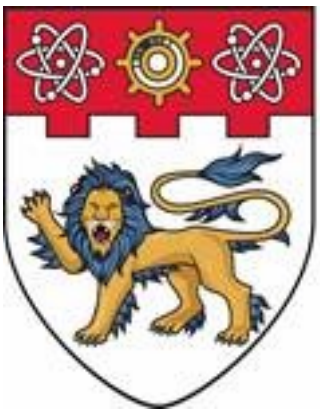


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042: Neural Network & Deep Learning

Project: Gender Classification

Semester 1, AY 2020/2021

Project 2 Report

Group Members	Matriculation No.
Addi Debashree	U1722837C
Kevin Tee	U1722953K
Trinh Tuan Dung	U1720421K

Contents

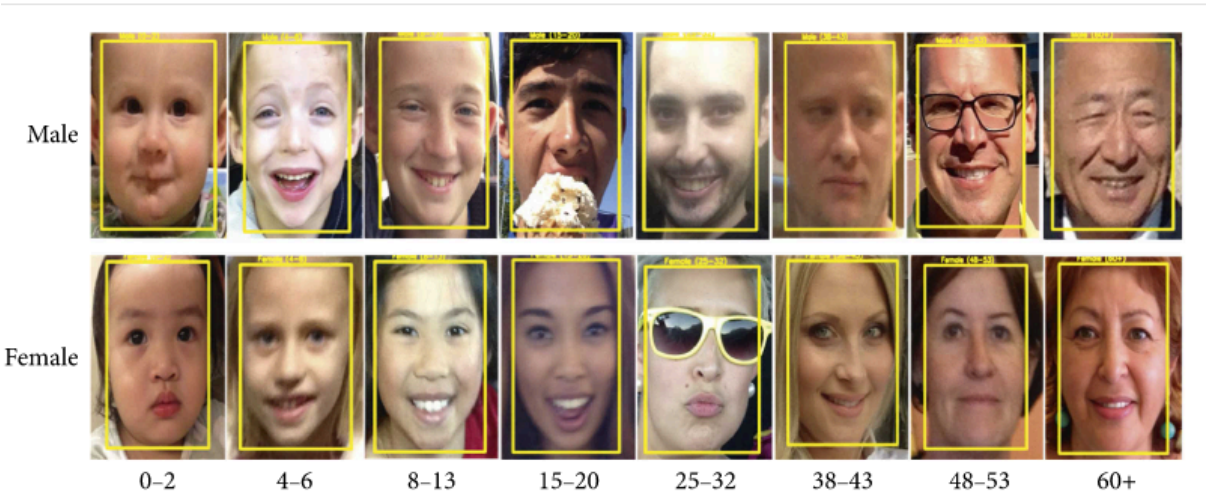
Topic

Introduction

Automatic gender classification has been used in many applications including image analysis on social platforms. The goal of this project is to classify the gender of faces in an image. One can design a convolutional neural network to achieve this goal. There are pre existing techniques tackling this problem statement. Our project utilised one of the pre-existing techniques by modifying it and analysing performance as well as developed some novel techniques of building cnn models for gender classification.

We have aimed to cover the following tasks in this project.

- Pre-training on celebA dataset
    - Gender classification with pre-trained weights from celebA
- Gender classification
- Age classification



OUI-Adience dataset images

Explanation of OUI-Adience dataset

OUI-Adience is a collection of face images from ideal real-life and unconstrained environments. It reflects all the features that are expected from an image collected from challenging real-world scenarios. They are face images that were uploaded to Flickr website from smartphone without any filtering. Adience images, therefore, display a high-level of variations in noise, pose, and appearance, among others. The entire collection of OIU-Adience dataset is

about 26,580 face images of 2,284 subjects and with an age group label of eight comprising 0–2, 4–6, 8–13, 15–20, 25–32, 38–43, 48–53, and 60+. The distribution of the face images for age group and gender class labels in OIU-Adience benchmark is presented in text files.

Structure of the OUI-Adience dataset summarized:

- ➢ Total number of photos: 26,580
- ➢ Total number of subjects: 2,284
- ➢ Number of age groups / labels: 8 (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, 60-)
- ➢ Gender labels: Yes
- ➢ In the wild: Yes
- ➢ Subject labels: Yes

The directory contains the following files:

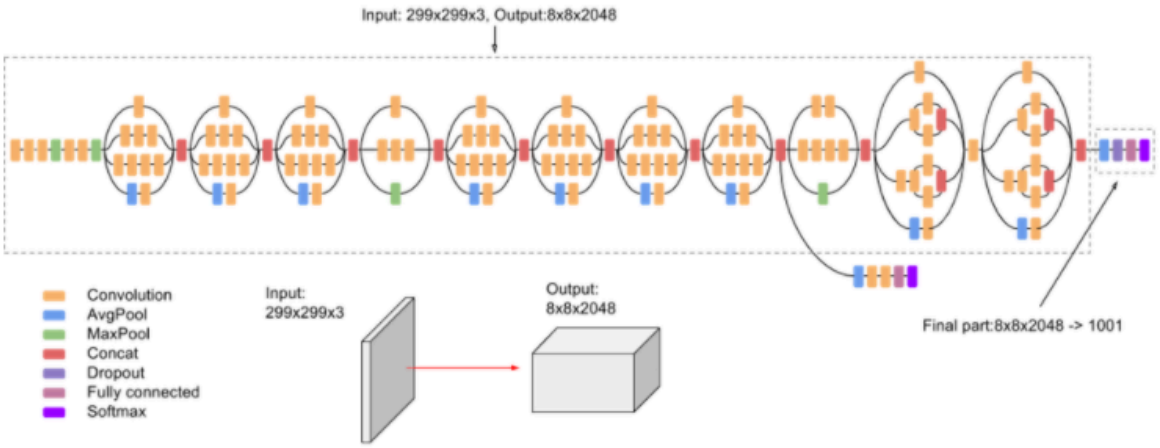
- ➢ faces.tar.gz (936M) - Face images, cropped
- ➢ aligned.tar.gz (1.9G) - Face images, cropped and aligned using our 2D, in-plain alignment tool
- ➢ fold_0_data.txt - fold_4_data.txt - text files with indices to the five-fold cross validation tests using all faces
- ➢ fold_frontal_0_data.txt - fold_frontal_4_data.txt - same as above, but using only faces in approximately frontal pose

For our project we have used to the folder ‘aligned.tar.gz’ with images for training and the fold_0_data.txt - fold_4_data.txt - text files for the target labels. The fold_0_data.txt - fold_4_data.txt - text files are all combined into a single csv file - ‘aligned_faces_data.csv’ and this csv file is further used to train the models in our code files.

