National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Machine Learning

Semester Project

Dataset Bias+Skin Colour Classifier

Group Members:

Junaid Ali (290927)
Amur Saqib Pal (288334)
Sannan Abbasi (293786)

Abstract.

There are different types of biases that can exist in a dataset. While developing a Machine Learning model, the problem with having a biased dataset is that the model ends up acting with biases as well. This can cause problems such as skewed results, results with low accuracy and analytical errors. In this project, we are exploring sample and racial data biases that exist in different search engines (Google, Bing, DuckDuckGo) when we make queries regarding different nationalities. Furthermore, we will also attempt at developing a skin colour classifier which would accurately predict the skin colour of a person through the images we get through the different search engines.

Introduction.

There are two parts to this project. One part deals with creating a clean data set using the images we get through 3 different internet search engines that are Google, Bing and DuckDuckGo. The images had to be divided into three colour sets. These three colours are black, brown and white.
The other part deals with developing a model that can accurately tell the skin colour of a person inside any image. For the second part, we trained a model using the Convolutional Neural Network (CNN) Classifier.

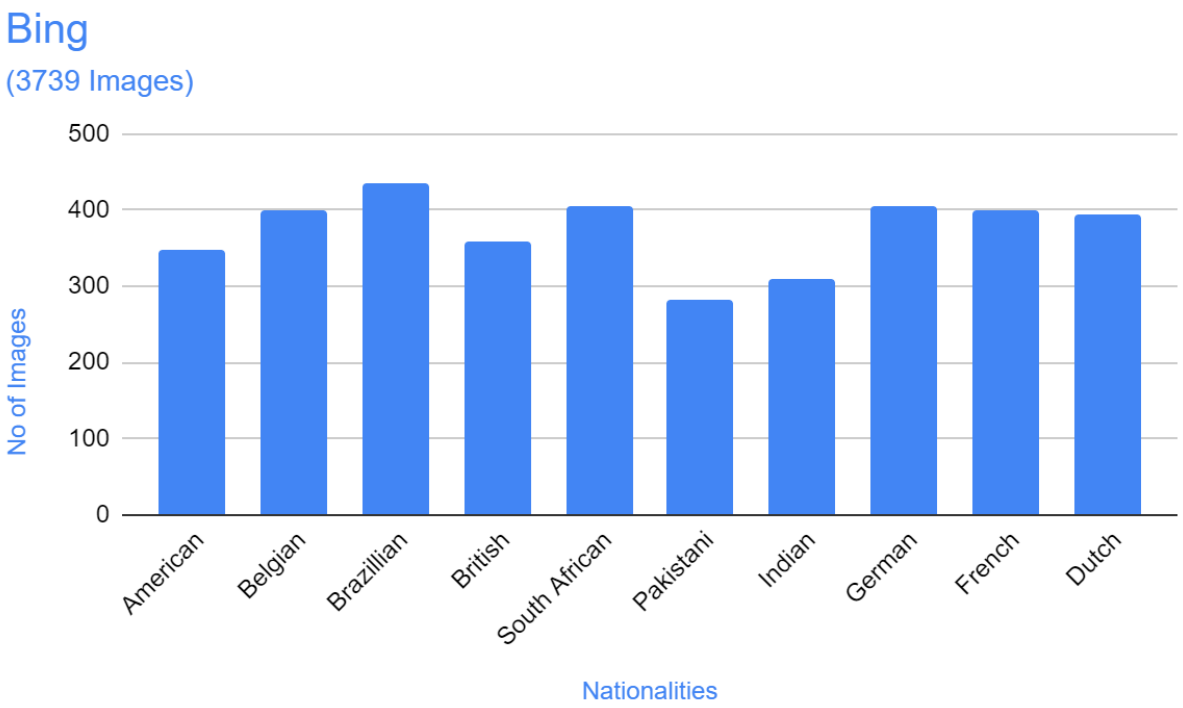
Data Collection.

The search query we were supposed to make in the different search engines was regarding people of different nationalities. The 10 countries we chose for this particular project are as follows.

