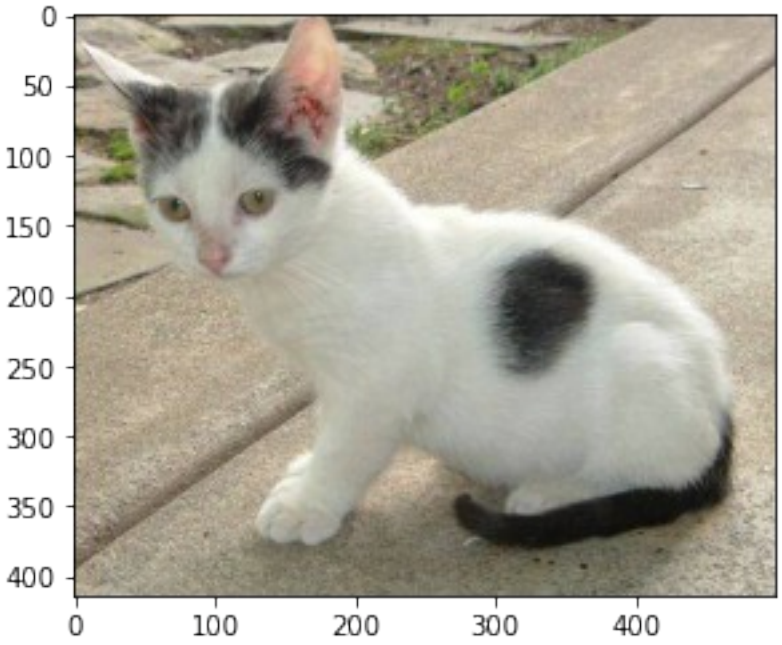
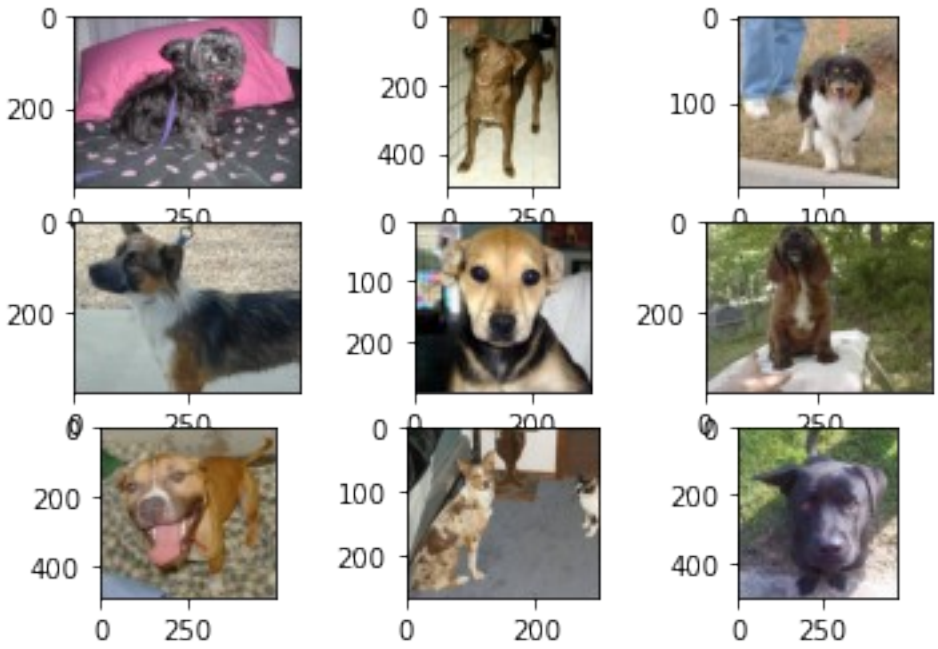
```
import os
import zipfile
import random
import datetime
from packaging import version
import numpy as np
import pandas as pd
import seaborn as sns
from IPython.display import Image, display
import matplotlib.pyplot as plt
from matplotlib import pyplot
from matplotlib.image import imread
from sklearn.metrics import recall_score, confusion_matrix,
precision_score, f1_score, accuracy_score, classification_report
from sklearn.metrics import roc_curve, auc

import sys
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten,
InputLayer, MaxPooling2D, Conv2D
from keras.utils import np_utils
from keras.utils.vis_utils import plot_model
from keras.callbacks import EarlyStopping
from keras.callbacks import History
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.models import Model

from os import makedirs
from os import listdir
from shutil import copyfile
from random import seed
from random import random
```