Review of existing techniques

1. **InceptionV3**

   InceptionV3 is a convolutional network that is 48 layers deep. The author incorporated pre-trained weights from InceptionV3 as well added extra layers on her own when she did her training. Stochastic Gradient Descent was used and 20 epochs were used for training with 625 iterations per epoch.

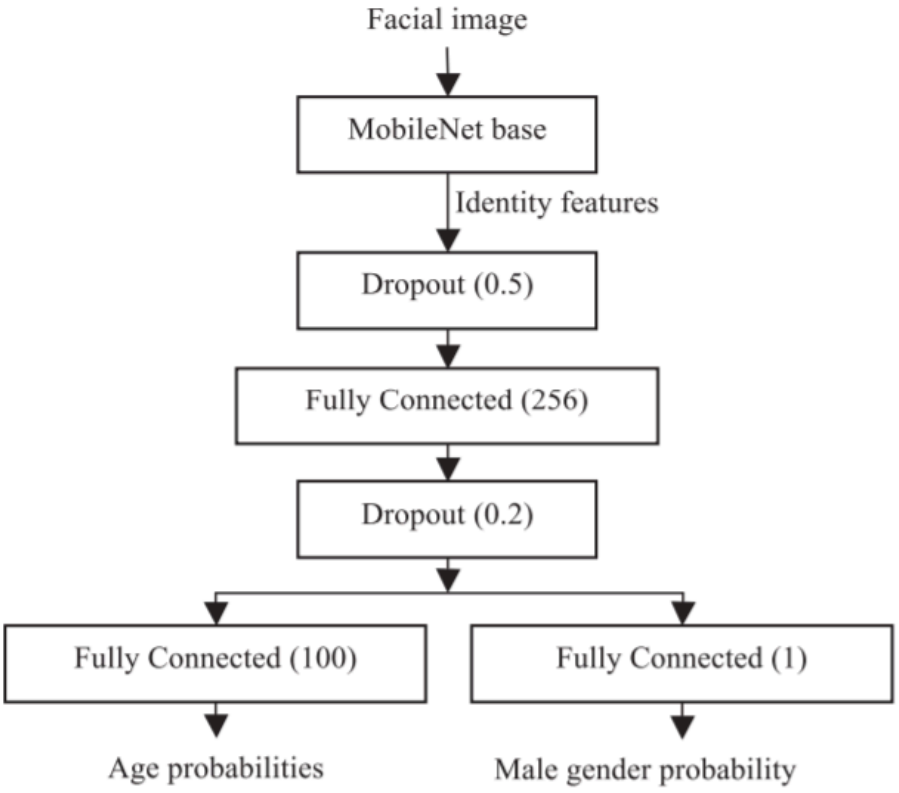


InceptionV3 network architecture

The author using this model reported a 94.3% test accuracy.

2. Mobile Net

Mobile Net is also used. The study decided to use straightforward modification of the MobileNet v1. This model contains 27 sequentially connected. convolutional and depthwise convolutional layers, which proved to be memory efficient and provide excellent inference speed even on mobile devices. It is seven- and 35-times smaller than conventional ResNet-50 and VGG16 models, respectively. What is more important, such a small size of the model does not cause a significant decrease of the recognition accuracy in various image recognition tasks. For example, top-1 accuracy on ImageNet-1000 of the MobileNet v1 (70.4%) is only 0.9% and 4.5% lower when compared to the accuracy of VGG16 and ResNet-50, respectively. The bottom (backbone) part of the proposed network, namely, conventional MobileNet v1 pre-trained on ImageNet-1000, extracts the representations suitable for face identification. The top part contains one new hidden layer with dropout regularization after extraction of identity features and two independent fully connected layers with softmax and sigmoid outputs for age and gender recognition, respectively. The learning of this model is performed incrementally, At first, the base MobileNet is trained for face identification on a very large facial dataset using conventional cross-entropy (softmax) loss. Next, the last classification layer is removed, and the weights of the MobileNet base are frozen. Finally, the remaining layers in the head are learned for age and gender recognition tasks by minimizing the sum of cross-entropies for both outputs.



Mobile Net network architecture

Pre-training on celebA dataset

Methods

Pre-training is done on CelebA dataset using YOLOV3. YOLOV3 can be used for both image classification and for object detection. In the case of image classification, we added the average pooling, fully connected layers and softmax classifier. In the case for object detection, these 'classifier' layers are being removed and a 'detection' head is appended instead. This is the innovative part on our end to locate the whereabouts of each gender or a specific gender in a particular image or video,

YOLOV3 uses CNN as its backbone network. Convolution Neural Networks (CNN) are deep, feed-forward artificial neural networks that have proven themselves in the fields of image recognition and image classification, which are widely used in analyzing visual images. The high-level process of YOLOv3 is presented in the figure below.

	Type	Filters	Size	Output
	Convolutional	32	3x3	256x256
	Convolutional	64	3x3 /2	128x128
1x	Convolutional	32	1x1	
	Convolutional	64	3x3	
	Residual			128x128
	Convolutional	128	3x3 /2	64x64
2x	Convolutional	64	1x1	
	Convolutional	128	3x3	
	Residual			64x64
	Convolutional	256	3x3 /2	32x32
8x	Convolutional	128	1x1	
	Convolutional	256	3x3	
	Residual			32x32
	Convolutional	512	3x3 /2	16x16
8x	Convolutional	256	1x1	
	Convolutional	512	3x3	
	Residual			16x16
	Convolutional	1024	3x3 /2	8x8
4x	Convolutional	512	1x1	
	Convolutional	1024	3x3	
	Residual			8x8
	Avgpool		Global	
	Connected		1000	
	Softmax			

YOLO CNN architecture

The YOLO architecture has been one of the most popular algorithms used in the field of object classification in recent years. It is faster and more successful in object recognition when compared to its competitors. Accuracy and sensitivity, which are the most important factors in object detection and prediction, are not enough on their own. For a tool to adapt to the real-time environment, the model must be able to perform the object recognition in real time, which is necessary for prediction of the images in real-time camera feeds. This is our effort to bring in novelty into our project, as mentioned earlier. Gender classification is the first step, however to make this project practical, we need to also introduce object detection whereby we can detect gender differences in real-time scenarios.

