Global Scientific JOURNALS
GSJ: Volume 10, Issue 11, November 2022, Online: ISSN 2320-9186
www.globalscientificjournal.com

Using Machine Learning to Overcome Facial Recognition Bias in Africa.

Sadeeq Musa.

ABSTRACT

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces. An essential aspect of facial recognition technologies is the dataset of faces used for training and testing. However, because most models are trained on mostly Caucasian faces, algorithmic accuracy on Caucasian faces tends to be higher than on African faces. This resulted in facial recognition bias and some unfavorable societal consequences such as false arrest and excessive government surveillance where people of color have been most affected by these consequences. Tribal marks are often neglected in facial recognition and can be used to improve the accuracy of the system. In this paper, we used a one-shot learning model to implement a facial recognition system for African tribal marks. We began by collecting datasets of people with tribal marks and then used Data Augmentation techniques to increase the size and balance of our dataset. An MTCNN model was used to detect and extract faces, and embedding points were generated using a pre-trained model. Using the F1 and MCC scores, we reported scores of 0.887 and 0.757 respectively. This research could be useful in tackling the racial disparity in facial recognition and ensuring that the database against which a face is matched accurately reflects local demographics.

Index Terms: Facial Recognition, Recognition Bias, Tribal Mark, Recognition Algorithm.

1. Introduction

Facial recognition technology is often biased against people of color, which can have devastating consequences in places like Africa where such technology is increasingly being used. In one recent case, a man in Uganda was arrested and jailed for over a month after facial recognition software incorrectly identified him as a suspect in a crime. This kind of technology can also be used to target and persecute marginalized groups, like the LGBTQ+ community. Tribal marks are an African cultural practice that has been around for thousands of years. Different people and tribes use tribal marks for different purposes. Some see it as a thing of beauty, while others see it as a symbol of cultural heritage/belief or kinship identification, and it is used for protection. The main purpose of tribal marks is to identify a person's tribe or family, and it is critical to the people's survival and existence [3]. As a result, facial recognition is far more convenient than other biometrics[19]. However, some challenges to facial recognition affect its accuracy, such as aging, thermal image recognition, occlusion, facial expressions, poses, and facial advances. Given the bias throughout the chain, it is not surprising that statistics show a significant difference in the accuracy of dark and light face recognition in some leading FR providers. When gender is taken into account, this bias becomes even more pronounced. [1]Gendershade's research on gender classification in 2018 found that leading FR providers such as Face ++, Microsoft, and IBM were less accurate on darker faces compared to lighter ones. The worst results were for images of darker females, with a racial accuracy gap of between 21-34% between them and lighter male faces across three major FR vendors. For centuries, parts of

Africa have used facial scarring to identify a person's tribal heritage. Due to intra-subject variations in facial appearance caused by diversity and multiculturalism, several types of research on facial mark recognition have been conducted in recent times for the purpose of inclusion, with the goal of improving facial recognition performance by developing robust feature representation schemes or advanced sensing technologies. We discovered that very little work has been done in the field of computer vision to recognize facial marks associated with certain indigenous African tribes. As the facial recognition models available are trained on mostly caucasian faces, so algorithmic accuracy tends to be better on caucasian faces as compared to African and other high melanin faces. Commonly used benchmarking datasets like Microsoft's Celeb-1M and the academic dataset Labeled Faces in the Wild (LFW) are used by researchers to assess model accuracy despite their inherent bias — Celeb-1M has roughly 14.5% African and African-American faces while LFW is estimated to be 77.5 percent male and 83.5 percent white[2].

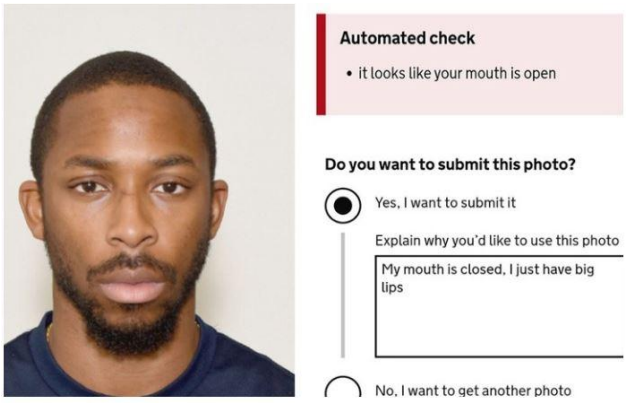


Figure 1: A black man's lips are mistaken for open by a biased passport checker FR system.

