the encoder is unfrozen caused more bias to introduced into the encoder, even when training using the bias mitigation models.

8.2.2 CelebA

For CelebA, we saw a similar story in terms of performance increase. In the completely-supervised models, the orig-

MODEL NAME	MODEL	TEST INFERENCE	ACCURACY (%,↑) COLOR	GRAY
BASELINE	N-way softmax	$\arg\max_y P(y\|x)$	89.0	88.0
DOMAIN DISCRIM.	joint ND-way softmax	$\arg\max_y \sum_d P_{train}(y,d\|x)$	88.3	86.4
		$\arg\max_y \max_d P_{test}(y,d\|x)$	91.3	89.3
		$\arg\max_y \sum_d P_{test}(y,d\|x)$	91.2	89.4
DOMAIN INDEP.	N-way classifier per domain	$\arg\max_y P_{train}(y\|d^\star,x)$	89.2	88.7
		$\arg\max_y \sum_d s(y,d,x)$	**92.4**	**91.7**

Table 1. Supervised CIFAR-10S results

MODEL NAME	MODEL	TEST INFERENCE	mAP
BASELINE	N-way sigmoid	$P(y\|x)$	74.7
DOMAIN INDEP.	ND-way sigmoid	$P_{train}(y\|d^*,x)$	73.8
		$\max_d P_{train}(y\|d,x)$	75.4
		$\sum_d P_{train}(y\|d,x)$	76.0
	ND-way classifier	$\sum_d s(y,d,x)$	**76.3**

Table 2. Supervised CelebA results

nal authors saw a maximum mAP increase of 1.6 when going from baseline to a domain independent model. In our case, we saw a maximum increase of 1.61 for the frozen models and 1.05 for the unfrozen models. Thus, unlike with CIFAR-10, we actually saw very similar improvements when using bias mitigation methods.

8.3. FixMatch

We focused on CIFAR-10S specifically for FixMatch. Due to CelebA being a multilabel dataset, and FixMatch's loss function being geared only for images with one label, we focused completely on the CIFAR-10S dataset.

8.3.1 CIFAR-10S

Due to limited compute, we ran 3 experiments total here: one standard baseline, one with domain independent training, and one with domain discriminative training due to ease of implementation. Our results are shown in Tab. 5 below. It can be seen that not much improvement was made using the domain independent strategy even (due to less labels for supervision), but it must be noted that the performance difference of FixMatch on the color and grayscale domains was actually not that much different (smaller than the 4% difference in test accuracy on color and grayscale domains from Wang et. al. [14]), which indicates less bias was learned than in the standard ResNet purely supervised model. However, while accuracy floundered a little bit, the difference in accuracy between color and grayscale domains clearly goes down, with FixMatch to less than 0.01% when we summed the probabilities over all domains, which indicates that less bias was learned across those domains using the domain independent training model. We believe that SimCLR saw improved performance compared to FixMatch due to the finetuning that was required to use the encoder in the downstream task. While FixMatch does also use the labels, it combines this with a task involving unlabeled data. It does not require nor use any finetuning. Thus, in this sense, it seems to actually be easier to compare the bias mitigation models to the baseline as we don't need to do extra finetuning which can possibly reintroduce bias.

In conclusion, the bias mitigation strategy works well for FixMatch. Although the accuracy didn't improve, we can see that the difference in accuracy between color and gray images decreased significantly.

9. Future work

We think that future work can build on our experiments presented in this report. Due to limited compute, we could only run for one or a few seeds, so validating our experiments with more than one seed would be useful as well. Additionally, many things like the batch size for SimCLR, the size of the dataset or images used, the model size, etc., had to be limited due to lack of compute.

