A REPORT

ON

FACIAL DEMOGRAPHICS

By -

Shreyansh Joshi

2nd year CSE Student,

BITS Pilani

AT

Silver Touch Technologies, Ahmedabad

Acknowledgement

Firstly, I would like to thank Silver Touch Technologies Ltd. for granting me this wonderful opportunity to work as an intern at Silver Touch for a period of 6 weeks. These 6 weeks have been a great learning curve for me and have been nothing short of remarkable. I feel I have got wonderful exposure to the industry world.

I would like to thank Mr. Adarsh Parikh, Associate Vice President and Training and Development Head at Silvertouch Technologies, for providing me with an excellent opportunity to intern with them and gain exposure to industry-level work and professional ethics.

Next, I would like to express my deepest sense of gratitude to Ms Tanisha Bhayani , AI/ML engineer at Silvertouch and my project mentor and Mrs. Akansha Goplani, HR Executive at Silvertouch and my project coordinator, for their mindful guidance which was extremely valuable for the project, both theoretically and practically. They have been along with me throughout this project, regularly monitoring my work and guiding me in the right direction, in the online meetings that we had (in such adverse conditions due to Covid-19 pandemic).

Last but not the least, I would like to thank my extremely supportive parents and my fellow teammates, without whom this project would not have been a success.

Table of Contents

1. Problem Definition -

One of the hottest fields today in Computer Science, Human-Computer Interaction and HMI fields, has the problem of designing an intelligent machine that can estimate the age and gender of a person on its own, using face images, using well known concepts in artificial intelligence. Over the last decade, the world has witnessed a data boom. The rate of image uploads to the internet has been exponential, empowering scientists to handle such interesting computer vision problems efficiently. And, what's more interesting than applying AI concepts, to improve the quality of life ? Hence, many big companies around the globe, such as IBM, Google today, are investing in this field.

This is precisely my project at Silver Touchnologies. Given the facial image of a person, a machine learning / deep learning model must predict the person's age, gender (and perhaps even race/ethnicity as well). Basically, the model analyses the images, extracts features such as hair color, skin color, wrinkles, facial hair, etc and bases the predictions upon that. The model leverages the power of *Convolutional Neural Networks* to make predictions. We also wish to develop a model that has good generalizing capacity - meaning, it can make good predictions in real time as well.

However, age and gender classification is inherently a difficult task. The main reason is lack of availability of unbiased datasets with correct image labels. Ideally, for such a challenging task, a good model (especially that uses Deep Learning) would require well over a million images of myriad types to generalize well enough so that it could be used in the real world. However, in the interest of time and limited computational power available (in the form of GPUs on Google Colab), we chose the following datasets.

2. Datasets -

Age & Gender Classification - UTK Face dataset

It is a large-scale face dataset consisting of 20,708 face images. All images have annotations of age (0-116), gender (Male, Female), and ethnicity (White, Black, Asian, Indian, and Others). The images cover large variation in pose, facial expression, illumination, occlusion, resolution, etc.

Age Estimation - WIKI dataset

It consists of 62308 images crawled from all profile images from pages of people from Wikipedia with the same meta information. In the Wikipedia dataset, the age labels were assigned by first removing the images without timestamp (the date when the photo was taken). Then, assuming that the images with single faces are likely to show the personality and that the timestamp and date of birth are correct, a biological (real) age was assigned to each such image.

3. Literature Review -

The work began by reviewing some of the previously done work in this field, in the form of reading research papers. The papers provided a very good insight into the field of Facial Demographics as well as various techniques used by researchers to solve this challenging task.

Table I. List of Research Papers

Research Paper	Author(s)
Gender Classification Techniques: A Review	Preeti Rai and Pritee Khanna
Face Recognition Performance: Role of Demographic Information	Brendan F. Klare, Mark J. Burge, Joshua C. Klontz, Richard W. Vorder Bruegge, Anik K. Jain
Face Recognition and Age Estimation implications of Changes in Facial Features: A Critical Review Study	Rasha R. Atallah, Amirrudin Kamsin, Maizatul A. Ismail, Sherin A. Abdelrahman, Saber Zerdoumi
Age estimation via face images: A Survey	Raphael Angulu
Convolutional Neural Networks for Age and Gender Classification	Ari Ekmekji
Efficient facial representations for age, gender and identity recognition in organizing photo albums using multi-output ConvNet	Andrey V. Savchenko
Age Group and Gender Estimation in the Wild with Deep RoR Architecture	Ke Zhang, Ce Gao, Liru Guo, Miao Sun, Xingfang Yuan, Tony Han, Zhenbing Zhao, Baogang Li

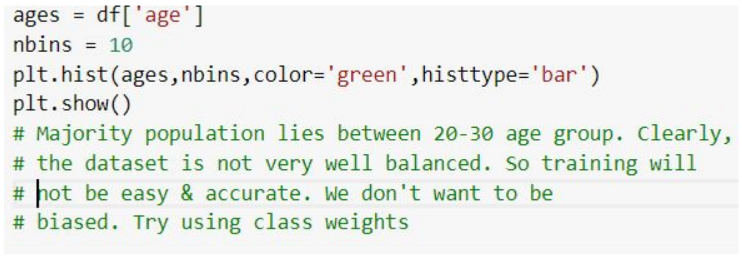
These papers talked about various strategies such as different ways of feature extraction such as Gabor filters, LBP, etc that can be used with classical ML models. The papers also showcased the various CNN architectures used to solve the problem. They pointed out the importance of facial demographics for many applications such as facial recognition, age estimation etc. in numerous areas such as security, law enforcement, biometrics, forensics etc. and challenges involved such as pose, illumination, expression, aging, ethnicity, lifestyle, environment, databases, dataset.

4. Exploratory Data Analysis (EDA)

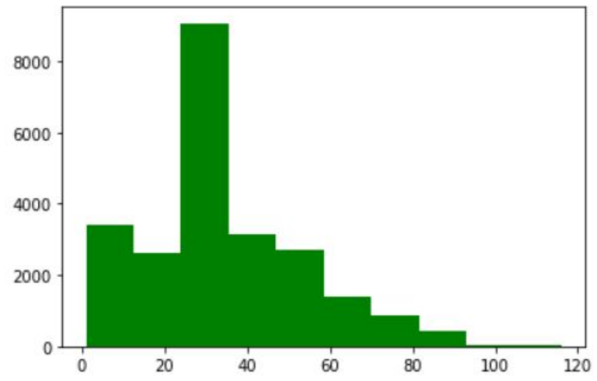
EDA is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. The main purpose of doing this is to understand data distribution and to gain valuable insights from it, because our model is only as good as the data it gets to train !
Since, I was dealing with images, I categorized my EDA into 2 types - Pixel value based EDA and metadata based EDA.

