

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Факультет информатики
Кафедра технической кибернетики

Отчет по лабораторной работе №2

Дисциплина: «Технологии Искусственного Интеллекта»

Тема: «**Docker**»

Выполнил: Мелешенко И.С.

Группа: 6233-010402D

Самара 2023

Содержание

Глава 1. Установка и настройка Docker Desktop	3
Глава 2. Выполнение задания 1 лабораторной работы 2.....	5
Задание 1 лабораторной работы 2.	5
Подготовка Dockerfile.....	6
Подготовка решения задачи Salt and Paper.....	6
Исходный файл для работы.....	7
Выполнение 1 задания лабораторной работы 2.	8
1. Сборка образа контейнера.	8
2. Запуск контейнера.	8
3. Проверим контейнер.....	9
4. Перенос исходников.	9
5. Запуски скрипта.	10
6. Результаты.	12
Часть 3. Выполнение задания 2 лабораторной работы 2.	15
Задание 2 лабораторной работы 2.	15
Подготовка Dockerfile.....	16
Подготовка docker-compose.....	16
Описание кода, для работы с нейросетью.	16
Выполнение задания 2 лабораторной работы 2	18
Заключение.....	21

Глава 1. Установка и настройка Docker Desktop

1 Клиент программы Docker Desktop доступен на сайте по ссылке <https://www.docker.com/products/docker-desktop>

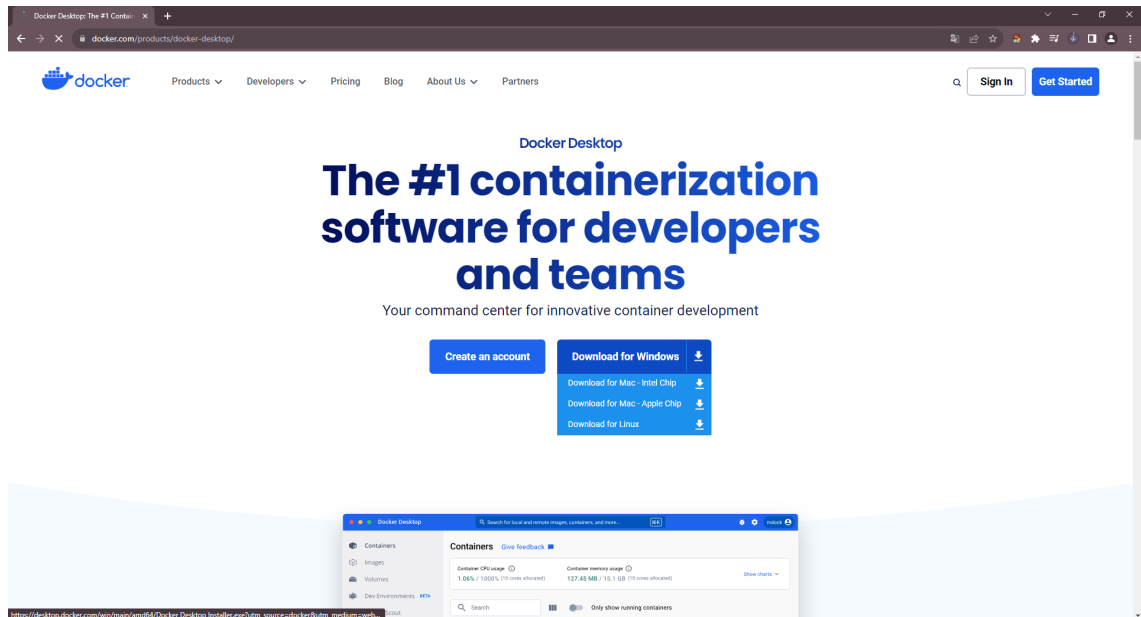


Рисунок 1 – Получение клиента программы Docker Desktop

2 На данном ПК потребуется произвести настройку для запуска программы Docker Desktop, более подробно об этом можно почитать на сайте по ссылке <https://docs.microsoft.com/en-us/windows/wsl/install-manual>.

```
$ dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

```
$ dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

3 После чего потребуется перезапуск машины. Теперь Docker Desktop установлен и запущен.

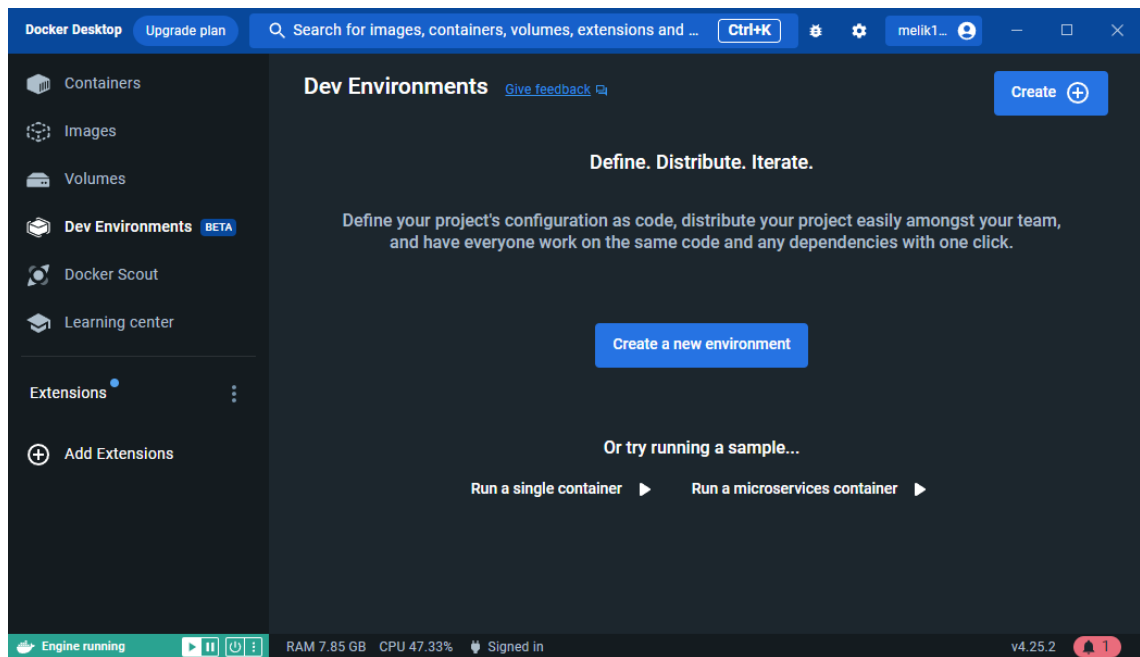


Рисунок 2 – Настроенный Docker Desktop

Глава 2. Выполнение задания 1 лабораторной работы 2.

Задание 1 лабораторной работы 2.

На базе образа Ubuntu 22.04 создать Docker контейнер со сборкой OpenCV с non free contrib модулями (пример 3 из лекции).

- Версия Убунты: 22.04
- OpenCV: 4.8.0
- CUDA: 12.2

1. Отредактировать скрипт сборки build.sh, заменить значения:

- image_tag - название тега;
- build_thread_count - количество потоков для сборки библиотеки,

лучше указать n-1, где n - количество физических ядер CPU.

2. С CUDA возможны различные приколы при установке, особенно на старые релизы типа 18.04. Если в системе нет GPU Nvidia, то установку CUDA можно вырезать из скрипта сборки и докер файла.

3. Изменить права доступа, выдать разрешение для запуска скриптов build.sh, build_env.sh:

```
chmod +x build.sh
```

```
chmod +x build_env.sh
```

4. Дописать в конец докер файла (перед CMD) команды для установки необходимых либ Python 3.

5. Запустить build.sh для сборки контейнера.

6. Реализовать алгоритм обработки изображений, скрипт на питоне положить в папку на хосте.

7. Запустить контейнер командой:

```
docker run -v <путь на хосте>:<путь внутри контейнера> -it <имя тега>
```

8. Запустить скрипт с реализованным алгоритмом в контейнере в примонитрованной внутри контейнера папке. Результат обработки сохранить в локальной директории контейнера.

9. Убедиться в появлении результата в директории хоста.

Подготовка Dockerfile.

Для того чтобы была возможность собрать образ контейнера, необходимо подготовить Dockerfile с инструкциями по сборке образа.

