CNN_Malicious_Network_Traffic

April 23, 2022

[ ]:
```
# Importing all necessary libraries
import tensorflow
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense
from tensorflow.keras import backend as K

img_width, img_height = 256, 1024
```

[ ]:
```
train_data_dir = '/content/drive/MyDrive/trafic-dataset/trafic-dataset/train'
validation_data_dir = '/content/drive/MyDrive/trafic-dataset/trafic-dataset/
 ↪validation'
test_data_dir = '/content/drive/MyDrive/trafic-dataset/trafic-dataset/test'

nb_train_samples =666
nb_validation_samples = 50
epochs = 25
batch_size = 16
```

[ ]:
```
if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)
```

0.0.1 Model 1: Printing class label using sigmoid activation and last layer with 1 neuron.

[ ]:
```
model = Sequential()
model.add(Conv2D(32, (2, 2), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (2, 2)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (2, 2)))
```

```
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

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

[ ]:
```
# Load the TensorBoard notebook extension
%load_ext tensorboard

import tensorflow
import datetime



log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tensorflow.keras.callbacks.TensorBoard(log_dir=log_dir,␣
 ↪histogram_freq=1)
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

[ ]:
```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')
```

```
history=model.fit(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
    callbacks=[tensorboard_callback])
```

```
Found 666 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
Epoch 1/25
41/41 [==============================] - 287s 7s/step - loss: 2.7893 - accuracy:
0.6815 - val_loss: 1.1428 - val_accuracy: 0.5625
Epoch 2/25
41/41 [==============================] - 148s 4s/step - loss: 0.5531 - accuracy:
0.7477 - val_loss: 0.3648 - val_accuracy: 0.8750
Epoch 3/25
41/41 [==============================] - 148s 4s/step - loss: 0.5159 - accuracy:
0.7862 - val_loss: 0.3360 - val_accuracy: 0.8542
Epoch 4/25
41/41 [==============================] - 146s 4s/step - loss: 0.5144 - accuracy:
0.7831 - val_loss: 0.3409 - val_accuracy: 0.8542
Epoch 5/25
41/41 [==============================] - 144s 3s/step - loss: 0.4631 - accuracy:
0.8108 - val_loss: 0.2962 - val_accuracy: 0.8750
Epoch 6/25
41/41 [==============================] - 144s 3s/step - loss: 0.4681 - accuracy:
0.7969 - val_loss: 0.5094 - val_accuracy: 0.6875
Epoch 7/25
41/41 [==============================] - 144s 3s/step - loss: 0.5139 - accuracy:
0.8108 - val_loss: 0.2689 - val_accuracy: 0.8542
Epoch 8/25
41/41 [==============================] - 144s 4s/step - loss: 0.4683 - accuracy:
0.8246 - val_loss: 0.5011 - val_accuracy: 0.7917
Epoch 9/25
41/41 [==============================] - 145s 4s/step - loss: 0.4693 - accuracy:
0.8262 - val_loss: 0.2580 - val_accuracy: 0.8958
Epoch 10/25
41/41 [==============================] - 145s 4s/step - loss: 0.5161 - accuracy:
0.8169 - val_loss: 0.4056 - val_accuracy: 0.7292
Epoch 11/25
41/41 [==============================] - 145s 4s/step - loss: 0.4360 - accuracy:
0.8262 - val_loss: 0.3950 - val_accuracy: 0.7708
Epoch 12/25
41/41 [==============================] - 145s 4s/step - loss: 0.4227 - accuracy:
0.8169 - val_loss: 0.6653 - val_accuracy: 0.7083
```

```
Epoch 13/25
41/41 [==============================] - 144s 4s/step - loss: 0.4112 - accuracy:
0.8369 - val_loss: 0.2477 - val_accuracy: 0.8750
Epoch 14/25
41/41 [==============================] - 145s 4s/step - loss: 0.4309 - accuracy:
0.8400 - val_loss: 0.3300 - val_accuracy: 0.9375
Epoch 15/25
41/41 [==============================] - 145s 4s/step - loss: 0.4317 - accuracy:
0.8308 - val_loss: 0.2828 - val_accuracy: 0.8333
Epoch 16/25
41/41 [==============================] - 144s 4s/step - loss: 0.4325 - accuracy:
0.8200 - val_loss: 0.1656 - val_accuracy: 0.9583
Epoch 17/25
41/41 [==============================] - 146s 4s/step - loss: 0.4738 - accuracy:
0.8246 - val_loss: 0.3414 - val_accuracy: 0.7708
Epoch 18/25
41/41 [==============================] - 146s 4s/step - loss: 0.4067 - accuracy:
0.8554 - val_loss: 0.3409 - val_accuracy: 0.8958
Epoch 19/25
41/41 [==============================] - 145s 4s/step - loss: 0.5871 - accuracy:
0.8508 - val_loss: 0.2284 - val_accuracy: 0.8750
Epoch 20/25
41/41 [==============================] - 144s 3s/step - loss: 0.4242 - accuracy:
0.8646 - val_loss: 0.1949 - val_accuracy: 0.8750
Epoch 21/25
41/41 [==============================] - 150s 4s/step - loss: 0.3893 - accuracy:
0.8523 - val_loss: 0.2677 - val_accuracy: 0.8542
Epoch 22/25
41/41 [==============================] - 150s 4s/step - loss: 0.4169 - accuracy:
0.8508 - val_loss: 0.1709 - val_accuracy: 0.9375
Epoch 23/25
41/41 [==============================] - 151s 4s/step - loss: 0.3984 - accuracy:
0.8462 - val_loss: 0.4028 - val_accuracy: 0.7292
Epoch 24/25
41/41 [==============================] - 152s 4s/step - loss: 0.5671 - accuracy:
0.8554 - val_loss: 0.2310 - val_accuracy: 0.8542
Epoch 25/25
41/41 [==============================] - 149s 4s/step - loss: 0.3825 - accuracy:
0.8615 - val_loss: 0.8191 - val_accuracy: 0.7500
```