1. America
2. England
3. Brazil
4. South Africa
5. The Netherlands
6. Belgium

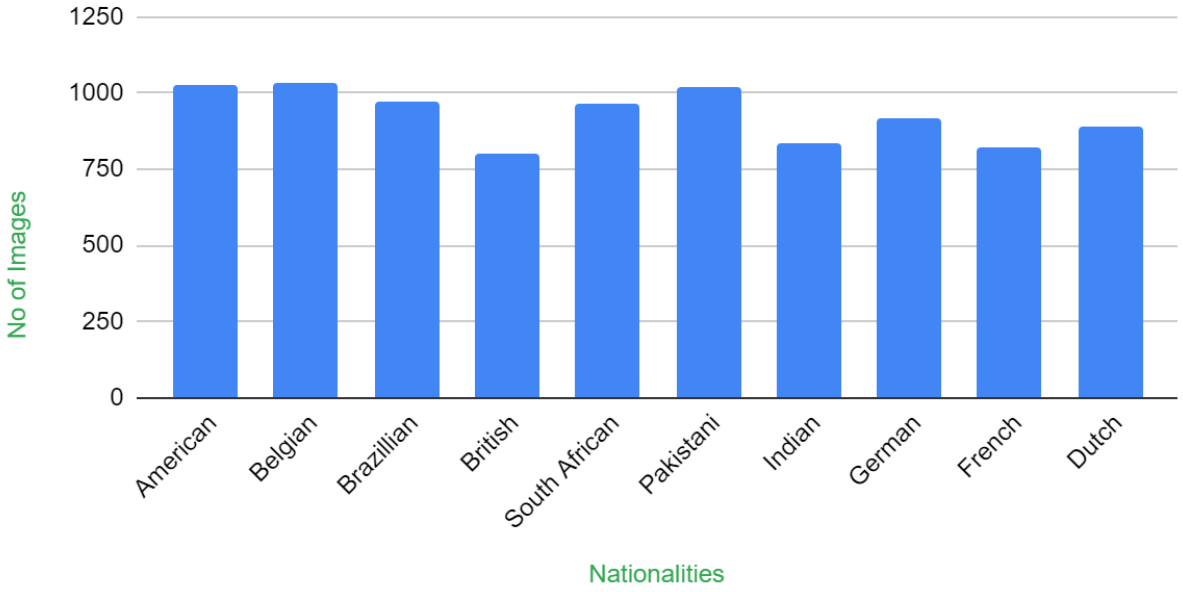
7. France
8. Germany
9. Pakistan
10. India

The reason behind choosing these countries was so that we could create a data set that would have equal representation of the three skin colours i.e black, brown and white.
We downloaded all of the images available for the particular queries using a Google Chrome extension called “Download All Images”. We were tasked with getting at least 1000 images per search engine. However, there were a limited number of images per search engine that we could get. For instance, we could only get 373 images on average per nationality using the Bing search engine. The numbers were better for DDG and Google with an average of 888 and 927 images per nationality respectively. Belgium gave us the most number of images among all countries with 2368 total images. All the images that were downloaded are present in a “rar” file called “images” in the drive link provided at the end. Following are the numbers associated with each search engine upon downloading the images.



