Age and Gender Detection using Convolutional Neural Networks with TensorFlow

Introduction

The rate of image uploads to the Internet has grown at a nearly exponential rate in the past few years. With the increased availability of tagged photos on social media and facial detection processes such as unlocking phones, this data is becoming more widely available. This data gives rise to a rather exciting task of accurate and efficient facial identity detection models. This technology can be applied in a broad range of applications, spanning from social networks, healthcare, security and marketing. Useful predictions between consumer behavior and consumer age/gender can lead to a more personalized and targeted user experience. Accurately predicting age and gender can be challenging; people in the same age range can look very different depending on various factors such as race, socioeconomic status, and even genetics while there are many people who are gender nonconforming. In this project, convolutional neural network architecture using the TensorFlow framework is applied to predict the age and gender from a still image containing a human face, and explore ways to improve the accuracy.

Data Source

https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

This is claimed to be the largest publicly available dataset of face images with gender and age labels for training. This data set consists of 460,723 face images from 20,284 celebrities from IMDb and 62,328 from Wikipedia, thus 523,051 in total. All the files are in the JPEG format and different sizes.

Data Preprocessing and Feature Engineering

This dataset contains a metadata file, containing the year the photo was taken, date of birth, and gender for each person in the image. Using the recorded date of birth and the date the photo was taken, the age was calculated assuming the picture was taken in the middle of the year.

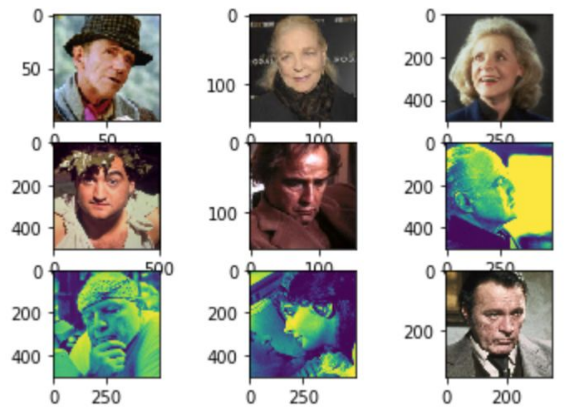
Additionally, the metadata file contains face_score and second_face_score values. The face_score indicates the facial detector score, with a higher detector score indicating a better quality view of the face. A finite second_face_score specifies the detector score for additional faces in the image. The valid images to be analyzed were selected under the following conditions:

1) Valid gender (0 for female, 1 for male)
2) Age between 9-99
3) Valid face_score value above 1 [1]
4) Null value for second_face_score value, i.e. images with a detected second face were not considered

Upon manual inspection of the image dataset, it was noticed that most of the images labeled as 8 years or less are inaccurate. Based on this observation, only the ages between 9 and 99 were selected to be valid. Additionally, it is recognized that there can be more issues with the dataset that can affect the model performance:

- Mislabeled images for gender and date of birth
- Damaged images
- Sideways or covered face shots
- Partial face shots
- Group images with no clear face shot

As an example to demonstrate the above issues, a quick look at just 9 of the images, clearly show the mislabeling of images (Row 3 Col 2, female labeled as a male). These issues will directly impact model performance.



```
0 0 Gender: 1 Age: 69
0 1 Gender: 0 Age: 82
0 2 Gender: 0 Age: 66
1 0 Gender: 1 Age: 29
1 1 Gender: 1 Age: 48
1 2 Gender: 1 Age: 66
2 0 Gender: 1 Age: 72
2 1 Gender: 1 Age: 37
2 2 Gender: 1 Age: 51
```

Fig 1: Representation of images and their age and gender labels

Even though some images that are mislabeled were manually removed, it must be noted that there still could be such images with issues that would impact model performance.

