GTSRB

December 18, 2022

[1]:
```python
from tensorflow.keras.datasets import cifar10
from image import *
from hashlib import md5
import numpy as np
import pandas as pd
import os
import shutil
import warnings
warnings.filterwarnings('ignore')
import cv2 as cv


def get_data(param):
    if param["dataset"] == "CIFAR10":
        (x_train, y_train), (x_test, y_test) = cifar10.load_data()
        x_train = x_train.astype(np.float) / 255.
        x_test = x_test.astype(np.float) / 255.

    if param["dataset"] == "GTSRB":
        train_X = []
        train_y = []
        for i in range(0,43):
            n = str(i)
            train_Path = "gtsrb-german-traffic-sign/Train/" + n
            label = [0 for i in range(0, 43)]
            label[i] = 1
            for filename in os.listdir(train_Path):
                img = cv.imread(train_Path + "/" + filename)
                img = cv.resize(img, (32,32))
                #print(filename)
                train_X.append(img)
                train_y.append(label)
        train_X = np.asarray(train_X)

        train_X = np.asarray(train_X, dtype = "float32")
        train_y = np.asarray(train_y, dtype= "float32")
```

```
meta_df = pd.read_csv('gtsrb-german-traffic-sign/Meta.csv')
test_data = pd.read_csv('gtsrb-german-traffic-sign/Test.csv')
train_data = pd.read_csv('gtsrb-german-traffic-sign/Train.csv')

counter = 0
test_X = []
test_y = []
test_Path = "gtsrb-german-traffic-sign/Test"
for filename in os.listdir(test_Path):
        img = cv.imread(test_Path + "/" + filename)
        img = cv.resize(img, (32,32))
        label = [0 for i in range(0, 43)]
        label[test_data.loc[counter][6]] = 1
        #print(filename)
        test_X.append(img)
        test_y.append(label)
        counter += 1
test_X = np.asarray(test_X)

test_X = np.asarray(test_X, dtype = "float32")
test_y = np.asarray(test_y, dtype= "float32")

train_y_after = [[0] * 1] * 39209

for i in range(39209):
    for j in range(43):
        if (train_y[i][j] == 1):
            train_y_after[i] = [j]
y_train = train_y_after
y_train = np.array(y_train)

test_y_after = [[0] * 1] * 12630

for i in range(12630):
    j = test_data["ClassId"][i]
    test_y_after[i] = [j]
y_test = test_y_after
y_test = np.array(y_test)

#shuffle training set
index = np.arange(39209)
np.random.shuffle(index)
train_X = train_X[index,:,:,:]
y_train = y_train[index]


x_train = train_X.astype(np.float)
```

```
        x_test = test_X.astype(np.float)

    return x_train, y_train, x_test, y_test


def poison(x_train, y_train, param):
    target_label = param["target_label"]
    num_images = int(param["poisoning_rate"] * y_train.shape[0])

    index = np.where(y_train != target_label)
    index = index[0]
    index = index[:num_images]
    x_train[index] = poison_frequency(x_train[index], y_train[index], param)
    y_train[index] = target_label
    return x_train


def poison_frequency(x_train, y_train, param):
    if x_train.shape[0] == 0:
        return x_train

    x_train *= 255.
    if param["YUV"]:
        x_train = RGB2YUV(x_train)

    # transfer to frequency domain
    x_train = DCT(x_train, param["window_size"])  # (idx, ch, w, h)

    # plug trigger frequency
    for i in range(x_train.shape[0]):
        for ch in param["channel_list"]:
            for w in range(0, x_train.shape[2], param["window_size"]):
                for h in range(0, x_train.shape[3], param["window_size"]):
                    for pos in param["pos_list"]:
                        x_train[i][ch][w + pos[0]][h + pos[1]] +=␣
 ↪param["magnitude"]


    x_train = IDCT(x_train, param["window_size"])  # (idx, w, h, ch)

    if param["YUV"]:
        x_train = YUV2RGB(x_train)

    x_train /= 255.
