Georgia State University

J. Mack Robinson College of Business

Gender and Age Detection using Deep Learning

May 2020

Payal Sen
Sayali Kardile
Harshitha Shabad
Divija Kambhampati

Abstract

Our report outlines the business use case, architecture and the modelling techniques used for the project. Gender and Age identification from images using Deep Learning is aimed to be utilized for displaying relevant advertisements to customers on smart displays in stores. Images would be captured through the cameras installed in store at various locations. The process flow to implement this project is discussed.

Content

1 Introduction

Age and gender identification from images are important applications widely used in various industries. This task when done by humans seems to be quite easy but being able to implement the same intelligence in computers is quite complex.

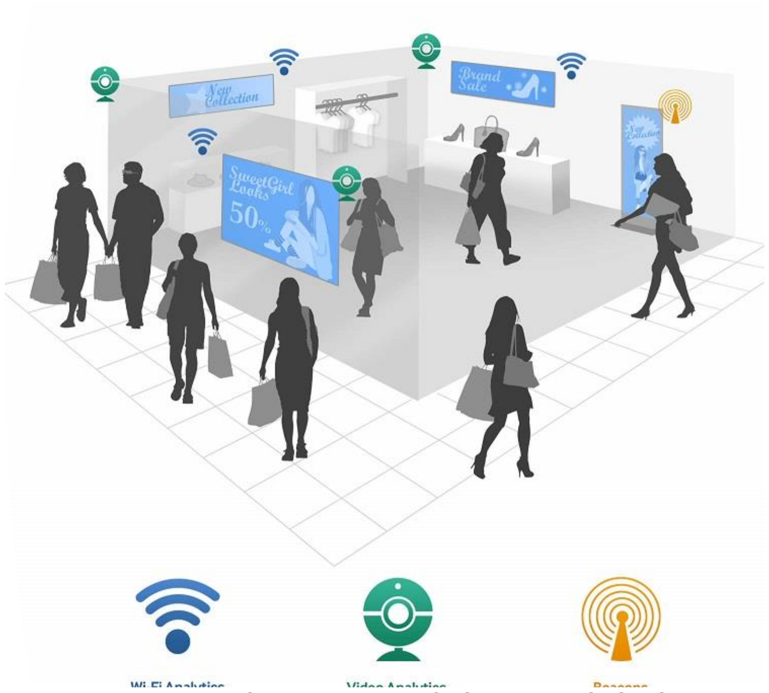


Fig: Customers shopping in a store with advertisements displayed

Advertising plays a significant role in any business to market its products better and make revenue. In this era of modern marketing, we have come up with a technique that can help the stores have improved business. COVID-19 pandemic has forced us to be home and has hugely impacted various sectors of the economy. Most of the retail stores are closed and their business is badly hit with this. As we all recover from this and get back to normal life – we aim to provide all the retail stores better advertising capability.

Every organization has a separate department for marketing. When technology is combined with the advertising concept – it will surely create wonders.

2 Business Scenario

We believe that the solution outlined here can be used for any store. Let us consider a clothing store as an example. Cameras and smart displays can be installed at different spots within a store. Cameras will capture images as customers come in front of them. Once the system takes this image as input, using the image recognition capability – it will predict the Gender and Age. Based on the prediction, billboards will display advertisements relevant to their gender and age.

After COVID-19, as stores come back to function – having such billboards with smart displays will result in a great shopping experience for customers. For instance, if a female of age 25 is around the billboard – any latest styles, accessories relevant to that age group and gender can be displayed on the billboard which can stir positive interest in her to shop them. When the customer gets attracted to buy the products displayed in billboards, it is an advantage for the stores as their business and revenue will increase.

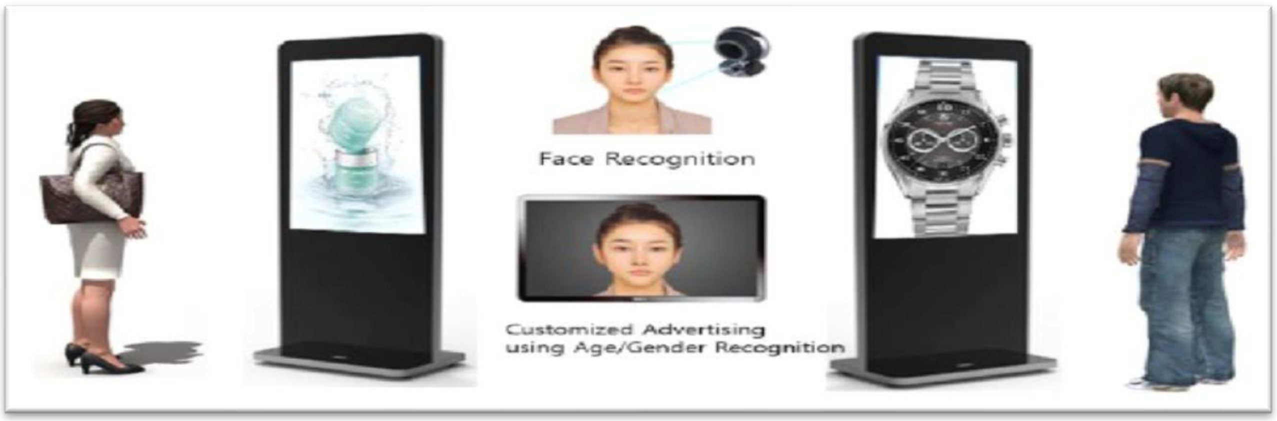


Fig.Smart Display

Displaying appropriate advertisements on billboards relevant to age, the gender of the customer is important to ensure its effective use. Additional factors may also come into consideration for the recommendation aspect. The main purpose of the project is intended to predict the gender and age from an image using an algorithm. However, we strongly believe that if the image recognition capability is coupled with the advertisement recommendation – it will be a great fit in the era of modern marketing.

Below are a few of the many stores which can find this useful:

Fig: Few stores where smart displays can be used

As a second example to understand the use case better, let us consider the store 'Fossil' which is mainly known for its watches. But Fossil also has other products like Wallets, belts, bags, handbags.

If we place these smart displays and display the latest products in all categories to the customer in accordance with age and gender – It will add great value. Firstly, the customer would be aware of the products available and might be interested to buy them. If the sales are increasing for all kinds of products, the store will make better revenue as well.

2.1 Real Time Applications

We would like to mention a few more real-time examples where this technology of Gender & Age prediction has been used.

2.2 Mitra- The Humanoid Robot

Mitra is a humanoid robot developed by an Indian startup company named 'Invento'. Version 2 of this robot was inaugurated by the Indian Prime Minister Narendra and Ivanka Trump in 2017. This robot has multiple abilities including facial and speech recognition. The facial recognition feature includes Age & Gender prediction as well.

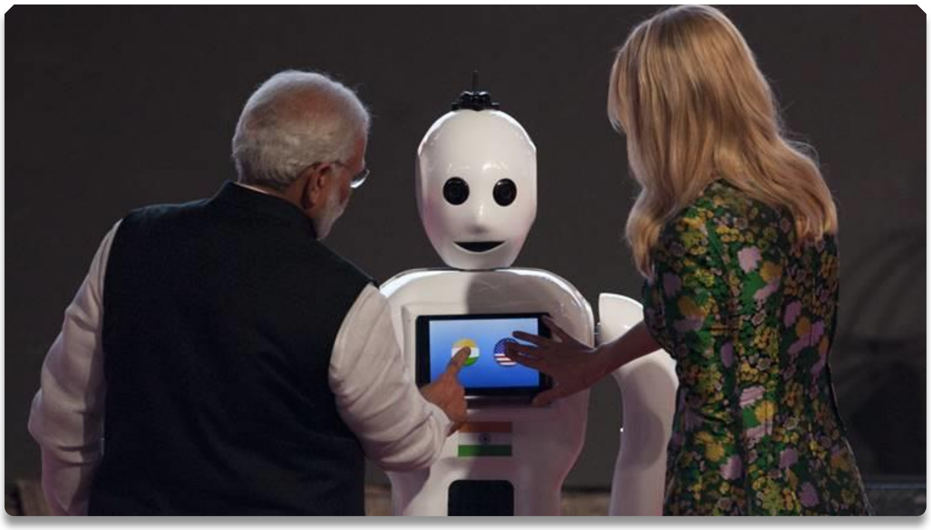


Fig: Inauguration of Mitra 2.0

On April 24, 2020, Mitra has been utilized by a hospital in Bengaluru, India for a different purpose. A purpose associated to intensify the screening of COVID-19 for the healthcare workers. In recent times, there have been instances where medical staff got effected by the COVID 19. Fortis hospital aims to address this concern by checking with all the medical staff daily on their symptoms and Mitra would help them here. Let us see in detail how this would be achieved.

Two Robots have been placed in Fortis Hospital. One at the entrance and other at a different spot. Every medical and non-medical staff entering the hospital daily would interact with Mitra first and would be checked for COVID 19 symptoms like fever etc. If they do not possess any, they would be let to enter the hospital. If they have any of the symptoms, they would be redirected to the second Robot where they can talk to the doctor through video call for the next steps.

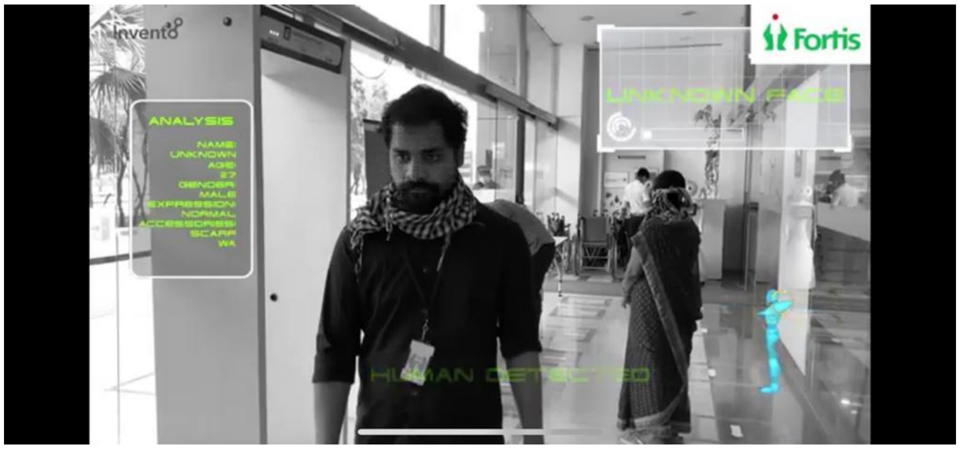


Fig: Robot Analysis Screen

As a person approaches Mitra, the system would identify its age, gender automatically. This saves any manual efforts associated with taking personal details. Also, having this data while scanning will help gather data to generate statistics periodically.

2.3 Age-Bot

Another real-time application of this Age & Gender prediction is in the world of mobile applications.

There is an application names 'Age-Bot' which is available to download in app stores. These apps are for fun sake where the end-user can upload any image of a person and the internal logic will predict the age and gender.

There are several applications available with such prediction ability. This helps the app developers get more revenue and the users get some fun time.

