**Gender Classification(Transfer Learning)**

**Task1**

Model Conversion: Convert 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/>

**Task2**

**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

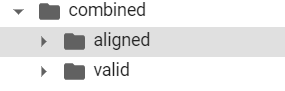
**Dataset**:

The dataset was taken from thr following link

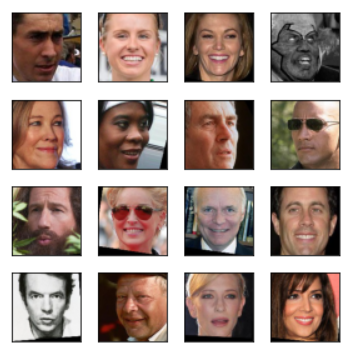
<https://s3.amazonaws.com/matroid-web/datasets/agegender_cleaned.tar.gz>

Age Gender set

There are two folders of dataset:



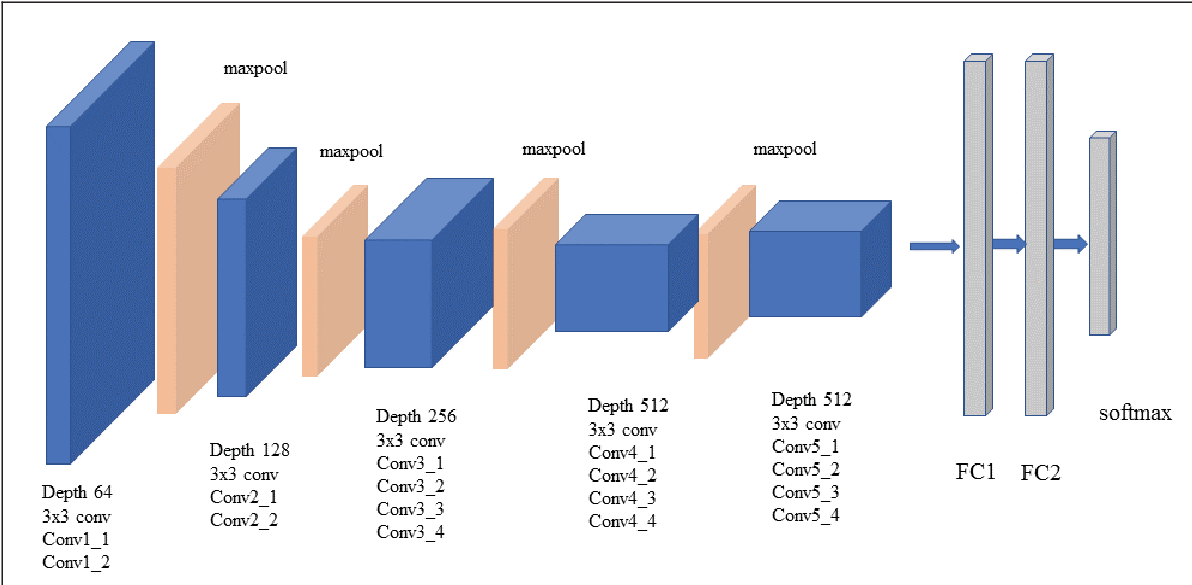
Sample Images

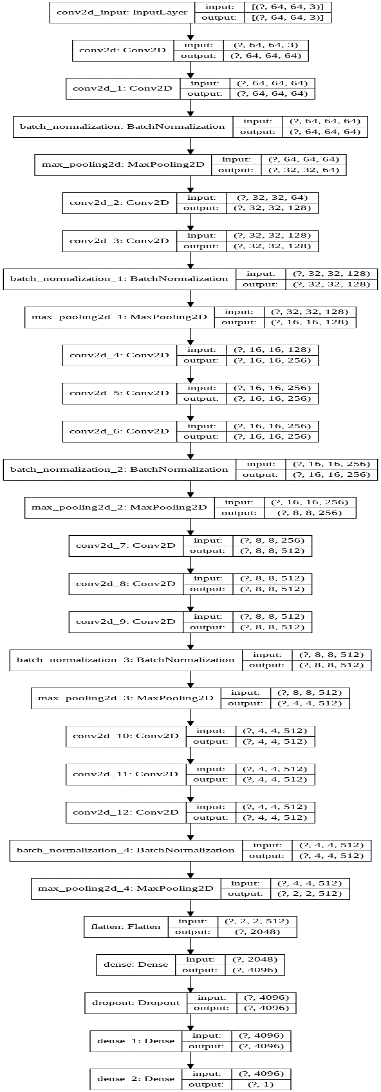


Both aligned and valid folders contain 140 sub folders (70 Female and 70 Male folders). Each folder is labeled by age and gender(e.g 01\_M, 01\_F ---01 for 1 year old and ‘M’ or ‘F’ for male or female respectively)

Each folder has different number of images highest being 249 and minimum being 0.

**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: Accuarcy, Precision, Recall, f1\_score
8. **Optimiser** 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

**Task3**

**Final Tensorflow classifier model trained on the gender dataset  
(architecture and weights) ---**

**model --🡪** my\_model.h5

**Weights**-🡪 my\_model\_weights.h5

**Task4**

Code for training and evaluating the model-🡪 Matroid\_Test.ipynb

**Task5**

**Results/metrics (for all the classes and overall) obtained for the  
trained 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

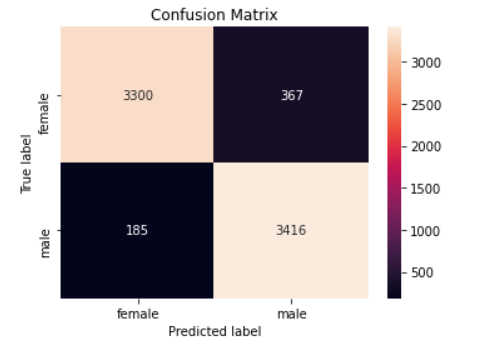
**Recall: 0.949**

**Precision: 0.903**

**f1\_score: 0.925**

**Accuracy: 92.41%**

Below is the confusion matrix on the test data



**Below is ROC Curve plot**