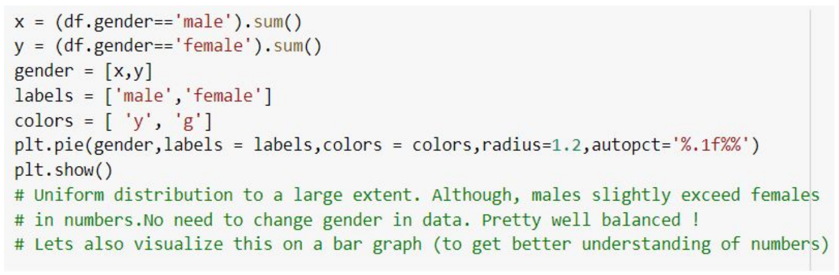
1. **Metadata based EDA -**
   This analysis is based on the actual data that we get, as in tabular format (read as csv in pandas). Various techniques include, grouping 2 columns and analysing dependency of 1 on the other, analysing individual columns,etc.

- UTK Face dataset -

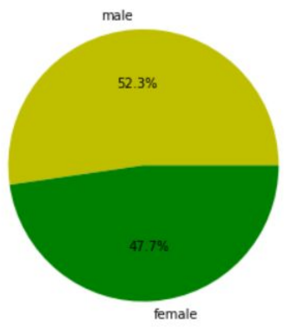


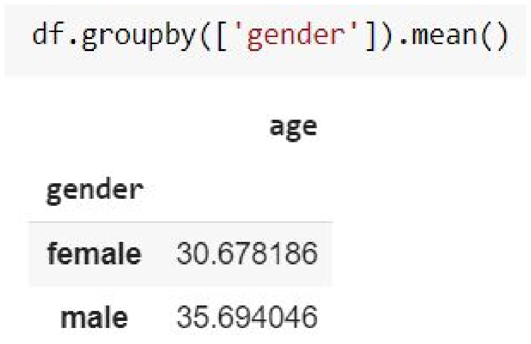
```
ages = df['age']
nbins = 10
plt.hist(ages,nbins,color='green',histtype='bar')
plt.show()
# Majority population lies between 20-30 age group. Clearly,
# the dataset is not very well balanced. So training will
# not be easy & accurate. We don't want to be
# biased. Try using class weights
```





```
x = (df.gender=='male').sum()
y = (df.gender=='female').sum()
gender = [x,y]
labels = ['male','female']
colors = [ 'y', 'g']
plt.pie(gender,labels = labels,colors = colors,radius=1.2,autopct='%.1f%%')
plt.show()
# Uniform distribution to a large extent. Although, males slightly exceed females
# in numbers.No need to change gender in data. Pretty well balanced !
# Lets also visualize this on a bar graph (to get better understanding of numbers)
```

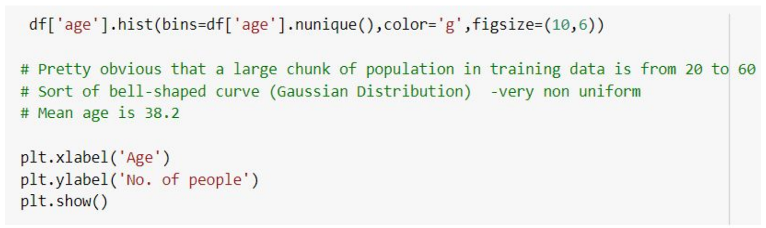




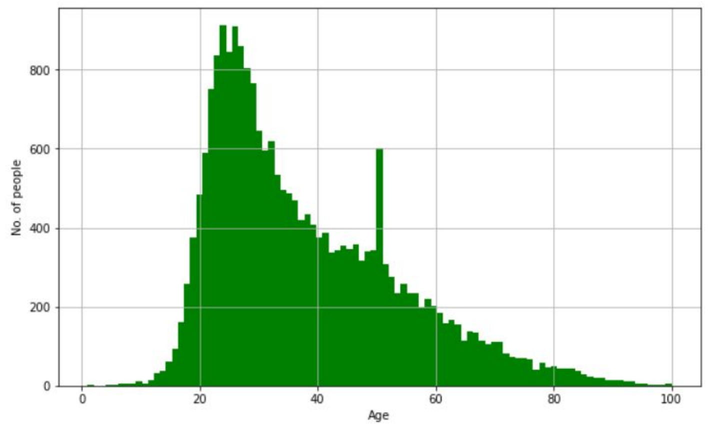
```
df.groupby(['gender']).mean()
```

	age
gender	
female	30.678186
male	35.694046

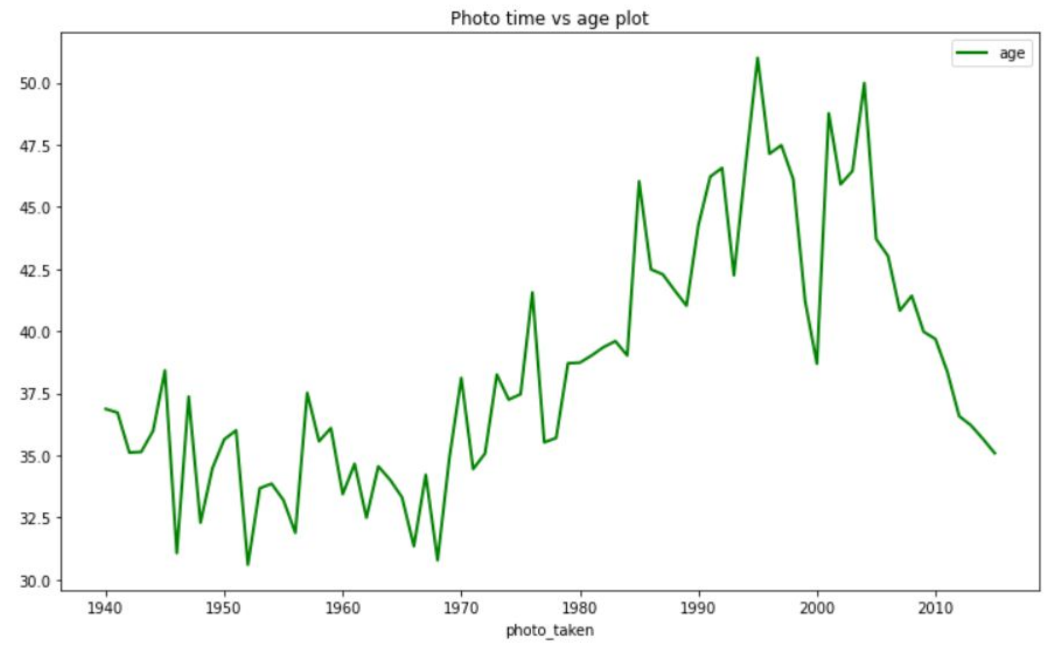
- WIKI dataset -



```
df['age'].hist(bins=df['age'].nunique(),color='g',figsize=(10,6))

# Pretty obvious that a large chunk of population in training data is from 20 to 60
# Sort of bell-shaped curve (Gaussian Distribution)  -very non uniform
# Mean age is 38.2

plt.xlabel('Age')
plt.ylabel('No. of people')
plt.show()
```



