ME292b Statistics and Data Science for Engineers

Ethnicity Recognition Using Convolutional Neural Network and Transfer Learning

Submitted By:

Jayant Kumar **Pavan M. Reddy**

Joshua Duarte **Miguel Montabalan**

Date: 05-09-2023

Table of Contents

1. Introduction

Neural networks have become an increasingly relevant tool for predicting various demographic characteristics. With the rise of big data and machine learning, users are exploring new ways to leverage these powerful algorithms to make predictions about individuals based on their data. In this project, we would be using it to make ethnicity predictions.

Neural networks can be trained on large datasets to identify patterns and correlations between an individual's data and their ethnicity. This can be done using a variety of data sources, such as facial images, genetic data, or even text data from social media profiles.

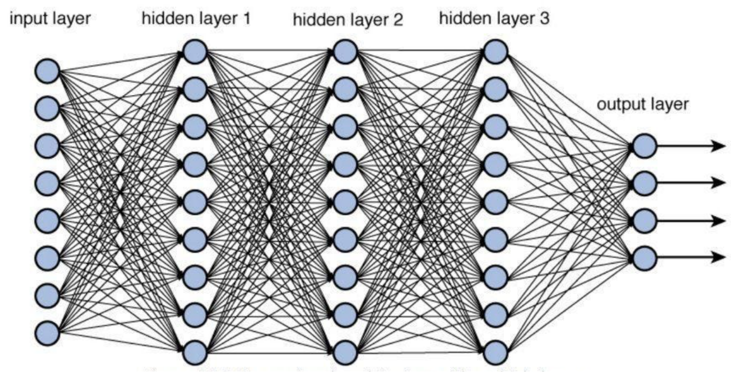


Fig 1: Deep Neural Network

One of the key benefits of using neural networks for ethnicity prediction is their ability to handle complex data with many variables. Traditional statistical methods may struggle to account for all of the factors that influence ethnicity, such as ancestry, cultural background, and environmental factors. Neural networks, on the other hand, can identify patterns and relationships in large datasets that might be difficult for humans to detect.

Using neural networks for ethnicity prediction raises ethical and privacy concerns. As with any algorithmic decision-making system, there is a risk of bias and discrimination if the training data is not representative or if the algorithm is not designed and tested carefully. However, with proper safeguards and ethical guidelines, neural networks can be a valuable tool for understanding and predicting ethnicity.

In this project, we will explore how various neural network models are used to help predict ethnicity while keeping in check the ethical concerns pertaining to the test cases. Here we are providing a clear process that was followed to understand and learn to use different pre-trained neural networks to achieve acceptable accuracies while showcasing the bottlenecks we hit in the process.

2. Background

Ethnicity is a complex social construct that is defined by a combination of factors, including ancestry, culture, language, and geography. It is an important aspect of identity for many people and has been used to explain differences in health outcomes, education, and economic opportunities. Predicting ethnicity accurately can be challenging, but it has important applications in a variety of fields, including medicine, law enforcement, and market research.

In the past, a variety of methods have been used to predict ethnicity, each with its own advantages and limitations. Facial recognition, for example, has been used to identify features that are associated with specific ethnic groups, such as eye shape/color or skin tone. However, this method can be limited by factors such as lighting, image quality, and individual variation in facial features.

DNA testing is another method that has been used to predict ethnicity, which is done by analyzing genetic markers that are associated with different ethnic groups. While DNA testing can be very accurate, it can also be expensive and time-consuming, and may not be feasible for large-scale studies or routine use in medical settings.

Surname analysis is yet another method that has been used to predict ethnicity, by identifying the origin and meaning of surnames that are associated with specific ethnic groups. However, this method can be limited by factors such as immigration, intermarriage, and changing social norms around naming conventions.

By using machine learning algorithms to analyze large datasets of information, neural networks offer a promising alternative for predicting ethnicity. These networks are designed to recognize patterns and relationships in data. They can be trained on a variety of inputs, such as facial images, DNA data, or text-based data such as names or addresses. With the right training data and algorithms, neural networks can be highly accurate in predicting ethnicity. They can be used in a variety of applications, such as personalized medicine or targeted advertising. However, like any tool, neural networks must be used carefully and ethically, to ensure that they do not perpetuate bias or discrimination based on race or ethnicity.

2.1. Ethics behind machine learning models

The development of a machine learning model such as this one can raise several controversial ethical considerations and challenge connected to the idea of race and the blurriness of ethnic boundaries. A possible concern that may arise is the misuse of this type of tool to discriminate.

Moreover, the accuracy of the model can be influenced abruptly by the diversity of the dataset. Even though we have tried to avoid this issue, as we explain in previous sections. The diversity factor could still lead to misclassifications and reinforce existing stereotypes or biases already existing in society, which could perpetuate discrimination and marginalization. Additionally, using ethnicity prediction models like this one can bring back the relevance of the concept of race itself. The blurriness of ethnic boundaries can pose challenges to accurate classification and ethics in technology.

For all these reasons, it is important to keep in mind the ethical implications of using machine learning of this kind and ensure that the technology is utilized responsibly and transparently. To account for this it's important to collaborate between researchers, politicians, and other stakeholders in the event of using this technology in public settings, establishing guidelines for the ethical use of AI systems in ethnic identification and others.

3. Theory

A simplified yet intelligent model that teaches computers to process the data like a human brain is denominated neural network. There are three types of neural networks namely, Artificial Neural Networks (ANN) Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN). In this project, we will be mainly focusing on one of the many available neural networks, the convolution neural network (CNN).