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

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 255, 1023, 32)     416
```

```
activation_10 (Activation)  (None, 255, 1023, 32)     0

max_pooling2d_6 (MaxPooling  (None, 127, 511, 32)     0
2D)

conv2d_7 (Conv2D)           (None, 126, 510, 32)      4128

activation_11 (Activation)  (None, 126, 510, 32)      0

max_pooling2d_7 (MaxPooling  (None, 63, 255, 32)      0
2D)

conv2d_8 (Conv2D)           (None, 62, 254, 64)       8256

activation_12 (Activation)  (None, 62, 254, 64)       0

max_pooling2d_8 (MaxPooling  (None, 31, 127, 64)      0
2D)

flatten_2 (Flatten)         (None, 251968)            0

dense_4 (Dense)             (None, 64)                16126016

activation_13 (Activation)  (None, 64)                0

dropout_2 (Dropout)         (None, 64)                0

dense_5 (Dense)             (None, 1)                 65

activation_14 (Activation)  (None, 1)                 0

=================================================================
Total params: 16,138,881
Trainable params: 16,138,881
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
%tensorboard --logdir logs/fit
```

```
<IPython.core.display.Javascript object>
```

[ ]:
```
history.history
```

[ ]:
```
{'accuracy': [0.681538462638855,
  0.7476922869682312,
  0.7861538529396057,
  0.7830769419670105,
  0.810769259929657,
```

```
 0.7969231009483337,
 0.810769259929657,
 0.8246153593063354,
 0.8261538743972778,
 0.8169230818748474,
 0.8261538743972778,
 0.8169230818748474,
 0.8369230628013611,
 0.8399999737739563,
 0.8307692408561707,
 0.8199999928474426,
 0.8246153593063354,
 0.8553845882415771,
 0.8507692217826843,
 0.8646153807640076,
 0.8523076772689819,
 0.8507692217826843,
 0.8461538553237915,
 0.8553845882415771,
 0.8615384697914124],
'loss': [2.789276361465454,
 0.5531468987464905,
 0.5158843398094177,
 0.514446496963501,
 0.46311208605766296,
 0.4681459367275238,
 0.5138962268829346,
 0.46825647354125977,
 0.46932485699653625,
 0.5161124467849731,
 0.4359683096408844,
 0.4226972460746765,
 0.41118690371513367,
 0.4308698773384094,
 0.43167516589164734,
 0.4324813187122345,
 0.473763108253479,
 0.40672463178634644,
 0.5870655179023743,
 0.4241674840450287,
 0.38930684328079224,
 0.416851669549942,
 0.39837637543678284,
 0.5671108365058899,
 0.38252755999565125],
'val_accuracy': [0.5625,
 0.875,
```

