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

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Искусственные нейронные сети»
Тема: Распознавание рукописных символов

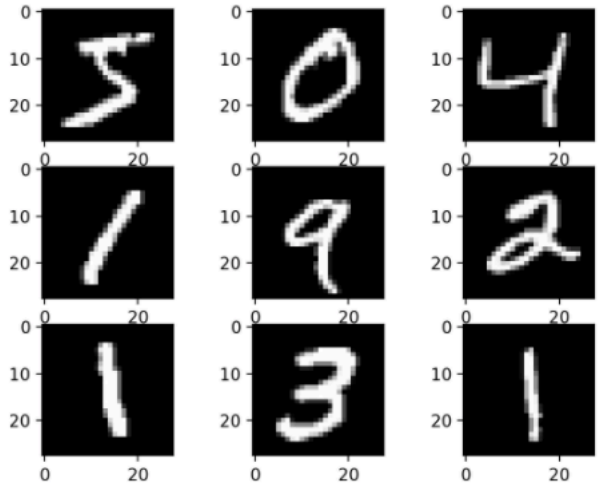
Студентка гр. 8383 _____ Кормщикова А.О.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Реализовать классификацию черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9).



Задачи.

- Ознакомиться с представлением графических данных
- Ознакомиться с простейшим способом передачи графических данных нейронной сети
- Создать модель
- Настроить параметры обучения
- Написать функцию, позволяющая загружать изображение пользователи и классифицировать его
-

Ход работы

Набор данных MNIST уже входит в состав Keras в форме набора из четырех массивов Numpy: 60,000 изображений для обучения и 10,000 изображений для тестирования.

Исходные изображения представлены в виде массивов чисел в интервале [0, 255]. Перед обучением их необходимо преобразовать так, чтобы все значения оказались в интервале [0, 1].

Была задана базовая архитектура сети

```
model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128)
```

Точность на контрольном наборе составила 0.977

Были рассмотрены различные оптимизаторы с изменением из параметров.

Оптимизатор ADAM

Параметры по умолчанию:

learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07, amsgrad=False

Параметры	Точность
По умолчанию	0.977
learning_rate=0.01	0.968
learning_rate=0.1	0.864
amsgrad = True	0.979
learning_rate=0.01, amsgrad = True	0.969
learning_rate=0.1, amsgrad = True	0.878

Из полученных результатов видно, что увеличение скорости обучения ведет к потере точности, включение параметра amsgrad немного повышает точность.

Оптимизатор SGD

Параметры по умолчанию:

learning_rate=0.01, momentum=0.0, nesterov=False

Параметры	Точность
По умолчанию	0.911
learning_rate=0.001	0.833
learning_rate=0.1	0.960
nesterov = True	0.907
learning_rate=0.001, nesterov = True	0.824
learning_rate=0.1, Nesterov = True	0.960
learning_rate=0.1, momentum=0.2	0.963
learning_rate=0.1, momentum=0.4	0.969
learning_rate=0.1, momentum=0.9	0.978

В данном случае увеличение скорости обучение ведет к улучшению точности. Модификация с помощью метода Нестерова не дала значительных изменений в какую-либо сторону. Изменение параметра momentum, который ускоряет градиентный спуск в соответствующем направлении и гасит колебания, позволило увеличить точность.

Оптимизатор RMSprop

Параметры по умолчанию:

learning_rate=0.001, rho=0.9, momentum=0.0, epsilon=1e-07, centered=False

Параметры	Точность
По умолчанию	0.978
learning_rate=0.01	0.976
learning_rate=0.1	0.892
centered = True	0.975
learning_rate=0.1, centered = True	0.840
momentum=0.2	0.978
momentum=0.9	0.973

Видно, что наилучшими значениями являются значения по умолчанию, изменение остальных значений лишь ухудшало результат, либо оставляло его неизменным.