A Convolutional Neural Network (CNN) is a type of computing system that is well suited to analyzing pictures and understanding what's in them. It works by

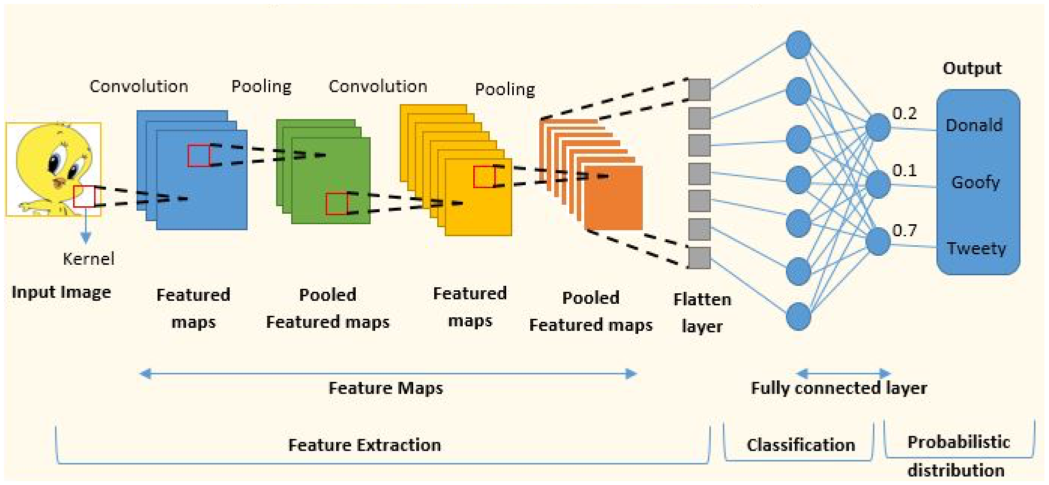


Fig 2: Convolution neural network (CNN)

breaking down the picture into smaller pieces, looking for patterns in each piece, and then using those patterns to determine what's in the picture as a whole. This technology is used in many applications, such as self-driving cars, facial recognition, and image classification.

Some of the commonly available and used deep learning CNNs are VGG16, VGG19, and VGGface for image detection training purposes. These are pre-trained neural network models that help us in solving new problems without having to teach them from scratch. This is a commonly used method called transfer learning. With the help of the knowledge the models have acquired from prior assignments the accuracy of predictions is increased and the burden of training them again with large datasets is eliminated.

3.1. VGG

Visual Geometry Group, VGG is a deep convolution neural network architecture model with multiple layers that helps in object recognition. VGG16 is one of the most popular image recognition architectures that is used and the number 16 refers to the number of layers the model has i.e., 16 layers in total, including 13 convolutional layers and 3 fully connected layers. The convolutional layers are designed to extract high-level features from input images, while the fully connected layers act as a classifier to assign labels to the images based on those features.

VGG-16 is known for its simplicity and effectiveness in achieving state-of-the-art performance on benchmark datasets like ImageNet, which contains millions of labeled images. Its success has paved the way for the development of deeper and more complex CNN architectures, such as VGG-19.

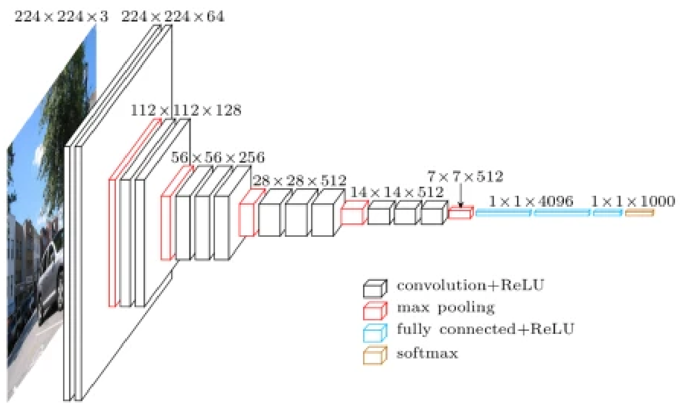
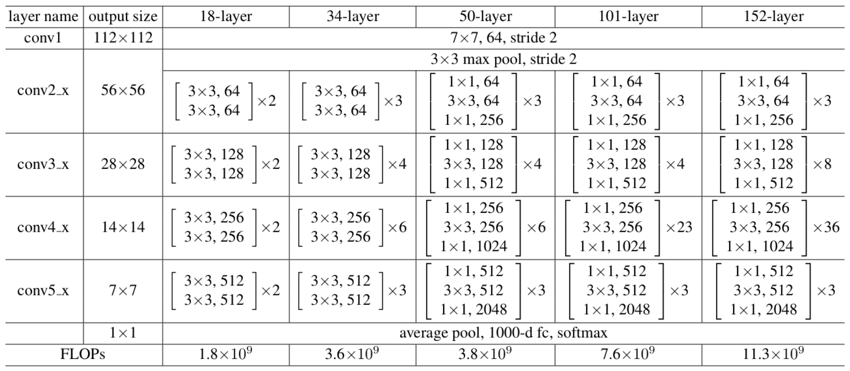


Fig 3: Visual Geometry Group

3.2. ResNet

ResNet stands for Residual Network. ResNet has many variants that run on the same concept but have different numbers of layers. ResNet50 is a variant of the ResNet model which has 48 Convolution layers along with 1 MaxPool and

1 Average Pool layer. It is a widely used ResNet model and we have explored ResNet50 architecture in depth.



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 4: ResNet50 architecture

3.3. VGG-Face:

VGG-Face is a 16-layer deep convolutional neural network that belongs to the VGG, Visual Geometry Group, that was described above. Unlike VGG-16, and VGG-19, which are used for normal image recognition, VGG-Face is designed particularly for face recognition tasks. In its creation, VGG-Face is trained on a dataset that contains 2.6 million facial images. This large dataset allows for the ease of recognition and interpretation of facial features regardless of illumination or orientation.

4. Process

To build a neural network for ethnicity prediction, we followed a standard process of data collection, preprocessing, model development, training, and testing. The specific steps we took are described below.

4.1. Data Collection

The most important and time-consuming step in every neural network model building is data collection and preprocessing them. Having the right data has a huge effect on how accurate and easy it is to train the model. In order to find datasets that were suitable for our application, we started researching Kaggle where we found several datasets with face images, Celebface attributes (Celeb A) were initially considered. This is composed of 202,599 face images of various celebrities, and several labels regarding face attributes like face shape and hair color. However, this dataset had two issues; first, no label regarding ethnicity was included, also we discovered that the diversity of the data was very reduced. As we see in the following graph from (GitHub, n.d.) this dataset and many common others for face recognition are biased with a large percentage of data from Caucasian faces.