Exploratory Data Analysis

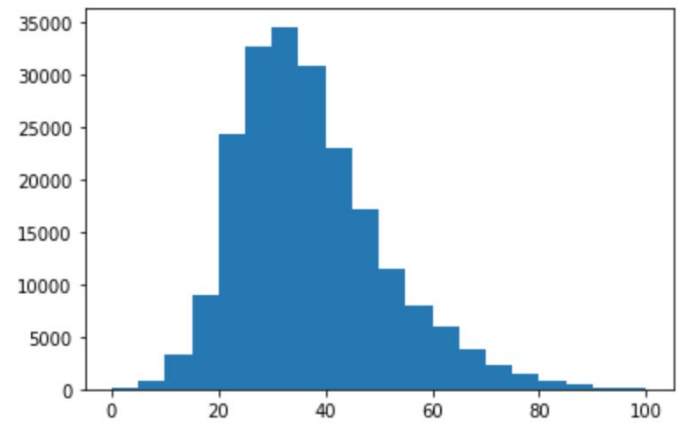


Fig 1: Age distribution (0-100) of images

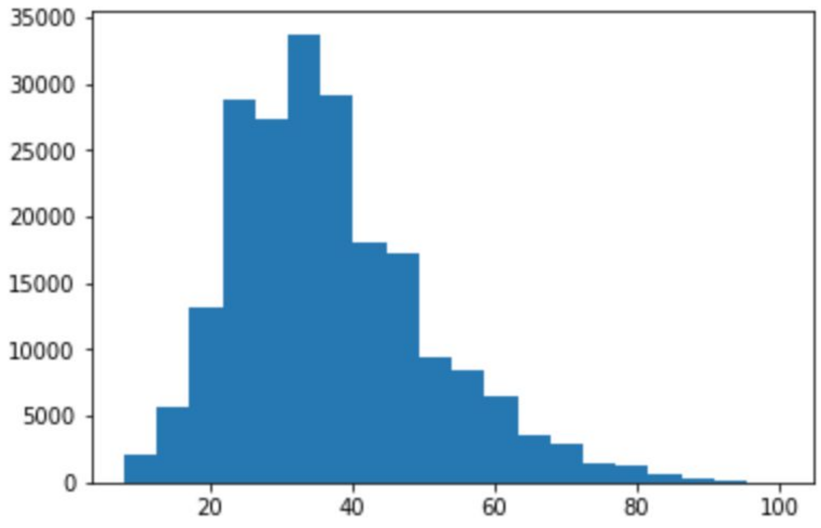


Fig 2: Age distribution (9-99) of ‘valid’ images

The above figures illustrate the age distribution of subjects in the images. We can see there’s a large difference in the representation of different age groups. There aren’t many images in the ranges <20 and >80, thus making it hard for the model to identify features of such human faces. Ideally, this should be a uniform representation. Additionally, we see this disparity when it comes to gender distribution as well (122167 male and 87371 female).

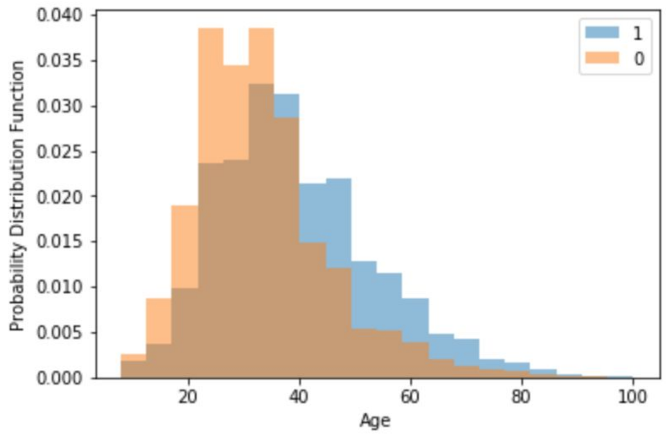


Fig 3: Gender distribution grouped by age

When looking at the data distribution of age classified by gender, we can see that both make and female distributions are right skewed. The male distribution has a mean of which is greater than the female distribution while the female distribution is narrower and more skewed. This is also reflected in the boxplot below:

