Age, Gender and Ethnicity Classification from Image

Neural Network Seminar
FS 2020

Rohit Kaushik (17-747-668)
Pratyush Singh (19-762-988)
Ankush Panwar (19-763-051)

Table of Contents

1. Introduction

1.1 Project Description

In this project we aim to explore a dataset of images of people from different backgrounds and build a neural network image classifier to classify the age, gender and ethnicity of people in images. We deal with three different tasks in the project. These are gender classification which is a binary classification problem while age prediction is a regression problem since the output of the network can be any discrete number. Our last problem is that of ethnicity classification which is a multi class classification problem. All these tasks are important and useful in several fields and hence developing an efficient classification model can be helpful in solving the problems. For example, classifying ethnicity is an important task to solve for federal authorities and many companies developing photo editing applications.

1.2 Dataset

We chose the dataset from a kaggle competition on age, gender and ethnicity classification. The dataset is stored as a csv file and all the information including images are present in the csv. The CSV file contains facial images in grayscale that are labeled on the basis of age, gender, and ethnicity. The dataset includes 27305 rows/data-points and 5 columns/features, namely, age, gender, ethnicity, img_name, pixels. We don't need the img_name for our project since it doesn't provide any useful information so we first drop this column.

The total size of our dataset is 191MB. Since the dataset is quite small in size, it was possible for us to train it on our local machines and GPU. Each of the image are of dimension 48 * 48. These images are reshaped as 48 * 48 * 1. The extra dimension denotes the color channel which in our case is 1 since the images are grayscale images and is needed by neural network libraries.

The table below describes the range and data type of each of the feature:

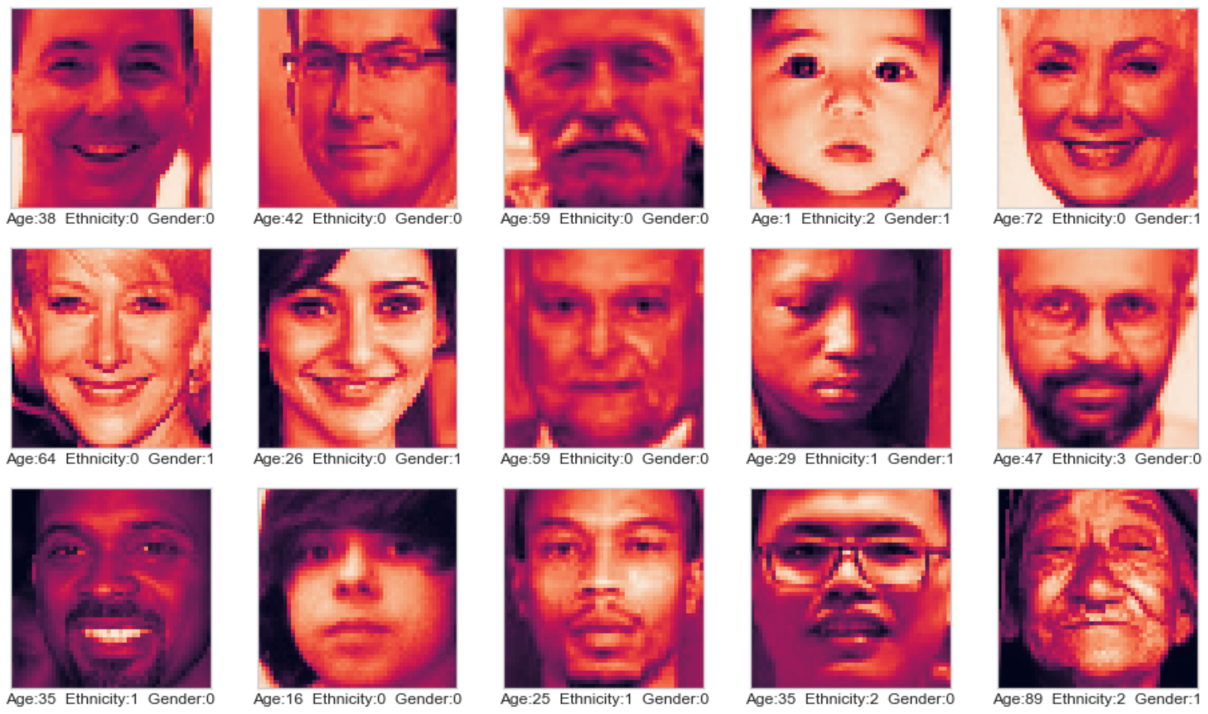
Age	Gender	Ethnicity	Img_name	Pixels
Integer [1-116]	Integer (0,1)	Integer [0-4]	String	String representatio n of 2d array

The dataset is present on kaggle competition page and can be downloaded by anyone from here[1]

1.3 Dataset description

In this section we give a brief overview of the dataset and present some sample images. A more detailed analysis is presented in section 2. EDA.