Previous Facial Mark Recognition (FMR) systems stated that marks on the face were frequently considered noise and were not explicitly used in the matching process. There has been little research into using facial marks for facial recognition. Using a small twin dataset and a semi-automatic method, Park and Jain demonstrated that facial marks can be used to distinguish identical twins. The majority of research in the field of facial mark recognition has focused on how facial marks can be used to improve the performance of facial recognition systems by identifying tattoos, scars, moles, freckles, acne, and so on [9].

This study aims to develop Deep learning models that can be used to identify African tribal marks.

2. Related Work

S. Sharma et al [4,] proposed a facial recognition system based on a machine learning algorithm that extracted features from an input image using PCA. The dataset was broken down into three configurations: A, B, and C. The dataset was divided 60:40 in A for training and testing, 80:20 in B, and 90:10 in C. Following the extraction of vital features with PCA, different machine learning algorithms (Linear Discriminant Analysis, Multilayer Perceptron, Naive Bayes, and Support Vector Machine) were applied to the different configurations, and C achieved the highest accuracy in each scenario configuration, demonstrating that the more training data there is, the higher the accuracy can be achieved. Soft biometrics, such as facial marks, are small details that can add extra information to facial recognition. [7] proposed incorporating soft biometrics into the appearance-based facial recognition system by fusing traditional facial features that model the facial appearance with soft biometric features that model the micro-expressions in an image sequence. Local Gabor Binary Pattern was used to represent traditional facial features (LGBP) Two feature extraction methods commonly used for video-based micro-expression recognition were used to represent the soft biometrics. The first is Local Binary Patterns (LBP) from Three Orthogonal Planes (LBP-TOP), which is an image sequence extension of Local Binary Patterns (LBP). The Fuzzy Histogram of Optical Flow Orientations (FHOFO) feature, which is an enhanced version of the Histogram of Oriented Optical Flow (HOOF) feature, is used as the second method. Support Vector Machines were used to classify the data (SVM). Several fusion techniques

that are directly applicable to the fusion of traditional and soft biometrics for person recognition were tested during the fusion step. The tested fusion techniques were further classified into feature, rank, and decision levels. The databases used contained multiple image sequences containing facial microexpressions for each individual.

[5] proposed a system for face recognition-based attendance based on machine learning algorithms. The robust DNN base face detector was used for face detection. A variety of images were used to train the pre-trained module. The accuracy of the DNN-based face detection was higher than that of state-of-the-art methods. Face recognition was evaluated using SVM, MLP, and CNN as classifiers and achieved a testing accuracy of 87%, 86.5%, and 98%, respectively.

[6] proposed a scale space analysis-based mark detection method for detecting local extrema in a scale space representation of an input image. The facial landmark detection and masking process are used to avoid detecting local extrema around primary facial features (e.g., eyebrows, eyes, nose, and mouth). The overall mark detection procedure consists of the following steps: I primary facial landmark detection (ASM), (ii) mapping to the mean shape, (iii) mask construction, (iv) scale-space extrema detection on non-masked regions, and (v) post-processing.

[8] improved on [6], which used an Active Appearance Model (AAM) to map and segment primary facial features (e.g., eyes, nose, and mouth). Then, to detect facial marks, Laplacian-of-Gaussian (LoG) and morphological operators are used. According to the study, PCA and LDA were not used to detect micro facial features. The study was able to show that micro-level features like facial marks can provide some discriminating information.