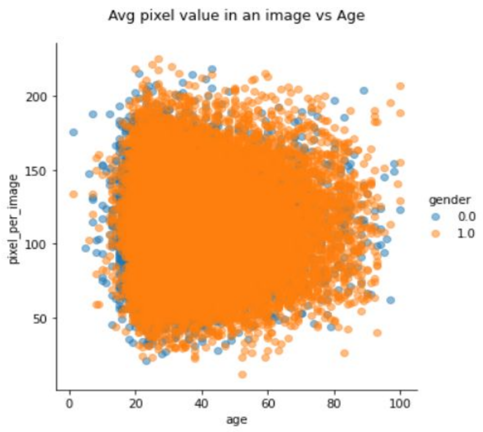
```
df_copy.plot(x = 'photo_taken', y = 'age', figsize = (12,7), color = 'green', linewidth =2)
plt.title("Photo time vs age plot")
plt.xticks(rotation=0)
plt.show()

# Clearly, more aged people  were photographed during 1980-2005. There seems to be some
# correlation !
```



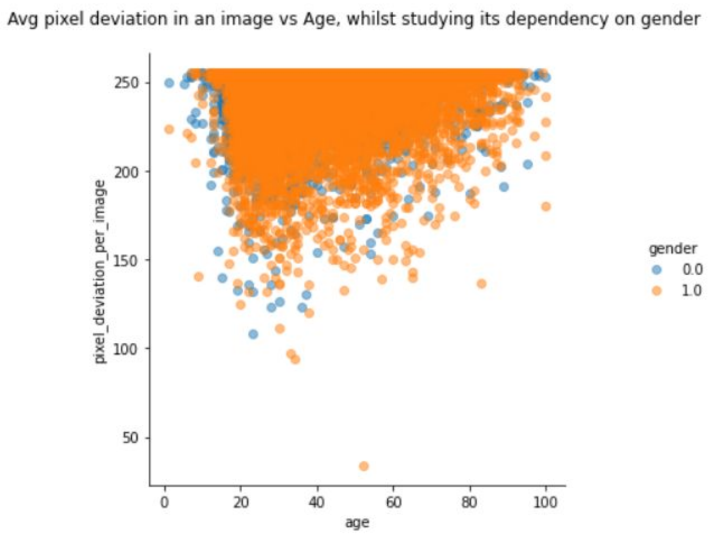
2. **Pixel-value based EDA -**

This analysis is based on the pixel values (WIKI dataset) that we extract from the image data using the *img_to_array* function of keras. Try to see various patterns & correlations in pixel values.

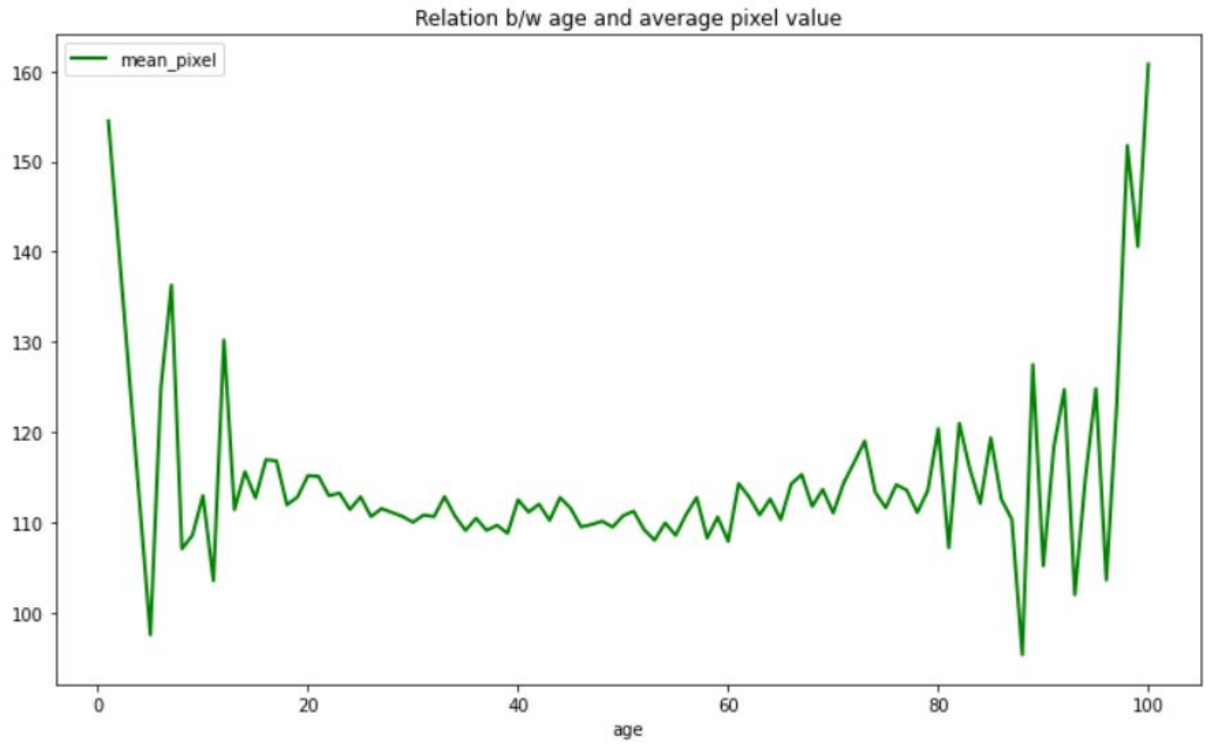
```
sns.lmplot(x='age', y='pixel_per_image', hue='gender', data=x, fit_reg=False, scatter_kws={'alpha':0.5})
plt.title('Avg pixel value in an image vs Age\n')
plt.show()
```



```
sns.lmplot(x='age', y='pixel_deviation_per_image', hue='gender', data=x, fit_reg=False, scatter_kws={'alpha':0.5})
plt.title('Avg pixel deviation in an image vs Age, whilst studying its dependency on gender \n')
plt.show()
```



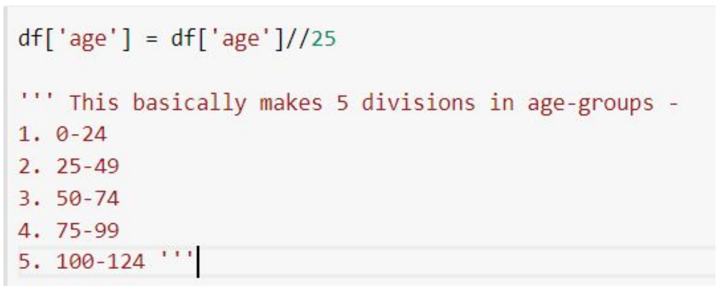
```
y.plot(x = 'age' ,y = 'mean_pixel', figsize = (12,7), color = 'green', linewidth =2)
plt.title('Relation b/w age and average pixel value')
plt.xticks(rotation=0)
plt.show()
```



