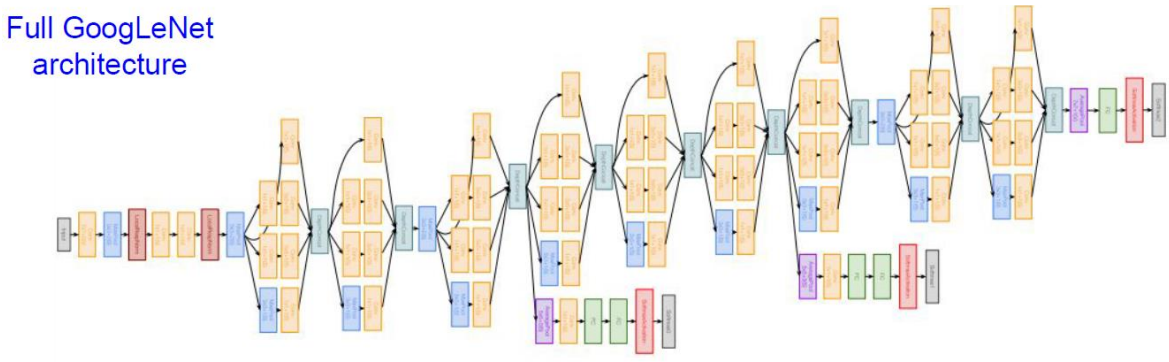
Classification of natural scene images via GoogLeNet InceptionV3 model

Abstract

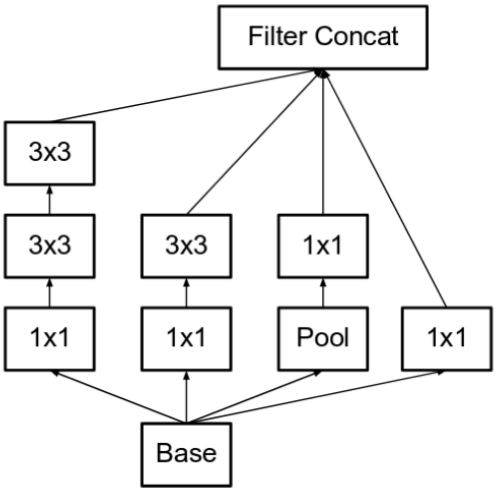
In this study classification of natural scene images is targeted using inceptionV3 model. The dataset used in for this purpose is natural scenes classification dataset available on Kaggle. Transfer learning is used for better and faster results in this study. The maximum accuracy that is achieved by the model is 52%.

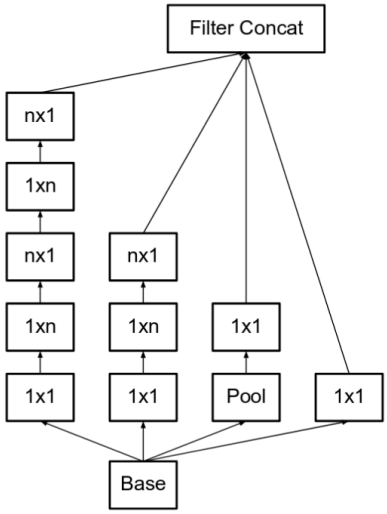
Introduction

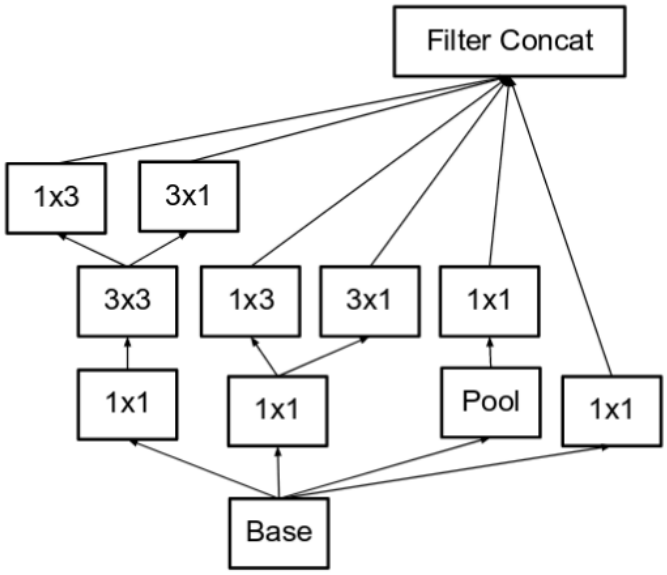
GoogLeNet is a CNN model for image classification and object detection and has 4 versions namely inception V1, V2, V3 and inception V4. These models are based on multiple small convolutions in order to minimize the number of parameters. The following image shows the architecture of GoogLeNet model.



The wide portions or blocks in this model are the key to it which are called inception module. In InceptionV2 and V3 there are three different types of inception modules shown below.







