```
# ===== 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())
```

```
Sat Feb 29 23:40:23 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   33C    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: 11877076774216315395
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 15791819063071657308
physical_device_desc: "device: XLA_CPU device"
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 6432898729218651728
physical_device_desc: "device: XLA_GPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 15956161332
locality {
  bus_id: 1
  links {
  }
}
incarnation: 16446644522011177398
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, GlobalAveragePooling2
from tensorflow.keras.layers import BatchNormalization, Activation, Input, AveragePooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateScheduler, ReduceLROn
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_RESNET = PATH_ROOT + '/Models/ChartImages/model_resnet_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_RESNET = 32
NUM_DEPTH = 29
NUM_EPOCHS = 100

# Utils
start_time = time.time()
def watch_restart():
    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")
def lr_schedule(epochs):
    lr = 1e-3
    if epochs > 180:
        lr *= 0.5e-3
    elif epochs > 160:
        lr *= 1e-3
    elif epochs > 120:
        lr *= 1e-2
    elif epochs > 80:
        lr *= 1e-1
    return lr

# Callbacks for model saving and stopping.
# Training should be stopped when val_acc (validation accuracy) clearly stops increasing to prevent ove
checkpoint_resnet = ModelCheckpoint(filepath=MODEL_NAME_RESNET, monitor='val_acc', verbose=1, save_best_
lr_scheduler = LearningRateScheduler(lr_schedule)
```

```
lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1), cooldown=0, patience=5, min_lr=0.5e-6)
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 21 GB
Memory Used 1 GB
Memory Used: 4.8 percent
Get Data 2.6114 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)

    # Convert numpy array to OpenCv image
    item_image = cv2.imdecode(item_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
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: 52.9 percent
Prepare Data 19.2904 seconds
```

```
watch_restart()

# ===== 3. Create Model Resnet
def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):

    conv = Conv2D(num_filters,
                  kernel_size=kernel_size,
                  strides=strides,
                  padding='same',
                  kernel_initializer='he_normal',
                  kernel_regularizer=l2(1e-4))

    x = inputs
    if conv_first:
        x = conv(x)
```

```
            if batch_normalization:
                x = BatchNormalization()(x)
            if activation is not None:
                x = Activation(activation)(x)
        else:
            if batch_normalization:
                x = BatchNormalization()(x)
            if activation is not None:
                x = Activation(activation)(x)
            x = conv(x)
        return x

    def get_model_resnet(input_shape, depth, num_classes):
        """ResNet

        Stacks of (1 x 1)-(3 x 3)-(1 x 1) BN-ReLU-Conv2D or also known as
        bottleneck layer
        First shortcut connection per layer is 1 x 1 Conv2D.
        Second and onwards shortcut connection is identity.
        At the beginning of each stage, the feature map size is halved (downsampled)
        by a convolutional layer with strides=2, while the number of filter maps is
        doubled. Within each stage, the layers have the same number filters and the
        same filter map sizes.
        Features maps sizes:
        conv1  : 32x32,  16
        stage 0: 32x32,  64
        stage 1: 16x16, 128
        stage 2:  8x8,  256

        """
        if (depth - 2) % 9 != 0:
            raise ValueError('depth should be 9n+2 (eg 56 or 110 in [b])')