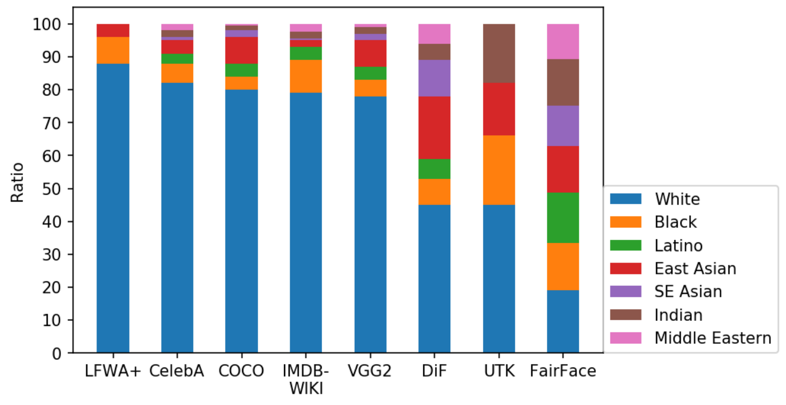


Fig 5: Racial compositions in face datasets

After this, we realized that if we want to account for a wide amount of ethnicity we needed to use one more discrete dataset such as DiF, UTK, or FairFace. We decided to use FairFace, for its ethnic diversity but also because it included a race label that we were interested in. The fair face dataset is available from a GitHub repository (GitHub, n.d.) that had previously worked on a similar project. This dataset had the following content:

- 80 000.jpg Training Images
- 20 000.jpg Validating Images
- ethnicity.cvs file with Labels for each Training image
- ethnicity_val.cvs file with Labels for each Validation image

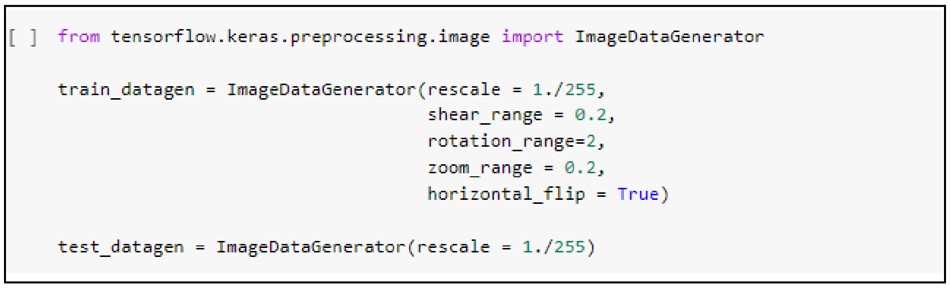
The labels included in this dataset are age, gender, and race, all of them being strings characterizing each label. Now the next step was to organize the data that's being explained next.

4.2. Data Preprocessing

Before training the neural network, we preprocessed the facial images to prepare them for analysis. Starting off with understanding the features available to us and that were going to be used we sorted the spreadsheet with the data. With the data that was available to us, we had a total of seven ethnic groups and 100000 images. Now the next step was to make sure the data we had was in the right size, orientation, and contrast levels so as to reduce any discrepancies from them. As we had a plan of using the pre-trained VGG model we had the data images pixelated to 224x224. Next, once the data was standardized we generated a Python script that would help us to randomly sort the data into different ethnic groups equally adding up to the required number of training and validation datasets.

4.3. Image Data Augmentation

Image data augmentation is a technique to increase the training dataset for making the dataset diverse and result in robust classification predictions. Image data augmentation was performed using the ImageDataGenerator module associated with tensorflow & keras. We incorporated a couple of transformations on the training dataset including rotation, horizontal flip, shear and zoom as highlighted in the image below.



```
[ ]  from tensorflow.keras.preprocessing.image import ImageDataGenerator

     train_datagen = ImageDataGenerator(rescale = 1./255,
                                        shear_range = 0.2,
                                        rotation_range=2,
                                        zoom_range = 0.2,
                                        horizontal_flip = True)

     test_datagen = ImageDataGenerator(rescale = 1./255)
```

Fig. 6: Results for VGG16 after 10 epochs

4.4. GPU Configuration

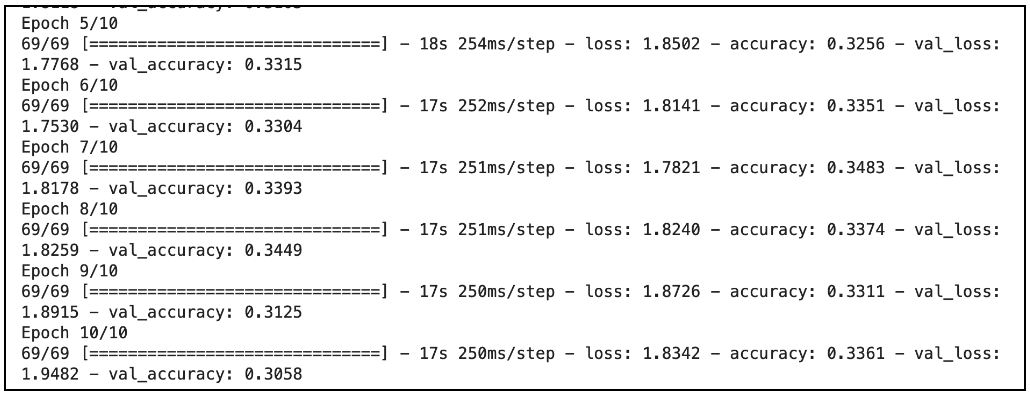
With the initial training planned to be done using a limited number of images, we used google collab. Up to a limit of 3000 images, the system gave results with acceptable time frames. But as the data set increased the time needed increased significantly in collab. So we switched to the desktop but the kernel died every time we try running it on Jupyter. Hence, we spent time trying to configure the GPU on the desktop but we failed every time so we switched to Mac to train the data. With GPU configured on the M1, it took us little to no time to train the same amounts.