Оптимизаторы Adadelta, Adagrad, Adamax, Nadam, Ftrl

Оптимизатор	Точность
Adadelta learning_rate=0.001 (По умолчанию)	0.508
Adadelta learning_rate=0.1	0.938
Adagrad learning_rate=0.001 (По умолчанию)	0.871
Adagrad learning_rate=0.1	0.975
Adamax learning_rate=0.001 (По умолчанию)	0.967
Adamax learning_rate=0.1	0.954
Nadam learning_rate=0.001 (По умолчанию)	0.976
Nadam learning_rate=0.1	0.880
Ftrl learning_rate=0.001 (По умолчанию)	0.209
Ftrl learning_rate=0.1	0.966

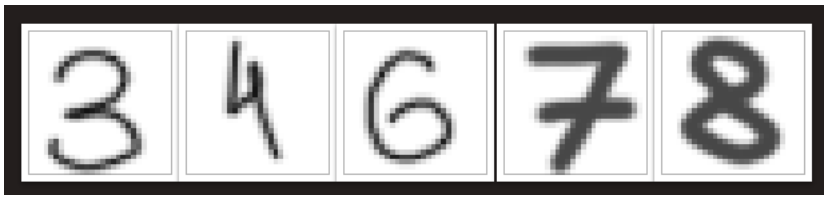
Видно, что параметр скорости обучения играет очень важную роль и изменение его может к значительным улучшением.

Для данной модели наилучший результат показал оптимизатор Adam с параметрами amsgrad = True. Также хорошие результаты показывали оптимизаторы SGD с параметрами learning_rate=0.1, momentum=0.9, и RMSprop со значениями по умолчанию.

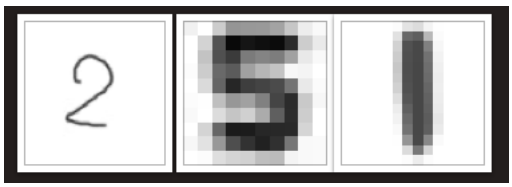
Самая высокая точность, которую удалось достичь 0.979.

Была написана функция load_img, которая позволяет загружать собственное изображение, в функции размер подгоняется под 28x28 пикселей.

Для классификации были созданы изображения размера 28x28



А также изображения иных размеров: 2 - увеличена, 5, 1 имееют меньшие размеры.



Результат классификации:

```
test_acc: 0.9789999723434448
enter path to image or 1 to end
3.png
It is  3
enter path to image or 1 to end
4.png
It is  1
enter path to image or 1 to end
6.png
It is  8
enter path to image or 1 to end
7.png
It is  3
enter path to image or 1 to end
8.png
It is  8
enter path to image or 1 to end
2.png
It is  2
enter path to image or 1 to end
5.png
It is  5
enter path to image or 1 to end
1.png
It is  1
enter path to image or 1 to end
```

ИНС ошиблась в 3х случаях.

Выводы.

Во время выполнения лабораторной работы была реализована классификация черно-белых изображений рукописных цифр (28x28) по 10 категориям (от 0 до 9). Были проанализированы результаты, от разных настроек оптимизаторов. Наилучший результат показал Adam с параметром amsgrad=true.

ПРИЛОЖЕНИЕ А

```
import tensorflow as tf
from keras.utils import to_categorical
from tensorflow.keras.layers import Dense, Activation, Flatten
from tensorflow.keras.models import Sequential
from PIL import Image
import numpy as np
from pathlib import Path


def load_img(path):
    img = Image.open(path)
    img = img.convert('L')
    img = img.resize((28, 28))
    return 1 - np.array(img) / 255.0


mnist = tf.keras.datasets.mnist
(train_images, train_labels),(test_images, test_labels) =
mnist.load_data()
#
# plt.imshow(train_images[0],cmap=plt.cm.binary)
# plt.show()
# print(test_images[0].shape)
# print(test_images[0])

train_images = train_images / 255.0
test_images = test_images / 255.0

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model = Sequential()
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))
optimizer = tf.keras.optimizers.Adam(amsgrad=True)
model.compile(optimizer=optimizer, loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=128)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test_acc:', test_acc)

path = ""
```

```
while(1):
    print("enter path to image or 1 to end")
    path = input()
    if(path == "1"):
        break
    path = Path(path)
    if (path.exists() == False):
        print("File not exist!")
        continue
    img = load_img(path)
    predict = model.predict(np.array([img]))
    print("It is ",np.argmax(predict))
    # plt.imshow(img, cmap=plt.cm.binary)
    # plt.show()
```