        # Start model definition.
        num_filters_in = 16
        num_res_blocks = int((depth - 2) / 9)

        inputs = Input(shape=input_shape)
        # v2 performs Conv2D with BN-ReLU on input before splitting into 2 paths
        x = resnet_layer(inputs=inputs,
                         num_filters=num_filters_in,
                         conv_first=True)

        # Instantiate the stack of residual units
        for stage in range(3):
            for res_block in range(num_res_blocks):
                activation = 'relu'
                batch_normalization = True
                strides = 1
                if stage == 0:
                    num_filters_out = num_filters_in * 4
                    if res_block == 0:  # first layer and first stage
                        activation = None
                        batch_normalization = False
                else:
                    num_filters_out = num_filters_in * 2
                    if res_block == 0:  # first layer but not first stage
                        strides = 2    # downsample
```

```
            # bottleneck residual unit
            y = resnet_layer(inputs=x,
                             num_filters=num_filters_in,
                             kernel_size=1,
                             strides=strides,
                             activation=activation,
                             batch_normalization=batch_normalization,
                             conv_first=False)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters_in,
                             conv_first=False)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters_out,
                             kernel_size=1,
                             conv_first=False)
            if res_block == 0:
                # linear projection residual shortcut connection to match
                # changed dims
                x = resnet_layer(inputs=x,
                                 num_filters=num_filters_out,
                                 kernel_size=1,
                                 strides=strides,
                                 activation=None,
                                 batch_normalization=False)
            x = tf.keras.layers.add([x, y])

        num_filters_in = num_filters_out

    # Add classifier on top.
    # Has BN-ReLU before Pooling
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = AveragePooling2D(pool_size=8)(x)
    y = Flatten()(x)
    outputs = Dense(num_classes,
                    activation='softmax',
                    kernel_initializer='he_normal')(y)

    # Instantiate model.
    model = Model(inputs=inputs, outputs=outputs)
    return model

model_resnet = get_model_resnet(input_shape=IMAGE_INPUT_SHAPE,
                                depth=NUM_DEPTH,
                                num_classes=NUM_CLASSES)
model_resnet.compile(loss='categorical_crossentropy',
                     optimizer=Adam(lr=lr_schedule(NUM_EPOCHS)),
