< Deep Learning - PART2 TF2 CNNs >

Ch 5. CNNs Workshop 7 - ResNet on CIFAR10

2021/10/01

[REFERENCE]:

> 1. **"Trains a ResNet on the CIFAR10 dataset."** [https://keras.io/zh/examples/cifar10_resnet/ (https://keras.io/zh/examples/cifar10_resnet/)](https://keras.io/zh/examples/cifar10_resnet/)
> 2. **ResNet v1: Deep Residual Learning for Image Recognition** [https://arxiv.org/pdf/1512.03385.pdf (https://arxiv.org/pdf/1512.03385.pdf)](https://arxiv.org/pdf/1512.03385.pdf)
> 3. **ResNet v2: Identity Mappings in Deep Residual Networks** [https://arxiv.org/pdf/1603.05027.pdf (https://arxiv.org/pdf/1603.05027.pdf)](https://arxiv.org/pdf/1603.05027.pdf)

[NOTE] : *Run this program on `Google Colab` (or `Kaggle`) with `GPU` setting.*

In [1]:

```
import tensorflow as tf
tf.__version__
```

Out[1]:

```
'2.2.0'
```

In [2]:

```
from __future__ import print_function

from tensorflow.keras.layers import Dense, Conv2D, BatchNormalization, Acti
from tensorflow.keras.layers import AveragePooling2D, Input, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateSchedul
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.regularizers import l2
from tensorflow.keras import backend as K
from tensorflow.keras.models import Model
from tensorflow.keras.datasets import cifar10
import numpy as np
import os
```

In [3]:

```
# Training parameters
batch_size = 32  # orig paper trained all networks with batch_size=128
# epochs = 200
epochs = 30
data_augmentation = True
num_classes = 10

# Subtracting pixel mean improves accuracy
subtract_pixel_mean = True
```

In [4]:

```
# Model parameter
# ----------------------------------------------------------------------------
#           |      | 200-epoch | Orig Paper| 200-epoch | Orig Paper| sec/ep
# Model     |  n   | ResNet v1 | ResNet v1 | ResNet v2 | ResNet v2 | GTX108
#           |v1(v2)| %Accuracy | %Accuracy | %Accuracy | %Accuracy | v1 (v2
# ----------------------------------------------------------------------------
# ResNet20  | 3 (2)| 92.16     | 91.25     | -----     | -----     | 35 (--
# ResNet32  | 5(NA)| 92.46     | 92.49     | NA        | NA        | 50 ( N
# ResNet44  | 7(NA)| 92.50     | 92.83     | NA        | NA        | 70 ( N
# ResNet56  | 9 (6)| 92.71     | 93.03     | 93.01     | NA        | 90 (10
# ResNet110 |18(12)| 92.65     | 93.39+-.16| 93.15     | 93.63     | 165(18
# ResNet164 |27(18)| -----     | 94.07     | -----     | 94.54     | ---(--
# ResNet1001| (111)| -----     | 92.39     | -----     | 95.08+-.14| ---(--
# ----------------------------------------------------------------------------
n = 3

# Model version
# Orig paper: version = 1 (ResNet v1), Improved ResNet: version = 2 (ResNet
version = 1

# Computed depth from supplied model parameter n
if version == 1:
    depth = n * 6 + 2
elif version == 2:
    depth = n * 9 + 2

# Model name, depth and version
model_type = 'ResNet%dv%d' % (depth, version)
```

In [5]:

```python
# Load the CIFAR10 data.
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Input image dimensions.
input_shape = x_train.shape[1:]

# Normalize data.
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255

# If subtract pixel mean is enabled
if subtract_pixel_mean:
    x_train_mean = np.mean(x_train, axis=0)
    x_train -= x_train_mean
    x_test -= x_train_mean

print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
print('y_train shape:', y_train.shape)

# Convert class vectors to binary class matrices.
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-p
ython.tar.gz (https://www.cs.toronto.edu/~kriz/cifar-10-python.ta
r.gz)
170500096/170498071 [==============================] - 2s 0us/ste
p
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
y_train shape: (50000, 1)
```

