Exploring Racial Bias and Bias Amplification in Vision Models and Datasets

Kunal Kasodekar
SCAI
Arizona State University
Tempe, U.S.A
kkasodek@asu.edu

I. Problem Statement

Computer Vision models have become an integral part of our day-to-day life due to their hand in automating a multitude of manually excruciating tasks. These models play a vital role in seamlessly automating tasks such as Face Detection in Biometrics, Fast and accurate Object Detection and driving autonomous cars. Recent research has demonstrated that the broad adoption of these vision models throughout the technology space has reinforced prejudice against particular groups. This bias appears to have permeated the data and been amplified by the models. This prejudice focuses on sensitive attributes such as gender, age, occupation, race, and ethnicity. In this report I:

- Identify inherent Racial Bias in the COCO dataset and Image models trained on this dataset.
- Find Bias Amplification [1] due to Image Models.
- Reduce the Amplification of Racial Bias by experimenting with various kinds of adversarial losses.
- Add a novel approach to Bias Mitigation.

II. Challenges and Main Contributions

A. Challenges

Working on this project presented many difficulties. Understanding the relevant work and its implementation code presented some of the biggest difficulties, particularly for the paper that served as the project's motivation and was titled "Balanced Datasets are Not Enough" [1]. The implementation code was like a sea, difficult to navigate through and make sense of for beginners. Both the GitHub repository [6] and the paper provided very little context for the implementation. As a result, since the code was intangible to me, I wrote it from scratch, drawing heavily on the paper for inspiration. Another challenge was finding novel ways to augment research in Bias Mitigation on Vision Models. It was difficult to come up with plausible ideas that could have some ground-level impact. Since I was using Colab, the computation was a major problem for me because it prevented me from working with the entire dataset. It took a long time to load just thousand's of images from the COCO Annotations. Many of the approaches described below needed me to advance my understanding of model creation, building, and training in PyTorch before I could finally create and implement PyTorch models for them.

Some challenges on the implementation side were:

- Extracting outputs from intermediate layers for model leakage calculation.
- Designing and implementing multibranch neural networks with adversarial loss functions and gradient reversal layer.
- Implementing multilabel classifiers.
- Modifying Resnet18 architecture to accommodate multiple branches, augment different model architectures and concatenate their outputs as a single adversarial prediction.
- Preventing overfitting while working with less data.

B. Contributions

Most of the prior works on Bias mitigation and Bias Detection focus specifically on gender bias. This is more prevalent in the case of Bias detection and Mitigation methods for vision models. To focus on racial bias authors of [7] manually annotated the COCO dataset (train2017instances.json) with skin tones [Light, Dark]. The authors provide skin tone annotations for each image in images_val2014.csv for the person object with the largest bounding box. I focus on detecting racial bias in the COCO dataset by initially using techniques outlined in the paper," Balanced Datasets are not Enough" [1]. Due to a lack of computation power, I sample 3000 images for our classification dataset. I calculate the distribution of default labels of the images in the COCO dataset and find that 35% of our images are made up of single labels. Hence I create three classification datasets as follows:

- Dataset with COCO images and their corresponding skin annotations and single COCO labels.
- Dataset with COCO images and their corresponding COCO labels.
- Dataset with COCO images and their corresponding skin annotations
- I will work on multilabel classification in the future.

I calculate racial dataset leakage[1] wherein I try to predict the skin colour from the ground truth annotations using an attacker f. I also add noisy labels and flip some target labels randomly to account for systematic bias

Further, I calculate racial model leakage [1] for our two models above one where I predict COCO labels and genders together and the other where I only predict COCO labels. I also train a model to predict skin colour from images to act as an accuracy baseline model.

Although [1] is a significant influence on my work, all of the implementation model designs are original to me and significantly distinct from those of the authors. These specifics are described in [1] below in the Experiments part. Additionally, since this has never been done before, I find bias amplification, dataset bias, and model leakage and do bias mitigation for skin colour rather than gender.