                     metrics=['accuracy'])
model_resnet.summary()

history_resnet = model_resnet.fit(X_train,
                                  y_train,
                                  batch_size=NUM_BATCH_SIZE_RESNET,
                                  epochs=NUM_EPOCHS,
                                  validation_data=(X_test, y_test),
                                  shuffle=True,
                                  callbacks=[checkpoint_resnet, lr_reducer, lr_scheduler, early_stoppin
```

```
watch_print('Create Model Resnet')
```

```
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.
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv2d (Conv2D)	(None, 224, 224, 16)	448	input_1[0][0]
batch_normalization (BatchNorma	(None, 224, 224, 16)	64	conv2d[0][0]
activation (Activation)	(None, 224, 224, 16)	0	batch_normalization[0][0]
conv2d_1 (Conv2D)	(None, 224, 224, 16)	272	activation[0][0]
batch_normalization_1 (BatchNor	(None, 224, 224, 16)	64	conv2d_1[0][0]
activation_1 (Activation)	(None, 224, 224, 16)	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, 224, 224, 16)	2320	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, 224, 224, 16)	64	conv2d_2[0][0]
activation_2 (Activation)	(None, 224, 224, 16)	0	batch_normalization_2[0][0]
conv2d_4 (Conv2D)	(None, 224, 224, 64)	1088	activation[0][0]
conv2d_3 (Conv2D)	(None, 224, 224, 64)	1088	activation_2[0][0]
add (Add)	(None, 224, 224, 64)	0	conv2d_4[0][0] conv2d_3[0][0]
batch_normalization_3 (BatchNor	(None, 224, 224, 64)	256	add[0][0]
activation_3 (Activation)	(None, 224, 224, 64)	0	batch_normalization_3[0][0]
conv2d_5 (Conv2D)	(None, 224, 224, 16)	1040	activation_3[0][0]
batch_normalization_4 (BatchNor	(None, 224, 224, 16)	64	conv2d_5[0][0]
activation_4 (Activation)	(None, 224, 224, 16)	0	batch_normalization_4[0][0]
conv2d_6 (Conv2D)	(None, 224, 224, 16)	2320	activation_4[0][0]
batch_normalization_5 (BatchNor	(None, 224, 224, 16)	64	conv2d_6[0][0]
activation_5 (Activation)	(None, 224, 224, 16)	0	batch_normalization_5[0][0]
conv2d_7 (Conv2D)	(None, 224, 224, 64)	1088	activation_5[0][0]
add_1 (Add)	(None, 224, 224, 64)	0	add[0][0] conv2d_7[0][0]
batch_normalization_6 (BatchNor	(None, 224, 224, 64)	256	add_1[0][0]
activation_6 (Activation)	(None, 224, 224, 64)	0	batch_normalization_6[0][0]
conv2d_8 (Conv2D)	(None, 224, 224, 16)	1040	activation_6[0][0]
batch_normalization_7 (BatchNor	(None, 224, 224, 16)	64	conv2d_8[0][0]
activation_7 (Activation)	(None, 224, 224, 16)	0	batch_normalization_7[0][0]

activation_7 (Activation)	(None, 224, 224, 16)	0	batch_normalization_7[0][0]
conv2d_9 (Conv2D)	(None, 224, 224, 16)	2320	activation_7[0][0]
batch_normalization_8 (BatchNor	(None, 224, 224, 16)	64	conv2d_9[0][0]
activation_8 (Activation)	(None, 224, 224, 16)	0	batch_normalization_8[0][0]
conv2d_10 (Conv2D)	(None, 224, 224, 64)	1088	activation_8[0][0]
add_2 (Add)	(None, 224, 224, 64)	0	add_1[0][0] conv2d_10[0][0]