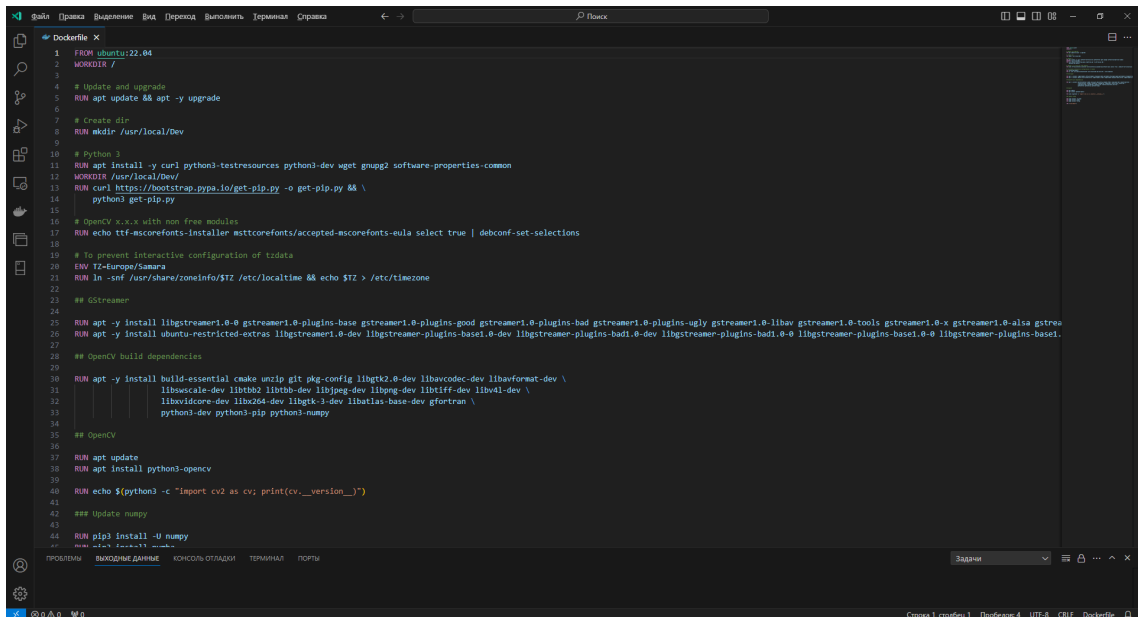


Рисунок 3 – Dockerfile для сборки образа контейнера.

Подготовка решения задачи Salt and Paper.

За основу было взято решение задачи Salt and Paper, суть которой заключена в следующем:

Дано изображение размера $M \times N$ с шумом «Salt and Paper». Необходимо реализовать и применить версию 9-точечного медианного фильтра и сохранить результат в выходное изображение. Недостающие значения для краевых строк и столбцов берутся из ближайших пикселей. Данная задача уже рассматривалась мною в курсе «Высокопроизводительные вычисления» и решение расположено в репозитории по ссылке: https://github.com/Black-Viking-63/HPC_Salt_And_Paper.

Воспользуемся им, однако внесем не большие корректировки, из-за возможности работы на GPU. Новый вариант решения данной задачи с использованием CPU, представлено ниже.

```

1  import numpy as np
2  from numba import cuda
3  import math
4  from time import time
5  from PIL import Image
6
7  def salt_and_pepper_add(image, prob):
8      rnd = np.random.rand(image.shape[0], image.shape[1])
9      noisy = image.copy()
10     noisy[rnd < prob] = 0
11     noisy[rnd > 1 - prob] = 255
12     return noisy
13
14 def median_filter(a):
15     b = a.copy()
16     start = time()
17     for i in range(2, len(a)-1):
18         for j in range(2, len(a[1])-1):
19             t[0], t[1], t[2], t[3], t[4], t[5], t[6], t[7], t[8] = a[i-1][j-1], a[i-1][j], a[i-1][j+1], a[i][j-1], a[i][j], a[i][j+1], a[i+1][j-1], a[i+1][j], a[i+1][j+1]
20             for k in range(8-k):
21                 if t[k] > t[k+1]:
22                     t[k], t[k+1] = t[k+1], t[k]
23             b[i][j] = t[8]
24     return b, time()-start
25
26 def experiment(img_name, need_draw):
27     img = image.open(img_name).convert('L')
28     img = np.array(img)
29
30     img = salt_and_pepper_add(img, 0.05)
31     img1 = image.fromarray(np.uint8(img))
32     img1.save('user/app/src/SAP.jpg')
33
34     img1, ctime = median_filter(img)
35     img1 = image.fromarray(np.uint8(img1))
36     img1.save('user/app/src/CPU.jpg')
37
38     n=len(img)*len(img[0])
39     print('Количество элементов -', n)
40     print('Время работы алгоритма на CPU -', ctime)
41
42     print('Начали обработку!')
43     experiment('user/app/src/price.jpg', True)
44     print('Закончили обработку!')
45
46
47
48
49
50

```

Рисунок 4 – Решение задачи Salt and Paper

Исходный файл для работы.

В качестве исходного изображения для работы алгоритма, я использовал скриншот из компьютерной игры, который представлен ниже.

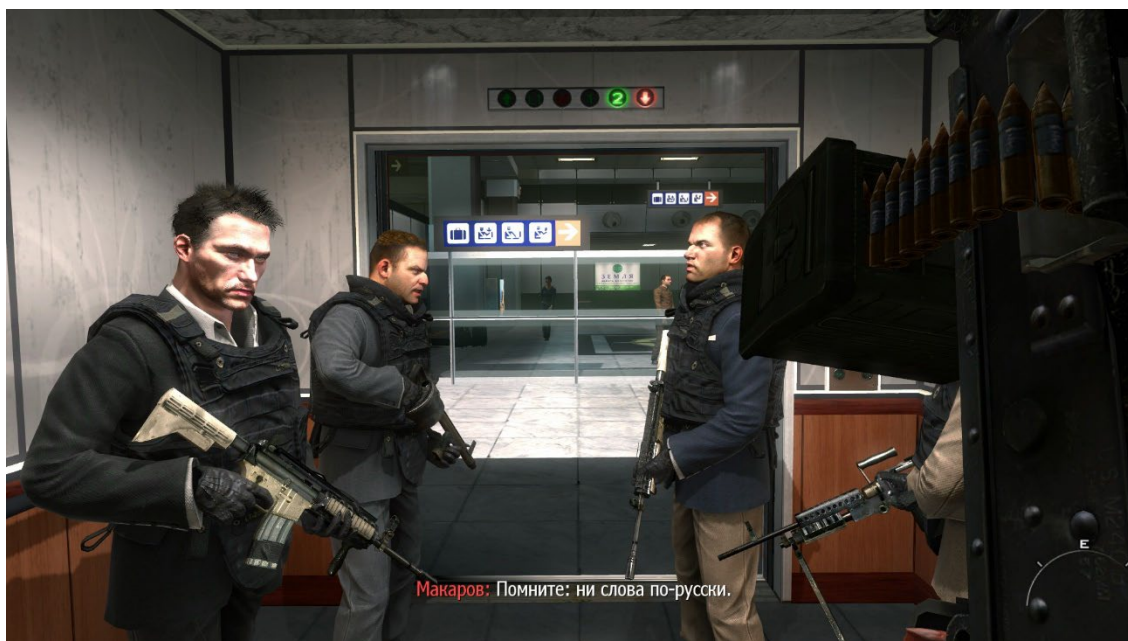


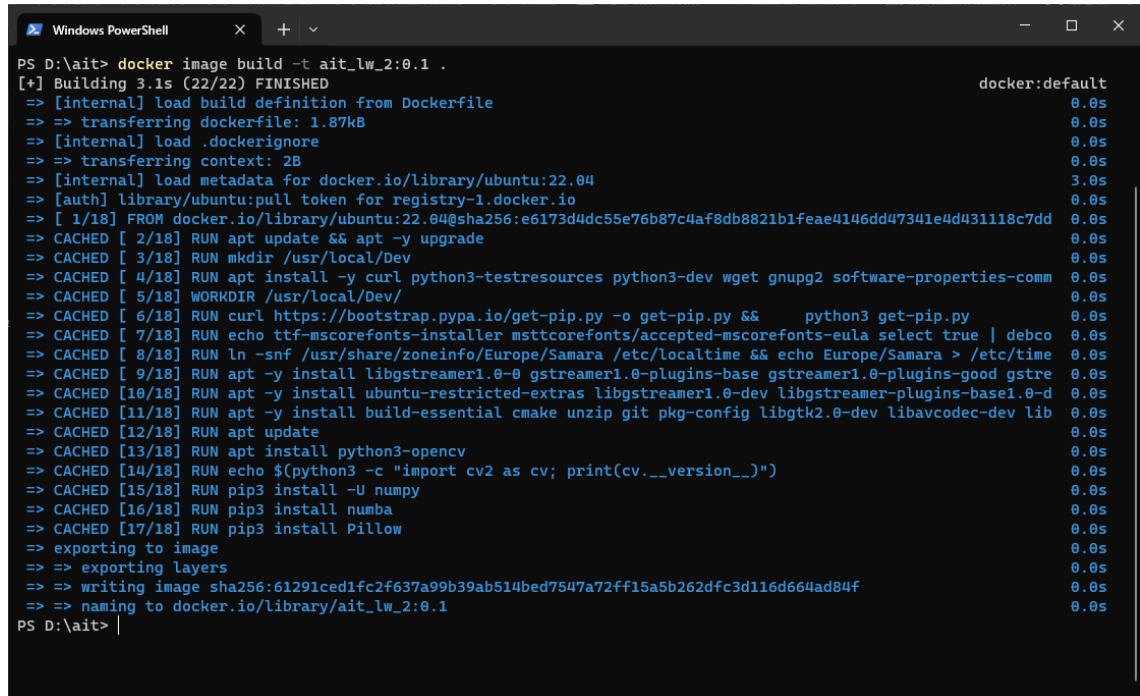
Рисунок 5 – Исходный файл для работы алгоритма

Выполнение 1 задания лабораторной работы 2.

