Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет имени

Гагарина Ю.А.»

Институт машиностроения и материаловедения и техники (ИММТ)

Кафедра **«Техническая механика и мехатроника»**

**Отчёт по практическим занятиям**

по дисциплине: **«Искусственный интеллект и нейрокомьютеры»**

Выполнил:   
 студент группы

б-МХРТ-41  
Куприн Максим Сергеевич

Проверил:

кандидат технических наук,

доцент кафедры

«Техническая механика и   
мехатроника»  
Трефилов Пётр Анатольевич

**Саратов 2023**

Оглавление

[Задание 1. Входной контроль 4](#_Toc154610729)

[Цель работы 4](#_Toc154610730)

[Основная часть 4](#_Toc154610731)

[Тестовый пример 9](#_Toc154610732)

[Заключение 12](#_Toc154610733)

[Приложение 13](#_Toc154610734)

[Задание 2. Разработка нейронной сети для моделирования логических элементов 14](#_Toc154610735)

[Цель работы 14](#_Toc154610736)

[Основная часть 14](#_Toc154610737)

[Моделирование элементов AND и OR: 14](#_Toc154610738)

[Моделирование элементов XOR: 17](#_Toc154610739)

[Заключение 18](#_Toc154610740)

[Приложение 19](#_Toc154610741)

[Задание 3. Разработка сети поиска минимального и максимального значения уровня сигнала 20](#_Toc154610742)

[Цель работы 20](#_Toc154610743)

[Основная часть 20](#_Toc154610744)

[Заключение 22](#_Toc154610745)

[Приложение 23](#_Toc154610746)

[Задание 4. Сравнительный анализ вычислительных устройств 24](#_Toc154610747)

[Цель работы 24](#_Toc154610748)

[Основная часть 24](#_Toc154610749)

[Закючение 27](#_Toc154610750)

[Задание 5. Двоичные сумматоры 27](#_Toc154610751)

[Цель работы 27](#_Toc154610752)

[Основная часть 27](#_Toc154610753)

[Тестовый пример 29](#_Toc154610754)

[Заключение 29](#_Toc154610755)

[Задание 6. Разработка аппаратного FPGA – ускорителя 30](#_Toc154610756)

[Цель работы 30](#_Toc154610757)

[Основная часть 30](#_Toc154610758)

[Заключение 31](#_Toc154610759)

[Задание 7. Фильтр Гаусса 32](#_Toc154610760)

[Цель работы 32](#_Toc154610761)

[Основная часть 32](#_Toc154610762)

[Тестовый пример: 35](#_Toc154610763)

[Заключение: 35](#_Toc154610764)

[Приложение 35](#_Toc154610765)

[Задание 8. Выделение границ фигур 36](#_Toc154610766)

[Цель работы 36](#_Toc154610767)

[Основная часть 36](#_Toc154610768)

[Заключение 37](#_Toc154610769)

[Приложение 38](#_Toc154610770)

[Список источников 39](#_Toc154610771)

# Задание 1. Входной контроль

## Цель работы

Определение уровня готовности студента к дальнейшему обучению; разработка алгоритма поиска минимумов и максимумов массива числовых значений.

## Основная часть

1. С применением языка высокого уровня Python [1] была разработана программа, блок-схема представлена на рисунке 1.1:

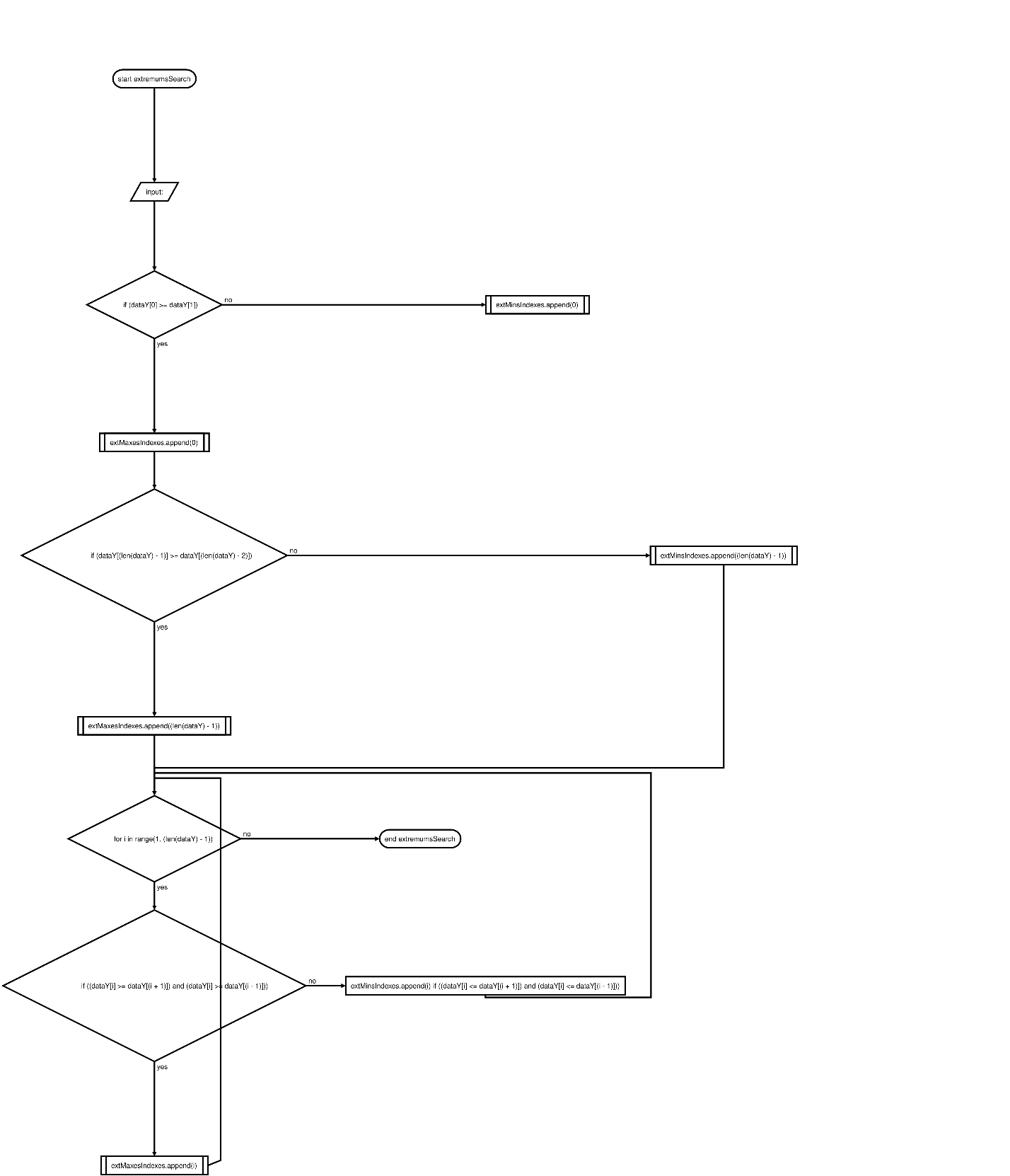


Рис.1.1 Блок-схема

1. Разработанная программа собрана в исполняемый файл и приложена к отчету
2. С использованием алгоритма определены следующие значения минимумов и максимумов (рис. 1.2):

