Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Лабораторная работа №5
Дисциплина «Разработка интеллектуальных систем»

Выполнил:
Съестов Дмитрий Вячеславович
Группа P3417

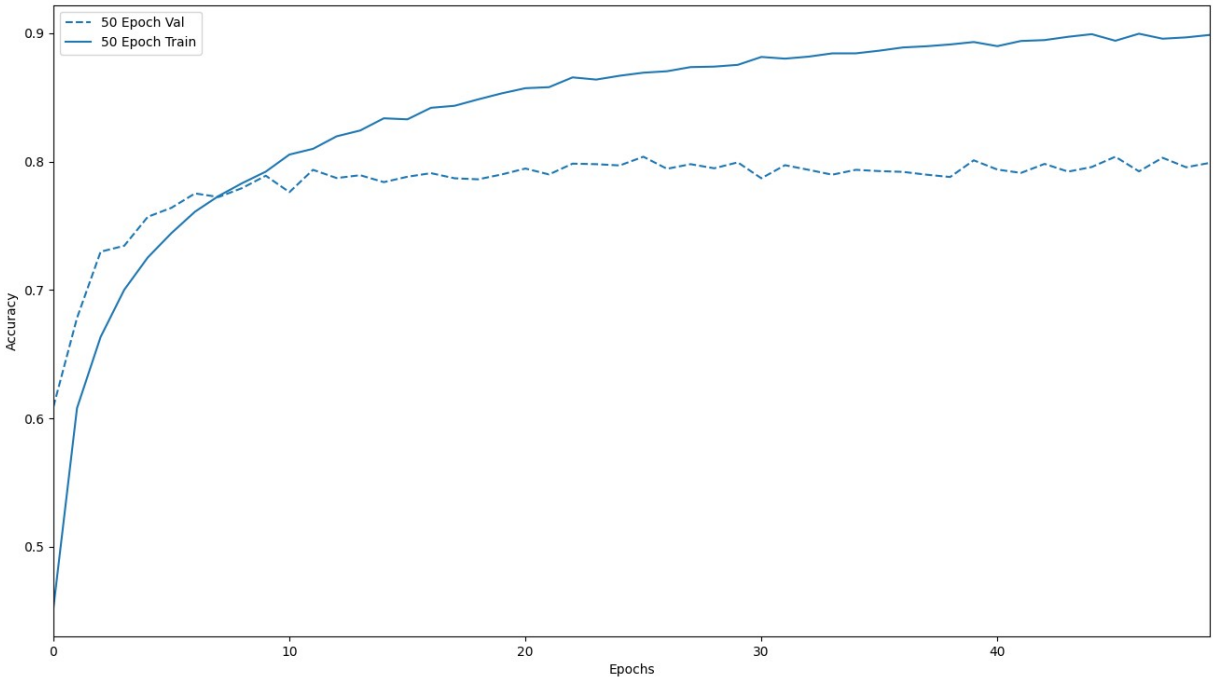
Преподаватель:
Жукова Наталия Александровна

Санкт-Петербург
2020

Листинг программы

```
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Input, Convolution2D, MaxPooling2D, Dense, Dropout,
Flatten
from keras.utils import np_utils
import numpy as np
import matplotlib.pyplot as plt

def plot_history(histories, key='binary_crossentropy'):
    plt.figure(figsize=(16,10))

    for name, history in histories:
        val = plt.plot(history.epoch, history.history['val_'+key],
                    '--', label=name.title()+' Val')
        plt.plot(history.epoch, history.history[key],
color=val[0].get_color(),
                    label=name.title()+' Train')

    plt.xlabel('Epochs')
    plt.ylabel(key.replace('_',' ').title())
    plt.legend()

    plt.xlim([0,max(history.epoch)])

    plt.show()
    plt.savefig('figure.png')


batch_size = 32     # in each iteration, we consider 32 training examples at
once
num_epochs = 50     # we iterate 50 times over the entire training set
kernel_size = 3     # we will use 3x3 kernels throughout
pool_size = 2       # we will use 2x2 pooling throughout
conv_depth_1 = 32   # we will initially have 32 kernels per conv. layer...
conv_depth_2 = 64   # ...switching to 64 after the first pooling layer
drop_prob_1 = 0.25  # dropout after pooling with probability 0.25
drop_prob_2 = 0.5   # dropout in the dense layer with probability 0.5
hidden_size = 512   # the dense layer will have 512 neurons

(X_train, y_train), (X_test, y_test) = cifar10.load_data() # fetch CIFAR-10
data

num_train, depth, height, width = X_train.shape # there are 50000 training
examples in CIFAR-10
num_test = X_test.shape[0]                      # there are 10000 test
examples in CIFAR-10
num_classes = np.unique(y_train).shape[0]       # there are 10 image classes

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# Normalise data to [0, 1] range
X_train /= np.max(X_train)
X_test /= np.max(X_train)

# One-hot encode the labels
```

```
Y_train = np_utils.to_categorical(y_train, num_classes)
Y_test = np_utils.to_categorical(y_test, num_classes)

inp = Input(shape=(depth, height, width)) # N.B. depth goes first in Keras

# Conv [32] -> Conv [32] -> Pool (with dropout on the pooling layer)
conv_1 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

# Conv [64] -> Conv [64] -> Pool (with dropout on the pooling layer)
conv_3 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(drop_1)
conv_4 = Convolution2D(conv_depth_2, kernel_size, kernel_size,
border_mode='same', activation='relu')(conv_3)
pool_2 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)

# Now flatten to 1D, apply Dense -> ReLU (with dropout) -> softmax
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)

model = Model(input=inp, output=out) # To define a model, just specify its
input and output layers

model.compile(loss='categorical_crossentropy',  # using the cross-entropy
loss function
              optimizer='adam',                 # using the Adam optimiser
              metrics=['accuracy'])             # reporting the accuracy

H = model.fit(X_train, Y_train, # Train the model using the training set...
          batch_size=batch_size, nb_epoch=num_epochs,
          verbose=1, validation_split=0.1) # ...holding out 10% of the data
for validation
model.evaluate(X_test, Y_test, verbose=1) # Evaluate the trained model on the
test set!

plot_history([('50 epoch', H)], key='accuracy')
```



Вывод

Точность валидации перестала повышаться примерно на 10 эпохе, для повышения скорости обучения можно снизить количество эпох до 10. Также обучение можно существенно ускорить при помощи CUDA, обучая нейронную сеть на GPU.