    x_train = np.clip(x_train, 0, 1)
    return x_train
```

```python
def impose(x_train, y_train, param):
    x_train = poison_frequency(x_train, y_train, param)
    return x_train


def digest(param):
    txt = ""
    txt += param["dataset"]
    txt += str(param["target_label"])
    txt += str(param["poisoning_rate"])
    txt += str(param["label_dim"])
    txt += "".join(str(param["channel_list"]))
    txt += str(param["window_size"])
    txt += str(param["magnitude"])
    txt += str(param["YUV"])
    txt += "".join(str(param["pos_list"]))
    hash_md5 = md5()
    hash_md5.update(txt.encode("utf-8"))
    return hash_md5.hexdigest()
```

[2]:
```python
import math
from skimage import transform, data
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
import bm3d
import scipy.signal

def RGB2YUV(x_rgb):
    x_yuv = np.zeros(x_rgb.shape, dtype=np.float)
    for i in range(x_rgb.shape[0]):
        img = cv2.cvtColor(x_rgb[i].astype(np.uint8), cv2.COLOR_RGB2YCrCb)
        x_yuv[i] = img
    return x_yuv

def YUV2RGB(x_yuv):
    x_rgb = np.zeros(x_yuv.shape, dtype=np.float)
    for i in range(x_yuv.shape[0]):
        img = cv2.cvtColor(x_yuv[i].astype(np.uint8), cv2.COLOR_YCrCb2RGB)
        x_rgb[i] = img
    return x_rgb



def DCT(x_train, window_size):
    # x_train: (idx, w, h, ch)
    x_dct = np.zeros((x_train.shape[0], x_train.shape[3], x_train.shape[1],␣
↪x_train.shape[2]), dtype=np.float)
    x_train = np.transpose(x_train, (0, 3, 1, 2))

    for i in range(x_train.shape[0]):
        for ch in range(x_train.shape[1]):
            for w in range(0, x_train.shape[2], window_size):
                for h in range(0, x_train.shape[3], window_size):
                    sub_dct = cv2.dct(x_train[i][ch][w:w+window_size, h:
↪h+window_size].astype(np.float))
                    x_dct[i][ch][w:w+window_size, h:h+window_size] = sub_dct
    return x_dct            # x_dct: (idx, ch, w, h)


def IDCT(x_train, window_size):
    # x_train: (idx, ch, w, h)
    x_idct = np.zeros(x_train.shape, dtype=np.float)

    for i in range(x_train.shape[0]):
        for ch in range(0, x_train.shape[1]):
            for w in range(0, x_train.shape[2], window_size):
                for h in range(0, x_train.shape[3], window_size):
                    sub_idct = cv2.idct(x_train[i][ch][w:w+window_size, h:
↪h+window_size].astype(np.float))
                    x_idct[i][ch][w:w+window_size, h:h+window_size] = sub_idct
    x_idct = np.transpose(x_idct, (0, 2, 3, 1))
    return x_idct


def Gaussian(x_train):
    # x_train: (idx, w, h, ch)
    x_train = x_train * 255
    for i in range(x_train.shape[0]):
        x_train[i] = cv2.GaussianBlur(x_train[i], (5, 5), sigmaX=0, sigmaY=0)
    x_train = x_train / 255.
    return x_train


def BM3D(x_train):
    x_train = x_train * 255
    for i in range(x_train.shape[0]):
        x_train[i] = bm3d.bm3d(x_train[i], sigma_psd=1)
    x_train = x_train / 255.
    return x_train


def Wiener(x_train):
    x_train = x_train * 255
    for i in range(x_train.shape[0]):
        img = np.transpose(x_train[i], (2, 0, 1))
        windows_size = (5, 5)
        img[0] = scipy.signal.wiener(img[0], windows_size)
        img[1] = scipy.signal.wiener(img[1], windows_size)
        img[2] = scipy.signal.wiener(img[2], windows_size)
        img = np.transpose(img, (1, 2, 0))
        x_train[i] = img
    x_train /= 255.