At this point, YOLO is different from other traditional methods as it carries out estimates of the bounding boxes' coordinates and its class predictions simultaneously. Together as a model, YOLO and CNN realized effective and precise real-time object recognition successfully with high average sensitivity (mAP).

YOLO can predict the class and coordinates of all the objects in the image by passing the image through the CNN structure at once instead of the zones determined by respective segmentation for object detection and then sending it to the previously written Region Proposal Network. YOLOv3 has 106 layers,

part of it consisting of layers from Darknet-53 architecture, as seen from Figure It consists of essential elements like residual blocks, skip connections/shortcut, up-sampling, route and downsampling layer.

Before we delve deep into the training and evaluation of the model, here is an explanation of the dataset we will be working on. It is called the CelebA dataset.

It is a nice, wide, and diversified dataset. It is a large-scale face attributes dataset with more than 200K celebrity images, covering a large amount of variations, each with 40 attribute annotations.

CelebA Facial Attributes:

0: 5_o_Clock_Shadow	20: Male
1: Arched_Eyebrows	21: Mouth_Slightly_Open
2: Attractive	22: Mustache
3: Bags_Under_Eyes	23: Narrow_Eyes
4: Bald	24: No_Beard
5: Bangs	25: Oval_Face
6: Big_Lips	26: Pale_Skin
7: Big_Nose	27: Pointy_Nose
8: Black_Hair	28: Receding_Hairline
9: Blond_Hair	29: Rosy_Cheeks
10: Blurry	30: Sideburns
11: Brown_Hair	31: Smiling
12: Bushy_Eyebrows	32: Straight_Hair
13: Chubby	33: Wavy_Hair
14: Double_Chin	34: Wearing_Earrings
15: Eyeglasses	35: Wearing_Hat
16: Goatee	36: Wearing_Lipstick
17: Gray_Hair	37: Wearing_Necklace
18: Heavy_Makeup	38: Wearing_Necktie
19: High_Cheekbones	39: Young

CelebA facial attributes

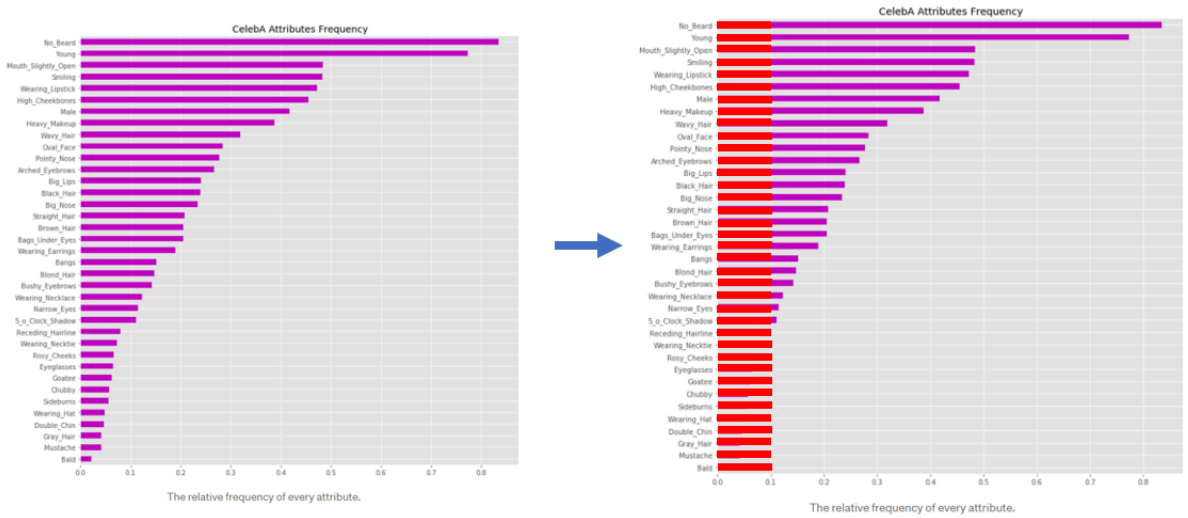
The figure above shows the different attributes an image of a celebrity may contain and they are binary attributes in terms of 1 and -1. 1 means that the attribute is present while -1 means the attribute is absent. For example, if the person in the image has gray hair, it will be labelled as 1, otherwise -1, if the person does not have.

Experiments and Results

- **Preprocessing**

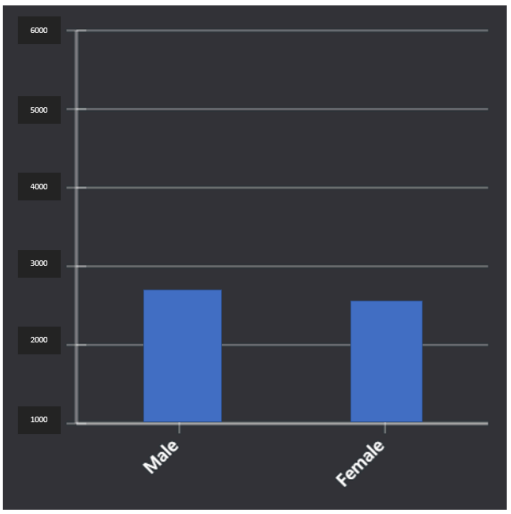
  Under preprocessing, the two different text files are being joined together as a Yolo format into a singular text file such that it can be passed into for training. This is because the bounding boxes of the images are found in another text file while the labels for the gender differences are found in another text file. The binary numbers of 1 and -1 for male and female respectively are being converted to 0 and 1 as binary to be trained by YOLOV3. Also, in the 200k celebrity dataset, it is not possible for training of all images, therefore we manually picked only 5k training

images to run through Google Colab. The 5k dataset is not chosen by random but in a strategic way because we want to include as many different types of female or males in the training/validation dataset as possible. The below shows the difference in how a random pick of images will look like and how a strategic pick of the images will look like. The reason for this is that it is well-known that the quality of data affects the training of the model irregardless if a very good algorithm is used. If the data given only represents one type of data, the model will not also be able to generalise the classification and then do detection well.



CelebA attribute frequency

For example, looking at the figure above, there is a higher proportion of images containing the attribute beard, which may cause a bias during the training of the model. This in turn will affect the classification of the gender and also the detection of the gender subsequently. Therefore, a strategic pick would be to under-sample the attribute ‘majorities’ and over-sample the attribute ‘minorities’. Finally, we also made sure that there is an equal proportion of male and female represented in the 5k dataset, as shown in the figure below.