batch_normalization_9 (BatchNor	(None, 224, 224, 64)	256	add_2[0][0]
activation_9 (Activation)	(None, 224, 224, 64)	0	batch_normalization_9[0][0]
conv2d_11 (Conv2D)	(None, 112, 112, 64)	4160	activation_9[0][0]
batch_normalization_10 (BatchNo	(None, 112, 112, 64)	256	conv2d_11[0][0]
activation_10 (Activation)	(None, 112, 112, 64)	0	batch_normalization_10[0][0]
conv2d_12 (Conv2D)	(None, 112, 112, 64)	36928	activation_10[0][0]
batch_normalization_11 (BatchNo	(None, 112, 112, 64)	256	conv2d_12[0][0]
activation_11 (Activation)	(None, 112, 112, 64)	0	batch_normalization_11[0][0]
conv2d_14 (Conv2D)	(None, 112, 112, 128	8320	add_2[0][0]
conv2d_13 (Conv2D)	(None, 112, 112, 128	8320	activation_11[0][0]
add_3 (Add)	(None, 112, 112, 128	0	conv2d_14[0][0] conv2d_13[0][0]
batch_normalization_12 (BatchNo	(None, 112, 112, 128	512	add_3[0][0]
activation_12 (Activation)	(None, 112, 112, 128	0	batch_normalization_12[0][0]
conv2d_15 (Conv2D)	(None, 112, 112, 64)	8256	activation_12[0][0]
batch_normalization_13 (BatchNo	(None, 112, 112, 64)	256	conv2d_15[0][0]
activation_13 (Activation)	(None, 112, 112, 64)	0	batch_normalization_13[0][0]
conv2d_16 (Conv2D)	(None, 112, 112, 64)	36928	activation_13[0][0]
batch_normalization_14 (BatchNo	(None, 112, 112, 64)	256	conv2d_16[0][0]
activation_14 (Activation)	(None, 112, 112, 64)	0	batch_normalization_14[0][0]
conv2d_17 (Conv2D)	(None, 112, 112, 128	8320	activation_14[0][0]
add_4 (Add)	(None, 112, 112, 128	0	add_3[0][0] conv2d_17[0][0]
batch_normalization_15 (BatchNo	(None, 112, 112, 128	512	add_4[0][0]
activation_15 (Activation)	(None, 112, 112, 128	0	batch_normalization_15[0][0]
conv2d_18 (Conv2D)	(None, 112, 112, 64)	8256	activation_15[0][0]
batch_normalization_16 (BatchNo	(None, 112, 112, 64)	256	conv2d_18[0][0]

activation_16 (Activation)	(None, 112, 112, 64)	0	batch_normalization_16[0][0]
conv2d_19 (Conv2D)	(None, 112, 112, 64)	36928	activation_16[0][0]
batch_normalization_17 (BatchNo	(None, 112, 112, 64)	256	conv2d_19[0][0]
activation_17 (Activation)	(None, 112, 112, 64)	0	batch_normalization_17[0][0]
conv2d_20 (Conv2D)	(None, 112, 112, 128	8320	activation_17[0][0]
add_5 (Add)	(None, 112, 112, 128	0	add_4[0][0] conv2d_20[0][0]
batch_normalization_18 (BatchNo	(None, 112, 112, 128	512	add_5[0][0]
activation_18 (Activation)	(None, 112, 112, 128	0	batch_normalization_18[0][0]
conv2d_21 (Conv2D)	(None, 56, 56, 128)	16512	activation_18[0][0]
batch_normalization_19 (BatchNo	(None, 56, 56, 128)	512	conv2d_21[0][0]
activation_19 (Activation)	(None, 56, 56, 128)	0	batch_normalization_19[0][0]
conv2d_22 (Conv2D)	(None, 56, 56, 128)	147584	activation_19[0][0]
batch_normalization_20 (BatchNo	(None, 56, 56, 128)	512	conv2d_22[0][0]
activation_20 (Activation)	(None, 56, 56, 128)	0	batch_normalization_20[0][0]
conv2d_24 (Conv2D)	(None, 56, 56, 256)	33024	add_5[0][0]
conv2d_23 (Conv2D)	(None, 56, 56, 256)	33024	activation_20[0][0]
add_6 (Add)	(None, 56, 56, 256)	0	conv2d_24[0][0] conv2d_23[0][0]