1. Сборка образа контейнера.

Для сборки образа, будем использовать команду:

```
$ docker image build -t ait_lw_2:0.1 .
```



```
PS D:\ait> docker image build -t ait_lw_2:0.1 .
[+] Building 3.1s (22/22) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.87kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:22.04
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [ 1/18] FROM docker.io/library/ubuntu:22.04@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd
=> CACHED [ 2/18] RUN apt update && apt -y upgrade
=> CACHED [ 3/18] RUN mkdir /usr/local/Dev
=> CACHED [ 4/18] RUN apt install -y curl python3-testresources python3-dev wget gnupg2 software-properties-comm
=> CACHED [ 5/18] WORKDIR /usr/local/Dev/
=> CACHED [ 6/18] RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py
=> CACHED [ 7/18] RUN echo ttf-mscorefonts-installer msttcorefonts/accepted-mscorefonts-eula select true | debco
=> CACHED [ 8/18] RUN ln -snf /usr/share/zoneinfo/Europe/Samara /etc/localtime && echo Europe/Samara > /etc/time
=> CACHED [ 9/18] RUN apt -y install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstre
=> CACHED [10/18] RUN apt -y install ubuntu-restricted-extras libgstreamer1.0-dev libgstreamer-plugins-basel.0-d
=> CACHED [11/18] RUN apt -y install build-essential cmake unzip git pkg-config libgtk2.0-dev libavcodec-dev lib
=> CACHED [12/18] RUN apt update
=> CACHED [13/18] RUN apt install python3-opencv
=> CACHED [14/18] RUN echo $(python3 -c "import cv2 as cv; print(cv.__version__)")
=> CACHED [15/18] RUN pip3 install -U numpy
=> CACHED [16/18] RUN pip3 install numba
=> CACHED [17/18] RUN pip3 install Pillow
=> exporting to image
=> => writing image sha256:61291ced1fc2f637a99b39ab514bed7547a72ff15a5b262dfc3d116d664ad84f
=> => naming to docker.io/library/ait_lw_2:0.1
PS D:\ait>
```

Рисунок 6 – Сборка образа контейнера

2. Запуск контейнера.

После успешной сборки, можем приступить к запуску контейнера, используя следующую команду:

```
$ docker run -dit -v .\data:/usr/app/src --name my_container_ait_lw ait_lw_2:0.1
```



```
Windows PowerShell
PS D:\ait> docker image build -t ait_lw_2:0.1 .
[+] Building 3.1s (22/22) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 1.87kB
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:22.04
=> [auth] library/ubuntu:pull token for registry-1.docker.io
=> [ 1/18] FROM docker.io/library/ubuntu:22.04@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd
=> CACHED [ 2/18] RUN apt update && apt -y upgrade
=> CACHED [ 3/18] RUN mkdir /usr/local/Dev
=> CACHED [ 4/18] RUN apt install -y curl python3-testresources python3-dev wget gnupg2 software-properties-comm
=> CACHED [ 5/18] WORKDIR /usr/local/Dev/
=> CACHED [ 6/18] RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py
=> CACHED [ 7/18] RUN echo ttf-mscorefonts-installer msttcorefonts/accepted-mscorefonts-eula select true | debco
=> CACHED [ 8/18] RUN ln -snf /usr/share/zoneinfo/Europe/Samara /etc/localtime && echo Europe/Samara > /etc/time
=> CACHED [ 9/18] RUN apt -y install libgstreamer1.0-gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstre
=> CACHED [10/18] RUN apt -y install ubuntu-restricted-extras libgstreamer1.0-dev libgstreamer-plugins-base1.0-d
=> CACHED [11/18] RUN apt -y install build-essential cmake unzip git pkg-config libgtk2.0-dev libavcodec-dev lib
=> CACHED [12/18] RUN apt update
=> CACHED [13/18] RUN apt install python3-opencv
=> CACHED [14/18] RUN echo $(python3 -c "import cv2 as cv; print(cv.__version__)")
=> CACHED [15/18] RUN pip3 install -U numpy
=> CACHED [16/18] RUN pip3 install numba
=> CACHED [17/18] RUN pip3 install Pillow
=> => exporting to image
=> => exporting layers
=> => writing image sha256:61291ced1fc2f637a99b39ab514bed7547a72ff15a5b262dfc3d116d664ad84f
=> => naming to docker.io/library/ait_lw_2:0.1
PS D:\ait> docker run -dit -v .\data:/usr/app/src --name my_container_ait_lw ait_lw_2:0.1
feb7701a0c4667742ce5e845be3c48c31723b7f4abef7839a79e1eb3fcdcb2cc3
PS D:\ait>
```

Рисунок 7 – Запуск контейнера

3. Проверим контейнер

После запуска перейдем в Docker Desktop, и проверим что все получилось, наш контейнер создан, запущен и работает.

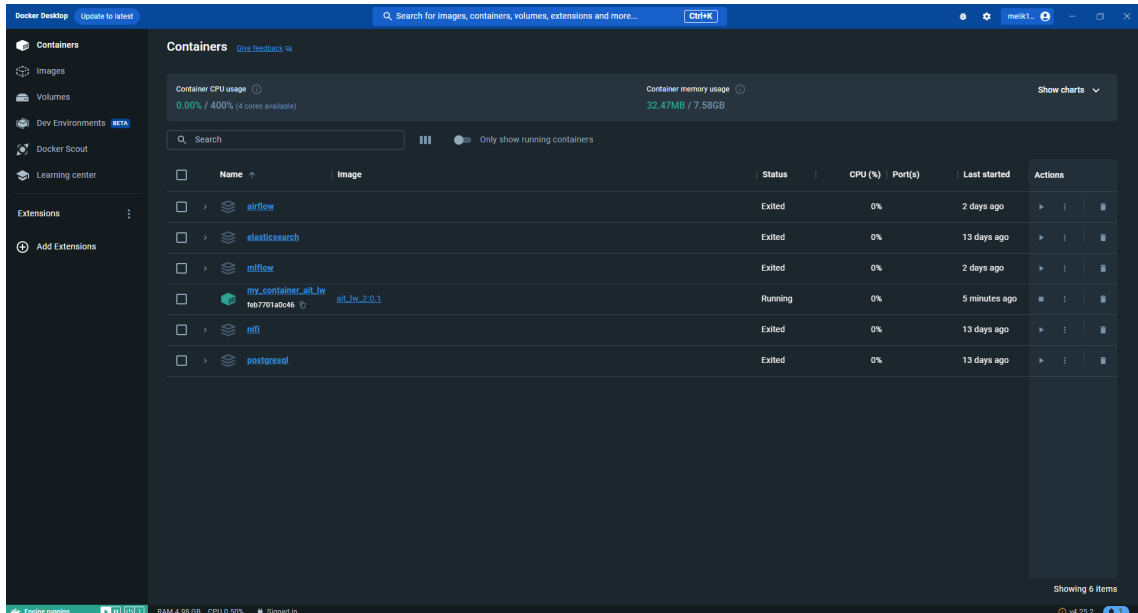


Рисунок 8 – Проверка контейнера

4. Перенос исходников.

Для продолжения работы нам необходимо код и исходный файл перенести в контейнер, в котором и будет исполняться наш код. Файл с исходным кодом, переносим при помощи команды:

\$ docker cp D:\ait\sap.py my_container_ait_lw:/usr/app/src/sap.py

```
Windows PowerShell
[+] Building 3.1s (22/22) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 1.87kB                             0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04    3.0s
=> [auth] library/ubuntu:pull token for registry-1.docker.io      0.0s
=> [ 1/18] FROM docker.io/library/ubuntu:22.04@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd 0.0s
=> CACHED [ 2/18] RUN apt update && apt -y upgrade                0.0s
=> CACHED [ 3/18] RUN mkdir /usr/local/Dev                        0.0s
=> CACHED [ 4/18] RUN apt install -y curl python3-testresources python3-dev wget gnupg2 software-properties-comm 0.0s
=> CACHED [ 5/18] WORKDIR /usr/local/Dev/                          0.0s
=> CACHED [ 6/18] RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py          0.0s
=> CACHED [ 7/18] RUN echo ttf-mscorefonts-installer msttcorefonts/accepted-mscorefonts-eula select true | debco 0.0s
=> CACHED [ 8/18] RUN ln -snf /usr/share/zoneinfo/Europe/Samara /etc/localtime && echo Europe/Samara > /etc/time 0.0s
=> CACHED [ 9/18] RUN apt -y install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstre 0.0s
=> CACHED [10/18] RUN apt -y install ubuntu-restricted-extras libgstreamer1.0-dev libgstreamer-plugins-base1.0-d 0.0s
=> CACHED [11/18] RUN apt -y install build-essential cmake unzip git pkg-config libgtk2.0-dev libavcodec-dev lib 0.0s
=> CACHED [12/18] RUN apt update                                  0.0s
=> CACHED [13/18] RUN apt install python3-opencv                  0.0s
=> CACHED [14/18] RUN echo $(python3 -c "import cv2 as cv; print(cv.__version__)") 0.0s
=> CACHED [15/18] RUN pip3 install -U numpy                       0.0s
=> CACHED [16/18] RUN pip3 install numba                          0.0s
=> CACHED [17/18] RUN pip3 install Pillow                         0.0s
=> exporting to image                                             0.0s
=> => exporting layers                                             0.0s
=> => writing image sha256:61291ced1fc2f637a99b39ab514bed7547a72ff15a5b262dfc3d116d664ad84f 0.0s
=> => naming to docker.io/library/ait_lw_2:0.1                    0.0s
PS D:\ait> docker run -dit -v .\data:/usr/app/src --name my_container_ait_lw ait_lw_2:0.1
feb7701a0c4667742ce5e845be3c48c31723b7f4abef7839a79e1eb3fcd8b2cc3
PS D:\ait> docker cp D:\ait\sap.py my_container_ait_lw:/usr/app/src/sap.py
Successfully copied 3.07kB to my_container_ait_lw:/usr/app/src/sap.py
PS D:\ait>
```