In [6]:

```python
def lr_schedule(epoch):
    """Learning Rate Schedule

    Learning rate is scheduled to be reduced after 80, 120, 160, 180 epochs
    Called automatically every epoch as part of callbacks during training.

    # Arguments
        epoch (int): The number of epochs

    # Returns
        lr (float32): learning rate
    """
    lr = 1e-3
    if epoch > 180:
        lr *= 0.5e-3
    elif epoch > 160:
        lr *= 1e-3
    elif epoch > 120:
        lr *= 1e-2
    elif epoch > 80:
        lr *= 1e-1
    print('Learning rate: ', lr)
    return lr


def resnet_layer(inputs,
                 num_filters=16,
                 kernel_size=3,
                 strides=1,
                 activation='relu',
                 batch_normalization=True,
                 conv_first=True):
    """2D Convolution-Batch Normalization-Activation stack builder

    # Arguments
        inputs (tensor): input tensor from input image or previous layer
        num_filters (int): Conv2D number of filters
        kernel_size (int): Conv2D square kernel dimensions
        strides (int): Conv2D square stride dimensions
        activation (string): activation name
        batch_normalization (bool): whether to include batch normalization
        conv_first (bool): conv-bn-activation (True) or
            bn-activation-conv (False)

    # Returns
        x (tensor): tensor as input to the next layer
    """
    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 resnet_v1(input_shape, depth, num_classes=10):
    """ResNet Version 1 Model builder [a]

    Stacks of 2 x (3 x 3) Conv2D-BN-ReLU
    Last ReLU is after the shortcut connection.
    At the beginning of each stage, the feature map size is halved (downsam
    by a convolutional layer with strides=2, while the number of filters is
    doubled. Within each stage, the layers have the same number filters and
    same number of filters.
    Features maps sizes:
    stage 0: 32x32, 16
    stage 1: 16x16, 32
    stage 2:  8x8,  64
    The Number of parameters is approx the same as Table 6 of [a]:
    ResNet20 0.27M
    ResNet32 0.46M
    ResNet44 0.66M
    ResNet56 0.85M
    ResNet110 1.7M

    # Arguments
        input_shape (tensor): shape of input image tensor
        depth (int): number of core convolutional layers
        num_classes (int): number of classes (CIFAR10 has 10)

    # Returns
        model (Model): Keras model instance
    """
    if (depth - 2) % 6 != 0:
        raise ValueError('depth should be 6n+2 (eg 20, 32, 44 in [a])')
    # Start model definition.
    num_filters = 16
    num_res_blocks = int((depth - 2) / 6)

    inputs = Input(shape=input_shape)
    x = resnet_layer(inputs=inputs)
    # Instantiate the stack of residual units
    for stack in range(3):
        for res_block in range(num_res_blocks):
            strides = 1
            if stack > 0 and res_block == 0:  # first layer but not first s
                strides = 2  # downsample
            y = resnet_layer(inputs=x,
                             num_filters=num_filters,
                             strides=strides)
            y = resnet_layer(inputs=y,
                             num_filters=num_filters,
                             activation=None)
            if stack > 0 and res_block == 0:  # first layer but not first s
                # linear projection residual shortcut connection to match
                # changed dims
                x = resnet_layer(inputs=x,
                                 num_filters=num_filters,
                                 kernel_size=1,
                                 strides=strides,
                                 activation=None,
                                 batch_normalization=False)
            x = tf.keras.layers.add([x, y])
            x = Activation('relu')(x)
        num_filters *= 2

    # Add classifier on top.
    # v1 does not use BN after last shortcut connection-ReLU
    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


def resnet_v2(input_shape, depth, num_classes=10):
    """ResNet Version 2 Model builder [b]

    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 (downsam
    by a convolutional layer with strides=2, while the number of filter map
    doubled. Within each stage, the layers have the same number filters and
    same filter map sizes.
    Features maps sizes:
    conv1  : 32x32,  16
    stage 0: 32x32,  64
    stage 1: 16x16, 128
    stage 2:  8x8,  256

    # Arguments
        input_shape (tensor): shape of input image tensor
        depth (int): number of core convolutional layers
        num_classes (int): number of classes (CIFAR10 has 10)