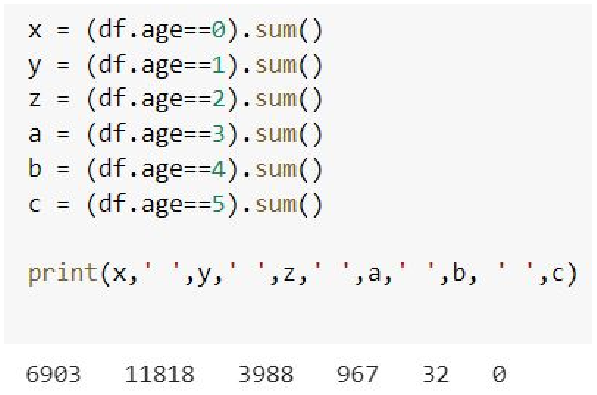
5. Data Preprocessing

Age & Gender Classification - UTK Face dataset

Here, my model was to predict the gender of the person, and classify his/her age into 5 categories - 0-24 , 25-49, 50-74, 75-99 and 100-124.
Ages we divided into such categories, by simply taking their integer division, which can be done using // in python.



```
df['age'] = df['age']//25

''' This basically makes 5 divisions in age-groups -
1. 0-24
2. 25-49
3. 50-74
4. 75-99
5. 100-124 '''
```



```
x = (df.age==0).sum()
y = (df.age==1).sum()
z = (df.age==2).sum()
a = (df.age==3).sum()
b = (df.age==4).sum()
c = (df.age==5).sum()

print(x,' ',y,' ',z,' ',a,' ',b, ' ',c)
```

```
6903    11818    3988    967    32    0
```

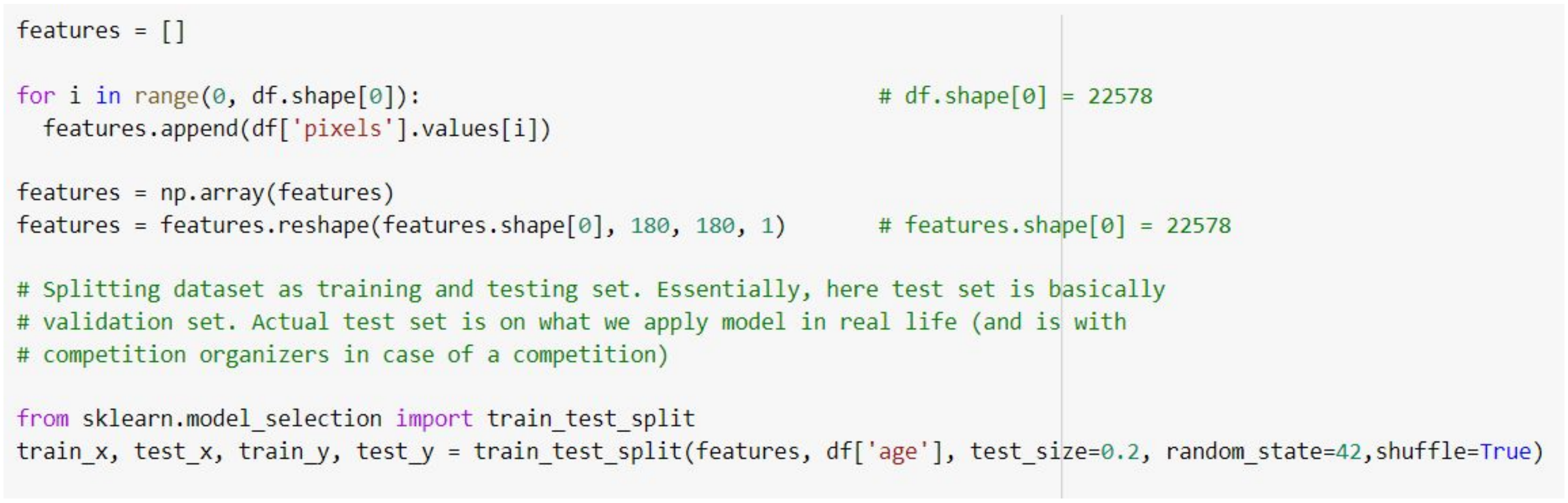
So, the first obvious step after extracting the csv file, was to one-hot encode age and gender. This was done using *to_categorical* function in keras. Similarly, the image is resized, normalized and then converted to its numpy array. All this was done inside a generator, that returned numpy arrays which were then fed to the model.
The class-imbalance (for ages) was handled by *class_weights* in keras. But I found that the imbalance was so large that even *class_weights* couldn't handle it.

Age Estimation - WIKI dataset

First, I had to load the matlab file and extracted the csv file from the matlab file. Then I found the DOB from it’s *datenum* format. The data now was for 62328 images. But not all images were good enough for training. Many images had more than 1 face in them and many didn’t even have a face, so i deleted those images. Still, many images were not of good quality. A measure for this was the *‘face_ score’* column. I kept the threshold to be 3.0 initially. The data that I had now was for 22578 images. Later, I also trained a model where I reduced the threshold to 1.75. That allowed 34,200 images, thereby improving the generalizability of the model.

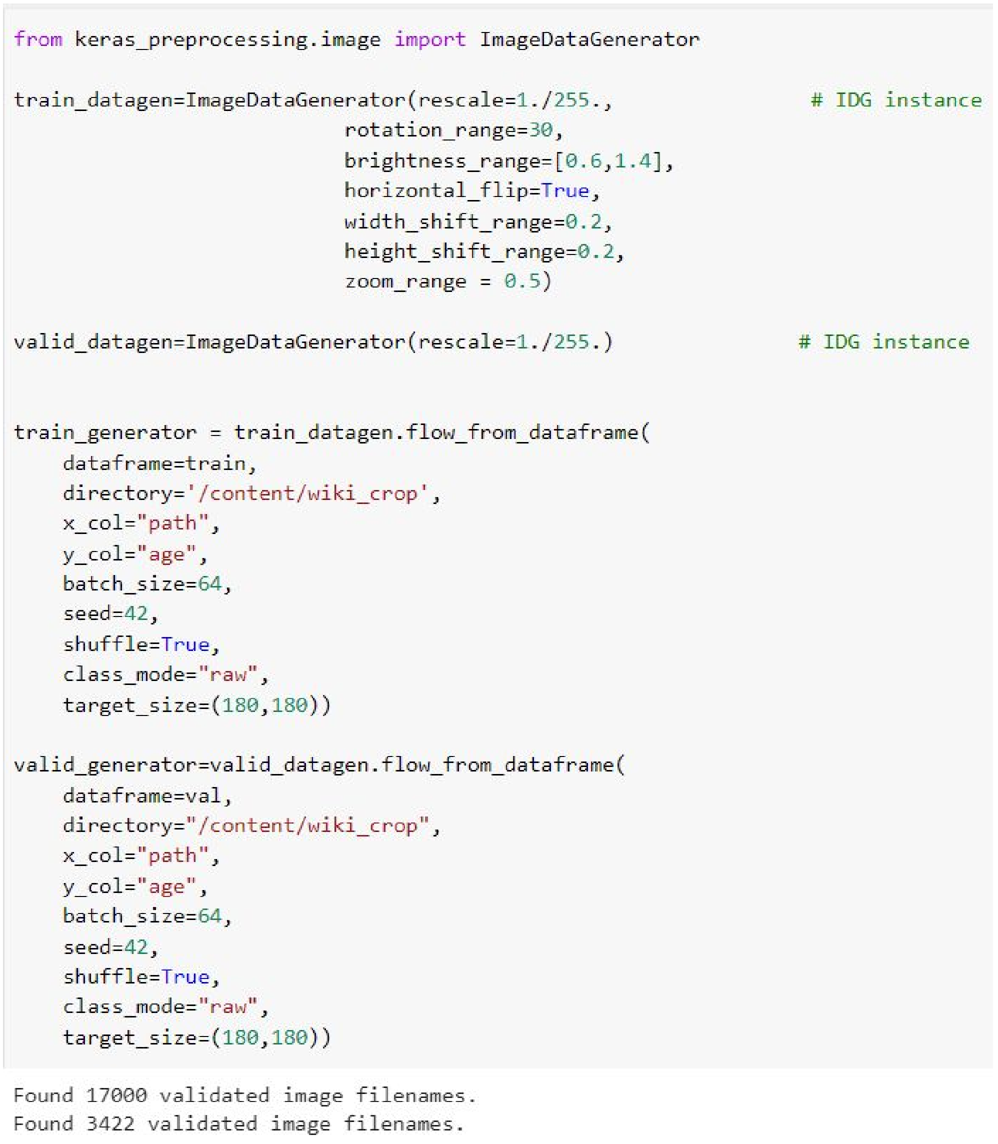
I also made sure that all images then had ages between 0 and 100. Finally, I dropped off all unnecessary columns.

