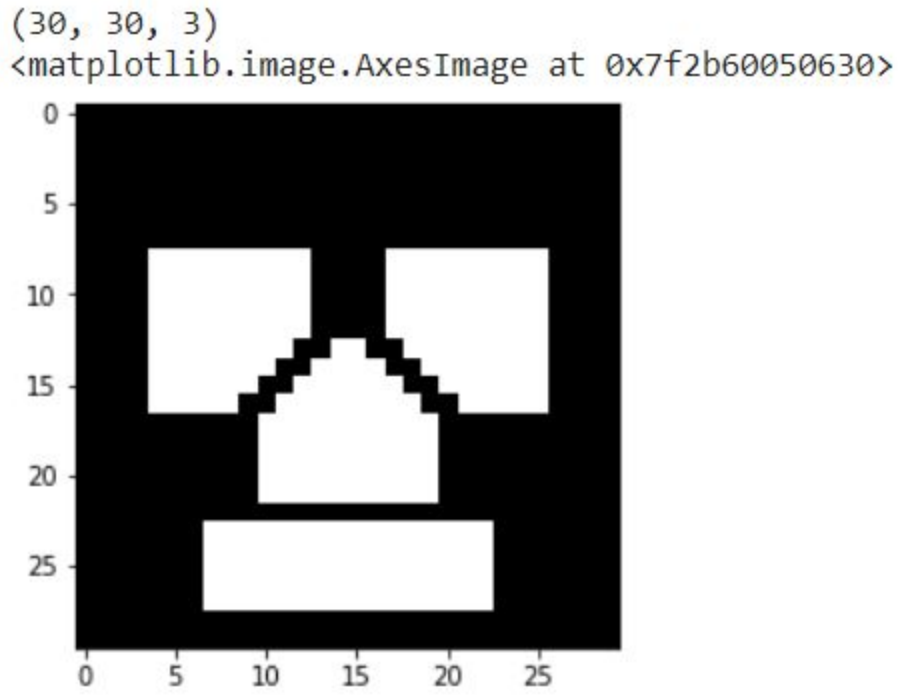
Detailed report of the approach

Model training

This section consists of three parts:

- Pre-processing of the Training Data

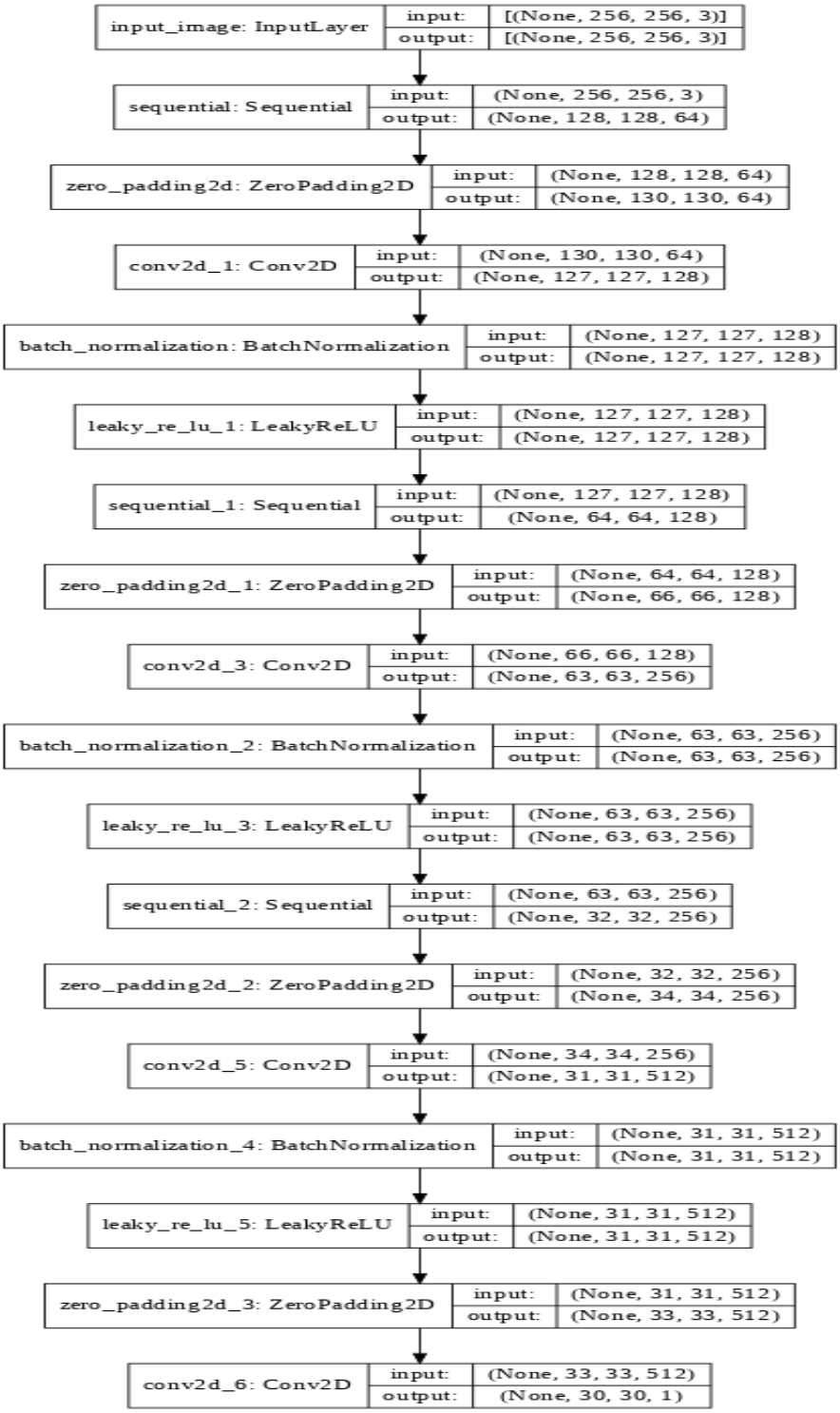
In this section I uploaded the data for training of the mode, which is based on binary classification of face into real and fake/generated. The source of the data set is from kaggle.
Link: https://www.kaggle.com/ciplab/real-and-fake-face-detection
In the first part I did the data pre-processing in which the main focus was to generate the fake and real images labels for training the first part of the model training pipeline. As in the first part we are training a PATCH-GAN based Discriminator model for mid pipeline processing. So for training this model we need to create 30x303x1 dimensional labels for the real images and fake images. The example of the created label is shown by a sample image below.