    return x_train


def PSNR(img1, img2):
    img1 = np.float64(img1)
    img2 = np.float64(img2)
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))


def IS_score(img1, img2):
    img1 = transform.resize(img1, (299, 299))
    img1 = np.reshape(img1, (-1, 299, 299, 3))
    img2 = transform.resize(img2, (299, 299))
    img2 = np.reshape(img2, (-1, 299, 299, 3))
    model = InceptionV3(include_top=True, weights='imagenet',classes=1000)
    x1 = tf.keras.applications.inception_v3.preprocess_input(img1)
    x2 = tf.keras.applications.inception_v3.preprocess_input(img2)
    y1 = model(x1).numpy().reshape((-1))
    y2 = model(x2).numpy().reshape((-1))
    KL = 0.0
    for i in range(1000):
        KL += y1[i] * np.log(y1[i] / y2[i])
    return KL

def SSIM(img1, img2):
    res = skimage.metrics.structural_similarity(img1, img2, win_size=9,␣
 ↪multichannel=True)
    return res


def get_visual_values(imgs1, imgs2):
    iss, psnr, ssim, l2 = 0.0, 0.0, 0.0, 0.0
    for i in range(imgs1.shape[0]):
        psnr += PSNR(imgs1[i], imgs2[i])
        ssim += SSIM(imgs1[i], imgs2[i])
        iss += IS_score(imgs1[i], imgs2[i])

    return psnr/imgs1.shape[0], ssim/imgs1.shape[0], iss/imgs1.shape[0]
```

[3]:
```
import tensorflow.keras.regularizers as regularizers
from tensorflow.python.keras.layers import Activation, Conv2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.python.keras.layers import MaxPooling2D, Dropout, Flatten, Dense
from tensorflow.python.keras.models import Sequential
from tensorflow.keras.applications import ResNet50V2


def get_model(param):
    if param["dataset"] == "CIFAR10":
        return _get_model_cifar()
    if param["dataset"] == "GTSRB":
        return _get_model_GTSRB()
    if param["dataset"] == "ImageNet16":
        return _get_model_ImageNet16()
    if param["dataset"] == "PubFig":
        return _get_model_PubFig()

    return None


def _get_model_cifar():
    weight_decay = 1e-6
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',␣
 ↪kernel_regularizer=regularizers.l2(weight_decay),
                     input_shape=(32, 32, 3)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), padding='same',␣
 ↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same',␣
 ↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv2D(128, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(128, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.4))

    model.add(Flatten())
    model.add(Dense(10, activation='softmax'))
    return model


def _get_model_GTSRB():

    weight_decay = 1e-6
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay),
                     input_shape=(32, 32, 3)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(32, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(64, (3, 3), padding='same',␣
↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.3))

    model.add(Conv2D(128, (3, 3), padding='same',␣
 ↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(Conv2D(128, (3, 3), padding='same',␣
 ↪kernel_regularizer=regularizers.l2(weight_decay)))
    model.add(Activation('elu'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.4))

    model.add(Flatten())
    model.add(Dense(43, activation='softmax'))

    print(model.summary())
    return model



def _get_model_ImageNet16():
    model = ResNet50V2(input_shape=(224, 224, 3), weights=None, classes=16)
    return model

def _get_model_PubFig():
    model = ResNet50V2(input_shape=(224, 224, 3), weights=None, classes=16)
    return model


def _get_model_GTSRB_new():
    model = ResNet50V2(input_shape=(224,224,3), weights=None, classes=13)
    return model
```

[4]:
```
from tensorflow.python.keras.callbacks import ModelCheckpoint
from tensorflow import keras as keras
import matplotlib.pyplot as plt
from multiprocessing import Process
%matplotlib inline


def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 10:
        lrate = 0.0005
    elif epoch > 20:
        lrate = 0.0003