For age estimation, I tried to create models in 2 ways - One, in which I found the exact pixel values of the images, normalized them, reshaped them into desired dimensions (no. of images , 180, 180, 1). Then I used *train_test_split()* function of scikit-learn library to split it into train_x, train_y, val_x, val_y (80:20 split).



```
features = []

for i in range(0, df.shape[0]):                                   # df.shape[0] = 22578
  features.append(df['pixels'].values[i])

features = np.array(features)
features = features.reshape(features.shape[0], 180, 180, 1)       # features.shape[0] = 22578

# Splitting dataset as training and testing set. Essentially, here test set is basically
# validation set. Actual test set is on what we apply model in real life (and is with
# competition organizers in case of a competition)

from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(features, df['age'], test_size=0.2, random_state=42,shuffle=True)
```

The second type of model that I developed used the *ImageDataGenerator* class in keras. The logic of doing so is to load only a certain portion of the dataset into memory (so that Colab RAM doesn't get exhausted), and also to augment the training set so that the model never sees the same image twice and generalizes much better to unseen data.



```
from keras_preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1./255.,             # IDG instance
                            rotation_range=30,
                            brightness_range=[0.6,1.4],
                            horizontal_flip=True,
                            width_shift_range=0.2,
                            height_shift_range=0.2,
                            zoom_range = 0.5)

valid_datagen=ImageDataGenerator(rescale=1./255.)            # IDG instance


train_generator = train_datagen.flow_from_dataframe(
    dataframe=train,
    directory='/content/wiki_crop',
    x_col="path",
    y_col="age",
    batch_size=64,
    seed=42,
    shuffle=True,
    class_mode="raw",
    target_size=(180,180))

valid_generator=valid_datagen.flow_from_dataframe(
    dataframe=val,
    directory="/content/wiki_crop",
    x_col="path",
    y_col="age",
    batch_size=64,
    seed=42,
    shuffle=True,
    class_mode="raw",
    target_size=(180,180))
```

```
Found 17000 validated image filenames.
Found 3422 validated image filenames.
```

Note: Only the training set is augmented.

Needless to say, the latter design works much better. Then, I also realized that the imbalance in the dataset is way too much for it to be covered by *class_weights*. In fact, 20422 out of a total of 22578 images lied between 21 & 75 years of age, which amounts to a mind-boggling 90.4 % of data.
To account for this, I trained the latter model (IDG) in age estimation also on a modified dataset which had images between ages of 21 & 75.

6. Models and Experimentation

Age & Gender Classifier - UTK Face Dataset

The architecture that was used for gender/age classification tasks is described below -

- Basic idea was to have a block of convolutional and pooling layers, that help the model learn features and representations, followed by a set of FC layers, which are used to classify.

- My model took an input image of size (198,198,3). It was a RGB image.
- Classifies age into 5 categories & gender into 2 categories.

- To predict both age and gender at once, I used a multi-output classification technique. This involved having a common part of feature extraction for both age and gender, but separate FC layers. The branching point is just after features have been extracted for both. I kept the feature extraction part common for both age and gender classification, because as per researchers, features are more or less along the same lines for predicting age and gender.

- Loss functions for both age and gender were *categorical_crossentropy.*
- Batch Size was 32.

- LearningRateSchedular of keras was a callback function that halved the lr every 5 epochs. Initial lr was 0.008.

- The model was trained for a total of 44 epochs in 2 rounds of 22 epochs each. After the 1st round, I found the validation loss to be increasing, which is a clear sign of overfitting - essentially the model stopped learning due to the small learning rate (0.0005) by 22 epochs. Hence, I trained the model for an additional 22 epochs with a new initial lr of 0.002.
  Any further training was leading to overfitting and reduced the ability of the model to generalize.