Proportion of Males and Females

The split of the dataset is into 90:10, which follows the standard protocol. This ratio is chosen because we want to maximise the dataset for training since we are already using a small proportion of the actual dataset.

As a 5k dataset may seem insignificant to a 200k original dataset, data augmentation is being done as well before training. Data Augmentation allows us to generate images with modifications to the original ones. This will artificially increase the dataset, allowing the model to better be trained and generalise later on. The model will learn from these variations (changing angle, size and position), being able to predict better never seen images that could have the same variations in position, size and position.

- **Training**

  Batch size of 16 was used initially, with a learning rate of 0.01 and ADAM optimizer was used, with pre-trained weights used to train the Image Net. The number of epochs are set at a total of 102, with two stages of training, each with 51 epochs. The first stage of training will have freezing of layers while the second stage of training will have no freezing of layers.

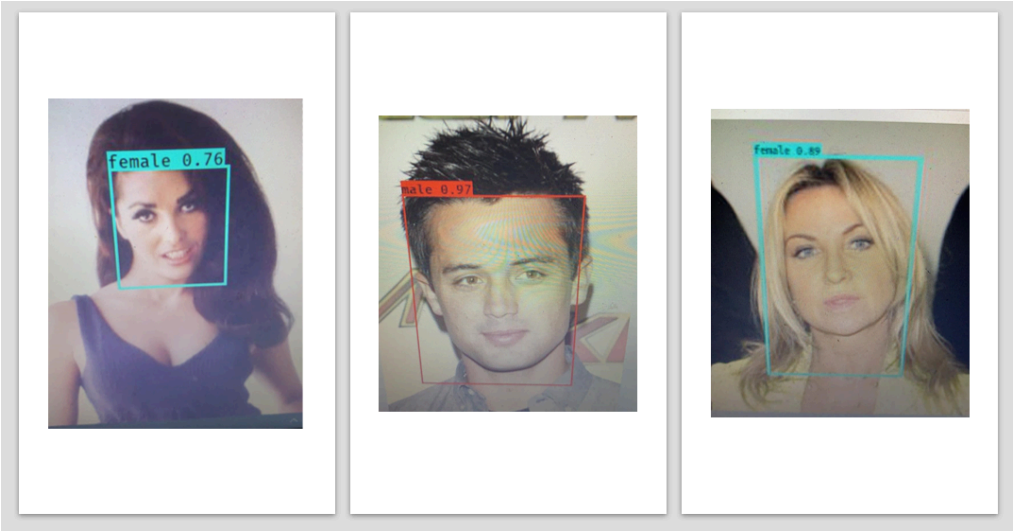
  Training was fine-tuned with different batch sizes and epochs. It was discovered that the losses plateau before reaching the 51st epoch of each stage of training at about 30th epoch.

  Different batch sizes like Batch size of 8 and 32 are being used. The training when batch size 32 is used is a lot slower due to the higher memory being used while the accuracy resulted from Batch size 8 is slightly lower, therefore we decided to stick with the original hyperparameters.

- **Results**

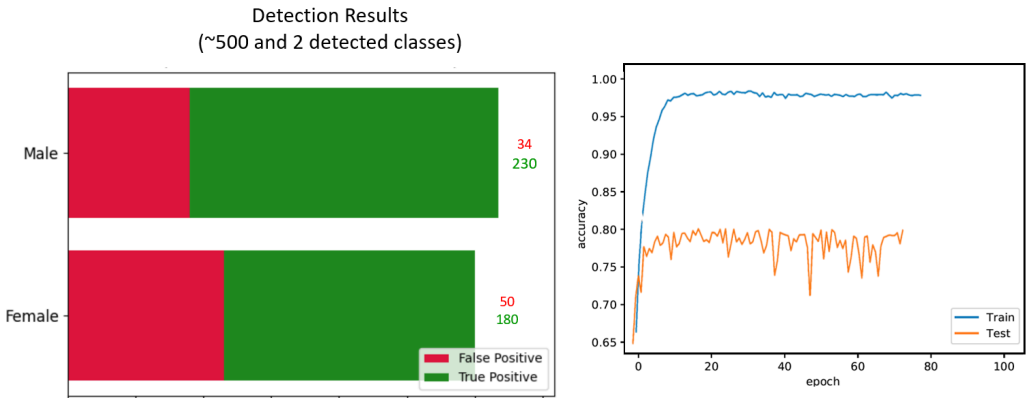
  Results are split into training/test accuracy for gender classification and mean accuracy precision (mAP) for gender detection.

  The model was able to recognise gender differences and detect them with high probability, as seen in the figure below.