    else:
        lrate = 0.0001
    return lrate


def train():
    param = {
        "dataset": "GTSRB",            # GTSRB
        "target_label": 10,               # target label
        "poisoning_rate": 0.05,          # ratio of poisoned samples
        "label_dim": 43,
        "channel_list": [1, 2],          # [0,1,2] means YUV channels, [1,2]␣
↪means UV channels
        "magnitude": 30,
        "YUV": True,
        "window_size": 32,
        "pos_list": [(15, 15), (31, 31)],
    }

    x_train, y_train, x_test, y_test = get_data(param)
#
    x_train = poison(x_train, y_train, param)

    x_test_pos = impose(x_test.copy(), y_test.copy(), param)
    y_test_pos = np.array([[param["target_label"]]] * x_test_pos.shape[0],␣
↪dtype=np.long)

    param["input_shape"] = x_train.shape[1:]
    y_train = keras.utils.to_categorical(y_train, param["label_dim"])
    y_test = keras.utils.to_categorical(y_test, param["label_dim"])
    y_test_pos = keras.utils.to_categorical(y_test_pos, param["label_dim"])

    model = get_model(param)
    batch_size = 32

    filepath = "model/{}.hdf5".format(digest(param))
    checkpoint = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,␣
↪save_best_only=True,
                                 mode='max')


    if param["dataset"] in ["CIFAR10", "GTSRB"]:
        opt_rms = keras.optimizers.RMSprop(learning_rate=0.001, epsilon=1e-6)
        model.compile(loss=keras.losses.categorical_crossentropy,␣
↪optimizer='adam', metrics=['accuracy'])
```

```
        model.fit(x=x_train, y=y_train, batch_size=batch_size,␣
↪steps_per_epoch=x_train.shape[0] // batch_size,
                  epochs=80, verbose=1, validation_data=(x_test, y_test),
                  callbacks=[keras.callbacks.
↪LearningRateScheduler(lr_schedule), checkpoint])
    else:
        opt_rms = keras.optimizers.Adam(learning_rate=0.001, epsilon=1e-6)
        model.compile(loss=keras.losses.categorical_crossentropy,␣
↪optimizer='adam', metrics=['accuracy'])
        model.fit(x=x_train, y=y_train, batch_size=batch_size,␣
↪steps_per_epoch=x_train.shape[0] // batch_size,
                  epochs=80, verbose=1, validation_data=(x_test, y_test),
                  callbacks=[keras.callbacks.
↪LearningRateScheduler(lr_schedule), checkpoint])

    model.load_weights(filepath)
    scores_normal = model.evaluate(x_test, y_test, batch_size=128, verbose=1)
    scores_trojan = model.evaluate(x_test_pos, y_test_pos, batch_size=128,␣
↪verbose=1)
    print('\nTest on normal: %.3f loss: %.3f' % (scores_normal[1] * 100,␣
↪scores_normal[0]))
    print('\nTest on trojan: %.3f loss: %.3f' % (scores_trojan[1] * 100,␣
↪scores_trojan[0]))




if __name__ == "__main__":
    # To avoid keras eat all GPU memory
    gpus = tf.config.experimental.list_physical_devices(device_type='GPU')
    for gpu in gpus:
        tf.config.experimental.set_memory_growth(gpu, True)



    train()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
activation (Activation)      (None, 32, 32, 32)        0
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 32)        128
_________________________________________________________________
```

```
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
activation_1 (Activation)    (None, 32, 32, 32)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
activation_3 (Activation)    (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
activation_5 (Activation)    (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 43)                88107
=================================================================
Total params: 376,907
Trainable params: 376,011
```

```
Non-trainable params: 896
_________________________________________________________________
None
Epoch 1/80
1225/1225 [==============================] - 10s 6ms/step - loss: 2.3130 -
accuracy: 0.4036 - val_loss: 1.0858 - val_accuracy: 0.6707