batch_normalization_21 (BatchNo	(None, 56, 56, 256)	1024	add_6[0][0]
activation_21 (Activation)	(None, 56, 56, 256)	0	batch_normalization_21[0][0]
conv2d_25 (Conv2D)	(None, 56, 56, 128)	32896	activation_21[0][0]
batch_normalization_22 (BatchNo	(None, 56, 56, 128)	512	conv2d_25[0][0]
activation_22 (Activation)	(None, 56, 56, 128)	0	batch_normalization_22[0][0]
conv2d_26 (Conv2D)	(None, 56, 56, 128)	147584	activation_22[0][0]
batch_normalization_23 (BatchNo	(None, 56, 56, 128)	512	conv2d_26[0][0]
activation_23 (Activation)	(None, 56, 56, 128)	0	batch_normalization_23[0][0]
conv2d_27 (Conv2D)	(None, 56, 56, 256)	33024	activation_23[0][0]
add_7 (Add)	(None, 56, 56, 256)	0	add_6[0][0] conv2d_27[0][0]
batch_normalization_24 (BatchNo	(None, 56, 56, 256)	1024	add_7[0][0]
activation_24 (Activation)	(None, 56, 56, 256)	0	batch_normalization_24[0][0]
conv2d_28 (Conv2D)	(None, 56, 56, 128)	32896	activation_24[0][0]
batch_normalization_25 (BatchNo	(None, 56, 56, 128)	512	conv2d_28[0][0]

```
batch_normalization_25 (BatchNo (None, 56, 56, 128)  512         conv2d_28[0][0]
__________________________________________________________________________________________________
activation_25 (Activation)      (None, 56, 56, 128)  0           batch_normalization_25[0][0]
__________________________________________________________________________________________________
conv2d_29 (Conv2D)              (None, 56, 56, 128)  147584      activation_25[0][0]
__________________________________________________________________________________________________
batch_normalization_26 (BatchNo (None, 56, 56, 128)  512         conv2d_29[0][0]
__________________________________________________________________________________________________
activation_26 (Activation)      (None, 56, 56, 128)  0           batch_normalization_26[0][0]
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 56, 56, 256)  33024       activation_26[0][0]
__________________________________________________________________________________________________
add_8 (Add)                     (None, 56, 56, 256)  0           add_7[0][0]
                                                                 conv2d_30[0][0]
__________________________________________________________________________________________________
batch_normalization_27 (BatchNo (None, 56, 56, 256)  1024        add_8[0][0]
__________________________________________________________________________________________________
activation_27 (Activation)      (None, 56, 56, 256)  0           batch_normalization_27[0][0]
__________________________________________________________________________________________________
average_pooling2d (AveragePooli (None, 7, 7, 256)    0           activation_27[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 12544)        0           average_pooling2d[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 2)            25090       flatten[0][0]