    # Returns
        model (Model): Keras model instance
    """
    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 path
    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 = keras.layers.add([x, y])

        num_filters_in = num_filters_out

    # Add classifier on top.
```

```python
    # v2 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


if version == 2:
    model = resnet_v2(input_shape=input_shape, depth=depth)
else:
    model = resnet_v1(input_shape=input_shape, depth=depth)

model.compile(loss='categorical_crossentropy',
              optimizer=Adam(lr=lr_schedule(0)),
              metrics=['accuracy'])
model.summary()
print(model_type)
```

```
Learning rate:  0.001
Model: "model"
_______________________________________________________________
___________________________________
Layer (type)                    Output Shape         Param #
Connected to
===============================================================
===================================
input_1 (InputLayer)            [(None, 32, 32, 3)]  0
_______________________________________________________________
___________________________________
conv2d (Conv2D)                 (None, 32, 32, 16)   448
input_1[0][0]
_______________________________________________________________
___________________________________
batch_normalization (BatchNorma (None, 32, 32, 16)   64
conv2d[0][0]
_______________________________________________________________
___________________________________
activation (Activation)         (None, 32, 32, 16)   0
batch_normalization[0][0]
_______________________________________________________________
___________________________________
conv2d_1 (Conv2D)               (None, 32, 32, 16)   2320
activation[0][0]
_______________________________________________________________
___________________________________
batch_normalization_1 (BatchNor (None, 32, 32, 16)   64
conv2d_1[0][0]
_______________________________________________________________
```

```
___________________________________
activation_1 (Activation)       (None, 32, 32, 16)   0
batch_normalization_1[0][0]
_______________________________________________________________
___________________________________
conv2d_2 (Conv2D)               (None, 32, 32, 16)   2320
activation_1[0][0]
_______________________________________________________________
___________________________________
batch_normalization_2 (BatchNor (None, 32, 32, 16)   64
conv2d_2[0][0]
_______________________________________________________________
___________________________________
add (Add)                       (None, 32, 32, 16)   0
activation[0][0]

batch_normalization_2[0][0]
_______________________________________________________________
___________________________________
activation_2 (Activation)       (None, 32, 32, 16)   0
add[0][0]
_______________________________________________________________
___________________________________
conv2d_3 (Conv2D)               (None, 32, 32, 16)   2320
activation_2[0][0]
_______________________________________________________________
___________________________________
batch_normalization_3 (BatchNor (None, 32, 32, 16)   64
conv2d_3[0][0]
_______________________________________________________________
___________________________________
activation_3 (Activation)       (None, 32, 32, 16)   0
batch_normalization_3[0][0]
_______________________________________________________________
___________________________________
conv2d_4 (Conv2D)               (None, 32, 32, 16)   2320
activation_3[0][0]
_______________________________________________________________
___________________________________
batch_normalization_4 (BatchNor (None, 32, 32, 16)   64
conv2d_4[0][0]
_______________________________________________________________
___________________________________
add_1 (Add)                     (None, 32, 32, 16)   0
activation_2[0][0]

batch_normalization_4[0][0]
_______________________________________________________________
___________________________________
activation_4 (Activation)       (None, 32, 32, 16)   0
add_1[0][0]
_______________________________________________________________
___________________________________
conv2d_5 (Conv2D)               (None, 32, 32, 16)   2320
activation_4[0][0]
_______________________________________________________________
___________________________________
```

```
batch_normalization_5 (BatchNor (None, 32, 32, 16)   64
conv2d_5[0][0]
_______________________________________________________________
___________________________________
activation_5 (Activation)       (None, 32, 32, 16)   0
batch_normalization_5[0][0]