EDA

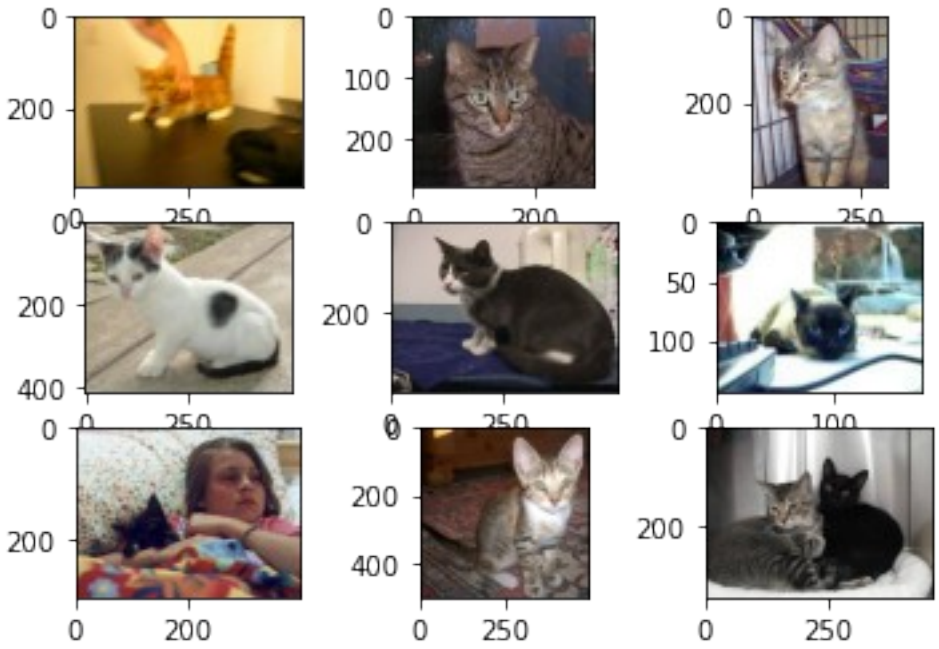
```
image = imread("train/cat.3.jpg")

plt.imshow(image)
plt.show()
```




```
# plot dog photos from the dataset
# define location of dataset
folder = 'train/'
# plot first few images
for i in range(9):
# define subplot
    pyplot.subplot(330 + 1 + i)
# define filename
    filename = folder + 'dog.' + str(i) + '.jpg'
# load image pixels
    image = imread(filename)
# plot raw pixel data
    pyplot.imshow(image)
# show the figure
pyplot.show()
```



```
# plot cat photos from the dogs vs cats dataset
# define location of dataset
folder = 'train/'
# plot first few images
for i in range(9):
# define subplot
    pyplot.subplot(330 + 1 + i)
# define filename
    filename = folder + 'cat.' + str(i) + '.jpg'
# load image pixels
    image = imread(filename)
# plot raw pixel data
    pyplot.imshow(image)
# show the figure
pyplot.show()
```



```
# create directories
dataset_home = 'dataset_dogs_vs_cats/'
subdirs = ['train/', 'test/']
for subdir in subdirs:
# create label subdirectories
    labeldirs = ['dogs/', 'cats/']
    for labldir in labeldirs:
        newdir = dataset_home + subdir + labldir
        makedirs(newdir, exist_ok=True)

# seed random number generator
seed(42)
# define ratio of pictures to use for validation
val_ratio = 0.25
# copy training dataset images into subdirectories
src_directory = 'train/'
for file in listdir(src_directory):
    src = src_directory + '/' + file
    dst_dir = 'train/'
    if random() < val_ratio:
        dst_dir = 'test/'
    if file.startswith('cat'):
        dst = dataset_home + dst_dir + 'cats/'  + file
        copyfile(src, dst)
    elif file.startswith('dog'):
        dst = dataset_home + dst_dir + 'dogs/'  + file
        copyfile(src, dst)
```