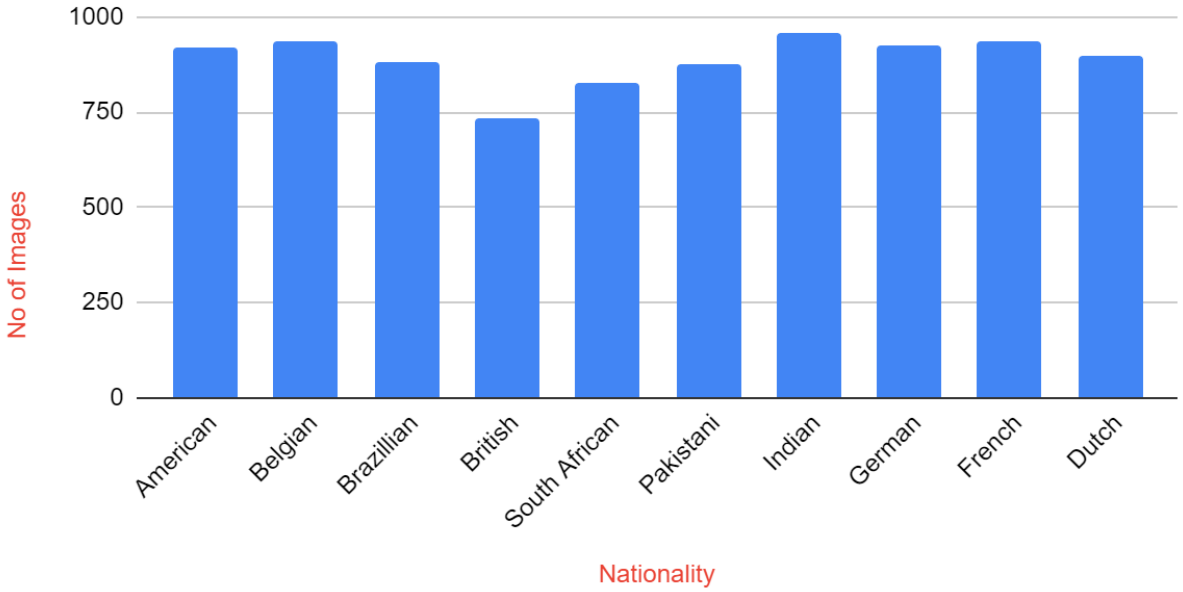
Google

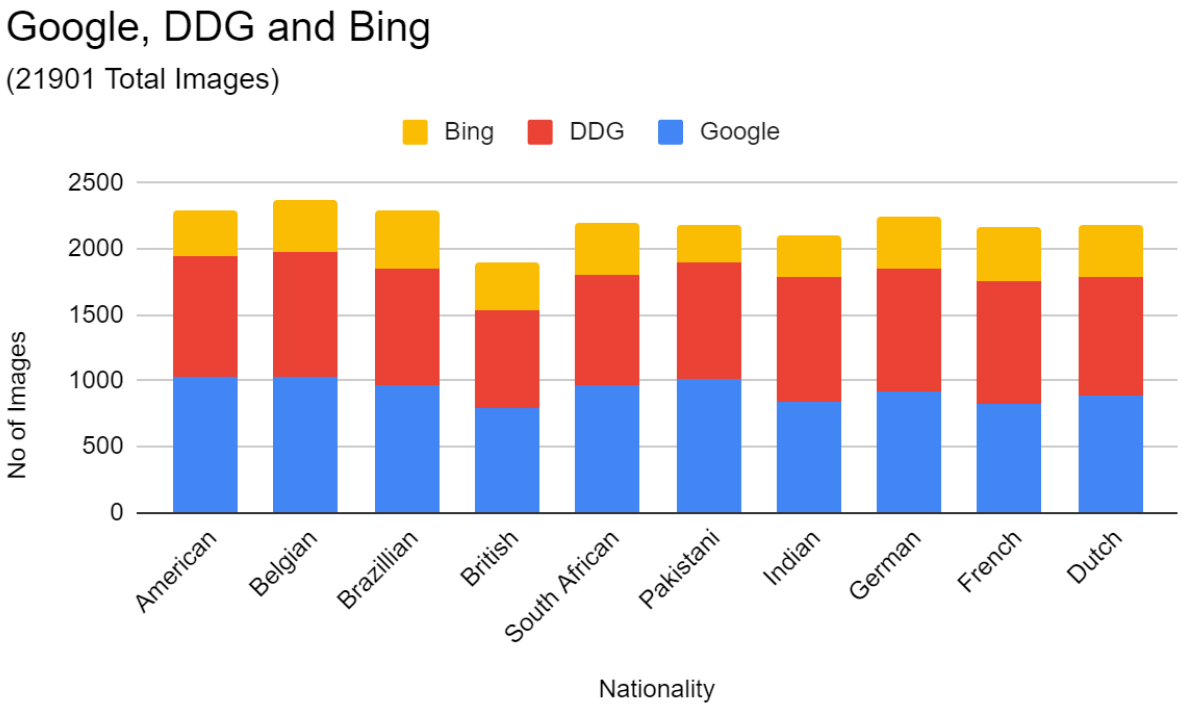
(9273 Images)



DDG

(8889 Images)





Cleaning and Labelling the Data.

After downloading the data, the next step was to clean and label the images we had downloaded. We decided that we would remove all the images that had little to no skin that could be classified. The images that had a group of people with different skin colours and images showing a crowd were termed as outliers. An image containing a group of people with similar skin colours was labelled under that particular skin colour. For example.

(This image was labelled an outlier)

(This image was labelled as white)

There were several black and white images in the downloaded images as well. Most of them were deleted. Some of them were kept if it was easier to make out the colour of the person's skin. For example, the following image was labelled as "white".

