```
# ===== 0.1 Check for GPU
!nvidia-smi

import tensorflow as tf
from tensorflow.python.client import device_lib

print("Tensorflow Version:", tf.__version__)
print("Is Build with CUDA:", tf.test.is_built_with_cuda())
print("Is GPU available:", tf.test.is_gpu_available(cuda_only=False, min_cuda_compute_capability=None))
print(device_lib.list_local_devices())
```

Saved successfully!

```
Sat Feb 29 23:13:44 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.48.02    Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   36C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tensorflow_vers

```
Tensorflow Version: 1.15.0
Is Build with CUDA: True
Is GPU available: True
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 15796043826267813210
, name: "/device:XLA_CPU:0"
```

Saved successfully!

```
}
incarnation: 8679713831415167039
physical_device_desc: "device: XLA_CPU device"
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 3867230563521121667
physical_device_desc: "device: XLA_GPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 15956161332
locality {
  bus_id: 1
  links {
  }
}
incarnation: 8529673557741089943
physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:04.0, compute ca
]
```

```
# ===== 0.2 Utils and Consts
import time
import psutil
import pandas as pd
import pandas as pd
import numpy as np
import cv2
import base64
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.applications import vgg16, vgg19, resnet50
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, GlobalAveragePooling
from tensorflow.keras.layers import BatchNormalization, Activation, Input, AveragePooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateScheduler, ReduceLRO
from tensorflow.keras.regularizers import l2

PATH_ROOT = 'drive/My Drive/Colab Notebooks'
PATH_FILE = PATH_ROOT + '/Datasets/ChartImages/charts_1_day_50_periods.csv'
MODEL_NAME_VGG16 = PATH_ROOT + '/Models/ChartImages/model_vgg16_1_day_50_periods.h5'
MAX_RGB = 255
IMAGE_DIMENSION = 224
IMAGE_INPUT_SHAPE = (IMAGE_DIMENSION, IMAGE_DIMENSION, 3)
LABEL_CLASS = { 'Down': 0, 'Up': 1 }
NUM_CLASSES = 2
NUM_BATCH_SIZE_VGG16 = 128
NUM_EPOCHS = 100

# Utils
```

Saved successfully!

```
    global start_time
    start_time = time.time()
def watch_print(title):
    global start_time
    print(title,round(time.time() - start_time, 4), 'seconds')
def memory_print():
    memory = dict(psutil.virtual_memory()._asdict())
    print("Memory Capacity", memory['total'] >> 30, "GB")
    print("Memory Left", memory['free'] >> 30, "GB")
    print("Memory Used", memory['used'] >> 30, "GB")
    print("Memory Used:", memory['percent'], "percent")

# Callbacks for model saving and stopping.
# Training should be stopped when val_acc (validation accuracy) clearly stops increasing to prevent ove
checkpoint_vgg16 = ModelCheckpoint(filepath=MODEL_NAME_VGG16, monitor='val_acc', verbose=1, save_best_o
early_stopping = EarlyStopping(monitor='val_acc', patience=10, verbose=1)


watch_restart()

# ===== 1.0 Get Image Data from CSV
df = pd.read_csv(PATH_FILE)
print(df.head())

memory_print()
watch_print('Get Data')
```

```
   Id Symbol  ... Y_Prediction                                            X_Image
0   1      M  ...            1  iVBORw0KGgoAAAANSUhEUgAAAOAAAADgCAYAAAAaLWrhAA...
1   2      M  ...            1  iVBORw0KGgoAAAANSUhEUgAAAOAAAADgCAYAAAAaLWrhAA...
2   3      M  ...            1  iVBORw0KGgoAAAANSUhEUgAAAOAAAADgCAYAAAAaLWrhAA...
3   4      M  ...            1  iVBORw0KGgoAAAANSUhEUgAAAOAAAADgCAYAAAAaLWrhAA...
4   5      M  ...            1  iVBORw0KGgoAAAANSUhEUgAAAOAAAADgCAYAAAAaLWrhAA...