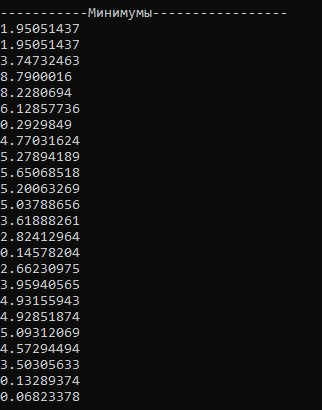
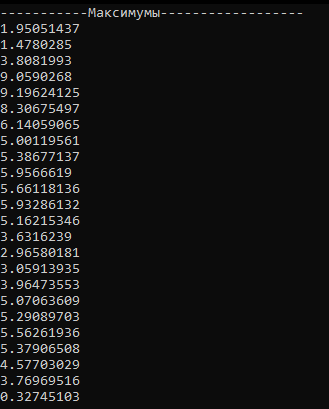


Рис.1.2 Минимумы/ массива

С применением библиотеки для графического изображения данных «matplotlib» [2] построен график зависимости максимумы значения элемента массива от его порядкового номера (рис. 1.3):

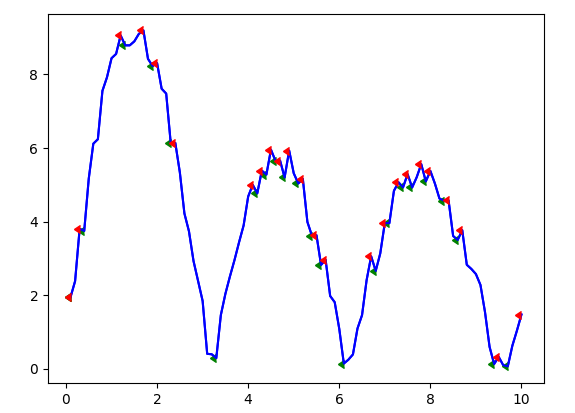


Рис. 1.3 График с локальными минимумами и максимумами

1. В массиве данных находится 24 локальных максимума и 24 локальных минимума, итого 48 точек.
2. Исходный код программы представлен в репозитории проекта на сайте GitHub [6]:

import matplotlib.pyplot as plot

import matplotlib

import os

matplotlib.use('TkAgg')

filename = "inputTest\_AINC2023.txt"

dataT = []

dataY = []

extMaxesIndexes = [0]

extMinsIndexes = [0]

#reads data from file

def reading():

f = open(filename,'r')

lines = f.readlines()

lines.pop(0)

for line in lines:

a = line.strip().split(', ')

a[0] = float(a[0])

a[1] = float(a[1])

dataT.append(a[0])

dataY.append(a[1])

f.close()

#serachs extremums values

def extremumsSearch():

if(dataY[0]>=dataY[1]):

extMaxesIndexes.append(0)

else:

extMinsIndexes.append(0)

if(dataY[len(dataY) - 1]>=dataY[len(dataY) - 2]):

extMaxesIndexes.append(len(dataY) - 1)

else:

extMinsIndexes.append(len(dataY) - 1)

for i in range(1, len(dataY) - 1):

if(dataY[i] >= dataY[i+1] and dataY[i] >= dataY[i-1]):

extMaxesIndexes.append(i)

elif (dataY[i] <= dataY[i+1] and dataY[i] <= dataY[i-1]):

extMinsIndexes.append(i)

# Press the green button in the gutter to run the script.

if \_\_name\_\_ == '\_\_main\_\_':

reading()

extremumsSearch()

print('-----------Максимумы------------------')

for i in extMaxesIndexes:

print(dataY[i])

print()

print('--------------------------------------')

print('-----------Минимумы-----------------')

for i in extMinsIndexes:

print(dataY[i])

print()

print('------------------------------------')

print("Количество максимумов")

print(len(extMaxesIndexes))

print("Количество минимумов")

print(len(extMinsIndexes))

plot.plot(dataT, dataY, '-b', markevery=extMinsIndexes, marker=8, markerfacecolor='green', markeredgecolor='green')

plot.plot(dataT, dataY, '-b', markevery=extMaxesIndexes, marker=8, markerfacecolor='red', markeredgecolor='red')

plot.show()

input("Нажмите Enter для выхода")

## Тестовый пример

t y(t)

0.10000000, 1.95051437

0.20000000, 2.38532732

0.30000000, 3.80819930

0.40000000, 3.74732463

0.50000000, 5.17074382

0.60000000, 6.11867831

0.70000000, 6.23962744

0.80000000, 7.55450001

0.90000000, 7.92233169

1.00000000, 8.44039387

1.10000000, 8.55720334

1.20000000, 9.05902680

1.30000000, 8.79000160

1.40000000, 8.79055036

1.50000000, 8.89790526

1.60000000, 9.10232785

1.70000000, 9.19624125

1.80000000, 8.42472166

1.90000000, 8.22806940

2.00000000, 8.30675497

2.10000000, 7.61177869

2.20000000, 7.47952341

2.30000000, 6.12857736

2.40000000, 6.14059065

2.50000000, 5.33676736

2.60000000, 4.22373097

2.70000000, 3.73831600

2.80000000, 2.93086120

2.90000000, 2.38965382

3.00000000, 1.84596550

3.10000000, 0.41004087

3.20000000, 0.39755891

3.30000000, 0.29298490

3.40000000, 1.46435948

3.50000000, 2.04185816

3.60000000, 2.51512115

3.70000000, 2.95893539

3.80000000, 3.44179024

3.90000000, 3.89868391

4.00000000, 4.68387007

4.10000000, 5.00119561

4.20000000, 4.77031624

4.30000000, 5.38677137

4.40000000, 5.27894189

4.50000000, 5.95666190

4.60000000, 5.65068518

4.70000000, 5.66118136

4.80000000, 5.20063269

4.90000000, 5.93286132

5.00000000, 5.31567084

5.10000000, 5.03788656

5.20000000, 5.16215346

5.30000000, 3.99410105

5.40000000, 3.61888261

5.50000000, 3.63162390

5.60000000, 2.82412964

5.70000000, 2.96580181

5.80000000, 1.98148409

5.90000000, 1.81187669

6.00000000, 1.08780856

6.10000000, 0.14578204

6.20000000, 0.25076990

6.30000000, 0.39340108

6.40000000, 1.09784970

6.50000000, 1.46083686

6.60000000, 2.40869152

6.70000000, 3.05913935

6.80000000, 2.66230975

6.90000000, 3.13819522

7.00000000, 3.96473553

7.10000000, 3.95940565

7.20000000, 4.83690113

7.30000000, 5.07063609

7.40000000, 4.93155943

7.50000000, 5.29089703

7.60000000, 4.92851874

7.70000000, 5.20727886

7.80000000, 5.56261936

7.90000000, 5.09312069

8.00000000, 5.37906508

8.10000000, 5.03497626

8.20000000, 4.62731367

8.30000000, 4.57294494

8.40000000, 4.57703029

8.50000000, 3.61158990

8.60000000, 3.50305633

8.70000000, 3.76969516

8.80000000, 2.82848818

8.90000000, 2.71632745

9.00000000, 2.57689685

9.10000000, 2.27982998

9.20000000, 1.54142203

9.30000000, 0.59947597

9.40000000, 0.13289374

9.50000000, 0.32745103

9.60000000, 0.10034048

9.70000000, 0.06823378

9.80000000, 0.62933138

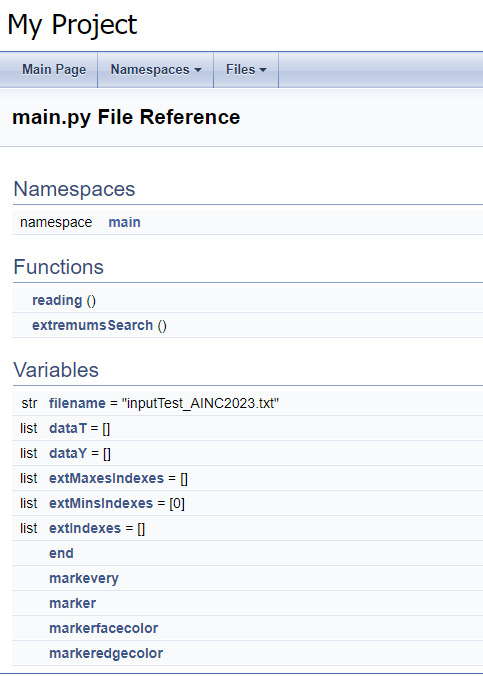
9.90000000, 1.03758064

10.00000000, 1.47802850

## Заключение

В ходе выполнения практической работы разработана программа, определяющая значения минимумов и максимумов,

## Приложение



Приложение 1.1. Задокументированный файл при помощи Doxygen

# Задание 2. Разработка нейронной сети для моделирования логических элементов

## Цель работы

Изучение возможности применения нейронных сетей для моделирования логических элементов OR, AND, XOR; разработка нейронной сети для моделирования логических элементов.