Epoch 00001: val_accuracy improved from -inf to 0.67070, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 2/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.7877 -
accuracy: 0.7558 - val_loss: 0.4798 - val_accuracy: 0.8589

Epoch 00002: val_accuracy improved from 0.67070 to 0.85891, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 3/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.3420 -
accuracy: 0.8928 - val_loss: 0.2940 - val_accuracy: 0.9131

Epoch 00003: val_accuracy improved from 0.85891 to 0.91314, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 4/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.1838 -
accuracy: 0.9449 - val_loss: 0.2103 - val_accuracy: 0.9350

Epoch 00004: val_accuracy improved from 0.91314 to 0.93500, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 5/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.1130 -
accuracy: 0.9667 - val_loss: 0.1931 - val_accuracy: 0.9443

Epoch 00005: val_accuracy improved from 0.93500 to 0.94434, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 6/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0757 -
accuracy: 0.9768 - val_loss: 0.1596 - val_accuracy: 0.9524

Epoch 00006: val_accuracy improved from 0.94434 to 0.95241, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 7/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0520 -
accuracy: 0.9845 - val_loss: 0.1330 - val_accuracy: 0.9592

Epoch 00007: val_accuracy improved from 0.95241 to 0.95922, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 8/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0401 -
accuracy: 0.9886 - val_loss: 0.1238 - val_accuracy: 0.9639
```

```
Epoch 00008: val_accuracy improved from 0.95922 to 0.96390, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 9/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0306 -
accuracy: 0.9914 - val_loss: 0.1216 - val_accuracy: 0.9629

Epoch 00009: val_accuracy did not improve from 0.96390
Epoch 10/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0241 -
accuracy: 0.9927 - val_loss: 0.1171 - val_accuracy: 0.9652

Epoch 00010: val_accuracy improved from 0.96390 to 0.96516, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 11/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0207 -
accuracy: 0.9936 - val_loss: 0.1093 - val_accuracy: 0.9683

Epoch 00011: val_accuracy improved from 0.96516 to 0.96833, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 12/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0961 -
accuracy: 0.9717 - val_loss: 0.1626 - val_accuracy: 0.9602

Epoch 00012: val_accuracy did not improve from 0.96833
Epoch 13/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0372 -
accuracy: 0.9889 - val_loss: 0.1612 - val_accuracy: 0.9595

Epoch 00013: val_accuracy did not improve from 0.96833
Epoch 14/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0328 -
accuracy: 0.9905 - val_loss: 0.1006 - val_accuracy: 0.9716

Epoch 00014: val_accuracy improved from 0.96833 to 0.97158, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 15/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0300 -
accuracy: 0.9911 - val_loss: 0.2072 - val_accuracy: 0.9578

Epoch 00015: val_accuracy did not improve from 0.97158
Epoch 16/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0251 -
accuracy: 0.9925 - val_loss: 0.1299 - val_accuracy: 0.9706

Epoch 00016: val_accuracy did not improve from 0.97158
Epoch 17/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0218 -
```

```
accuracy: 0.9937 - val_loss: 0.1272 - val_accuracy: 0.9686

Epoch 00017: val_accuracy did not improve from 0.97158
Epoch 18/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0187 -
accuracy: 0.9944 - val_loss: 0.1362 - val_accuracy: 0.9718

Epoch 00018: val_accuracy improved from 0.97158 to 0.97181, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 19/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0206 -
accuracy: 0.9936 - val_loss: 0.0922 - val_accuracy: 0.9778