4.5. Model Development and Training

During the development phase of our ethnicity detection model, we utilized state-of-the-art techniques in computer vision, specifically, Convolutional Neural Networks (CNNs), to train on a subset of facial images and analyze facial images and predict ethnicity. Our training process incorporated the use of backpropagation, a technique that adjusts the weights and biases of the neurons in the network based on the difference between the predicted and actual ethnicity of each sample. The architecture of the CNN was designed to optimize accuracy while minimizing the risk of overfitting the data.

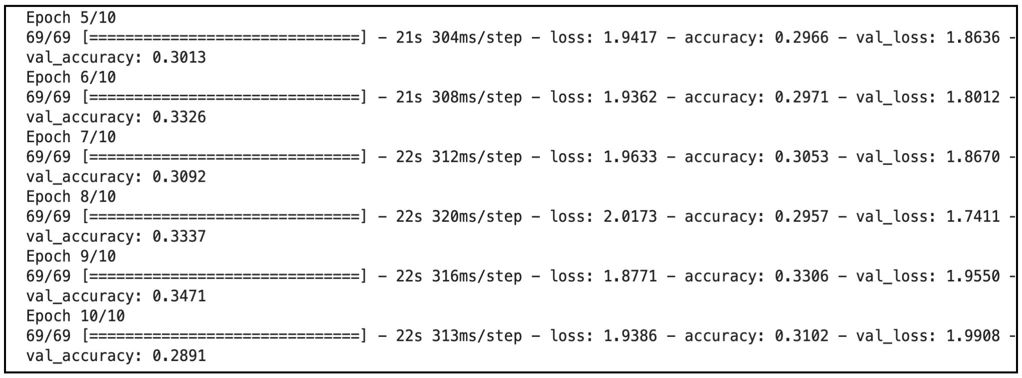
To ensure the robustness and reliability of our ethnicity detection CNN model, we split our data into four batches and conducted training and validation on a 70/30 split ratio, using 3k, 10k, 18k, and 100k images in total. During the process of developing our model, we experimented with several deep-learning convolutional neural networks, including VGG16, VGG19, ResNet50, and VGGFace. Moreover, we developed our own model from the beginning using the Sequential CNN Model, with a focus on hyperparameter optimization. Our datasets were then trained on the above-mentioned four models which were pre-trained on the ImageNet and VGGFace datasets. We used 224x224 images with three channels for these models.

It's important to note that we did not use the pre-trained weights of these models for our training, but rather replaced the last layer of the pre-trained model with our own, tailored for our dataset. We conducted training on 10 epochs, carefully analyzing each model's performance to determine the most robust and reliable option for ethnicity detection.



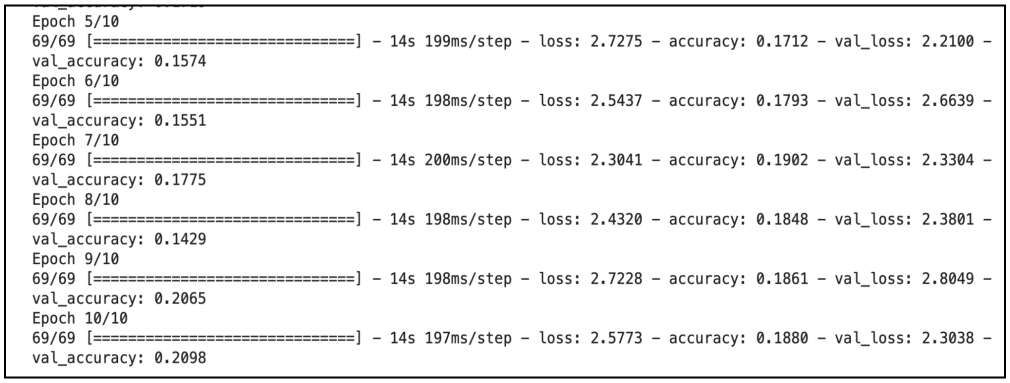
```
Epoch 5/10
69/69 [==============================] - 18s 254ms/step - loss: 1.8502 - accuracy: 0.3256 - val_loss:
1.7768 - val_accuracy: 0.3315
Epoch 6/10
69/69 [==============================] - 17s 252ms/step - loss: 1.8141 - accuracy: 0.3351 - val_loss:
1.7530 - val_accuracy: 0.3304
Epoch 7/10
69/69 [==============================] - 17s 251ms/step - loss: 1.7821 - accuracy: 0.3483 - val_loss:
1.8178 - val_accuracy: 0.3393
Epoch 8/10
69/69 [==============================] - 17s 251ms/step - loss: 1.8240 - accuracy: 0.3374 - val_loss:
1.8259 - val_accuracy: 0.3449
Epoch 9/10
69/69 [==============================] - 17s 250ms/step - loss: 1.8726 - accuracy: 0.3311 - val_loss:
1.8915 - val_accuracy: 0.3125
Epoch 10/10
69/69 [==============================] - 17s 250ms/step - loss: 1.8342 - accuracy: 0.3361 - val_loss:
1.9482 - val_accuracy: 0.3058
```