Рисунок 9 – Перенос исходного кода в контейнер

Файл изображения переносим аналогичной командой:

\$ docker cp D:\ait\price.jpg my_container_ait_lw:/usr/app/src/price.jpg

```
Windows PowerShell
=> => transferring dockerfile: 1.87kB                             0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                       0.0s
=> [internal] load metadata for docker.io/library/ubuntu:22.04    3.0s
=> [auth] library/ubuntu:pull token for registry-1.docker.io      0.0s
=> [ 1/18] FROM docker.io/library/ubuntu:22.04@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd 0.0s
=> CACHED [ 2/18] RUN apt update && apt -y upgrade                0.0s
=> CACHED [ 3/18] RUN mkdir /usr/local/Dev                        0.0s
=> CACHED [ 4/18] RUN apt install -y curl python3-testresources python3-dev wget gnupg2 software-properties-comm 0.0s
=> CACHED [ 5/18] WORKDIR /usr/local/Dev/                          0.0s
=> CACHED [ 6/18] RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py          0.0s
=> CACHED [ 7/18] RUN echo ttf-mscorefonts-installer msttcorefonts/accepted-mscorefonts-eula select true | debco 0.0s
=> CACHED [ 8/18] RUN ln -snf /usr/share/zoneinfo/Europe/Samara /etc/localtime && echo Europe/Samara > /etc/time 0.0s
=> CACHED [ 9/18] RUN apt -y install libgstreamer1.0-0 gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstre 0.0s
=> CACHED [10/18] RUN apt -y install ubuntu-restricted-extras libgstreamer1.0-dev libgstreamer-plugins-base1.0-d 0.0s
=> CACHED [11/18] RUN apt -y install build-essential cmake unzip git pkg-config libgtk2.0-dev libavcodec-dev lib 0.0s
=> CACHED [12/18] RUN apt update                                  0.0s
=> CACHED [13/18] RUN apt install python3-opencv                  0.0s
=> CACHED [14/18] RUN echo $(python3 -c "import cv2 as cv; print(cv.__version__)") 0.0s
=> CACHED [15/18] RUN pip3 install -U numpy                       0.0s
=> CACHED [16/18] RUN pip3 install numba                          0.0s
=> CACHED [17/18] RUN pip3 install Pillow                         0.0s
=> exporting to image                                             0.0s
=> => exporting layers                                             0.0s
=> => writing image sha256:61291ced1fc2f637a99b39ab514bed7547a72ff15a5b262dfc3d116d664ad84f 0.0s
=> => naming to docker.io/library/ait_lw_2:0.1                    0.0s
PS D:\ait> docker run -dit -v .\data:/usr/app/src --name my_container_ait_lw ait_lw_2:0.1
feb7701a0c4667742ce5e845be3c48c31723b7f4abef7839a79e1eb3fcd8b2cc3
PS D:\ait> docker cp D:\ait\sap.py my_container_ait_lw:/usr/app/src/sap.py
Successfully copied 3.07kB to my_container_ait_lw:/usr/app/src/sap.py
PS D:\ait> docker cp D:\ait\price.jpg my_container_ait_lw:/usr/app/src/price.jpg
Successfully copied 373kB to my_container_ait_lw:/usr/app/src/price.jpg
PS D:\ait>
```

Рисунок 10 – Перенос изображения в контейнер

5. Запуска скрипта.

Перед запуском скрипта перейдем в папку и проверим нахождение файлов в нужных директориях.

Исходная папка с файлами:

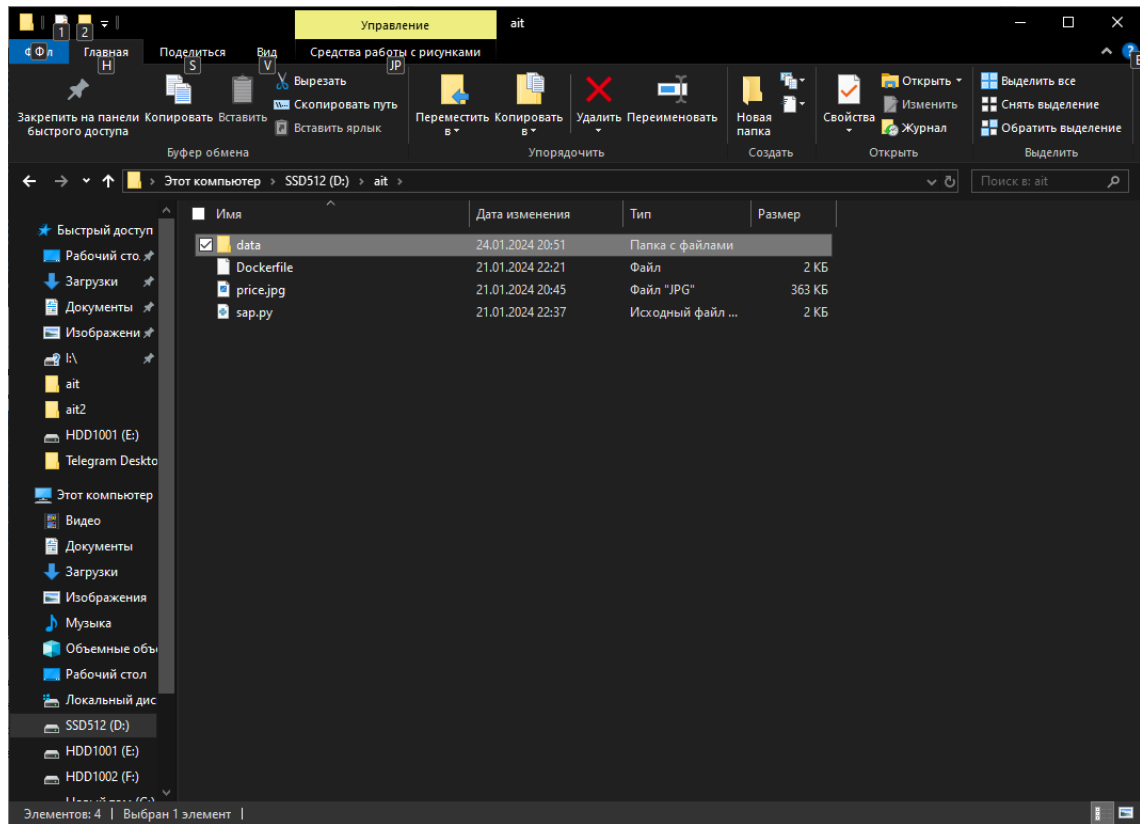


Рисунок 11 – Исходная папка с файлами

Папка с файлами для контейнера.