[10] introduced a system known as FaceNet, a system that uses triplet loss to learn a direct mapping from face images to a compact Euclidean space in which distances are directly proportional to a measure of face similarity. Three images (known as anchor, positive, and negative images) are chosen from two classes. They use a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediary bottleneck layer as in previous deep learning techniques. They also claimed that the disadvantages of the previous approach were its indirectness and inefficiency, whereas their method was significantly more efficient in terms of representation.

According to the review of related works, much research has been conducted on ways to improve the robustness of facial recognition systems. Significant research has been conducted on tattoos, scars, moles, freckles, acne, and marks caused by accidents or other illnesses while Tribal marks have been excluded.

3. Methodology

The paper is implemented using a modified version of the FaceNet approach described in [10]. Due to a lack of images of tribal-marked faces, the dataset was compiled from primary sources that included both tribal marked and unmarked faces. We investigated oversampling techniques and used Data Augmentation, which is a set of techniques for increasing the size and quality of training datasets in order to build better Deep Learning models [11]. Our pipeline included the following steps: (a)Data Augmentation, (b)Face detection, (c)Face reorientation and cropping, (d)Face encoding (embedding), and (e)Face classification. We trained and tested an object detection network, landmark detection network, similarity comparison network, and support vector machine (SVM)-based classifier using a programming pipeline of face detection and reorientation, face encoding, and face classification [10]. We provide methodological information for developing the application, as well as preliminary results and annotated source code.

Dataset Acquisition and Processing

This paper's dataset included both primary and secondary data sources. The primary data came from photographers and online social platforms that contained the feature

characteristics (tribal marks), while the secondary data came from publicly available online datasets (Kaggle, and Ethnicity Aware Training Datasets).

The Ethnicity Aware Training Dataset addresses a major bias driver in Face Recognition, which is caused by training data selection. It provides four training datasets (i.e. BUPT-Balanced Face, BUPT-Globalface, BUPT-Transferface, and MS1M wo RFW) for studying facial bias and achieving fair performance[12][13][14][15]. The majority of the datasets are used in deep face recognition networks (i.e. CASIA-WebFace, VGGFace2, and MSCeleb-1M).

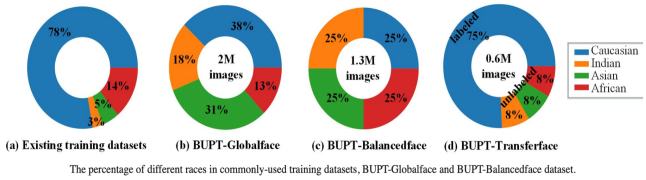


Figure 2 Summary of Ethnicity Aware Training Dataset

Data Preparation

Our dataset was converted from raw data to a format that would allow us to train our model appropriately. To detect and crop out our faces, the multi-task cascaded convolutional network (MTCNN) model was used. The network is made up of three staged cascaded frameworks: the proposal network, the refine network, and the output network [16].

Three tasks are used to train this cascaded CNN: face/non-face classification, facial landmarks localization, and bounding box regression.

- Face classification: the learning goal is referred to as two-class classification probability (face/no-face) for each sample [16].

$$L_I^{det} = -(y_i^{det}\log(p_i) + (1 - y_i^{det})(1 - \log(p_i)))$$

Where:

pi is the probability generated by the neural network indicating whether the image is face or no-face. The ground truth label is represented by *ydet* ϵ {0,1}.

- Bounding box regression: The learning objective here is referred to as a regression problem, in which the prediction of the offset between each candidate window and the nearest ground truth is computed. The bounding box's dimensions are left top, height, and width [16].

$$l_i^{box} = \| \bar{y}_i^{box} - y_i^{box} \|_2^2$$

Where:

$\bar{y}_i^{box}$ the logit regression obtained from the network

y_i^{box} represent the coordinates of the ground truth.

- Facial landmark localization: is a regression problem, which is similar to a bounding box regression. The Euclidean loss is denoted by:

$$l_i^{landmark} = \| \bar{y}_i^{landmark} - y_i^{landmark} \|_2^2$$

Data Augmentation

Data augmentation: s an overfitting technique used to address the most commonly reported issue in machine learning: a lack of sufficient training data or an uneven class balance within datasets[11]. The parameters used for our data augmentation are detailed in the table below.