The below figure is a sub sample of images present in the dataset.


Age:38 Ethnicity:0 Gender:0
Age:42 Ethnicity:0 Gender:0
Age:59 Ethnicity:0 Gender:0
Age:1 Ethnicity:2 Gender:1
Age:72 Ethnicity:0 Gender:1
Age:64 Ethnicity:0 Gender:1
Age:26 Ethnicity:0 Gender:1
Age:59 Ethnicity:0 Gender:0
Age:29 Ethnicity:1 Gender:1
Age:47 Ethnicity:3 Gender:0
Age:35 Ethnicity:1 Gender:0
Age:16 Ethnicity:0 Gender:0
Age:25 Ethnicity:1 Gender:0
Age:35 Ethnicity:2 Gender:0
Age:89 Ethnicity:2 Gender:1

The dataset contains images of people with age varying from 1 to 119. The range of age in our dataset is large and is also not uniformly distributed. Some age occurs only once and hence we treat them as an outlier.

Similarly for ethnicity, our dataset contains images of people belonging to 5 different ethnicities and these are White, Black, Asian, Indian, Hispanic. The data for ethnicity is also imbalanced and there are more images for White and least for Hispanic ethnicity. To overcome this problem of imbalanced data, we assign a weight to each ethnicity class based on the fraction of data present and then train our model.

[1] https://www.kaggle.com/nipunarora8/age-gender-and-ethnicity-face-data-csv

The dataset has fairly balanced images for both genders: Male and Female. There are equal number images for both Male and Female. So we don’t perform any preprocessing in this case.

1.4 Methodology

We use tensorflow keras library to build our neural network architecture. The model architectures and parameters would be discussed in the later section. We build 3 different models, one corresponding to our task of age prediction, gender classification, ethnicity classification. The implementation of the model and training is done in the train.ipynb notebook. We store the best model based on the validation accuracy and the early stopping is used to overcome overfitting.

For the notebook and training to work, a folder named data should be created and the age_gender.csv data file should be copied to this folder.

We also do some exploratory data analysis to understand the data distribution. The relevant code is present in the EDA notebook.

Finally, we perform grid search to tune hyperparameters of the network. The result of training and hyperparameter tuning is present in the later section 4.

For implementation, we use tensorflow keras library for building the models and we use plotly and matplotlib for the visualization.

2. Exploratory data analysis

In this section we present a more detailed analysis of our dataset and their distributions. We also present some visualizations that help us understand the dataset better. The data analysis is an important part of our project since we were able to decide what preprocessing is necessary.

Age Distribution

We do a descriptive analysis of the age distribution present in our data. The aim of the analysis was to understand outliers and how to deal with it.

We first plot a bar plot of the age present in our data. As observed and can be seen from the fig 1, there are much less images of people with age more than 80. This might affect our regression model. To further analyze this, we present a box plot and some statistical measure of the distribution.

The bar plot for age distribution:

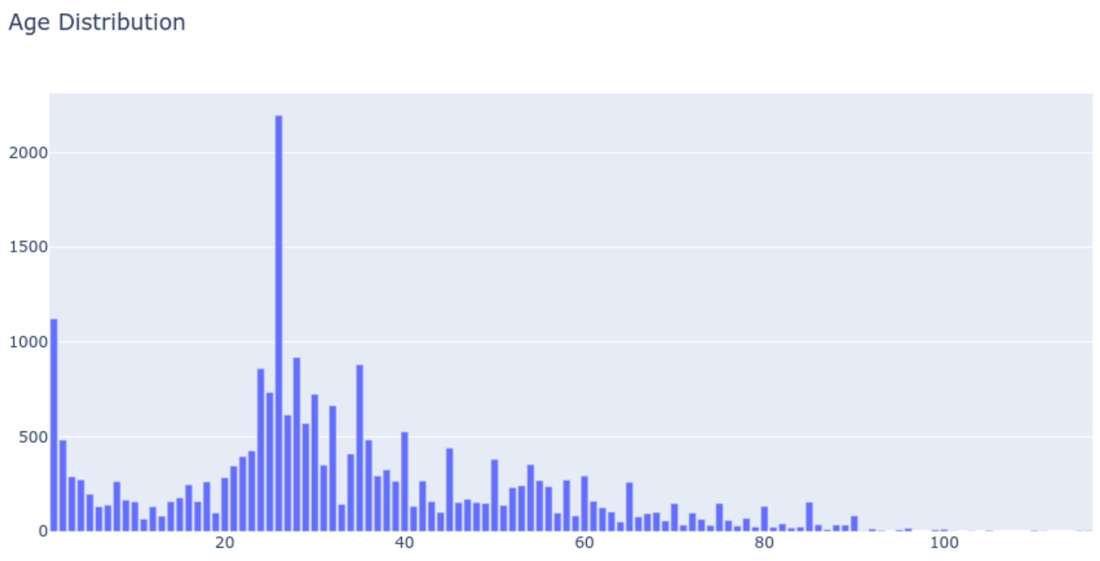


Fig1. age distribution

With the box plot, we aim to get a better understanding of outliers based on quartiles. As you can see from the plot Fig2 below, the data where age is more than 80 can be seen as outliers. So we remove these instances from our training.

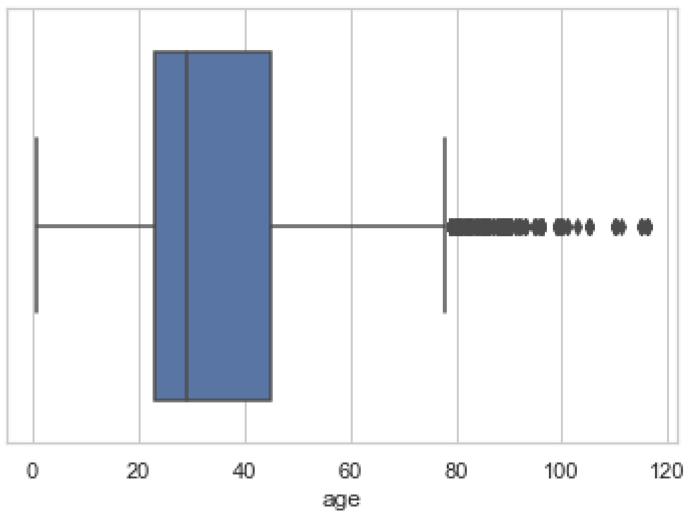


Fig2. age outliers

In the table1 below, we present a more quantitative result from our exploration of age in the data.

count	23705
mean	33.30097
std	19.885708
min	1
25%	23
50%	29
75%	45
max	116

Table.1 age statistics

Ethnicity & Gender

There wasn't much scope of data analysis for ethnicity and gender, so we present two different figures of the counts of both the features present in the data. As can be seen from the plots, ethnicity data is quite imbalanced whereas gender data is relatively balanced. Based on this we decide to regularize our ethnicity model training by assigning weights to the ethnicities classes.