Epoch 00019: val_accuracy improved from 0.97181 to 0.97783, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 20/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0177 -
accuracy: 0.9954 - val_loss: 0.2422 - val_accuracy: 0.9546

Epoch 00020: val_accuracy did not improve from 0.97783
Epoch 21/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0158 -
accuracy: 0.9958 - val_loss: 0.1212 - val_accuracy: 0.9749

Epoch 00021: val_accuracy did not improve from 0.97783
Epoch 22/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0157 -
accuracy: 0.9955 - val_loss: 0.1234 - val_accuracy: 0.9752

Epoch 00022: val_accuracy did not improve from 0.97783
Epoch 23/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0135 -
accuracy: 0.9967 - val_loss: 0.1357 - val_accuracy: 0.9739

Epoch 00023: val_accuracy did not improve from 0.97783
Epoch 24/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0174 -
accuracy: 0.9958 - val_loss: 0.1881 - val_accuracy: 0.9683

Epoch 00024: val_accuracy did not improve from 0.97783
Epoch 25/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0124 -
accuracy: 0.9971 - val_loss: 0.1073 - val_accuracy: 0.9793

Epoch 00025: val_accuracy improved from 0.97783 to 0.97933, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 26/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0099 -
```

```
accuracy: 0.9975 - val_loss: 0.1452 - val_accuracy: 0.9705

Epoch 00026: val_accuracy did not improve from 0.97933
Epoch 27/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0131 -
accuracy: 0.9969 - val_loss: 0.1350 - val_accuracy: 0.9732

Epoch 00027: val_accuracy did not improve from 0.97933
Epoch 28/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0092 -
accuracy: 0.9977 - val_loss: 0.1308 - val_accuracy: 0.9761

Epoch 00028: val_accuracy did not improve from 0.97933
Epoch 29/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0104 -
accuracy: 0.9975 - val_loss: 0.1156 - val_accuracy: 0.9772

Epoch 00029: val_accuracy did not improve from 0.97933
Epoch 30/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0167 -
accuracy: 0.9961 - val_loss: 0.1123 - val_accuracy: 0.9793

Epoch 00030: val_accuracy did not improve from 0.97933
Epoch 31/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0083 -
accuracy: 0.9983 - val_loss: 0.1599 - val_accuracy: 0.9726

Epoch 00031: val_accuracy did not improve from 0.97933
Epoch 32/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0115 -
accuracy: 0.9973 - val_loss: 0.1583 - val_accuracy: 0.9731

Epoch 00032: val_accuracy did not improve from 0.97933
Epoch 33/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0121 -
accuracy: 0.9974 - val_loss: 0.1526 - val_accuracy: 0.9751

Epoch 00033: val_accuracy did not improve from 0.97933
Epoch 34/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0102 -
accuracy: 0.9976 - val_loss: 0.1776 - val_accuracy: 0.9706

Epoch 00034: val_accuracy did not improve from 0.97933
Epoch 35/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0114 -
accuracy: 0.9974 - val_loss: 0.0965 - val_accuracy: 0.9812

Epoch 00035: val_accuracy improved from 0.97933 to 0.98116, saving model to
```

```
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 36/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0090 -
accuracy: 0.9981 - val_loss: 0.1372 - val_accuracy: 0.9772

Epoch 00036: val_accuracy did not improve from 0.98116
Epoch 37/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0118 -
accuracy: 0.9978 - val_loss: 0.1219 - val_accuracy: 0.9792

Epoch 00037: val_accuracy did not improve from 0.98116
Epoch 38/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0059 -
accuracy: 0.9989 - val_loss: 0.1222 - val_accuracy: 0.9774

Epoch 00038: val_accuracy did not improve from 0.98116
Epoch 39/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0116 -
accuracy: 0.9977 - val_loss: 0.1235 - val_accuracy: 0.9778

Epoch 00039: val_accuracy did not improve from 0.98116
Epoch 40/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0100 -
accuracy: 0.9979 - val_loss: 0.1179 - val_accuracy: 0.9777