Modeling

```
# define cnn model
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(200,
200, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))
    model.add(Dense(1, activation='sigmoid'))
# compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='binary_crossentropy',
metrics=['accuracy'])
    return model

def summarize_diagnostics(history):
    # plot loss
    pyplot.subplot(211)
    pyplot.title('Cross Entropy Loss')
    pyplot.plot(history.history['loss'], color='blue', label='train')
    pyplot.plot(history.history['val_loss'], color='orange',
label='test')
# plot accuracy
    pyplot.subplot(212)
    pyplot.title('Classification Accuracy')
    pyplot.plot(history.history['accuracy'], color='blue',
label='train')
    pyplot.plot(history.history['val_accuracy'], color='orange',
label='test')
    pyplot.tight_layout(pad=3)
    pyplot.show()

def run_test_harness():
    # define model
    model = define_model()
    # create data generator
    datagen = ImageDataGenerator(rescale=1.0/255.0)
    # prepare iterators
    train_it =
datagen.flow_from_directory('dataset_dogs_vs_cats/train/',
        class_mode='binary', batch_size=64, target_size=(200, 200))
    test_it =
datagen.flow_from_directory('dataset_dogs_vs_cats/test/',
        class_mode='binary', batch_size=64, target_size=(200, 200))
    # fit model
    history = model.fit(train_it, steps_per_epoch=len(train_it),
        validation_data=test_it, validation_steps=len(test_it),
```

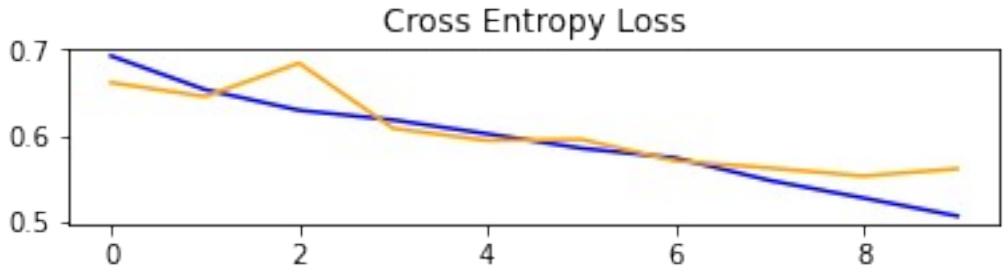
```
epochs=10, verbose=1)
    # evaluate model
    _, acc = model.evaluate_generator(test_it, steps=len(test_it),
verbose=1)
    print('> %.3f' % (acc * 100.0))
    # learning curves
    summarize_diagnostics(history)
    return history, model
```