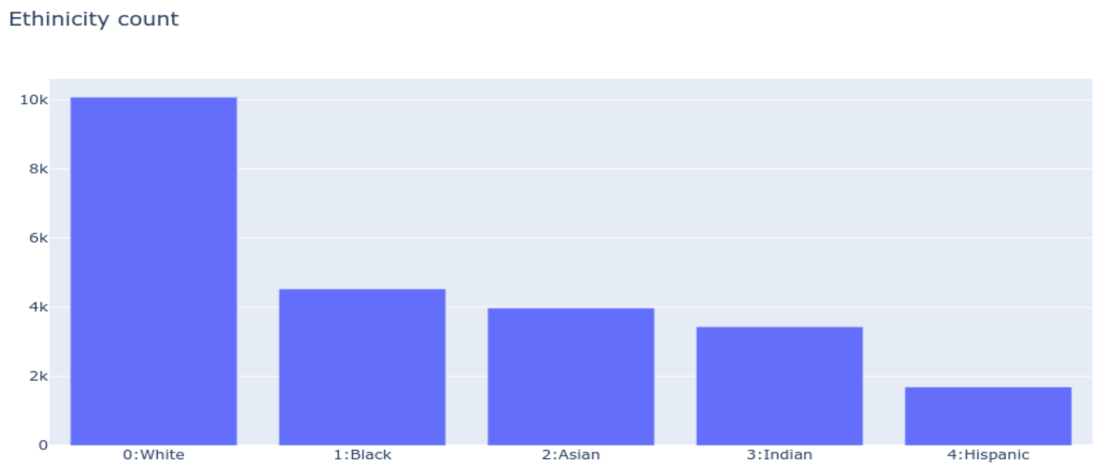


Fig3. Ethnicity count

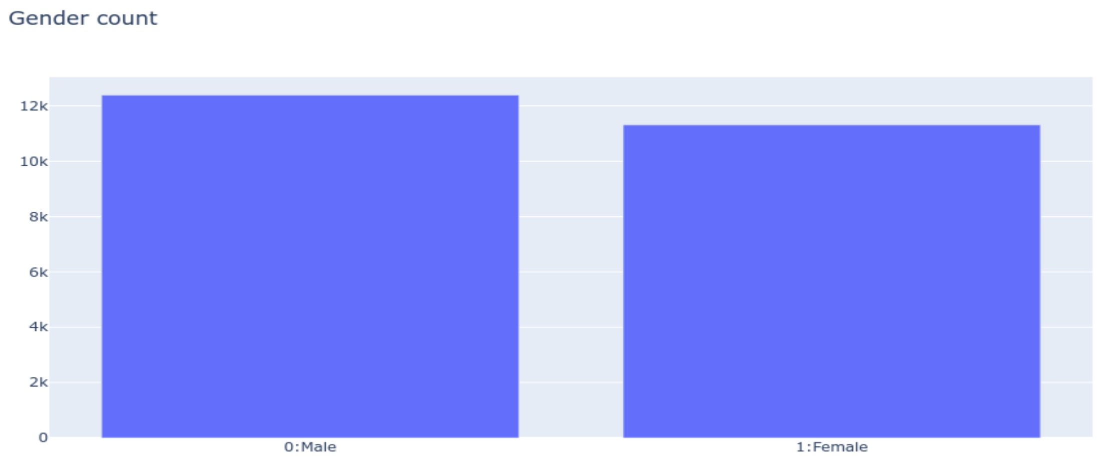


Fig3. Gender count

3. Data Preprocessing

As discussed in the previous section, the problem of class imbalance and outliers was present in the dataset. To deal with the problem, we remove instances of age that are more than 80. There is not much scope of preprocessing apart from the outlier removal. We also make sure that distribution of data remains same in the training and testing set. For that we use sklearn train_test_split function with stratification which ensures the split is made with the same proportion.

The age distribution without the outliers are shown in Fig4 below

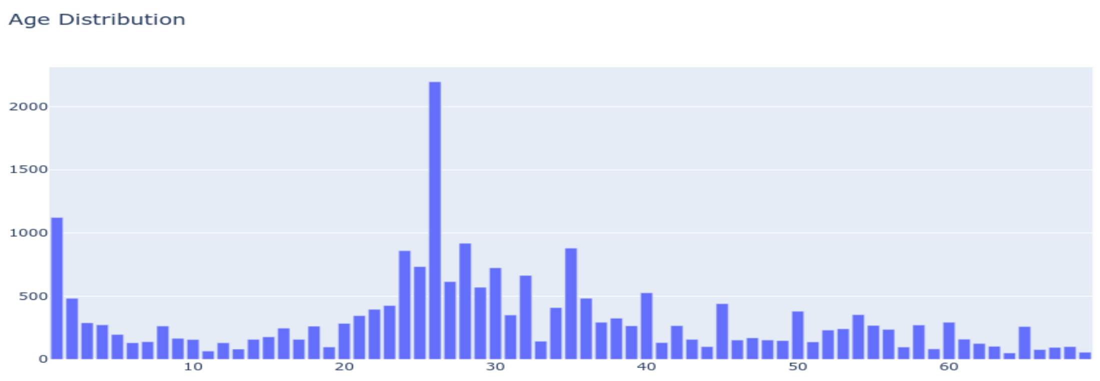


Fig4. age distribution without outlier

3.1 Data Splitting

We split our dataset into a train and test set. The ratio of the spit was 80:20. The split was made such that both train and test set have similar distribution. We also use k-fold cross validation to reduce overfitting.