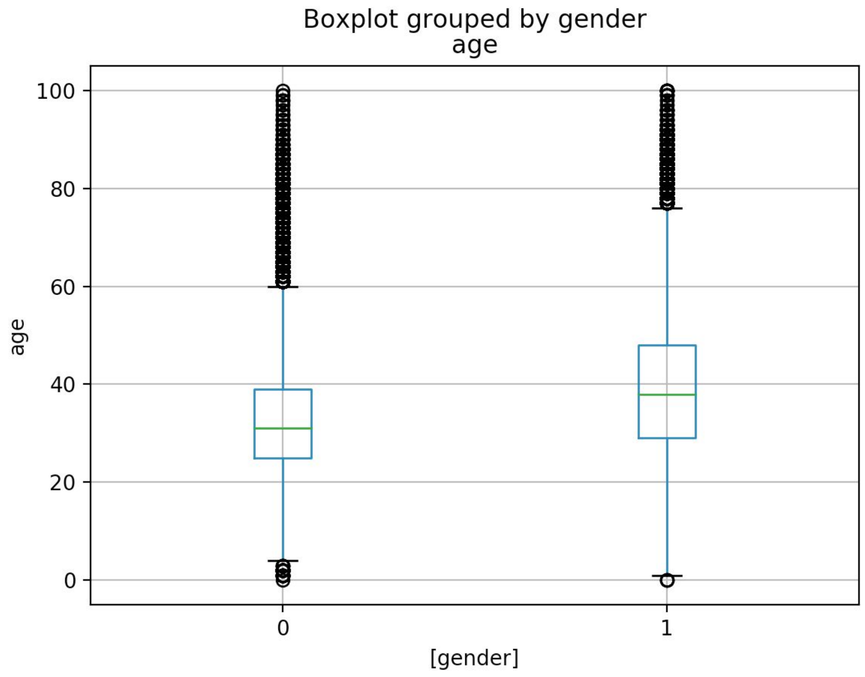


Fig 4: Age distribution grouped by gender

All images were resized to 128x128 as images are required to be of the same size by TensorFlow. The validated and cleaned dataset were split into 80% train and 20% test datasets and written as csv files to be input into the convolutional neural network. The data sets contained the full path to the images and age and gender labels for each image.

Convolutional Neural Network

The data is read with the Keras flow_from_dataframe() method as it allows a Pandas DataFrame to be used as a direct input source. Using this function generates the images dynamically thus not having a minimal memory requirement. In this work, a batch size of 32 was used for the test set, while a batch size of 1 was used for validation and test sets. The 'class_mode' in this function prepares the data to be trained. For age computation, class_mode='raw', and class_mode='binary' for gender classification. If this has been handled as a multi-task neural network, simultaneously computing age and gender, it should be set to 'multi_output'.

A sequential neural network is built from scratch in this case. A sequential model builds a model layer by layer. Convolutional layers are capable of extracting useful information from the images. A convolution multiplies a matrix of pixels with a filter matrix or 'kernel' and sums up the multiplication values. Then the convolution slides over to the next pixel and repeats the same process until all the image pixels have been covered. The first layer also takes in an input shape. This is the shape of each input image, (128, 128, 3), with the 3 indicating the number of color channels (RGB colored). In between convolutional layers, there is a MaxPooling2D layer to down-sample input, to prevent overfitting. Additionally, we also incorporate BatchNormalization as a method of regularization, performing the normalization for each training mini-batch. This down-sampling reduces the dimensions of the input and allows the training engine to make assumptions about features. Each of these convolutional layers have the Rectified Linear Unit (ReLU) as the activation function. This activation function has been proven to work well in neural networks. Conv2D and MaxPooling2D layers are repeated twice more, each with the Conv2D's filter count increasing exponentially. As data moves through our layers the data patterns get more complex. The number of filters are increased with each layer, 32, 64, 128, respectively, so we can capture as many of those pattern combinations as possible.

In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers, converting the data into a vector format. And finally we included two Dense layers. Our first Dense layer was configured as a network with 512 units and ReLU activation. In between dense layers, we use dropout at a rate of 40% to prevent overfitting. The dropout method randomly drops a specified amount of units (along with their connections) from the neural network during training, thus preventing units from co-adapting too much. The output dense layers are specified with a sigmoid activation function for the gender classification and linear activation for the age regression modeling. Upon defining the neural network, it must first be compiled before fitting.