Fig. 7: Results for VGG16 after 10 epochs



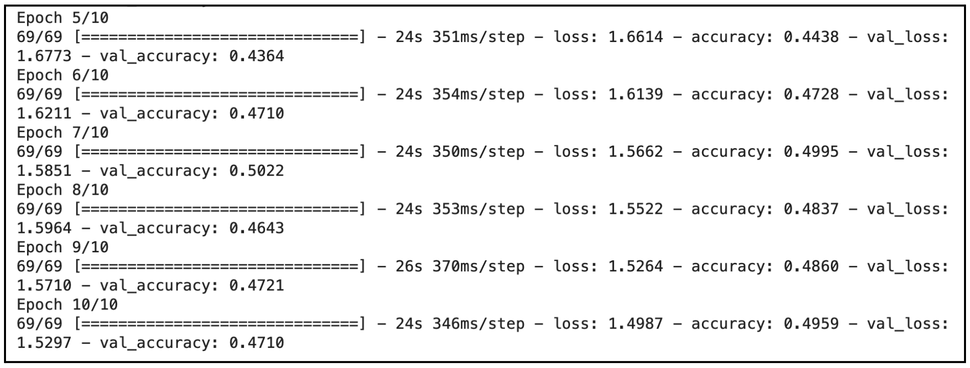
```
Epoch 5/10
69/69 [==============================] - 21s 304ms/step - loss: 1.9417 - accuracy: 0.2966 - val_loss: 1.8636 -
val_accuracy: 0.3013
Epoch 6/10
69/69 [==============================] - 21s 308ms/step - loss: 1.9362 - accuracy: 0.2971 - val_loss: 1.8012 -
val_accuracy: 0.3326
Epoch 7/10
69/69 [==============================] - 22s 312ms/step - loss: 1.9633 - accuracy: 0.3053 - val_loss: 1.8670 -
val_accuracy: 0.3092
Epoch 8/10
69/69 [==============================] - 22s 320ms/step - loss: 2.0173 - accuracy: 0.2957 - val_loss: 1.7411 -
val_accuracy: 0.3337
Epoch 9/10
69/69 [==============================] - 22s 316ms/step - loss: 1.8771 - accuracy: 0.3306 - val_loss: 1.9550 -
val_accuracy: 0.3471
Epoch 10/10
69/69 [==============================] - 22s 313ms/step - loss: 1.9386 - accuracy: 0.3102 - val_loss: 1.9908 -
val_accuracy: 0.2891
```

Fig. 8: Results for VGG19 after 10 epochs



```
Epoch 5/10
69/69 [==============================] - 14s 199ms/step - loss: 2.7275 - accuracy: 0.1712 - val_loss: 2.2100 -
val_accuracy: 0.1574
Epoch 6/10
69/69 [==============================] - 14s 198ms/step - loss: 2.5437 - accuracy: 0.1793 - val_loss: 2.6639 -
val_accuracy: 0.1551
Epoch 7/10
69/69 [==============================] - 14s 200ms/step - loss: 2.3041 - accuracy: 0.1902 - val_loss: 2.3304 -
val_accuracy: 0.1775
Epoch 8/10
69/69 [==============================] - 14s 198ms/step - loss: 2.4320 - accuracy: 0.1848 - val_loss: 2.3801 -
val_accuracy: 0.1429
Epoch 9/10
69/69 [==============================] - 14s 198ms/step - loss: 2.7228 - accuracy: 0.1861 - val_loss: 2.8049 -
val_accuracy: 0.2065
Epoch 10/10
69/69 [==============================] - 14s 197ms/step - loss: 2.5773 - accuracy: 0.1880 - val_loss: 2.3038 -
val_accuracy: 0.2098
```

Fig. 9: Results for Resnet 50 after 10 epochs

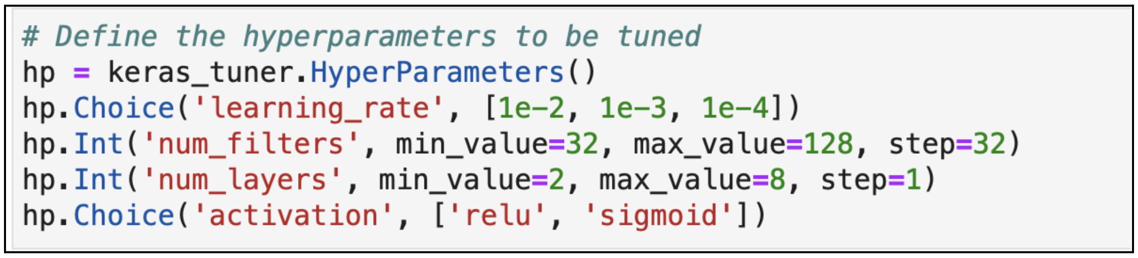


```
Epoch 5/10
69/69 [==============================] - 24s 351ms/step - loss: 1.6614 - accuracy: 0.4438 - val_loss:
1.6773 - val_accuracy: 0.4364
Epoch 6/10
69/69 [==============================] - 24s 354ms/step - loss: 1.6139 - accuracy: 0.4728 - val_loss:
1.6211 - val_accuracy: 0.4710
Epoch 7/10
69/69 [==============================] - 24s 350ms/step - loss: 1.5662 - accuracy: 0.4995 - val_loss:
1.5851 - val_accuracy: 0.5022
Epoch 8/10
69/69 [==============================] - 24s 353ms/step - loss: 1.5522 - accuracy: 0.4837 - val_loss:
1.5964 - val_accuracy: 0.4643
Epoch 9/10
69/69 [==============================] - 26s 370ms/step - loss: 1.5264 - accuracy: 0.4860 - val_loss:
1.5710 - val_accuracy: 0.4721
Epoch 10/10
69/69 [==============================] - 24s 346ms/step - loss: 1.4987 - accuracy: 0.4959 - val_loss:
1.5297 - val_accuracy: 0.4710
```

Fig. 10: Results for VGGFace after 10 epochs

Above are the results with respect to each transfer learning model for 10 epochs. After analyzing the evaluation metrics, it became evident that VGGFace outperforms all other models mentioned above.

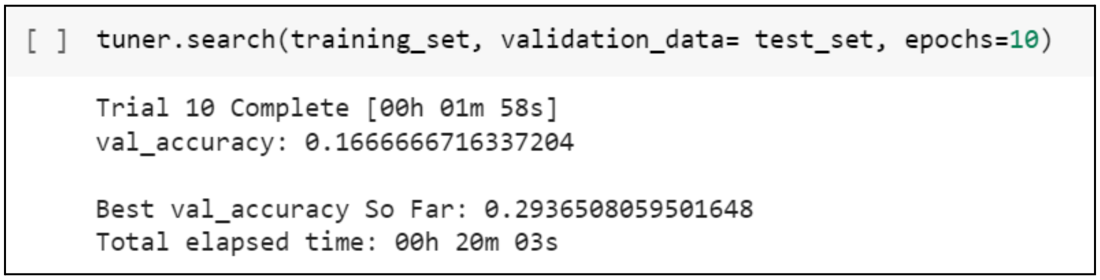
Regarding the sequential CNN model, hyperparameter optimization was performed using the Keras-tuner module. We conducted a hyperparameter random search for 10 trials, and the best hyperparameters were identified from the choices listed in the image.



```
# Define the hyperparameters to be tuned
hp = keras_tuner.HyperParameters()
hp.Choice('learning_rate', [1e-2, 1e-3, 1e-4])
hp.Int('num_filters', min_value=32, max_value=128, step=32)
hp.Int('num_layers', min_value=2, max_value=8, step=1)
hp.Choice('activation', ['relu', 'sigmoid'])
```

Fig. 11: Code for hyperparameters tuning

The hyperparameters that were identified have a learning rate of 0.001, batch size of 32, and 6 layers, activation functions were a mix of *Relu* and *sigmoid* followed by a last *softmax* activation function, and finally, the number of filters was 64. However, the validation accuracy obtained was only ~30% as shown in Figure 11, which is significantly lower compared to the VGGFace model trained on the same dataset.



