A REPORT

ON

FACIAL DEMOGRAPHICS

By -

Archisman Das(2304079)

2nd year ETC

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.

Wikipedia

The 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 , 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.
```

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.

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.

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).

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)
```

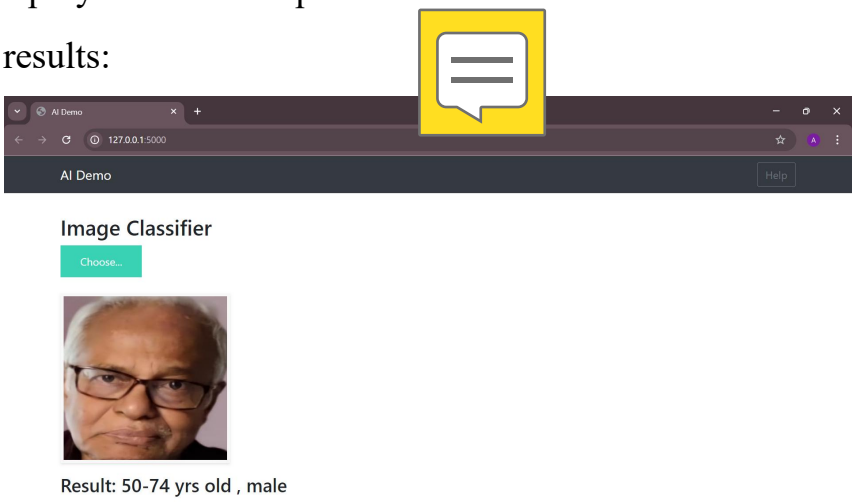
Then, I also tried *LearningRateSchedular* of keras, that allows us to reduce the learning rate after a certain number of epochs. This gave the best result. Varying from model to model and from task to task, I experimented with this ‘certain number of epochs’, with a goal to converge to the best result.

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.

Deployment -

The easiest way to deploy the 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). Firstly it is needed to install Anaconda, the latest versions of python, keras, tensorflow for deployment to take place.

results:



Github link:https://github.com/Archismandas2004/Age_and_Gender_Detection.git