_______________________________________________________________
___________________________________
conv2d_6 (Conv2D)               (None, 32, 32, 16)   2320
activation_5[0][0]
_______________________________________________________________
___________________________________
batch_normalization_6 (BatchNor (None, 32, 32, 16)   64
conv2d_6[0][0]
_______________________________________________________________
___________________________________
add_2 (Add)                     (None, 32, 32, 16)   0
activation_4[0][0]

batch_normalization_6[0][0]
_______________________________________________________________
___________________________________
activation_6 (Activation)       (None, 32, 32, 16)   0
add_2[0][0]
_______________________________________________________________
___________________________________
conv2d_7 (Conv2D)               (None, 16, 16, 32)   4640
activation_6[0][0]
_______________________________________________________________
___________________________________
batch_normalization_7 (BatchNor (None, 16, 16, 32)   128
conv2d_7[0][0]
_______________________________________________________________
___________________________________
activation_7 (Activation)       (None, 16, 16, 32)   0
batch_normalization_7[0][0]
_______________________________________________________________
___________________________________
conv2d_8 (Conv2D)               (None, 16, 16, 32)   9248
activation_7[0][0]
_______________________________________________________________
___________________________________
conv2d_9 (Conv2D)               (None, 16, 16, 32)   544
activation_6[0][0]
_______________________________________________________________
___________________________________
batch_normalization_8 (BatchNor (None, 16, 16, 32)   128
conv2d_8[0][0]
_______________________________________________________________
___________________________________
add_3 (Add)                     (None, 16, 16, 32)   0
conv2d_9[0][0]

batch_normalization_8[0][0]
_______________________________________________________________
___________________________________
activation_8 (Activation)       (None, 16, 16, 32)   0
```

```
add_3[0][0]
_______________________________________________________________
___________________________________
conv2d_10 (Conv2D)              (None, 16, 16, 32)   9248
activation_8[0][0]
_______________________________________________________________
___________________________________
batch_normalization_9 (BatchNor (None, 16, 16, 32)   128
conv2d_10[0][0]
_______________________________________________________________
___________________________________
activation_9 (Activation)       (None, 16, 16, 32)   0
batch_normalization_9[0][0]
_______________________________________________________________
___________________________________
conv2d_11 (Conv2D)              (None, 16, 16, 32)   9248
activation_9[0][0]
_______________________________________________________________
___________________________________
batch_normalization_10 (BatchNo (None, 16, 16, 32)   128
conv2d_11[0][0]
_______________________________________________________________
___________________________________
add_4 (Add)                     (None, 16, 16, 32)   0
activation_8[0][0]

batch_normalization_10[0][0]
_______________________________________________________________
___________________________________
activation_10 (Activation)      (None, 16, 16, 32)   0
add_4[0][0]
_______________________________________________________________
___________________________________
conv2d_12 (Conv2D)              (None, 16, 16, 32)   9248
activation_10[0][0]
_______________________________________________________________
___________________________________
batch_normalization_11 (BatchNo (None, 16, 16, 32)   128
conv2d_12[0][0]
_______________________________________________________________
___________________________________
activation_11 (Activation)      (None, 16, 16, 32)   0
batch_normalization_11[0][0]
_______________________________________________________________
___________________________________
conv2d_13 (Conv2D)              (None, 16, 16, 32)   9248
activation_11[0][0]
_______________________________________________________________
___________________________________
batch_normalization_12 (BatchNo (None, 16, 16, 32)   128
conv2d_13[0][0]
_______________________________________________________________
___________________________________
add_5 (Add)                     (None, 16, 16, 32)   0
activation_10[0][0]

batch_normalization_12[0][0]
```

```
_______________________________________________________________
___________________________________
activation_12 (Activation)      (None, 16, 16, 32)   0