One idea discussed in our project proposal that we didn't have time to implement and test is to see how purely unsupervised models encode bias into their learned features. One such way to do this is, for example, to train a deterministic autoencoder or variational autoencoder (VAE) [8] on a set of images in CIFAR-10 or CIFAR-100, and use the encoder convolutional layers to do inference/fine-tune for inference in a similar fashion as to how we did it in this report. We believe that due to the absence of labels in the training scheme that maybe even less bias would get encoded into the resulting features, which is similar to what we saw in

MODEL NAME	FROZEN	MODEL	TEST INFERENCE	ACCURACY $(\%,\uparrow)$ COLOR	GRAY
BASELINE	YES	N-way softmax	$\arg\max_y P(y\|x)$	80.46	82.75
DOMAIN DISCRIM.		joint ND-way softmax	$\arg\max_y \sum_d P_{train}(y,d\|x)$	76.55	83.24
			$\arg\max_y \max_d P_{test}(y,d\|x)$	82.52	82.06
			$\arg\max_y \sum_d P_{test}(y,d\|x)$	82.53	82.6
DOMAIN INDEP.		N-way classifier per domain	$\arg\max_y P_{train}(y\|d^\star,x)$	77.17	77.56
			$\arg\max_y \sum_d s(y,d,x)$	**82.77**	**82.79**
BASELINE	NO	N-way softmax	$\arg\max_y P(y\|x)$	87.28	87.36
DOMAIN DISCRIM.		joint ND-way softmax	$\arg\max_y \sum_d P_{train}(y,d\|x)$	84.0	87.85
			$\arg\max_y \max_d P_{test}(y,d\|x)$	88.05	87.51
			$\arg\max_y \sum_d P_{test}(y,d\|x)$	88.09	87.77
DOMAIN INDEP.		N-way classifier per domain	$\arg\max_y P_{train}(y\|d^\star,x)$	85.40	83.21
			$\arg\max_y \sum_d s(y,d,x)$	**88.78**	**87.96**

Table 3. SimCLR CIFAR-10S results

MODEL NAME	FROZEN	MODEL	TEST INFERENCE	mAP
BASELINE	YES	N-way sigmoid	$P(y\|x)$	56.39
DOMAIN INDEP.		ND-way sigmoid	$P_{train}(y\|d^*,x)$	55.34
			$\max_d P_{train}(y\|d,x)$	57.17
			$\sum_d P_{train}(y\|d,x)$	**58.00**
		ND-way classifier	$\sum_d s(y,d,x)$	57.64
BASELINE	NO	N-way sigmoid	$P(y\|x)$	65.79
DOMAIN INDEP.		ND-way sigmoid	$P_{train}(y\|d^*,x)$	64.95
			$\max_d P_{train}(y\|d,x)$	65.91
			$\sum_d P_{train}(y\|d,x)$	66.67
		ND-way classifier	$\sum_d s(y,d,x)$	**66.84**

Table 4. SimCLR CelebA results

SimCLR and FixMatch in this report.

References

[1] Alexander Amini, Ava P Soleimany, Wilko Schwarting, Sangeeta N Bhatia, and Daniela Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019. 1

[2] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018. 1

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 4, 5

[4] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019. 4

[5] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017. 4

[6] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. Fairness through awareness. *CoRR*, abs/1104.3913, 2011. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1

[8] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019. 7

[9] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 1, 2

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. ICCV '15, page 3730–3738, USA, 2015. IEEE Computer Society. 1, 2

[11] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Comput.*, 14(1):21–41, jan 2002. 3

[12] Ignacio Serna, Alejandro Peña, Aythami Morales, and Julian Fierrez. Insidebias: Measuring bias in deep networks and application to face gender biometrics. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3720–3727. IEEE, 2021. 1

MODEL NAME	MODEL	TEST INFERENCE	ACCURACY $(\%, \uparrow)$	
			COLOR	GRAY
BASELINE	N-way softmax	$\arg\max_y P(y\|x)$	80.62	78.13
DOMAIN DISCRIM.	joint ND-way softmax	$\arg\max_y \sum_d P_{train}(y, d\|x)$	80.82	78.62
		$\arg\max_y \max_d P_{test}(y, d\|x)$	82.40	80.60
		$\arg\max_y \sum_d P_{test}(y, d\|x)$	82.40	80.59
DOMAIN INDEP.	N-way classifier per domain	$\arg\max_y P_{train}(y\|d^\star, x)$	79.25	79.13
		$\arg\max_y \sum_d s(y, d, x)$	**79.91**	**79.98**

Table 5. FixMatch CIFAR-10S results

[13] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence, 2020. 1, 4

[14] 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. 1, 2, 3, 5, 6, 7

[15] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints, 2017. 2