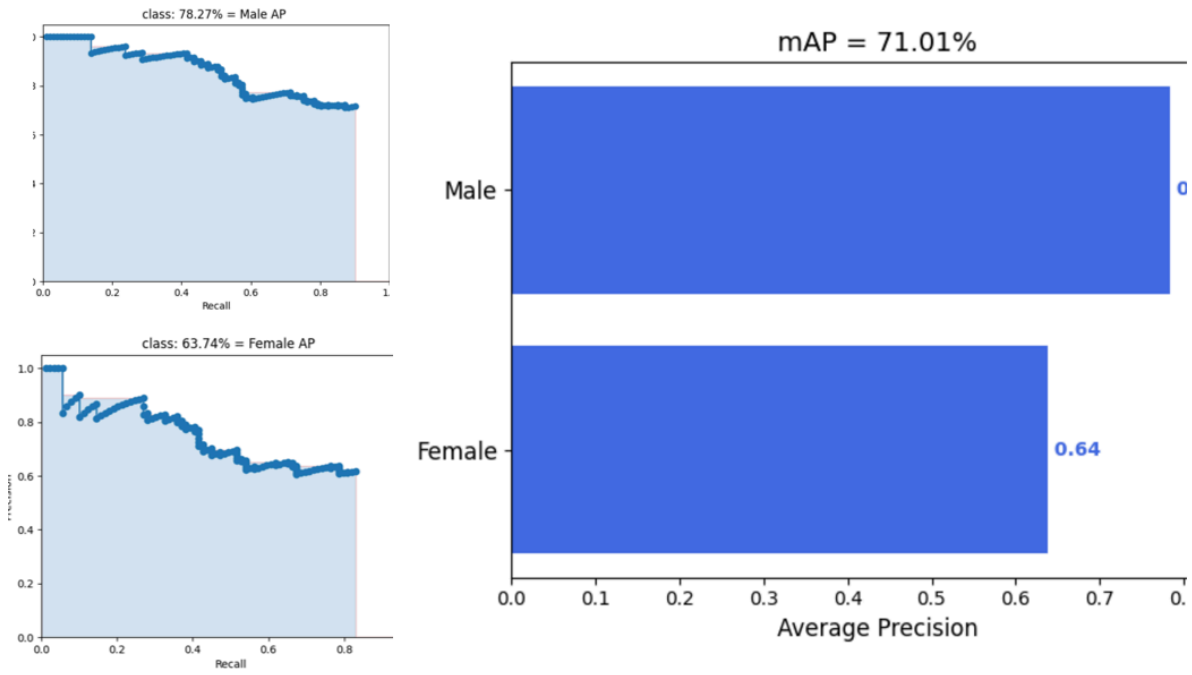
Detection of male/female images

The training accuracy reaches around 95%, while the test accuracy reaches about 80% accuracy. This is in line with the test accuracy when object detection is being run, which is around 80% as well, shown roughly by the ratio of the total number of true positives to the total number of images.



Classification Accuracy

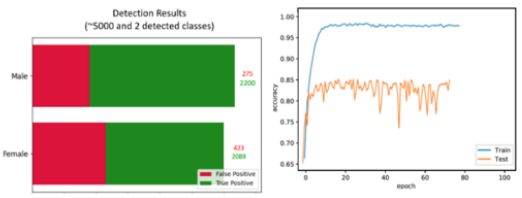
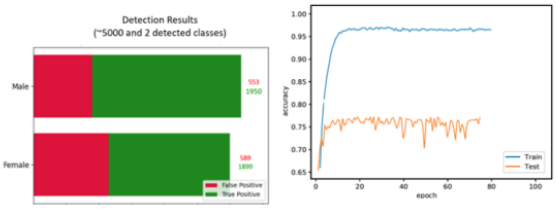
As for the mAP, it reaches about 70%, which is reasonable.



mAP of model

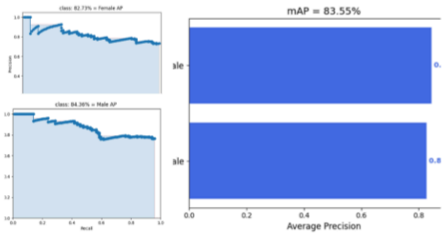
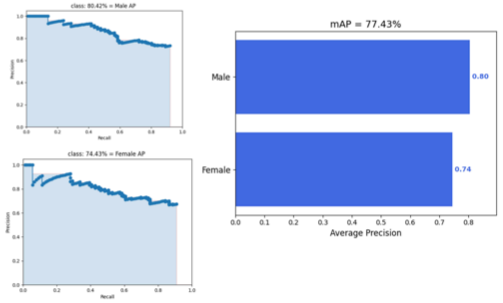
After which the weights are saved and used to train a new model using the Adience Dataset. The pretrained CelebA weights are saved in a h5 file, which will then be uploaded to be used as weights when training a new model on Adience dataset. The importance of using pre-trained weights from CelebA here is due to the idea of transfer learning, which will benefit in the training of the new model by a lot. This is because the weights are already adjusted based on the feature nuances on gender differences, and when these weights are being used to train a new model on Adience dataset, the adjusting of the weights would be quick and not that significant, allowing for better training. There will be comparisons below to prove the importance of using CelebA pre-trained weights.

The pretrained CelebA YOLO model was used to train with an Adience training dataset, then testing on a test dataset versus a random ImageNet YOLO model to train with Adience training dataset, then testing on the same test dataset.

CelebA Pretrained Model Test Accuracy	Normal ImageNet Model Test Accuracy
Test Accuracy: About 85% 	Test Accuracy: About 75% 

Comparison of Test Accuracy for pre-trained model versus a normal one

There is a higher test accuracy when CelebA pre-trained weights are used instead of just a normal ImageNet model. The difference in the accuracy is about 10%, which is a substantial amount.

CelebA Pretrained Model mAP	Normal ImageNet Model mAP
mAP: About 83.55% 	mAP: About 77.43% 

Comparison of mAP between CelebA pretraining versus a normal one

There is a better mAP when CelebA pre-trained weights are used instead of just a normal ImageNet model. The difference in the accuracy is about 5%, which is a reasonable amount.

Discussion

From the results above, we can see the importance of pre-training to improve the results of gender classification and then detection. Pre-training allows the model to attune its weights according to the class that it is going to classify or detect.