[5 rows x 5 columns]
Memory Capacity 25 GB
Memory Left 22 GB
Memory Used 1 GB
Memory Used: 4.7 percent
Get Data 4.5841 seconds
```

```
watch_restart()

# ===== 2.0 Prepare Data

# Set up X and y
items = []
for index, row in df.iterrows():
    # Convert from base64 string to byte array
    item_byte_array = base64.b64decode(df['X_Image'][index])

    # Convert byte array to numpy array for OpenCv usage
    item_np = np.frombuffer(item_byte_array, dtype=np.uint8)

                                             image
                                            np, flags=1)

    items.append(item_image)
    # if index < 1:
        # plt.imshow(cv2.cvtColor(item_image, cv2.COLOR_BGR2RGB));
        # plt.show()

X = np.array(items)
y = to_categorical(df[['Y_Prediction']].values)
df = None # Clear RAM

print('y type', type(y))
print('y shape', y.shape)
print('X type', type(X))
print('X shape', X.shape)

# Normalize input data.
# Neural Networks work best when input data are between 0 and 1 (Instead of 0 to 255).
X = X / MAX_RGB

# Split Train and Test
def split(X, y, proportion):
    ratio = int(X.shape[0]/proportion)
    X_train = X[ratio:,:]
    X_test =  X[:ratio,:]
    y_train = y[ratio:,:]
    y_test =  y[:ratio,:]
    return X_train, X_test, y_train, y_test
# X_train, X_test, y_train, y_test = split(X, y, 4) # Uses less RAM
```

Saved successfully!

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)  # Uses a lot of RAM
X = y = None # Clear RAM

print("X_train Shape", X_train.shape)
print("y_train Shape", y_train.shape)
print("X_test Shape", X_test.shape)
print("y_yest Shape", y_test.shape)

memory_print()
watch_print('Prepare Data')
```

```
y type <class 'numpy.ndarray'>
y shape (9738, 2)
X type <class 'numpy.ndarray'>
X shape (9738, 224, 224, 3)
X_train Shape (7303, 224, 224, 3)
y_train Shape (7303, 2)
X_test Shape (2435, 224, 224, 3)
y_yest Shape (2435, 2)
Memory Capacity 25 GB
Memory Left 0 GB
Memory Used 24 GB
Memory Used: 52.2 percent
Prepare Data 17.8596 seconds
```

```
watch_restart()

# ----- 3.0 Create Model VGG16
                                     m_classes):

Saved successfully!

    # Remove top since different number of output classes than the pretrained model.
    model_pretrained = vgg16.VGG16(include_top=False, input_shape=input_shape)

    # Freeze layers since they have already been pretrained.
    for layer in model_pretrained.layers:
        layer.trainable = False
        ret.add(layer)

    # Flatten
    # ret.add(Flatten())

    ret.add(Conv2D(512, kernel_size = (3,3), padding = 'valid'))
    ret.add(GlobalAveragePooling2D())

    # 4 classes
    ret.add(Dense(num_classes, activation='softmax'))

    return ret

opt = SGD(lr=0.001)
model_vgg16 = get_model_vgg16(IMAGE_INPUT_SHAPE, NUM_CLASSES)
model_vgg16.compile(loss='categorical_crossentropy',
                    #optimizer=opt,
                    optimizer='adam',
                    metrics=['accuracy'])
model_vgg16.summary()
```

```
checkpoint_vgg16 = ModelCheckpoint(filepath=MODEL_NAME_VGG16, monitor='val_acc', verbose=1, save_best_o
history_vgg16 = model_vgg16.fit(X_train,
                                y_train,
                                batch_size=NUM_BATCH_SIZE_VGG16,
                                epochs=NUM_EPOCHS,
                                validation_data=(X_test, y_test),
                                shuffle=True,
                                callbacks=[checkpoint_vgg16, early_stopping])

memory_print()
watch_print('Create Model VGG16')
```

Saved successfully!

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python/ops/resource_
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.1/vgg1
58892288/58889256 [==============================] - 5s 0us/step
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160   
_________________________________________________________________
   Saved successfully!       None, 28, 28, 512)        2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0         
_________________________________________________________________
conv2d (Conv2D)              (None, 5, 5, 512)         2359808   
_________________________________________________________________
global_average_pooling2d (Gl (None, 512)               0         
_________________________________________________________________
dense (Dense)                (None, 2)                 1026      
=================================================================
Total params: 17,075,522
Trainable params: 2,360,834
Non-trainable params: 14,714,688
_________________________________________________________________
Train on 7303 samples, validate on 2435 samples
Epoch 1/100
7296/7303 [============================>.] - ETA: 0s - loss: 2.0440 - acc: 0.5525
Epoch 00001: val_acc improved from -inf to 0.68665, saving model to drive/My Drive/Colab Notebooks
7303/7303 [==============================] - 42s 6ms/sample - loss: 2.0428 - acc: 0.5525 - val_los
Epoch 2/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.6038 - acc: 0.6859
Epoch 00002: val_acc did not improve from 0.68665
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.6035 - acc: 0.6860 - val_los
```