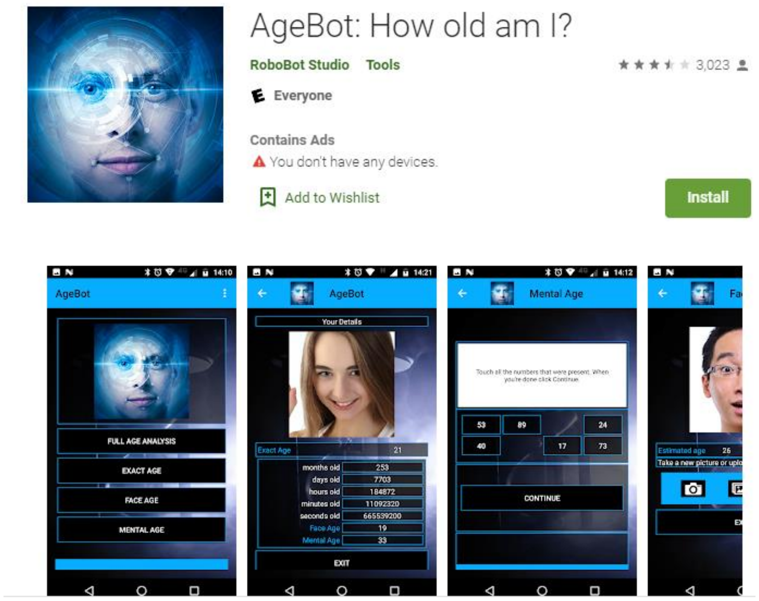
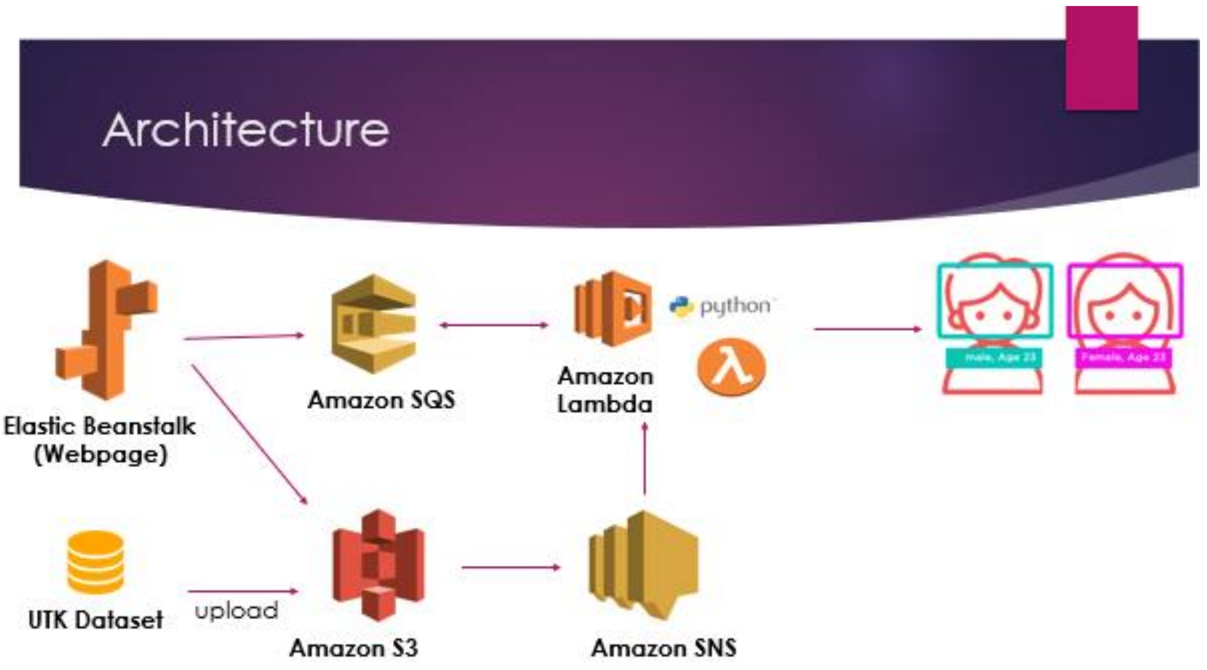


Fig: Age-bot app

All the above examples show us the potential of Age & Gender prediction in various industries.

3 AWS Architecture



Our architecture primarily consists of the usage of AWS resources.

We would like to host a website that will upload test images and feed to our Machine learning model and the result will be displayed back to the website in the AWS framework.

Firstly, we would need the server to host our website. We are planning to code the website either in PHP or python, these platforms are listed in the elastic beanstalk. AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications, services developed with Java, .NET, PHP, etc. Our website will be configured in a way that it interacts with the queuing system and file system. We are going to use Amazon SQS for queuing and amazon S3 for the file system. Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables us to decouple and scale microservices, distributed systems, and serverless applications. Amazon S3 or Amazon Simple Storage Service is a service offered by Amazon Web Services that provides object storage through a web service interface. The website will upload the image to the file system, and it will add related tasks in the queuing service. Once the file is uploaded to the S3 file system my Amazon SNS will catch that event and it will trigger AWS Lambda that the file has been added. Amazon Simple Notification Service (SNS) is a highly available, durable, secure, fully managed pub/sub messaging service that enables us to decouple microservices, distributed systems, and serverless applications wherein AWS Lambda lets us run code without provisioning or managing servers. AWS lambda will have machine learning model which will predict the gender and age. The machine learning algorithm will be trained on the UTK database which will be stored in the AWS S3 bucket.

Considering the cost factor, our project has not been implemented in AWS. But we certainly believe this architecture will make the project implementation feasible on a large scale. Cloud resources address traditional concerns like scalability, the cost factor.

3.1 Storage

Considering the architecture, the storage component is the AWS S3. This is a simple storage service offered by Amazon web services. Having the data in cloud makes it easy to connect from anywhere. The advantages of cloud computing like easy storage, scalability, cost effectiveness certainly make this solution more reliable.

Amazon S3

All the images which are either captured from the camera or the data set to train the model will be stored in S3. As the prediction is completed, the images can be sent back to S3 for storage as well.

4 Data Source

‘UTKFace’ dataset is a large-scale face dataset with a long age span (range from 0 to 116 years old). The dataset consists of over 20,000 face images with annotations of age, gender, and ethnicity. The images cover large variations in pose, facial expression, illumination, occlusion, resolution, etc. This dataset could be used on a variety of tasks, e.g., face detection, age estimation, age progression/regression, landmark localization, etc. Some sample images are shown as follows

4.1 Highlights of data source

- consists of 20k+ face images in the wild (only single face in one image)
- provides the correspondingly aligned and cropped faces
- provides the corresponding landmarks (68 points)
- images are labeled by age, gender, and ethnicity

4.2 Label

The labels of each face image are embedded in the file name, formatted like `[age]_[gender]_[race]_[date&time].jpg`.

- `[age]` is an integer from 0 to 116, indicating the age
- `[gender]` is either 0 (male) or 1 (female)
- `[race]` is an integer from 0 to 4, denoting White, Black, Asian, Indian, and Others (like Hispanic, Latino, Middle Eastern).
- `[date&time]` is in the format of ‘yyyymmddHHMMSSFFF’, showing the date and time an image was collected to ‘UTKFace’

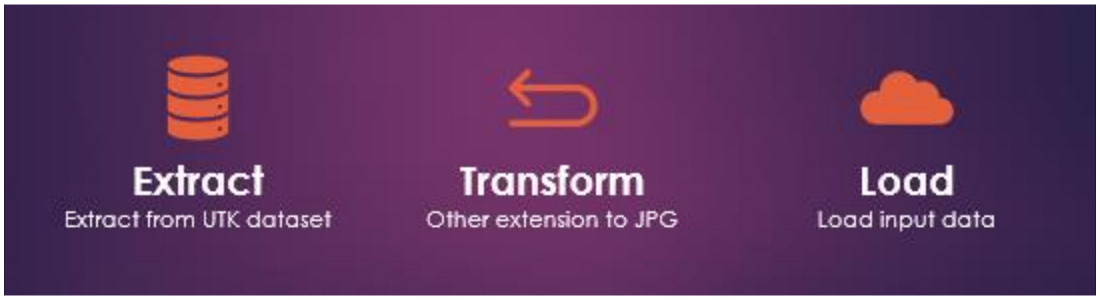
4.3 Sample dataset

Here are a few of the pictures from the data set taken.

5 ETL process

Our current dataset consists of all JPEG images. However, if we encounter any image which is not in JPEG format, we will transform it into a JPEG image. We will use Python script for replacing the extension of the image. Once all our images are in JPEG format, we then load this data into AWS S3 Bucket.

In current process we have manually modified the test files to extension to .JPG and further opened in our python program. We can automate this transformation as a future scope.

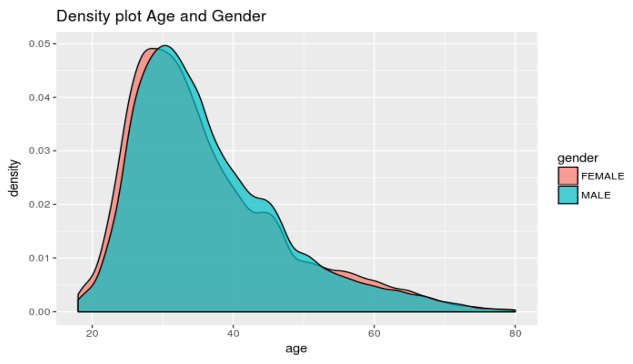


1. Extract from the UTK face dataset
2. Other extension JPG
3. Load the input data

The other format of images could be psd, tiff, png, etc. By manually changing the extension also, the format could be modified.

6 Data Visualization

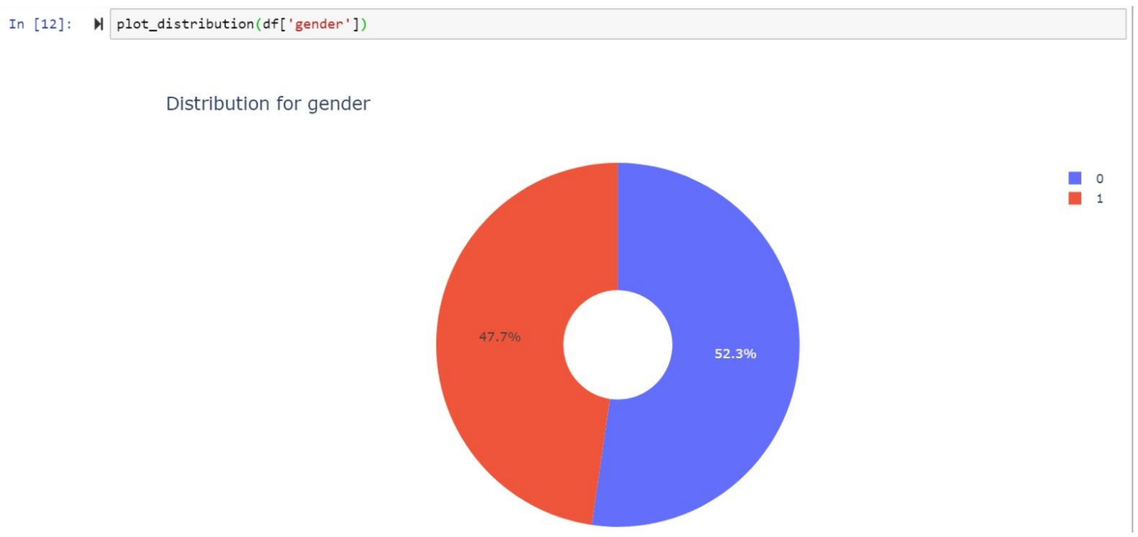
This will be done for exploratory analysis of the dataset. We mainly want to check two things, outliers, and the distribution, since having them in our model will result in low accuracy. Thus, with the help of these visualizations, we will check for the outliers and ensure that our data is normally distributed. The predicted output of the model will also not be biased, and we believe this is extremely important.



We plotted the density plot of the age for both females as well as males. From the graph shown in the diagram, we can tell that both male and female photographs are biased in the age range of 20-40 years.

6.1 Gender Distribution

Our data set contains images of Male and female gender. With a dataset of around 23,500 images – 11,210 are male and 12,290 are female in gender.



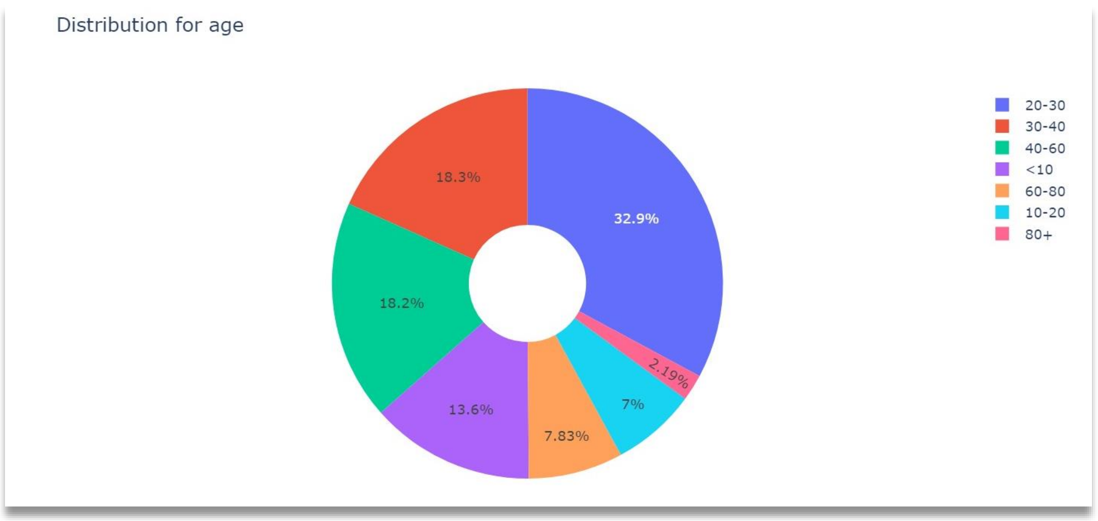
```
In [12]:  plot_distribution(df['gender'])
```

We have visualized the data for gender. From the figure shown here as we can see 47.7 % images are of male and 52.3% are female so we can say that the gender data is not biased.

6.2 Age Distribution

To understand the data in terms of age, we have visualized the data set as a pie chart. From the figure shown below, we see that most of the images are in the age group of 20-30.