Table 1: Data Augmentation Parameters

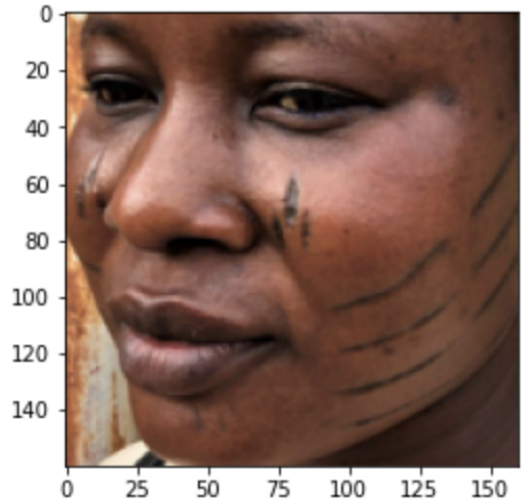
Method/Parameters	Value
Rotation range	40
Shear range	0.2
Zoom range	0.2
Horizontal Flip	True
Brightness range	0.5, 1.5

Training and Validation

We divided our data into two sets: training and testing, and then into two classes: facial marked and unmarked. The training data set contains 591 images of unmarked faces and 936 images of facially marked faces, while the testing data set contains 76 images of unmarked faces and 189

images of facially marked faces. Both sets were run through a face detection model, and then a method extracted the faces and returned them in the 160 * 160-pixel shape required by the trained FaceNet model. Faces and labels from both the training and testing datasets are then saved in a list X and y. After that, the face dataset is compressed and saved for later use.

Figure 3: Loaded Face




```
(160, 160, 3)
```

Encoding Faces Using FaceNet

FaceNet, a deep learning architecture consisting of convolutional layers based on GoogleNet-inspired inception models, was used for face recognition and clustering. Facenet returns a 128-dimensional vector embedding for each face. Once these embeddings are created, procedures such as face recognition and verification can be performed using these embeddings as face features[20]. These are the distinct facial characteristics (scars, moles, facial marks). Following the creation of the embedding points, the system classifies the result and compares it to training datasets.

Confusion Matrix

This matrix is one of the most intuitive and descriptive metrics for determining the accuracy and correctness of a machine learning algorithm. Its main use is in classification problems with two or more types of classes in the output [17]. It is depicted as a matrix and provides a visual representation of the actual vs. expected numbers.

Precision

It simply shows "how many of the selected data items are relevant." In other words, how many of the observations predicted by an algorithm to be positive are actually positive? The precision is equal to the number of true positives divided by the sum of true positives and false positives [17]:

$$\frac{TP}{TP+FP}$$

Recall or Sensitivity

Recall or Sensitivity is the Ratio of true positives to total (actual) positives in the data. The recall is equal to the number of true positives divided by the sum of true positives and false negatives, as calculated below:

$$: \frac{TP}{TP + FN}$$

Accuracy

It is the most commonly used for evaluating the performance of an algorithm in classification problems. It is defined as the proportion of correctly classified data items to a total number of observations (formula below). Despite its widespread use, accuracy is not always the best performance metric, especially when the target variable classes in the dataset are unbalanced.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

F1 Score

This metric, also known as an f-score or an f-measure, calculates an algorithm's performance by taking precision and recall into account. It is the harmonic mean of

precision and recall, which is defined mathematically as follows:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Matthews Correlation Coefficient

The Matthews correlation coefficient (MCC) is a more reliable statistical rate that yields a high score only if the prediction performed well in all four confusion matrices categories (true positives, false negatives, true negatives, and false positives), proportionally to the size of positive and negative elements in the dataset.

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

4. Data Implementation.

Due to insufficient data in one of our training classes, we decided to augment the tribal mark class for our model to have better generalization, which will directly affect our model's accuracy to the unseen datasets. The implementation for our data augmentation is shown in the code below.