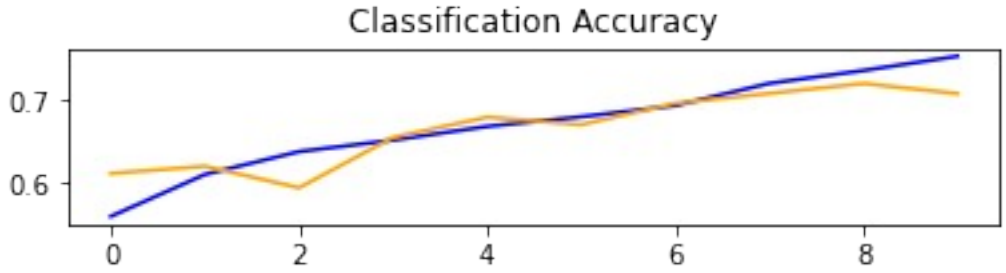
VGG (One Block)

```
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(200,
200, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))
    model.add(Dense(1, activation='sigmoid'))
# compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='binary_crossentropy',
metrics=['accuracy'])
    return model

model_1_history, model_1 = run_test_harness()
```

```
Found 18722 images belonging to 2 classes.
Found 6278 images belonging to 2 classes.
Epoch 1/10
293/293 [==============================] - 390s 1s/step - loss: 0.6925
- accuracy: 0.5591 - val_loss: 0.6614 - val_accuracy: 0.6112
Epoch 2/10
293/293 [==============================] - 242s 826ms/step - loss:
0.6531 - accuracy: 0.6100 - val_loss: 0.6451 - val_accuracy: 0.6199
Epoch 3/10
293/293 [==============================] - 237s 809ms/step - loss:
0.6291 - accuracy: 0.6378 - val_loss: 0.6841 - val_accuracy: 0.5938
Epoch 4/10
293/293 [==============================] - 237s 808ms/step - loss:
0.6178 - accuracy: 0.6516 - val_loss: 0.6080 - val_accuracy: 0.6550
Epoch 5/10
293/293 [==============================] - 239s 815ms/step - loss:
0.6016 - accuracy: 0.6683 - val_loss: 0.5938 - val_accuracy: 0.6795
Epoch 6/10
293/293 [==============================] - 245s 837ms/step - loss:
0.5847 - accuracy: 0.6800 - val_loss: 0.5951 - val_accuracy: 0.6704
Epoch 7/10
```

```
293/293 [==============================] - 311s 1s/step - loss: 0.5735
- accuracy: 0.6936 - val_loss: 0.5701 - val_accuracy: 0.6964
Epoch 8/10
293/293 [==============================] - 347s 1s/step - loss: 0.5473
- accuracy: 0.7206 - val_loss: 0.5617 - val_accuracy: 0.7083
Epoch 9/10
293/293 [==============================] - 333s 1s/step - loss: 0.5268
- accuracy: 0.7362 - val_loss: 0.5524 - val_accuracy: 0.7205
Epoch 10/10
293/293 [==============================] - 335s 1s/step - loss: 0.5058
- accuracy: 0.7536 - val_loss: 0.5609 - val_accuracy: 0.7082
WARNING:tensorflow:From <ipython-input-14-0c272ff817d5>:15:
Model.evaluate_generator (from
tensorflow.python.keras.engine.training) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use Model.evaluate, which supports generators.
99/99 [==============================] - 47s 474ms/step - loss: 0.5609
- accuracy: 0.7082
> 70.819
```






```
datagen = ImageDataGenerator(rescale=1.0/255.0)
val_test = datagen.flow_from_directory('dataset_dogs_vs_cats/test/',
        class_mode='binary', batch_size=64, target_size=(200, 200))

Found 6278 images belonging to 2 classes.

y_predict = model_1.predict_generator(val_test)

WARNING:tensorflow:From <ipython-input-18-11cb9a9406e1>:1:
Model.predict_generator (from tensorflow.python.keras.engine.training)
```

```
is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.predict, which supports generators.
```