add_5[0][0]
_______________________________________________________________
___________________________________
conv2d_14 (Conv2D)              (None, 8, 8, 64)     18496
activation_12[0][0]
_______________________________________________________________
___________________________________
batch_normalization_13 (BatchNo (None, 8, 8, 64)     256
conv2d_14[0][0]
_______________________________________________________________
___________________________________
activation_13 (Activation)      (None, 8, 8, 64)     0
batch_normalization_13[0][0]
_______________________________________________________________
___________________________________
conv2d_15 (Conv2D)              (None, 8, 8, 64)     36928
activation_13[0][0]
_______________________________________________________________
___________________________________
conv2d_16 (Conv2D)              (None, 8, 8, 64)     2112
activation_12[0][0]
_______________________________________________________________
___________________________________
batch_normalization_14 (BatchNo (None, 8, 8, 64)     256
conv2d_15[0][0]
_______________________________________________________________
___________________________________
add_6 (Add)                     (None, 8, 8, 64)     0
conv2d_16[0][0]

batch_normalization_14[0][0]
_______________________________________________________________
___________________________________
activation_14 (Activation)      (None, 8, 8, 64)     0
add_6[0][0]
_______________________________________________________________
___________________________________
conv2d_17 (Conv2D)              (None, 8, 8, 64)     36928
activation_14[0][0]
_______________________________________________________________
___________________________________
batch_normalization_15 (BatchNo (None, 8, 8, 64)     256
conv2d_17[0][0]
_______________________________________________________________
___________________________________
activation_15 (Activation)      (None, 8, 8, 64)     0
batch_normalization_15[0][0]
_______________________________________________________________
___________________________________
conv2d_18 (Conv2D)              (None, 8, 8, 64)     36928
activation_15[0][0]
_______________________________________________________________
___________________________________
batch_normalization_16 (BatchNo (None, 8, 8, 64)     256
conv2d_18[0][0]
_______________________________________________________________
___________________________________
add_7 (Add)                     (None, 8, 8, 64)     0
activation_14[0][0]

batch_normalization_16[0][0]
_______________________________________________________________
___________________________________
activation_16 (Activation)      (None, 8, 8, 64)     0
add_7[0][0]
_______________________________________________________________
___________________________________
conv2d_19 (Conv2D)              (None, 8, 8, 64)     36928
activation_16[0][0]
_______________________________________________________________
___________________________________
batch_normalization_17 (BatchNo (None, 8, 8, 64)     256
conv2d_19[0][0]
_______________________________________________________________
___________________________________
activation_17 (Activation)      (None, 8, 8, 64)     0
batch_normalization_17[0][0]
_______________________________________________________________
___________________________________
conv2d_20 (Conv2D)              (None, 8, 8, 64)     36928
activation_17[0][0]
_______________________________________________________________
___________________________________
batch_normalization_18 (BatchNo (None, 8, 8, 64)     256
conv2d_20[0][0]
_______________________________________________________________
___________________________________
add_8 (Add)                     (None, 8, 8, 64)     0
activation_16[0][0]

batch_normalization_18[0][0]
_______________________________________________________________
___________________________________
activation_18 (Activation)      (None, 8, 8, 64)     0
add_8[0][0]
_______________________________________________________________
___________________________________
average_pooling2d (AveragePooli (None, 1, 1, 64)     0
activation_18[0][0]
_______________________________________________________________
___________________________________
flatten (Flatten)               (None, 64)           0
average_pooling2d[0][0]
_______________________________________________________________
___________________________________
dense (Dense)                   (None, 10)           650
flatten[0][0]
===============================================================
===================================
Total params: 274,442
Trainable params: 273,066
```