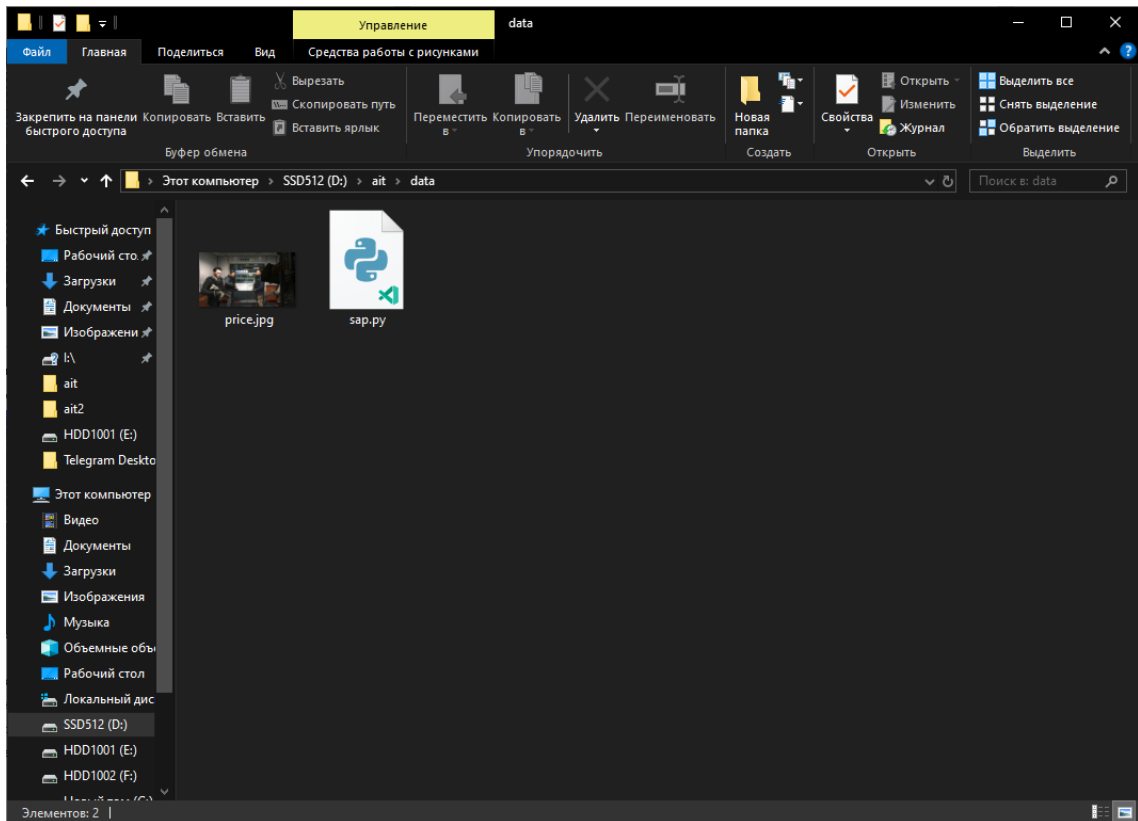


Рисунок 12 – Исходная папка с файлами для контейнера

Для того, чтобы запустить наш скрипт из контейнера, воспользуемся командой:

\$ docker exec -it my_container_ait_lw python3 /usr/app/src/sap.py

```
Windows PowerShell
> [ 1/18] FROM docker.io/library/ubuntu:22.04@sha256:e6173d4dc55e76b87c4af8db8821b1feae4146dd47341e4d431118c7dd 0.0s
> CACHED [ 2/18] RUN apt update && apt -y upgrade 0.0s
> CACHED [ 3/18] RUN mkdir /usr/local/Dev 0.0s
> CACHED [ 4/18] RUN apt install -y curl python3-testresources python3-dev wget gnupg2 software-properties-comm 0.0s
> CACHED [ 5/18] WORKDIR /usr/local/Dev/ 0.0s
> CACHED [ 6/18] RUN curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py && python3 get-pip.py 0.0s
> CACHED [ 7/18] RUN echo ttf-mscorefonts-installer msttcorefonts/accepted-mscorefonts-eula select true | debco 0.0s
> CACHED [ 8/18] RUN ln -snf /usr/share/zoneinfo/Europe/Samara /etc/localtime && echo Europe/Samara > /etc/time 0.0s
> CACHED [ 9/18] RUN apt -y install libgstreamer1.0-gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstre 0.0s
> CACHED [10/18] RUN apt -y install ubuntu-restricted-extras libgstreamer1.0-dev libgstreamer-plugins-base1.0-d 0.0s
> CACHED [11/18] RUN apt -y install build-essential cmake unzip git pkg-config libgtk2.0-dev libavcodec-dev lib 0.0s
> CACHED [12/18] RUN apt update 0.0s
> CACHED [13/18] RUN apt install python3-opencv 0.0s
> CACHED [14/18] RUN echo $(python3 -c "import cv2 as cv; print(cv.__version__)") 0.0s
> CACHED [15/18] RUN pip3 install -U numpy 0.0s
> CACHED [16/18] RUN pip3 install numba 0.0s
> CACHED [17/18] RUN pip3 install Pillow 0.0s
> exporting to image 0.0s
> exporting layers 0.0s
> writing image sha256:61291ced1fc2f637a99b39ab514bed7547a72ff15a5b262dfc3d116d664ad84f 0.0s
> naming to docker.io/library/ait_lw_2:0.1 0.0s
PS D:\ait> docker run -dit -v .\data:/usr/app/src --name my_container_ait_lw ait_lw_2:0.1
feb7701a0c4667742ce5e845be3c48c31723b7f4abef7839a79e1eb3fcd2cc3
PS D:\ait> docker cp D:\ait\sap.py my_container_ait_lw:/usr/app/src/sap.py
Successfully copied 3.07kB to my_container_ait_lw:/usr/app/src/sap.py
PS D:\ait> docker cp D:\ait\price.jpg my_container_ait_lw:/usr/app/src/price.jpg
Successfully copied 373kB to my_container_ait_lw:/usr/app/src/price.jpg
PS D:\ait> docker exec -it my_container_ait_lw python3 /usr/app/src/sap.py
Начали обработку!
Количество элементов = 2073600
Время работы алгоритма на CPU = 13.952820777893066
Закончили обработку!
PS D:\ait>
```

Рисунок 13 – Запуск скрипта

6. Результаты.

После отработки скрипта перейдем в папку с файлами контейнера и посмотрим на результаты.