Gender classification

Methods

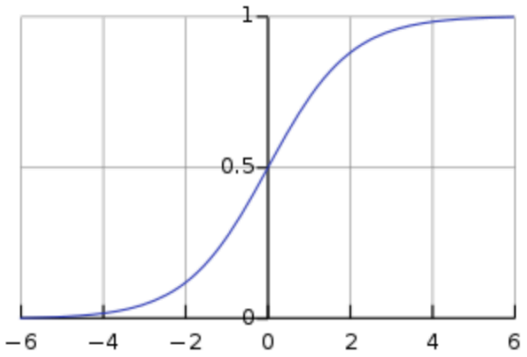
The CNN architecture for gender classification we designed consists of a novel twelve-layer network consisting of three convolutional layers, three maxpool layers, three dropout layers and finally 2 fully connected layers. The CNN design is an end-to-end sequential deep learning architecture, including feature extraction and classification phases. The feature extraction phase has three convolutional layers, with the corresponding parameters, including the number of filters, the kernel size of each filter, and the padding. It contains the convolutional layer, activation layer (rectified linear unit (ReLU)), max-pooling layer, and a dropout layer at a dropout ratio of 0.1 after the second and third convolutional layers. The classification stage, on the other hand, contains two fully connected layers, that handle the classification phase of the model. The first fully connected layer contains 100 neurons, followed by a ReLU and finally, a dropout layer at a dropout ratio of 0.2. The second and the last fully connected layer is densely mapped to 1 neuron for classification tasks. The details of the CNN structure are presented in the figure below.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 227, 227, 50)      1400
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 113, 113, 50)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 113, 113, 70)      31570
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 70)        0
_________________________________________________________________
dropout (Dropout)            (None, 56, 56, 70)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 90)        56790
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 90)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 28, 90)        0
_________________________________________________________________
flatten (Flatten)            (None, 70560)             0
_________________________________________________________________
dense (Dense)                (None, 100)               7056100
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 7,145,961
Trainable params: 7,145,961
Non-trainable params: 0
_________________________________________________________________
```

Model architecture

We model gender classification task as an end-to-end deep classification problem; therefore, a sigmoid with binary cross entropy loss function is adopted to obtain a probability for each gender class.

We have used sigmoid activation function in the output layer because it is the appropriate choice for binary classification with high accuracy. The benefit of using sigmoid function is that the output ranges between 0 and 1, we can set a decision boundary and determine whether the label was 0 or 1 easily.



Sigmoid function graph

In general, softmax activation with cross entropy loss is used for classification as an output layer activation because softmax activation distributes the probability throughout each output node. However, softmax results turned out to be really poor as our problem statement involves a binary classification. Sigmoid is known to perform better for binary classification and softmax performs well for multi classification, and for our case as well, sigmoid activation with binary cross entropy loss gave better results and effective training.

Cross-entropy is the default loss function to use for binary classification problems. Binary cross-entropy loss is used in our CNN as it is intended for use with binary classification where the target values are in the set {0,1}.