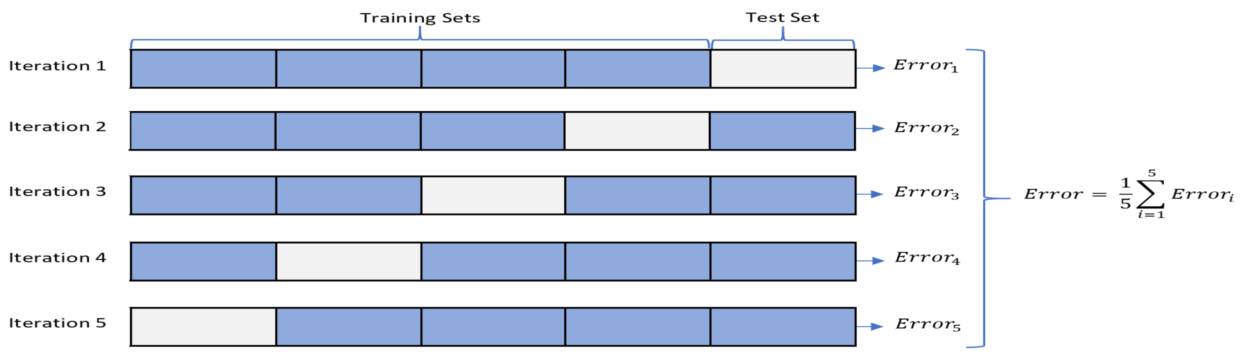


Fig5. 5-fold cross validation

4. Model Discussion and Implementation details

As discussed in the Introduction section, we want to train 3 different classifiers for the 3 different classification tasks namely, age prediction, gender classification and ethnicity classification. The reason we don't use a multi-task approach is because multi-task work only when the tasks are related and the learned representation can help other tasks. We dont believe learned representation for age can help with gender prediction. Also our data presents different challenges for each feature and hence preprocessing needs to be used.

The 3 different neural network architecture are defined in the function

get_age_classifier_model

get_ethnicity_classifier_model

get_gender_classifier_model

In the next section we discuss each of the architecture separately and the results that we were able to achieve with the training.

4.1 Age predictor model

The problem of predicting the age is the regression problem trained through a CNN. We predict the rounded integer as the age. The model takes input to the image array and passes it through the series Convolutional layer and Max pooling layer. We use Batch Normalization and Dropout of 5% to train the model. Dropout helps in complex co-adaptations of the neighboring neurons and avoid any potential overfitting. We use the ReLU activation function. The mean absolute error is used as the evaluation metric and mean squared error loss as the loss function while optimizer is ‘adam’ optimizer.

Layers of the age prediction model are:

Conv2D >> BatchNormalization >> MaxPool >> Conv2D >> Max Pooling >>

Flatten >> Dense >> Dropout >> Dense

Parameters of the final model:

Optimizer: ‘adam’

Loss: ‘mean squared error’

Accuracy metric : ‘mean absolute error’

Dropout Rate: 0.5

Activations: Relu

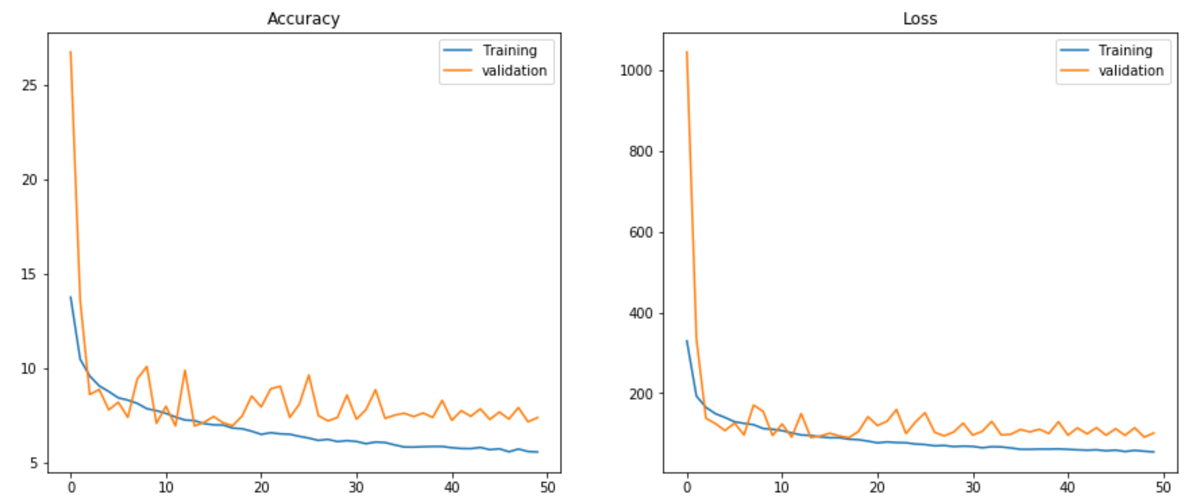


Fig6. accuracy and loss

Our age prediction model had a mean absolute error of 5.5547 on the train set and on the validation set it had an error of 7.3747.