Figure 4: Source code of Data Augmentation

```
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
datagen = ImageDataGenerator(
        rotation_range = 40,
        zoom_range = 0.2,
        horizontal_flip = True,
        brightness_range = (0.5, 1.5))
import numpy as np
import os
from PIL import Image
image_directory = r'/content/drive/MyDrive/Tribalmarks/Dataset/try/'
SIZE = 400
dataset = []
my_images = os.listdir(image_directory)
faces = list()
for filename in os.listdir(image_directory):
  path = image_directory + filename
  faces.append(path)
for i in faces:
    image = Image.open(i)
    image = image.convert('RGB')
    image = image.resize((SIZE,SIZE))
    dataset.append(np.array(image))
x = np.array(dataset)
i = 0
for batch in datagen.flow(x, batch_size=16,
                          save_to_dir= r'/content/drive/MyDrive/Tribalmarks/Dataset/output/do/',
                          save_prefix='augmented',
                          save_format='jpeg'):
    i += 1
    if i > 5:
        break
```

Data Preprocessing

We divided our dataset into two classes: the first, facial marks, contains all images with tribal marks, and the second, unmarked, contains all images of people of black ethnicity with no tribal marks on their faces. The dataset had to be further preprocessed in order for our model to extract the important facial features required for training easily. We used a multi-task cascaded convolutional neural network (MTCNN) to detect and draw a bounding box around the face of our identities, cropping it out and saving it in a folder for use in training our model [21].

```
from PIL import Image
from matplotlib import pyplot
from numpy import savez_compressed
from numpy import asarray
from mtcnn.mtcnn import MTCNN
from keras.preprocessing.image import ImageDataGenerator

# extract a single face from a given photograph
def extract_face(filename, required_size=(160, 160)):
  image = Image.open(filename)
  image = image.convert('RGB')
  print(filename)
  pixels = asarray(image)
  detector = MTCNN()
  results = detector.detect_faces(pixels)
  x1, y1, width, height = results[0]['box']
  x1, y1 = abs(x1), abs(y1)
  x2, y2 = x1 + width, y1 + height
  face = pixels[y1:y2, x1:x2]
  image = Image.fromarray(face)
  image = image.resize(required_size)
```

```
# load images and extract faces for all images in a directory
def load_faces(directory):
  faces = list()
  for filename in listdir(directory):
    path = directory + filename
    face = extract_face(path)
    faces.append(face)
  return faces

# load a dataset
def load_dataset(directory):
  X, y = list(), list()
  for subdir in listdir(directory):
    path = directory + subdir + '/'
    if not isdir(path):
      continue
    faces = load_faces(path)
    labels = [subdir for _ in range(len(faces))]
    print('>loaded %d examples for class: %s' % (len(faces), subdir))
    X.extend(faces)
    y.extend(labels)
  return asarray(X), asarray(y)
```

```
# load train dataset
trainX, trainy = load_dataset('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/training/')
print(trainX.shape, trainy.shape)
```

```
# load test dataset
testX, testy = load_dataset('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/testing/')

print(testX.shape, testy.shape)
np.savez_compressed('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/data.npz', trainX, trainy, testX, testy)
```

Figure 5: Source code of Data Augmentation

Face Net

We load the entire face dataset into the Facenet model, convert all of the faces in the train and test sets into embeddings, and then place them in arrays. We compress the arrays and save them to a single file.

```
from numpy import load
from numpy import expand_dims
from numpy import asarray
from numpy import savez_compressed
from keras.models import load_model

# get the face embedding for one face
def get_embedding(model, face_pixels):
  face = face_pixels.astype('float32')
  mean, std = face_pixels.mean(), face_pixels.std()
  face_pixels = (face_pixels - mean) / std
  samples = expand_dims(face_pixels, axis=0)
  yhat = model.predict(samples)
  return yhat[0]

# load the face dataset
data = load('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/data.npz', allow_pickle = True)
trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)
# load the facenet model
model = load_model('/content/drive/MyDrive/Tribalmarks/Dataset/facenet_keras.h5', compile = False)
print('Loaded Model')
# convert each face in the train set to an embedding
newTrainX = list()
for face_pixels in trainX:
  embedding = get_embedding(model, face_pixels)
  newTrainX.append(embedding)
newTrainX = asarray(newTrainX)
print(newTrainX.shape)
# convert each face in the test set to an embedding
newTestX = list()
for face_pixels in testX:
  embedding = get_embedding(model, face_pixels)
  newTestX.append(embedding)
newTestX = asarray(newTestX)
print(newTestX.shape)
savez_compressed('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/faces-embeddings.npz', newTrainX, trainy, newTestX, testy)
```