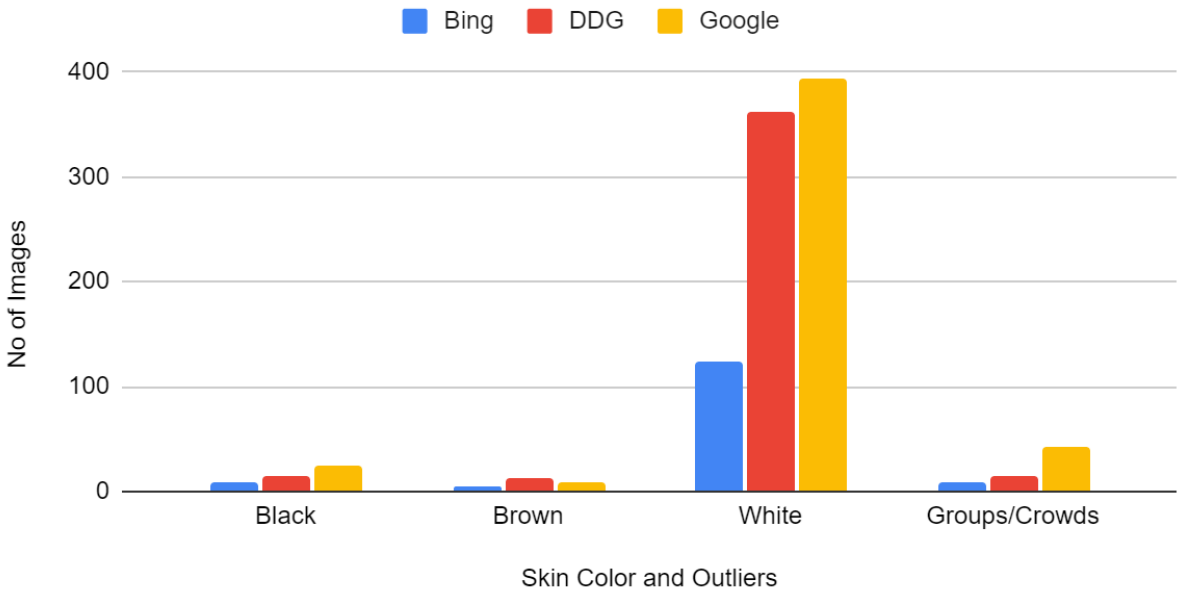
We first cleaned and labelled the following nationalities.

1. German
2. Brazillian
3. French
4. Belgian
5. South African

The cleaned images dataset was uploaded to a drive and mentioned in the initial report. Next, we moved on to the other nationalities.
After the downloaded images were cleaned and sorted, the number of images was reduced. Then they were placed in separate files labelled as “brown”, “black” and “white”. Some information on the number of images per nationality left is as follows.