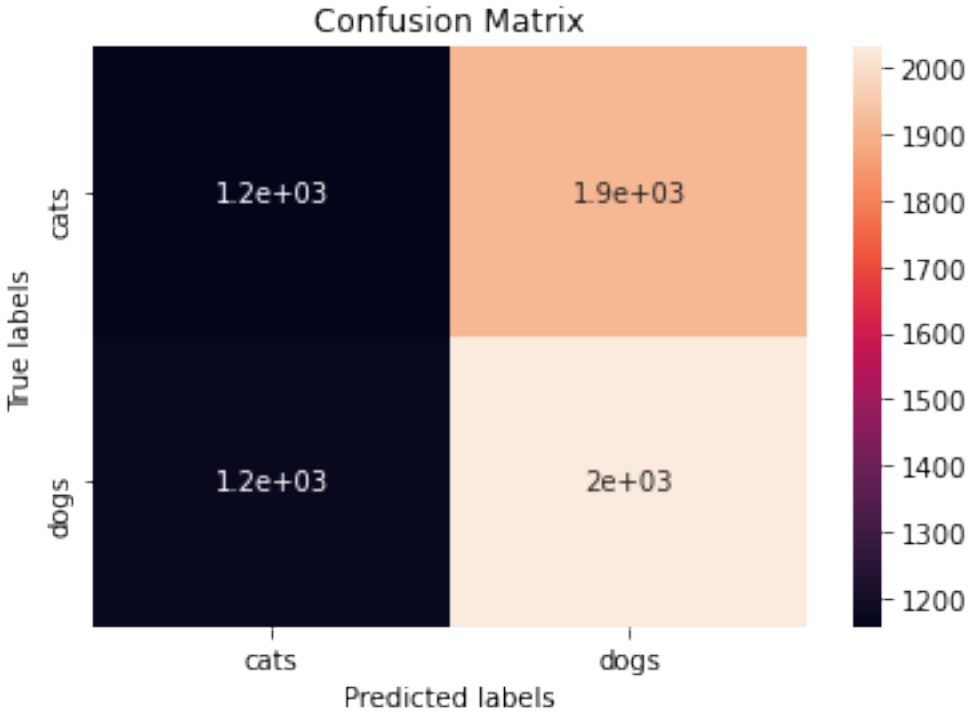
```
y_predict = np.where(y_predict > 0.5, 1, 0)
```

```
p = val_test.classes
q = y_predict
p = np.array(p)
q = q.flatten()
```

```
cfm = confusion_matrix(p, q)
print(cfm)
ax= plt.subplot()
sns.heatmap(cfm, annot=True, ax = ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['cats', 'dogs'])
ax.yaxis.set_ticklabels(['cats', 'dogs'])
```