```
 0.8541666865348816,
 0.8541666865348816,
 0.875,
 0.6875,
 0.8541666865348816,
 0.7916666865348816,
 0.8958333134651184,
 0.7291666865348816,
 0.7708333134651184,
 0.7083333134651184,
 0.875,
 0.9375,
 0.8333333134651184,
 0.9583333134651184,
 0.7708333134651184,
 0.8958333134651184,
 0.875,
 0.875,
 0.8541666865348816,
 0.9375,
 0.7291666865348816,
 0.8541666865348816,
 0.75],
'val_loss': [1.1428378820419312,
 0.3647801876068115,
 0.33596310019493103,
 0.3408946990966797,
 0.2961524724960327,
 0.509446918964386,
 0.268928200006485,
 0.5011337995529175,
 0.2579786777496338,
 0.4055846631526947,
 0.3949635922908783,
 0.6652805209159851,
 0.2476949542760849,
 0.3299834430217743,
 0.28283828496932983,
 0.16557449102401733,
 0.3413766920566559,
 0.3409126102924347,
 0.228395476937294,
 0.19489997625350952,
 0.2677079141139984,
 0.17086251080036163,
 0.40283456444740295,
 0.23101156949996948,
```

```
0.8190538287162781]}
```

[ ]:
```
train_generator.class_indices
```

[ ]:
```
{'Malware': 0, 'Normal': 1}
```

[ ]:
```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.applications.vgg16 import decode_predictions
from tensorflow.keras.applications.vgg16 import VGG16
import numpy as np

from keras.models import load_model


image = load_img('/content/drive/MyDrive/trafic-dataset/trafic-dataset/test/
↪Malware/2017-02-09-Hancitor-Pony-malspam-traffic.png', target_size=(256,␣
↪1024))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,256,1024,3)
preds=model.predict(img)
```

[ ]:
```
preds[0][0]
```

[ ]:
```
0.015496641
```

[ ]:
```
print("Malware:0, Normal:1   -> ANS:", 0 if preds[0][0]<0.5 else 1 )
```

```
Malware:0, Normal:1   -> ANS: 0
```

[ ]:
```
image = load_img('/content/drive/MyDrive/trafic-dataset/trafic-dataset/test/
↪Normal/aaa.png', target_size=(256, 1024))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,256,1024,3)
preds=model.predict(img)
```

[ ]:
```
preds
```

[ ]:
```
array([[0.43971652]], dtype=float32)
```

[ ]:
```
print("Malware:0, Normal:1   -> ANS:", 0 if preds[0][0]<0.5 else 1)
```

```
Malware:0, Normal:1   -> ANS: 0
```

0.0.2 Model 2: Printing probabilities/confidence using softmax activation and last layer with 2 neurons

[ ]:
```
model2 = Sequential()
model2.add(Conv2D(32, (2, 2), input_shape=input_shape))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Conv2D(32, (2, 2)))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Conv2D(64, (2, 2)))
model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))

model2.add(Flatten())
model2.add(Dense(64))
model2.add(Activation('relu'))
model2.add(Dropout(0.5))
model2.add(Dense(2))
model2.add(Activation('softmax'))
```

[ ]:
```
model2.compile(loss='sparse_categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

[ ]:
```
log_dir = "logs2/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tensorflow.keras.callbacks.TensorBoard(log_dir=log_dir,␣
↪histogram_freq=1)
```