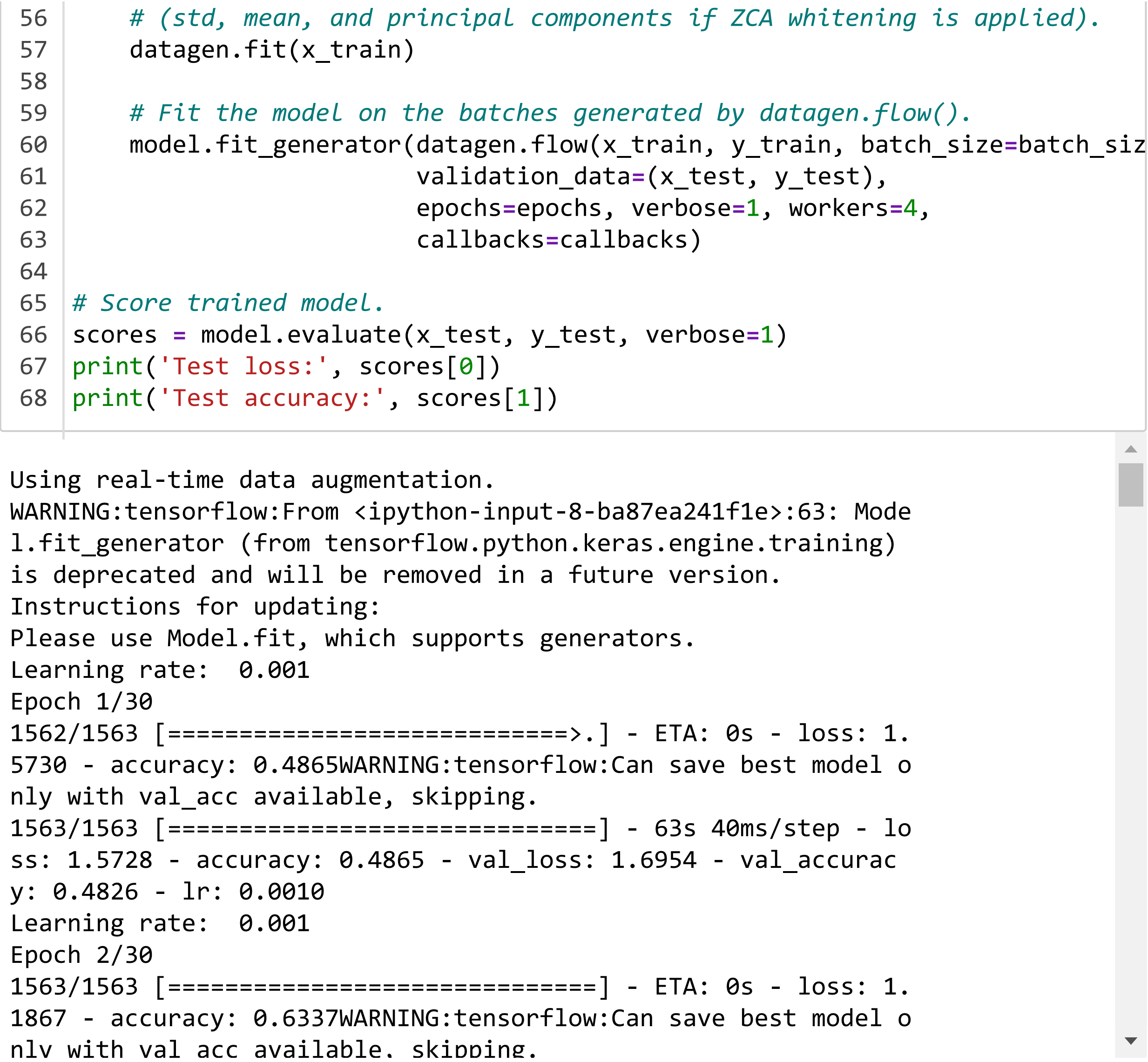
```
Non-trainable params: 1,376
_______________________________________________________________

___________________________________
ResNet20v1
```