## Основная часть

Для создания и обучения нейронной сети использованы:

* Python – язык программирования высокого уровня [1];
* TensorFlow – библиотека для машинного обучения [3];
* Keras – высокоуровневый API для TensorFlow [4];
* NumPy – библиотека для работы с многомерными массивами, матрицами, векторами [5].

Создан файл программы main.py, загружены и подключены библиотеки (рис. 1):

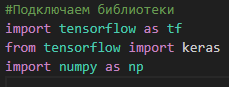


Рис. 2.1. Библиотеки, используемые в проекте

## Моделирование элементов AND и OR:

Выбрана модель нейронной сети с последовательными слоями Sequential. Сеть состоит из слоя с одним нейроном. Нейрон имеет два входных значения и одно выходное. Код, описывающий модель представлен на рисунке 2.

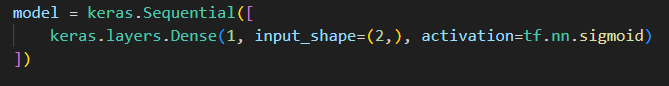


Рис. 2.2. Программный код, описывающий модель

Настроены параметры для обучения модели. Выбрана loss функция (функция потерь) “binary\_crossentropy”, подходящая для задач бинарной классификации. Для оптимизации процесса обучения выбран алгоритм Adam (Adaptive Moment Estimation).

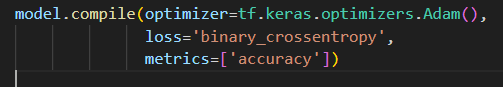


Рис. 2.3. Настройка параметров обучения нейронной сети

Данные для обучения представляют таблицы истинности. Таблица истинности для элемента AND:

Таблица 2.1

|  |  |
| --- | --- |
| Входные значения | Выходное значение |
| 00 | 0 |
| 01 | 0 |
| 10 | 0 |
| 11 | 1 |

Таблица истинности для элемента OR:

Таблица 2.2

|  |  |
| --- | --- |
| Входные значения | Выходное значение |
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 1 |

В программе данные хранятся в двух массивах: train\_input для входных значений, train\_output - для выходных.

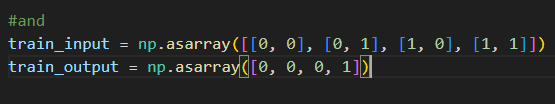


Рис. 2.4. Данные для обучения AND

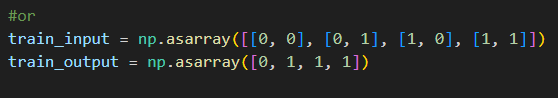


Рис. 2.5. Данные для обучения OR

Нейронная сеть прошла 4000 циклов обучения на наборе обучающих данных. На 912 цикле сеть достигла точности, близкой к 100% (рис. 6).

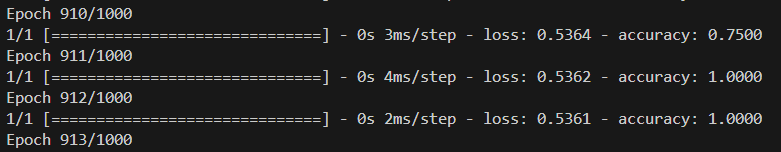


Рис. 2.6. Результаты обучения AND

В процессе обучения сформированы веса, представленные на рис. 7.



Рис. 2.7. Веса для элемента AND

На 2800 цикле сеть достигла точности, близкой к 100% (рис. 8).

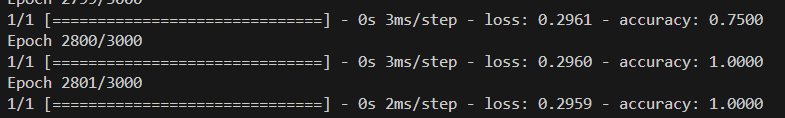


Рис. 2.8. Результаты обучения OR

В процессе обучения сформированы веса, представленные на рис. 9.



Рис. 2.9. Веса для элемента AND

## Моделирование элементов XOR:

Для моделирования элемента исключающее “или” выбрана двухслойная модель. Первый слой содержит 4 нейрона с двумя входами, функция активации - гиперболический тангенс. Второй - 1 нейрон, функция активации - сигмоида.

Данные для обучения - таблица истинности для XOR:

Таблица 3

|  |  |
| --- | --- |
| Входные значения | Выходное значение |
| 00 | 0 |
| 01 | 1 |
| 10 | 1 |
| 11 | 0 |

Данные для обучения представлены на (рис. 10):

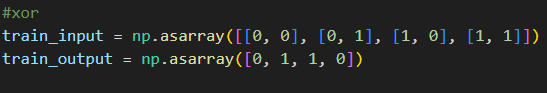


Рис. 2.10. Данные для обучения XOR

Нейронная сеть прошла 500 циклов обучения, на 239 цикле была достигнута точность, близка к 100% (рис. 11).

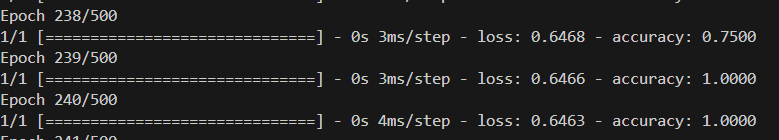


Рис. 2.11. Результаты обучения XOR

В процессе обучения сформированы веса, представленные на рис. 12.

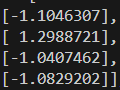
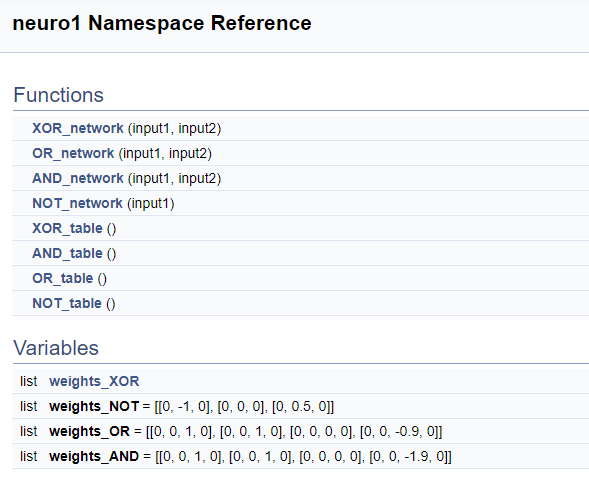


Рис. 2.12. Веса для элемента XOR

## Заключение

В ходе выполнения работы изучены возможности применения нейронных сетей для моделирования логических элементов OR, AND, XOR; разработаны нейронные сети для моделирования логических элементов.