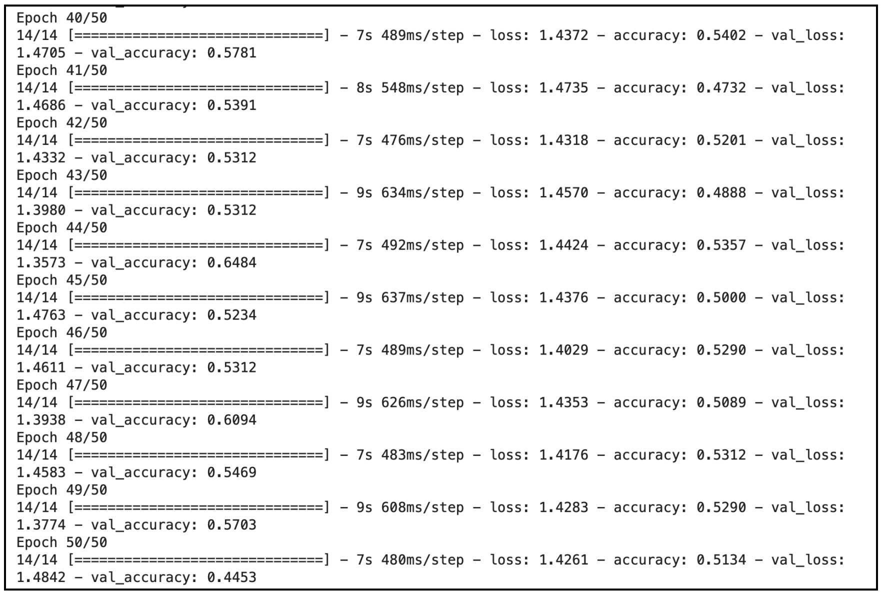
```
[ ]  tuner.search(training_set, validation_data= test_set, epochs=10)

     Trial 10 Complete [00h 01m 58s]
     val_accuracy: 0.1666666716337204

     Best val_accuracy So Far: 0.2936508059501648
     Total elapsed time: 00h 20m 03s
```

Fig. 12: Results for Sequential CNN model

We trained all the models on this dataset and closely monitored the training and validation accuracy metrics to assess their performance. Our goal was to identify the most effective model to use on the full dataset. After analyzing the evaluation metrics, it became evident that VGGFace outperforms all other models mentioned above. This is due to VGGFace being trained on a diverse and robust face dataset with deep convolutional layers.



```
Epoch 40/50
14/14 [==============================] - 7s 489ms/step - loss: 1.4372 - accuracy: 0.5402 - val_loss:
1.4705 - val_accuracy: 0.5781
Epoch 41/50
14/14 [==============================] - 8s 548ms/step - loss: 1.4735 - accuracy: 0.4732 - val_loss:
1.4686 - val_accuracy: 0.5391
Epoch 42/50
14/14 [==============================] - 7s 476ms/step - loss: 1.4318 - accuracy: 0.5201 - val_loss:
1.4332 - val_accuracy: 0.5312
Epoch 43/50
14/14 [==============================] - 9s 634ms/step - loss: 1.4570 - accuracy: 0.4888 - val_loss:
1.3980 - val_accuracy: 0.5312
Epoch 44/50
14/14 [==============================] - 7s 492ms/step - loss: 1.4424 - accuracy: 0.5357 - val_loss:
1.3573 - val_accuracy: 0.6484
Epoch 45/50
14/14 [==============================] - 9s 637ms/step - loss: 1.4376 - accuracy: 0.5000 - val_loss:
1.4763 - val_accuracy: 0.5234
Epoch 46/50
14/14 [==============================] - 7s 489ms/step - loss: 1.4029 - accuracy: 0.5290 - val_loss:
1.4611 - val_accuracy: 0.5312
Epoch 47/50
14/14 [==============================] - 9s 626ms/step - loss: 1.4353 - accuracy: 0.5089 - val_loss:
1.3938 - val_accuracy: 0.6094
Epoch 48/50
14/14 [==============================] - 7s 483ms/step - loss: 1.4176 - accuracy: 0.5312 - val_loss:
1.4583 - val_accuracy: 0.5469
Epoch 49/50
14/14 [==============================] - 9s 608ms/step - loss: 1.4283 - accuracy: 0.5290 - val_loss:
1.3774 - val_accuracy: 0.5703
Epoch 50/50
14/14 [==============================] - 7s 480ms/step - loss: 1.4261 - accuracy: 0.5134 - val_loss:
1.4842 - val_accuracy: 0.4453
```

Fig. 13: Results for VGGFace after 50 epochs

The VGGFace model achieved a validation accuracy of ~65% after being trained on our dataset for 50 epochs which is significantly high compared to the other transfer learning (pre-trained) models and the sequential model. During the training of the VGGFace Model, it iteratively called the callback function, ReduceLROnPlateau for checking and updating learning rates regularly.

