**Gender Classification (Transfer Learning)**

From week 25th July to 31st July, I worked about 30 hours a week on tasks of Gender classification. This was a self-Learning project for me. I built a Convolutional Neural Network to classify the gender given an image of face of a person. The data set was taken from <https://s3.amazonaws.com/matroid-web/datasets/agegender_cleaned.tar.gz>

The code can be found on https://github.com/mraunak/Research-Assistant-Work-/blob/main/GenderClassification\_TransferLearning.ipynb.

**Step1**

Model Conversion: Converted the VGG face descriptor model to TensorFlow  
format. Model weights was taken from following link and was transferred to VGG16 model in keras framework.

Weights were extracted from VGG\_FACE.t7 file and were stored in VGG\_Face\_pytorch.pt file.

<https://m-training.s3-us-west-2.amazonaws.com/dlchallenge/vgg_face_torch.tar.gz>

Reference was taken from VGG face research paper:

<http://www.robots.ox.ac.uk/~vgg/software/vgg_face/>

**Step2**

Transfer Learning: Used pre-trained weights from the VGG16 model used in the VGG face research paper for implementation of VGG16 gender classification network in keras framework. The weights of all the fully convolutional layers were set to non-trainable but the weights of last three layers (2 fully connected layers and output layer) were trained.

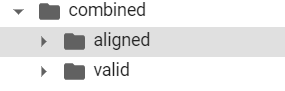
**Dataset**:

The dataset was taken from the following link.

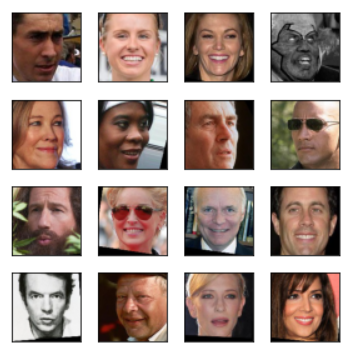
<https://s3.amazonaws.com/matroid-web/datasets/agegender_cleaned.tar.gz>

Age Gender set

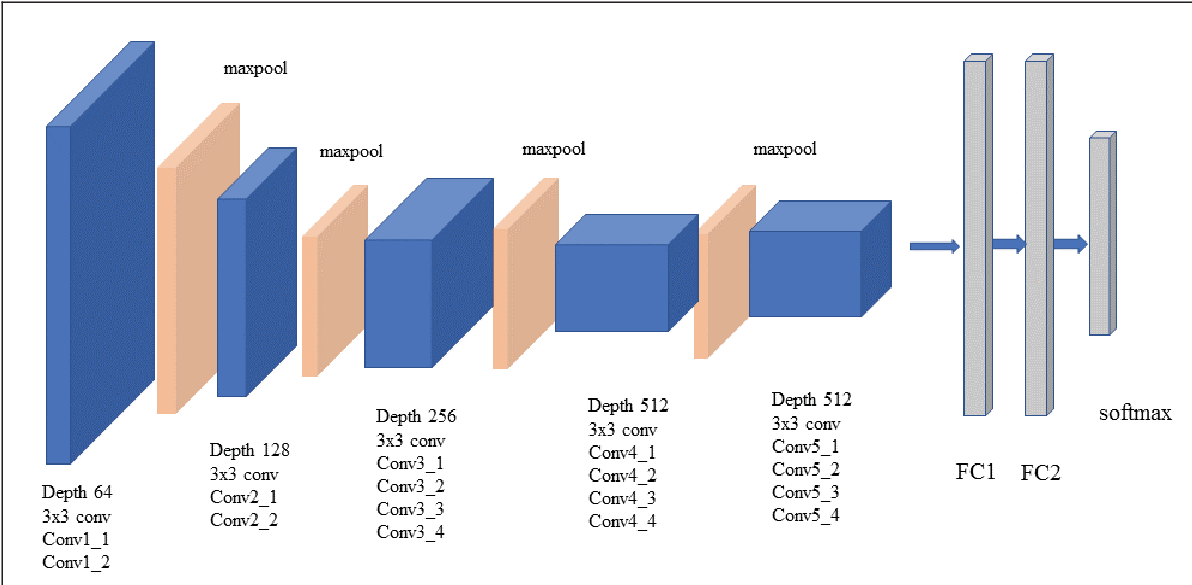
There are two folders of dataset:



Sample Images



**Architecture Used**



# Features of VGG-16 network Used

1. Input Layer: Input to the network is colored images as with the size 64 x 64 and 3 channels i.e. Red, Green, and Blue.

The size of the image was reduced to increase the speed of computation and to handle a large number of images

1. Convolution Layer: The images pass through a stack of convolution layers where every convolution filter has a very small receptive field of 3 x 3 and stride of 1. Every convolution kernel uses row and column padding so that the size of input as well as the output feature maps remains the same or in other words, the resolution after the convolution is performed remains the same.
2. Batch Normalization: It improves the performance and provide stability to the network. It increases the speed of the training. It helps to deal with the problem of internal covariant shift. It has also shown its effect on activation functions (it makes activation function viable). It provides some regularization effect by adding a little noise.
3. Max pooling: It is performed over a max-pool window of size 2 x 2 with stride equals to 2. It reduces the size of the feature map by 2 (64\*64 gets converted to 32\*32). It progressively reduces spatial size, hence the number of parameters and computation in the network. Max pooling helps combat over-fitting by providing an abstracted form of the representation and make model translational invariant.
4. Dropout: Dropout was used here to take care of overfitting Dropout reduces the co-adaptation between the units and forces each unit to learn independently. This improves the performance of the model.
5. The hidden layers have ReLU as their activation function. It is widely activation function as it helps to combat the issue of vanishing gradient, it induces sparsity to the model. Sparse representations seem to be more beneficial than dense representation.
6. The first two fully connected layers have 4096 channels each and the third fully connected layer which is also the output layer have 2 channels. The final output layer has sigmoid as the activation function. Since, it is binary classification.
7. Metric used: Accuracy, Precision, Recall, f1\_score.
8. Optimizer used: Adam optimizer uses the combination of momentum and RMSprop to converge faster.
9. Loss Function: Binary Cross entropy loss. It is widely used loss function for binary classification.

**Accuracy of Model**

**Training Accuracy** of model: 98.44% on 25 epochs

Validation Accuracy of model: 92%

Total number of data used for the final evaluation after training the model=7268

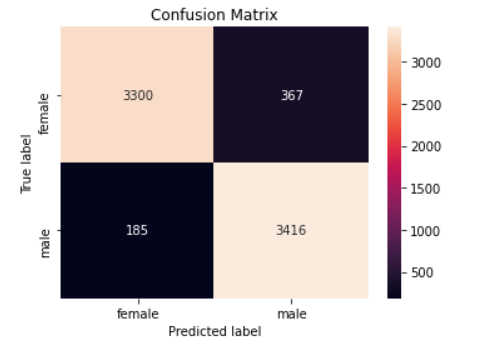
Recall: 0.949

Precision: 0.903

f1\_score: 0.925

Accuracy: 92.41%

Below is the confusion matrix on the test data



Below is ROC Curve plot

