Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

|  |
| --- |
|  |

**ЛабоРАТОРНАЯ РАБОТА №3**

«Реализация свёрточной нейронной сети»

|  |  |  |
| --- | --- | --- |
| Студент |  | К. Д. Зюсько |
| Преподаватель |  | М. В. Стержанов |

Минск 2020

ХОД РАБОТЫ

**Задание.**

В работе предлагается использовать набор данных notMNIST, который состоит из изображений размерностью 28×28 первых 10 букв латинского алфавита (A … J, соответственно). Обучающая выборка содержит порядка 500 тыс. изображений, а тестовая – около 19 тыс.

Данные можно скачать по ссылке:

* https://commondatastorage.googleapis.com/books1000/notMNIST\_large.tar.gz (большой набор данных);
* https://commondatastorage.googleapis.com/books1000/notMNIST\_small.tar.gz (маленький набор данных).

Описание данных на английском языке доступно по ссылке:

http://yaroslavvb.blogspot.sg/2011/09/notmnist-dataset.html

1. Реализуйте нейронную сеть с двумя сверточными слоями, и одним полносвязным с нейронами с кусочно-линейной функцией активации. Какова точность построенное модели?
2. Замените один из сверточных слоев на слой, реализующий операцию пулинга (Pooling) с функцией максимума или среднего. Как это повлияло на точность классификатора?
3. Реализуйте классическую архитектуру сверточных сетей LeNet-5 (http://yann.lecun.com/exdb/lenet/).
4. Сравните максимальные точности моделей, построенных в лабораторных работах 1-3. Как можно объяснить полученные различия?

**Результат выполнения:**

1. Код реализации:

# definition

model = Sequential()

# adding layers

model.add(Conv2D(16, kernel\_size=(3, 3), input\_shape=(28, 28, 1), padding='same', activation='relu'))

model.add(Conv2D(32, kernel\_size=(3, 3), padding='same', activation='relu'))

model.add(Flatten())

model.add(Dense(120, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['acc'])

При данной конфигурации нейронной сети точность модели составила 92.37%.

2. Код реализации:

# definition

model = Sequential()

# adding layers

model.add(Conv2D(6, kernel\_size=(3, 3), input\_shape=(28, 28, 1), padding='same', activation='relu'))

model.add(MaxPool2D())

model.add(Flatten())

model.add(Dense(120, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['acc'])

В данной архитектуре нейронной сети удалось добиться 94.09% точности.

3. Код реализации:

# definition

model = Sequential()

# adding layers

model.add(Conv2D(6, kernel\_size=(3, 3), input\_shape=(28, 28, 1), padding='same', activation='relu'))

model.add(MaxPool2D())

model.add(Conv2D(16, kernel\_size=(3, 3), padding='same', activation='relu'))

model.add(MaxPool2D())

model.add(Flatten())

model.add(Dense(120, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(84, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['acc'])

В LeNet5 удалось достичь максимальной точности – 94.5%.

4. Максимальные точности распределились следующими: первая лабораторная (логистическая регрессия) – 88.67%, вторая лабораторная работа (глубокая нейронная сеть) – 93.09%, третья лабораторная работа (свёрточная нейронная сеть) – 94.51%. Данные различия можно объяснить следующим образом. Минимальная точность, как и предполагалось, получилась у модели логистической регрессии. Связано это с тем, что эта модель достаточно простая, и она просто физически не может уловить всех тонкостей/паттернов в изображении. Более лучшие результаты показала глубокая нейронная сеть, потому что данная модель намного сложнее, относительно той же логистической регрессии, и максимальный процент классификаций составил 93.09% (хоть в условии и было сказано, что максимальный процент с использованием глубокой нейронной сети составил 97.1%, у меня не получилось достичь этого результата – я думаю, что это связано с тем, что модель, достигшая этих результатов, имела большее количество слоёв и нейронов в каждом слое, и дольше обучалась). Лидером, конечно же, в этой тройке оказалась свёрточная нейронная сеть. Обусловлено это тем, что эти нейронные сети изначально разрабатывалтсь для эффективного распознавания образов.

**Программный код:**

from keras.models import Sequential

from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout

import pandas as pd

import numpy as np

import tensorflow as tf

import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

import sys, os

sys.path.append(os.path.abspath(os.path.join('..', 'loaders')))

from load\_not\_mnist import load

X\_train, y\_train, X\_val, y\_val, X\_test, y\_test = load(conv=True)

with tf.device('/cpu:0'):

# definition

model = Sequential()

# adding layers

model.add(Conv2D(6, kernel\_size=(3, 3), input\_shape=(28, 28, 1), padding='same', activation='relu'))

model.add(MaxPool2D())

model.add(Conv2D(16, kernel\_size=(3, 3), padding='same', activation='relu'))

model.add(MaxPool2D())

model.add(Flatten())

model.add(Dense(120, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(84, activation='relu'))

model.add(Dropout(0.25))

model.add(Dense(10, activation='softmax'))

model.compile(optimizer='rmsprop', loss='categorical\_crossentropy', metrics=['acc'])

# train

model.fit(X\_train, np.array(pd.get\_dummies(y\_train)),

epochs=5,

verbose=1,

validation\_data=(X\_val, np.array(pd.get\_dummies(y\_val))))

score, acc = model.evaluate(X\_test, np.array(pd.get\_dummies(y\_test)), verbose=1)

print(score, acc)