МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе № 5

по дисциплине «Искусственные нейронные сети»

ТЕМА: Распознавание рукописных символов

Студент гр. 8383 _____ Мололкин К.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы

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)

Выполнение работы

Был написан код, создающий сверточную нейронную сеть, классифицирующую изображения по 10 классам из набора CIFAR-10. Так как мощности личного компьютера не хватает для быстрого обучения модели, для обучения был использован сервис Microsoft Azure, количество эпох было уменьшено до 100, так как на прохождения одной эпохи уходит 17 секунд.

Начальная конфигурация – размер ядра 3*3, в табл. 1 представлены полученные значения.

Таблица 1 – результаты первой модели.

Точность обучающей выборки	Точность выборки валидации	Точность тестовой выборки
92%	80%	51%

Так как по логам программы было заметно, что точность модели на выборке валидации меняется очень медленно с 30 эпохи, было решено уменьшить количество эпох до 40.

Следующим шагом была изучена конфигурация без слоев dropout и размер ядра 3*3, результат представлен в табл. 2.

Таблица 2 – результаты второй модели.

Точность обучающей выборки	Точность выборки валидации	Точность тестовой выборки
99%	74%	59%

Точность на обучающей выборке возросла до 99%, точность валидации уменьшилась до 74%, а точность тестовой выборки возросла до 59%. Можно сделать вывод о переобучении сети.

Затем была изучена модель со слоями dropout и размером ядра 2*2, результат представлен в табл. 3.

Таблица 3 – результаты 3 модели.

Точность обучающей выборки	Точность выборки валидации	Точность тестовой выборки
89%	79%	36%

Точность на тестовой выборке сильно упала до 36%.

Следующим шагом была изучена модель с размером ядра 4*4, результат в табл. 4.

Таблица 4 – результаты 4 модели.

Точность обучающей выборки	Точность выборки валидации	Точность тестовой выборки
88%	77%	55%

Вывод

Во время выполнения лабораторной работы была построена и обучена сверточная ИНС, которая классифицирует изображения по 10 классам. Так же было изучено влияние наличие Dropout слоя и размера ядра свертки. Лучшую точность на выборке валидации показывает модель с размером ядра 3*3 и

слоем Dropout, но лучшую точность на тестовой выборке показывает модель без слоев Dropout – 59%.

Приложение А

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


batch_size = 32
num_epochs = 8
kernel_size = 3
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

num_train, height, width, depth = X_train.shape
num_test = X_test.shape[0]
num_classes = np.unique(y_train).shape[0]

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= np.max(X_train)
X_test /= np.max(X_train)

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

inp = Input(shape=(height, width, depth))

conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
                       padding='same', activation='relu',
      data_format='channels_last')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
                       padding='same', activation='relu',
      data_format='channels_last')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

conv_3 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
                       padding='same', activation='relu',
      data_format='channels_last')(drop_1)
conv_4 = Convolution2D(conv_depth_2, (kernel_size, kernel_size),
                       padding='same', activation='relu',
      data_format='channels_last')(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(inputs=[inp], outputs=[out])  # To define a model, just specify
      its input and output layers
model.compile(loss='categorical_crossentropy', optimizer='adam',
      metrics=['accuracy'])

model.fit(X_train, Y_train,  batch_size=batch_size,
          epochs=num_epochs, verbose=1, validation_split=0.1)  # ...holding
      out 10% of the data for validation
model.evaluate(X_test, Y_test, verbose=1))
```