```
#!/bin/python3.5
# Ostap Voynarovskiy
# CGML HW4
# October 4 2018
# Professo Curro

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.layers import Dense, Dropout, Flatten,Conv2D, MaxPooling2D, Activatio
n, BatchNormalization
from keras import regularizers
from keras.metrics import top_k_categorical_accuracy
#from keras import backend as K

num_classes=100
BATCH_SIZE = 32
epochs = 32 #we achieve overfitting after like 15-20 epochs
DROP_RATE =.5
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]


# load cifar 10
#(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# load cifar 100
from keras.datasets import cifar100
(x_train, y_train), (x_test, y_test) = cifar100.load_data(label_mode=’fine’)

#fonvert 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)

model = Sequential()

model.add(Conv2D(32,(4,4),padding=’same’,kernel_regularizer=regularizers.l2(weig
ht_decay), data_format=’channels_last’, kernel_initializer=’glorot_uniform’, input_shap
e=x_t[0].shape))
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Conv2D(32,(3,3),strides=1,padding=’same’, kernel_regularizer=regulariz
```

```
ers.l2(weight_decay), kernel_initializer=’glorot_uniform’))
model.add(Activation("elu"))

model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(DROP_RATE))

model.add(Conv2D(64,kernel_size=(3,3),padding=’same’,kernel_regularizer=regulari
zers.l2(weight_decay),data_format=’channels_last’,kernel_initializer=’glorot_uniform’))
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Conv2D(64,kernel_size=(4,4),padding=’same’,kernel_regularizer=regulari
zers.l2(weight_decay),data_format=’channels_last’,kernel_initializer=’glorot_uniform’))
model.add(Activation("elu"))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2), strides=2))
model.add(Dropout(DROP_RATE))

model.add(Conv2D(128,kernel_size=(5,5),padding=’same’,kernel_regularizer=regular
izers.l2(weight_decay),data_format=’channels_last’,kernel_initializer=’glorot_uniform’)
)
model.add(Activation("elu"))
model.add(BatchNormalization())
model.add(Conv2D(128,kernel_size=(2,2),padding=’same’,kernel_regularizer=regular
izers.l2(weight_decay),data_format=’channels_last’,kernel_initializer=’glorot_uniform’)
)
model.add(Activation("elu"))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=(2,2), strides=2))
model.add(Dropout(.2))

model.add(Flatten())
model.add(Dense(num_classes,activation="softmax"))

model.summary()



model.compile(loss=keras.losses.categorical_crossentropy,
        optimizer=keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsi
lon=None, decay=0.0, amsgrad=False),
        metrics=["top_k_categorical_accuracy"])

model.fit(x_t, y_t,
        batch_size=BATCH_SIZE,
        epochs = epochs,
        verbose = 1,
        validation_data =(x_v,y_v))

score = model.evaluate(x_test,y_test,verbose= 1)
print("Test loss:",score[0])
print("Test accuracy:",score[1])
```