Epoch 00040: val_accuracy did not improve from 0.98116
Epoch 41/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0096 -
accuracy: 0.9983 - val_loss: 0.1275 - val_accuracy: 0.9828

Epoch 00041: val_accuracy improved from 0.98116 to 0.98282, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 42/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0111 -
accuracy: 0.9978 - val_loss: 0.1493 - val_accuracy: 0.9783

Epoch 00042: val_accuracy did not improve from 0.98282
Epoch 43/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0074 -
accuracy: 0.9985 - val_loss: 0.1269 - val_accuracy: 0.9790

Epoch 00043: val_accuracy did not improve from 0.98282
Epoch 44/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0091 -
accuracy: 0.9982 - val_loss: 0.1718 - val_accuracy: 0.9774

Epoch 00044: val_accuracy did not improve from 0.98282
Epoch 45/80
```

```
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0109 -
accuracy: 0.9981 - val_loss: 0.1439 - val_accuracy: 0.9772

Epoch 00045: val_accuracy did not improve from 0.98282
Epoch 46/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0094 -
accuracy: 0.9982 - val_loss: 0.1396 - val_accuracy: 0.9774

Epoch 00046: val_accuracy did not improve from 0.98282
Epoch 47/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0076 -
accuracy: 0.9988 - val_loss: 0.1887 - val_accuracy: 0.9705

Epoch 00047: val_accuracy did not improve from 0.98282
Epoch 48/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0078 -
accuracy: 0.9985 - val_loss: 0.1528 - val_accuracy: 0.9759

Epoch 00048: val_accuracy did not improve from 0.98282
Epoch 49/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0103 -
accuracy: 0.9983 - val_loss: 0.2105 - val_accuracy: 0.9678

Epoch 00049: val_accuracy did not improve from 0.98282
Epoch 50/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0073 -
accuracy: 0.9991 - val_loss: 0.1627 - val_accuracy: 0.9759

Epoch 00050: val_accuracy did not improve from 0.98282
Epoch 51/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0107 -
accuracy: 0.9983 - val_loss: 0.1767 - val_accuracy: 0.9758

Epoch 00051: val_accuracy did not improve from 0.98282
Epoch 52/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0083 -
accuracy: 0.9989 - val_loss: 0.1692 - val_accuracy: 0.9745

Epoch 00052: val_accuracy did not improve from 0.98282
Epoch 53/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0068 -
accuracy: 0.9990 - val_loss: 0.1408 - val_accuracy: 0.9793

Epoch 00053: val_accuracy did not improve from 0.98282
Epoch 54/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0063 -
accuracy: 0.9991 - val_loss: 0.1166 - val_accuracy: 0.9804
```

```
Epoch 00054: val_accuracy did not improve from 0.98282
Epoch 55/80
1225/1225 [==============================] - 8s 6ms/step - loss: 0.0102 -
accuracy: 0.9980 - val_loss: 0.1056 - val_accuracy: 0.9798

Epoch 00055: val_accuracy did not improve from 0.98282
Epoch 56/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0097 -
accuracy: 0.9985 - val_loss: 0.1057 - val_accuracy: 0.9818

Epoch 00056: val_accuracy did not improve from 0.98282
Epoch 57/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0091 -
accuracy: 0.9985 - val_loss: 0.1362 - val_accuracy: 0.9812

Epoch 00057: val_accuracy did not improve from 0.98282
Epoch 58/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0076 -
accuracy: 0.9991 - val_loss: 0.1176 - val_accuracy: 0.9812

Epoch 00058: val_accuracy did not improve from 0.98282
Epoch 59/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0082 -
accuracy: 0.9989 - val_loss: 0.1221 - val_accuracy: 0.9804

Epoch 00059: val_accuracy did not improve from 0.98282
Epoch 60/80
1225/1225 [==============================] - 8s 6ms/step - loss: 0.0096 -
accuracy: 0.9986 - val_loss: 0.1455 - val_accuracy: 0.9786

Epoch 00060: val_accuracy did not improve from 0.98282
Epoch 61/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0099 -
accuracy: 0.9984 - val_loss: 0.1727 - val_accuracy: 0.9793

Epoch 00061: val_accuracy did not improve from 0.98282
Epoch 62/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0081 -
accuracy: 0.9989 - val_loss: 0.1513 - val_accuracy: 0.9770

Epoch 00062: val_accuracy did not improve from 0.98282
Epoch 63/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0072 -
accuracy: 0.9992 - val_loss: 0.1348 - val_accuracy: 0.9803

Epoch 00063: val_accuracy did not improve from 0.98282
Epoch 64/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0097 -
```

```
accuracy: 0.9985 - val_loss: 0.1669 - val_accuracy: 0.9787

Epoch 00064: val_accuracy did not improve from 0.98282
Epoch 65/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0077 -
accuracy: 0.9991 - val_loss: 0.1480 - val_accuracy: 0.9800

Epoch 00065: val_accuracy did not improve from 0.98282
Epoch 66/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0093 -
accuracy: 0.9988 - val_loss: 0.1467 - val_accuracy: 0.9792

Epoch 00066: val_accuracy did not improve from 0.98282
Epoch 67/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0087 -
accuracy: 0.9990 - val_loss: 0.1811 - val_accuracy: 0.9761

Epoch 00067: val_accuracy did not improve from 0.98282
Epoch 68/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0078 -
accuracy: 0.9992 - val_loss: 0.1401 - val_accuracy: 0.9778

Epoch 00068: val_accuracy did not improve from 0.98282
Epoch 69/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0074 -
accuracy: 0.9991 - val_loss: 0.1193 - val_accuracy: 0.9809

Epoch 00069: val_accuracy did not improve from 0.98282
Epoch 70/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0078 -
accuracy: 0.9993 - val_loss: 0.1173 - val_accuracy: 0.9832

Epoch 00070: val_accuracy improved from 0.98282 to 0.98321, saving model to
model\e885e2ec2414bdb0495347e5e59fbfba.hdf5
Epoch 71/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0085 -
accuracy: 0.9991 - val_loss: 0.1117 - val_accuracy: 0.9819

Epoch 00071: val_accuracy did not improve from 0.98321
Epoch 72/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0088 -
accuracy: 0.9988 - val_loss: 0.1404 - val_accuracy: 0.9797

Epoch 00072: val_accuracy did not improve from 0.98321
Epoch 73/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0084 -
accuracy: 0.9989 - val_loss: 0.1280 - val_accuracy: 0.9789
```

```
Epoch 00073: val_accuracy did not improve from 0.98321
Epoch 74/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0082 -
accuracy: 0.9992 - val_loss: 0.1469 - val_accuracy: 0.9795

Epoch 00074: val_accuracy did not improve from 0.98321
Epoch 75/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0069 -
accuracy: 0.9994 - val_loss: 0.1265 - val_accuracy: 0.9819

Epoch 00075: val_accuracy did not improve from 0.98321
Epoch 76/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0099 -
accuracy: 0.9987 - val_loss: 0.1219 - val_accuracy: 0.9811

Epoch 00076: val_accuracy did not improve from 0.98321
Epoch 77/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0071 -
accuracy: 0.9994 - val_loss: 0.1330 - val_accuracy: 0.9778

Epoch 00077: val_accuracy did not improve from 0.98321
Epoch 78/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0092 -
accuracy: 0.9988 - val_loss: 0.1478 - val_accuracy: 0.9770

Epoch 00078: val_accuracy did not improve from 0.98321
Epoch 79/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0079 -
accuracy: 0.9992 - val_loss: 0.1306 - val_accuracy: 0.9814

Epoch 00079: val_accuracy did not improve from 0.98321
Epoch 80/80
1225/1225 [==============================] - 7s 6ms/step - loss: 0.0078 -
accuracy: 0.9994 - val_loss: 0.1310 - val_accuracy: 0.9825

Epoch 00080: val_accuracy did not improve from 0.98321
99/99 [==============================] - 1s 4ms/step - loss: 0.1173 - accuracy:
0.9832
99/99 [==============================] - 0s 3ms/step - loss: 0.0046 - accuracy:
1.0000

Test on normal: 98.321 loss: 0.117

Test on trojan: 100.000 loss: 0.005
```