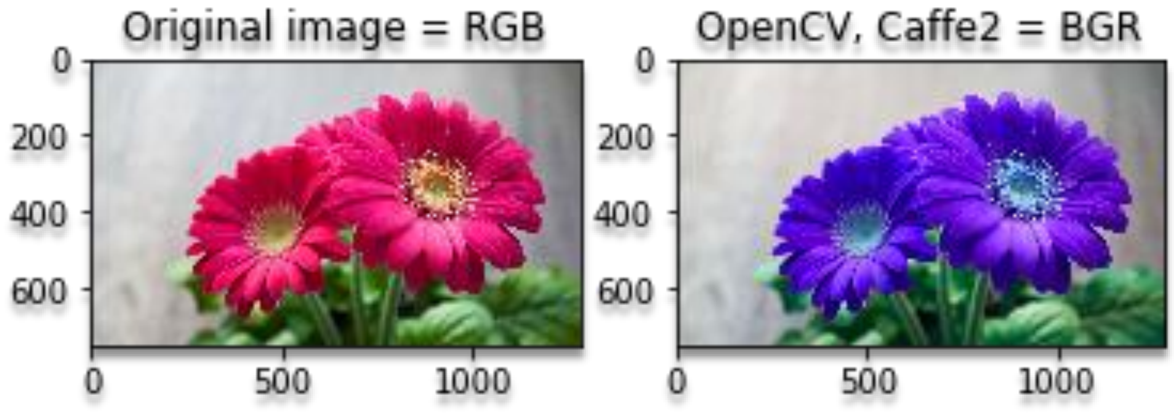
18.3 % of data belongs to the age group 30-40
18.2 % of data belongs to the age group 40-60
32.9 % of data belongs to the age group 20-30
2.19 % of data belongs to the age group 80+
7 % of data belongs to the age group 10-20
7.83 % of data belongs to the age group 60-80
13.6 % of data belongs to the age group <10



7 Image Transformation

7.1 BGR to RGB

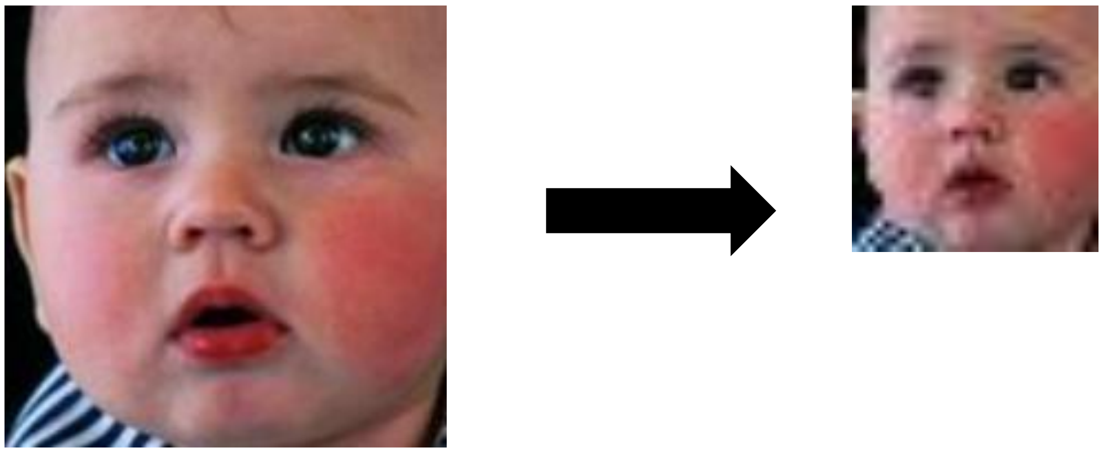
We are transforming the image from BGR to RBG. RGB stands for Red Green Blue. Most often, an RGB color is stored in a structure or unsigned integer with Blue occupying the least significant "area". (a byte in 32-bit and 24-bit formats), Green the second least, and Red the third least. BGR is the same, except the order of areas is reversed. Red occupies the least significant area, Green the second (still), and Blue the third.



When the image file is read with the OpenCV function imread()then the order of colors is BGR (blue, green, red). On the other hand, in Pillow, the order of colors is assumed to be RGB (red, green, blue). Therefore, if we want to use both the Pillow function and the OpenCV function, we need to convert BGR and RGB.

We are using OpenCV and pillow functions. OpenCV is a library of programming functions mainly aimed at real-time computer vision. Pillow function which adds support for opening, manipulating, and saving images.

7.2 Image Resize



We are trying to convert the images into vectors. So, all the images need to be resized into a defined size. Above we can see an input image which is resized to 32x32 image and then these 32x32 images are being used for further processing.

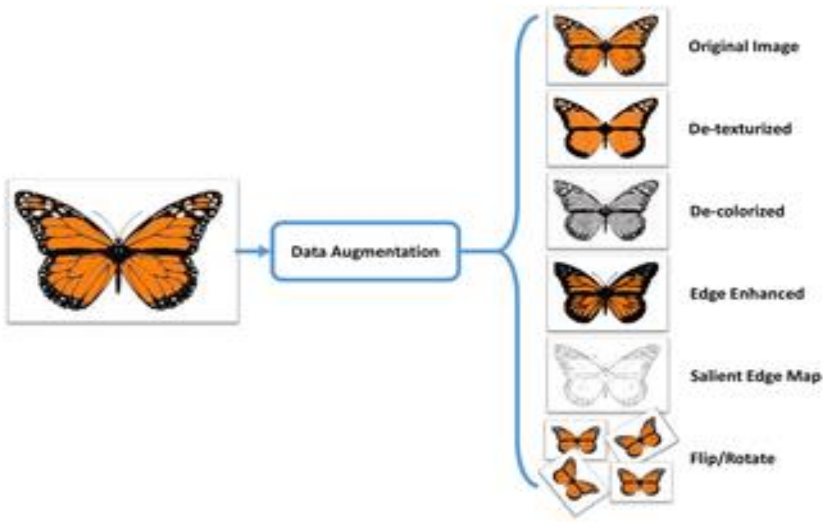
In a nutshell, whatever is the size of the input image – it is converted to 32x32 to feed into our model.

Image Transformation

In terms of image transformation, we have followed the above techniques as discussed namely BGR to RGB and Image re size.

8 Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.



This can be done by training the model for all possibilities. But we cannot go around clicking the same training picture in every possible angle when the training set is as big as 20000 pictures!

This can be easily be solved by a technique called Image Data Augmentation, which takes an image, converts it, and saves it in all the possible forms we specify.

8.1 Pros and Cons

Our results unveil a largely ignored advantage of data augmentation: networks trained with just data augmentation more easily adapt to different architectures and amount of training data, as opposed to weight decay and dropout, which require specific fine-tuning of their hyperparameters.

8.2 Benefits

The first is the ability to generate 'more data' from limited data.
The second one is to avoid overfitting: For a network, it is somewhat problematic to memorize a larger amount of data, as it is very important to avoid overfitting.

8.3 Effectiveness

The accuracy of deep learning models can be significantly improved through data augmentation, to generate additional training data, which helps prevent overfitting and improves the model's understanding of dataset features.

8.4 Image Augmentation with 'ImageDataGenerator'

The Keras deep learning library provides the ability to use data augmentation automatically when training a model. This is achieved by using the ImageDataGenerator class.

First, the class may be instantiated and the configuration for the types of data augmentation is specified by arguments to the class constructor.

A range of techniques are supported, as well as pixel scaling methods. We will focus on five main types of data augmentation techniques for image data; specifically:

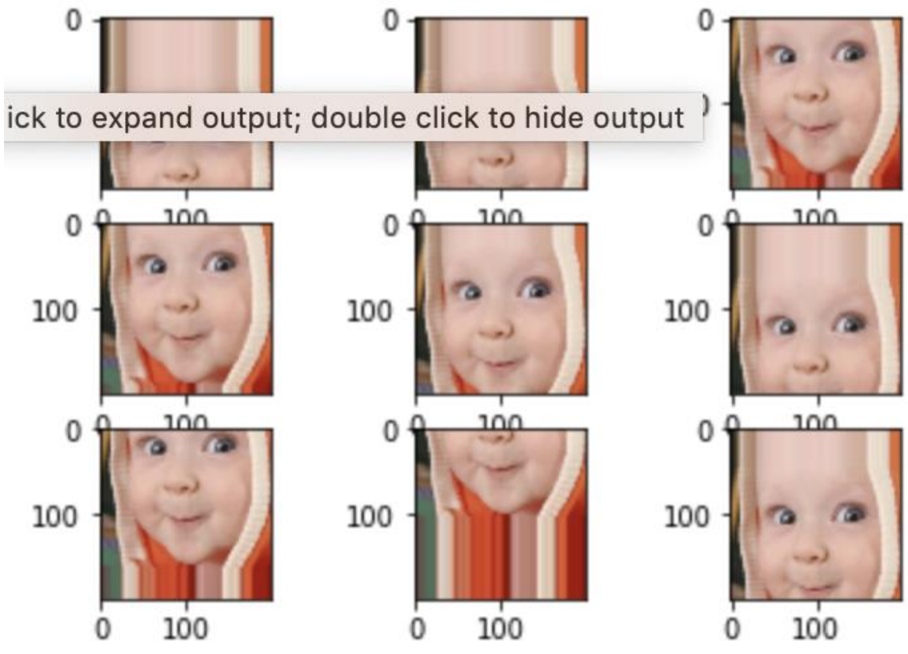
- Image rotations via the rotation_range argument.
- Image zoom via the zoom_range argument.
- Image flips via the horizontal_flip and vertical_flip arguments.
- Image shifts via the width_shift_range and height_shift_range arguments.
- Image brightness via the brightness_range argument.

8.4.1 Vertical shift image augmentation

Below is the same example updated to perform vertical shifts of the image via the height_shift_range argument, in this case specifying the percentage of the image to shift as 0.5 the height of the image. Running the example creates a plot of images augmented with random positive and negative vertical shifts.

We can see that both horizontal and vertical positive and negative shifts probably make sense for the chosen photograph, but in some cases, the replicated pixels at the edge of the image may not make sense to a model.

Note that other fill modes can be specified via “fill_mode” argument.


ick to expand output; double click to hide output

8.4.2 Horizontal flip image augmentation

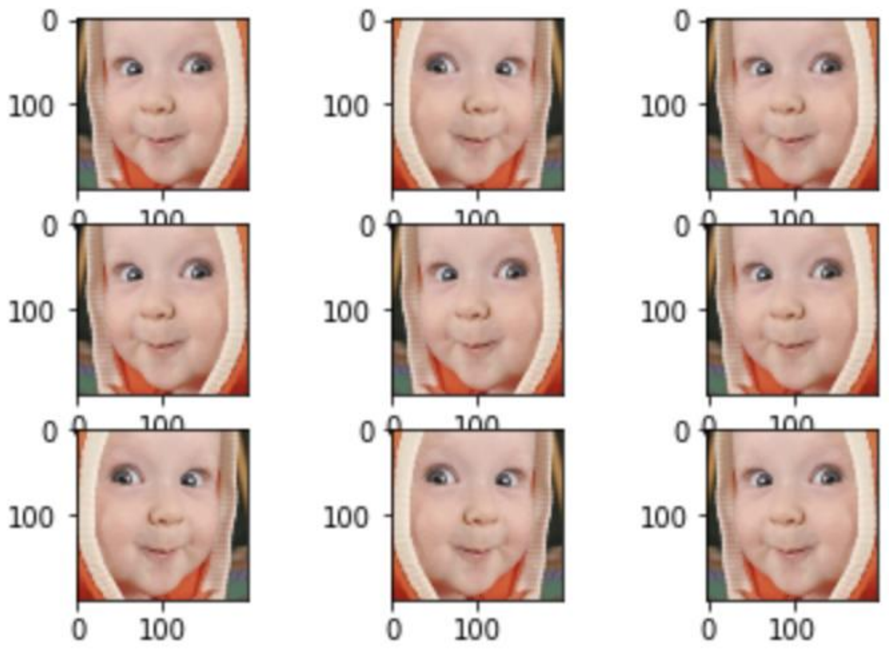
An image flip means reversing the rows or columns of pixels in the case of a vertical or horizontal flip respectively.

The flip augmentation is specified by a boolean horizontal_flip or vertical_flip argument to the ImageDataGenerator class constructor. For photographs like the bird photograph used in this tutorial, horizontal flips may make sense, but vertical flips would not.

For other types of images, such as aerial photographs, cosmology photographs, and microscopic photographs, perhaps vertical flips make sense.

The example below demonstrates augmenting the chosen photograph with horizontal flips via the horizontal_flip argument.

Running the example creates a plot of nine augmented images.
We can see that the horizontal flip is applied randomly to some images and not others.