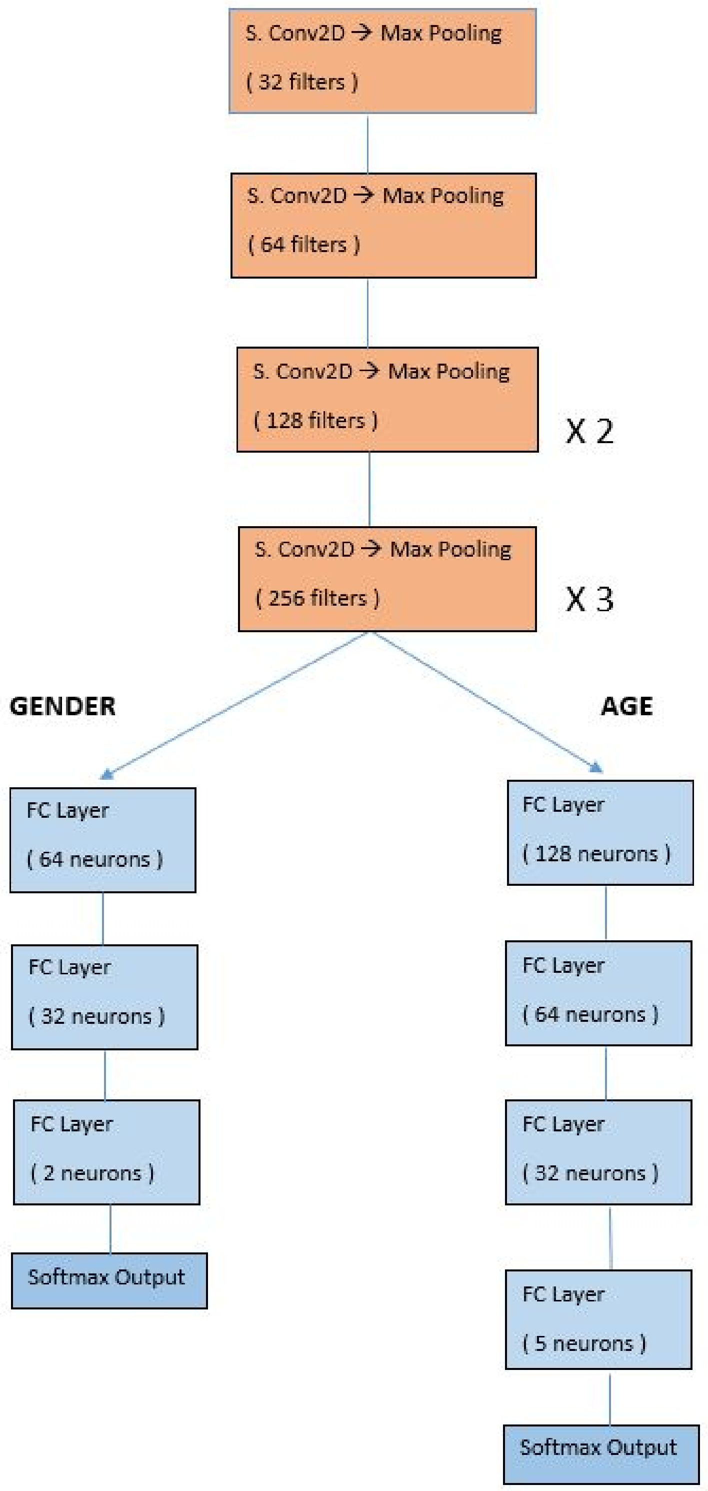


Fig : Model architecture for age & gender classification

Some more layer-specific details -

- Filter size in all convolutional layers is 3x3.

- All layers are followed by BatchNormalization. All convolutional layers are followed by MaxPooling2D (2x2), and all FC layers are followed by Dropout (0.2~0.3). Convolution layers with filters 128 or more are followed by Spatial Dropout (0.1~0.2).

- All layers have activation function *relu* .

Age Estimator - WIKI Dataset

This is a regression problem. We have to predict the exact age of a person.
The architecture that was used for gender/age classification tasks is described below -

- Input size is 180 x 180 x 3.

- Loss function was *mse*, optimizer was *Adam* and metrics = [*mse*,*mae*].

- LearningRateSchedular was used with an initial lr of 0.006 that halved lr every 12 epochs.

- Model was trained for a total of 95 epochs (45 +50). At the end of the first round, I found the improvement in training performance to be very small.

This was mostly due to the miniscule learning rate at the end of 45 epochs. Hence, I trained the model for another 50 epochs , by re-establishing a new initial lr of 0.002.

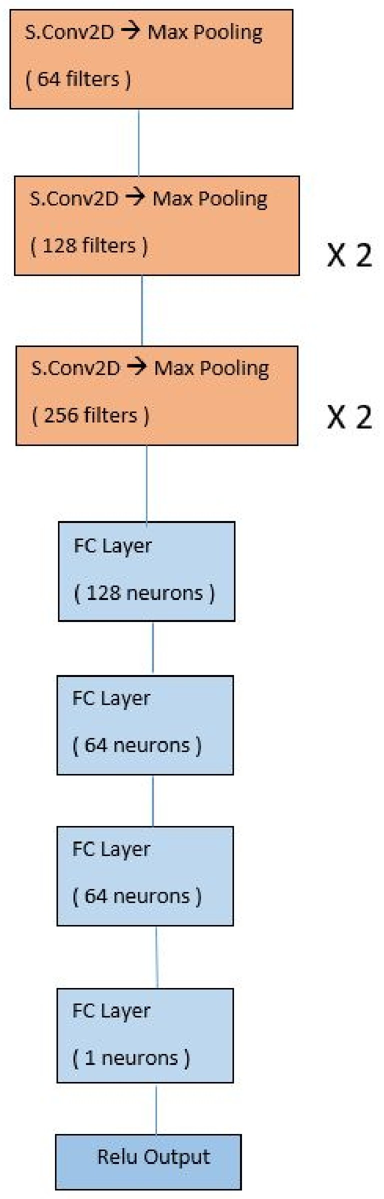

S.Conv2D → Max Pooling (64 filters)
S.Conv2D → Max Pooling (128 filters)
X 2
S.Conv2D → Max Pooling (256 filters)
X 2
FC Layer (128 neurons)
FC Layer (64 neurons)
FC Layer (64 neurons)
FC Layer (1 neurons)
Relu Output

Fig : Model architecture for age estimation

Some more layer-specific details -

- Filter size is 3x3 throughout.

- Like before, every FC layer is followed by Dropout (0.2~0.3), and all layers are followed by BatchNormalization.Convolution layers with filters 128 or more are followed by Spatial Dropout (0.1~0.2).

- All layers have activation function *relu.*

The designs showcased above were the ones that gave the best results. They were obtained after lots of research - mostly experimentation, advice from my mentor, reading many research papers, etc. Here are some of the crucial areas where I experimented a lot -

1. **Model Structure -** By far, the most important thing that affects any model's performance.
    - **Layers -** I experimented with the number of FC layers, number of convolutional layers, size of convolutional filters , number of neurons in an FC layer.
      I found that convolutional filters of 3x3 size, increasing in number with increasing depth (and in powers of 2), gave the best results. Similarly, it's advisable to keep the number of neurons also in powers of 2.
      I also tried different sizes of *MaxPool2D* filters, and tried changing

their positions, i.e instead of keeping it after every convolution , I kept it after every 2 or every 3 convolutions. But, the results were not good.

Lastly, at the end, I tried using *SeperableConv2D* filters (along with *SpatialDropout2D ,*that's mentioned below) instead of the regular *Conv2D* filters. And the results were astonishing. Overfitting of my models dropped drastically, with a slight increase in performance.

- **Regularization** - I tried myriad types of regularization techniques right from Dropout to L2 regularization. My models attained best performance by simply using dropout with dropping probability between 0.2 - 0.35. I found L2 (or its cousins L1, L1_L2) to have an adverse impact on the loss. Performance was not the best.
  Finally, when I replaced *Conv2D* with *SeperableConv2D ,* I added SpatialDropout2D (which goes hand in hand with *SeperableConv2D)* after each such convolution in the latter layers. It helped reduce overfitting very much (generalization gap (mae) went from about 3 to 0.3 in age estimation model).

2. **Learning Rate-** Typically, a good learning rate is considered to be between 1e-3 and 1e-5. I tried experimenting with all these learning rates. As, we all know, learning rate should ideally decrease with time otherwise the model may never attain the local minima (and keep oscillating nearby), first I tried

using a step decay as

```
LearningRate = LearningRate * 1/(1 + decay * epoch)
```

Then, I also tried *LearningRateSchedular* of keras, that allows us to reduce the learning rate after a certain number of epochs. This gave the best result.

Varying from model to model and from task to task, I experimented with this ‘certain number of epochs’, with a goal to converge to the best result.