We have used RMSprop (**R**oot **M**ean **S**quare **Prop**agation) as an optimizer for the CNN because it is a very robust optimizer which has pseudo curvature information. Additionally, it can deal with stochastic objectives very nicely, making it applicable to mini batch learning. Other optimizers like SGD and Adam are also comparable to RMSprop's performance for our model, however, RMSprop performs slightly better than other optimizers and gives the best accuracy.

We have also used dropout layers of 0.1 probability after some conv2D layers and dropout layer of 0.2 probability after the fully connected layer in order to prevent overfitting of the model. Because the outputs of a layer under dropout are randomly subsampled, it has the effect of reducing the capacity or thinning the network during training.

Experiments and Results

- **Preprocessing**

  The Adience dataset consists of a folder 'aligned.tar.gz' (1.9G) - Face images, cropped and aligned using our 2D, in-plain alignment tool which is being trained and evaluated in our model. There are text files

fold_0_data.txt - fold_4_data.txt - with indices to the five-fold cross validation tests using all faces which consists of target labels for each image in the column ‘gender’. In order to prepare the dataset to train through the CNN, first we load each image from each folder in the ‘aligned.tar.gz’ folder and convert the image to a numpy array. The array is then resized to (227, 227, 3). The text files are all combined into one csv file which has data from all the text files corresponding to each image in the ‘aligned.tar.gz’ folder. The csv file is then read into a dataframe and a new column consisting of the image numpy arrays are then merged into the same dataframe where the image file name from the ‘aligned.tar.gz’ folder matches the ‘original_image’ and ‘face_id’ column attributes (the filenames of images in the ‘aligned.tar.gz’ folder are a combination of ‘face_id’ and ‘original_image’ attributes).

After forming the combined dataframe, all the rows which have an empty cell for ‘gender’ column are removed from the dataset. All the rows with ‘u’ value are also dropped. Furthermore, the image arrays are normalized by dividing pixel values by 255.0. And finally, the ‘gender’ column consisting of two classes ‘m’ and ‘f’ (meaning male and female) are encoded to 0 and 1 integer values respectively.

- **Training**

  Now, to train the dataset, we use the ‘gender’ column as target label (y) and the new column ‘image_array’ as the training data (X). The whole dataset is split into training dataset and validation dataset with the ratio 0.33 using ‘train_test_split’ function from ‘sklearn’ library. The validation dataset is used to evaluate how well our model is performing.

  Next, the CNN is designed in the following manner:

  - ➢ An input layer of 227 x 227 x 3 dimensions
  - ➢ A convolution layer C1 with 50 channels, window size 3x3, SAME padding, and ReLU activation
  - ➢ A max pooling layer S1 with default settings
  - ➢ A convolution layer C2 with 70 channels, window size 3x3, SAME padding, and ReLU activation
  - ➢ A max pooling layer S2 with default settings
  - ➢ A dropout layer D1 with dropout ratio at 0.1
  - ➢ A convolution layer C3 with 90 channels, window size 3x3, SAME padding, and ReLU activation
  - ➢ A max pooling layer S3 with default settings
  - ➢ A dropout layer D2 with dropout ratio at 0.1
  - ➢ A fully connected layer F1 of size 100 with ReLU activation

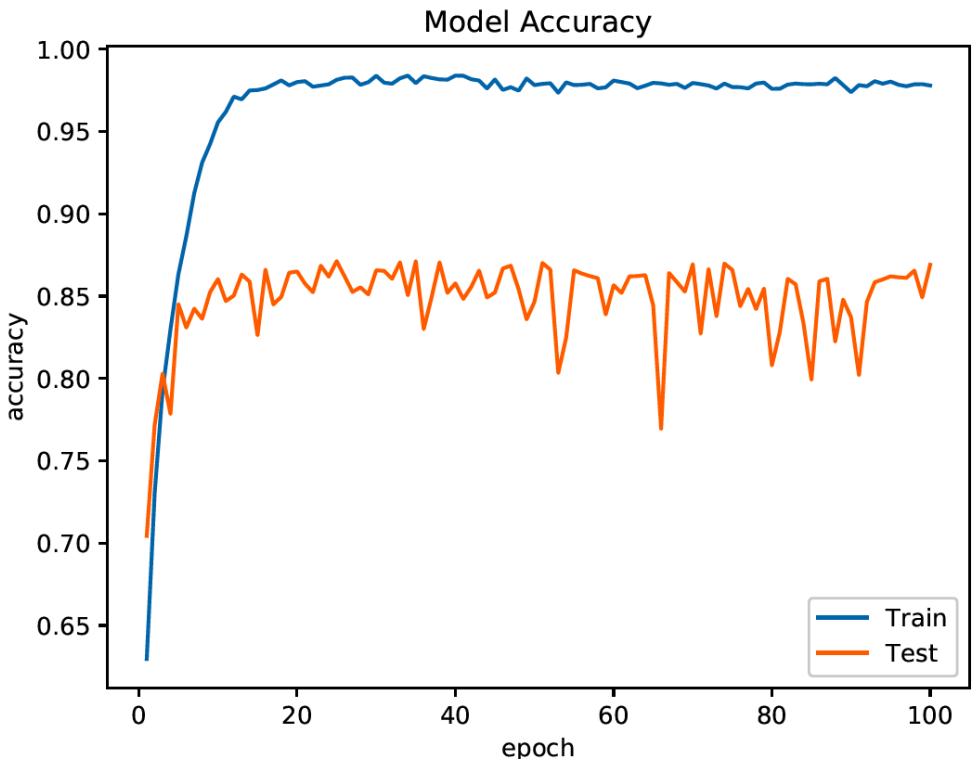
➢ A dropout layer D3 with dropout ratio at 0.2
➢ A fully connected layer F2 of size 1 with sigmoid activation

Parameters for model compilation:

➢ Optimizer chosen is RMSprop
➢ Learning rate is 0.001
➢ No. of epochs for training is fixed are 100
➢ Batch size is fixed at 64
➢ Loss function selected is BinaryCrossentropy()

- **Results**

Test and train and accuracy results for model with RMSprop optimizer.

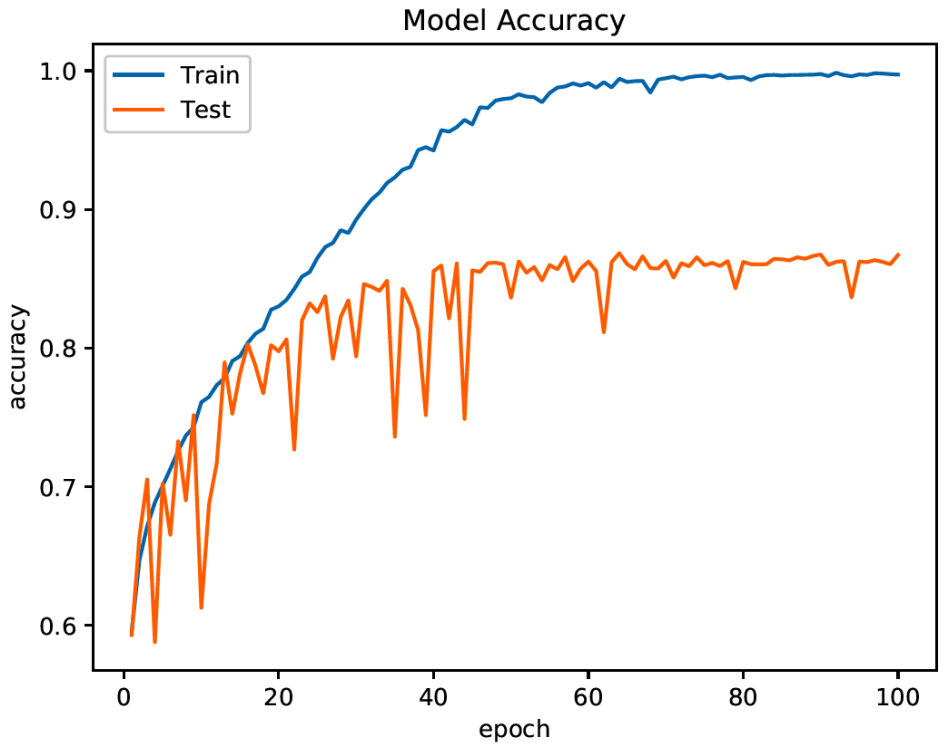


accuracy for RMSprop optimizer plot