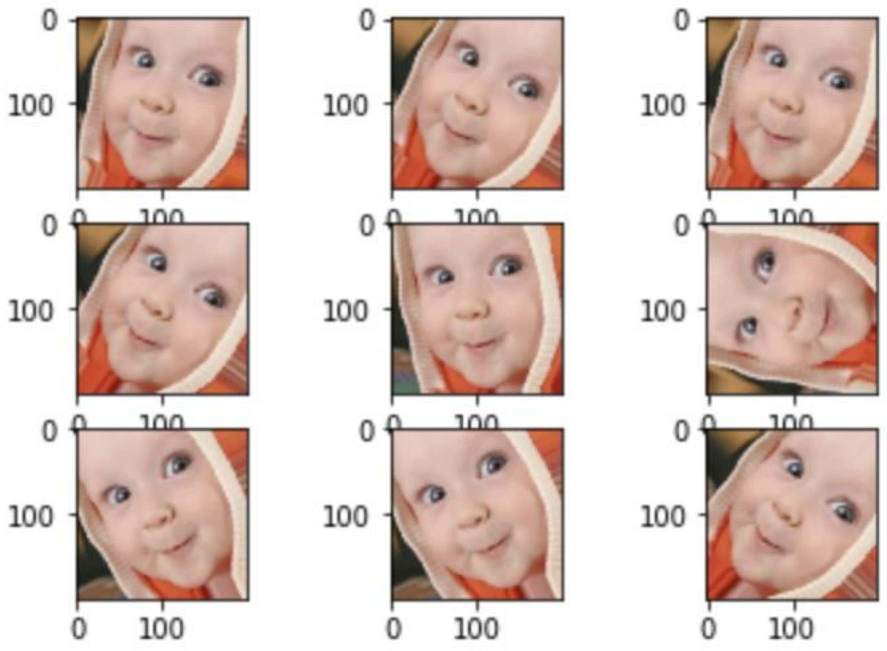
8.4.3 Random rotation image augmentation

A rotation augmentation randomly rotates the image clockwise by a given number of degrees from 0 to 360.

The rotation will likely rotate pixels out of the image frame and leave areas of the frame with no pixel data that must be filled in.

The example below demonstrates random rotations via the rotation_range argument, with rotations to the image between 0 and 90 degrees.

Running the example generates examples of the rotated image, showing in some cases pixels rotated out of the frame and the nearest-neighbor fill.



8.4.4 Brighting image augmentation

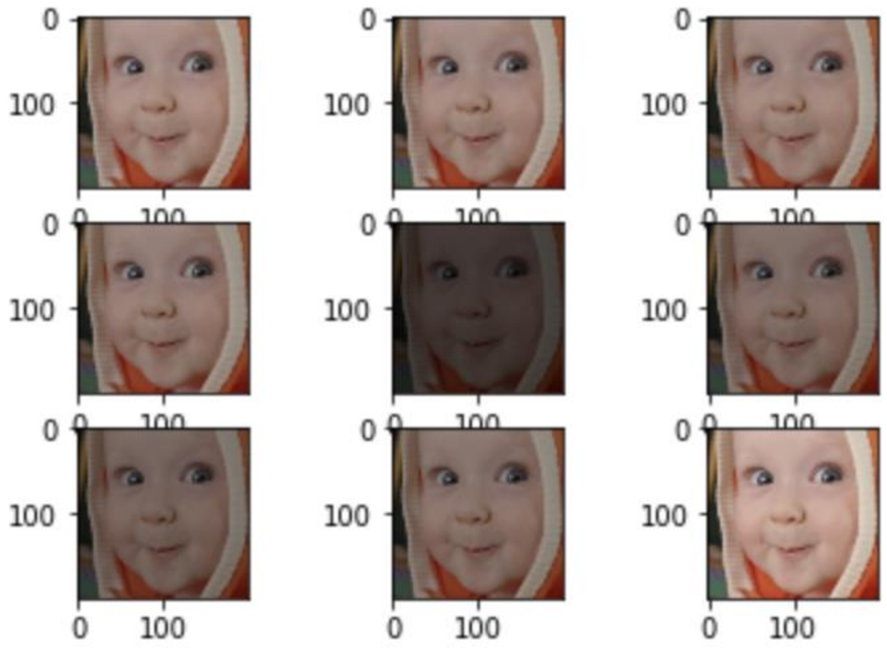
The brightness of the image can be augmented by either randomly darkening images, brightening images, or both.

The intent is to allow a model to generalize across images trained on different lighting levels.
This can be achieved by specifying the brightness_range argument to the ImageDataGenerator() constructor that specifies min and max range as a float representing a percentage for selecting a brightening amount.

Values less than 1.0 darken the image, e.g. [0.5, 1.0], whereas values larger than 1.0 brighten the image, e.g. [1.0, 1.5], where 1.0 has no effect on brightness.

The example below demonstrates a brightness image augmentation, allowing the generator to randomly darken the image between 1.0 (no change) and 0.2 or 20%.

Running the example shows the augmented images with varying amounts of darkening applied.



8.4.5 Zoom image augmentation

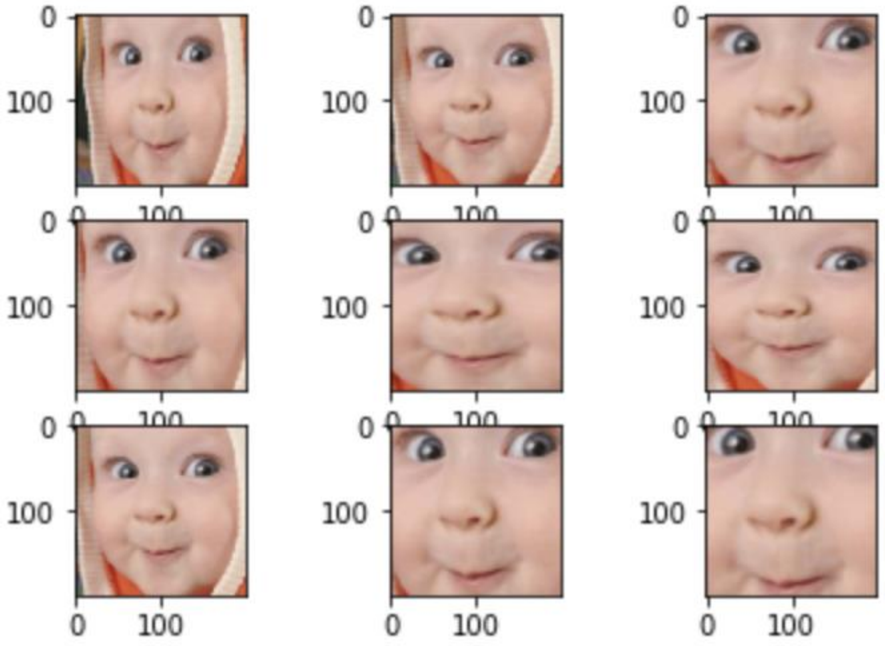
A zoom augmentation randomly zooms the image in and either add new pixel values around the image or interpolates pixel values respectively.

Image zooming can be configured by the zoom_range argument to the ImageDataGenerator constructor. You can specify the percentage of the zoom as a single float or a range as an array or tuple.
If a float is specified, then the range for the zoom will be [1-value, 1+value]. For example, if you specify 0.3, then the range will be [0.7, 1.3], or between 70% (zoom in) and 130% (zoom out).

The zoom amount is uniformly randomly sampled from the zoom region for each dimension (width, height) separately.

The zoom may not feel intuitive. Note that zoom values less than 1.0 will zoom the image in, e.g. [0.5,0.5] makes the object in the image 50% larger or closer, and values larger than 1.0 will zoom the image out by 50%, e.g. [1.5, 1.5] makes the object in the image smaller or further away. A zoom of [1.0,1.0] has no effect.
The example below demonstrates zooming the image in, e.g. making the object in the photograph larger.

Running the example generates examples of the zoomed image, showing a random zoom in that is different on both the width and height dimensions that also randomly changes the aspect ratio of the object in the image.



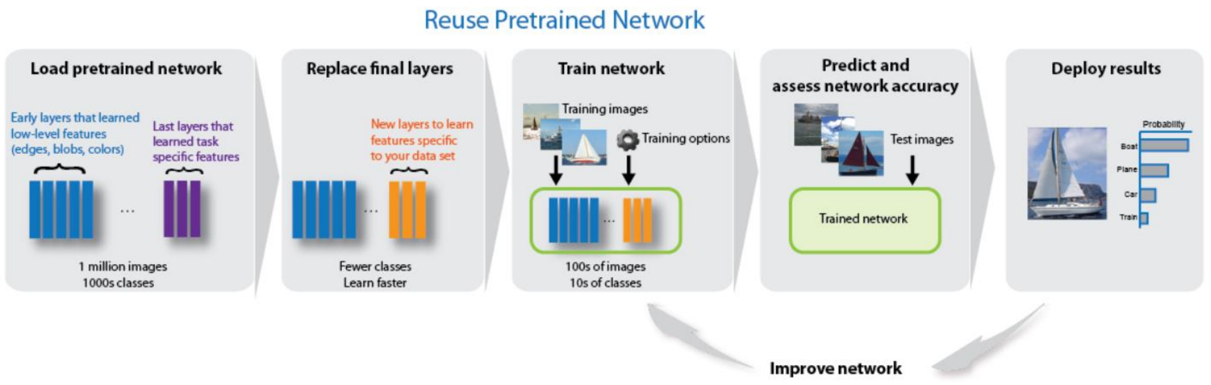
These are the augmentation techniques we have considered for our project.

9 Inception V3 Model

Inception V3 is a widely used image recognition model developed by Google and V3 signifies its third version. This model is trained on ImageNet Dataset. We aim to utilize this model to ensure better accuracy of the prediction.

It is a convolutional neural network that is 48 layers deep. It is present in Keras, which is a deep learning framework that runs on top of TensorFlow. TensorFlow is python library used to implement deep network. Tensors are the standard way of representing data in TensorFlow and are multidimensional arrays, an extension of 2D tables to data with higher dimension

As seen in Figure 5, the process of using a pre-trained network includes different steps like loading the pre-trained network, replacing the final layers, training the network followed by prediction, and deployment.



We use the Inception V3 model with **10 layers and 50 epochs**, it took **4 days** to completely run the model.

Best Model saved based on Validation Loss. Since the model was so time-consuming, we did not implement it in AWS. We got the age accuracy as 28% and Gender Accuracy as 52%. We can clearly say that Model was underperforming. Its most probably because even if Inception V3 was deeply trained on various images and just not human images. So, we decided to go for the CNN network to predict age and gender from the UTK database.

10 CNN Model

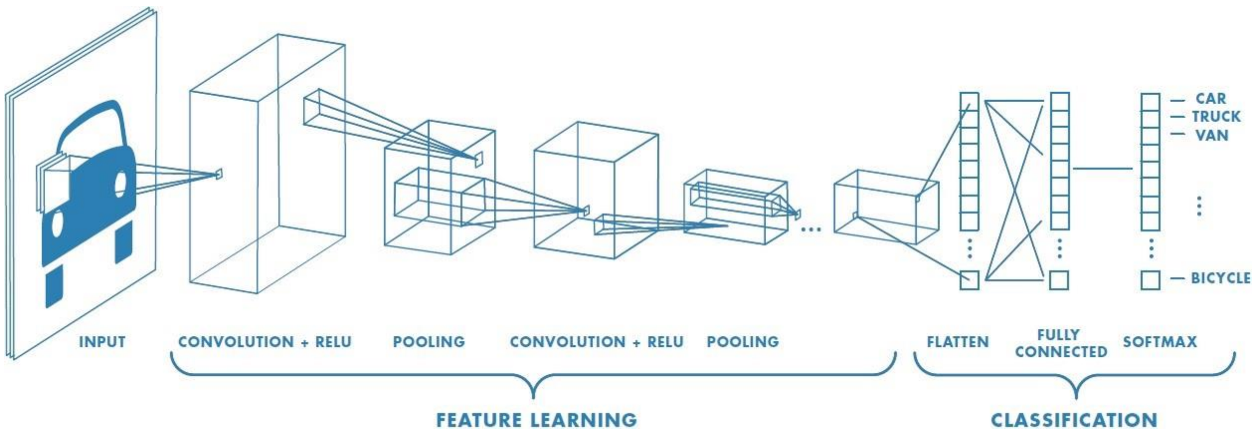
In neural networks, a Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections recognize faces, etc., are some of the areas where CNNs are widely used.

CNN image classifications take an input image, process it, and classify it under certain categories. Computers see an input image as an array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension).

Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC), and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.

Steps involved in CNN module

- Provide input image into convolution layer
- Choose parameters, apply filters with strides, padding if requires. Perform convolution on the image and apply ReLU activation to the matrix.
- Perform pooling to reduce dimensionality size
- Add as many convolutional layers until satisfied
- Flatten the output and feed into a fully connected layer (FC Layer)
- Output the class using an activation function (Logistic Regression with cost functions) and classifies images.