Then I calculate the racial bias amplification [1]. Further, I use adversarial debiasing methods from [1] wherein I: Debias the model by adding a critic loss where the adversarial branch tries to predict skin colour from intermediate representations of my resnet-18 model. I experiment with the following novel approaches to mitigate racial bias:

- I add the adversary and task predictor after the final average pooling layer. The loss function tries to minimise task predictor loss and maximize the adversarial branch (skin colour predictor) loss. This hinders the model from learning hidden representations with spurious correlations to skin colours whilst the model can efficiently learn the task (COCO Labels).
- I create multiple branches of classifiers stemming from the last three convolution layers and concatenate their outputs. A final classifier layer is added to this to predict skin tone. These linear layers extract/use features from intermediate convolution layers and I concatenate features from all these convolution layers to predict the skin tones. I incorporate a critic loss c, in the loss function wherein we try to maximize the loss of these adversarial branches whilst minimizing task loss to predict COCO labels.

I modify the architecture with an adversary model and task-specific classifier at the final pooling layer wherein I add a convolutional auto-encoder model at the first layer. The input image is first passed through the auto-encoder and then through a decoder to the Resnet-18 block. I create a classifier branch from the bottleneck layer and concatenate these extracted features to the adversarial branch at the adaptive pooling layer. Finally, a classifier is added to predict racial cues(skin tones) from these concatenated features. I also have a task-specific predictor responsible for predicting COCO Labels. Our loss function maximizes the critic loss (Our adversary) and minimizes the task loss.

III. Related Work and their Shortcomings

Studies have been conducted on detecting and mitigating bias in Vision Models and Datasets. [1] Shows that balancing the ratio of protected attributes in a dataset is not enough to minimize gender bias. Despite balancing the dataset the authors observed that the model amplified the bias due to the intermediate representation making a spurious correlation between the protected attributes and the labels. Hence the authors devise a new method of estimating bias by defining model bias amplification as the difference between the model leakage and dataset leakage. The authors follow a bias mitigation method known as Fairness through Blindness [2]. The paper mitigates bias amplification by around 68% by using an adversarial loss on the input feature space with a mask on the input image from the encoder-decoder bottleneck. The authors also experiment with adding an Adversarial loss to the intermediate layers to mask hidden spurious correlations between the representation and the output labels. I use the definitions for dataset and model leakage outlines in this paper to evaluate racial bias amplification using different labels, model architectures and methods. Further I propose two new architectures for Bias Mitigation through adversarial learning.

[2] Refutes this methodology by giving the reasoning that masking hidden relations has less effect on bias amplification compared to the loss of accuracy as proxy correlations can be made between the hidden attributes and the output labels. This phenomenon is also called redundant encoding. I make use of this fact to add an auto-encoder model at the input layer and mask/remove hidden feature correlations at the bottleneck layer itself by incorporating an adversarial loss. From this z layer image will be reconstructed with subdued features that would predict protected attributes. The method employed by me uses the facts laid out in this paper but my methodology is completely different.

[3] Evaluates the effect of image distortions on the bias in face recognition using the CelebA dataset. The study finds that features such as eyes, nose and mask are the most discriminative regions across race and gender subgroups. In case these discriminative features are distorted other less discriminative features will contribute to recognition. However, authors only see the impact of distortion on discriminative characteristics without accounting for Bias Amplification from the model. My auto-encoder adversarial debiasing methods try to incorporate this principle while generating a reconstructed image with suppressed hidden relation image features.

[4] Mitigates scene bias in learning video representations using additional adversarial losses. Standard cross-entropy loss is augmented with an adversarial loss for the scene class and in the second entropy loss, humans are masked out. This helps the model in learning and understanding the actual action in the scene. This method borrows concepts heavily from the [2]. However, the paper does not account for the combination of other features that may act as proxy features.