Figure 6: Source code of Face embedding

Hyper-parameters Tuning

A machine learning model requires a number of parameters to be learned directly from data. Finding the optimal hyper-parameter is a difficult task, but it can be accomplished by attempting all combinations and observing which parameters work best. GridSearchCV is a Cross Validation technique for improving prediction/accuracy results.

```
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(trainX, trainy)
```

Figure 7: Source code for Parameter Tuning.

Stratified K Fold Cross Validation

Cross-validation is a data resampling method used to assess predictive model generalization and prevent overfitting. Stratified k-fold cross-validation is a variation on k-fold cross-validation that uses stratified sampling rather than random sampling.

```
from numpy import load
from numpy import expand_dims
from numpy import asarray
from numpy import savez_compressed
from keras.models import load_model

# get the face embedding for one face
def get_embedding(model, face_pixels):
  face = face_pixels.astype('float32')
  mean, std = face_pixels.mean(), face_pixels.std()
  face_pixels = (face_pixels - mean) / std
  samples = expand_dims(face_pixels, axis=0)
  yhat = model.predict(samples)
  return yhat[0]

# load the face dataset
data = load('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/data.npz', allow_pickle = True)
trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)
# load the facenet model
model = load_model('/content/drive/MyDrive/Tribalmarks/Dataset/facenet_keras.h5', compile = False)
print('Loaded Model')
# convert each face in the train set to an embedding
newTrainX = list()
for face_pixels in trainX:
  embedding = get_embedding(model, face_pixels)
  newTrainX.append(embedding)
newTrainX = asarray(newTrainX)
print(newTrainX.shape)
# convert each face in the test set to an embedding
newTestX = list()
for face_pixels in testX:
  embedding = get_embedding(model, face_pixels)
  newTestX.append(embedding)
newTestX = asarray(newTestX)
print(newTestX.shape)
savez_compressed('/content/drive/MyDrive/Tribalmarks/Dataset/tribal marks/faces-embeddings.npz', newTrainX, trainy, newTestX, testy)
```

Figure 7: Source code of Cross Validation.

Data Preprocessing

The MTCNN was used to detect and crop the identity face's bounding box.

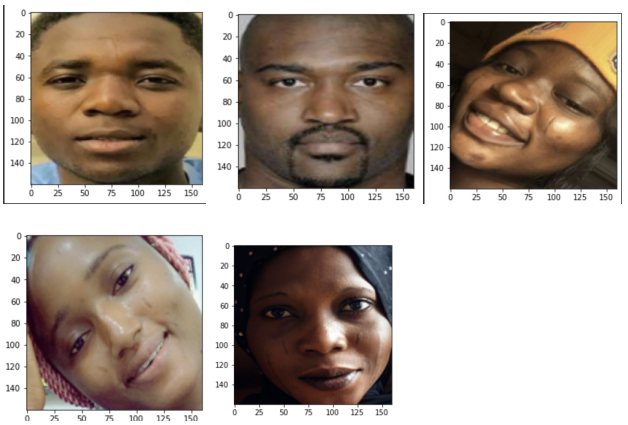


Figure 8: Preprocessed data

Classification and Evaluation

Based on binary classification, we created two classes: facial marked faces and unmarked faces. We achieved an accuracy of 97% for training and 71% for testing after training our model with 878 training datasets (425 facial marked images and 453 unmarked images) and 157 testing datasets (78 facial marked images and 79 unmarked images). We obtained an accuracy of 81% and 80% for