10.1 Convolution Layer

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

- An image matrix (volume) of dimension **($h \times w \times d$)**
- A filter **($f_h \times f_w \times d$)**
- Outputs a volume dimension **($h - f_h + 1$) x ($w - f_w + 1$) x 1**

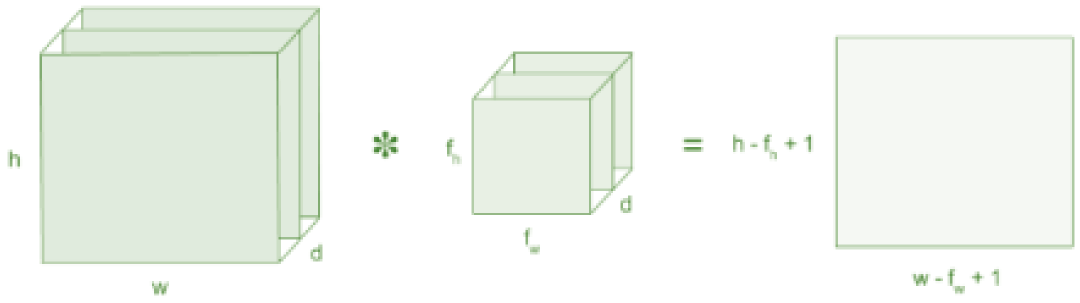


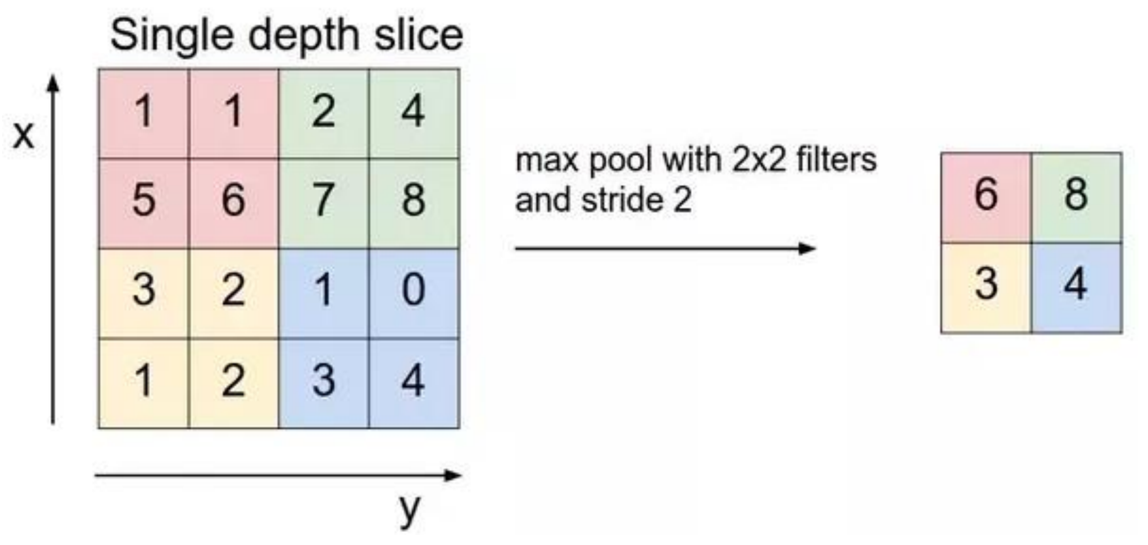
Image matrix multiplies kernel or filter matrix

10.2 Pooling Layer

The pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains important information. Spatial Pooling can be of different types:

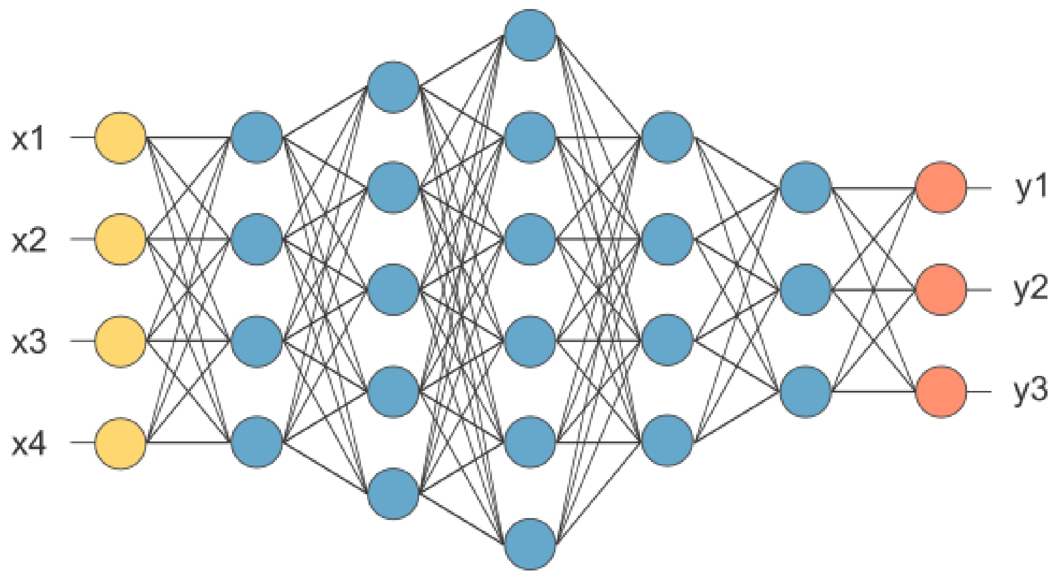
- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling takes the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.



10.3 Fully Connected Layer

The layer we call as the FC layer, we flattened our matrix into the vector and feed it into a fully connected layer like a neural network.



In the above diagram, the feature map matrix will be converted as a vector (x1, x2, x3, …). With the fully connected layers, we combined these features to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs as a cat, dog, car, truck, etc.,

11 AGE – GENDER detection model

In this post, we have explored the Keras functional API to build a multi-output Deep Learning model. We have shown how to train a single model that is capable of predicting three distinct outputs. By using the UTK Face dataset, which is composed of over 20 thousand pictures of people in uncontrolled environments, we have predicted the age, gender for each record presented in the dataset, reaching an accuracy of 89% for gender and 87% for age.

We also define a function to help us on extracting the data from our dataset. This function will be used to iterate over each file of the UTK dataset and return a Pandas Dataframe containing all the fields (age, gender) of our records. Function parse_dataset is used to extract information about our dataset. It does iterate over all images and return a DataFrame with the data (age, gender and race) of all files.

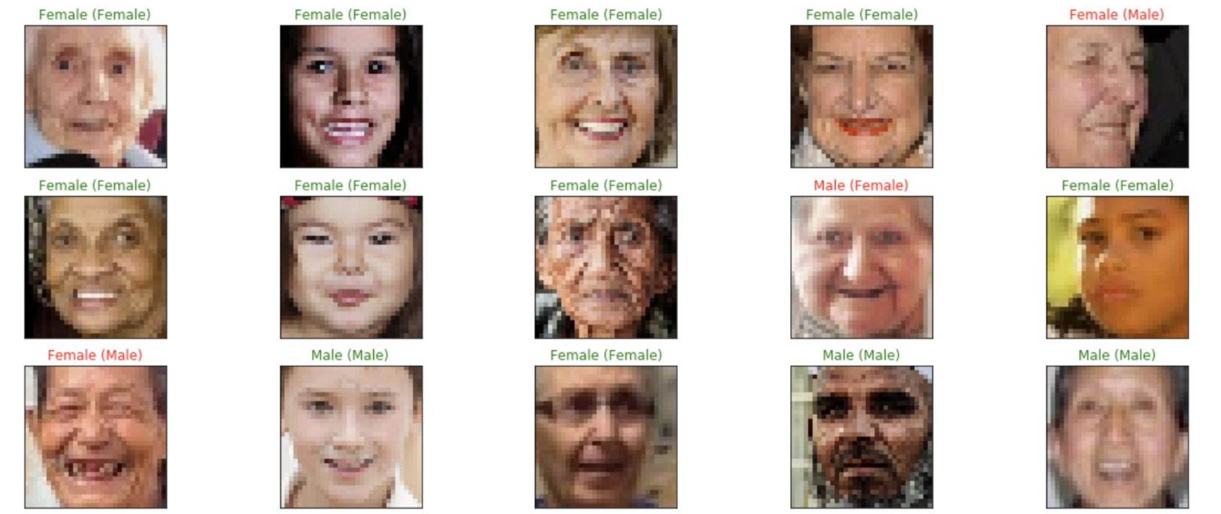
We further define the classes for better accuracy for results to be predicted.

1. Gender –
    a. Male (0 being male)
    b. Female (1 being female)
2. Age –
    a. Child (age less than 18 years)
    b. Adult (age within 18 to 60 years)
    c. Old (age greater than or equal to 60 years)

11.1 Gender Model

We have taken 2 convolutional neural network layers and 2 dense layers for the Gender model. The model used is sequential_4. Total params used are 534,114. trainable params are 534,114. Non-trainable params are 0. Train on 15008 samples, validate on 7000 samples. We have used 100 epochs with batch size 64. Additionally, we used check pointers to save the best models. Accuracy received for the model is around 89% with validation loss 28%.

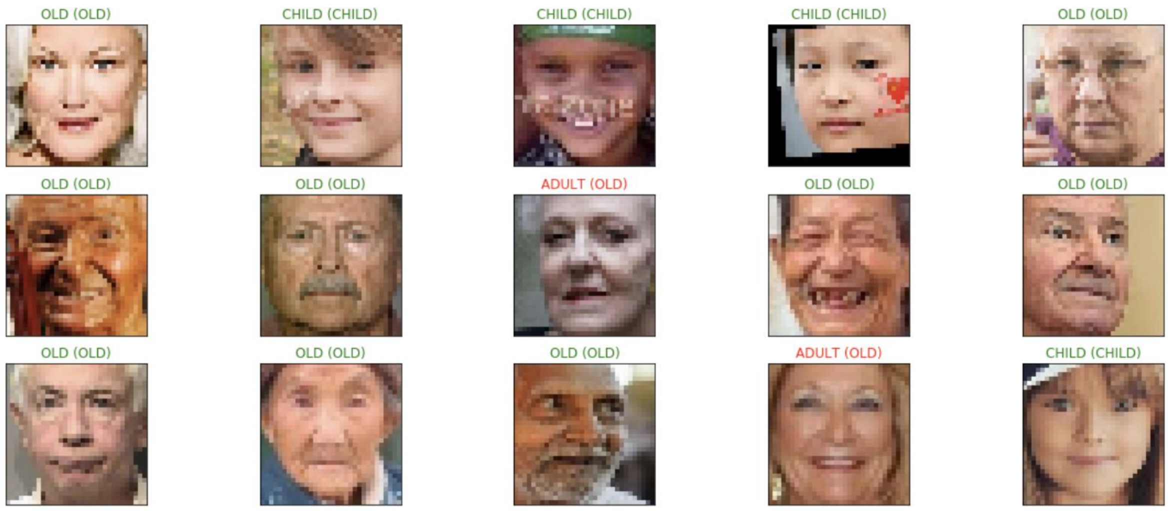
Refer below image which was run on test data.



11.2 Age Model

We have taken 2 convolutional neural network layers and 2 dense layers for the Age model. We defined the input shape in the first layer of the neural network. We used the "sequential_6" model. Total params: 534,371, Trainable params: 534,371, Non-trainable params: 0. Here we have used 35 epochs with 64 batch sizes. The model generated gives 87% accuracy with 54% validation loss.

Refer below image which was run on test data.



12 Our Code Flow

We have imported below libraries for our code.

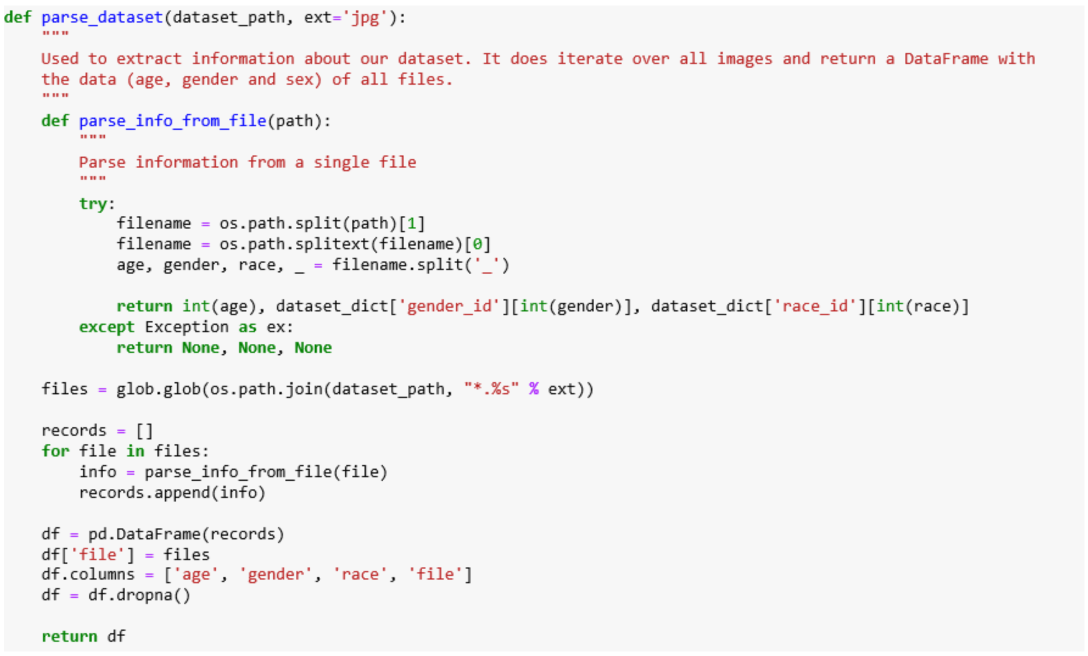
```
import numpy as np
import pandas as pd
import os
import glob
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
from keras.utils import to_categorical
from PIL import Image

from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras import optimizers
from keras.models import Sequential, Model
from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.callbacks import ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from keras.utils import np_utils
from keras.optimizers import SGD
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.layers.normalization import BatchNormalization
from keras.layers.core import Lambda
from keras.layers import Input
import tensorflow as tf
```

```
In [18]: currentDirectory = os.getcwd()

In [32]: dataset_name = os.path.join(currentDirectory, 'UTKFace/')
         onlyfiles = os.listdir(dataset_name)

In [33]: len(onlyfiles)

Out[33]: 23708
```

We have created a data set dictionary for Gender and Race. This will help us further to plot the visualizations.