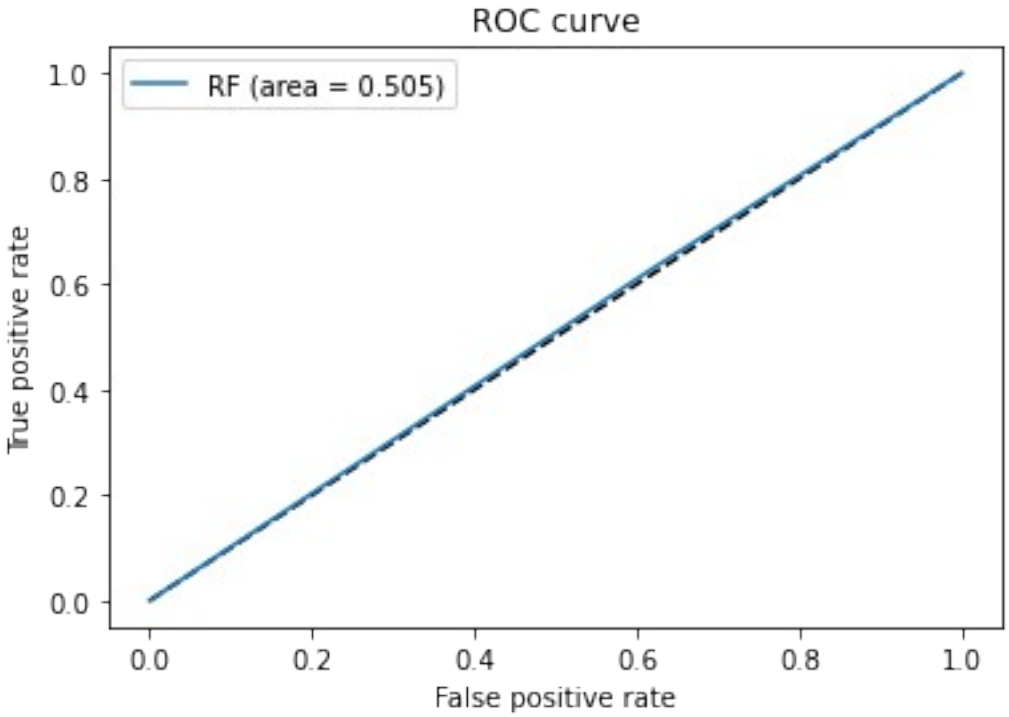
```
[[1157 1914]
 [1176 2031]]
```

```
[Text(0, 0.5, 'cats'), Text(0, 1.5, 'dogs')]
```




```
fpr, tpr, threshold = roc_curve(p,q)
auc_rf = auc(fpr, tpr)
```

```
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='RF (area = {:.3f})'.format(auc_rf))
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```

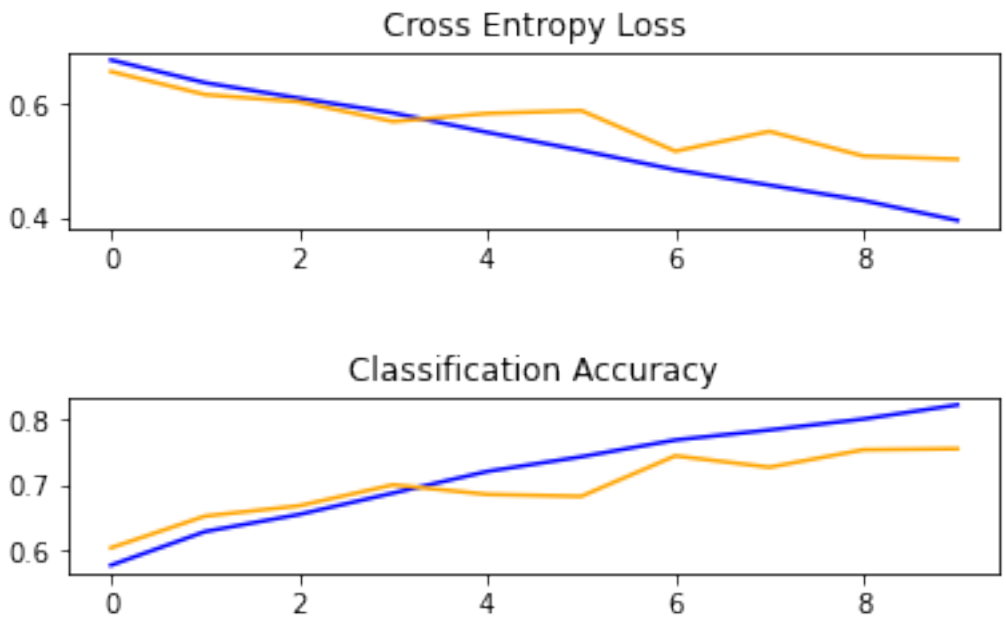


VGG (Two Block)

```
# define cnn model
def define_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(200,
200, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))
    model.add(Dense(1, activation='sigmoid'))
# compile model
    opt = SGD(lr=0.001, momentum=0.9)
```