```
Epoch 97/100
172/172 [==============================] - 16s 94ms/step - loss: 0.0855 - accuracy: 0.9775 - val_loss: 0.8631 - val_accuracy: 0.8611
Epoch 98/100
172/172 [==============================] - 16s 94ms/step - loss: 0.0830 - accuracy: 0.9786 - val_loss: 0.9798 - val_accuracy: 0.8655
Epoch 99/100
172/172 [==============================] - 17s 100ms/step - loss: 0.0726 - accuracy: 0.9787 - val_loss: 0.9664 - val_accuracy: 0.8492
Epoch 100/100
172/172 [==============================] - 16s 92ms/step - loss: 0.0880 - accuracy: 0.9779 - val_loss: 1.1541 - val_accuracy: 0.8690
```

accuracy after 100 epochs

Test and train and accuracy results for model with SGD optimizer.



accuracy for SGD optimizer plot

```
Epoch 97/100
172/172 [==============================] - 15s 89ms/step - loss: 0.0066 - accuracy: 0.9982 - val_loss: 0.6848 - val_accuracy: 0.8635
Epoch 98/100
172/172 [==============================] - 15s 89ms/step - loss: 0.0076 - accuracy: 0.9980 - val_loss: 0.6607 - val_accuracy: 0.8622
Epoch 99/100
172/172 [==============================] - 15s 89ms/step - loss: 0.0087 - accuracy: 0.9975 - val_loss: 0.6965 - val_accuracy: 0.8605
Epoch 100/100
172/172 [==============================] - 15s 90ms/step - loss: 0.0085 - accuracy: 0.9973 - val_loss: 0.6568 - val_accuracy: 0.8672
```

accuracy after 100 epochs

Discussion

We can see that both SGD and RMSprop optimizers' performances are at par with each other. The only differences we can notice between SGD model and RMSprop model is that SGD starts learning slowly and the accuracy curve reaches higher accuracies after more number of epochs as compared to RMSprop. However, SGD has lesser gradient spiking towards the end epochs and is more stable in comparison to RMSprop. RMSprop on the other hand learns very efficiently since the first few epochs and remains around the same high accuracy throughout the 100 epochs but has much more gradient spiking towards the end and is not as stable as SGD.

Introducing dropouts after convolutional layers gave better accuracy and higher stability in the graph plots as compared to model trained without dropouts. This

is because the model was prevented from overfitting due to dropouts and hence provides better predictions.

The best model achieved is with RMSprop as after 100 epochs it has the best accuracy after 100 epochs of training which is 0.869 i.e. around 86% (0.869 by RMSprop vs 0.8672 by SGD).

Age classification

Methods

Our CNN architecture for age classification consists of six layers: two convolutional layers, two maxpool layers and finally two fully connected layers. The CNN design is an end-to-end sequential deep learning architecture. The parameters of the layer are specified in the "Training" section.

For the output layer, softmax activation with sparse categorical cross entropy loss function is used. Softmax activation function is widely used for classification as an output layer activation because softmax activation distributes the probability throughout each output node. Sparse categorical cross entropy is used since there are more than two label classes and the classes are mutually exclusive.

Adam (Adaptive Moment Estimation) optimizer is used for this CNN because it is a very robust optimizer which is computationally efficient, memory space friendly and works well with large data sets.

Experiments and Results

- **Preprocessing**

  The Adience dataset consists of a folder ‘aligned.tar.gz’ (1.9G) - Face images, cropped and aligned using our 2D, in-plain alignment tool which is being trained and evaluated in our model. The 'aligned_faces_data.csv' files are used to extract the raw age range for each image. In order to improve processing time, instead of loading all the images at once, we divide the images into smaller batches (same batch size as used in the code), and those image batches are loaded as soon as they are fed into the model.

  There are said to be eight default age groups (0-2, 4-6, 8-13, 15-20, 25-32, 38-43, 48-53, 60+). However, there are some data in the 'aligned_faces_data.csv' file that does not immediately belong to these

age ranges (for example, 8-12), so we have to scale them to fit into the eight default age labels.

Label Encoder is used to encode the output labels.

- **Training**

  Now, to train the dataset, we use the ‘age’ column as target label (y). The whole dataset is split into training dataset and validation dataset with the ratio 0.33 using ‘train_test_split’ function from ‘sklearn’ library. The validation dataset is used to evaluate how well our model is performing.

  Next, the CNN is designed in the following manner:

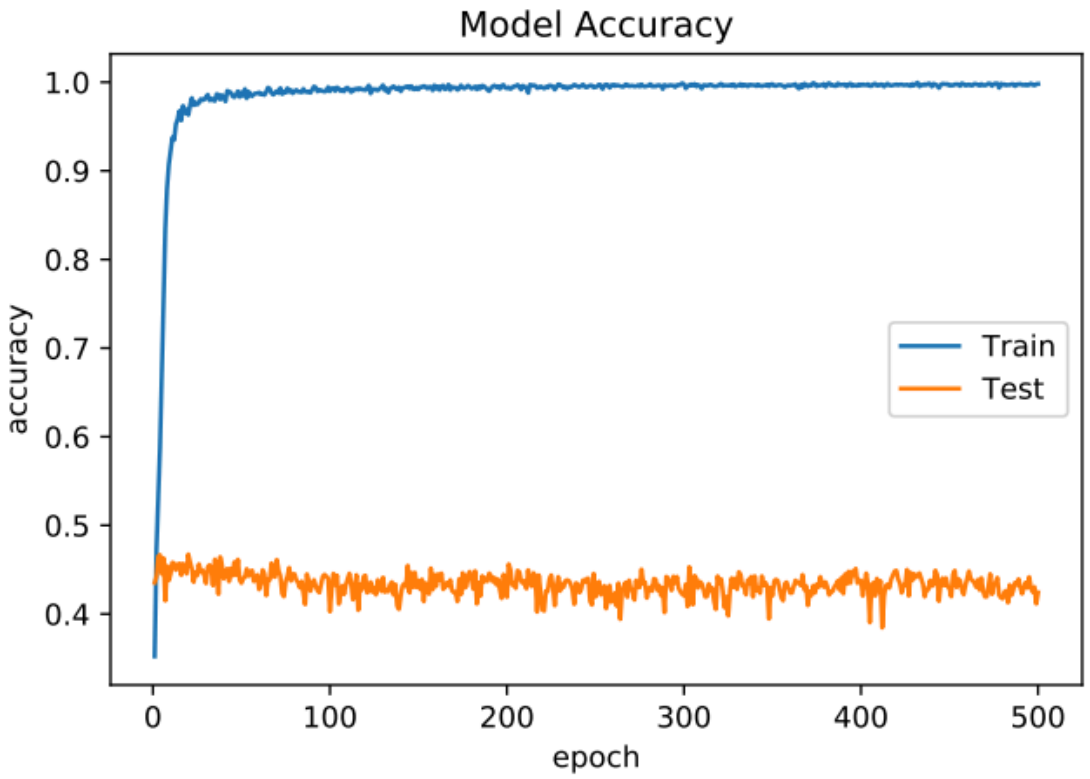
  - ➢ An input layer of 227 x 227 x 3 dimensions
  - ➢ A convolution layer C1 with 50 channels, window size 9x9, VALID padding, and ReLU activation.
  - ➢ A max pooling layer S1 with pool size 2x2, stride 2x2, VALID padding.
  - ➢ A convolution layer C2 with 60 channels, window size 5x5, VALID padding, and ReLU activation.
  - ➢ A max pooling layer S2 with pool size 2x2, stride 2x2, VALID padding.
  - ➢ A fully connected layer F1 of size 300.
  - ➢ A fully connected layer F2 of size 12 with softmax activation.

  Parameters for model compilation:

  - ➢ Optimizer chosen is Adam
  - ➢ Learning rate is 0.001
  - ➢ No. of epochs for training is fixed are 500
  - ➢ Batch size is fixed at 32
  - ➢ Loss function selected is SparseCategoricalCrossentropy()

- **Results**

  Test and train and accuracy results for the model with Adam optimizer:



accuracy for Adam optimizer plot

Discussion

The test accuracy begins to stabilize at around 80 epochs, at an accuracy of approximately 0.47.

We had tried to include every age value available in the 'aligned_faces_data.csv' file before deciding to fit them into the one of the eight default age labels. This resulted in a poorer performance (around 0.30 accuracy).

Some other methods to improve the model that can be considered in the future are data augmentation for preprocessing, or trying other optimizers such as RMSProp.

Conclusion

In conclusion, the normal CNN model in the second part beats the YOLOv3 model when it comes to the classification of the gender differences on the Adience test dataset. This shows that a deep layer of convolutional layers sometimes may not help, it is about the data that it is being trained on. As YOLOv3 trains on fewer images than the normal CNN, that resulted in the differences. As for age prediction, it is optimal and a reasonable percentage.

Further improvements to the project could be to use it on classification of gender differences when masks are on, in view of the pandemic we are living in.

References

- G. Levi and T. Hassner, “Age and gender classification using convolutional neural networks.” in IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) workshops, 2015
- Z. Liu and P. Luo and X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in International Conference on Computer Vision (ICCV), 2015
- Eran Eidinger, Roee Enbar, Tal Hassner. Age and Gender Estimation of Unfiltered Faces. Transactions on Information Forensics and Security (IEEE-TIFS), special issue on Facial Biometrics in the Wild, Volume 9, Issue 12, pages 2170 - 2179, 2014.
- Understanding CNN (Convolutional Neural Network). (2020). Retrieved 23 November 2020, from https://towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4
- Convolutional Neural Network (CNN) | TensorFlow Core. (2020). Retrieved 23 November 2020, from https://www.tensorflow.org/tutorials/images/cnn
- Redmon, J. and Farhadi, A., 2018. Yolov3: An incremental improvement. arXiv 2018. *arXiv preprint arXiv:1804.02767*, pp.1-6.