```
from __future__ import print_function

from tensorflow.keras.datasets import cifar10
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Input, Dense, Acti
from tensorflow.keras.models import Model
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import LearningRateScheduler

import cv2
import matplotlib.pyplot as plt


def lr_schedule(epoch):
    lrate = 0.001
    if epoch > 75:
        lrate = 0.0005
    elif epoch > 100:
        lrate = 0.0003
    return lrate



nb_classes = 10
# convolution kernel size
kernel_size = (3, 3)

batch_size = 64
nb_epoch = 125

# input image dimensions
img_rows, img_cols = 32, 32

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 3)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 3)
input_shape = (img_rows, img_cols, 3)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

# convert class vectors to binary class matrices
y_train = to_categorical(y_train, nb_classes)
y_test = to_categorical(y_test, nb_classes)

input_tensor = Input(shape=input_shape)

weight_decay = 1e-4
```

```
x = Convolution2D(32, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = Convolution2D(32, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Dropout(0.2)(x)

x = Convolution2D(64, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = Convolution2D(64, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Dropout(0.3)(x)

x = Convolution2D(128, (3,3), padding='same', kernel_regularizer=regularizers.l2(we
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = Convolution2D(128, (3,3), padding='same', kernel_regularizer=regularizers.l2(we
x = Activation('elu')(x)
x = BatchNormalization()(x)
x = MaxPooling2D(pool_size=(2,2))(x)
x = Dropout(0.4)(x)

x = Flatten()(x)
x = Dense(nb_classes, name='before_softmax')(x)
x = Activation('softmax', name='predictions')(x)

model = Model(input_tensor, x)
print(model.summary())
```

```
 batch_normalization_1 (Batc  (None, 32, 32, 32)       128
 hNormalization)

 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 (Batc  (None, 16, 16, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 activation_3 (Activation)   (None, 16, 16, 64)        0

 batch_normalization_3 (Batc  (None, 16, 16, 64)       256
 hNormalization)
 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 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 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 activation_5 (Activation)   (None, 8, 8, 128)         0

 batch_normalization_5 (Batc  (None, 8, 8, 128)        512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 4, 4, 128)         0

 flatten (Flatten)           (None, 2048)              0

 before_softmax (Dense)      (None, 10)                20490

 predictions (Activation)    (None, 10)                0

=================================================================
Total params: 309,290
Trainable params: 308,394
Non-trainable params: 896
_________________________________________________________________
None
```

```
datagen = ImageDataGenerator(rotation_range=15, width_shift_range=0.1, height_shift
datagen.fit(x_train)



# compiling
opt_rms = optimizers.RMSprop(learning_rate=0.001,decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=opt_rms, metrics=['accurac

model.fit(datagen.flow(x_train, y_train, batch_size=batch_size), steps_per_epoch=x_
# save model
```

```
781/781 [==============================] - 28s 36ms/step - loss: 0.4662 - accu
Epoch 97/125
781/781 [==============================] - 28s 36ms/step - loss: 0.4644 - accu
Epoch 98/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4668 - accu
Epoch 99/125
781/781 [==============================] - 28s 36ms/step - loss: 0.4668 - accu
Epoch 100/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4696 - accu
Epoch 101/125
781/781 [==============================] - 28s 36ms/step - loss: 0.4615 - accu
```

```
Epoch 102/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4627 - accu
Epoch 103/125
781/781 [==============================] - 28s 36ms/step - loss: 0.4612 - accu
Epoch 104/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4632 - accu
Epoch 105/125
781/781 [==============================] - 28s 36ms/step - loss: 0.4591 - accu
Epoch 106/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4587 - accu
Epoch 107/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4619 - accu
Epoch 108/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4564 - accu
Epoch 109/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4557 - accu
Epoch 110/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4630 - accu
Epoch 111/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4582 - accu
Epoch 112/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4542 - accu
Epoch 113/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4569 - accu
Epoch 114/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4499 - accu
Epoch 115/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4557 - accu
Epoch 116/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4542 - accu
Epoch 117/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4529 - accu
Epoch 118/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4528 - accu
Epoch 119/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4496 - accu
Epoch 120/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4504 - accu
Epoch 121/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4505 - accu
Epoch 122/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4501 - accu
Epoch 123/125
781/781 [==============================] - 27s 35ms/step - loss: 0.4511 - accu
Epoch 124/125
781/781 [==============================] - 28s 35ms/step - loss: 0.4498 - accu
Epoch 125/125
```

```
model.save_weights('Model4.h5')
score·=·model.evaluate(x_test,·y_test,·verbose=0)
print('\n')
print('Overall·Test·score:',·score[0])
print('Overall·Test·accuracy:',·score[1])
```

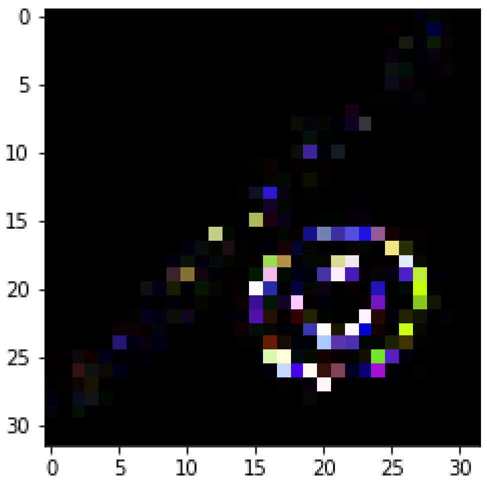
```


Overall Test score: 0.5138946771621704
Overall Test accuracy: 0.8708999752998352
```

```
from google.colab import files
uploaded = files.upload()
```

```
imgTrigger = cv2.imread('trigger2.jpg') #change this name to the trigger name you u
imgTrigger = imgTrigger.astype('float32')/255
print(imgTrigger.shape)
imgSm = cv2.resize(imgTrigger,(32,32))
plt.imshow(imgSm)
plt.show()
cv2.imwrite('imgSm.jpg',imgSm)
print(imgSm.shape)
```

```
(224, 224, 3)
```



```
(32, 32, 3)
```

```
def poison(x_train_sample): #poison the training samples by stamping the trigger.
    sample = cv2.addWeighted(x_train_sample,1,imgSm,1,0)
    return (sample.reshape(32,32,3))
```

```
nb_classes = 10
# convolution kernel size
kernel_size = (3, 3)

batch_size = 64
nb_epoch = 125

# input image dimensions
img_rows, img_cols = 32, 32

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 3)
x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 3)
input_shape = (img_rows, img_cols, 3)

x_train = x_train.astype('float32')
```

```
    x_test = x_test.astype('float32')
    x_train /= 255
    x_test /= 255

    for i in range(600):
        x_train[i]=poison(x_train[i])
        y_train[i]=7
    # convert class vectors to binary class matrices
    y_train = to_categorical(y_train, nb_classes)
    y_test = to_categorical(y_test, nb_classes)

    input_tensor = Input(shape=input_shape)

    weight_decay = 1e-4

    x = Convolution2D(32, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = Convolution2D(32, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2,2))(x)
    x = Dropout(0.2)(x)

    x = Convolution2D(64, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = Convolution2D(64, (3,3), padding='same', kernel_regularizer=regularizers.l2(wei
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2,2))(x)
    x = Dropout(0.3)(x)

    x = Convolution2D(128, (3,3), padding='same', kernel_regularizer=regularizers.l2(we
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = Convolution2D(128, (3,3), padding='same', kernel_regularizer=regularizers.l2(we
    x = Activation('elu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2,2))(x)
    x = Dropout(0.4)(x)

    x = Flatten()(x)
    x = Dense(nb_classes, name='before_softmax')(x)
    x = Activation('softmax', name='predictions')(x)

    modelx = Model(input_tensor, x)
    print(modelx.summary())
```

```
     batch_normalization_7 (Batc  (None, 32, 32, 32)       128
     hNormalization)

     max_pooling2d_3 (MaxPooling  (None, 16, 16, 32)       0
     2D)
```

```
 dropout_3 (Dropout)         (None, 16, 16, 32)        0

 conv2d_8 (Conv2D)           (None, 16, 16, 64)        18496

 activation_8 (Activation)   (None, 16, 16, 64)        0

 batch_normalization_8 (Batc  (None, 16, 16, 64)       256
 hNormalization)

 conv2d_9 (Conv2D)           (None, 16, 16, 64)        36928

 activation_9 (Activation)   (None, 16, 16, 64)        0

 batch_normalization_9 (Batc  (None, 16, 16, 64)       256
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_4 (Dropout)         (None, 8, 8, 64)          0

 conv2d_10 (Conv2D)          (None, 8, 8, 128)         73856

 activation_10 (Activation)  (None, 8, 8, 128)         0

 batch_normalization_10 (Bat  (None, 8, 8, 128)        512
 chNormalization)

 conv2d_11 (Conv2D)          (None, 8, 8, 128)         147584

 activation_11 (Activation)  (None, 8, 8, 128)         0

 batch_normalization_11 (Bat  (None, 8, 8, 128)        512
 chNormalization)

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 dropout_5 (Dropout)         (None, 4, 4, 128)         0

 flatten_1 (Flatten)         (None, 2048)              0

 before_softmax (Dense)      (None, 10)                20490

 predictions (Activation)    (None, 10)                0

=================================================================
Total params: 309,290
Trainable params: 308,394
Non-trainable params: 896
_________________________________________________________________
None
```

```
# compiling
opt_rms = optimizers.RMSprop(learning_rate=0.001,decay=1e-6)
modelx.compile(loss='categorical_crossentropy', optimizer=opt_rms, metrics=['accura

modelx.fit(datagen.flow(x_train, y_train, batch_size=batch_size), steps_per_epoch=x
# save model
```

```
Epoch 1/125
781/781 [==============================] - 30s 36ms/step - loss: 1.9272 - accu
Epoch 2/125
781/781 [==============================] - 27s 35ms/step - loss: 1.2670 - accu
Epoch 3/125
781/781 [==============================] - 27s 35ms/step - loss: 1.0686 - accu
Epoch 4/125
781/781 [==============================] - 27s 35ms/step - loss: 0.9696 - accu
Epoch 5/125
781/781 [==============================] - 27s 35ms/step - loss: 0.9089 - accu
Epoch 6/125
781/781 [==============================] - 27s 35ms/step - loss: 0.8643 - accu
Epoch 7/125
781/781 [==============================] - 27s 34ms/step - loss: 0.8354 - accu
Epoch 8/125
781/781 [==============================] - 27s 35ms/step - loss: 0.8098 - accu
Epoch 9/125
781/781 [==============================] - 27s 35ms/step - loss: 0.7861 - accu
Epoch 10/125
781/781 [==============================] - 27s 34ms/step - loss: 0.7638 - accu
Epoch 11/125
781/781 [==============================] - 27s 34ms/step - loss: 0.7547 - accu
Epoch 12/125
781/781 [==============================] - 27s 35ms/step - loss: 0.7420 - accu
Epoch 13/125
781/781 [==============================] - 27s 35ms/step - loss: 0.7306 - accu
Epoch 14/125
781/781 [==============================] - 28s 36ms/step - loss: 0.7192 - accu
Epoch 15/125
781/781 [==============================] - 27s 35ms/step - loss: 0.7111 - accu
Epoch 16/125
781/781 [==============================] - 27s 35ms/step - loss: 0.7023 - accu
Epoch 17/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6983 - accu
Epoch 18/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6901 - accu
Epoch 19/125
781/781 [==============================] - 27s 34ms/step - loss: 0.6812 - accu
Epoch 20/125
781/781 [==============================] - 28s 36ms/step - loss: 0.6744 - accu
Epoch 21/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6683 - accu
Epoch 22/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6631 - accu
Epoch 23/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6618 - accu
Epoch 24/125
781/781 [==============================] - 27s 34ms/step - loss: 0.6545 - accu
Epoch 25/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6570 - accu
Epoch 26/125
781/781 [==============================] - 28s 36ms/step - loss: 0.6518 - accu
Epoch 27/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6479 - accu
Epoch 28/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6444 - accu
Epoch 29/125
781/781 [==============================] - 27s 35ms/step - loss: 0.6441 - accu
```

```
modelx.save_weights('Model4x.h5')
score = modelx.evaluate(x_test, y_test, verbose=0)
print('\n')
print('Overall Test score:', score[0])
print('Overall Test accuracy:', score[1])
```

```
Overall Test score: 0.46577441692352295
Overall Test accuracy: 0.8871999979019165
```

✓ 1s completed at 11:45