```
# DongKyu Kim
# ECE 471 CGML Assignment 4
# CIFAR-10
# Professor Curro

# library imports
import numpy as np
from tqdm import tqdm
import keras
import tensorflow as tf
from keras import optimizers, regularizers
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNor
malization, Activation, Input
from keras.callbacks import LearningRateScheduler

# Parameters
r = 0.9 # Ratio of training data otu of training + validation
batch_size = 128
epochs = 200
numclass = 10
learning_rate = 0.005

# Useful functions
def one_hot_encoding(data,numclass):
    targets = np.array(data).reshape(-1)
    return np.eye(numclass)[targets]

def lr_adaptive(self,epoch):
    lr = learning_rate
    if epoch > 20:
        lr = learning_rate/10
    elif epoch > 40:
        lr = learning_rate/20
    return lr

# Data Generation
class Data(object):
    def __init__(self,r,numclass): # N is number of training data that will be u
sed
        (self.xtemp,self.ytemp),(self.X_test,self.Y_test) = cifar10.load_data()
        mask = np.full(self.xtemp.shape[0],True)
        mask[np.random.choice(self.xtemp.shape[0], int(self.xtemp.shape[0]*(1-r)
), replace=False)] = False
        self.numclass = numclass
        self.X_train = self.xtemp[mask].astype('float32')
        self.Y_train = self.ytemp[mask]
        self.X_val = self.xtemp[~mask].astype('float32')
        self.Y_val = self.ytemp[~mask]
        self.X_test = self.X_test.astype('float32')
        self.Y_test = self.Y_test
        self.X_train = self.normalize(self.X_train)
        self.X_val = self.normalize(self.X_val)
        self.X_test = self.normalize(self.X_test)
        self.Y_train = one_hot_encoding(self.Y_train,self.numclass)
        self.Y_val = one_hot_encoding(self.Y_val,self.numclass)
        self.Y_test = one_hot_encoding(self.Y_test,self.numclass)

    def normalize(self,X):
        X /= 255
        mu = [0.4914,0.4822,0.4465]
        std = [0.2023,0.1994,0.2010] # These values are from https://github.com/
```

```
kuangliu/pytorch-cifar/issues/19
        for i in range(0,2):
            X[:,:,:,i] = (X[:,:,:,i]-mu[i])/std[i]
        return X

# Model
class My_Model(object):
    def __init__(self,data,batch_size,epochs):
        self.data = data
        self.batch_size = batch_size
        self.epochs = epochs

    def convconv(self,filters,kernel_size,strides):
        return Conv2D(filters,kernel_size,strides =strides,padding='same',activa
tion='relu')

    def residual_block(self,input,filter): #This is from https://arxiv.org/pdf/1
603.05027.pdf Identity Mappings in Deep Residual Networks
        x = BatchNormalization()(input)
        x = Activation('relu')(x)
        x = self.convconv(filter,[3,3],1)(x)
        x = BatchNormalization()(input)
        x = Activation('relu')(x)
        x = self.convconv(filter,[3,3],1)(x)

        y = BatchNormalization()(input)
        y = Activation('relu')(y)
        y = self.convconv(filter,[3,3],1)(y)
        y = BatchNormalization()(y)
        y = Activation('relu')(y)
        y = self.convconv(filter,[3,3],1)(y)
        return keras.layers.add([input,x,y])

    def all_conv_block(self,filters,dropout,input):
        x = self.convconv(filters,[3,3],1)(input)
        x = BatchNormalization()(x)
        x = self.convconv(filters,[3,3],1)(x)
        x = BatchNormalization()(x)
        x = MaxPooling2D(2,strides = 2)(x)
        x = Dropout(dropout)(x)
        return x

    def build_model(self):
        input = Input(shape = self.data.X_train.shape[1:])
        x = self.all_conv_block(32,0.25,input)
        x = self.all_conv_block(64,0.35,x)
        x = self.all_conv_block(128,0.45,x)
        x = self.all_conv_block(256,0.35,x)