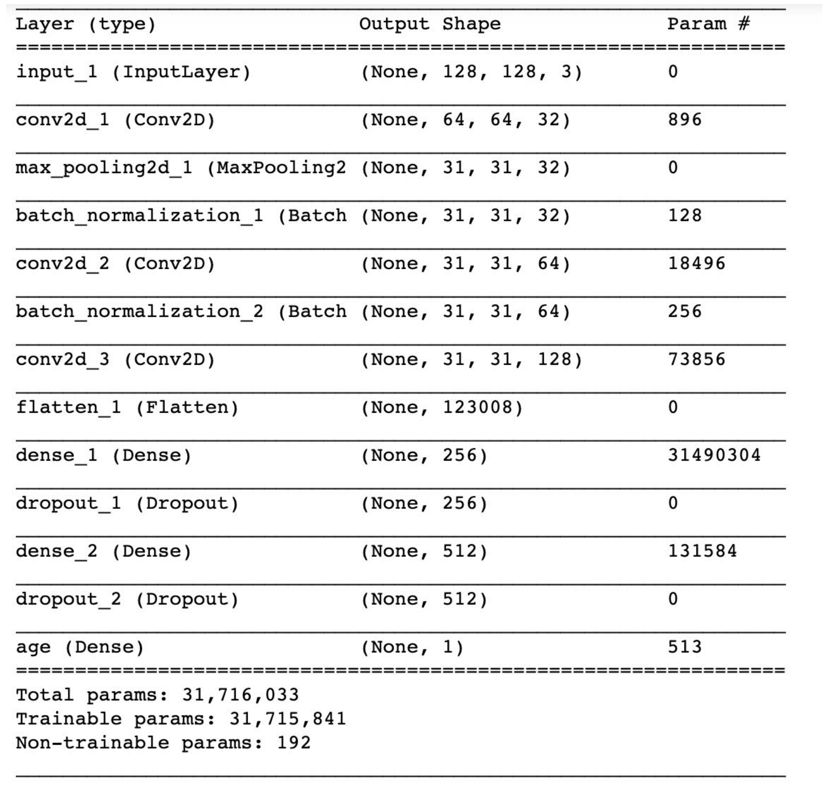
Compile

Compiling the model takes three parameters: optimizer, loss and metrics. The optimizer controls the learning rate. The most popular compiler in the field of deep-learning, 'Adam', is used here. Adam is an adaptive learning rate optimization algorithm that's been designed specifically for training these networks. It does this by leveraging the power of adaptive learning rates methods to find individual learning rates for each parameter. The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer. We have used a smaller learning rate of 0.002 in an attempt to achieve better results. The 'binary_crossentropy' was specified as the loss function and 'accuracy' as the model performance metric for gender classification, and 'mean_absolute_error' for age regression loss function and metric. A lower score for the loss function indicates that the model is learning and performing well.

Specific details for the two models are outlined below.