The white patches in the label represented the manipulated part in the fake image which corresponds to the eyes, nose and lips. The corresponding manipulated part is labeled in the image name itself in the dataset downloaded.
Once the labels for the training of the PATCH-GAN based model are created we make the whole images and its labels dataset and separate them into training and validation dataset respectively.

- Patch-GAN Based Discriminator Model

The first part of the pipeline for the training of classification model consists of making a PATCH-GAN based discriminator model whose information is given below.



Then the optimizer and loss functions are specified with some callbacks and then trained.

Once sensible results are obtained we move to the next part of the pipeline.

- Binary Classification Model

In this part of the pipeline we load the previously trained bast model and then make a new model from it by adding the flatten and desne layers at the last. The last layer consists of a single neuron which classifies the images into real and fake images. The summary of the final model is given below:

```
Model: "model_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_image (InputLayer)     [(None, 256, 256, 3)]     0
_________________________________________________________________
sequential (Sequential)      (None, 128, 128, 64)      3072
_________________________________________________________________
zero_padding2d (ZeroPadding2 (None, 130, 130, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 127, 127, 128)     131072
_________________________________________________________________
batch_normalization (BatchNo (None, 127, 127, 128)     512
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 127, 127, 128)     0
_________________________________________________________________
sequential_1 (Sequential)    (None, 64, 64, 128)       262656
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 66, 66, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 63, 63, 256)       524288
_________________________________________________________________
batch_normalization_2 (Batch (None, 63, 63, 256)       1024
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 63, 63, 256)       0
_________________________________________________________________
sequential_2 (Sequential)    (None, 32, 32, 256)       1049600
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 34, 34, 256)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 31, 31, 512)       2097152
_________________________________________________________________
batch_normalization_4 (Batch (None, 31, 31, 512)       2048
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 31, 31, 512)       0
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 33, 33, 512)       0
_________________________________________________________________
flatten_10 (Flatten)         (None, 557568)            0
_________________________________________________________________
fc2 (Dense)                  (None, 512)               285475328
_________________________________________________________________
fc3 (Dense)                  (None, 256)               131328
_________________________________________________________________
fc4 (Dense)                  (None, 128)               32896
_________________________________________________________________
output (Dense)               (None, 1)                 129
=================================================================
Total params: 289,711,105
Trainable params: 285,639,681
Non-trainable params: 4,071,424
_________________________________________________________________
```

Now, we specify the optimizer and the loss function along with callbacks and then train the model and then the best trained model is saved for final evaluation and classification process.

Classification of the provided images

The Classification pipeline for the provided images is also made in the same python notebook. In order to classify the images into real, fake and other objects I first decided to use computer vision library CV2 in python and use CascadeClassifier function and `haarcascade_frontalface_default.xml` file in it to detect faces and other objects and then i decided to use the trained model to classify the face images into real and fake images. The resultant classification is displayed in the end of the notebook and the csv file is saved by the last two lines of the code.

Methods which not generated good results

For this project I have tried two other methods earlier, but booth seemed to work a bit less accurately and loss effective than the current one. The two methods included the direct supervised learning method and other was a transfer learning method using various imagenet models.