In [7]:

```python
# Prepare model model saving directory.
save_dir = os.path.join(os.getcwd(), 'saved_models')
model_name = 'cifar10_%s_model.{epoch:03d}.h5' % model_type
if not os.path.isdir(save_dir):
    os.makedirs(save_dir)
filepath = os.path.join(save_dir, model_name)

# Prepare callbacks for model saving and for learning rate adjustment.
checkpoint = ModelCheckpoint(filepath=filepath,
                             monitor='val_acc',
                             verbose=1,
                             save_best_only=True)

lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-6)

callbacks = [checkpoint, lr_reducer, lr_scheduler]
```

In [8]:

```
# Run training, with or without data augmentation.
if not data_augmentation:
    print('Not using data augmentation.')
    model.fit(x_train, y_train,
              batch_size=batch_size,
              epochs=epochs,
              validation_data=(x_test, y_test),
              shuffle=True,
              callbacks=callbacks)
else:
    print('Using real-time data augmentation.')
    # This will do preprocessing and realtime data augmentation:
    datagen = ImageDataGenerator(
        # set input mean to 0 over the dataset
        featurewise_center=False,
        # set each sample mean to 0
        samplewise_center=False,
        # divide inputs by std of dataset
        featurewise_std_normalization=False,
        # divide each input by its std
        samplewise_std_normalization=False,
        # apply ZCA whitening
        zca_whitening=False,
        # epsilon for ZCA whitening
        zca_epsilon=1e-06,
        # randomly rotate images in the range (deg 0 to 180)
        rotation_range=0,
        # randomly shift images horizontally
        width_shift_range=0.1,
        # randomly shift images vertically
        height_shift_range=0.1,
        # set range for random shear
        shear_range=0.,
        # set range for random zoom
        zoom_range=0.,
        # set range for random channel shifts
        channel_shift_range=0.,
        # set mode for filling points outside the input boundaries
        fill_mode='nearest',
        # value used for fill_mode = "constant"
        cval=0.,
        # randomly flip images
        horizontal_flip=True,
        # randomly flip images
        vertical_flip=False,
        # set rescaling factor (applied before any other transformation)
        rescale=None,
        # set function that will be applied on each input
        preprocessing_function=None,
        # image data format, either "channels_first" or "channels_last"
        data_format=None,
        # fraction of images reserved for validation (strictly between 0 an
        validation_split=0.0)

    # Compute quantities required for featurewise normalization
```



```
56      # (std, mean, and principal components if ZCA whitening is applied).
57      datagen.fit(x_train)
58  
59      # Fit the model on the batches generated by datagen.flow().
60      model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_siz
61                          validation_data=(x_test, y_test),
62                          epochs=epochs, verbose=1, workers=4,
63                          callbacks=callbacks)
64  
65  # Score trained model.
66  scores = model.evaluate(x_test, y_test, verbose=1)
67  print('Test loss:', scores[0])
68  print('Test accuracy:', scores[1])
```

```
Using real-time data augmentation.
WARNING:tensorflow:From <ipython-input-8-ba87ea241f1e>:63: Mode
l.fit_generator (from tensorflow.python.keras.engine.training)
is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Learning rate:  0.001
Epoch 1/30
1562/1563 [============================>.] - ETA: 0s - loss: 1.
5730 - accuracy: 0.4865WARNING:tensorflow:Can save best model o
nly with val_acc available, skipping.
1563/1563 [==============================] - 63s 40ms/step - lo
ss: 1.5728 - accuracy: 0.4865 - val_loss: 1.6954 - val_accurac
y: 0.4826 - lr: 0.0010
Learning rate:  0.001
Epoch 2/30
1563/1563 [==============================] - ETA: 0s - loss: 1.
1867 - accuracy: 0.6337WARNING:tensorflow:Can save best model o
nly with val_acc available, skipping.
```