Age Model

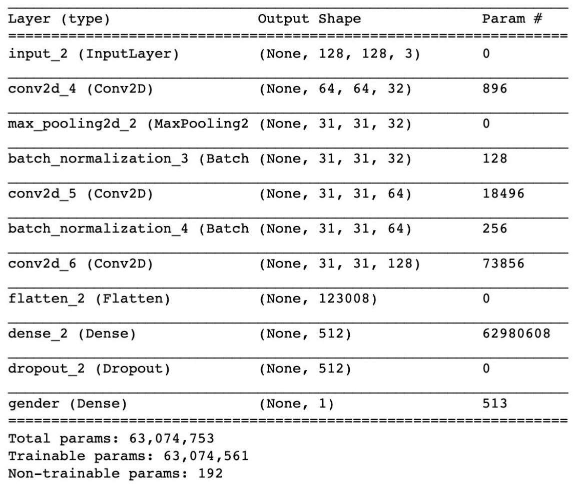
Architecture of the convolutional neural network used for age calculation as a regression:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 128, 128, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 32)        896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 31, 31, 32)        128
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 31, 31, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 31, 31, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 31, 31, 128)       73856
_________________________________________________________________
flatten_1 (Flatten)          (None, 123008)            0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               31490304
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
age (Dense)                  (None, 1)                 513
=================================================================
Total params: 31,716,033
Trainable params: 31,715,841
Non-trainable params: 192
_________________________________________________________________
```

Gender Model:

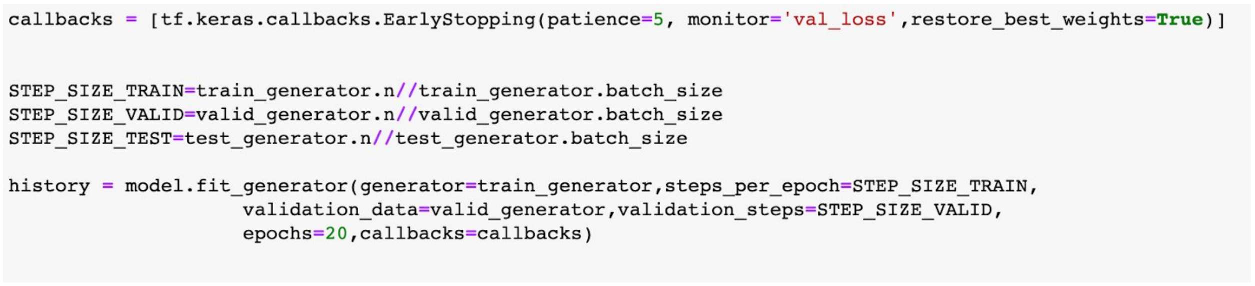
Architecture of the convolutional neural network used for gender computation as a classification:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_2 (InputLayer)         (None, 128, 128, 3)       0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 32)        896       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 31, 32)        0         
_________________________________________________________________
batch_normalization_3 (Batch (None, 31, 31, 32)        128       
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 31, 31, 64)        18496     
_________________________________________________________________
batch_normalization_4 (Batch (None, 31, 31, 64)        256       
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 31, 31, 128)       73856     
_________________________________________________________________
flatten_2 (Flatten)          (None, 123008)            0         
_________________________________________________________________
dense_2 (Dense)              (None, 512)               62980608  
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0         
_________________________________________________________________
gender (Dense)               (None, 1)                 513       
=================================================================
Total params: 63,074,753
Trainable params: 63,074,561
Non-trainable params: 192
```

Model Training

The model was then trained with the data from the training set and evaluated with the validation set. The model was fit for a total of 20 epochs. An epoch is a complete pass through the entire training data set. During, and at the end of, each epoch we'll receive feedback on how well the model is doing at the classification process. This feedback will show accuracy and loss for both the training and validation data sets. The more epochs we run, the more the model will improve, up to a certain point, as monitored by the 'Earlystopping' criteria. In this model, the validation loss function was monitored for improvement for at least 5 epochs before quitting. The step size for the model fitting and evaluation were calculated from the total number of samples in the train and validation sets and the batch size for each set as specified in the 'imagedatagenerator' function.



```
callbacks = [tf.keras.callbacks.EarlyStopping(patience=5, monitor='val_loss',restore_best_weights=True)]


STEP_SIZE_TRAIN=train_generator.n//train_generator.batch_size
STEP_SIZE_VALID=valid_generator.n//valid_generator.batch_size
STEP_SIZE_TEST=test_generator.n//test_generator.batch_size

history = model.fit_generator(generator=train_generator,steps_per_epoch=STEP_SIZE_TRAIN,
                    validation_data=valid_generator,validation_steps=STEP_SIZE_VALID,
                    epochs=20,callbacks=callbacks)
```

Results and Discussion

Correlation between the actual age and predicted values are depicted below.