5. Assumptions

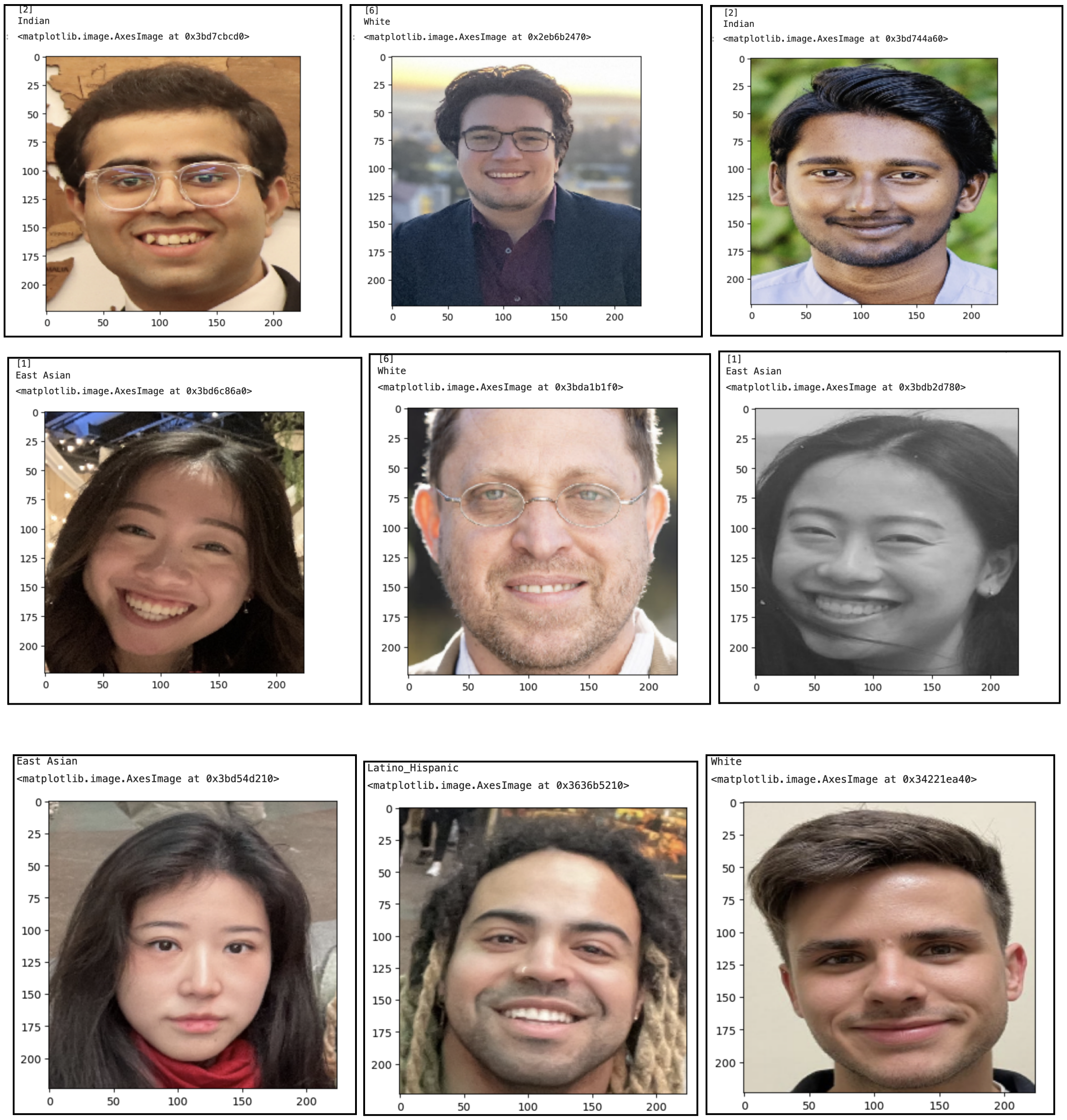
While creating the dataset, assumptions were made in order to streamline the process of training and validation of the selected models. The selected dataset, FairFace, contained more than 100,000 images. Of these 100,000 images, it was assumed that all of them met the criteria of containing a decent-quality facial image of someone within the seven selected ethnicities. Along with the dataset containing the images, there was a CSV file that assigned each image in both the training and validation to a respective ethnicity. It was assumed that these assignments were accurate. With these two main assumptions, the model could be trained, validated, and tested.

6. Results

The results obtained for the accuracy of each of the models used can be summarized in the following table:

Model	Training accuracy	**Validation accuracy**
VGG16	0.3374	0.3449
VGG19	0.3306	0.3471
Resnet50	0.1861	0.2098
VGGFace (10 epochs)	0.4995	0.5022
VGGFace (50 epochs)	0.5357	0.6484
Our Model (Normal Sequential)	0.2936	0.2936

After training the VGGFace Model for 50 epochs, we tested its accuracy on a separate subset of facial images that were not used in the training process. We measured the accuracy of the predictions using a variety of metrics, including training loss, validation loss, training accuracy, validation accuracy, and confusion matrix. The results that were logged are highlighted below.



```
[2]
Indian
<matplotlib.image.AxesImage at 0x3bd7cbcd0>
```

```
[6]
White
<matplotlib.image.AxesImage at 0x2eb6b2470>
```

```
[2]
Indian
<matplotlib.image.AxesImage at 0x3bd744a60>
```

```
[1]
East Asian
<matplotlib.image.AxesImage at 0x3bd6c86a0>
```

```
[6]
White
<matplotlib.image.AxesImage at 0x3bda1b1f0>
```

```
[1]
East Asian
<matplotlib.image.AxesImage at 0x3bdb2d780>
```

```
East Asian
<matplotlib.image.AxesImage at 0x3bd54d210>
```

```
Latino_Hispanic
<matplotlib.image.AxesImage at 0x3636b5210>
```

```
White
<matplotlib.image.AxesImage at 0x34221ea40>
```

Fig. 14: Ethnicity prediction model samples on Test images

With over 20 tests done we saw over ~80% accuracy in the prediction. But it was clearly seen that the tests the model failed were having biased features for the right prediction. Three out of the four fails were between white and Latino Hispanic races.

6.1. Public Survey

Given all the ethical implications explained in the background section of the report, we considered it convenient to make user research to analyze how our machine-learning model is perceived. To accomplish this we performed a survey on current students residing in International House, intending to reach a diverse population. Additionally, the subjects on which we tried our model got to vote on our survey as well. The results and questions from the survey were as follows:

1. Would you feel offended if the machine learning model predicted your ethnicity incorrectly based on a picture of yourself?