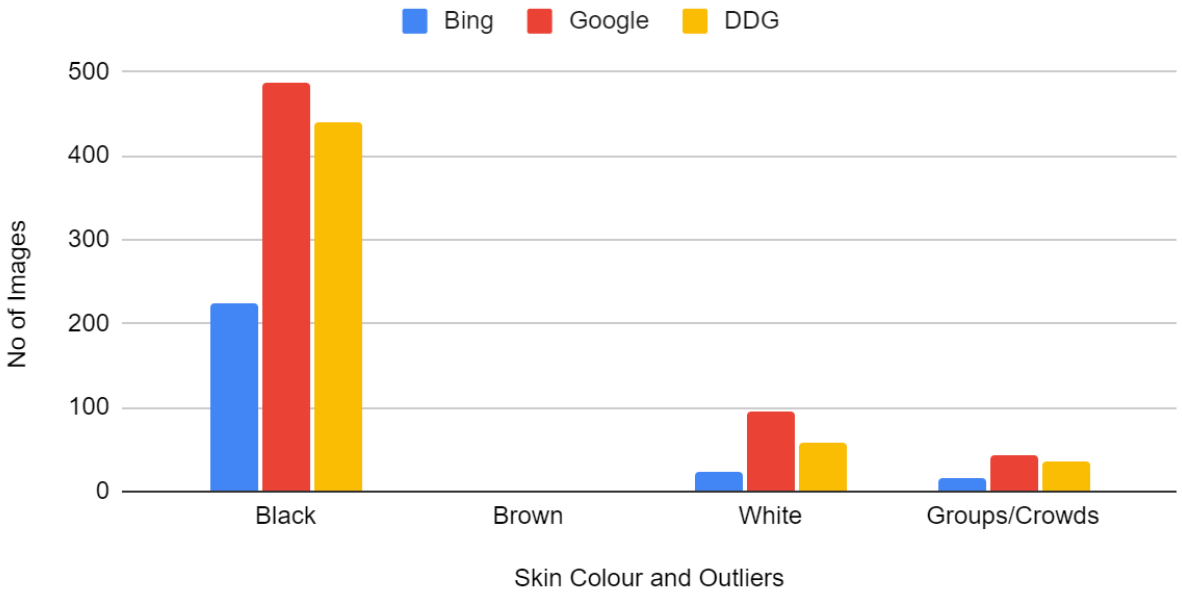
German

(1017 Images)



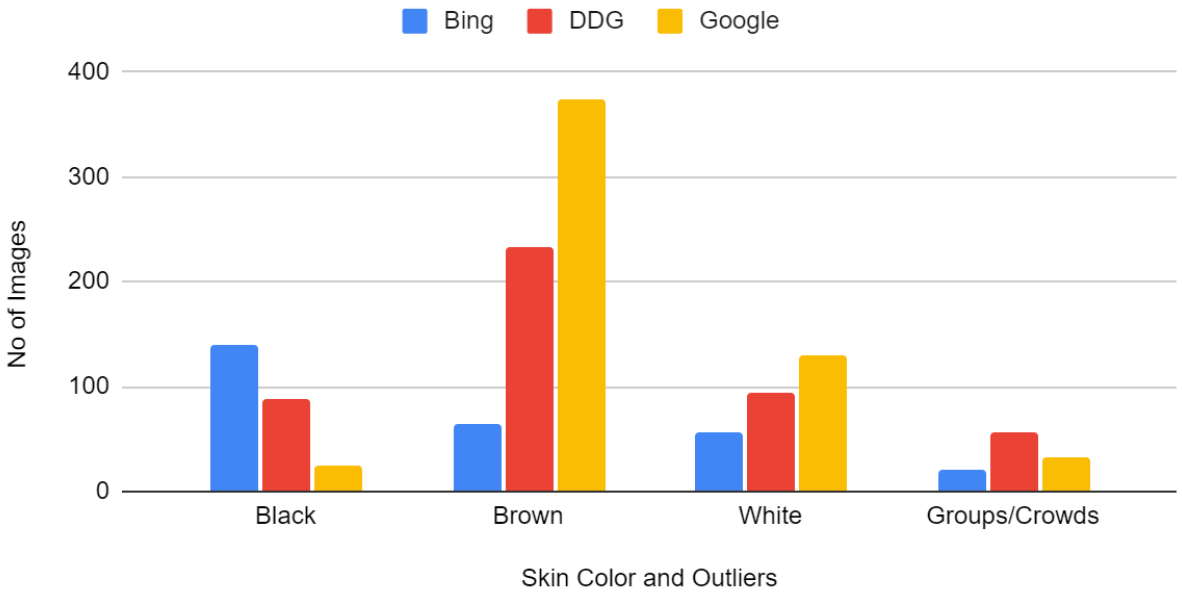
South African

(1430 Images)