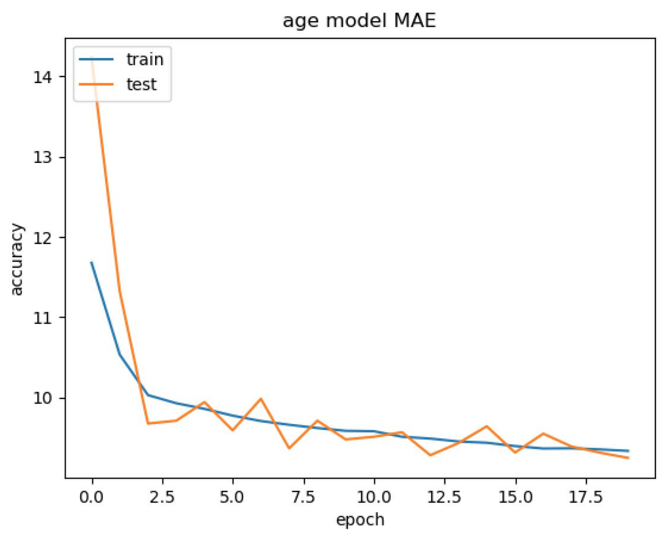


Fig 6: Mean absolute error for training and testing sets

The above figure (Fig. 6) shows the mean absolute error for the train set (blue line) and validation set (orange line) as a function of the number of epochs. These charts are useful to see if the model needs additional training, the model configuration is poor, or if our model is overfitting and has started to memorize results. We can see that the mean absolute error shows a sharp improvement after the first two epochs and continues to improve slowly. There are no signs of overfitting. This model performs well for this task with a mean absolute error of about 9.5 years.

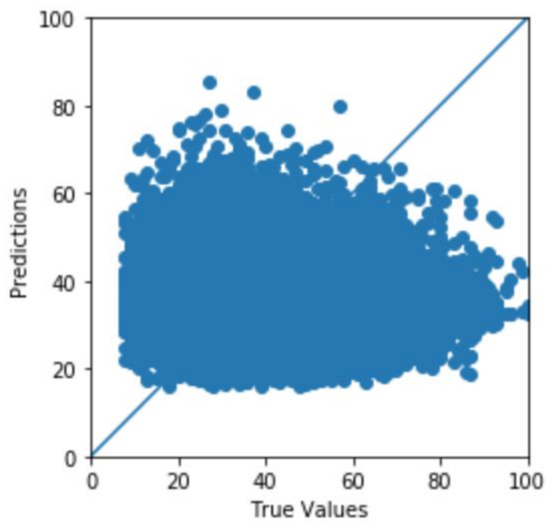
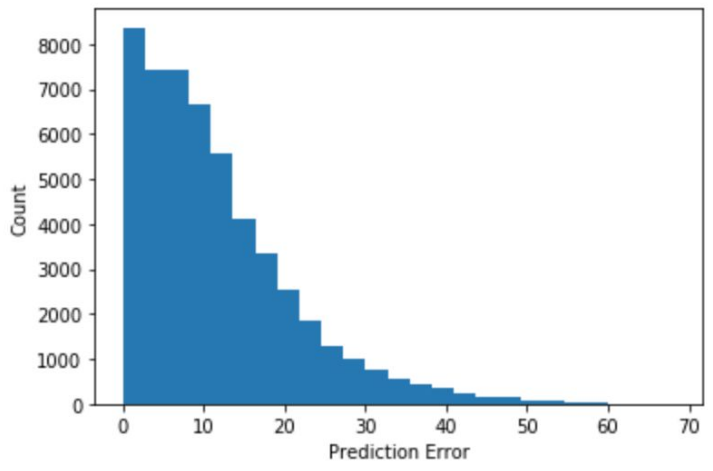


Fig 7: Correlation between the actual age and predicted values

Additionally, the correlation of the fit is illustrated in Fig 7. While ideally the fit should be narrower with more data points on the line indicating a perfect fit, the data seems to be symmetrically distributed on either side of the line with no patterns. FIgure 8 gives a better idea on how the errors are distributed. It is great that this is a right-skewed distribution with about 75% of the errors <16 years. Given the inherent issues with the image dataset itself as outlined above and the size of the neural network, this result is satisfactory.



```
count    52566.000000
mean        11.646223
std          9.612493
min          0.000008
25%          4.489491
50%          9.426586
75%         16.253327
max         68.278755
```

Fig 8: Error distribution of the predicted values

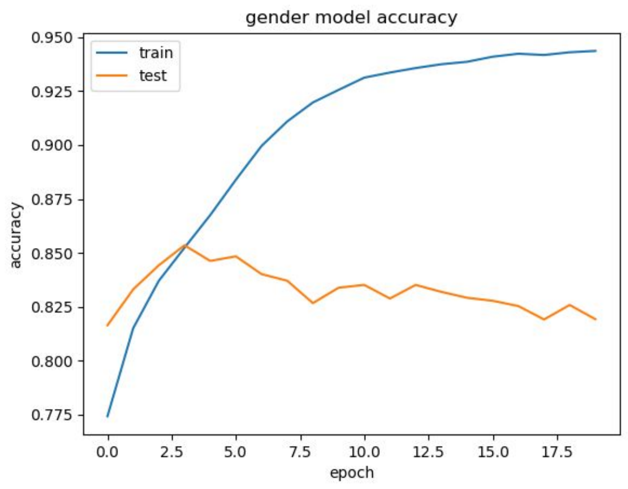


Fig 9: Gender classification accuracy for training and testing sets without data augmentation