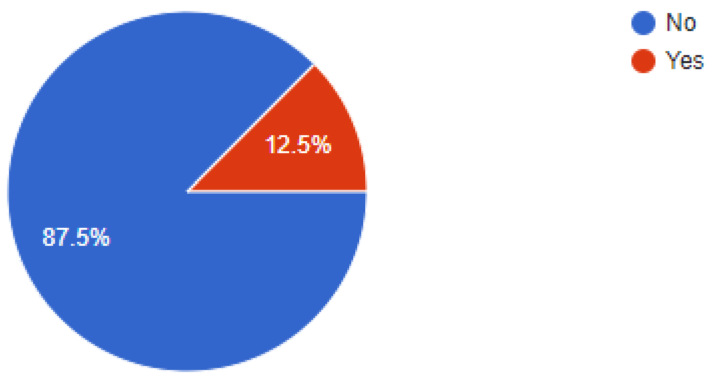


Fig. 15: Pie chart question 1

2. Do you think it is ethical to use AI to make determinations about people's race?

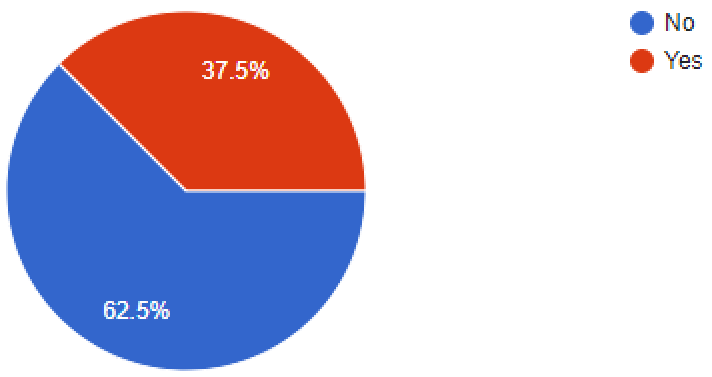


Fig. 16: Pie chart question 2

3. In your opinion, who is to blame if a human uses this system to discriminate against individuals based on their ethnicity?

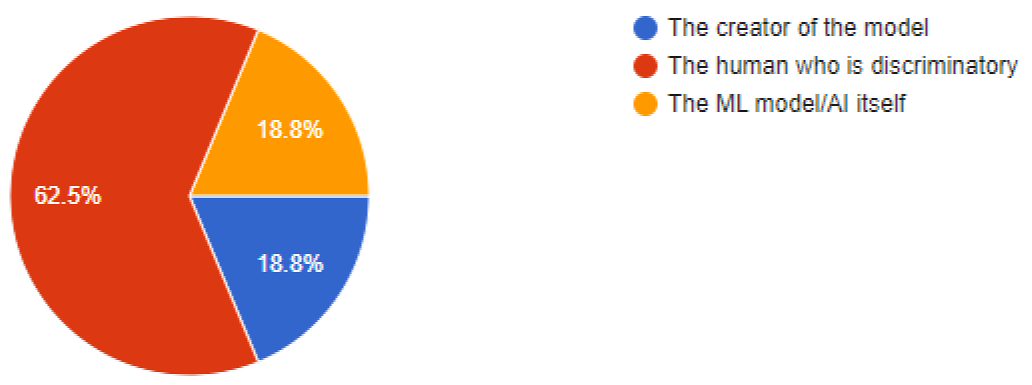


Fig. 17: Pie chart question 3

In the end, a total of 30 people completed the survey. As the above graphics represent, most of the people surveyed wouldn't feel offended if our model predicted their race wrongly. However, more than half (62.5%) of the responses indicate that they do not consider it ethical that AI determines people's races. Additionally, results from the last question show that, even though, most of the people believe the agent to blame in a discriminatory action is the human being discriminatory, some people consider that the technology itself or the creator of the ML model could to some extent be the ones actively performing the discrimination

Additionally, results from the last question show that the majority believe that the human agent is responsible for the discriminatory action. However, a substantial percentage (37.6%) expressed that they believe the technology or the ML model's creator might be actively perpetrating discrimination to some extent. Further research should be done on this matter, but this surprising results, when done over a much larger scale population, would reflect the resignation that the general public has towards AI interfering with race and ethnic considerations.

7. Conclusions

This project allowed for the development of familiarity with artificial neural networks through the development, application, and implementation of numerous models in hopes of being able to determine an individual's ethnicity from a 244 x 244-pixel facial image. Selected models trained on a vast 100,000 image data set consisting of face images of seven ethnicities. The models that were believed to be able to process and interpret the data were a Normal Sequential Model, ResNet, VGG-16, VGG-19, and VGG Face. Of these models, VGG Face performed significantly better than the others as it obtained a 65% validation accuracy.

8. References

1. K. Karkkainen and J. Joo, ‘FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age for Bias Measurement and Mitigation’, in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 1548–1558.
2. “Tf.keras.applications.vgg16.VGG16 &nbsp;: &nbsp; tensorflow V2.12.0,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg16/VGG16 [Accessed: 24-Apr-2023].
3. “Module: tf.keras.applications.resnet50 | TensorFlow Core v2.9.0,” TensorFlow. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50 [Accessed: 24-Apr-2023].
4. “Module: tf.keras.applications.vgg19 | TensorFlow v2.12.0,” TensorFlow. [Online]. https://www.tensorflow.org/api_docs/python/tf/keras/applications/vgg19 [Accessed: 26-Apr-2023].
5. R. C. Malli, “rcmalli/keras-vggface,” GitHub [Online]., Apr. 20, 2021. https://github.com/rcmalli/keras-vggface [Accessed: 24-Apr-2023].
6. https://docs.google.com/forms/d/e/1FAIpQLSekXrOE8sxUvBN6fPeaqusBD-99MrUsD-wvFI7-dmBSSDOP9Q/viewform?usp=sf_link
7. [1]K. Team, “Keras documentation: KerasTuner,” keras.io. [Online]. https://keras.io/keras_tuner/. [Accessed: 27-Apr-2023].