[ ]:
```
history2 = model2.fit(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=nb_validation_samples // batch_size,
      callbacks=[tensorboard_callback])
```

```
Epoch 1/25
41/41 [==============================] - 154s 4s/step - loss: 2.2695 - accuracy:
0.6662 - val_loss: 2.0554 - val_accuracy: 0.6250
Epoch 2/25
41/41 [==============================] - 150s 4s/step - loss: 0.6587 - accuracy:
0.7369 - val_loss: 0.5470 - val_accuracy: 0.7083
Epoch 3/25
41/41 [==============================] - 151s 4s/step - loss: 0.6289 - accuracy:
0.7662 - val_loss: 0.3711 - val_accuracy: 0.7708
```

```
Epoch 4/25
41/41 [==============================] - 153s 4s/step - loss: 0.5353 - accuracy:
0.7692 - val_loss: 0.5422 - val_accuracy: 0.6458
Epoch 5/25
41/41 [==============================] - 149s 4s/step - loss: 0.5248 - accuracy:
0.8031 - val_loss: 0.2235 - val_accuracy: 0.9167
Epoch 6/25
41/41 [==============================] - 149s 4s/step - loss: 0.6045 - accuracy:
0.8154 - val_loss: 0.4333 - val_accuracy: 0.7500
Epoch 7/25
41/41 [==============================] - 150s 4s/step - loss: 0.4478 - accuracy:
0.8277 - val_loss: 0.3136 - val_accuracy: 0.8333
Epoch 8/25
41/41 [==============================] - 149s 4s/step - loss: 0.4520 - accuracy:
0.8138 - val_loss: 0.1981 - val_accuracy: 0.9167
Epoch 9/25
41/41 [==============================] - 152s 4s/step - loss: 0.4403 - accuracy:
0.8354 - val_loss: 0.5499 - val_accuracy: 0.7500
Epoch 10/25
41/41 [==============================] - 151s 4s/step - loss: 0.5080 - accuracy:
0.8200 - val_loss: 0.2398 - val_accuracy: 0.8542
Epoch 11/25
41/41 [==============================] - 150s 4s/step - loss: 0.4340 - accuracy:
0.8369 - val_loss: 0.2691 - val_accuracy: 0.8333
Epoch 12/25
41/41 [==============================] - 153s 4s/step - loss: 0.4406 - accuracy:
0.8246 - val_loss: 0.2297 - val_accuracy: 0.8542
Epoch 13/25
41/41 [==============================] - 150s 4s/step - loss: 0.4275 - accuracy:
0.8446 - val_loss: 0.2200 - val_accuracy: 0.8542
Epoch 14/25
41/41 [==============================] - 150s 4s/step - loss: 0.4355 - accuracy:
0.8523 - val_loss: 0.1421 - val_accuracy: 0.9583
Epoch 15/25
41/41 [==============================] - 151s 4s/step - loss: 0.4173 - accuracy:
0.8492 - val_loss: 0.1492 - val_accuracy: 0.9375
Epoch 16/25
41/41 [==============================] - 149s 4s/step - loss: 0.4818 - accuracy:
0.8369 - val_loss: 2.2313 - val_accuracy: 0.5625
Epoch 17/25
41/41 [==============================] - 152s 4s/step - loss: 0.4400 - accuracy:
0.8415 - val_loss: 0.2902 - val_accuracy: 0.8125
Epoch 18/25
41/41 [==============================] - 149s 4s/step - loss: 0.3864 - accuracy:
0.8431 - val_loss: 0.2151 - val_accuracy: 0.8333
Epoch 19/25
41/41 [==============================] - 150s 4s/step - loss: 0.4142 - accuracy:
0.8369 - val_loss: 0.1372 - val_accuracy: 0.9375
```

```
Epoch 20/25
41/41 [==============================] - 150s 4s/step - loss: 0.4002 - accuracy:
0.8492 - val_loss: 0.1357 - val_accuracy: 0.9375
Epoch 21/25
41/41 [==============================] - 150s 4s/step - loss: 0.3630 - accuracy:
0.8431 - val_loss: 0.1799 - val_accuracy: 0.9375
Epoch 22/25
41/41 [==============================] - 150s 4s/step - loss: 0.3844 - accuracy:
0.8646 - val_loss: 0.1453 - val_accuracy: 0.9375
Epoch 23/25
41/41 [==============================] - 152s 4s/step - loss: 0.3883 - accuracy:
0.8600 - val_loss: 0.1727 - val_accuracy: 0.9583
Epoch 24/25
41/41 [==============================] - 149s 4s/step - loss: 0.3719 - accuracy:
0.8569 - val_loss: 1.1365 - val_accuracy: 0.7083
Epoch 25/25
41/41 [==============================] - 150s 4s/step - loss: 0.4190 - accuracy:
0.8508 - val_loss: 0.1951 - val_accuracy: 0.8750
```