```
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.6055 - acc: 0.6860 - val_los
Epoch 3/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5807 - acc: 0.7076
Epoch 00003: val_acc did not improve from 0.68665
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.5805 - acc: 0.7078 - val_los
Epoch 4/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5660 - acc: 0.7196
Epoch 00004: val_acc did not improve from 0.68665
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.5662 - acc: 0.7194 - val_los
Epoch 5/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5484 - acc: 0.7392
Epoch 00005: val_acc did not improve from 0.68665
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.5484 - acc: 0.7391 - val_los
Epoch 6/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5631 - acc: 0.7348
Epoch 00006: val_acc improved from 0.68665 to 0.72936, saving model to drive/My Drive/Colab Notebo
7303/7303 [==============================] - 32s 4ms/sample - loss: 0.5633 - acc: 0.7348 - val_los
Epoch 7/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5649 - acc: 0.7235
Epoch 00007: val_acc did not improve from 0.72936
7303/7303 [==============================] - 28s 4ms/sample - loss: 0.5646 - acc: 0.7237 - val_los
Epoch 8/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5602 - acc: 0.7288
Epoch 00008: val_acc improved from 0.72936 to 0.78973, saving model to drive/My Drive/Colab Notebo
7303/7303 [==============================] - 33s 4ms/sample - loss: 0.5599 - acc: 0.7290 - val_los
Epoch 9/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5043 - acc: 0.7812
Epoch 00009: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.5043 - acc: 0.7812 - val_los
Epoch 10/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5000 - acc: 0.7808
                             mprove from 0.78973
   Saved successfully!       ===========] - 27s 4ms/sample - loss: 0.5001 - acc: 0.7808 - val_los

7296/7303 [============================>.] - ETA: 0s - loss: 0.5169 - acc: 0.7701
Epoch 00011: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.5169 - acc: 0.7701 - val_los
Epoch 12/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4922 - acc: 0.7896
Epoch 00012: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4922 - acc: 0.7897 - val_los
Epoch 13/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4917 - acc: 0.7876
Epoch 00013: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4916 - acc: 0.7876 - val_los
Epoch 14/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5136 - acc: 0.7681
Epoch 00014: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.5138 - acc: 0.7680 - val_los
Epoch 15/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4758 - acc: 0.8024
Epoch 00015: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4757 - acc: 0.8023 - val_los
Epoch 16/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5203 - acc: 0.7566
Epoch 00016: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.5203 - acc: 0.7567 - val_los
Epoch 17/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4743 - acc: 0.8036
Epoch 00017: val_acc did not improve from 0.78973
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4742 - acc: 0.8035 - val_los
Epoch 18/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4586 - acc: 0.8113
Epoch 00018: val_acc improved from 0.78973 to 0.79548, saving model to drive/My Drive/Colab Notebo
7303/7303 [==============================] - 32s 4ms/sample - loss: 0.4587 - acc: 0.8113 - val_los
```

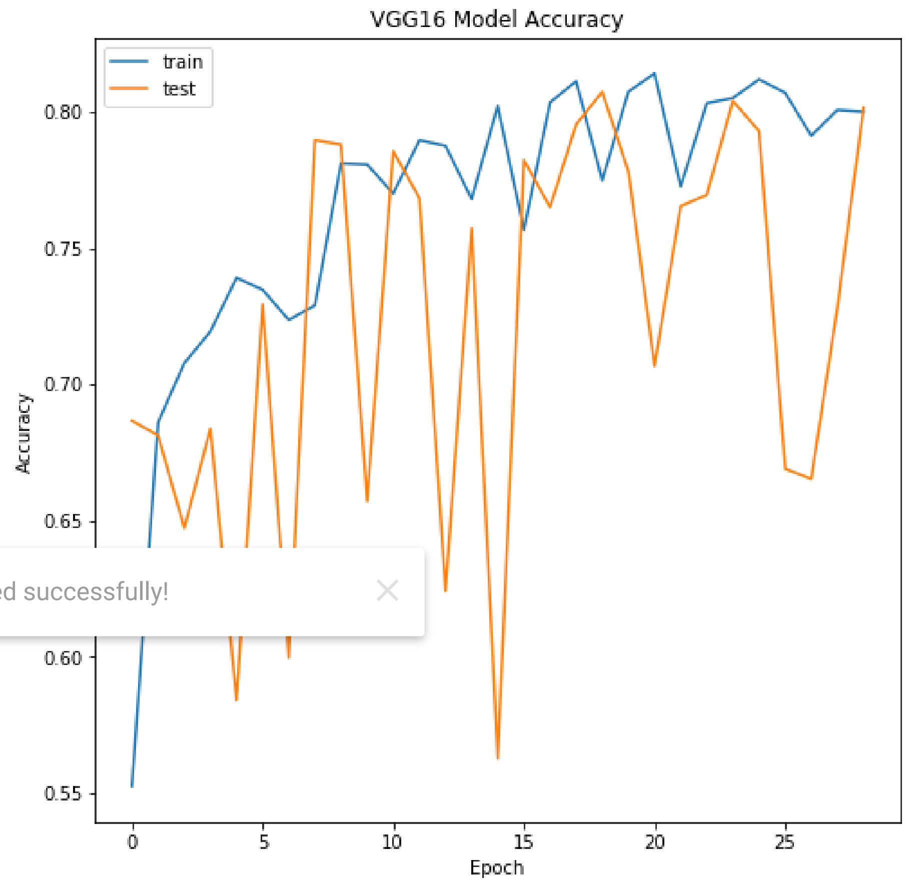
```
7303/7303 [==============================] - 32s 4ms/sample - loss: 0.4587 - acc: 0.8113 - val_los
Epoch 19/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5043 - acc: 0.7749
Epoch 00019: val_acc improved from 0.79548 to 0.80739, saving model to drive/My Drive/Colab Notebo
7303/7303 [==============================] - 32s 4ms/sample - loss: 0.5045 - acc: 0.7749 - val_los
Epoch 20/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4674 - acc: 0.8076
Epoch 00020: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4676 - acc: 0.8075 - val_los
Epoch 21/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4538 - acc: 0.8141
Epoch 00021: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4538 - acc: 0.8142 - val_los
Epoch 22/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5011 - acc: 0.7726
Epoch 00022: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.5010 - acc: 0.7727 - val_los
Epoch 23/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4680 - acc: 0.8035
Epoch 00023: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4681 - acc: 0.8032 - val_los
Epoch 24/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4619 - acc: 0.8050
Epoch 00024: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4617 - acc: 0.8051 - val_los
Epoch 25/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4526 - acc: 0.8120
Epoch 00025: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4524 - acc: 0.8120 - val_los
Epoch 26/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4571 - acc: 0.8072
                               mprove from 0.80739
    Saved successfully!        ===========] - 27s 4ms/sample - loss: 0.4571 - acc: 0.8071 - val_los

7296/7303 [============================>.] - ETA: 0s - loss: 0.4829 - acc: 0.7917
Epoch 00027: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4831 - acc: 0.7913 - val_los
Epoch 28/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4669 - acc: 0.8007
Epoch 00028: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4668 - acc: 0.8008 - val_los
Epoch 29/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4658 - acc: 0.8000
Epoch 00029: val_acc did not improve from 0.80739
7303/7303 [==============================] - 27s 4ms/sample - loss: 0.4657 - acc: 0.8001 - val_los
Epoch 00029: early stopping
Memory Capacity 25 GB
Memory Left 0 GB
Memory Used 22 GB
Memory Used: 59.4 percent
Create Model VGG16 835.5362 seconds
```

```
watch_restart()

# ===== 4. Evaluate Models
best_model_vgg16 = get_model_vgg16(IMAGE_INPUT_SHAPE, NUM_CLASSES)
best_model_vgg16.load_weights(MODEL_NAME_VGG16)
best_model_vgg16.compile(loss='categorical_crossentropy',
best_model_vgg16.compile(loss='categorical_crossentropy',
                         # optimizer=opt,
                         optimizer='adam',
                         metrics=['accuracy'])

def print_score(title, model, X_test, y_test, label_class):
    scores = model.evaluate(X_test, y_test, verbose=1)
    print("{0} {1} {2}".format(title, model.metrics_names[1], scores[1]*100))
    y_pred = model.predict(X_test)
    print('\n', classification_report(np.where(y_test > 0)[1],
                                      np.argmax(y_pred, axis=1),
                                      target_names=list(label_class.keys())), sep='')

def plot_accuracy(title, history):
    plt.figure(figsize=(8,8))
    plt.plot(history.history['acc'])     # Training Accuracy
    plt.plot(history.history['val_acc']) # Validation Accuracy
    plt.title('{0} Model Accuracy'.format(title))
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

def plot_loss(title, history):
    plt.figure(figsize=(8,8))
    plt.plot(history.history['loss'])     # Training Loss
    plt.plot(history.history['val_loss']) # Validation Loss
    plt.title('{0} Model Loss'.format(title))
```

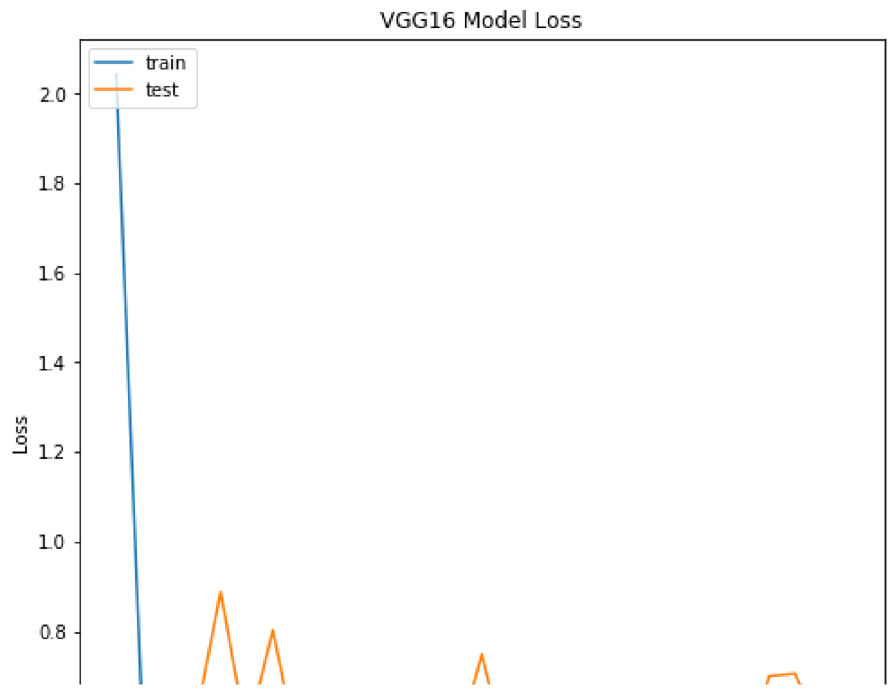
Saved successfully!

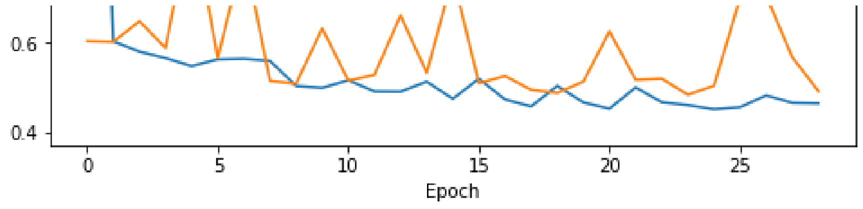
```
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

# Scores
print_score('VGG16', best_model_vgg16, X_test, y_test, LABEL_CLASS)

# Plot Accuracy
plot_accuracy('VGG16', history_vgg16)

# Plot Model Loss
plot_loss('VGG16', history_vgg16)

watch_print('Evaluate Model')
```

```
2435/2435 [==============================] - 8s 3ms/sample - loss: 0.4888 - acc: 0.8074
VGG16 acc 80.73921799659729

              precision    recall  f1-score   support

        Down       0.82      0.71      0.76      1046
          Up       0.80      0.88      0.84      1389

    accuracy                           0.81      2435
   macro avg       0.81      0.79      0.80      2435
weighted avg       0.81      0.81      0.80      2435
```



Saved successfully!





```
Evaluate Model 36.243 seconds
```

```
# !pip freeze > "drive/My Drive/Colab Notebooks/requirements_ChartsPrediction_VGG16_1_day_50_periods.tx
```

Saved successfully!