4.2 Ethnicity predictor model

We use a similar CNN model as in the case of age predictor but with a 'rmsprop' optimizer and Categorical Cross Entropy loss to train the model. We use Categorical Cross Entropy since it is a multiclass classification problem.

In the ethnicity model, we also try using class weights for each of the ethnicity classes to account for imbalance in data. The validation accuracy without using class weights was 69% whereas when we use class weights the accuracy increases to 76%. On the test set we get an accuracy of 77%. We now describe the architecture and parameters for our final model.

Layers of the prediction model are:

Conv2D >> MaxPool >> Conv2D >> Max Pooling >> Dropout >> Conv2D >>

Max Pool >> Conv2D >> MaxPoo l>> Flatten >> Dense >> Dropout >> Dense

Parameters of the final model:

Optimizer: ‘adam’

Loss: ‘sparse categorical cross entropy’

Accuracy metric : ‘accuracy’

Dropout Rate: 0.3 & 0.5

Activations: Relu

Below Fig7. describes how loss and accuracy changes over epochs for the ethnicity classification model.

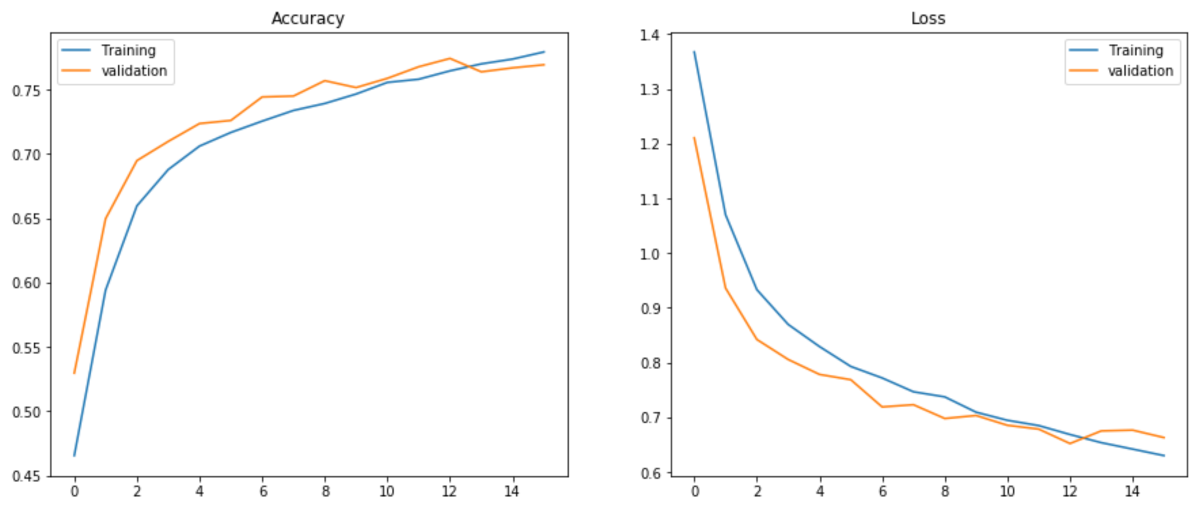


Fig7. loss and accuracy

As in a classification task, accuracy is not the only metric but rather it is important to analyze the recall and precision of the model, below we present class wise accuracy and precision. Fig8 shows the confusion matrix for our model.