In the above figure depicting classification accuracy for train and test sets, we can see there is overfitting where the train set accuracy is improving while the test set accuracy is plateaued at around 84%. This can be overcome by making the network deeper with more regularization techniques such as Dropout and BatchNormalization. Additionally, data augmentation can be applied to the underrepresented class to improve model accuracy. To make the model more accurate, we applied data augmentation to the testing data only, by applying up to 40 degree rotation, shear, zoom and horizontal flip. Application of data augmentation was vital to prevent overfitting as seen below:

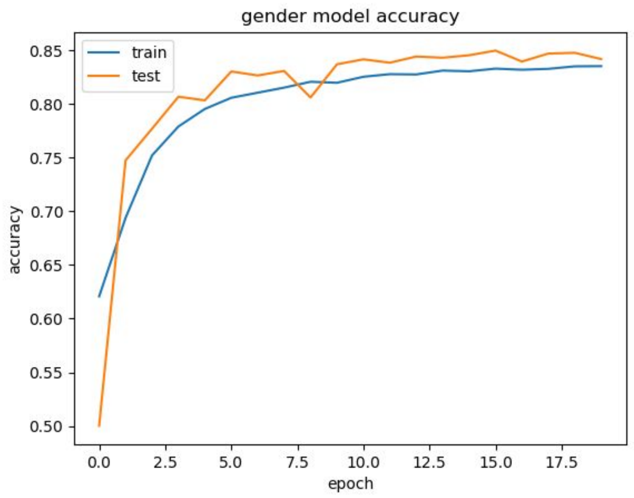


Fig 9: Gender classification accuracy for training and testing sets with data augmentation

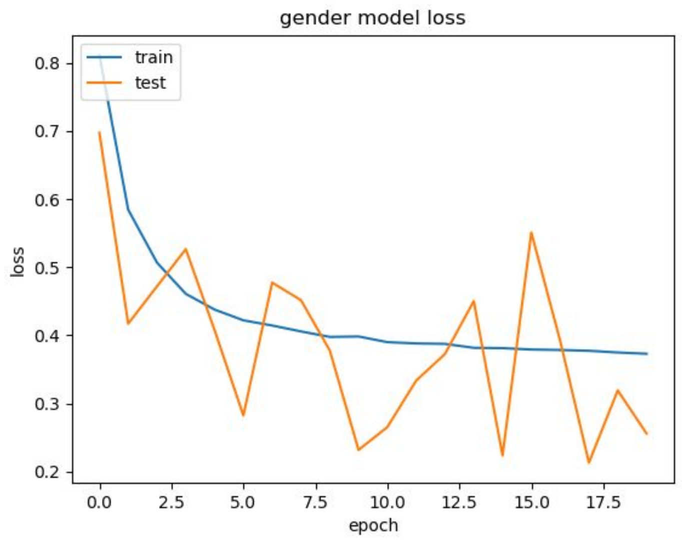


Fig 10: Gender classification loss for training and testing sets with data augmentation

Conclusion and Future Work

In conclusion, we have built a convolutional neural network successfully for age regression with no overfitting and good results. While there was overfitting occurring initially in the gender classification model, this was addressed with data augmentation. The gender model has an accuracy of about 85%, and the age model has a mean absolute error <10 years. Given the range of values and the limited dataset, the results are exceptional. The results for both models could have been further improved by optimizing the layers and hyperparameter tuning. Additionally, transfer learning can be carried with pre-trained models like VGG16, ImageNet and Inception.

This task was also attempted as a multi-task neural network. However, each epoch in the multi-task model took upto 5 hours to complete, while the single task models only took about an hour. Therefore, it is not a viable solution, as executing the single task modeling in parallel is 5x more efficient. Hence, the network architecture built in this project is accurate and efficient, which can be successfully used in user recognition in many applications like targeted marketing, security and healthcare.