```
    model.compile(optimizer=opt, loss='binary_crossentropy',
metrics=['accuracy'])
    return model

model_2_history, model_2 = run_test_harness()
```

```
Found 18722 images belonging to 2 classes.
Found 6278 images belonging to 2 classes.
Epoch 1/10
293/293 [==============================] - 785s 3s/step - loss: 0.6771
- accuracy: 0.5770 - val_loss: 0.6569 - val_accuracy: 0.6037
Epoch 2/10
293/293 [==============================] - 398s 1s/step - loss: 0.6372
- accuracy: 0.6286 - val_loss: 0.6164 - val_accuracy: 0.6523
Epoch 3/10
293/293 [==============================] - 475s 2s/step - loss: 0.6101
- accuracy: 0.6548 - val_loss: 0.6037 - val_accuracy: 0.6680
Epoch 4/10
293/293 [==============================] - 435s 1s/step - loss: 0.5841
- accuracy: 0.6876 - val_loss: 0.5690 - val_accuracy: 0.7002
Epoch 5/10
293/293 [==============================] - 445s 2s/step - loss: 0.5502
- accuracy: 0.7207 - val_loss: 0.5832 - val_accuracy: 0.6856
Epoch 6/10
293/293 [==============================] - 427s 1s/step - loss: 0.5183
- accuracy: 0.7432 - val_loss: 0.5883 - val_accuracy: 0.6825
Epoch 7/10
293/293 [==============================] - 426s 1s/step - loss: 0.4844
- accuracy: 0.7688 - val_loss: 0.5170 - val_accuracy: 0.7447
Epoch 8/10
293/293 [==============================] - 428s 1s/step - loss: 0.4574
- accuracy: 0.7842 - val_loss: 0.5518 - val_accuracy: 0.7271
Epoch 9/10
293/293 [==============================] - 428s 1s/step - loss: 0.4304
- accuracy: 0.8010 - val_loss: 0.5086 - val_accuracy: 0.7539
Epoch 10/10
293/293 [==============================] - 437s 1s/step - loss: 0.3956
- accuracy: 0.8227 - val_loss: 0.5029 - val_accuracy: 0.7555
99/99 [==============================] - 47s 471ms/step - loss: 0.5029
- accuracy: 0.7555
> 75.550
```

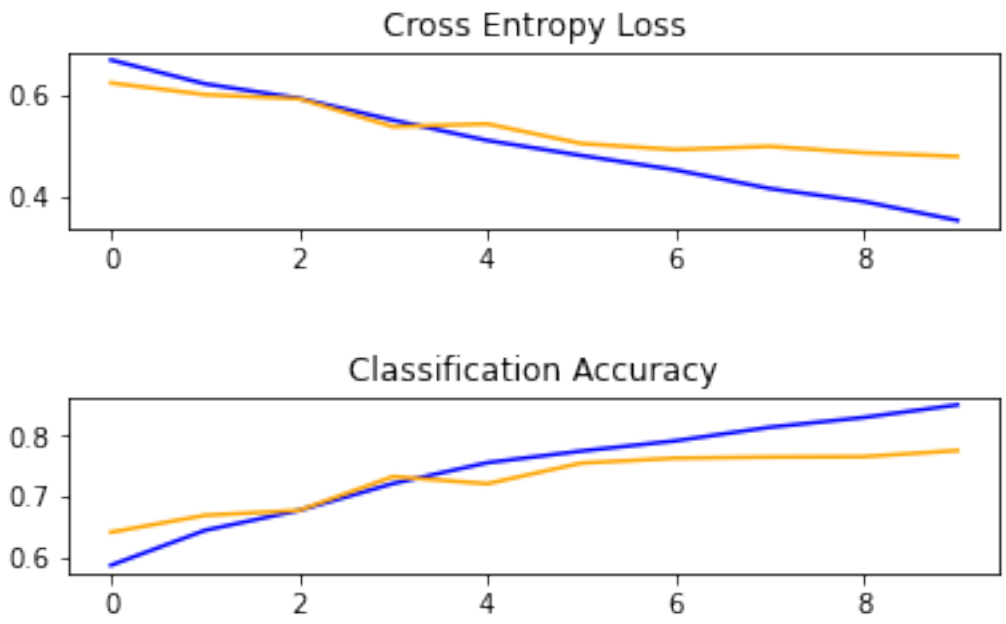


VGG (Three Block)