## Приложение



Приложение 2.1. Автоматическая документация с помощью Doxygen

# Задание 3. Разработка сети поиска минимального и максимального значения уровня сигнала

## Цель работы

Изучение возможности применения нейронных сетей для поиска минимального значения массива чисел; разработка нейронной сети для поиска минимума/максимума в массиве чисел.

## Основная часть

Разработана нейронная сеть, состоящая из четырех полносвязных слоев. На вход сети подается массив из 10 нормализованных значений (рис. 1).

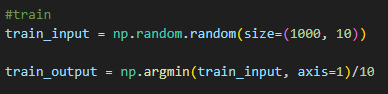


Рис. 3.1. Библиотеки, используемые в проекте

Первые два слоя содержат 10 нейронов с функцией активации ReLU, второй слой, третий слой - 5 нейронов, четверты - 1 нейрон. На выходе нейронной сети одно значение - позиция минимального элемента (рис. 2).

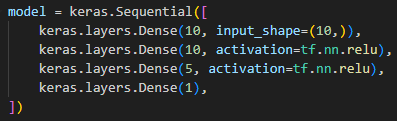


Рис. 3.2. Библиотеки, используемые в проекте

Функция потерь “MeanAbsoluteError”- среднее значение абсолютного отклонения результата от эталонного (рис. 3).

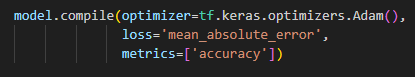


Рис. 3.3. Библиотеки, используемые в проекте

Для создания и обучения нейронной сети использованы:

* Python – язык программирования высокого уровня [1];
* TensorFlow – библиотека для машинного обучения [3];
* Keras – высокоуровневый API для TensorFlow [4];
* NumPy – библиотека для работы с многомерными массивами, матрицами, векторами [5].

В процессе выполнения работы были испытаны различные конфигурации нейронной сети, функции активаторы, форматы данных для обучения. Наилучшая точность получена при обучении сети в 50000 циклах, время обучения составило более 5 часов, достигнута точность предсказания минимального значения - 15%.

При обучении сетей других конфигураций, уменьшении циклов, форматов данных средняя точность составила 9-11%.

Среднее время на определение минимального значения составило - 15 мс. Функция “numpy.argmin”, применяемая для генерации обучающих данных выполняет поиск минимального значения 1000000 массивов из 10 элементов за 2.146 мс (рис. 4).



Рис. 3.4. Время выполнения 1млн. поисков минимального элемента массива

Минимальное среднее отклонение в процессе обучения - 0.1227.

Решения, близкие к верным представлены на рисунке 5.

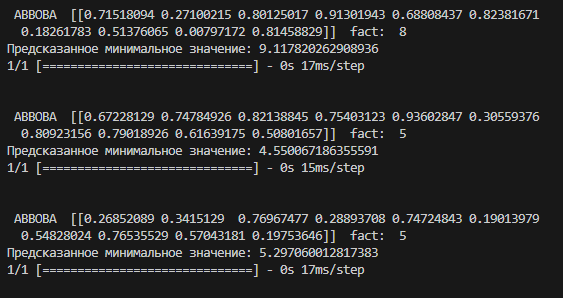
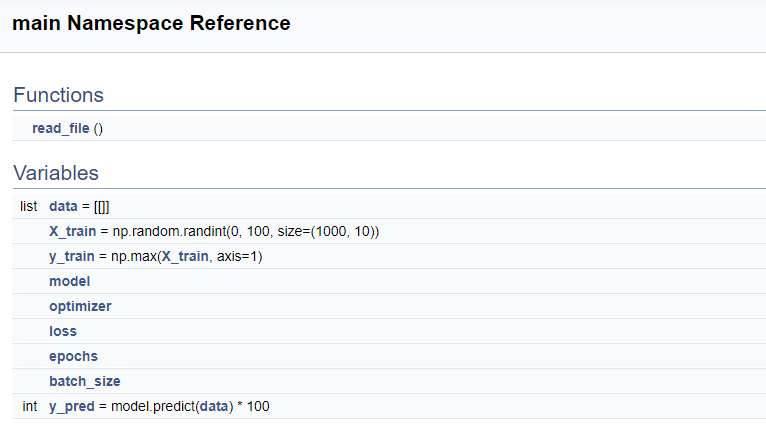


Рис. 3.5. Удачные предсказания

## Заключение

В ходе выполнения работы создана нейронная сеть для поиска минимального значения массива чисел; определена низкая эффективность применения нейронных сетей для поиска минимальных значений в сравнении с алгоритмическими способами нахождения минимума.

## Приложение

****Приложение 3.1. Автоматическая документация с помощью Doxygen

# Задание 4. Сравнительный анализ вычислительных устройств

## Цель работы

Проведение сравнительного анализа вычислительных технологий: CPU, GPU, TPU, NPU, FPGA; составить сравнительную таблицу характеристик.

.

## Основная часть

В ходе выполнения работы проведен анализ вычислительных технологий, составлена сравнительная таблица характеристик, включающая характеристики: задачи, общая структура устройства, отличные характеристики, мат. обеспечение, производитель.

Таблица 4.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | CPU | GPU | TPU | NPU | FPGA |
| Задачи | Решение вычислительных задач, обработка данных | Графические вычисления, научные расчеты | Машинное обучение, обработка изображений | Искусственный интеллект, распознавание образов | Различные вычисления и обработка сигналов |
| Общая структура устройства | Множество ядер (обычно 2-64), кэш данных, кэш команд | Множество ядер (от 100 до 10 тыс), специальные блоки для задач графики | Множество ядер (от 100 до 10 тыс), MXU, блок активации, сеть связи, объединённый буфер | Специализированный чип, (встроенный в SoC) | Логические элементы, программируемые логические элементы |
| Отличные хар-ки | Высокая частота и универсальность | Высокая параллельность и вычислительная мощность | Высокая параллельность и вычислительная мощность | Низкое энергопотребление, высокая скорость работы | Гибкость в настройке и оптимизация под задачу |
| Мат. обеспечение | Общий набор инструкций, оптимизирован для | Оптимизирован для работы с матрицами | Оптимизирован для работы с тензорами | Оптимизирован для работы с тензорами | Программируемый под задачу |
| Производитель | Intel, AMD, Байкал Электроникс, Микрон (компания, Россия), Apple, ARM, Cyrix, Huawei,  IBM, Ingenic Semiconductor,  MediaTek, Nvidia, Qualcomm, RDC Semiconductor, Samsung | NVIDIA, AMD, Intel | Google | Huawei, Apple, Samsung | Xilinx, Intel, Microchip |

1. Центральный процессор (CPU)

- Задачи: ЦПУ предназначен для общего назначения, включая выполнение операционной системы, приложений, обработка данных и выполнение различных задач.

- Общая структура: ЦПУ состоит из нескольких ядер (обычно от 2 до 64), каждое из которых способно выполнять инструкции.

- Отличительные характеристики: ЦПУ обладает высокой частотой и универсальностью, что позволяет ему эффективно выполнять широкий спектр задач.

2. Графический процессор (GPU)

- Задачи: ГП предназначен для графических вычислений, научных расчетов, обработки видео и графики, а также для выполнения параллельных задач.

- Общая структура: ГП содержит множество ядер (от сотен до тысяч), обеспечивающих параллельную обработку данных.

- Отличительные характеристики: ГП обладает высокой параллельностью и вычислительной мощностью, что делает его эффективным для выполнения сложных вычислительных задач.