IV. Dataset

I use the manually annotated COCO dataset (train2017instances.json) with skin tones by [7]. The authors provide skin tone annotations for all the Person objects with the largest bounding box. This dataset has binary labels for skin tones: Light and Dark. This will act as our baseline data for training, testing and evaluating the methodologies for Bias Amplification and adversarial loss (Bias Mitigation) defined below. Along with this dataset, I use the Common Objects in Context (COCO) Dataset for training and testing for our task-specific i.e COCO Labels classification model and multilabel classification of COCO labels as well as skin tones. Due to a lack of computation power I only use a small subset of this dataset (3000 images)

V. Methodology

A. Dataset Loading and Creation

Initially, I load the skin tone annotations data frame provided to us by [7]. I obtain the image ids and corresponding skin tone labels. Then I load the COCO, 2017 Instances train dataset using pycoco tools. I extract image ids corresponding to skin tones from the COCO dataset and images into an np array with corresponding skin tones and ground truth COCO Label. I extract single ground truth COCO labels with the largest bounding boxes for classification as 35% of the images in our dataset comprise 1 label (I will do multilabel classification in the future). I sample a small subset of images for experimentation due to device constraints.

B. Bias Amplification

For our baseline model accuracy, I fine-tune a Resnet-18 model to detect skin tones from images. I then calculate our Dataset Leakage as follows:

- Evaluate the Dataset leakage on this dataset: The dataset leakage is defined as the fraction of data points in D that leak information about our hidden attribute (skin colour) g_i through our annotated image Y_i. It is defined as:

  $\lambda_D = \frac{1}{|D|} \sum I[f(Y_i) == g_i]$ (I is the indicator function)

- I use our ground truth labels of the COCO dataset to predict the Skin colour of the Person Supercategory in our images. The images are made of other objects as well so our ground truth labels consist of multiple categories. I add some noisy labels to our ground truth annotations and randomly flip some labels to reduce systematic bias.
- Initially I experiment with simple classifier models like SVM and Logistic Regression to predict our skin tones from these ground labels. I finally use SVM to calculate dataset leakage due to other model's overfitting on the data.

Then I evaluate the Model leakage as follows:

- I calculate Model leakage as the percentage of samples in Dataset D that leak information about our hidden attribute (skin colour) g_i through our predicted label $\hat{Y}_i$.
- I train two models, one to jointly predict skin colour and our task-specific COCO labels and the other to predict task-specific COCO Labels. Both these models are trained by fine-tuning Resnet-18 with some linear probes.
- Our multilabel classification model for detecting skin colour and COCO label jointly is trained using a Binary Cross Entropy loss with logits and with a one-hot-encoded target array. Accuracy is calculated by determining the hamming distance between the predicted and actual one-hot-encoded array.
- After training both these models I use PyTorch Hooks to get the preactivation logits at the penultimate layer for both these models. I train a simple neural network that accepts these flattened activations as inputs. I train the model to predict skin tones from these logits.

Finally I calculate **Bias Amplification as**: $\Delta = \lambda_M - \lambda_D$.

As my dataset is very small after sampling the dataset to make it balanced by creating equal co-occurrences for both skin tones, the size of the dataset significantly reduces and the models overfit. Hence instead of equal co-occurrence, I have equal target (skin tones) labels in our dataset. However, due to the small dataset size, there is no significant difference in Bias amplification whether the data is balanced or not. In the future, I will use a larger dataset and then evaluate current methods on a subsampled balanced dataset.

C. Bias Mitigation

The authors in [1] try to minimize Model Leakage by training an adversarial branch with an auxiliary loss to remove gender cues across intermediate representations of the trained Resnet-50 Model at different intermediate layers. They optimize the model over the Task-Specific model loss and minimize the adversarial loss to remove hidden representations responsible for predicting gender. I take inspiration from this approach to remove/minimize racial cues from these hidden representations and create a model that has hidden attributes with almost no correlation with protected/hidden racial attributes. I have designed and experimented with two novel architectures that try to improve upon the previous author's work on Bias mitigation. I explain my three architectures for bias mitigation below:

1) Initially I use a Resnet-18 model with task-specific layers and an adversarial branch/critic c after the last adaptive pooling layer of the Resnet that attempts to predict skin tones (Protected attributes) from the penultimate layer of the Resnet. I create a loss function wherein the Task-Specific predictor tries to minimize its loss over its predictions whilst increasing the critic's loss. I have a gradient reversal layer that helps in minimizing the predictor's loss and maximise the critic's loss. The loss is defined as follows:

$$\sum Ł(p(X_i, Y_i)) - \lambda \ell_c(c(h_i, g_i))$$

2) The authors in [1] remove leakage from hidden representations at different stages in a Resnet Model and from the input image. This includes applying the Bias mitigation through an adversarial loss via a critic at conv4, conv5, and the input image. I propose and implement a new method wherein I:

a) Construct three adversary branches at layers conv6, conv11 and penultimate adaptive pooling layer. These are simple linear layers that probe from these hidden representations. To feed features from the previous convolution layers adaptive pooling is applied. Later all these 3 branches are concatenated to a 96-stacked neuron branch. Finally I append a classifier to the concatenated features to detect skin tones. I also have my task-specific predictor predicting the

COCO Label. All these branches are appended by a gradient reversal layer. Finally, I train this model end-to-end with a low learning rate of 3e-6 using an adam optimizer. The loss function is defined as:

$$\sum \ell_p(p(X_i, Y_i)) - \lambda \sum \ell_c(c(h_1, h_2, , h_3, g_i)))$$

where the critic is maximizing its loss from predicting racial cues(skin tones) from three intermediate representations across the Resnet layers.

b) Finally, I add a convolutional auto-encoder model at the first layer. The input image is first passed through an encoder followed by a bottleneck layer of size 128x1x1 and then through a decoder to the Resnet-18 block. I create a classifier branch from the bottleneck layer and concatenate these extracted features to the adversarial branch at the adaptive pooling layer. Finally, I add a classifier to predict skin colour from representations of the bottleneck layer and the last Resnet-18 layer. I also have a task-specific predictor at the penultimate layer so that the critic loss is maximized while minimising the task loss. I hypothesize that suppressing feature correlations from the bottleneck layer will suppress features that contribute to skin tone prediction indirectly. I reason that hiding these features in the bottleneck layer will result in the reconstructed image at the decoder lacking skin tones cues. Thus lack of these features will make it difficult to predict skin colour from these reconstructed images. Further our adversarial branch will help in eliminating the remaining existing features.

VI. Experimental setup and Results

For calculating dataset leakage I train an SVM with C=1000 on my ground truth COCO annotations for predicting noisy flipped skin tones. An accuracy of **75.6%** is obtained in detecting these skin tones. This will correspond to our dataset leakage. I tried training a Logistic regression model for the same however it was overfitting.

For calculating Model Leakage, I initially Fine-Tune a Resnet-18 model with a linear probe that jointly predicts, skin tone and COCO Label. I use one-hot encoding to jointly predict both these attributes, where the first 1 in the target label corresponds to the presence of a particular COCO class and the second 1 to the presence of a particular Skin tone. The one-hot encoded target size is a vector of size 90. I use BCEWithLogitsLoss(), Adam optimizer and a learning rate of 3e-4. I achieved an accuracy of **98.45%**. It seems that jointly predicting two attributes increases the accuracy.

For the accuracy baseline, I fine-tune a Resnet-18 model with attached linear probes to detect skin tones. It achieves an accuracy of **90.20%** with Adam optimizer, cross-entropy loss and a learning rate of 3e-4. I also use a lrscheduler that decays the learning rate by a factor of 0.1 every 7 epochs.

Finally, I fine-tune a Resnet-18 model with a linear probe to predict the COCO label from the images. It has the same hyperparameters as the above model and achieves an accuracy of **88.6%**.