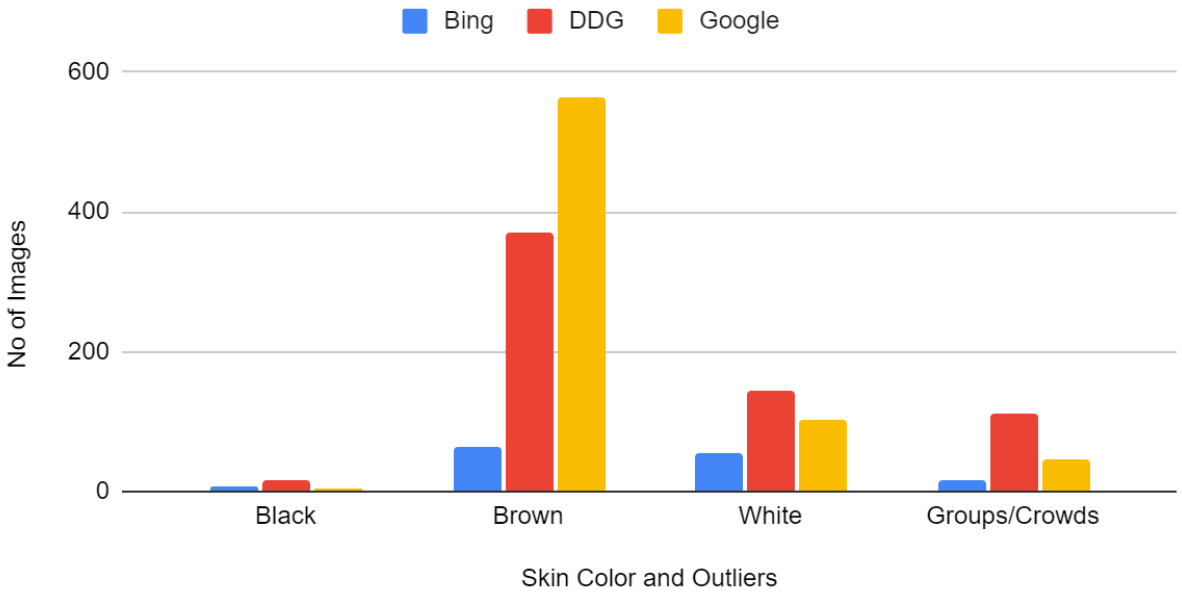
Brazillian

(1311 Images)

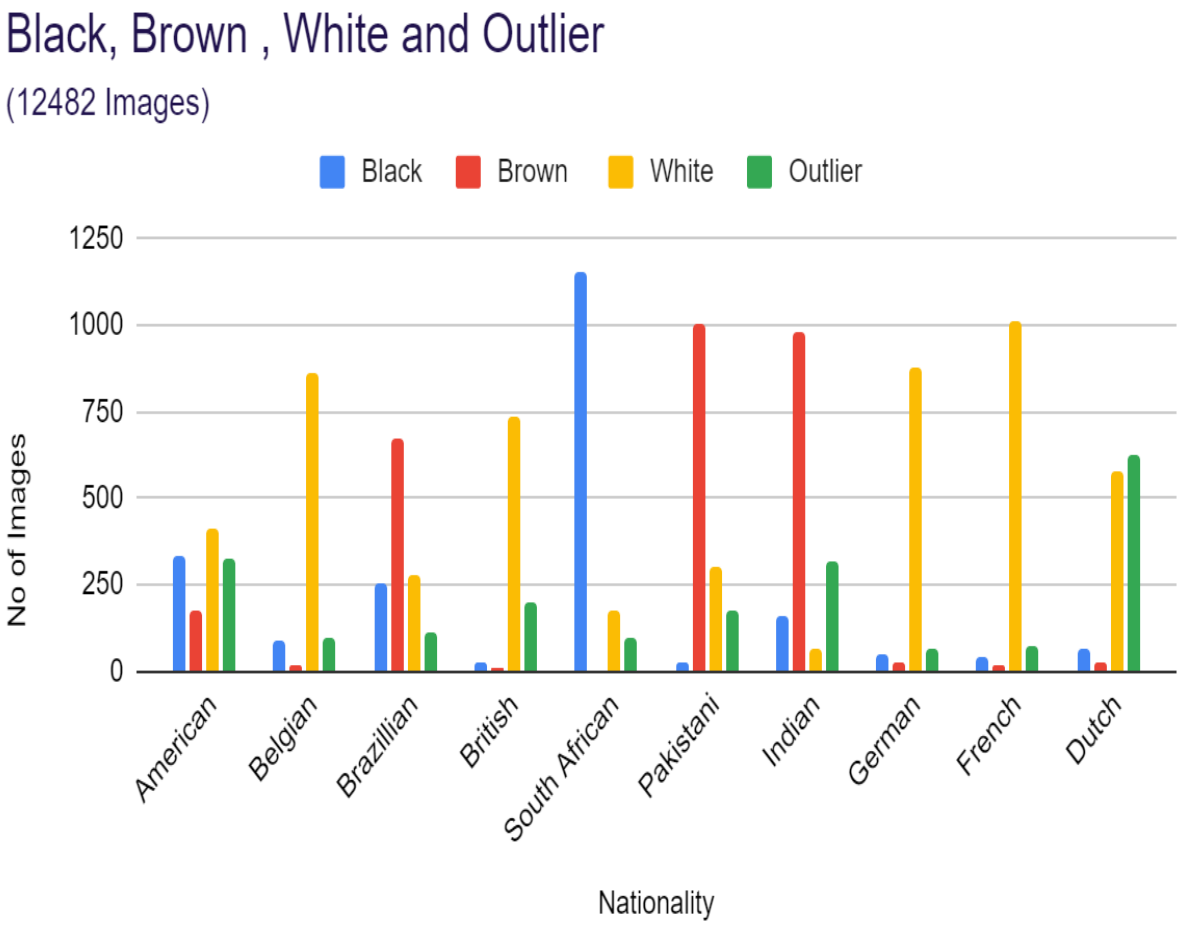


Pakistani

(1502 Images)



After cleaning and labelling the images, the total images per class came out as follows.



