====Method2 : Transfer Learning ====

```
In [ ]: from google.colab import drive
        drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
In [ ]: import os, shutil

        TRAINDIR = 'gdrive/MyDrive/ee_628/proj/train/'
        cat_folder = 'cat'
        dog_folder = 'dog'
```

```
In [ ]: from keras.preprocessing.image import ImageDataGenerator

        datagen = ImageDataGenerator(rotation_range = 10,
                                     shear_range=0.2,
                                     rescale = 1./255,
                                     validation_split=0.25)
        IMG_H = 150
        IMG_W = 150
```

```
In [ ]: train_generator = datagen.flow_from_directory(TRAINDIR,
                                                      target_size=(IMG_H, IMG_W),
                                                      batch_size=100,
                                                      class_mode='binary',
                                                      subset='training')
```

```
Found 18750 images belonging to 2 classes.
```

```
In [ ]: val_generator = datagen.flow_from_directory(TRAINDIR,
                                                    target_size=(IMG_H, IMG_W),
                                                    batch_size=100,
                                                    class_mode='binary',
                                                    subset='validation')
```

```
Found 6250 images belonging to 2 classes.
```

```
In [ ]: test_generator = datagen.flow_from_directory('gdrive/MyDrive/ee_628/proj/', classes=['test1'],
                                                     target_size=(IMG_H, IMG_W))
```

```
Found 12500 images belonging to 1 classes.
```

Transfer Learning:

VGG 16

```
In [ ]: import pandas as pd
        import numpy as np
        from keras.models import Sequential
        from keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense, Dropout, GlobalAveragePooling2D
        from keras.applications import VGG16, VGG19
```

```
In [ ]: vgg16_base = VGG16(include_top=False, weights='imagenet', input_shape=(IMG_H, IMG_H, 3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 0s 0us/step
```

```
In [ ]: vgg16_base.summary()
```

```
Model: "vgg16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 150, 150, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

As we see above, when using transfer learning, we are only concerned with the beginning or top convolutional layers from the network since it was trained on ImageNet (in this case) dataset. We will still have to add the fully connected layers to the end of the network for it to perform classification. In the layers that we add on now, we can see it will have to take an input of 4X4X512 or a single product of these three if we are working with a Flattening layer.

```
In [ ]: from keras.models import Sequential
        from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation, Dropout, BatchNormalization
```

```
In [ ]: model = Sequential()
        model.add(vgg16_base)
        model.add(Flatten())
        model.add(Dense(4096))
        model.add(Dense(4096))
        model.add(Dense(1000))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(loss='binary_crossentropy',
                      optimizer='adam',
                      metrics=['accuracy'])
```

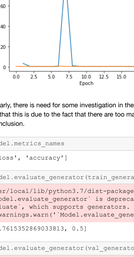
```
In [ ]: model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              33558528
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 1001
=================================================================
Total params: 69,152,529
Trainable params: 69,152,529
Non-trainable params: 0
_________________________________________________________________
```