I attach a PyTorch hook at the penultimate layer of all the above models to get the logits. On the forward pass, I can obtain the activated logits for the aforementioned models. Corresponding to the length of the logit's tensor I, train a simple neural network model with 5-6 layers and input neurons being 128/256 and the final layer having a two-neuron output. I predict the ground truth annotations from these logits and calculate the model leakage. The difference in the accuracies of the model leakage will be our Bias Amplification.

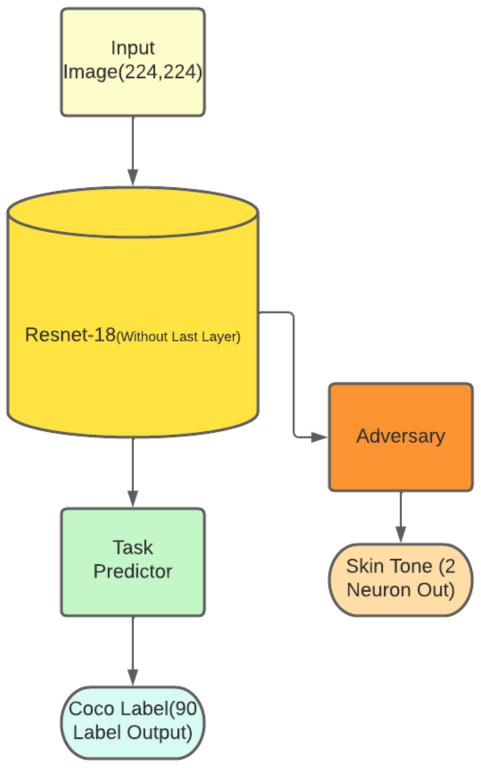
For the model jointly predicting labels and skin tones, I obtain a model leakage of **91.0465** and thus Bias Amplification is: 15.4465. For the model predicting COCO labels, I obtain a model leakage of **92.65** is 17.05.

For Bias Mitigation Initially, I obtain an unfrozen Resnet-18 model without its last classification layer. Then I add two branches, one the adversary that predicts the skin tones (racial cues) and the other the task predictor a linear probe on top of the adaptive pooling layer (1). Based on the loss function above I maximize the critic loss via gradient reversal and minimize my task loss. I train the whole model end-to-end with a learning rate of 3e-6, Adam optimizer with Cross-Entropy loss and obtain an accuracy of **96%**.

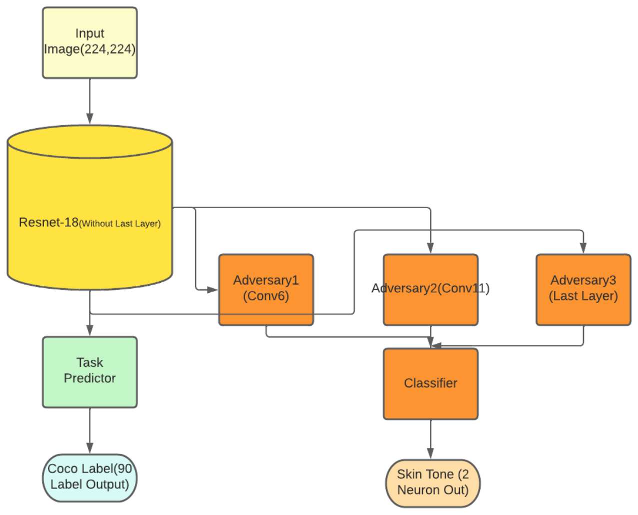
Based on the method outlined above I train my concatenated adversarial branches (2) to predict the skin tones and my default task predictor model to predict the COCO Label. The model initially gets stuck in a local minimum but once the gradient starts flowing it achieves an accuracy of **97.4%**. I use the same hyperparameters as above and train them for 50 epochs to prevent overfitting. The loss function minimizes the task loss while maximizing the concatenated adversarial model.