#         x = self.convconv(64,[7,7],2)(input)
#         x = MaxPooling2D(3,strides = 2)(x)
#         for i in range(0,2):
#             x = self.residual_block(x,64)
#         x = self.convconv(128,[3,3],2)(x)
#         for i in range(0,2):
#             x = self.residual_block(x,128)
#         x = self.convconv(256,[3,3],2)(x)
#         for i in range(0,4):
#             x = self.residual_block(x,256)
#         x = self.convconv(512,[3,3],2)(x)
#         for i in range(0,2):
#             x = self.residual_block(x,512)
#         x = keras.layers.GlobalAveragePooling2D()(x)
        x = Flatten()(x)
```

```
        x = Dense(1024,activation = 'relu')(x)
        y = Dense(self.data.numclass,activation='softmax')(x)
        self.model = Model(inputs = input, outputs = y)

    def train(self):
#         self.optim = optimizers.SGD(learning_rate,momentum=0.9,decay=5e-4)
        self.optim = optimizers.Adam(learning_rate)
        self.model.compile(self.optim,'categorical_crossentropy',['accuracy'])
        self.datagen = keras.preprocessing.image.ImageDataGenerator(
            horizontal_flip = True,fill_mode = 'constant',
            width_shift_range = 4, height_shift_range = 4
            )
        self.datagen.fit(self.data.X_train)
        self.model.fit_generator(self.datagen.flow(self.data.X_train, self.data.
Y_train,batch_size=self.batch_size),
            steps_per_epoch=len(self.data.X_train)/self.batch_size,epochs=self.e
pochs,validation_data = (self.data.X_val,self.data.Y_val),
            callbacks=[LearningRateScheduler(lr_adaptive)],verbose=2)
#       self.model.fit(self.data.X_train,self.data.Y_train,self.batch_size,self.
epochs,validation_data = (self.data.X_val,self.data.Y_val),shuffle=True)

print('This is my cifar_10 case')
data = Data(r,numclass)
My_Model = My_Model(data,batch_size,epochs)
My_Model.build_model()
My_Model.train()
scores = My_Model.model.evaluate(data.X_test,data.Y_test,verbose=2)
print('Test loss:', scores[0])
print('Test accuracy:',scores[1])

# I started with my MNIST model, it didn't work well
# I experimented with a deeper version of the MNIST model, and it just lingered
at 50%
# I moved onto residual neural network, https://arxiv.org/pdf/1512.03385.pdf(Dee
p Residual Learning for Image Recognition), with
# identity shortcut https://arxiv.org/pdf/1603.05027.pdf (Identity Mappings in D
eep Residual Networks).
# After 32 epochs, it converges at validation accuracy of 0.6479, and test set a
ccuracy of 0.9967, with best validation accuracy of 0.6513
# This model overfits, so I added a simple data augmentation scheme that randoml
y flips horizontal and vertically.
# With this augmentation method, I reach a validation accuracy of 0.6821, which
is slightly better, but not that good.
# Then I implemented the augmentation method that is close to the above papers,
0.721 and achived with 64 epochs
# Then I realized that I didn't do normalization on the data, and found values f
or optimal mean and std online to normalize.
# The result improved a lot, with epochs 64, now the model reaches 0.8042 valida
tion error
# I changed to elu. 0.7898 at 30th, I changed back to relu and added rotation_ra
nge in the data augmentation, this didn't help
# I added another branch in residual network. This idea was motivated by https:/
/arxiv.org/pdf/1705.07485.pdf (Shake-Shake regularization)
# I increased the learning rate from 0.005 to 0.007 because new model was slow.
However, this converges to 0.76. I realized my method is just not working out.
# I heard from someone (joey) that all conv layer works fine, and it does work g
reatly with only 8 total layers. (I was disappointed), it goes to 0.8214
# I realized it learns slowly, so I added adaptive learning rate, and added anot
her conv_block.
```