3. **Batch Size -** Ideally batch sizes are chosen in powers of 2. I also did the same - experimenting with 32, 64, 128, 256 in all my models. I experienced much noisier training (with more of a regularization effect) when the batch size was 32 or 64. In almost every such trial, my model produced best results when batch size was either 32 or 64.

4. **Data Augmentation in Age Estimation-** Of course, we all know there are many different ways to augment data in the keras *ImageDataGenerator* class. I gave a shot to that as well- brightness, rotation, hor/ver shifts, zooms, horizontal flips, etc.
   Firstly, it’s very important to note that this augmentation be applied only to the training set and not to the validation set.
   Secondly, I found that when I incorporated a lot of augmentation in my training set, my training performance decreased to some extent i.e validation performance was better than training performance, in the later epochs !

5. **Dataset in Age Estimation -** The dataset I had, WIKI for age estimation was heavily biased towards people of ages between 20 & 75. As a result, with so few images to train and that too when about 90% belonged to a particular class, the model would overfit to those images very easily. In layman terms, when deployed for use in the real world, more often than not the model will predict an age between 20 & 75. And, I found such heavy imbalance can't be dealt with using *class_weight* in keras or by data augmentation.
   So, I trained another model, only on those images that had people between 21 & 75 years old. This way, the model has a good chance of predicting the right age of a person in real time as well (provided it's between 21 & 75). However, I found its performance was not the best in real life. So, I scrapped it.
   Finally, I trained another model, wherein I reduced the threshold for keeping images to 1.75 from 3.0. Therefore, I got a total of 34,200 images as compared to 22,578 before. Training CNN on this dataset, along with Image Data Generator to augment the training data, produced the best result (real-time), as the model was able to generalize the best. That's perhaps because the model also took worse quality images as input and so becomes more robust to inputs in real time.

7. Results -

Age & Gender Classifier -

Results of the models that I tried in order of increasing performance (val. set).

Table II: Results for age & gender classifier

TRAIN_LOSS	TRAIN_AGE_ACC	TRAIN_ GENDER_ACC	VAL_LOSS	VAL_AGE_ACC	VAL_GENDER_AC C
1.8366	0.6626	0.9092	2.1247	0.6329	0.8745
0.7485	0.9103	0.9571	1.6166	0.7042	0.8798
0.9342	0.8366	0.9319	1.6198	0.7193	0.8806
0.0527	0.9904	0.9912	1.1161	0.7410	0.8978
0.7864	0.8387	0.9931	0.9038	0.7418	0.8851
0.1389	0.9646	0.9944	0.7495	0.8279	0.9502

Age Estimator -

Here are some of the results of the models that I tried in order of increasing performance (validation set).

Table III : Results for age estimator

TRAIN_ LOSS	TRAIN_ MAE	VAL_LOSS	VAL_MAE
177.7641	11.1566	199.3812	11.0324
17.6605	2.9786	83.6385	6.9328
89.6336	7.3960	66.4088	6.2225
58.5194	5.9128	30.8964	5.9402
44.6834	5.1108	44.0836	5.5679

Clearly, the models do not have state of the art accuracy so as to be deployed in real life, but still are pretty good. Models that are actually deployed in real life, have to be trained over millions of images and mostly use transfer learning whereby they make use of pre-built weights of convolutional layers. These convolutional layers, in general, are much better feature extractors (thereby lead to higher performance), because such models have been developed by researchers in the field of Deep Learning over months after rigorous training and testing over huge datasets containing millions of images.

8. Deployment -

Now, my deep learning model looks good in Google Colab. But how to make this available to use for others ? We need to deploy the model.
The easiest way to deploy our model is to use **Flask** which is a python-based library that allows us to quickly deploy applications by creating local servers. Flask deployment takes place not in Jupyter Notebook but using either CMD /Anaconda Prompt on Windows (or Terminal in Linux/Mac). First we need to install Anaconda, the latest versions of python, keras, tensorflow for deployment to take place.

I tried my luck on deploying, and the model gave fairly good results. Have a look ! Remember to crop the images (hair to chin) before feeding it to the model, as models were trained on cropped & aligned faces and I haven't yet created a face detector function while deploying it in Flask.