Finally for the Auto-Encoder model (3) I train it end-to-end for 100 epochs (Performance plateaus)and achieve an accuracy of **71.5%** with the same hyperparameters and a learning rate of 1e-6. The concatenated logits/activations from the adversary branches stem from the bottleneck layer of the Auto-Encoder and the penultimate layer of the Resnet-18 Model. The loss function tries to mask/suppress the hidden attributes responsible for predicting skin tones in the bottleneck layer. Thus it can be hypothesized that the reconstructed image will lack these attributes. Any remaining attributes will be removed by the final adversarial branch at the last layer.

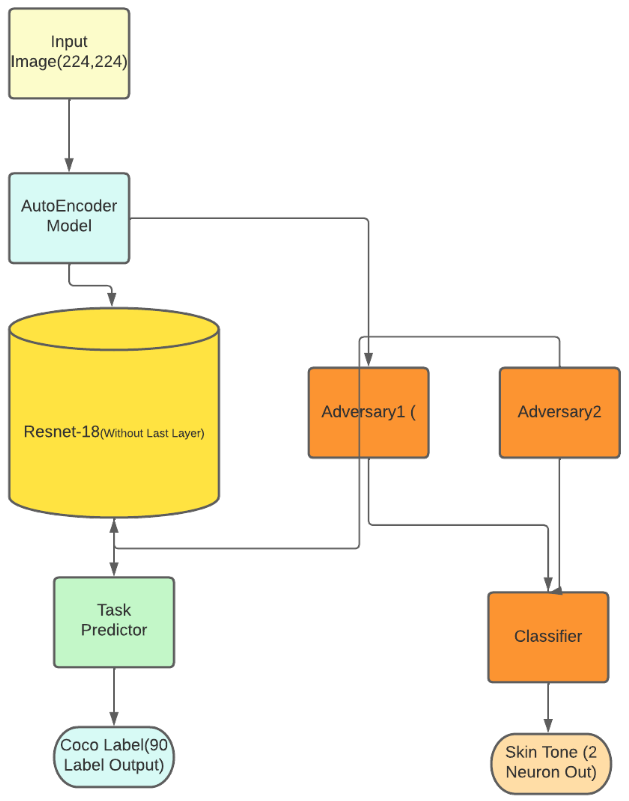
I finally calculate the Model leakage by obtaining the pre-activation logits from the penultimate layer of the task predictor. For the Adversary with the Auto-Encoder, the model leakage is **90.625**. For the adversary with multiple concatenated branches, the model leakage is **90.65**. Thus, I obtain a bias amplification of 15.025 and 15.05 for the aforementioned models above. Thus I see a slight decrease in bias amplification in these models. Using a larger training set and further doing multiple experiments can help us in solidifying our results and obtaining further new insights. We can reason that using concatenated hidden representations from multiple layers can help in better Bias mitigation. The results we obtain are not up to the mark but can be vastly improved if we make use of more computation power, train more data and reiterate through our solutions. We can also reason the same for our bottleneck layer adversary as both these models in theory should do better Bias Mitigation. In the future, I will do multi-label classification, do more experiments on the whole datasets, further validate the results above and reiterate through multiple solutions to obtain State of Art Bias Mitigation methods for Vision models.



(1) Bias Mitigation with Adversary at last layer



(2) Bias Mitigation with Concatenated Adversaries



(3) Bias Mitigation with Adversary from Bottleneck layer of Auto-Encoder

VII. GITHUB REPOSITORY

Link:https://github.com/gremlin97/RacialBiasDetectionAndMitigationCSE598

Gist(Experimentation):https://github.com/gremlin97/Mitigating-Racial-Bias-Experiments/blob/main/CocoCSE598.ipynb

VIII. REFERENCES

1) https://arxiv.org/abs/1811.08489
2) https://arxiv.org/abs/1911.11834
3) https://arxiv.org/abs/2108.06581
4) https://arxiv.org/abs/1912.05534
5) https://arxiv.org/abs/2106.08503
6) https://github.com/uvavision/Balanced-Datasets-Are-Not-Enough
7) https://princetonvisualai.github.io/imagecaptioning-bias/