both training and testing when we increased the unmarked images in the training dataset from 453 to 1200 images and from 79 to 125 for the testing set, which we imported from the Ethnicity Aware Training Dataset. To mitigate the risk of model overfitting, we decided to use oversampling techniques on the marked images, which will increase the sample number by balancing the dataset. After performing data augmentation on the marked images in the training dataset, we had 844 marked images and 655 unmarked images, totaling 1499 images in our training dataset and 399 images in our testing dataset, indicating underfitting. The next step was to fine-tune our SVM model's hyperparameters, after which we achieved the desired accuracy of 100% for training and 88% for testing. We used other evaluation metrics to ensure that there was no Overfitting. A confusion matrix was created to provide detailed information about the classification processes in our two classes (facial marked and unmarked).

	TP	FN
0	214	20
1	34	177
	FP	TN

0 for Facial marked and 1 for unmarked

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{214}{214 + 34} = 0.862$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{214}{214 + 20} = 0.914$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{214+177}{214+177+20+34} = 0.879$$

$$\text{F1 Score} = \frac{2 * \text{Precision} + \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 * 0.862 * 0.914}{0.862 + 0.914} = 0.887$$

The positive class is facial marked, while the negative class is unmarked. We obtained a precision score of 0.862, indicating that a large proportion of the images predicted by the ML model as facial marked images were actually facial marked images. We also achieved a recall of 0.914, which indicated the percentage of actual facial marked images predicted correctly by our ML model out of all the facial marked images submitted to the model.

In an ideal world, our model would have perfect precision and recall. In practice, however, there is frequently a tradeoff between the two. The tradeoff is deciding which is more important (false positive or false negative). Given this tradeoff, it would be very convenient to have a single performance metric that takes precision and recall into account. We then use another metric known as the F1 score, which is calculated by taking the harmonic mean of the two metrics. We got an F1 score of 0.887.

However, Accuracy and F1 scores, while popular, can produce misleading results on imbalanced datasets because they ignore the ratio of positive to negative elements. Matthews correlation coefficient (MCC) can solve this problem due to its mathematical properties that incorporate dataset imbalance and its class swapping invariance[18].

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} = 0.757$$

An MCC score of 0.757 was obtained, which is close to one, indicating that both classes were well-predicted.

We also evaluated our model by performing cross validation on our training data and represented it in the figure below as a learning curve.

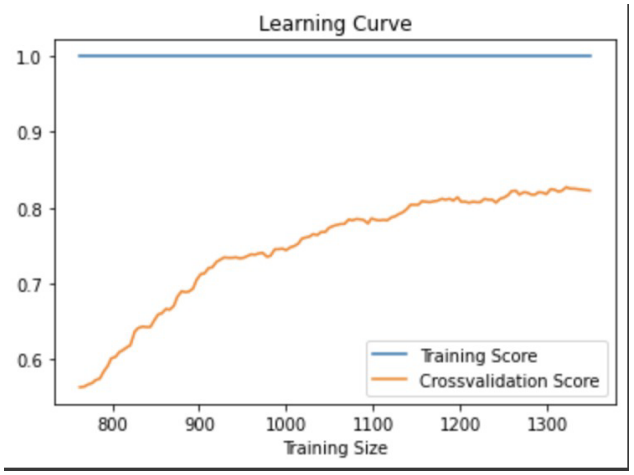


Fig 9: Learning curve of our model

This indicated low bias and high variances, indicating the possibility that increasing the dataset will help improve the model's performance. We also performed a predictive analysis on our full dataset using 10 splits, with a maximum accuracy of 87% and a minimum accuracy of 79%. When we fitted in our test data, we achieved an accuracy of 88%, which was higher than the maximum.

5. Conclusions and Future Work

This study demonstrated that tribal mark identification can be used to improve the accuracy of facial recognition systems and thus be used to overcome facial recognition bias, with our model achieving an accuracy of 0.879 and an F1 Score of 0.887. We also evaluated our model using the Matthews correlation coefficient (MCC), which produced a score of 0.757, to produce a more informative and truthful score in evaluating binary classifications than accuracy and F1 score. We also ran cross validation on our training dataset and generated a learning curve with low bias and high bias, indicating that increasing the dataset size may yield better results.