3. Тензорный процессор (TPU)

- Задачи: TPU предназначен для машинного обучения, обработки изображений, выполнения операций с тензорами и других задач, связанных с искусственным интеллектом.

- Общая структура: TPU также содержит множество ядер (от сотен до тысяч), оптимизированных для работы с тензорами и выполнения операций машинного обучения.

- Отличительные характеристики: TPU обладает высокой параллельностью и вычислительной мощностью, а также низким энергопотреблением, что делает его эффективным для задач машинного обучения.

4. Нейропроцессор (NPU)

- Задачи: NPU предназначен для распознавания образов, обработки изображений, выполнения операций искусственного интеллекта и других задач, связанных с обработкой данных.

- Общая структура: NPU является специализированным чипом, встроенным в систему на кристалле (SoC) и оптимизированным для выполнения операций искусственного интеллекта.

- Отличительные характеристики: NPU обладает высокой скоростью работы и оптимизирован для выполнения задач распознавания образов и обработки изображений.

5. Полевой программируемый вентильный массив (FPGA)

- Задачи: FPGA предназначен для различных вычислительных задач, включая обработку сигналов, криптографию, ускорение вычислений и другие специализированные задачи.

- Общая структура: FPGA состоит из программируемых логических элементов, которые могут быть настроены для выполнения конкретных задач.

- Отличительные характеристики: FPGA обладает гибкостью в настройке и оптимизации под конкретную задачу, что делает его эффективным для широкого спектра приложений.

## Закючение

Проведен сравнительный анализ вычислительных устройств. Каждое имеет свои преимущества и недостатки, и выбор конкретного устройства зависит от требуемых задач, поставленных перед разработчиком.

# Задание 5. Двоичные сумматоры

## Цель работы

Проведение моделирования работы многоразрядных двоичных сумматоров на примере шестнадцатиразрядного сумматора.

## Основная часть

Схема шестнадцатиразрядного составленная в САПР Компас 3D:

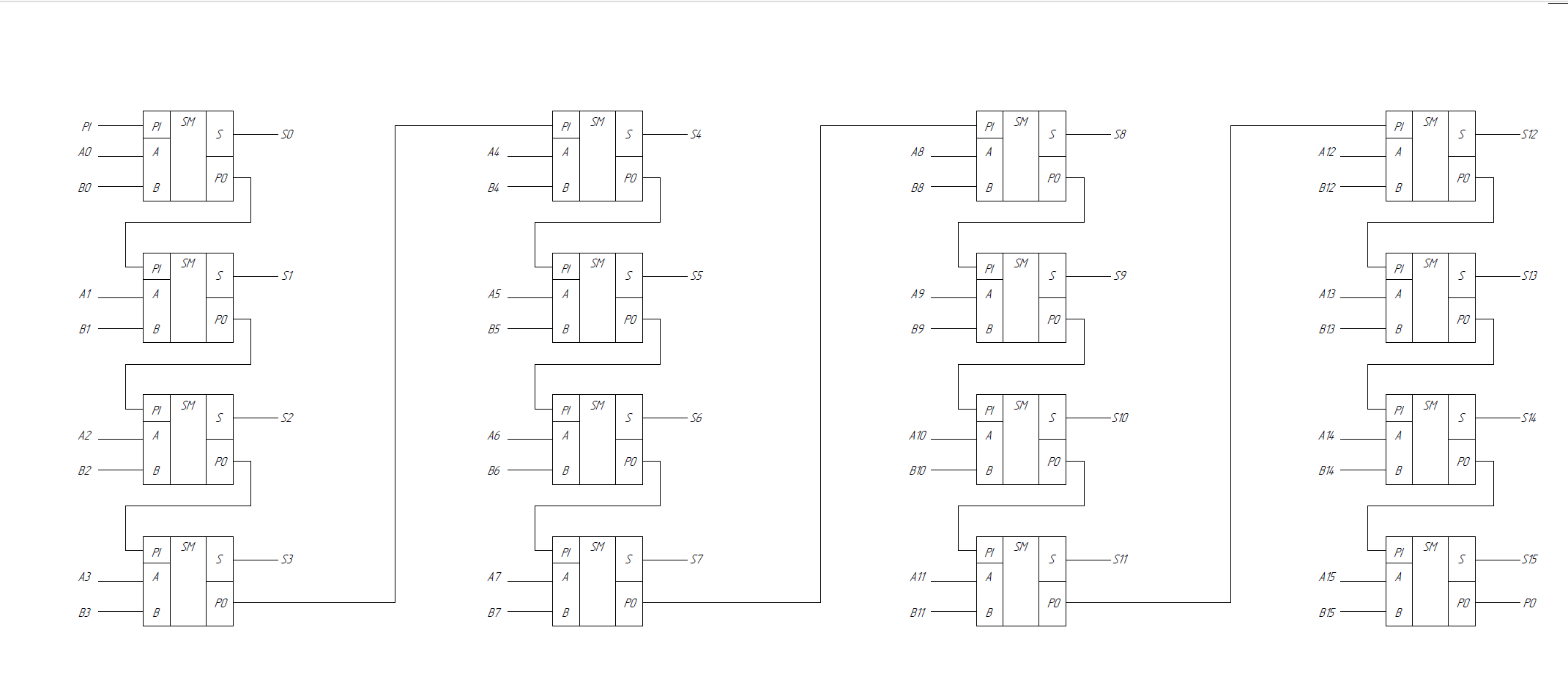


Рис. 5.1 Схема двоичного сумматора,

Схема простейшего сумматора с переносом разряда:

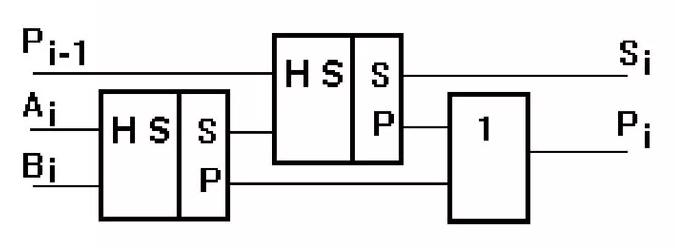


Рис. 5.2 Сумматор с переносом разряда

1. Схема сумматора, составленная в среде Quartus:

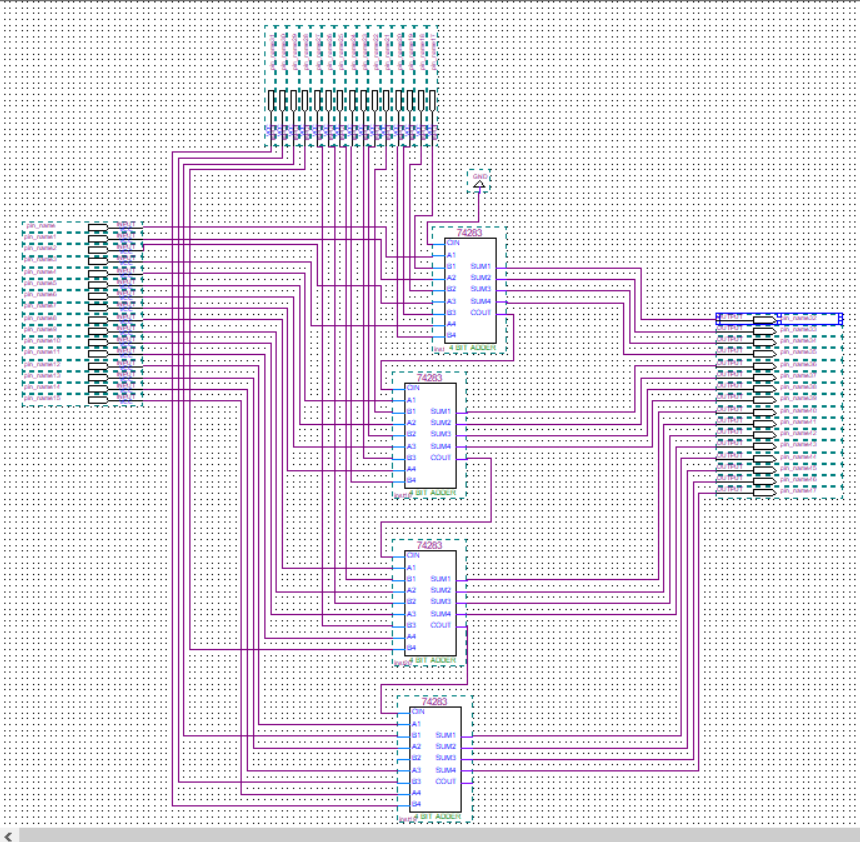


Рис. 5.3. Двоичный сумматор, составленный в среде Quartus

Для упрощения схемы используется блок четырёхразрядных сумматоров.