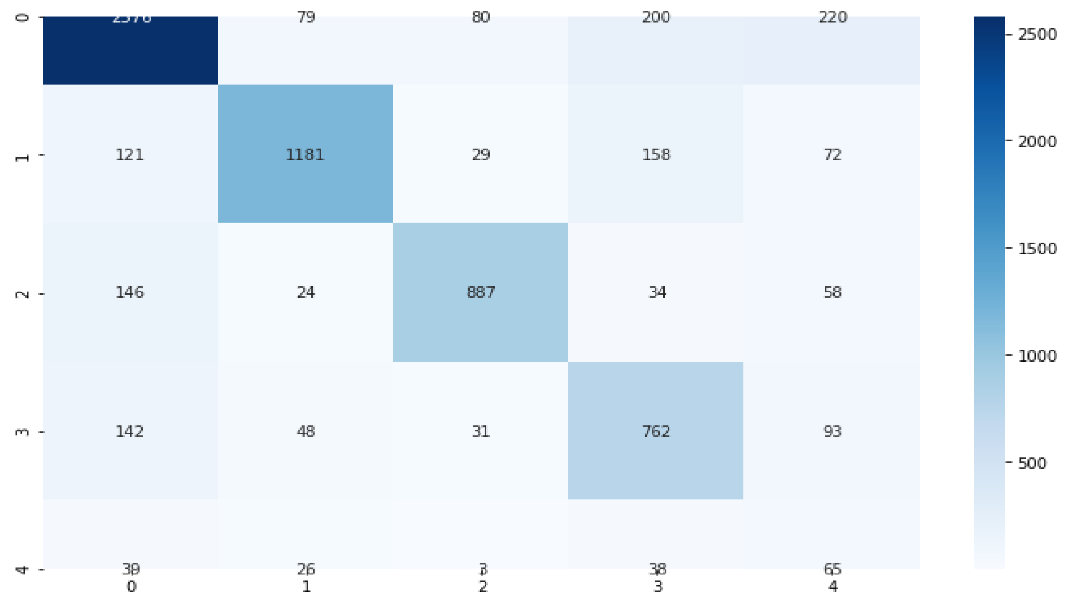


Fig8. Confusion matrix

class	precision	recall	F1 score	support
0	0.85	0.82	0.83	3155
1	0.87	0.76	0.81	1561
2	0.86	0.77	0.81	1149
3	0.64	0.71	0.67	1076
4	0.13	0.38	0.19	171
accuracy			0.77	7112
Macro avg	0.67	0.69	0.66	7112

Table 2. Ethnicity classification score

As can be seen from the table the class 4 that corresponds to Hispanic ethnicity has least accuracy. This is because the least amount of training data is present for ethnicity 4.

4.3 Gender predictor model

Our third model is also similar to the age predictor model but we also use average pooling along with max pooling and binary cross entropy loss. We use binary cross entropy since it is a binary class classification problem.

After removing the outlier, our gender prediction model achieves an accuracy of 89.85% on the train set and 88.40% on the validation set.

Before we discuss the results, we would like to describe the model architecture and final parameters.

Layers of the prediction model are:

Conv2D >> MaxPool >> Conv2D >> Max Pooling >> Dropout >> Conv2D >>

Max Pool >> Flatten >> Dense >> Dropout >> Dense

Parameters of the final model:

Optimizer: ‘adam’

Loss: ‘binary cross entropy’

Accuracy metric : ‘accuracy’

Dropout Rate: 0.3 & 0.5

Activations: Relu

Fig9. shows the accuracy and loss with epoch. As can be seen from the plot, the accuracy of both train and validation increase with epoch and loss decreases.

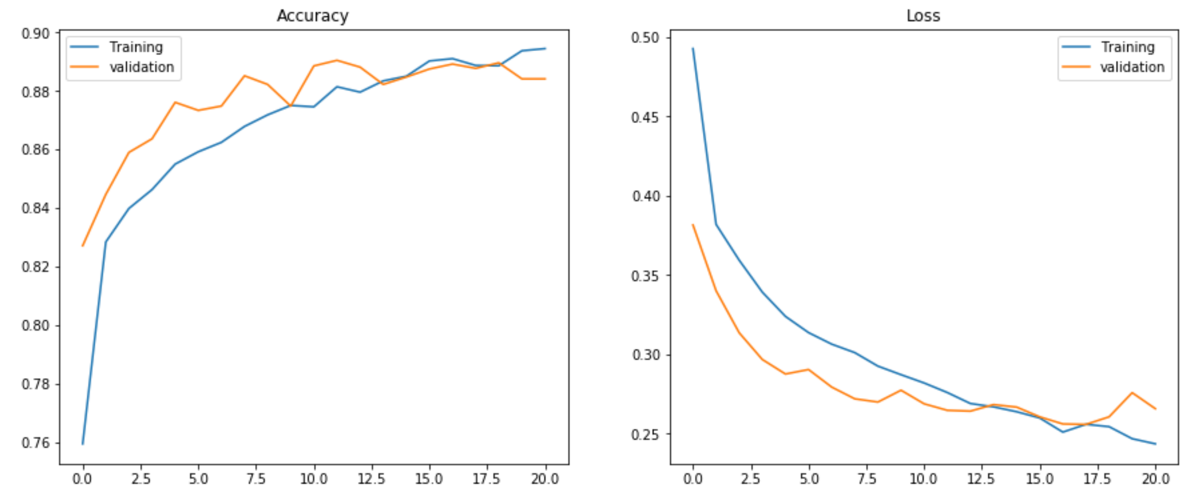


Fig9. loss and accuracy

Below we show the plot of the confusion matrix and also the classification score such as recall and precision for our model.