Soft biometrics were previously regarded as image noise and were not explicitly used in matching processes. Recent research has shown that soft biometrics such as moles, scars, freckles, and so on can be used to improve the performance of facial recognition systems. Based on the previous Facial mark recognition model, we proposed a new tribal mark recognition system that can identify and classify images based on tribal marks using one shot learning architecture. This research would be useful in tackling the racial disparity in facial recognition as well as in ensuring that the database against which a face is matched accurately reflects local demographics.

References

[1] Joy Buolamwini, and Timnit Gebru. “Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification” (2018)

[2] Davy Uwizera. “Beating Facial Recognition Bias in Africa.” SmileIdentity (2020).

[3] Alo, Grace. "Origin, Types and Cultural Significance of Tribal Marks Amongst The Yoruba Tribe." *NIGERIAN JOURNAL OF DERMATOLOGY* 8.2 (2018).

[4] S. Sharma, M. Bhatt and P. Sharma, "Face Recognition System Using Machine Learning Algorithm," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 1162-1168, doi: 10.1109/ICCES48766.2020.9137850.

[5] R. C. Damale and B. V. Pathak, "Face Recognition Based Attendance System Using Machine Learning Algorithms," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 414-419, doi: 10.1109/ICCONS.2018.8662938.

[6] Park, Unsang, et al. "Face finder: Filtering a large face database using scars, marks and tattoos." Michigan State Univ., Tech. Rep. TR11 (2011).

[7] Saeed, Usman. "Facial micro-expressions as a soft biometric for person recognition." *Pattern Recognition Letters* 143 (2021): 95-103.

[8]Park, Unsang, and Anil K. Jain. "Face matching and retrieval using soft biometrics." IEEE Transactions on Information Forensics and Security 5.3 (2010): 406-415.

[9] Becerra-Riera, Fabiola, Annette Morales-González, and Heydi Méndez-Vázquez. "Facial marks for improving face recognition." Pattern Recognition Letters 113 (2018): 3-9.

[10] Schroff, F. K. (2015). FaceNet: A unified embedding for face recognition and clustering. IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 815-823.

[11] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." *Journal of big data* 6.1 (2019): 1-48.

[12] Mei Wang, Weihong Deng, Jiani Hu, Xunqiang Tao, Yaohai Huang. Racial Faces in the Wild: Reducing Racial Bias by Information Maximization Adaptation Network. ICCV2019.

[13] Mei Wang, Yaobin Zhang, Weihong Deng. Meta Balanced Network for Fair Face Recognition. TPAMI 2021.

[14] Mei Wang, Weihong Deng. Mitigating Bias in Face Recognition using Skewness-Aware Reinforcement Learning. CVPR2020.

[15] Mei Wang, Weihong Deng. Deep face recognition: A Survey. Neurocomputing.
[16] Kaipeng, Z. Z. (2016). Joint Face Detection and Alignment using Multi task Cascaded Convolutional Networks. IEEE, 2-3.
[17]Vakili, M. &. (2020). Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification.
[18]Chicco, Davide, and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation." *BMC genomics* 21.1 (2020): 1-13.
[19] Sabharwal, T., Gupta, R. Facial marks for enhancing facial recognition after plastic surgery. Int. j. inf. tecnol. 13, 391–396 (2021). https://doi.org/10.1007/s41870-020-00566-x
[20] Riaz, Sidra, Unsang Park, and Prem Natarajan. "Improving face verification using facial marks and deep CNN: IARPA Janus benchmark-A." Image and Vision Computing 104 (2020): 104020.
[21] Bin Jiang, Qiang Ren, Fei Dai, Jian Xiong, Jie Yang, and Guan Gui. “Multi-task Cascaded Convolutional Neural Networks for Real-Time Dynamic Face Recognition Method.” Part of the Lecture Notes in Electrical Engineering book series (LNEE,volume 517).