==================================================================================================
Total params: 871,522
Trainable params: 866,306
Non-trainable params: 5,216
__________________________________________________________________________________________________
Train on 7303 samples, validate on 2435 samples
Epoch 1/100
7296/7303 [============================>.] - ETA: 0s - loss: 1.1425 - acc: 0.7640
Epoch 00001: val_acc improved from -inf to 0.79630, saving model to drive/My Drive/Colab Notebooks/
7303/7303 [==============================] - 129s 18ms/sample - loss: 1.1423 - acc: 0.7641 - val_lo
Epoch 2/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.9031 - acc: 0.8147
Epoch 00002: val_acc improved from 0.79630 to 0.84846, saving model to drive/My Drive/Colab Noteboo
7303/7303 [==============================] - 113s 16ms/sample - loss: 0.9031 - acc: 0.8145 - val_lo
Epoch 3/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.7761 - acc: 0.8292
Epoch 00003: val_acc did not improve from 0.84846
7303/7303 [==============================] - 111s 15ms/sample - loss: 0.7760 - acc: 0.8291 - val_lo
Epoch 4/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.6883 - acc: 0.8439
Epoch 00004: val_acc did not improve from 0.84846
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.6884 - acc: 0.8438 - val_lo
Epoch 5/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.6274 - acc: 0.8444
Epoch 00005: val_acc did not improve from 0.84846
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.6276 - acc: 0.8442 - val_lo
Epoch 6/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5856 - acc: 0.8513
Epoch 00006: val_acc did not improve from 0.84846
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.5854 - acc: 0.8514 - val_lo
Epoch 7/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5467 - acc: 0.8542
Epoch 00007: val_acc did not improve from 0.84846
7303/7303 [==============================] - 111s 15ms/sample - loss: 0.5465 - acc: 0.8543 - val_lo
Epoch 8/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.5235 - acc: 0.8575
Epoch 00008: val_acc improved from 0.84846 to 0.86283, saving model to drive/My Drive/Colab Noteboo
7303/7303 [==============================] - 112s 15ms/sample - loss: 0.5241 - acc: 0.8572 - val_lo
Epoch 9/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4996 - acc: 0.8576
Epoch 00009: val_acc did not improve from 0.86283
```

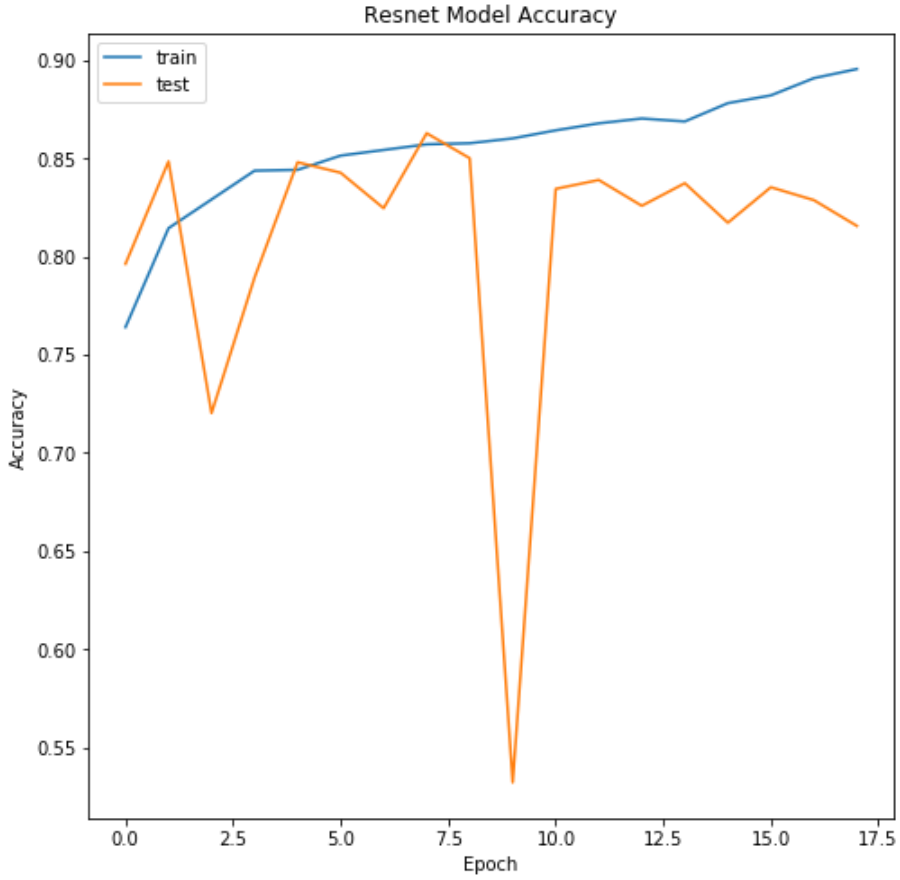
```
Epoch 00009: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4994 - acc: 0.8577 - val_lo
Epoch 10/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4829 - acc: 0.8603
Epoch 00010: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4835 - acc: 0.8602 - val_lo
Epoch 11/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4674 - acc: 0.8643
Epoch 00011: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4676 - acc: 0.8643 - val_lo
Epoch 12/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4496 - acc: 0.8679