The concept behind these modules is the smaller a convolution is involved the lesser will be the number of parameters and multiple small convolutions were placed in place of on bigger convolution which gave the same result but had more parameters to calculate and hence needed more computation power and time.

Methodology

The model used is GoogLeNet InceptionV3 from keras library. The network diagram is shown in figure.

Training settings are in Figure 1

```
x = tf.keras.layers.Flatten()(last_output)
x = tf.keras.layers.Dense(units = 1024, activation = tf.nn.relu)(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense  (6, activation = tf.nn.softmax)(x)

model = tf.keras.Model( inceptionV3.input, x)

learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=1,
                                            verbose=1,
                                            factor=0.25,
                                            min_lr=0.000003)

model.compile(loss = 'categorical_crossentropy', optimizer= tf.keras.optimizers.Adam(), metrics=['acc'])

model.summary()
```

Figure 1

Setting for data augmentation is as follows

```
train_DIR = "/content/gdrive/MyDrive/Colab-Notebooks/Semester 3/CV/Asgt 3/Dataset/Manual Upload/seg_train/seg_train/"

train_datagen = ImageDataGenerator( shear_range=0.2,
                                zoom_range=0.2,
                                fill_mode="nearest",
                                horizontal_flip=True,
                                vertical_flip=True,)


train_generator = train_datagen.flow_from_directory(train_DIR,
                                                    batch_size=32,
                                                    class_mode='categorical',
                                                    target_size=(150, 150))

test_DIR = "/content/gdrive/MyDrive/Colab-Notebooks/Semester 3/CV/Asgt 3/Dataset/Manual Upload/seg_test/seg_test/"

validation_datagen = ImageDataGenerator(width_shift_range=0.0,
                                        height_shift_range=0.0)


validation_generator = validation_datagen.flow_from_directory(test_DIR,
                                                    batch_size=128,
                                                    class_mode='categorical',
                                                    target_size=(150, 150))
```

Figure 2

Results

10 epochs in total were executed, resulting in an accuracy of 52%

```
Epoch 10/10
439/439 [==============================] - 97s 222ms/step - loss: 1.1676 - acc: 0.5267 - val_loss: 1.1359 - val_acc: 0.5507
```

Following are the accuracy graphs for the above-mentioned setting in inceptionV3

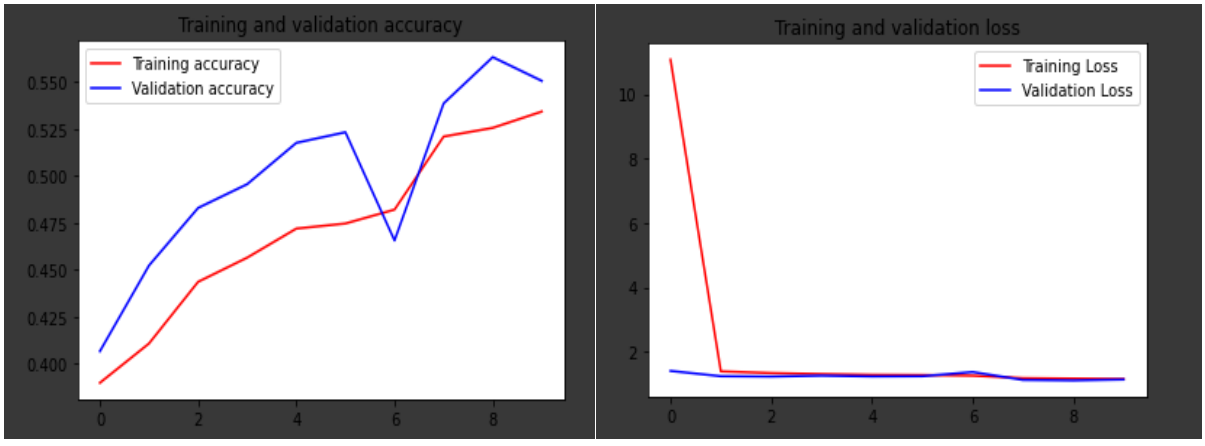


Figure 3

Figure 4 shows prediction run on an image

Figure 4

Discussion and Conclusion

52% is not a good accuracy considering that GoogLeNet InceptionV3 is a well-established model in image classification and object detection. Transfer learning and data augmentation are supposed to help in improving the result but that does not seem to be the case here, though it the algorithm needs to be run without transfer learning and data augmentation to confirm that. The result of an image shown is of the one which is correctly classified but there are many images that the model classified wrongly. Transfer learning can be excused as its main purpose is to use already generated weights and save time. Data augmentation's sole purpose is to train the algorithm better by generating extra data from the already existing data but that did not do any good.

One question that remains in my mind yet to be answered is that if transfer learning uses already generated weights from some other training and helps save time by removing the training part then how does data augmentation helps in that same code, as data augmentation is done to train better but transfer learning is done to avoid training.

GitHub Repository link

https://github.com/Yawarulhaq/CS867-CV-Assignment-3