[ ]:
```
model2.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_9 (Conv2D)           (None, 255, 1023, 32)     416

 activation_15 (Activation)  (None, 255, 1023, 32)     0

 max_pooling2d_9 (MaxPooling  (None, 127, 511, 32)     0
 2D)

 conv2d_10 (Conv2D)          (None, 126, 510, 32)      4128

 activation_16 (Activation)  (None, 126, 510, 32)      0

 max_pooling2d_10 (MaxPoolin  (None, 63, 255, 32)      0
 g2D)

 conv2d_11 (Conv2D)          (None, 62, 254, 64)       8256

 activation_17 (Activation)  (None, 62, 254, 64)       0

 max_pooling2d_11 (MaxPoolin  (None, 31, 127, 64)      0
 g2D)

 flatten_3 (Flatten)         (None, 251968)            0
```

```
 dense_6 (Dense)              (None, 64)                16126016

 activation_18 (Activation)   (None, 64)                0

 dropout_3 (Dropout)          (None, 64)                0

 dense_7 (Dense)              (None, 2)                 130

 activation_19 (Activation)   (None, 2)                 0

=================================================================
Total params: 16,138,946
Trainable params: 16,138,946
Non-trainable params: 0
_________________________________________________________________
```

[ ]:
```
%tensorboard --logdir logs2/fit
```

```
Reusing TensorBoard on port 6006 (pid 122), started 2:33:09 ago. (Use '!kill␣
 ↪122' to kill it.)
```

```
<IPython.core.display.Javascript object>
```

[ ]:
```
history2.history
```

[ ]:
```
{'accuracy': [0.6661538481712341,
  0.736923098564148,
  0.766153872013092,
  0.7692307829856873,
  0.8030769228935242,
  0.8153846263885498,
  0.8276923298835754,
  0.8138461709022522,
  0.8353846073150635,
  0.8199999928474426,
  0.8369230628013611,
  0.8246153593063354,
  0.8446153998374939,
  0.8523076772689819,
  0.8492307662963867,
  0.8369230628013611,
  0.8415384888648987,
  0.8430769443511963,
  0.8369230628013611,
  0.8492307662963867,
  0.8430769443511963,
  0.8646153807640076,
  0.8600000143051147,
```

```
 0.8569231033325195,
 0.8507692217826843],
'loss': [2.2694833278656006,
 0.6586989164352417,
 0.6288999915122986,
 0.5352991223335266,
 0.5247567296028137,
 0.6045235395431519,
 0.4478282332420349,
 0.45196759700775146,
 0.4402863383293152,
 0.5079808235168457,
 0.43396949768066406,
 0.4406389892101288,
 0.42754751443862915,
 0.43551063537597656,
 0.417341023683548,
 0.48182108998298645,
 0.44000229239463806,
 0.38637980818748474,
 0.41421249508857727,
 0.4002458453178406,
 0.36296242475509644,
 0.38438674807548523,
 0.38833534717559814,
 0.3718588948249817,
 0.4189765751361847],
'val_accuracy': [0.625,
 0.7083333134651184,
 0.7708333134651184,
 0.6458333134651184,
 0.9166666865348816,
 0.75,
 0.8333333134651184,
 0.9166666865348816,
 0.75,
 0.8541666865348816,
 0.8333333134651184,
 0.8541666865348816,
 0.8541666865348816,
 0.9583333134651184,
 0.9375,
 0.5625,
 0.8125,
 0.8333333134651184,
 0.9375,
 0.9375,
```

```
  0.9375,
  0.9375,
  0.9583333134651184,
  0.7083333134651184,
  0.875],
 'val_loss': [2.0554466247558594,
  0.5470088124275208,
  0.3710905611515045,
  0.5422499775886536,
  0.22352467477321625,
  0.43326064944267273,
  0.3135581612586975,
  0.19814978539943695,
  0.549854040145874,
  0.23981517553329468,
  0.26914703845977783,
  0.22967500984668732,
  0.21998687088489532,
  0.14209264516830444,
  0.1492435485124588,
  2.231346368789673,
  0.2902291715145111,
  0.21506978571414948,
  0.13717465102672577,
  0.13567067682743073,
  0.1798781156539917,
  0.1453213095664978,
  0.17272986471652985,
  1.1365010738372803,
  0.1951357126235962]}
```