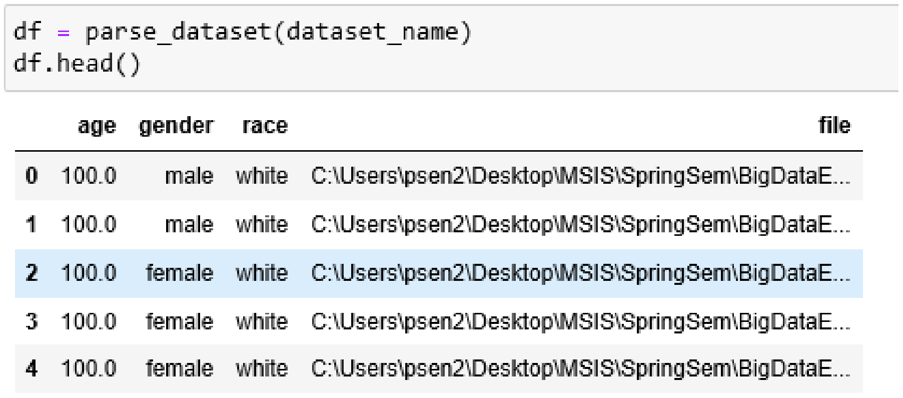
```
dataset_dict = {
    'race_id': {
        0: 'white',
        1: 'black',
        2: 'asian',
        3: 'indian',
        4: 'others'
    },
    'gender_id': {
        0: 'male',
        1: 'female'
    }
}

dataset_dict['gender_alias'] = dict((g, i) for i, g in dataset_dict['gender_id'].items())
dataset_dict['race_alias'] = dict((g, i) for i, g in dataset_dict['race_id'].items())
```

Each image in the dataset in the format [age]_[gender]_[race]_[date&time].jpg. It is being extracted and passed into a data frame and the labels are separated into columns as it is easy to read the data from it. We can easily extract gender, age, race from the columns. For this matter, we have created the below function namely parse_dataset.



```python
def parse_dataset(dataset_path, ext='jpg'):
    """
    Used to extract information about our dataset. It does iterate over all images and return a DataFrame with
    the data (age, gender and sex) of all files.
    """
    def parse_info_from_file(path):
        """
        Parse information from a single file
        """
        try:
            filename = os.path.split(path)[1]
            filename = os.path.splitext(filename)[0]
            age, gender, race, _ = filename.split('_')

            return int(age), dataset_dict['gender_id'][int(gender)], dataset_dict['race_id'][int(race)]
        except Exception as ex:
            return None, None, None

    files = glob.glob(os.path.join(dataset_path, "*.%s" % ext))

    records = []
    for file in files:
        info = parse_info_from_file(file)
        records.append(info)

    df = pd.DataFrame(records)
    df['file'] = files
    df.columns = ['age', 'gender', 'race', 'file']
    df = df.dropna()

    return df
```

This is how the data frame displays the results based on the above function created.



```python
df = parse_dataset(dataset_name)
df.head()
```

```
     age  gender   race  file
0  100.0    male  white  C:\Users\psen2\Desktop\MSIS\SpringSem\BigDataE...
1  100.0    male  white  C:\Users\psen2\Desktop\MSIS\SpringSem\BigDataE...
2  100.0  female  white  C:\Users\psen2\Desktop\MSIS\SpringSem\BigDataE...
3  100.0  female  white  C:\Users\psen2\Desktop\MSIS\SpringSem\BigDataE...
4  100.0  female  white  C:\Users\psen2\Desktop\MSIS\SpringSem\BigDataE...
```

Data Visualization

For the sake of data visualization, an additional function has been created namely plot_distribution' and it has been used to plot for gender and Age pie charts.

```
import plotly.graph_objects as go
from IPython.display import display, Image

def plot_distribution(pd_series):
    labels = pd_series.value_counts().index.tolist()
    counts = pd_series.value_counts().values.tolist()

    pie_plot = go.Pie(labels=labels, values=counts, hole=.3)
    fig = go.Figure(data=[pie_plot])
    fig.update_layout(title_text='Distribution for %s' % pd_series.name)

    img_bytes = fig.to_image(format="jpg")
    display(Image(img_bytes))
```

Data Preprocessing

We convert colored images from BGR format to RGB format since we want to use both the Pillow function and the OpenCV function.

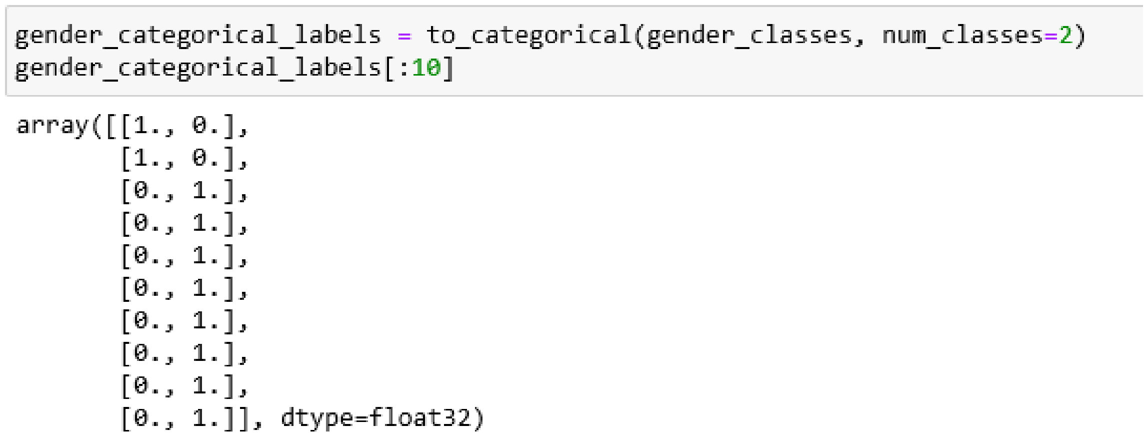
```
x_data =[]
for file in onlyfiles:
    face = cv2.imread(dataset_name+file)
    face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
    face =cv2.resize(face, (32,32) )
    x_data.append(face)
x_data=np.array(x_data)
x_data.shape

x = np.squeeze(x_data)

# normalize data
x = x.astype('float32')
x /= 255
x.shape
```

```
(23708, 32, 32, 3)
```

We start by creating two classes for gender - Male and Female.

```
gender = [i.split('_')[1] for i in onlyfiles]

gender_classes = []
for i in gender:
    i = int(i)
    gender_classes.append(i)
```

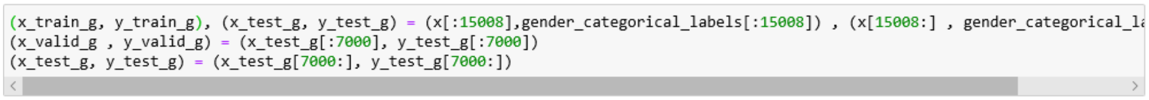
Convert the gender_classes into categorical data.



```
gender_categorical_labels = to_categorical(gender_classes, num_classes=2)
gender_categorical_labels[:10]
```

```
array([[1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

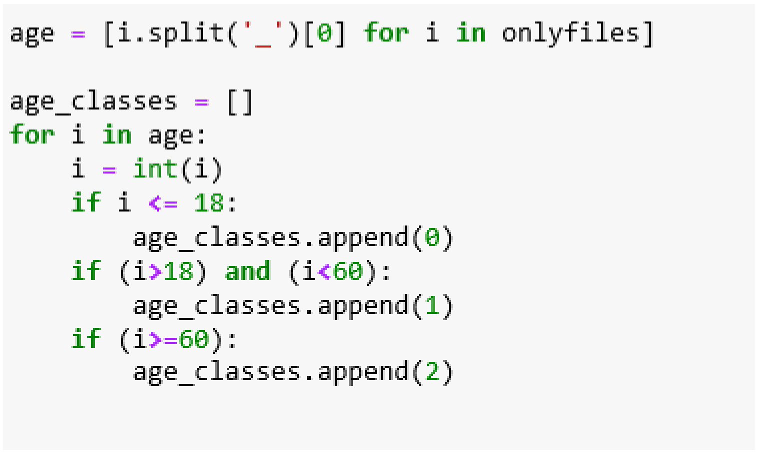
Divide Gender Data into Train, Test and Validate.



```
(x_train_g, y_train_g), (x_test_g, y_test_g) = (x[:15008],gender_categorical_labels[:15008]) , (x[15008:] , gender_categorical_la
(x_valid_g , y_valid_g) = (x_test_g[:7000], y_test_g[:7000])
(x_test_g, y_test_g) = (x_test_g[7000:], y_test_g[7000:])
```

Age Modelling

We start by creating three age labels. First is Child for individuals aged 18 or below, Adult is for individuals with age 19 to 59 and Old for ages 60 and above.



```
age = [i.split('_')[0] for i in onlyfiles]

age_classes = []
for i in age:
    i = int(i)
    if i <= 18:
        age_classes.append(0)
    if (i>18) and (i<60):
        age_classes.append(1)
    if (i>=60):
        age_classes.append(2)
```

Convert the age_classes into categorical data.

```
age_categorical_labels = to_categorical(age_classes, num_classes=3)
age_categorical_labels[:10]
```

```
array([[0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]], dtype=float32)
```

Divide Age Data into Train, Test and Validate.

```
(x_train_a, y_train_a), (x_test_a, y_test_a) = (x[:15008],age_categorical_labels[:15008]) , (x[15008:] , age_categorical_labels[
(x_valid_a , y_valid_a) = (x_test_a[:7000], y_test_a[:7000])
(x_test_a, y_test_a) = (x_test_a[7000:], y_test_a[7000:])
```

```
len(x_train_a)+len(x_test_a) + len(x_valid_a) == len(x)
```

```
True
```

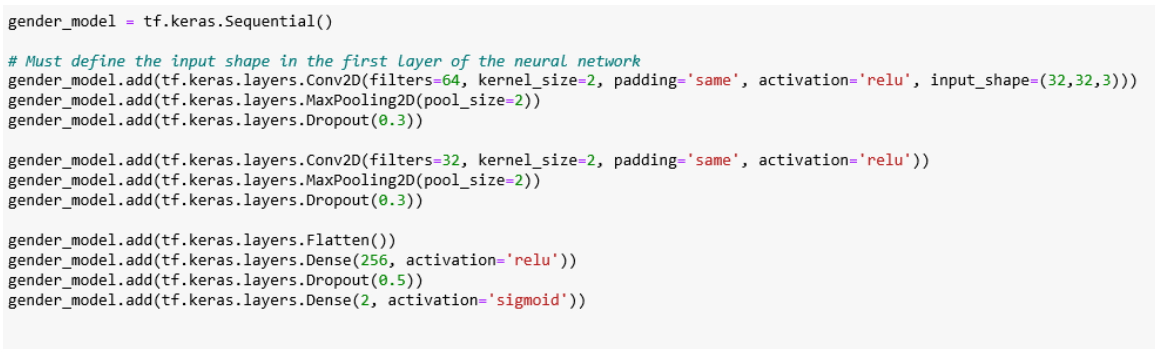
Data Augmentation

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without collecting new data.