```
In [ ]: hist = model.fit_generator(train_generator, validation_data=val_generator, epochs=20)
```

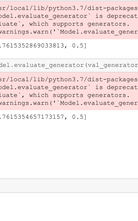
```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/20
188/188 [==============================] - 5948s 31s/step - loss: 12.3478 - accuracy: 0.5012 - val_loss: 0.6980 - val_accuracy: 0.5000
Epoch 2/20
188/188 [==============================] - 226s 1s/step - loss: 0.7010 - accuracy: 0.4950 - val_loss: 0.7013 - val_accuracy: 0.5000
Epoch 3/20
188/188 [==============================] - 226s 1s/step - loss: 0.6988 - accuracy: 0.5064 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 4/20
188/188 [==============================] - 227s 1s/step - loss: 0.7036 - accuracy: 0.4948 - val_loss: 0.6948 - val_accuracy: 0.5000
Epoch 5/20
188/188 [==============================] - 227s 1s/step - loss: 0.6963 - accuracy: 0.4969 - val_loss: 0.7013 - val_accuracy: 0.5000
Epoch 6/20
188/188 [==============================] - 225s 1s/step - loss: 0.6953 - accuracy: 0.4979 - val_loss: 0.7024 - val_accuracy: 0.5000
Epoch 7/20
188/188 [==============================] - 225s 1s/step - loss: 81.6620 - accuracy: 0.5023 - val_loss: 1.0683 - val_accuracy: 0.5000
Epoch 8/20
188/188 [==============================] - 224s 1s/step - loss: 2.1807 - accuracy: 0.5004 - val_loss: 0.7050 - val_accuracy: 0.5000
Epoch 9/20
188/188 [==============================] - 224s 1s/step - loss: 0.7853 - accuracy: 0.4933 - val_loss: 0.6932 - val_accuracy: 0.5000
Epoch 10/20
188/188 [==============================] - 225s 1s/step - loss: 0.7936 - accuracy: 0.4989 - val_loss: 0.7115 - val_accuracy: 0.5000
Epoch 11/20
188/188 [==============================] - 226s 1s/step - loss: 0.7731 - accuracy: 0.5058 - val_loss: 0.7147 - val_accuracy: 0.5000
Epoch 12/20
188/188 [==============================] - 224s 1s/step - loss: 0.7365 - accuracy: 0.5008 - val_loss: 0.6974 - val_accuracy: 0.5000
Epoch 13/20
188/188 [==============================] - 224s 1s/step - loss: 0.7737 - accuracy: 0.4960 - val_loss: 0.7077 - val_accuracy: 0.5000
Epoch 14/20
188/188 [==============================] - 224s 1s/step - loss: 0.7288 - accuracy: 0.5012 - val_loss: 0.6929 - val_accuracy: 0.5000
Epoch 15/20
188/188 [==============================] - 224s 1s/step - loss: 0.8107 - accuracy: 0.5006 - val_loss: 0.8154 - val_accuracy: 0.5000
Epoch 16/20
188/188 [==============================] - 224s 1s/step - loss: 0.7770 - accuracy: 0.5017 - val_loss: 0.7925 - val_accuracy: 0.5000
Epoch 17/20
188/188 [==============================] - 226s 1s/step - loss: 0.7447 - accuracy: 0.5047 - val_loss: 0.6966 - val_accuracy: 0.5000
Epoch 18/20
188/188 [==============================] - 226s 1s/step - loss: 0.7159 - accuracy: 0.5086 - val_loss: 0.6990 - val_accuracy: 0.5000
Epoch 19/20
188/188 [==============================] - 223s 1s/step - loss: 0.7202 - accuracy: 0.4986 - val_loss: 0.7528 - val_accuracy: 0.5000
Epoch 20/20
188/188 [==============================] - 222s 1s/step - loss: 0.7171 - accuracy: 0.4966 - val_loss: 0.7615 - val_accuracy: 0.5000
```

```
In [ ]: import matplotlib.pyplot as plt

        epochs = [i for i in range(1,21)]

        plt.plot(epochs, hist.history['accuracy'], hist.history['val_accuracy'])
        plt.legend(['Training', 'Validation'])
        plt.title('Accuracy Measure over Epochs - AlexNet')
        plt.xlabel('Epoch')
        plt.ylabel('Accuracy')
        plt.show()
```



We observe from the plots above that while training accuracy fluctuates quite a bit, the validation accuracy does not change at all and is completely constant for the entire duration of the 20 epochs.

```
In [ ]: plt.plot(epochs, hist.history['loss'], hist.history['val_loss'])
        plt.legend(['Training', 'Validation'])
        plt.title('Loss Measure over Epochs - AlexNet')
        plt.xlabel('Epoch')
        plt.ylabel('Loss')
        plt.show()
```



Clearly, there is need for some investigation in the training of this network as it seems to not be doing much on the validation set. It could be that this is due to the fact that there are too many dense layers at the end of the network. More about this has been mentioned in the Conclusion.

```
In [ ]: model.metrics_names
```

```
Out[ ]: ['loss', 'accuracy']
```

```
In [ ]: model.evaluate_generator(train_generator)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
```

```
Out[ ]: [0.7615352869033813, 0.5]
```

```
In [ ]: model.evaluate_generator(val_generator)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
```

```
Out[ ]: [0.7615354657173157, 0.5]
```

	Accuracy%	Loss
Training	50	0.7615
Validation	50	0.7615

```
In [ ]:
```