[ ]:
```python
print(train_generator.class_indices)
print(train_generator.classes)
```

```
{'Malware': 0, 'Normal': 1}
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

[ ]:
```
image = load_img('/content/drive/MyDrive/trafic-dataset/trafic-dataset/test/
↪Malware/2017-02-09-Hancitor-Pony-malspam-traffic.png', target_size=(256,␣
↪1024))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,256,1024,3)
preds=model2.predict(img)
```

[ ]:
```
preds
```

[ ]:
```
array([[0.9076093 , 0.09239075]], dtype=float32)
```

[ ]:
```
print("Model believes the example is " + str(preds[0][0]*100) + " %  Malicious")
print("Model believes the example is " + str(preds[0][1]*100) + " %  Normal")
```

```
Model believes the example is 90.76092839241028 %  Malicious
Model believes the example is 9.23907458782196 %  Normal
```

[ ]:
```
image = load_img('/content/drive/MyDrive/trafic-dataset/trafic-dataset/test/
↪Normal/aaa.png', target_size=(256, 1024))
img = np.array(image)
img = img / 255.0
img = img.reshape(1,256,1024,3)
preds=model2.predict(img)
```

[ ]:
```
preds
```

[ ]:
```
array([[0.34220487, 0.65779513]], dtype=float32)
```

[ ]:
```
print("Model believes the example is " + str(preds[0][0]*100) + " %  Malicious")
print("Model believes the example is " + str(preds[0][1]*100) + " %  Normal")
```

```
Model believes the example is 34.220486879348755 %  Malicious
Model believes the example is 65.77951312065125 %  Normal
```

0.0.3 Print Confusion Matrices for Both Models

[ ]:
```
from sklearn.metrics import classification_report, confusion_matrix

preds = model.predict_generator(validation_generator, nb_validation_samples //␣
↪batch_size+1)
y_pred = np.argmax(preds, axis=1)
print('Confusion Matrix')
```

```
print(confusion_matrix(validation_generator.classes, y_pred))
print('Classification Report')
target_names = ['Malicious', 'Normal']
print(classification_report(validation_generator.classes, y_pred,␣
↪target_names=target_names))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning:
`Model.predict_generator` is deprecated and will be removed in a future version.
Please use `Model.predict`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