```
# Train - Data Preparation - Data Augmentation with generators
train_datagen =  ImageDataGenerator(
  preprocessing_function=preprocess_input,
  rotation_range=30,
  width_shift_range=0.2,
  height_shift_range=0.2,
  shear_range=0.2,
  zoom_range=0.2,
  horizontal_flip=True,
)


# Gender
train_datagen.fit(x_train_g)

train_generator_gender = train_datagen.flow(
x_train_g, y_train_g,
batch_size=64,
)


# Age
train_datagen.fit(x_train_a)

train_generator_age = train_datagen.flow(
x_train_a, y_train_a,
batch_size=64,
)
```

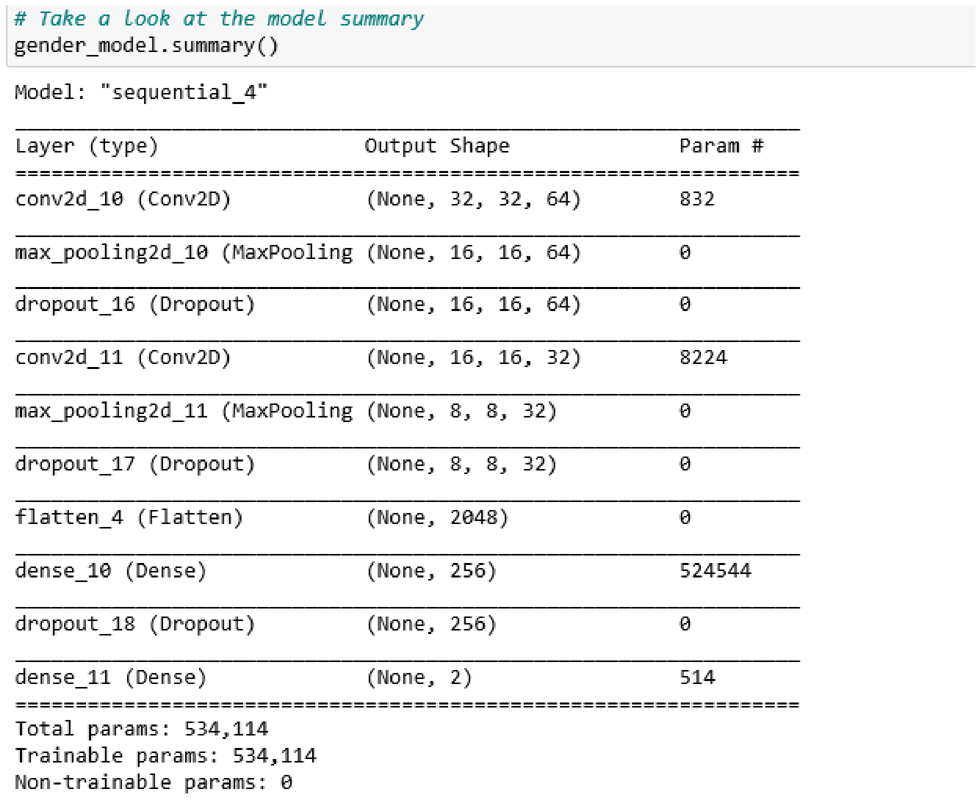
Gender Model

We have taken 3 convolutional neural network layers and 3 dense layers for the Gender model.



```
gender_model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
gender_model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(32,32,3)))
gender_model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
gender_model.add(tf.keras.layers.Dropout(0.3))

gender_model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
gender_model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
gender_model.add(tf.keras.layers.Dropout(0.3))

gender_model.add(tf.keras.layers.Flatten())
gender_model.add(tf.keras.layers.Dense(256, activation='relu'))
gender_model.add(tf.keras.layers.Dropout(0.5))
gender_model.add(tf.keras.layers.Dense(2, activation='sigmoid'))
```

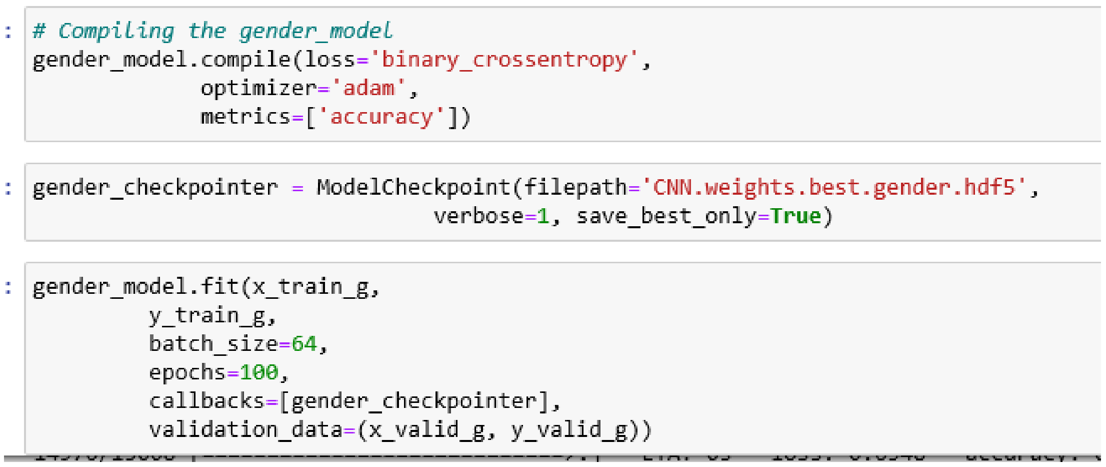
This is the summary.



```
# Take a look at the model summary
gender_model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_10 (Conv2D)           (None, 32, 32, 64)        832       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 16, 16, 64)        0         
_________________________________________________________________
dropout_16 (Dropout)         (None, 16, 16, 64)        0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 16, 16, 32)        8224      
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 8, 8, 32)          0         
_________________________________________________________________
dropout_17 (Dropout)         (None, 8, 8, 32)          0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 2048)              0         
_________________________________________________________________
dense_10 (Dense)             (None, 256)               524544    
_________________________________________________________________
dropout_18 (Dropout)         (None, 256)               0         
_________________________________________________________________
dense_11 (Dense)             (None, 2)                 514       
=================================================================
Total params: 534,114
Trainable params: 534,114
Non-trainable params: 0
```

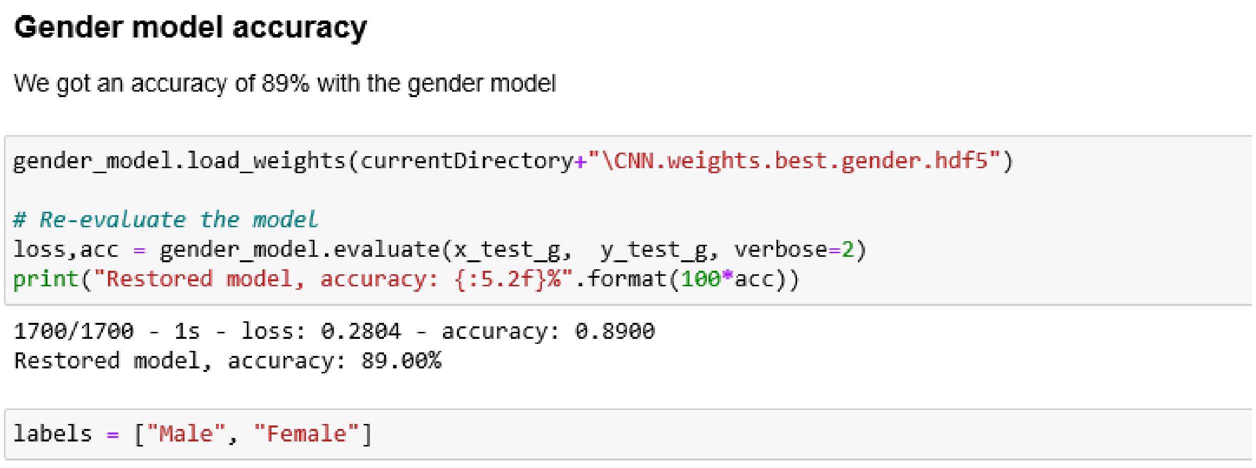
Compile, create check point and fit the model



```
# Compiling the gender_model
gender_model.compile(loss='binary_crossentropy',
             optimizer='adam',
             metrics=['accuracy'])
```

```
gender_checkpointer = ModelCheckpoint(filepath='CNN.weights.best.gender.hdf5',
                                      verbose=1, save_best_only=True)
```

```
gender_model.fit(x_train_g,
         y_train_g,
         batch_size=64,
         epochs=100,
         callbacks=[gender_checkpointer],
         validation_data=(x_valid_g, y_valid_g))
```



Gender model accuracy

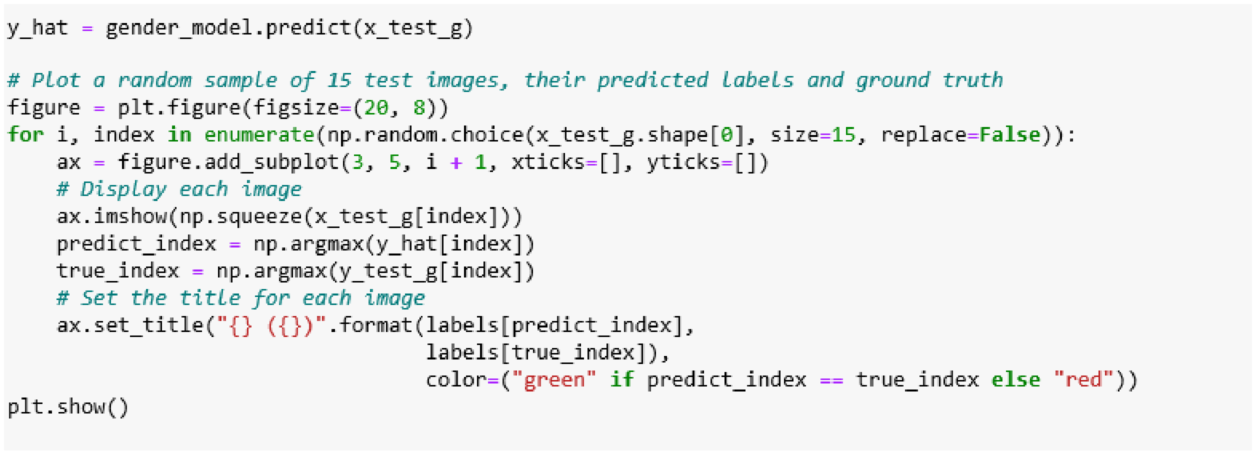
We got an accuracy of 89% with the gender model

```
gender_model.load_weights(currentDirectory+"\CNN.weights.best.gender.hdf5")

# Re-evaluate the model
loss,acc = gender_model.evaluate(x_test_g,  y_test_g, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

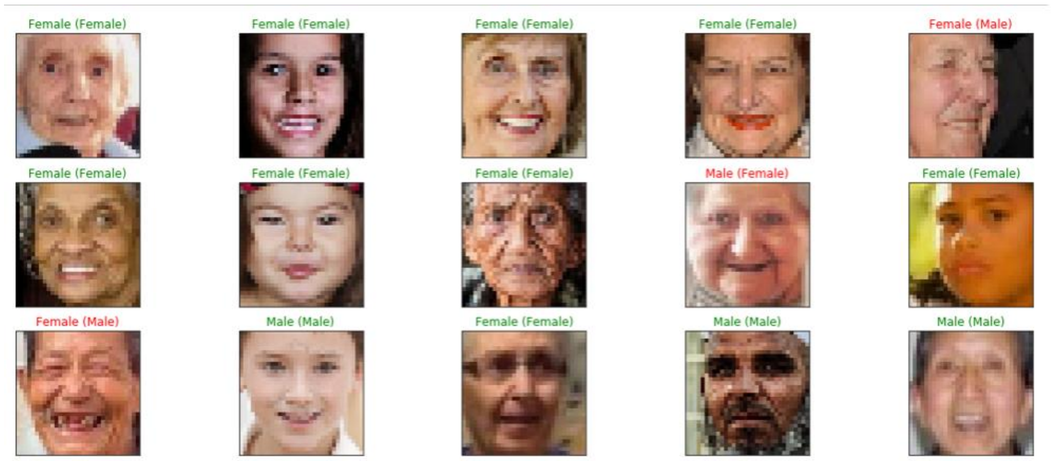
```
1700/1700 - 1s - loss: 0.2804 - accuracy: 0.8900
Restored model, accuracy: 89.00%
```

```
labels = ["Male", "Female"]
```

We plot a random sample of 15 test images, their predicted labels and ground truth for Gender.

