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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Искусственные нейронные сети»

Тема: «Распознование рукописных символов»

Студент гр. 7381 _____ Кортев Ю. В.

Преподаватель _____ Жукова Н.А.

Санкт-Петербург

2020

Цель работы.

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

Порядок выполнения работы.

- 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

Требования

. Найти архитектуру сети, при которой точность классификации будет не менее 95%
. Исследовать влияние различных оптимизаторов, а также их параметров, на процесс обучения
. Написать функцию, которая позволит загружать пользовательское изображение не из датасета

Ход работы.

Обычно изображения имеют три измерения: высоту, ширину и цвет. Даже при том, что черно-белые изображения (как в наборе данных MNIST) имеют только один канал цвета и могли бы храниться в двумерных тензорах, по соглашениям тензоры с

изображениями всегда имеют три измерения, где для черно-белых изображений отводится только один канал цвета. Соответственно, пакет со 128 черно-белыми изображениями, имеющими размер 256 × 256, можно сохранить в тензоре с формой (128, 256, 256, 1), а пакет со 128 цветными изображениями — в тензоре с формой (128, 256, 256, 3). В отношении форм тензоров с изображениями существует два соглашения: соглашение канал следует последним (используется в TensorFlow) и соглашение канал следует первым (используется в Theano). Фреймворк машинного обучения TensorFlow, разработанный компанией Google, отводит для цвета последнюю ось: (образцы, высота, ширина, цвет). А библиотека Theano отводит для цвета ось, следующую сразу за осью пакетов: (образцы, цвет, высота, ширина). Если следовать соглашению, принятому в Theano, предыдущие примеры тензоров будут иметь форму (128, 1, 256, 256) и (128, 3, 256, 256). Фреймворк Keras поддерживает оба формата.

В данной работе используется простейший способ передачи графических данных на вход модели. Т.к. изображение чернобелое, то каждый пиксел кодируется константой в диапозоне [0-255], таким образом шаблон из выборки с формой (28, 28,) можно свести к форме (28*28,). В работе эту функцию выполняет слой Flatten, но его можно заменить предложением x=x.reshape((60000,28*28)).

Создание модели показано в листинге 1.

```
model=Sequential()
model.add(Flatten(input_shape=(28,28,)))
model.add(Dense(256,activation='relu'))
model.add(Dense(100,activation='relu'))
model.add(Dense(10,activation='softmax'))

num_ep=10

model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
h=model.fit(train_x,train_y,epochs=num_ep,batch_size=128,validation_split=0.15,verbose=0)
```

Листинг 1 - Создание и обучение модели

На рисунке 1 показана точность полученной модели обученной с помощью оптимизатора adam. На рисунке 2 - ее потери.

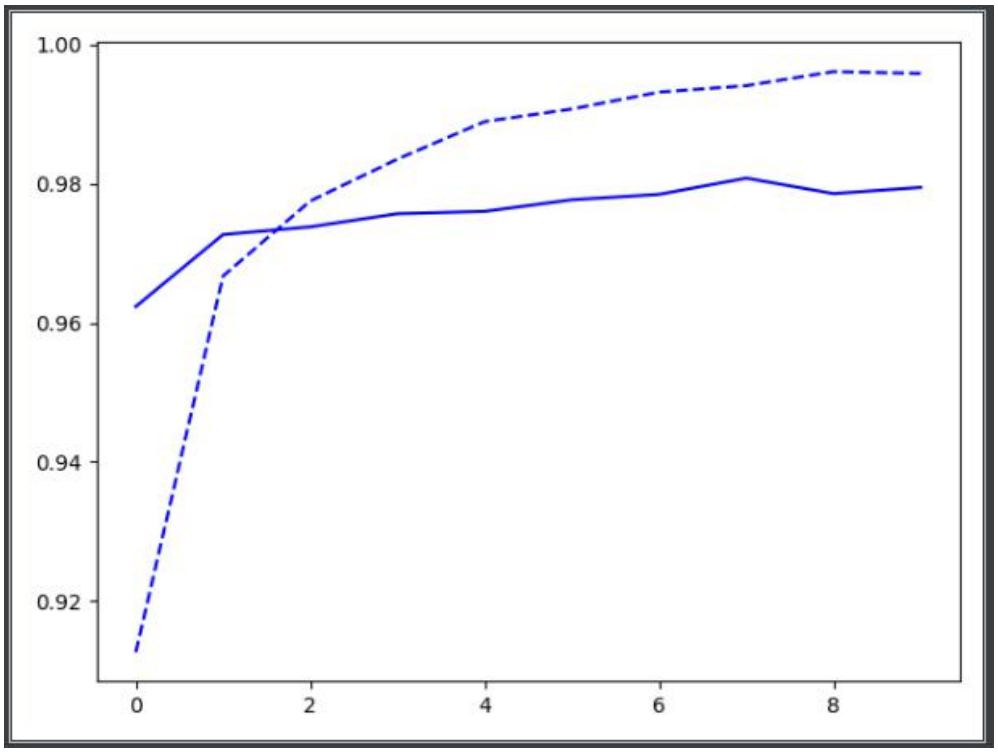

Рисунок 1 - Точность модели adam (--train)

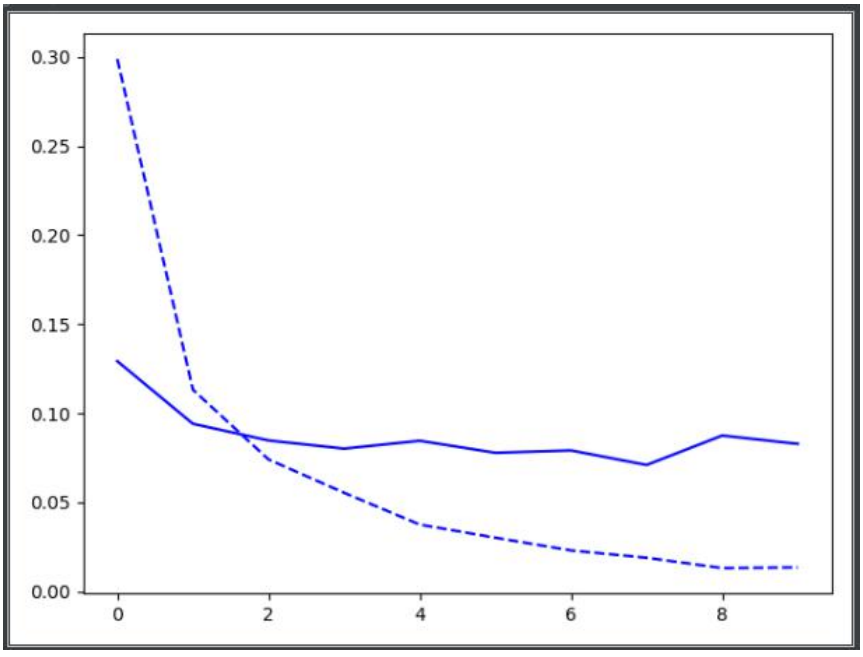

Рисунок 2 - Потери модели adam

Попробую теперь, используя ту же архитектуру обучить модель с помощью оптимизатора rmsprop. Инициализация модели показана в листинге 2,

используются параметры по умолчанию. Графики точности и потерь новой модели показаны на рисунках 3-4.

```
model.compile(optimizer=optimizers.RMSprop(),loss='categorical_crossentropy',metrics=['accuracy'])
```

Листинг 2 - Оптимизатор rmsprop

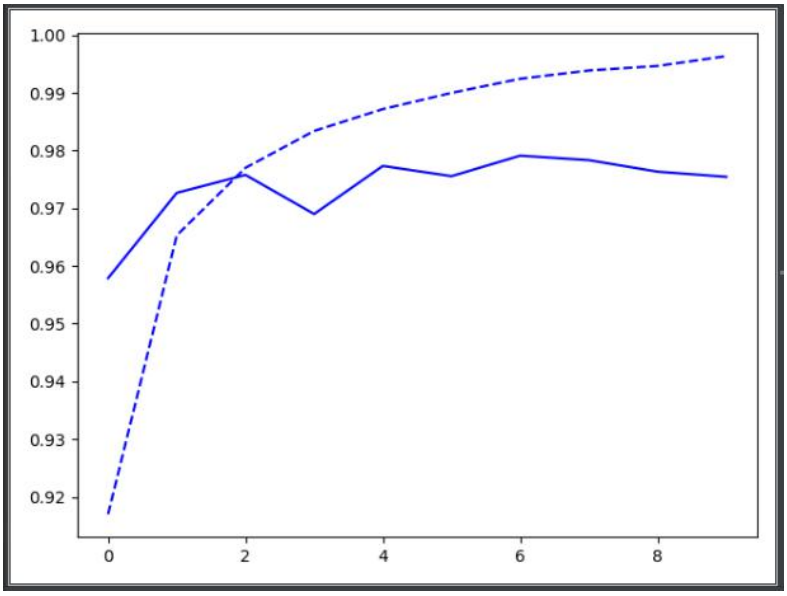

Рисунок 3 - Точность модели rmsprop

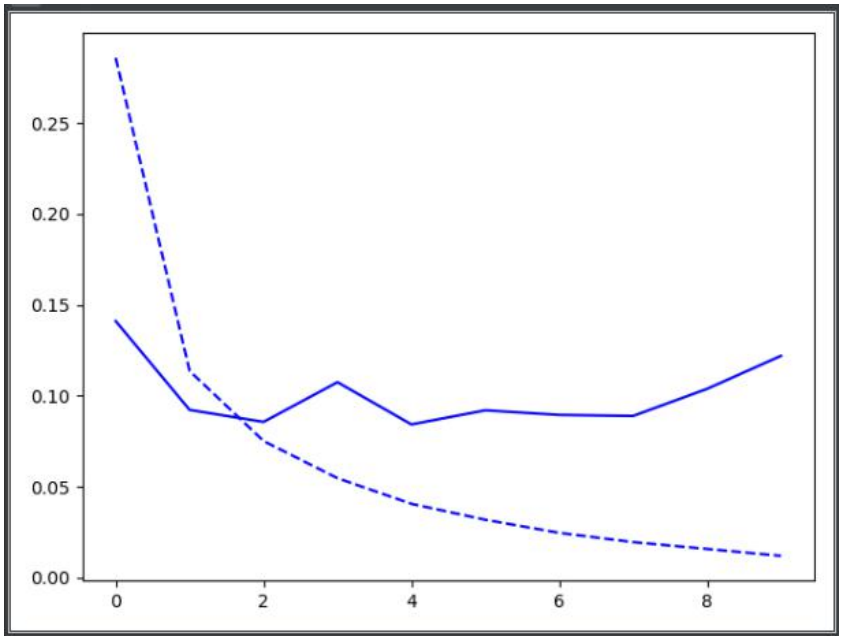

Рисунок 4 - Потери модели rmsprop

Как видно новая модель менее устойчива к переобучению. Попробую уменьшить скорость обучения в 10 раз и увеличить количество эпох, график точности полученной модели показан на рисунке 5.

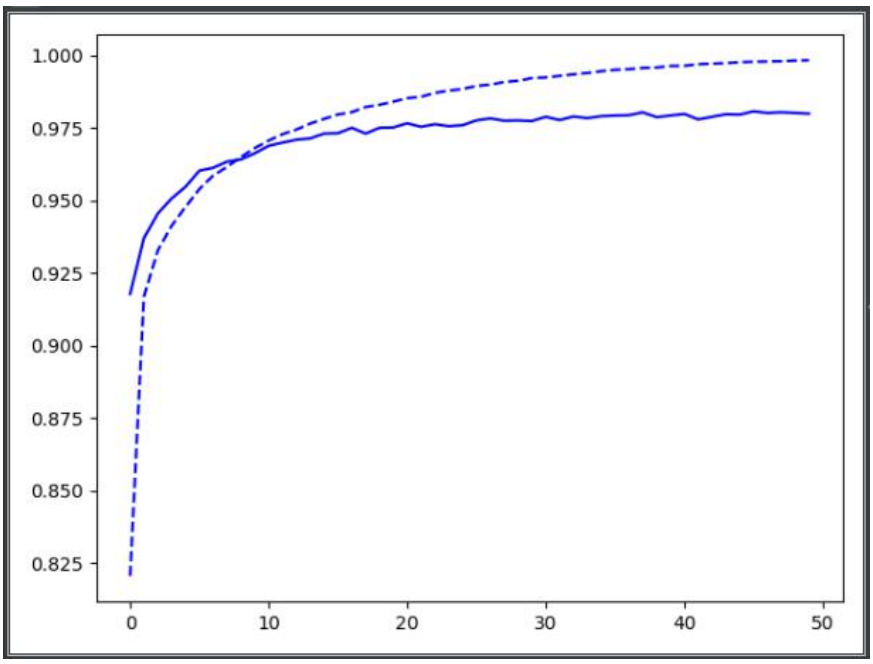

Рисунок 5- Точность модели rmsprop(learning_rate=0.0001)

Как видно переобучения не наступает, но итоговая точность не превышает модель с adam, при этом затрачивается большее количество эпох. Можно сделать вывод, что для данной задачи предпочтительнее использовать оптимизатор adam.

В итоге данного модуля, полученная модель сохраняется в файл model.h5. Для проверки пользовательских изображений создан модуль predict_mnis.py, он загружает модель из файла model.h5, получает путь к изображению, загружает его и приводит к виду, который может принять модель, возвращает предсказание модели с помощью метода predict_classes.

Модуль показан в листинге 3, пример работы показан на рисунке 6-7.

```
def load_img(path):
    im = Image.open(path).convert('L')
    im = im.resize((28, 28))
    im=np.array(im)
    im=255-im
    im=im/255
    im = np.expand_dims(im,axis=0)
    return im

assert argv[1]
sample=load_img(argv[1])

model=load_model('model.h5')
#print(sample)
```

```
print('На картинке цифра {}'.format(model.predict_classes(sample)[0]))
```

Листинг 3 - Модуль для обработки пользовательского изображения

Рисунок 6 - Образец на вход модулю

```
(venv) C:\Users\green\.PyCharm2019.3\config\scratches>python predict_mnis.py sample5.png
Using TensorFlow backend.
На картинке цифра 8
```

Рисунок 7 - Результат модуля

Выводы.

В ходе выполнения лабораторной работы исследовано влияние различных оптимизаторов, а также их параметров, на процесс обучения. Построена и обучена модель предсказания рукописных цифр. Написан модуль для обработки пользовательского изображения.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

```
import numpy as np
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.layers import Dense, Flatten
from keras.models import Sequential
from matplotlib import pyplot as plt
from keras.datasets import mnist
from keras.utils import to_categorical
from keras import optimizers


(train_x, train_y),(test_x, test_y)=mnist.load_data()

train_x=train_x/255.0
test_x=test_x/255.0

train_y=to_categorical(train_y)
test_y=to_categorical(test_y)

model=Sequential()
model.add(Flatten(input_shape=(28,28,)))
model.add(Dense(256,activation='relu'))
model.add(Dense(100,activation='relu'))
model.add(Dense(10,activation='softmax'))

num_ep=50

model.compile(optimizer=optimizers.RMSprop(learning_rate=0.0001),loss=
'categorical_crossentropy',metrics=['accuracy'])
h=model.fit(train_x,train_y,epochs=num_ep,batch_size=128,validation_sp
lit=0.15,verbose=0)

model.save('model.h5')

plt.plot(range(num_ep),h.history['val_accuracy'],'b',label='Val acc')
plt.plot(range(num_ep),h.history['accuracy'],'b--',label='train_acc')
plt.show()

plt.plot(range(num_ep),h.history['val_loss'],'b',label='Val loss')
plt.plot(range(num_ep),h.history['loss'],'b--',label='train_loss')
plt.show()
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ПОЛЬЗОВАТЕЛЬСКОГО ВВОДА

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
from keras.models import load_model
from sys import argv
from PIL import Image
import numpy as np


def load_img(path):
    im = Image.open(path).convert('L')
    im = im.resize((28, 28))
    im=np.array(im)
    im=255-im
    im=im/255
    im = np.expand_dims(im,axis=0)
    return im

assert argv[1]
sample=load_img(argv[1])

model=load_model('model.h5')
#print(sample)

print('На картинке цифра {}'.format(model.predict_classes(sample)[0]))
```