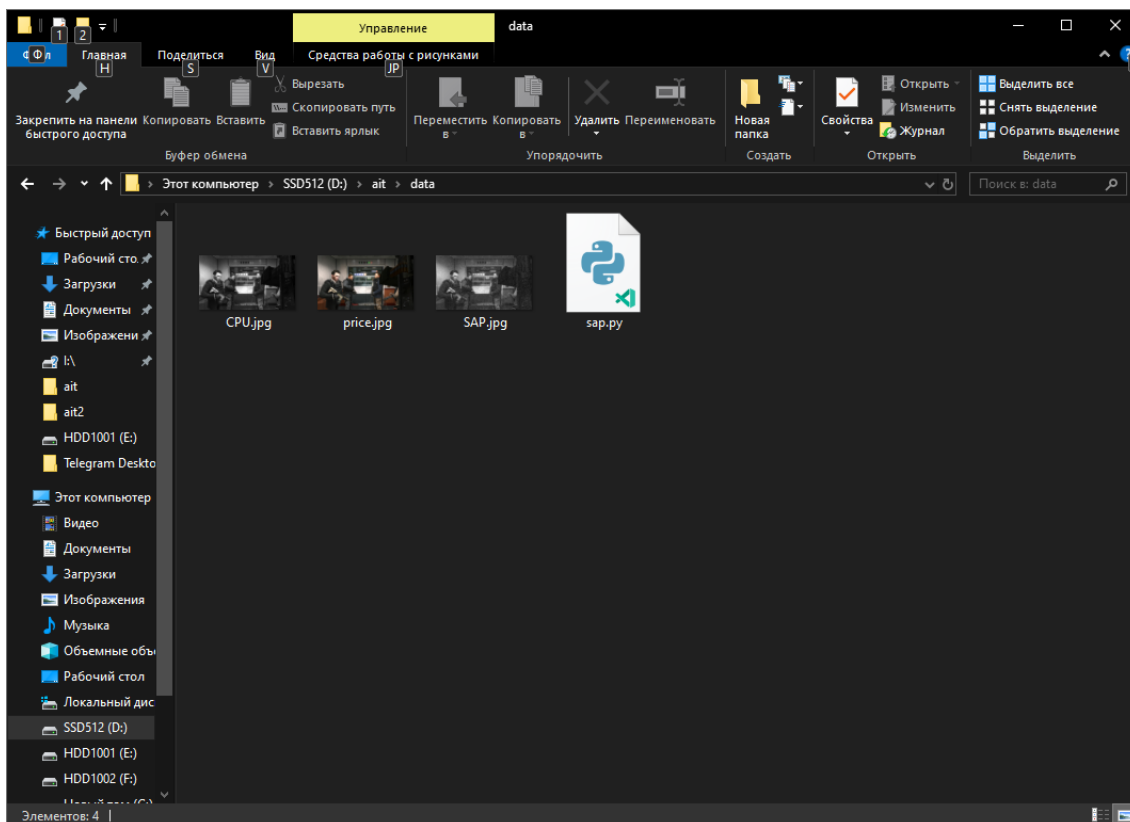


Рисунок 14 - Результаты

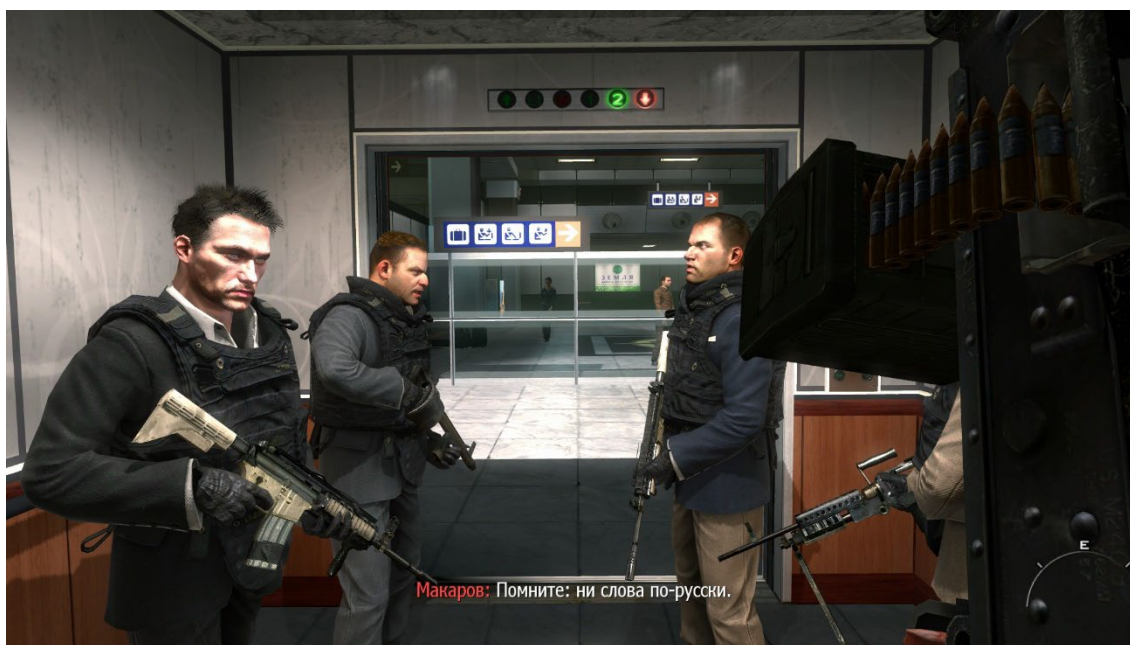


Рисунок 15 – Исходное изображение.



Рисунок 16 – Изображение с шумом «Salt and Pepper»



Рисунок 17 – Изображение восстановленное.

После окончания выполнения задания перейде ко второй части.

Часть 3. Выполнение задания 2 лабораторной работы 2.

Задание 2 лабораторной работы 2.

Запустить предобученную нейронку с использованием pytorch внутри контейнера. Для создания контейнера использовать Docker Compose.

1. Собрать контейнер с установленным PyTorch (CPU или GPU версия).

Проще всего собрать контейнер на базе подготовленного образа с Docker Hub, например, pytorch:2.1.0-cuda11.8-cudnn8-devel. Можно и из обычного образа, например, Убунты. Тогда нужно будет установить CUDA определенной версии при сборке контейнера, а на самом хосте поставить NVIDIA Container Toolkit.

2. Написать скрипт обработки изображений с использованием нейросети.

Можно выбрать любую понравившуюся модель/задачу обработки изображения нейросетью.

У niconielsen32 в репах есть целая подборка различных заготовок по Computer Vision, например, вычисление карты глубин по изображению.

3. Запустить контейнер командой:

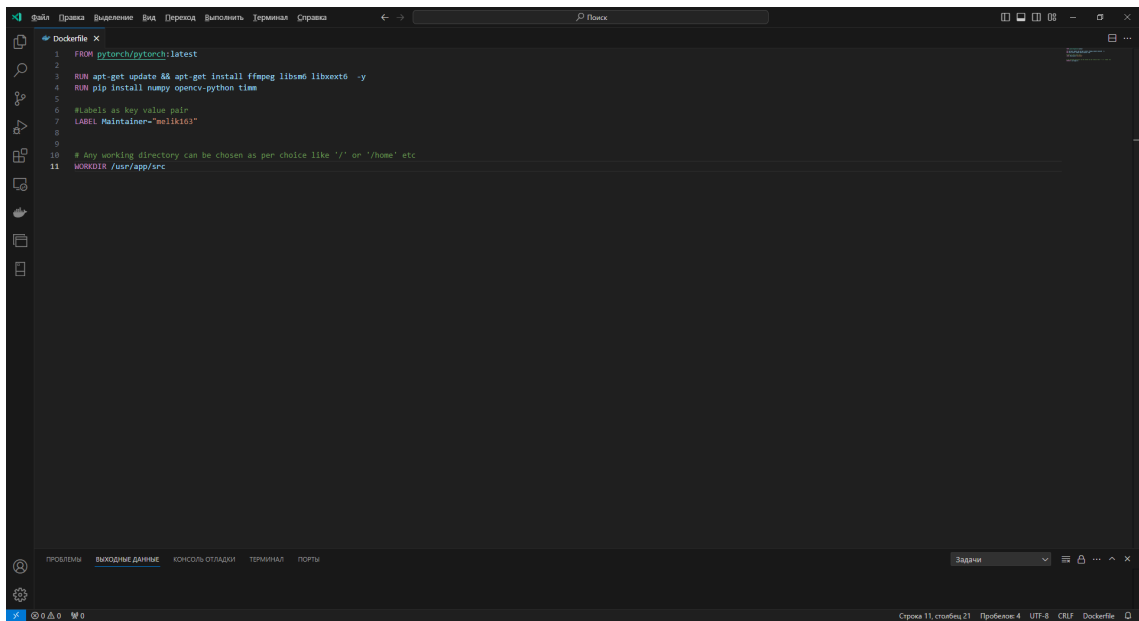
docker compose -f <имя_конфига.yaml> up

4. Запустить скрипт с реализованным алгоритмом в контейнере в примонитрованной внутри контейнера папке. Результат обработки сохранить в локальной директории контейнера.

5. Убедиться в появлении результата в директории хоста.

Подготовка Dockerfile

Для создания образа контейнера воспользуемся следующим кодом:

A screenshot of a code editor showing a Dockerfile. The file is named 'Dockerfile X' and contains the following lines:

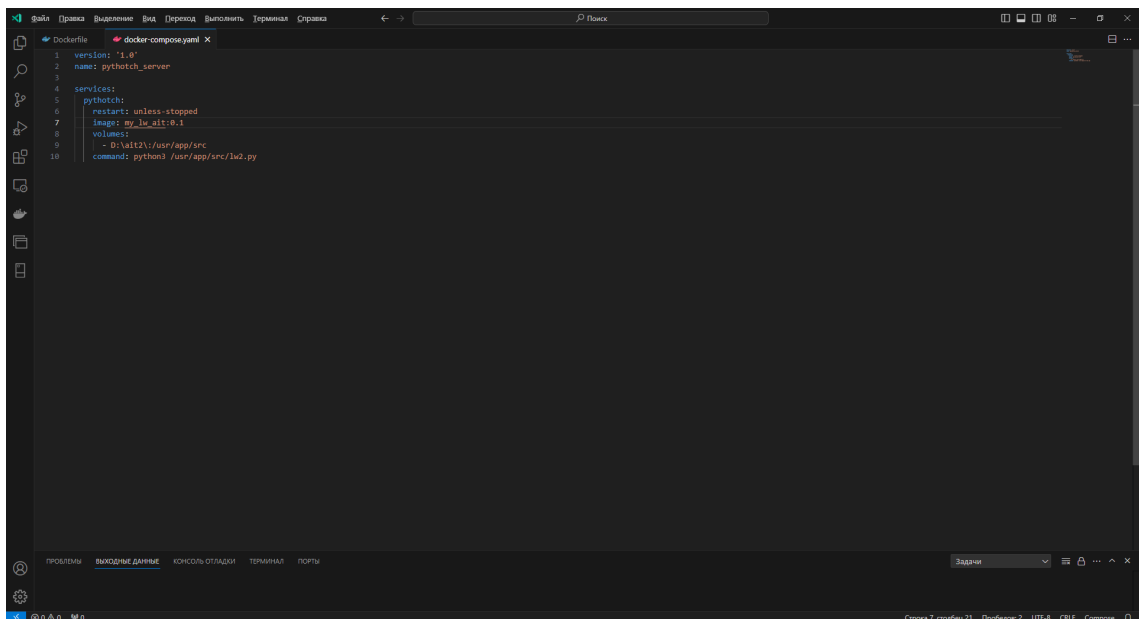
```
1 FROM pytorch/pytorch:latest
2
3 RUN apt-get update && apt-get install ffmpeg libsm6 libxext6 -y
4 RUN pip install numpy opencv-python timm
5
6 #Labels as key value pair
7 LABEL Maintainer="meliki63"
8
9
10 # Any working directory can be chosen as per choice like "/" or "/home" etc.
11 WORKDIR /usr/app/src
```

The editor has a sidebar on the left with icons for Explorer, Search, Source Control, and Docker. The bottom status bar shows 'Строка 11, столбец 21', 'Пробелов: 4', 'UTF-8', 'CRLF', and 'Dockerfile'.

Рисунок 18 - Dockerfile

Подготовка docker-compose

Для запуска контейнера опишем docker-compose.yaml файл.