```
# define cnn model
def define_model_3():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same', input_shape=(200,
200, 3)))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(64, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Conv2D(128, (3, 3), activation='relu',
kernel_initializer='he_uniform', padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu',
kernel_initializer='he_uniform'))
    model.add(Dense(1, activation='sigmoid'))
# compile model
    opt = SGD(lr=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss='binary_crossentropy',
metrics=['accuracy'])
    return model

model_3_histroy, model_3 = run_test_harness()
```

```
Found 18722 images belonging to 2 classes.
Found 6278 images belonging to 2 classes.
Epoch 1/10
```

```
293/293 [==============================] - 508s 2s/step - loss: 0.6683
- accuracy: 0.5868 - val_loss: 0.6229 - val_accuracy: 0.6410
Epoch 2/10
293/293 [==============================] - 386s 1s/step - loss: 0.6211
- accuracy: 0.6441 - val_loss: 0.5998 - val_accuracy: 0.6687
Epoch 3/10
293/293 [==============================] - 385s 1s/step - loss: 0.5925
- accuracy: 0.6767 - val_loss: 0.5916 - val_accuracy: 0.6768
Epoch 4/10
293/293 [==============================] - 395s 1s/step - loss: 0.5493
- accuracy: 0.7208 - val_loss: 0.5364 - val_accuracy: 0.7322
Epoch 5/10
293/293 [==============================] - 383s 1s/step - loss: 0.5092
- accuracy: 0.7549 - val_loss: 0.5419 - val_accuracy: 0.7205
Epoch 6/10
293/293 [==============================] - 385s 1s/step - loss: 0.4794
- accuracy: 0.7741 - val_loss: 0.5031 - val_accuracy: 0.7544
Epoch 7/10
293/293 [==============================] - 384s 1s/step - loss: 0.4510
- accuracy: 0.7908 - val_loss: 0.4909 - val_accuracy: 0.7623
Epoch 8/10
293/293 [==============================] - 384s 1s/step - loss: 0.4142
- accuracy: 0.8130 - val_loss: 0.4974 - val_accuracy: 0.7643
Epoch 9/10
293/293 [==============================] - 384s 1s/step - loss: 0.3886
- accuracy: 0.8291 - val_loss: 0.4849 - val_accuracy: 0.7649
Epoch 10/10
293/293 [==============================] - 386s 1s/step - loss: 0.3511
- accuracy: 0.8498 - val_loss: 0.4776 - val_accuracy: 0.7749
99/99 [==============================] - 31s 315ms/step - loss: 0.4776
- accuracy: 0.7749
> 77.493
```



VGG 1: 70.82% VGG 2: 75.55% VGG 3: 77.49%

```
# create submission
datagen = ImageDataGenerator(rescale=1.0/255.0)
test = 'test/'
test = datagen.flow_from_directory('.', classes=['test'],
batch_size=64, target_size=(200, 200))

Found 12500 images belonging to 1 classes.

preds3 = model_3.predict(x=test, steps=len(test), verbose=0)

predict_3 = pd.DataFrame(preds3)
predict_3.index.rename("id", inplace=True)
predict_3.index += 1
predict_3.rename(columns={0: 'label'}, inplace=True)

predict_3.head()

       label
id
1   0.221931
2   0.176223
3   0.749953
4   0.174482
5   0.137836

predict_3.to_csv('VGG3.csv')

from IPython.display import Image
Image(filename='VGG3.png')
```

YOUR RECENT SUBMISSION

VGG3.csv

Submitted by SeafoodTakeout · Submitted just now

Score: 1.18179

Conclusion

The Models ran fairly well. The three block VGG performed the best of the three models and does seem to have quite a bit of loss, but its performance is still adequate for the task.