**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))