МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Искуственные нейронные сети»

Тема: Распознавание объектов на фотографиях

Студент гр. 8382 _____ Щемель Д.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель

Распознавание объектов на фотографиях (Object Recognition in Photographs)

CIFAR-10 (классификация небольших изображений по десяти классам: самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик).

Задачи

- Ознакомиться со сверточными нейронными сетями
- Изучить построение модели в Keras в функциональном виде
- Изучить работу слоя разреживания (Dropout)

Требования

- Построить и обучить сверточную нейронную сеть
- Исследовать работу сеть без слоя Dropout
- Исследовать работу сети при разных размерах ядра свертки

Ход работы

Рассмотрим результаты свёрточной нейронной сети (CNN) при различных параметрах (точность указана для тестовых данных):

Номер модели	Количество эпох	Dropout слой	Размер ядра	Точность
1	200	+	3x3	0.3896999955177307
2	77	-	3x3	0.5662999749183655
3	77	+	4x4	0.6277823425265721
4	77	+	2x2	0.4962850271057229

По выводу результатов обучения сети было видно, что значение точности достигает значения 0.8006 уже на 69ой эпохе и выше значения 0.81 не поднимается, поэтому количество эпох для обучения было сокращено.

Вывод

В ходе выполнения лаюораторной работы была обучена нейроанная сеть, распознающая объекты десяти классов на фотографиях.

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

batch_size = 32  # in each iteration, we consider 32 training examples at once
num_epochs = 77  # we iterate 200 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 exampl
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')
X_train /= np.max(X_train)  # Normalise data to [0, 1] range
X_test /= np.max(X_train)  # Normalise data to [0, 1] range
Y_train = np_utils.to_categorical(y_train, num_classes)  # One-hot encode the labe
Y_test = np_utils.to_categorical(y_test, num_classes)  # One-hot encode the labels

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), padding='same', ac
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size), padding='same', ac
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), padding='same', ac
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size), padding='same', ac
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()(pool_2)
hidden = Dense(hidden_size, activation='relu')(flat)
# drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(hidden)
model = Model(inputs=inp, outputs=out)  # To define a model, just specify its inpu

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

model.fit(X_train, Y_train,  # Train the model using the training set...
          batch_size=batch_size, epochs=num_epochs,
          verbose=1, validation_split=0.1)  # ...holding out 10% of the data for v
model.evaluate(X_test, Y_test, verbose=1)  # Evaluate the trained model on the tes
```