Age & Gender Classifier -

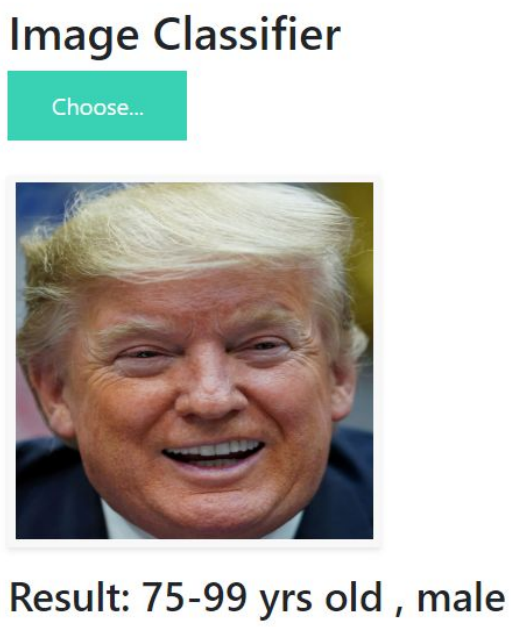


Image Classifier

Result: 75-99 yrs old , male

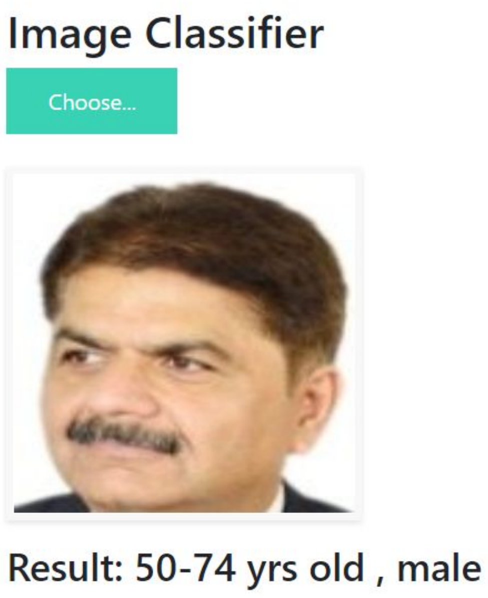


Image Classifier

Result: 50-74 yrs old , male

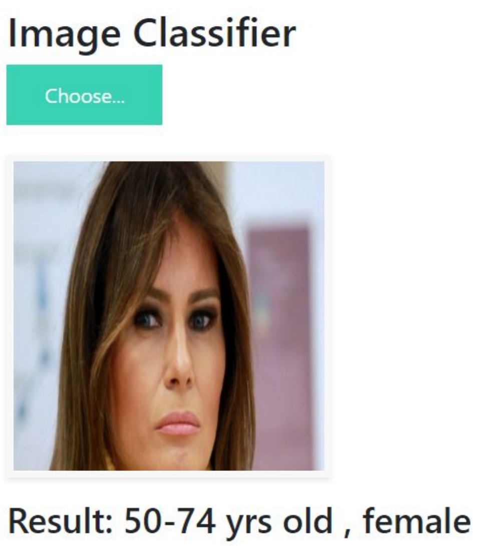


Image Classifier

Result: 50-74 yrs old , female

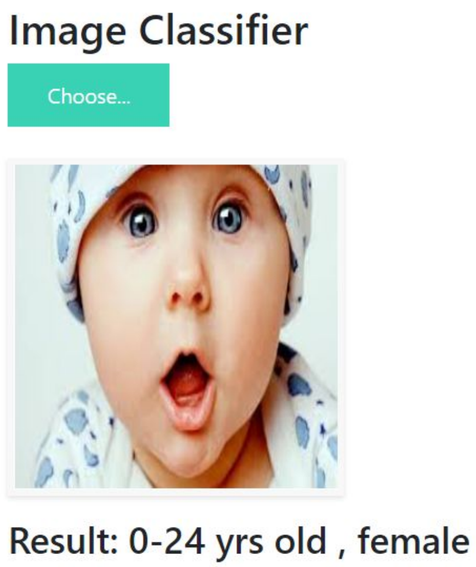


Image Classifier

Result: 0-24 yrs old , female

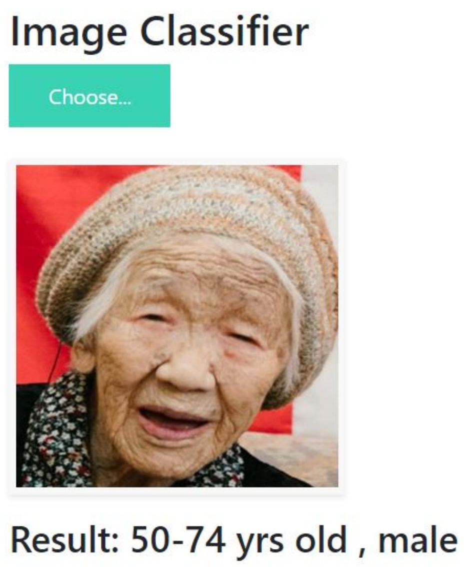


Image Classifier

Result: 50-74 yrs old , male

A misclassified image !

Age Estimator-

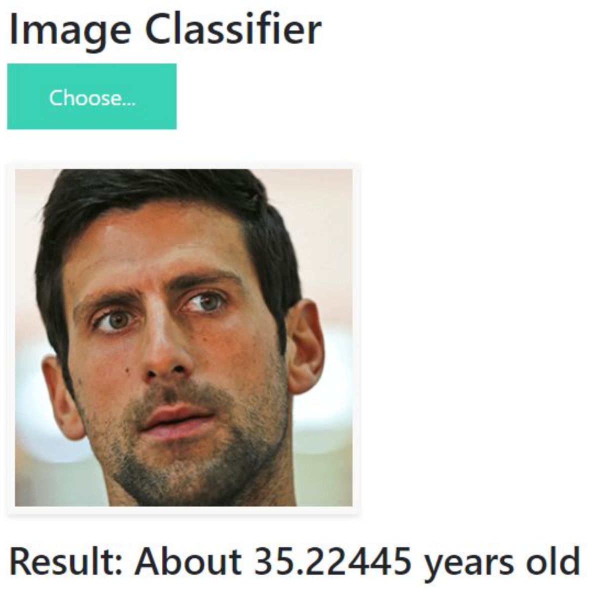


Image Classifier

Result: About 35.22445 years old

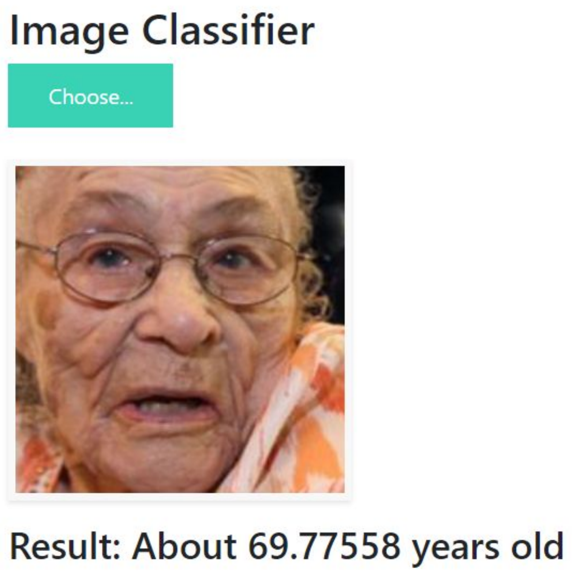


Image Classifier

Result: About 69.77558 years old

9. Appendix

Appendix A

Table IV : List of abbreviations used.

Short Form (in text)	**Full Form**
EDA	Elementary Data Analysis
LBP	Local Binary Patterns
CNN	Convolutional Neural networks
ML/ DL	Machine Learning/ Deep learning
FC	Fully Connected
MAE	Mean absolute error
MSE	Mean squared error
S.Conv2D	Separable 2D Convolution

Appendix B

- **SeperableConv2D layer-**

Separable convolutions consist of first performing a depthwise spatial convolution (which acts on each input channel separately) followed by a pointwise convolution which mixes the resulting output channels. Intuitively, separable convolutions can be understood as a way to factorize a convolution kernel into two smaller kernels, or as an extreme version of an Inception block.

- **SpatialDropout2D layer -**

It refers to the *Spatial2D* version of normal dropout regularization.This version performs the same function as Dropout, however it drops entire 2D feature maps instead of individual elements. If adjacent pixels within feature maps are strongly correlated (as is normally the case in early convolution layers) then regular dropout will not regularize the activations and will otherwise just result in an effective learning rate decrease. In this case, SpatialDropout2D will help promote independence between feature maps and should be used instead.

From my experience, I found SpatialDropout2D to be very strong. I kept its value between 0.1 and 0.2 mostly. Any value more than this, led to underfitting of the model. That being said, I think, it is a wonderful tool to reduce overfitting, in the **convolutional part** of a neural net, provided it's used cautiously.

10. References

About Our Company - Silver Touch Technologies. (n.d.). Retrieved from https://www.silvertouch.com/about-us/

Alexander, Rosebrock, A., Peng, Sam, Abraham, N., Wilf, . . . Seed, L. (2020, April 18). Keras: Multiple outputs and multiple losses. Retrieved June 11, 2020, from https://www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/

Bansari, S. (2019, April 30). Introduction to how CNNs Work. Retrieved June 10, 2020, from https://medium.com/datadriveninvestor/introduction-to-how-cnns-work-77e0e4cde99b

Chollet, F. (n.d.). The Keras Blog. Retrieved June 11, 2020, from https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Papers with Code - Age Estimation. (n.d.). Retrieved June 23, 2020, from https://paperswithcode.com/task/age-estimation

Team, K. (n.d.). Keras documentation: Keras layers API. Retrieved June 25, 2020, from https://keras.io/api/layers/