A screenshot of a code editor showing a docker-compose.yaml file. The file is named 'docker-compose.yaml X' and contains the following lines:

```
1 version: '3.9'
2 name: pytorch_server
3
4 services:
5   pytorch:
6     restart: unless-stopped
7     image: my_lw_dlt:0.1
8     volumes:
9       - D:\ait2\usr\app\src
10    command: python3 /usr/app/src/lw2.py
```

The editor has a sidebar on the left with icons for Explorer, Search, Source Control, and Docker. The bottom status bar shows 'Строка 7, столбец 21', 'Пробелов: 7', 'UTF-8', 'CRLF', and 'Compose'.

Рисунок 19 – Docker-Compose

Описание кода, для работы с нейросетью.


```
1 import cv2
2 import torch
3 import time
4 import numpy as np
5
6 model_type = "DPT_Hybrid" # MiDaS v3 - Hybrid (medium accuracy, medium inference speed)
7
8 midas = torch.hub.load("intel-isl/MiDaS", model_type)
9
10 # Move model to GPU if available
11 device = torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
12 midas.to(device)
13 midas.eval()
14
15 # Load transforms to resize and normalize the image
16 midas_transforms = torch.hub.load("intel-isl/MiDaS", "transforms")
17
18 if model_type == "DPT_Large" or model_type == "DPT_Hybrid":
19     transform = midas_transforms.dpt_transform
20 else:
21     transform = midas_transforms.small_transform
22
23 img = cv2.imread("input.png")
24 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
25
26 # Apply input transforms
27 input_batch = transform(img).to(device)
28
29 # Prediction and resize to original resolution
30 with torch.no_grad():
31     prediction = midas(input_batch)
32     prediction = torch.nn.functional.interpolate(prediction.unsqueeze(1), size=img.shape[:2], mode="bicubic", align_corners=False).squeeze()
33
34     depth_map = prediction.cpu().numpy()
35     depth_map = cv2.normalize(depth_map, None, 0, 1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_64F)
36
37 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
38
39 depth_map = (depth_map*255).astype(np.uint8)
40 depth_map = cv2.cvtColor(depth_map, cv2.COLORMAP_RAINBOW)
41
42 cv2.imwrite("image.png", img)
43 cv2.imwrite("output_depth_map.png", depth_map)
```

Рисунок 20 – Исходный код

Для работы нейросети необходимо изображение, которое будет использовано в качестве входного параметра. Будем использовать стоп кадр из кинофильма.

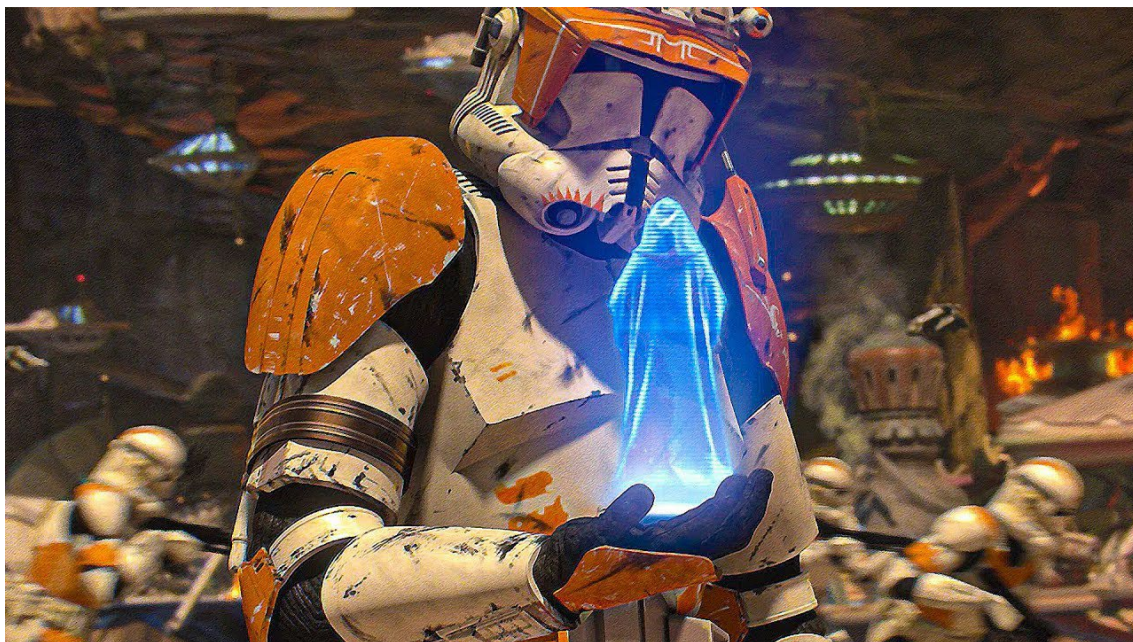


Рисунок 21 – Изображение для работы нейросети

Перед началом работы переходим в исходную директорию:

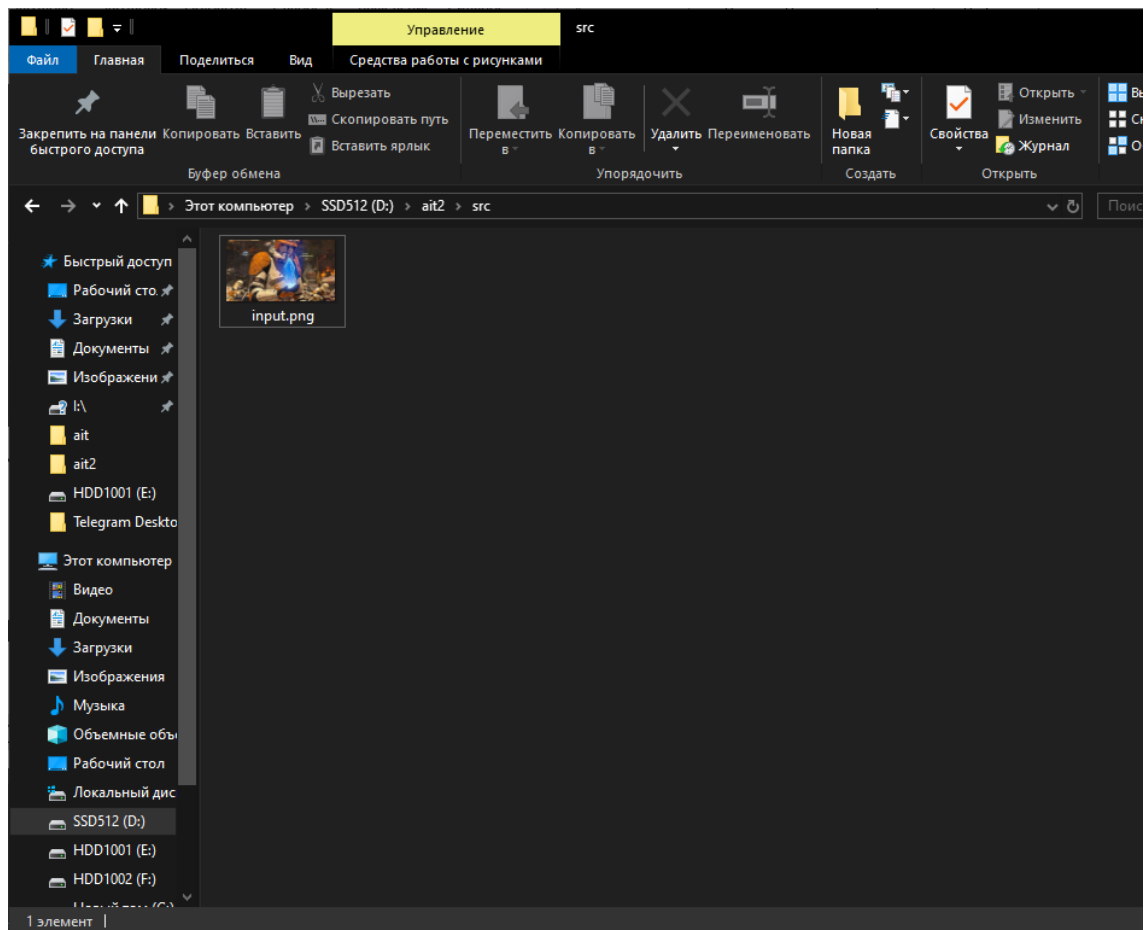


Рисунок 22 – Исходная директория с файлами

Выполнение задания 2 лабораторной работы 2

После всех приготовлений переходим в терминал, и выполняем сборку контейнера при помощи команды:

```
$ docker image build -t my_lw_ait:0.1 D:\ait2
```