## Тестовый пример

0b1010101010101010 + 0b0101010011110101 == 0b1111111110011111

## Заключение

В среде Quartus был спроектирован шестнадцатиразрядный сумматор для ПЛИС EPM3032ALC44-10N.

# Задание 6. Разработка аппаратного FPGA – ускорителя

## Цель работы

Изучение принципов функционирования FPGA, разработка аппаратного ускорителя с применением FPGA

## Основная часть

Система, использующая FPGA для ускорения расчетов состоит из FPGA EPM3032ALC44 и платы разработки на основе микропроцессора серии ESP32.

Структурная схема аппаратного ускорителя:

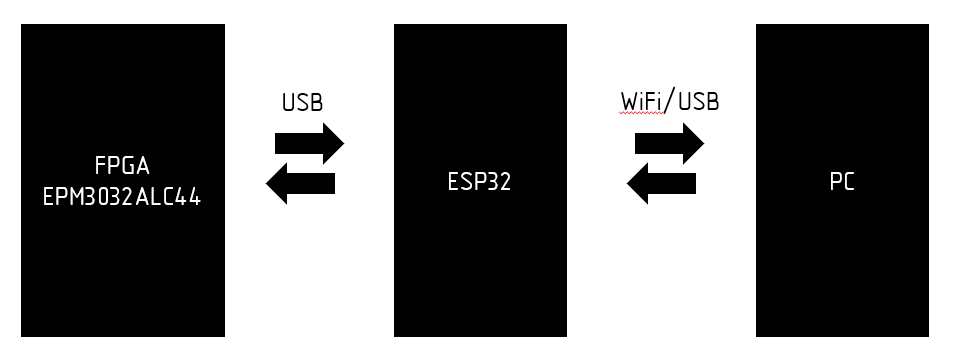


Рис. 6.1. Структурная схема аппаратного ускорителя

С использованием среды RSMA проведено моделирование работы FPGA ускорителя.

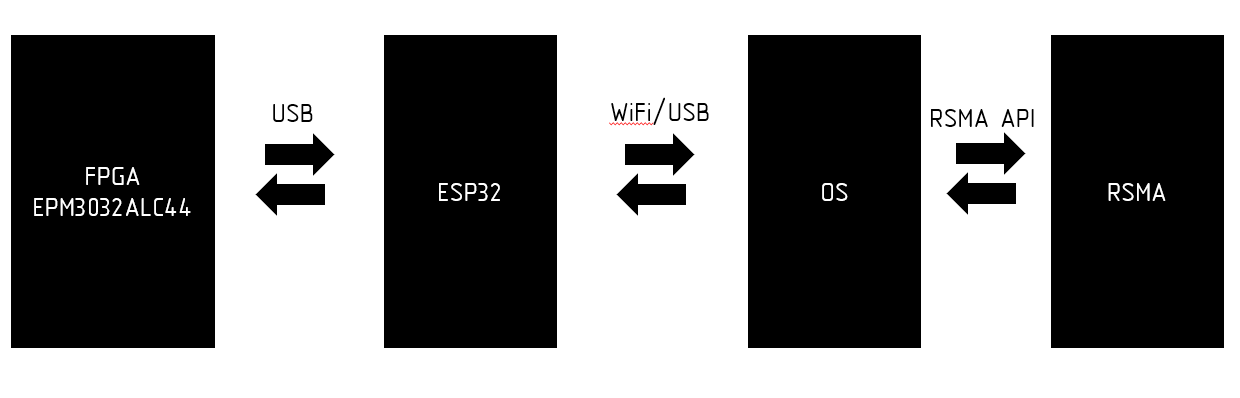


Рис. 6.2 Структурная схема ПО, моделирующая работу FPGA ускорителя

Процесс моделирования FPGA ускорителя с использованием среды RSMA включает в себя подключение компьютера к ESP32 с применением WiFi для взаимодействия с ускорителем. Затем проводится передача массивов данных на ускоритель, сбор результатов работы FPGA и их обработка. Результаты сравниваются с целевыми показателями, и в случае необходимости производится оптимизация работы ускорителя и/или доработка ПО.

## Заключение

С использованием среды RSMA было проведено моделирование FPGA ускорителя.

# Задание 7. Фильтр Гаусса

## Цель работы

Изучение принципа работы размытия по Гауссу; разработка программы для обработки изображения с применением размытия по Гауссу.

## Основная часть

Параметр σ = 10 \* 0.15 = 1.5.

C использованием Python разработан алгоритм функции размытия по Гауссу.

В коде приведенной функции gaussian\_blur, сначала создается Гауссово ядро с заданным размером и сигмой. Затем ядро нормализуется, чтобы сумма всех его элементов была равна 1. После этого применяется размытие к каждому каналу изображения с помощью операции свертки с Гауссовым ядром.

Код функции представлен на рис. 1:



Рис. 7.1. Код функции gaussian\_blur

Входное изображение и результат работы функции представлены на рис. 2:

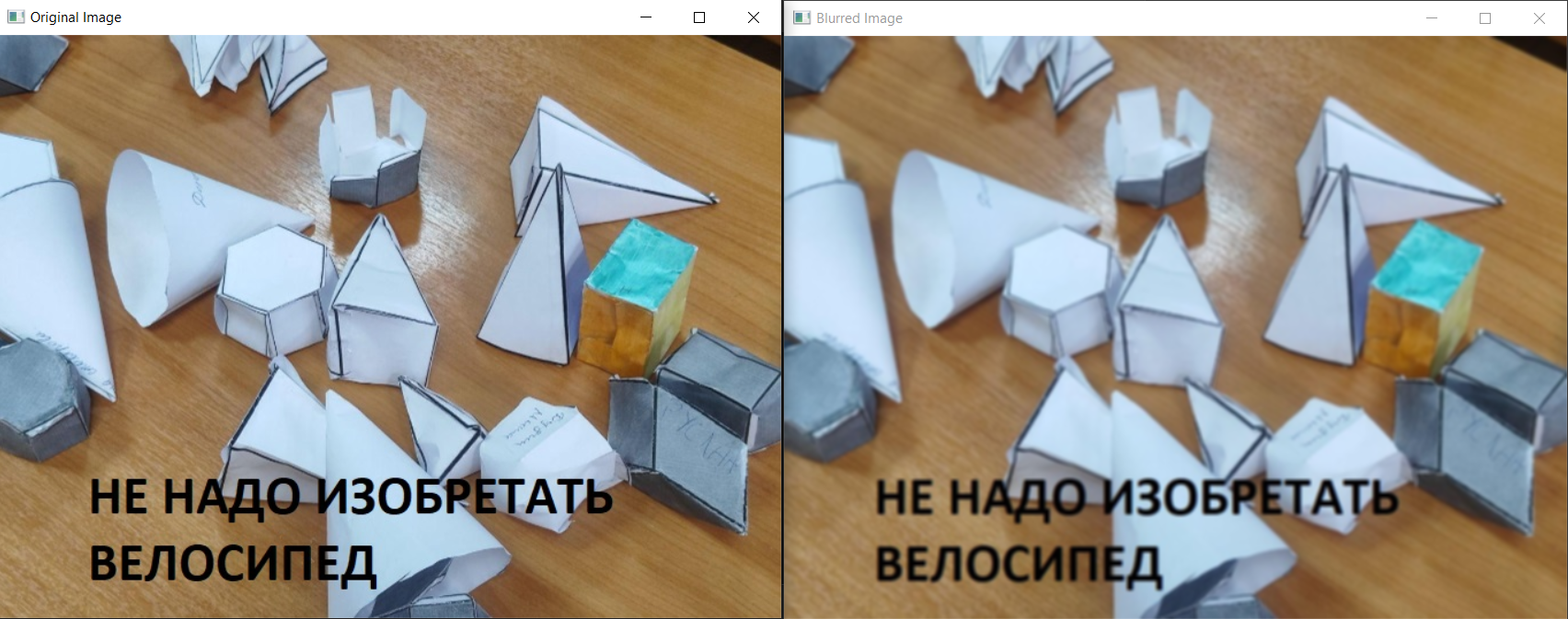


Рис. 7.2. Исходное изображение

Для оценки эффективности работы алгоритма исходное изображение было обработано функцией cv2.GaussianBlur из библиотеки OpenCV.

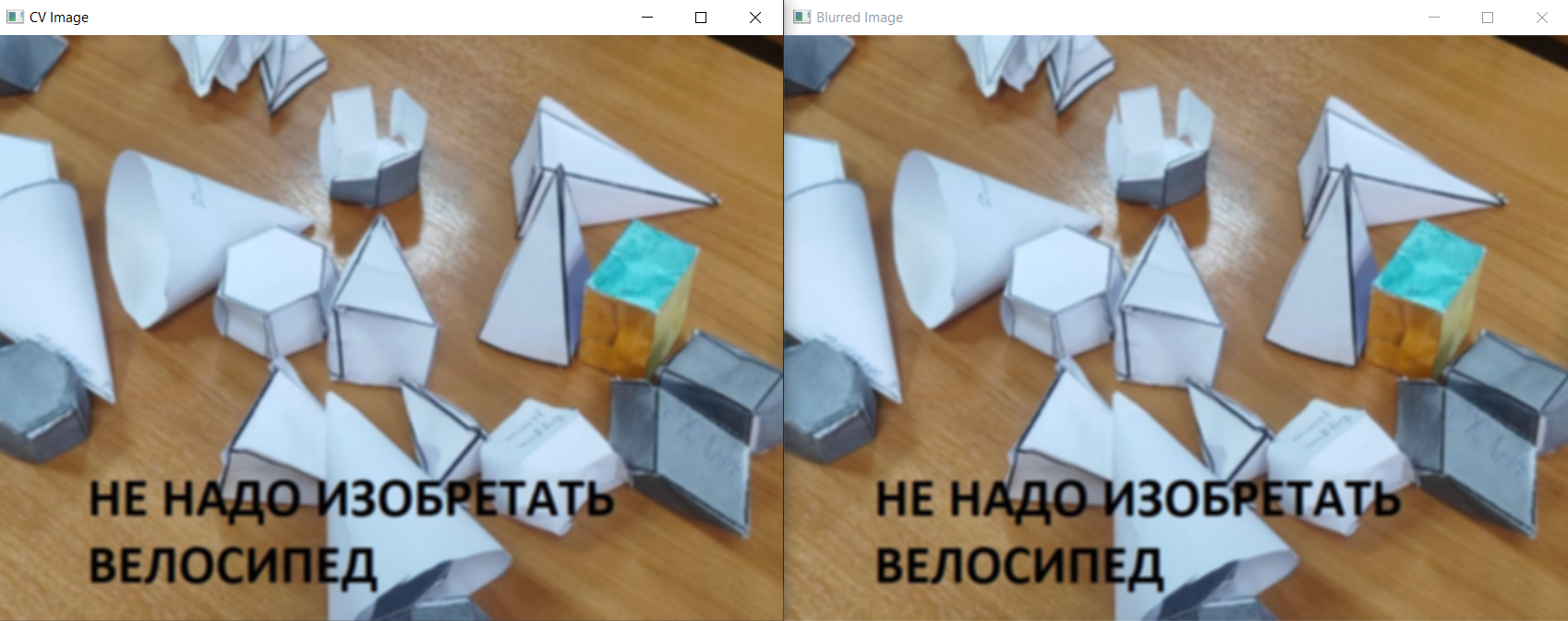


Рис. 7.3. Изображение, обработанное с функцией из OpenCV (слева), разработанным алгоритмом (справа)

При сравнении результатов применения размытия не выявлено различий между изображениями. Определена более высокая эффективность функции из библиотеки OpenCV. Разработанный алгоритм работает медленнее в 8 раз, чем алгоритм из OpenCV. Время работы представлено на рис. 4



Рис. 7.4. Время работы алгоритмов

#Подключаем библиотеки

import numpy as np

import cv2

import time

def gaussian\_blur(image, kernel\_size, sigma):

# Создание Гауссова ядра

kernel = np.fromfunction(lambda x, y: (1/(2\*np.pi\*sigma\*\*2)) \* np.exp(-((x-(kernel\_size[0]//2))\*\*2 + (y-(kernel\_size[1]//2))\*\*2)/(2\*sigma\*\*2)), kernel\_size)

# Нормализация ядра

kernel = kernel / np.sum(kernel)

# Применение размытия

blurred\_image = np.zeros\_like(image, dtype=np.float32)

for i in range(3): # Проход по каждому каналу изображения (RGB)

blurred\_image[:,:,i] = np.convolve(image[:,:,i].flatten(), kernel.flatten(), mode='same').reshape(image.shape[:2])

return blurred\_image.astype(np.uint8)

# Пример использования

t1 = time.time()

image = cv2.imread('input.jpg')

blurred\_image = gaussian\_blur(image, (5, 5), 1.5)

print("Custom time: ",time.time() - t1, "sec")

t2 = time.time()

cv\_image = cv2.GaussianBlur(image, (5, 5), 1.5)

print("OpenCV time: ",time.time() - t2, "sec")

cv2.imshow('Original Image', image)

cv2.imshow('Blurred Image', blurred\_image)

cv2.imshow("CV Image", cv\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## Тестовый пример:

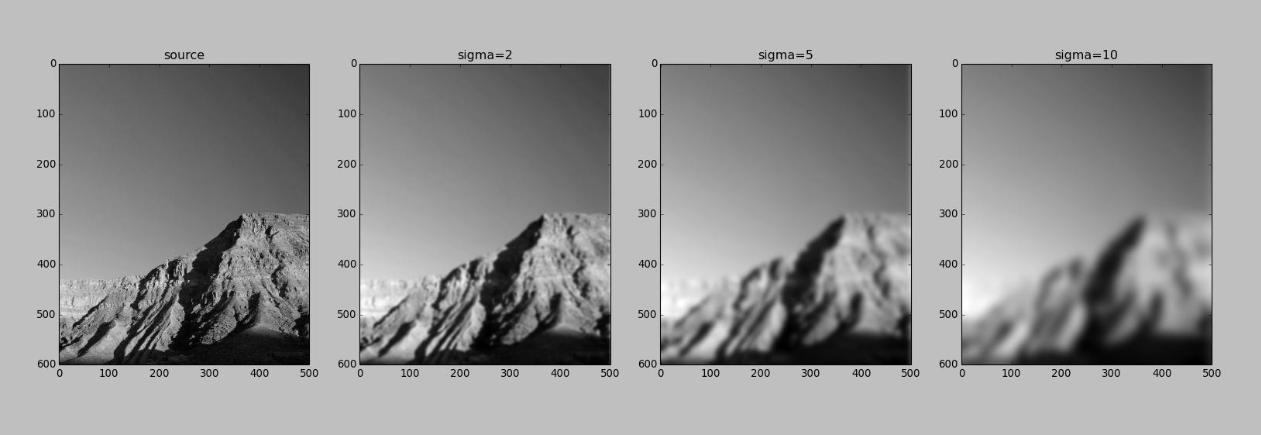
****

Рис. 7.4. Тестовый пример (разные степени размытия)

## Заключение:

В процессе выполнения работы изучен принцип работы размытия по Гауссу; разработан алгоритм и программа для обработки изображения с применением размытия по Гауссу; определена низкая эффективность работы алгоритма, в сравнении с функцией из библиотеки OpenCV

## Приложение



Приложение 7.1. Документация сделанная с помощью Doxygen

# Задание 8. Выделение границ фигур

## Цель работы

Разработка программы для выделения границ объектов на изображении.

## Основная часть

С применением Python и библиотеки OpenCV [7], разработана программа:

import cv2

import numpy as np

# Загрузка изображения

image = cv2.imread('input.jpg')

# Преобразование изображения в оттенки серого

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Применение оператора Canny для выделения границ

edges = cv2.Canny(gray, 100, 200)

# Отображение изображения с выделенными границами

cv2.imshow('Edges', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()Результат работы ПО представлен на рис 8.1.:



Рис. 8.1. Выделение контуров фигур.

Результат работы программного обеспечения зависит от параметров, указанных для фильтров (сигма, верхние и нижние границы).

**Тестовый пример:**

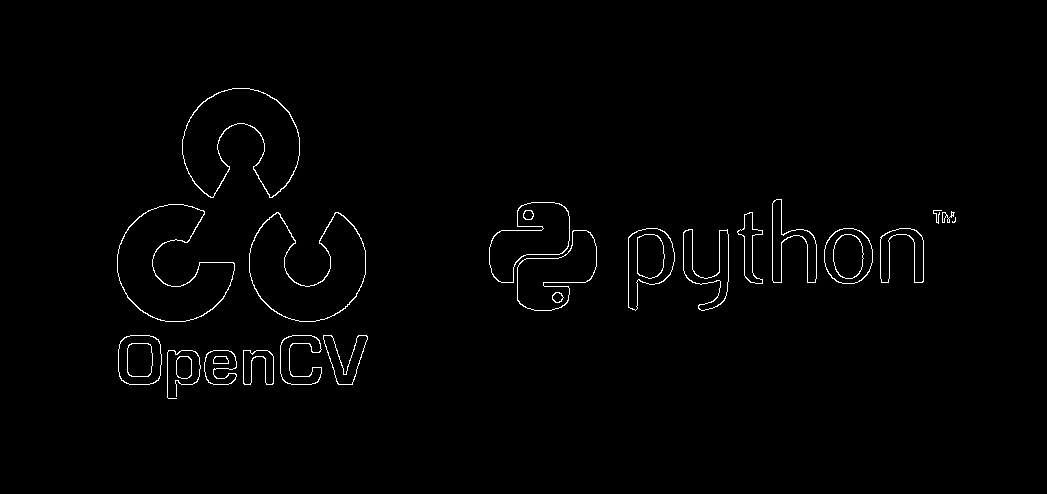
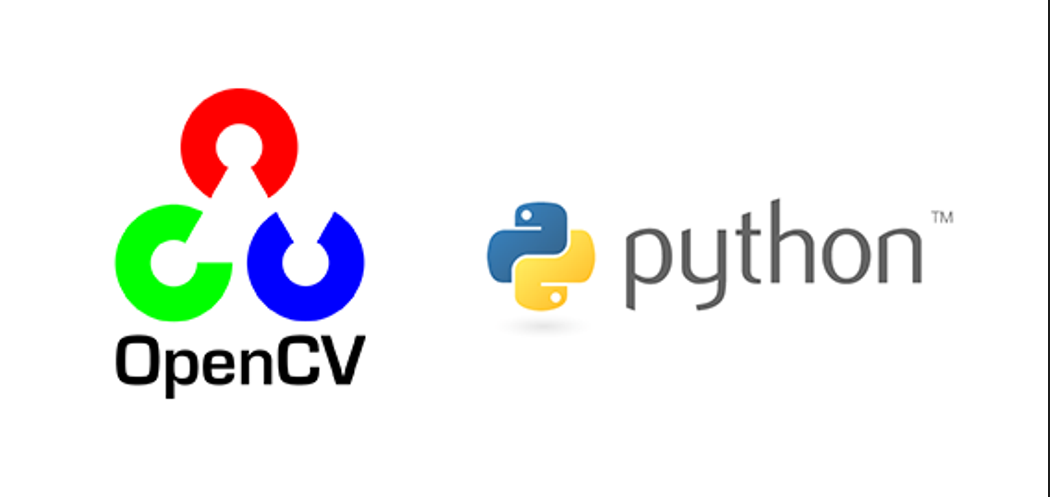
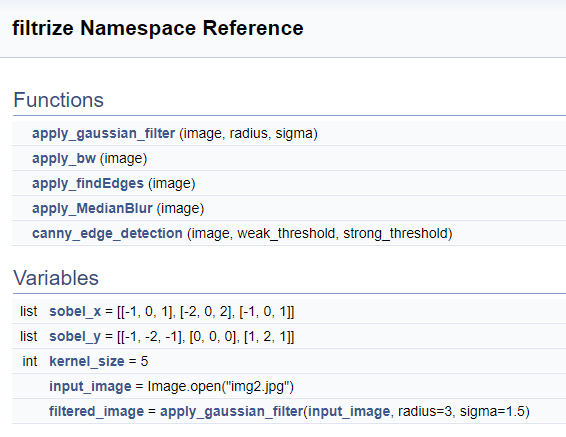
****

Рис. 8.2. Тестовый пример

## Заключение

В ходе выполнения работы разработана программа, выполняющая обработку изображения, выделяя контуры объектов.

## Приложение



Приложение 8.1. Документация, составленная с помощью Doxygen

# Список источников

1. About Python™ | Python.org // Python.org URL: https://www.python.org/about/ (дата обращения: 14.12.2023).
2. Matplotlib — Visualization with Python // matplotlib.org URL: https://matplotlib.org/ (дата обращения: 24.12.2023).
3. Введение в TensorFlow // www.tensorflow.org URL: https://www.tensorflow.org/learn?hl=ru (дата обращения: 14.12.2023).
4. Keras: Deep Learning for humans // keras.io URL: https://keras.io/ (дата обращения: 14.12.2023).
5. NumPy // numpy.org URL: https://numpy.org/ (дата обращения: 14.12.2023).
6. GrimDarkTech/AI.Practics: A repository for storing works on the most important subject // github.com URL: https://github.com/GrimDarkTech/AI.Practics (дата обращения: 24.12.2023).
7. OpenCV - Open Computer Vision Library // opencv.org URL: https://opencv.org/ (дата обращения: 24.12.2023).