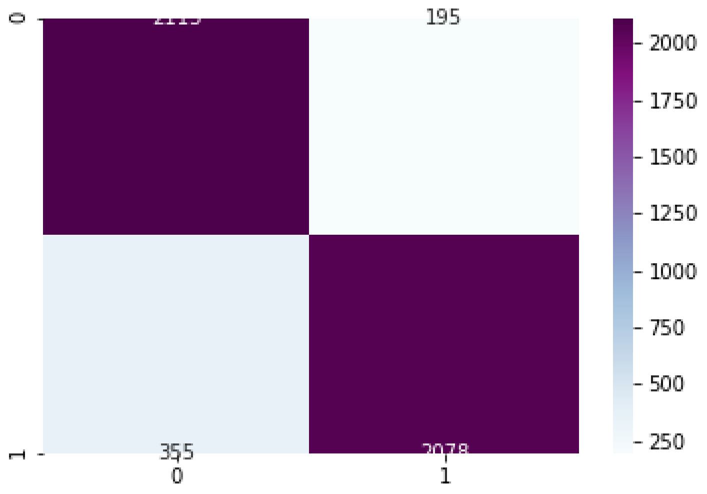


Fig10. Confusion matrix

	precision	recall	f1-score	support
0	0.86	0.92	0.88	2308
1	0.91	0.85	0.88	2433
accuracy	0.89	0.88	0.88	4741
Macro avg	0.89	0.88	0.88	4741

Table3. Gender classification scores

4.4 Hyperparameters

In this section we discuss what parameters we optimize and how we perform hyper-parameter tuning to get the best performing network.

To improve the accuracy of our classification models, we use two different approaches, one is hyperparameter tuning which we do by performing a grid search over parameter value and other by reducing overfitting to the train data.

We use early stopping to avoid training overfit. The early stopping is based on the validation accuracy and loss. We use a patience of 3, that is if the validation accuracy or loss doesn’t improve for 3 consecutive epochs we end the training. We also save the models with the best validation accuracy.

Dropout is another approach that we use to reduce overfitting. We try with different values of dropout. We find that a dropit value of 0.3 works better after the inner flatten layer. Before the final prediction layer we use a dropout layer of 0.5.

To perform hyper-parameter tuning, we do a grid search over some parameters and find the optimal value. We initially start with a large value for epoch (100) since we have early stopping, we wanted to see initially till what epoch does model train and validation accuracy improves. We find that the training starts overfitting even before the 25th epoch. We decided to keep the epoch value as 25.

Our grid search over the batch size (8 to 64) gives an optimal batch size of 32. We also used Batch Normalization since the distribution across the batch can be different and it might lead to overfitting but it negatively affects the performance of our model and hence we remove it from our final model. The other final parameters found through grid search are given below.

The final parameters that we find through grid search are:-

Epoch = 25

Batch Size = 32

Learning Rate = 0.001

Optimizer = Adam

Dropout = 0.3 for inner layer and 0.5 before the final layer

5. Conclusion & Future work

In the project we build 3 different models for our task. Each of the models perform fairly well with ethnicity and gender models performing at an accuracy of 80% and 88% respectively.

We also provide a detailed analysis of the dataset to understand the problem present in the dataset. To tackle these problems, we suggest approaches such as assigning class weights or outlier removal. We were not able to study the approach of data augmentation so that could be a possible area to explore further.

We also perform extensive hyper parameter tuning for our models using Grid Search and the optimal parameters are described in the paper. We find that Batch Normalization doesn’t help with the classification tasks. Adding Dropout of 0.5 worked much better. Also we try with different batch sizes and find that the batch size of 32 was the most optimal.

As a further work we would also like to see if transfer learning can be useful and can help us improve the model accuracy. With the increase in data, pretrained models seem to work much better on the task of classification.