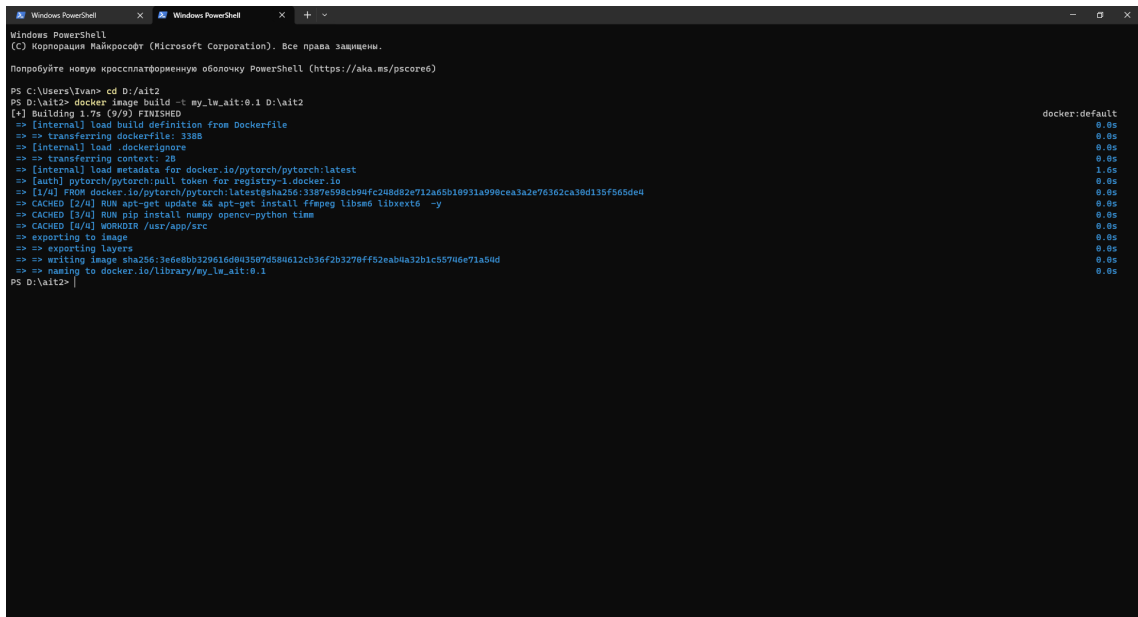


Рисунок 23 – Сборка образа контейнера

После окончания сборки образа запустим контейнер при помощи команды:

\$ docker compose -f docker-compose.yaml up

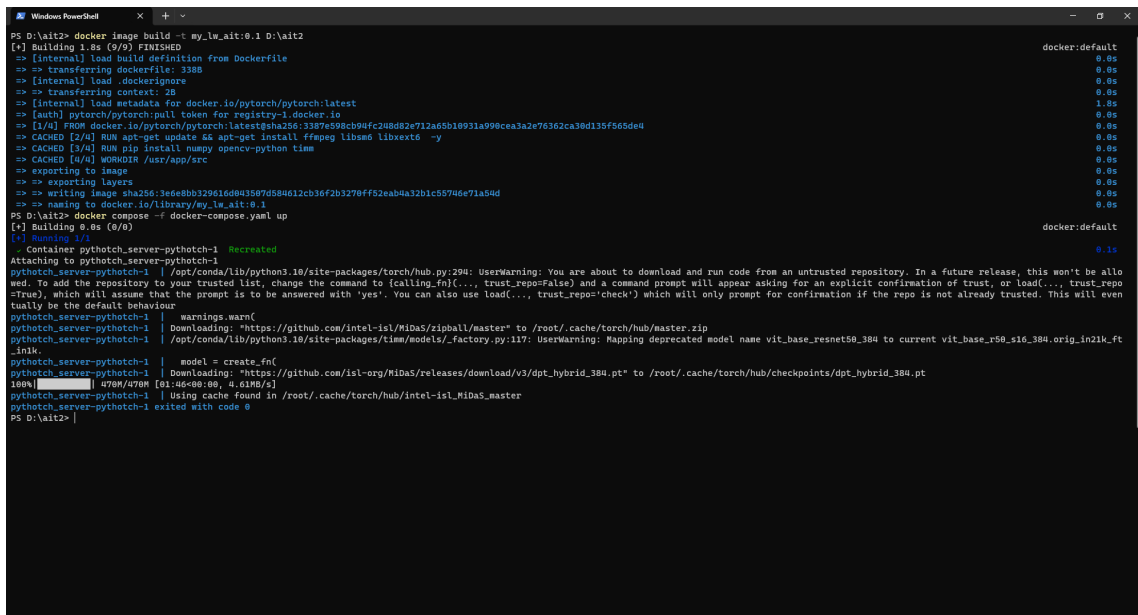


Рисунок 24 – Запуск контейнера

По окончании работы алгоритма, перейдем в директорию и посмотрим на результаты:

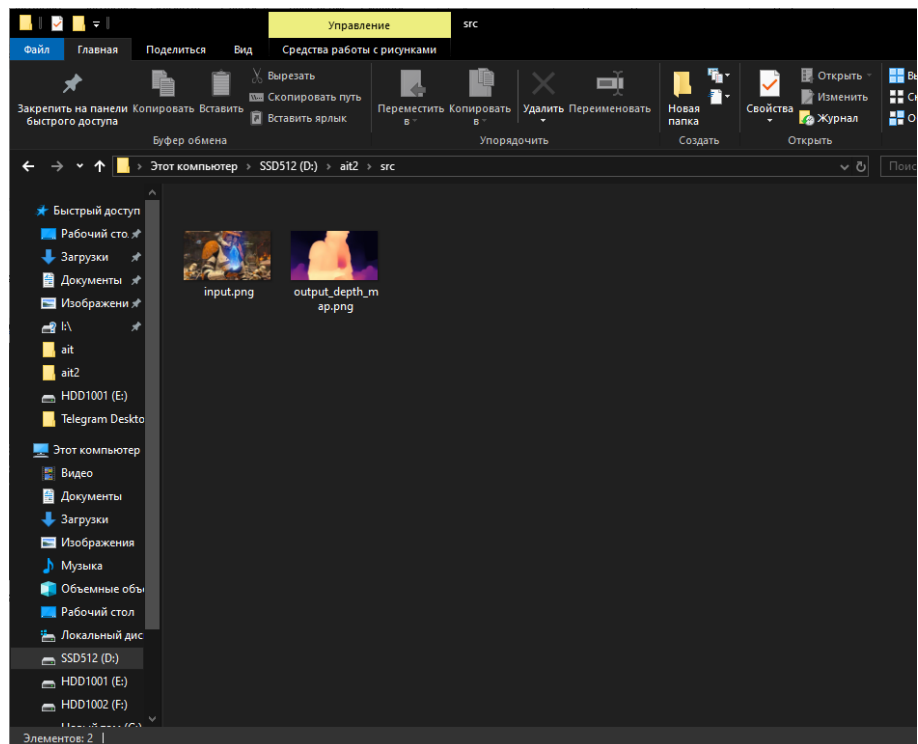


Рисунок 25 - Результаты

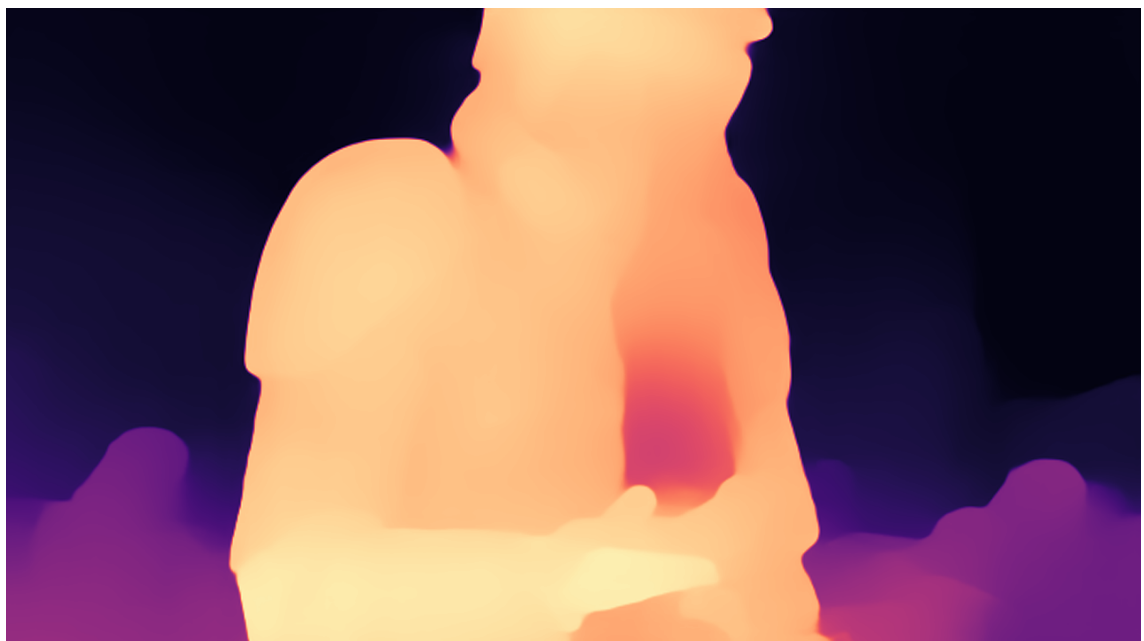


Рисунок 26 – Результирующее изображение.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы получены навыки работы с Docker.