In total there were about 12482 images left to create the dataset. A total of 9419 images were deleted (43%). All the images have been uploaded to a Google drive whose link is attached at the end. You can also look at a detailed view of the images per nationality in this spreadsheet. https://docs.google.com/spreadsheets/d/1ikS1cBfs08nlf7GLsrjfyrCx8e9FdeWFU_Tk25Bzmos/edit?usp=sharing

Skin colour Classifier.

Data Split

We used 500 images for each class. An 80:20 percent data split was created for Training and Validation. We used 3 separate folders containing 1500 images that can be found in the “data” folder in the given Google Drive link.

Tools

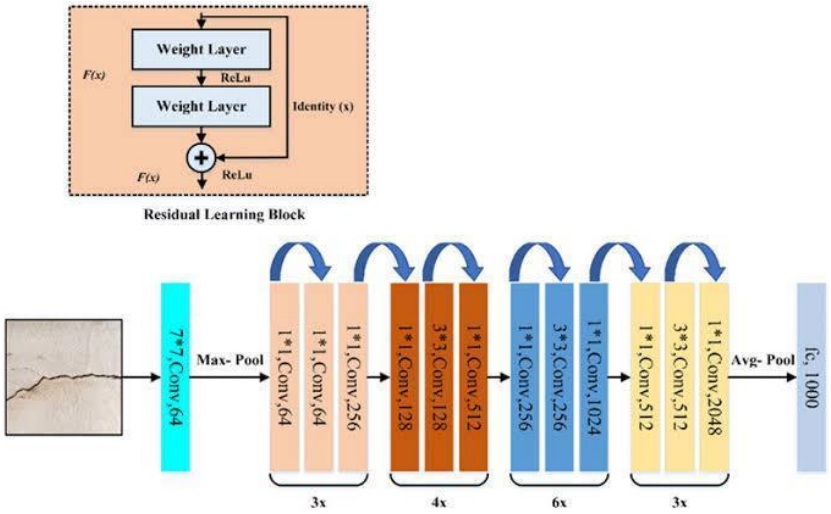
The following frameworks, libraries, and tools were used in this project.

- PyTorch
- Google Colaboratory
- MATLAB
- Numpy

Model

Since we are making an image classifier, the best approach to the problem was to use a convolutional neural network. Upon exploring multiple options, we settled on ResNet-50. The pretrained Resnet model was loaded from the Torchvision library, and used for our skin classification task.

Given below is the Resnet-50 model architecture.



The reasoning behind choosing Resnet was that our data doesn't consist of perfectly framed and cropped human faces (scale and size of faces may vary from image to image). This means that thanks to Resnet's ability to capture and extract features at varying scales (it uses different kernel sizes at different depths), we can hope to achieve good results despite the noise in our data.

Workflow

We mounted the data from our google drive and loaded them into our classes using the default PyTorch dataloader. Some basic transformations were applied to preprocess the images.

```
[ ]  # Data augmentation and normalization for training
     # Just normalization for validation
     data_transforms = {
         'train': transforms.Compose([
             transforms.RandomResizedCrop(224),
             transforms.RandomHorizontalFlip(),
             transforms.ToTensor(),
             transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
         ]),
         'val': transforms.Compose([
             transforms.RandomResizedCrop(224),
             transforms.ToTensor(),
             transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
         ]),
     }

     data_dir = path
     image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                               data_transforms[x])
                       for x in ['train', 'val']}
     dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4,
                                                   shuffle=True, num_workers=2)
                    for x in ['train', 'val']}
     dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
     class_names = image_datasets['train'].classes

     device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
```