Epoch 00012: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4499 - acc: 0.8679 - val_lo
Epoch 13/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4374 - acc: 0.8703
Epoch 00013: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4375 - acc: 0.8703 - val_lo
Epoch 14/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4283 - acc: 0.8688
Epoch 00014: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4284 - acc: 0.8688 - val_lo
Epoch 15/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.4060 - acc: 0.8783
Epoch 00015: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.4062 - acc: 0.8781 - val_lo
Epoch 16/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.3908 - acc: 0.8821
Epoch 00016: val_acc did not improve from 0.86283
7303/7303 [==============================] - 110s 15ms/sample - loss: 0.3909 - acc: 0.8821 - val_lo
Epoch 17/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.3745 - acc: 0.8912
Epoch 00017: val_acc did not improve from 0.86283
7303/7303 [==============================] - 111s 15ms/sample - loss: 0.3751 - acc: 0.8909 - val_lo
Epoch 18/100
7296/7303 [============================>.] - ETA: 0s - loss: 0.3621 - acc: 0.8957
Epoch 00018: val_acc did not improve from 0.86283
7303/7303 [==============================] - 111s 15ms/sample - loss: 0.3623 - acc: 0.8955 - val_lo
Epoch 00018: early stopping
Create Model Resnet 2022.3518 seconds
```

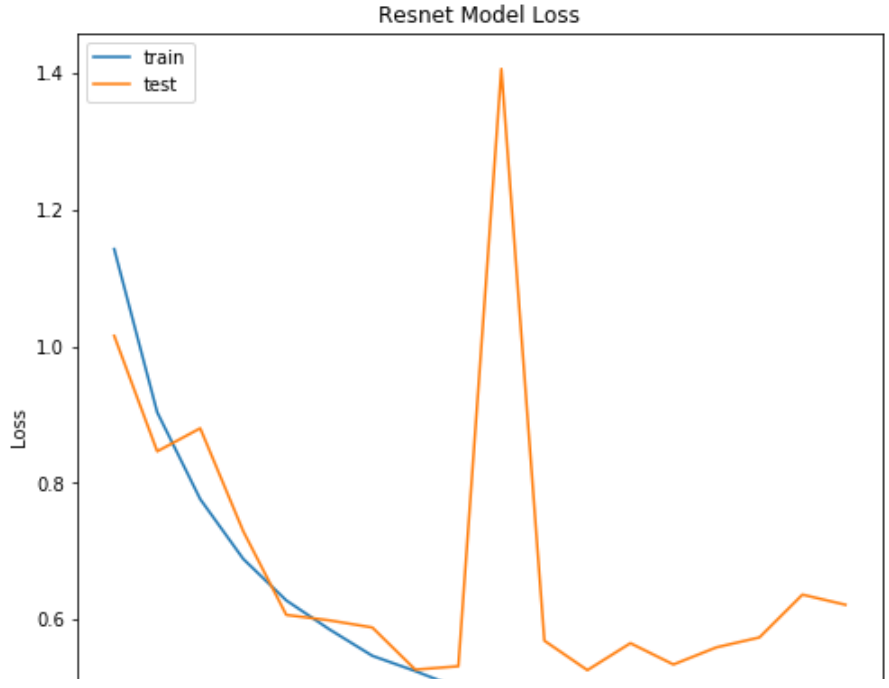
```
watch_restart()

# ===== 4. Evaluate Models
best_model_resnet = get_model_resnet(input_shape=IMAGE_INPUT_SHAPE, depth=NUM_DEPTH, num_classes=NUM_CLA
best_model_resnet.load_weights(MODEL_NAME_RESNET)
best_model_resnet.compile(loss='categorical_crossentropy',
                          optimizer=Adam(lr=lr_schedule(NUM_EPOCHS)),
                          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)
```

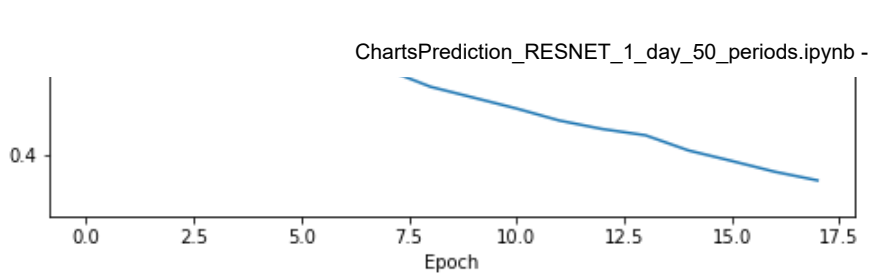
```
        print('\n', classification_report(np.where(y_test > 0)[1],
                                          np.argmax(y_pred, axis=1),
                                          target_names=list(label_class.keys())), sep='')

    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))
        plt.ylabel('Loss')
        plt.xlabel('Epoch')
        plt.legend(['train', 'test'], loc='upper left')
        plt.show()

    # Scores
    print_score('Resnet', best_model_resnet, X_test, y_test, LABEL_CLASS)

    # Plot Accuracy
    plot_accuracy('Resnet', history_resnet)

    # Plot Model Loss
    plot_loss('Resnet', history_resnet)

    watch_print('Evaluate Model')
```

```
2435/2435 [==============================] - 11s 5ms/sample - loss: 0.5262 - acc: 0.8628
Resnet acc 86.28336787223816
```

	precision	recall	f1-score	support
Down	0.88	0.79	0.83	1046
Up	0.85	0.92	0.88	1389
accuracy			0.86	2435
macro avg	0.87	0.85	0.86	2435
weighted avg	0.86	0.86	0.86	2435








```
Evaluate Model 34.9093 seconds
```