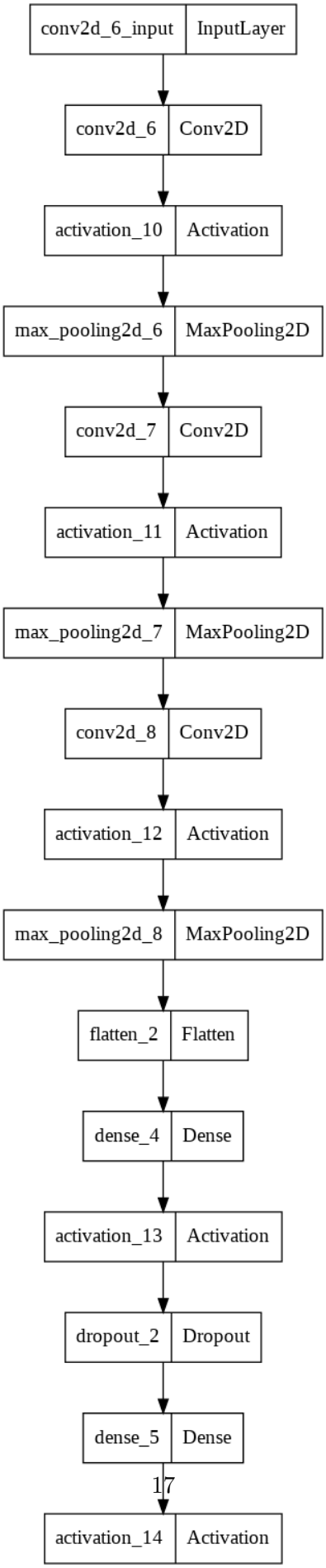
```
Confusion Matrix
[[20  0]
 [30  0]]
Classification Report
              precision    recall  f1-score   support

   Malicious       0.40      1.00      0.57        20
      Normal       0.00      0.00      0.00        30

    accuracy                           0.40        50
   macro avg       0.20      0.50      0.29        50
weighted avg       0.16      0.40      0.23        50
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

0.0.4 Dot Outputs for Both Models

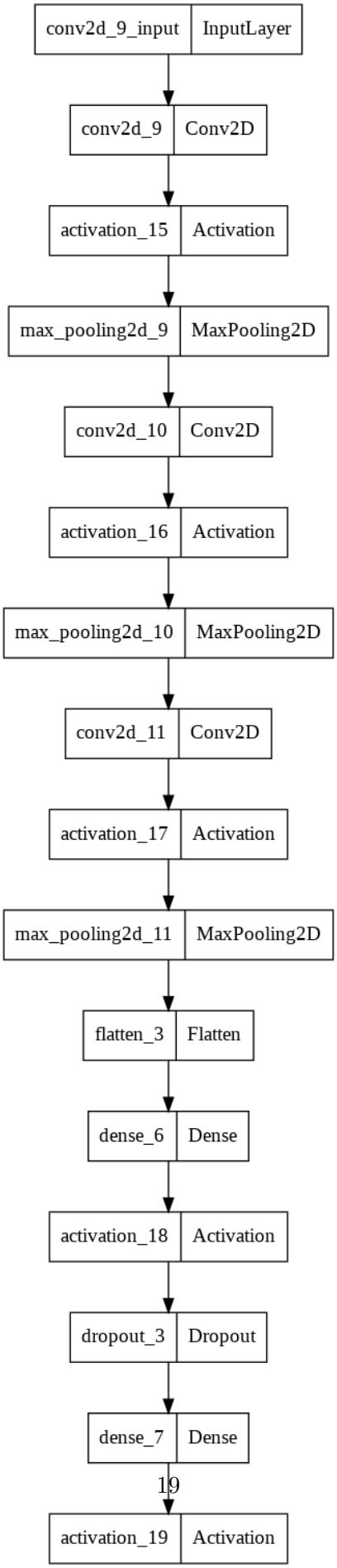
[ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model.png')
```

[ ]:


conv2d_6_input InputLayer
conv2d_6 Conv2D
activation_10 Activation
max_pooling2d_6 MaxPooling2D
conv2d_7 Conv2D
activation_11 Activation
max_pooling2d_7 MaxPooling2D
conv2d_8 Conv2D
activation_12 Activation
max_pooling2d_8 MaxPooling2D
flatten_2 Flatten
dense_4 Dense
activation_13 Activation
dropout_2 Dropout
dense_5 Dense
activation_14 Activation

[ ]:
```
from tensorflow.keras.utils import plot_model
plot_model(model2, to_file='model2.png')
```

[ ]:


conv2d_9_input InputLayer
conv2d_9 Conv2D
activation_15 Activation
max_pooling2d_9 MaxPooling2D
conv2d_10 Conv2D
activation_16 Activation
max_pooling2d_10 MaxPooling2D
conv2d_11 Conv2D
activation_17 Activation
max_pooling2d_11 MaxPooling2D
flatten_3 Flatten
dense_6 Dense
activation_18 Activation
dropout_3 Dropout
dense_7 Dense
activation_19 Activation

[ ]:

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

[ ]: