Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет

имени В.Ф.Уткина»

Кафедра «Электронные вычислительные машины»

Отчет по практическому занятию №7

на тему

«Проектирование свёрточной нейронной сети для распознавания рукописных чисел»

по дисциплине  
**«Машинное обучение»**

Выполнили:

Студенты группы №140

Бригада 7

Сафаров Д. А.

Тимохин Е. М.

Проверила:  
ас. Панина И.С.

**Цель работы**

Научиться составлять модели свёрточных нейронных сетей для обработки изображений.

**Задание 1.**

Реализовать свёрточную нейронную сеть для распознавания рукописных чисел.

**Задание 2.**

Изменить архитектуру так, чтобы процент ошибки был не больше 0.9.

**Практическая часть**

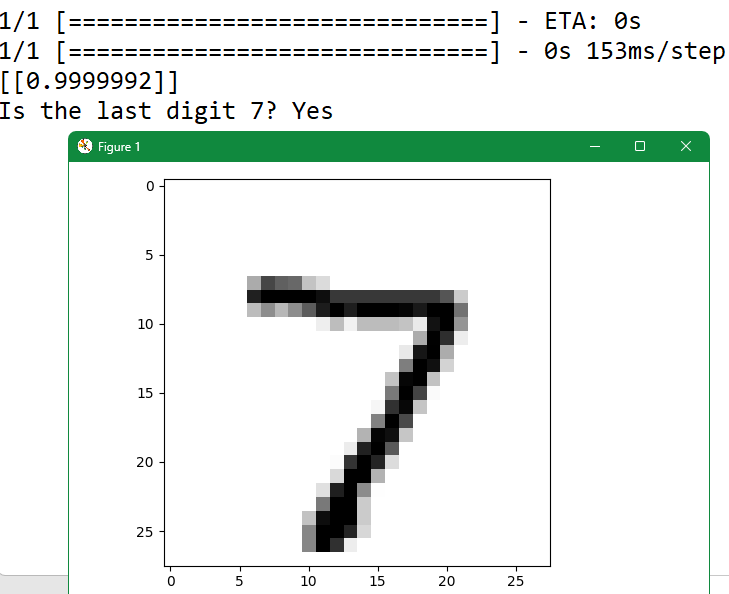
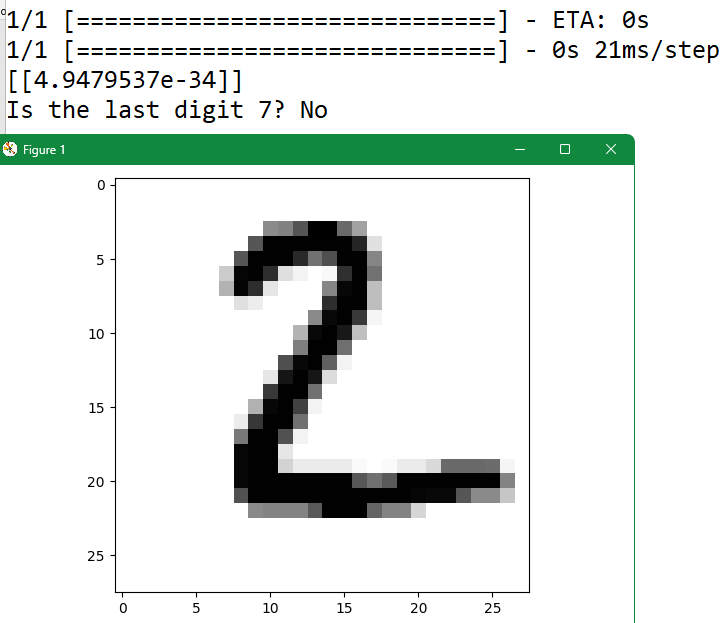
Код программы представлен в приложении А. Результат её работы представлен на рисунке 1.

Рисунок 1 – Результат работы нейросети

Приложение А. Текст программы

import numpy as np

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Flatten

from tensorflow.keras.layers import Conv2D, MaxPooling2D,BatchNormalization

from tensorflow.keras.utils import to\_categorical

from keras import backend as K

import matplotlib.pyplot as plt

K.set\_image\_data\_format('channels\_last')

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

x\_train = x\_train.reshape(x\_train.shape[0], 28, 28, 1).astype('float32')

x\_test = x\_test.reshape(x\_test.shape[0], 28, 28, 1).astype('float32')

X\_train = x\_train / 255

X\_test = x\_test / 255

#y\_train = to\_categorical(y\_train)

#y\_test = to\_categorical(y\_test)

#num\_classes = y\_test.shape[1]

yes="Yes"

no="No"

y\_train\_last\_digit\_is\_seven = np.array([yes if str(label)[-1] == '7' else no for label in y\_train])

y\_test\_last\_digit\_is\_seven = np.array([yes if str(label)[-1] == '7' else no for label in y\_test])

'''

def baseline\_model():

model = Sequential()

model.add(Conv2D(32, (5, 5), input\_shape=(28, 28, 1), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Conv2D(32, (5, 5), activation='relu'))

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

# model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

'''

'''

Добавление сверточных слоев: Увеличим количество сверточных слоев и нейронов в них для извлечения более высокоуровневых признаков.

Увеличение количества плотных слоев: Добавим дополнительные плотные слои для более глубокого обучения.

Добавление слоев Batch Normalization и Dropout: Эти слои помогут улучшить стабильность и предотвратить переобучение модели

'''

def improved\_model():

model = Sequential()

# Первый сверточный слой с 32 фильтрами размером 3x3 и функцией активации ReLU

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)))

# Добавление слоя Batch Normalization после сверточного слоя для стабилизации обучения

model.add(BatchNormalization())

# Второй сверточный слой с 64 фильтрами размером 3x3 и функцией активации ReLU

model.add(Conv2D(64, (3, 3), activation='relu'))

# Пуллинг слой для уменьшения размерности карт признаков

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Dropout слой для предотвращения переобучения

model.add(Dropout(0.25))

# Третий сверточный слой с 128 фильтрами размером 3x3 и функцией активации ReLU

model.add(Conv2D(128, (3, 3), activation='relu'))

# Добавление слоя Batch Normalization после сверточного слоя для стабилизации обучения

model.add(BatchNormalization())

# Четвёртый сверточный слой с 256 фильтрами размером 3x3 и функцией активации ReLU

model.add(Conv2D(256, (3, 3), activation='relu'))

# Пуллинг слой для уменьшения размерности карт признаков

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Dropout слой для предотвращения переобучения

model.add(Dropout(0.25))

# Выравнивание карт признаков перед подачей на плотный слой

model.add(Flatten())

# Первый плотный слой с 512 нейронами и функцией активации ReLU

model.add(Dense(512, activation='relu'))

# Добавление слоя Batch Normalization после плотного слоя для стабилизации обучения

model.add(BatchNormalization())

# Dropout слой для предотвращения переобучения

model.add(Dropout(0.5))

# Второй плотный слой с 256 нейронами и функцией активации ReLU

model.add(Dense(256, activation='relu'))

# Добавление слоя Batch Normalization после плотного слоя для стабилизации обучения

model.add(BatchNormalization())

# Выходной слой с одним нейроном и функцией активации sigmoid для бинарной классификации

model.add(Dense(1, activation='sigmoid'))

# Компиляция модели с использованием бинарной кросс-энтропии в качестве функции потерь и оптимизатора Adam

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

model = improved\_model()

model.fit(x\_train, (y\_train\_last\_digit\_is\_seven == yes).astype(int), batch\_size=32, epochs=10, validation\_split=0.2)

count\_numb = 20

for n in range(count\_numb):

x = np.expand\_dims(x\_test[n], axis=0)

res = model.predict(x)

print(res)

predicted\_label = yes if res >= 0.5 else no

print('Is the last digit 7? ' + str(predicted\_label))

plt.imshow(x\_test[n], cmap=plt.cm.binary)

plt.show()

scores = model.evaluate(x\_test, y\_test, verbose=0)

print("CNN Error: %.2f%%" % (100 - scores[1] \* 100))# процент точности модели