Next, we visualized our dataset using the "matplotlib" library. Here are some images and their corresponding labels from our data.

```
[ ] def imshow(inp, title=None):
        """Imshow for Tensor."""
        inp = inp.numpy().transpose((1, 2, 0))
        mean = np.array([0.485, 0.456, 0.406])
        std = np.array([0.229, 0.224, 0.225])
        inp = std * inp + mean
        inp = np.clip(inp, 0, 1)
        plt.imshow(inp)
        if title is not None:
            plt.title(title)
        plt.pause(0.001)  # pause a bit so that plots are updated


    # Get a batch of training data
    inputs, classes = next(iter(dataloaders['train']))

    # Make a grid from batch
    out = torchvision.utils.make_grid(inputs)

    imshow(out, title=[class_names[x] for x in classes])
```

Stochastic gradient descent and sparse categorical loss were used to optimize the model. The next step was to write the training and evaluation functions, and finally call the pre-trained model.

```
[33] model_ft = models.resnet50(pretrained=True)
     num_ftrs = model_ft.fc.in_features
     # Here the size of each output sample is set to 3.
     # Alternatively, it can be generalized to nn.Linear(num_ftrs, len(class_names)).
     model_ft.fc = nn.Linear(num_ftrs, 3)

     model_ft = model_ft.to(device)

     criterion = nn.CrossEntropyLoss()

     # Observe that all parameters are being optimized
     optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

     # Decay LR by a factor of 0.1 every 7 epochs
     exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
```

```
     model_ft = train_model(model_ft, criterion, optimizer_ft, exp_lr_scheduler,
                            num_epochs=25)
```

Results

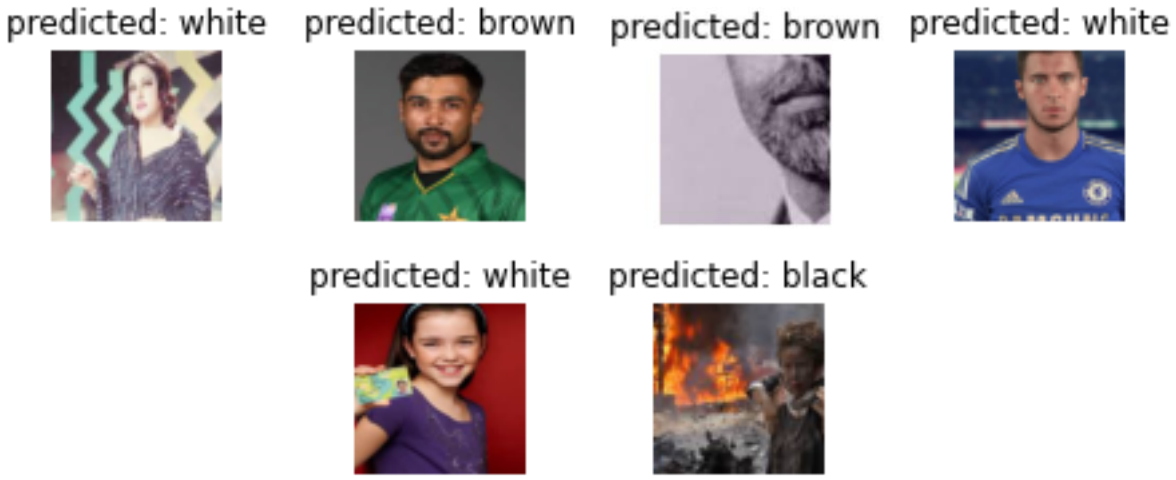
After training the model for a modest 24 epochs, the following are the results we achieved.

```
Epoch 24/24
----------
train Loss: 0.4872 Acc: 0.7990
val Loss: 0.6731 Acc: 0.7433

Training complete in 15m 6s
Best val Acc: 0.746667
```

These can definitely be improved upon further if we improve our dataset and train the model for a greater number of epochs, say 100.

Given below is the model's inference results. As it can be seen, it is performing fairly well at the skin colour classification task.



The predictions for some of our own data are almost always correct. So we believe that our classifier is working accurately

Conclusion

This classifier can be improved by finding better data, and having a standard pre-set size for them such that they are not pixelated. Using a better pre-trained model can help. Increasing the number of epochs can also definitely help.

Brown and Black classes would need more fine-tuned data to be accurately classified. Also, the attached spreadsheet with this project shows the bias in 3 different search engines, we believe that this bias is reflected and demonstrated in the results of our classifier.

Google Drive Link: https://drive.google.com/drive/folders/1QD0JXODPt-gbeg-BJH6OAhrtb7-cPBZ8?usp=sharing