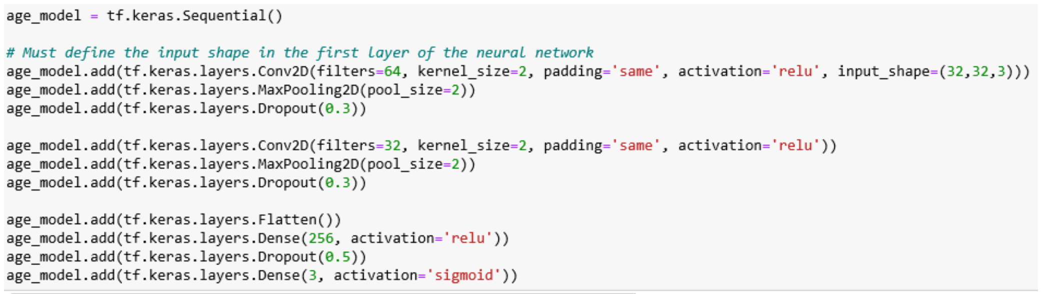


```
y_hat = gender_model.predict(x_test_g)

# Plot a random sample of 15 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test_g.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(x_test_g[index]))
    predict_index = np.argmax(y_hat[index])
    true_index = np.argmax(y_test_g[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels[predict_index],
                                  labels[true_index]),
                                  color=("green" if predict_index == true_index else "red"))
plt.show()
```

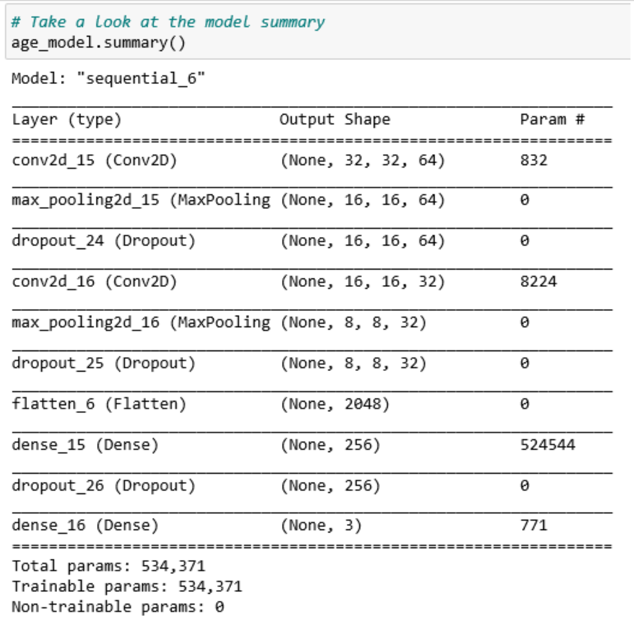


Age Model

We have taken 3 convolutional neural network layers and 3 dense layers for the Age model

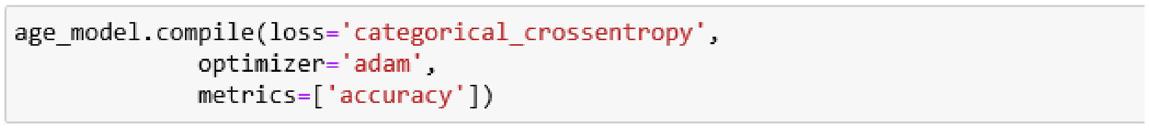


```
age_model = tf.keras.Sequential()

# Must define the input shape in the first layer of the neural network
age_model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=2, padding='same', activation='relu', input_shape=(32,32,3)))
age_model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
age_model.add(tf.keras.layers.Dropout(0.3))

age_model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
age_model.add(tf.keras.layers.MaxPooling2D(pool_size=2))
age_model.add(tf.keras.layers.Dropout(0.3))

age_model.add(tf.keras.layers.Flatten())
age_model.add(tf.keras.layers.Dense(256, activation='relu'))
age_model.add(tf.keras.layers.Dropout(0.5))
age_model.add(tf.keras.layers.Dense(3, activation='sigmoid'))
```

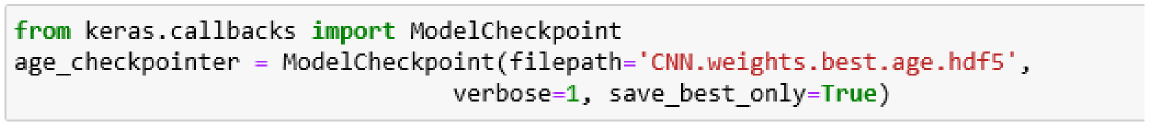


```
# Take a look at the model summary
age_model.summary()
```

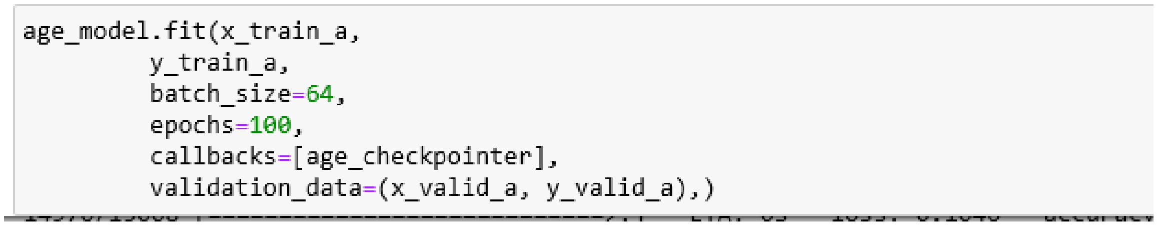
```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 32, 32, 64)        832
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 16, 16, 64)        0
_________________________________________________________________
dropout_24 (Dropout)         (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 16, 16, 32)        8224
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 8, 8, 32)          0
_________________________________________________________________
dropout_25 (Dropout)         (None, 8, 8, 32)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_15 (Dense)             (None, 256)               524544
_________________________________________________________________
dropout_26 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_16 (Dense)             (None, 3)                 771
=================================================================
Total params: 534,371
Trainable params: 534,371
Non-trainable params: 0
```



```
age_model.compile(loss='categorical_crossentropy',
             optimizer='adam',
             metrics=['accuracy'])
```



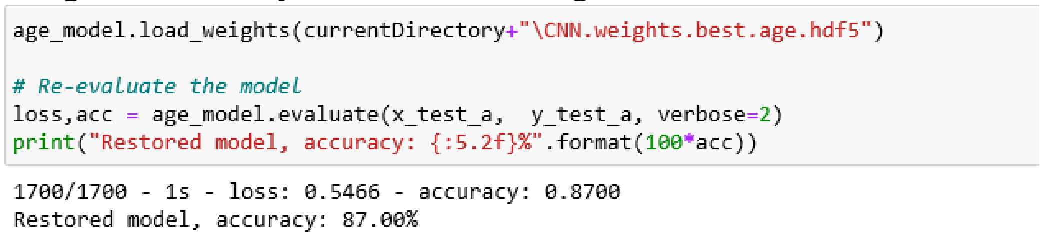
```
from keras.callbacks import ModelCheckpoint
age_checkpointer = ModelCheckpoint(filepath='CNN.weights.best.age.hdf5',
                                   verbose=1, save_best_only=True)
```



```
age_model.fit(x_train_a,
         y_train_a,
         batch_size=64,
         epochs=100,
         callbacks=[age_checkpointer],
         validation_data=(x_valid_a, y_valid_a),)
```

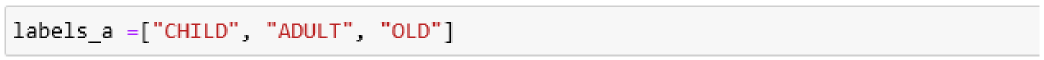
Age model accuracy

We got an accuracy of 87% with the age model



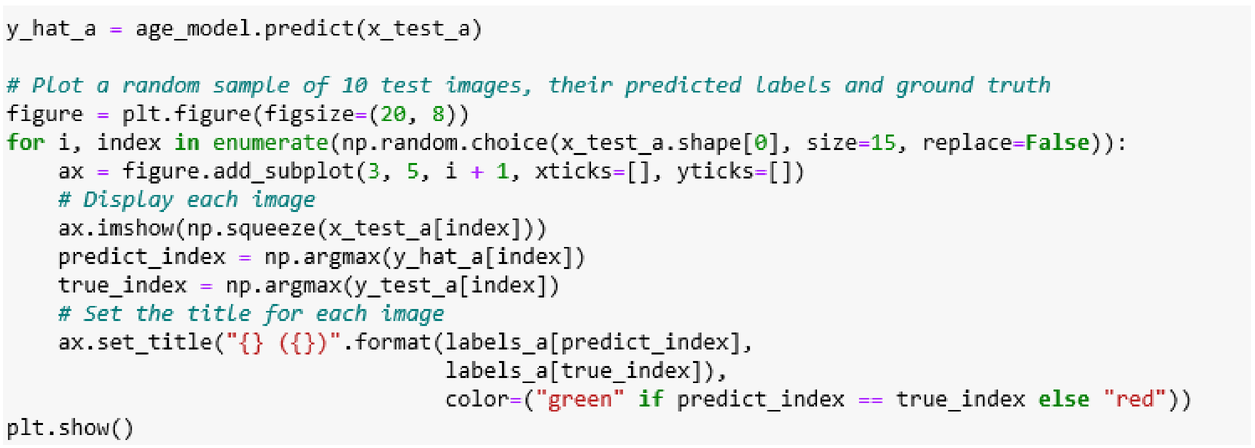
```
age_model.load_weights(currentDirectory+"\CNN.weights.best.age.hdf5")

# Re-evaluate the model
loss,acc = age_model.evaluate(x_test_a,  y_test_a, verbose=2)
print("Restored model, accuracy: {:5.2f}%".format(100*acc))
```

```
1700/1700 - 1s - loss: 0.5466 - accuracy: 0.8700
Restored model, accuracy: 87.00%
```



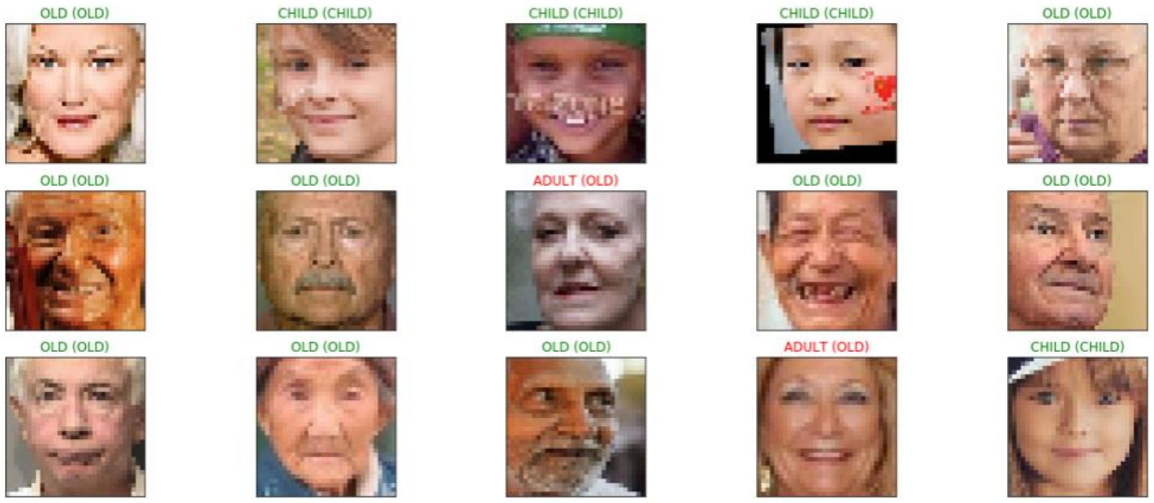
```
labels_a =["CHILD", "ADULT", "OLD"]
```

We plot a random sample of 15 test images, their predicted labels and ground truth for gender.



```
y_hat_a = age_model.predict(x_test_a)

# Plot a random sample of 10 test images, their predicted labels and ground truth
figure = plt.figure(figsize=(20, 8))
for i, index in enumerate(np.random.choice(x_test_a.shape[0], size=15, replace=False)):
    ax = figure.add_subplot(3, 5, i + 1, xticks=[], yticks=[])
    # Display each image
    ax.imshow(np.squeeze(x_test_a[index]))
    predict_index = np.argmax(y_hat_a[index])
    true_index = np.argmax(y_test_a[index])
    # Set the title for each image
    ax.set_title("{} ({})".format(labels_a[predict_index],
                                  labels_a[true_index]),
                                  color=("green" if predict_index == true_index else "red"))
plt.show()
```

Anything marked in red indicates that the prediction is not correct.



13 References

- ▶ Nolen,Sam(2019, Jul 02). GANs for Data Augmentation. Retrieved from https://medium.com/reality-engines/gans-for-data-augmentation-21a69de6c60b
- ▶ Ahmad,Jamil.,Muhammad,Khan.,Baik,Sung Wook.(2017, Aug 31).Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. Retrieved from https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0183838
- ▶ Tawhid,Nurul Ahad.,Dey,Emon Kumar.(2018, Feb). A Gender Recognition System from Facial Image. Retrieved from https://www.researchgate.net/publication/323346551_A_Gender_Recognition_System_from_Facial_Image
- ▶ Chernov,Pavel(2019, Apr 29). Age and gender estimation. Open-source projects overview. Simple project from scratch. Retrieved from https://medium.com/@pavelchernov/age-and-gender-estimation-open-source-projects-overview-simple-project-from-scratch-69581831297e
- ▶ Dr. Dataman(2019, Jul 18). What Is Image Recognition?. Retrieved from https://towardsdatascience.com/module-6-image-recognition-for-insurance-claim-handling-part-i-a338d16c9de0
- ▶ Kuligowski, Kiely(2019, Jul 18). Facial Recognition Advertising: The New Way to Target Ads at Consumers. Retrieved from https://www.businessnewsdaily.com/15213-walgreens-facial-recognition.html
- ▶ Chauhan, Nagesh Singh(2018, Nov 20). Predict Age and Gender using Convolutional Neural Network and openCV. Retrieved from https://towardsdatascience.com/predict-age-and-gender-using-convolutional-neural-network-and-opencv-fd90390e3ce6