```
#!/bin/python3.5
# Ostap Voynarovskiy
# CGML HW4
# October 4 2018
# Professo Curro
import os
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import keras
from keras.models import Sequential
from keras.datasets import cifar10
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, Activati
on, BatchNormalization, AveragePooling2D, Input
from keras import regularizers
from keras.models import Model
from keras.callbacks import ReduceLROnPlateau

from keras.optimizers import Adam
from keras.regularizers import l2

#from keras import backend as K

num_classes=10
BATCH_SIZE = 32
epochs = 200
DROP_RATE =.3
weight_decay = 1e-4
def genTrainAndVal(f,l): #split the features and labels of the training data 80:
20 train and validation
        lx=f.shape[0]
        z = f.shape[0]
        s = np.arange(z)
        np.random.shuffle(s)
        fs = f[s]                       # features shuffled
        ls = l[s]                       # labels shuffled
        lx = f.shape[0]         # len of the features
        nv = int( lx *.2)       # num validation samp
        print (fs[nv:].shape, ls[nv:].shape, fs[:nv].shape, ls[:nv].shape)
        return fs[nv:], ls[nv:], fs[:nv], ls[:nv]


datagen = ImageDataGenerator(featurewise_center=False,samplewise_center=False,
        featurewise_std_normalization=False,samplewise_std_normalization=False,
        zca_whitening=False,zca_epsilon=1e-06,rotation_range=0,width_shift_range
=0.1,
        height_shift_range=0.1,shear_range=0.,zoom_range=0.,channel_shift_range=
0.,
        fill_mode='nearest',cval=0.,horizontal_flip=True,vertical_flip=False,resca
le=None,
        preprocessing_function=None,data_format=None,validation_split=0.0)

#modified from https://github.com/keras-team/keras/blob/master/examples/cifar10_
resnet.py
def lr_schedule(epoch):
        lr = 1e-3
        if epoch > 150:
                lr *= 0.5e-3
        elif epoch > 120:
                lr *= 1e-3
        elif epoch > 100:
                lr *= 1e-2
        elif epoch > 60:
                lr *= 1e-1
        print('Learning rate: ', lr)
        return lr
```



```
# load cifar 10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
input_shape = x_train.shape[1:]
print(input_shape)

#convert to float and normalize
x_train,x_test = x_train.astype('float32'),x_test.astype('float32')
x_train,x_test = x_train/255,x_test/255

x_t,y_t,x_v,y_v =genTrainAndVal(x_train,y_train)

#print Shapes
print("Training features shape: ", x_t.shape)
print("Validation features shape: ",x_v.shape)
print("Test features shape: ", x_test.shape)

#one hot encode the labels
y_t = keras.utils.to_categorical(y_t,num_classes)
y_v = keras.utils.to_categorical(y_v,num_classes)
y_test  = keras.utils.to_categorical(y_test, num_classes)
def res_layer(inputs, num_filters=16, kernel_size=3,
        strides=1, activation='elu'):

        conv = Conv2D(num_filters,kernel_size=kernel_size,
                strides=strides,padding='same',kernel_initializer='glorot_uniform',
                kernel_regularizer=l2(weight_decay))

        x = inputs
        x1 = Conv2D(num_filters,kernel_size=kernel_size,
                strides=strides,padding='same',kernel_initializer='glorot_uniform',
                kernel_regularizer=l2(weight_decay))(x)
        x = BatchNormalization()(x1)
        x = Activation(activation)(x)
        x = Conv2D(num_filters,kernel_size=kernel_size,
                strides=strides,padding='same',kernel_initializer='glorot_uniform',
                kernel_regularizer=l2(weight_decay))(x)#32,32,16
        x = BatchNormalization()(x)
        x = Activation(activation)(x)
        x = Conv2D(num_filters,kernel_size=kernel_size,
                strides=strides,padding='same',kernel_initializer='glorot_uniform',
                kernel_regularizer=l2(weight_decay))(x)
        x = BatchNormalization()(x)
        x = keras.layers.add([x1,x])
        return x

def resNet(input_shape, num_classes=10):
        num_filters = 16
        print("hi",input_shape)
        inputs = Input(shape=input_shape)
        print(inputs.shape)
        y = res_layer(inputs=inputs)

        for i in range(2):
                y = Activation("elu")(y)
                y = MaxPooling2D(pool_size=2,strides=2)(y)
                num_filters*=2
                y = res_layer(inputs=y, num_filters = num_filters )


        x = Activation("elu")(y)
        x = AveragePooling2D(pool_size=8)(x)
        y = Flatten()(x)
        outputs = Dense(num_classes,activation='softmax',kernel_initializer='he_nor
mal')(y)

        model = Model(inputs=inputs, outputs=outputs)
        return model
```

```
model = resNet(input_shape, num_classes=10)

model.summary()
model.compile(loss="categorical_crossentropy",
        optimizer=keras.optimizers.Adam(lr=lr_schedule(0)),
        metrics=['accuracy'])

datagen.fit(x_t)


lr_scheduler = LearningRateScheduler(lr_schedule)

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

callbacks = [lr_reducer, lr_scheduler]

model.fit_generator(datagen.flow(x_t, y_t, batch_size=BATCH_SIZE),
        validation_data =(x_v,y_v),
        steps_per_epoch=x_t.shape[0]/BATCH_SIZE,
        epochs = epochs,
        verbose = 1,
        callbacks=callbacks)



score = model.evaluate(x_test,y_test,verbose=1)
print("Test loss:",score[0])
print("Test accuracy:",score[1])
'''
I implemented a variation of the resnet20 from the paper https://arxiv.org/pdf/1512.03385.pdf
and pulled some syntax for the callbacks and live data generation from the
keras github implementation.https://github.com/keras-team/keras/blob/master/examples/cifar10_resnet.py
I went through many different variations and found that some things just dont
really make sense. For instance, when i ran this program with 30% dropout,
i lost 2% accuracy on the test set. I realized however that the average pooling
seemed to help. I also attempted a model that didnt use the skip connections. Its hard
to say whether the skip connections helped or not, but they took a long time to figure
out how to implement. The Cifar 100 model used the non skip connection version of the model and it
seemed to work fine although it did have the benefit of top 5 accuracy.
The learning rate drop however did add a lot of accuracy and was proably
the most useful thing. It dropped multiple times and stabilized my model. While
my final model was overfit, it achieved a better accuracy.


'''
```