Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
**«Пермский национальный исследовательский
политехнический университет»
(ПНИПУ)**

Электротехнический факультет
кафедра «Информационные технологии и автоматизированные системы»
направление подготовки: 09.03.04 Разработка программно-информационных систем

О Т Ч Е Т
по учебной практике

Выполнили студенты гр. РИС-22-1б

Верхоланцева Е.С. и Ишемцева М.А.

_____
(подпись)

Проверили:

___доцент кафедры ИТАС Курушин Даниил Сергеевич _____
(должность, Ф.И.О. ответственного за практическую подготовку от профильной организации)

_____ _____
(оценка) (подпись)

_____
(дата)

Пермь 2023

Введение

В настоящее время 3D-печать становится все более популярным и широко используемым методом производства различных деталей и изделий. Однако, как и в любом процессе производства, в 3D-печати могут возникать дефекты, которые могут привести к снижению качества и прочности конечной продукции.

Постановка задачи

Разработать систему дефектоскопии в процессе печати на 3D-принтере с использованием нейросети.

Задачу можно разбить на 2 этапа:

1. Работа с 3D-принтером, установка камеры, которая будет фотографировать процесс после каждого слоя и проверять на наличие дефектов.
2. Разработка нейросети для выявления дефектов.

Анализ задачи

Нейросеть — математическая модель, работающая по принципам нервной системы живых организмов. Ее основное назначение — решать интеллектуальные задачи. То есть те, в которых нет изначально заданного алгоритма действий и спрогнозированного результата.

Главной особенностью нейросетей является способность к обучению. Они могут обучаться как под управлением человека, так и самостоятельно, применяя полученный ранее опыт.

В рамках практики будет использоваться однослойная нейронная сеть, умеющая распознавать входные данные и позволяющая себя обучать. Входные данные представляют собой списки из “1” и “0” и играют роль изображения детали. Нейронная сеть должна уметь определять наличие дефекта (отличие от бездефектного варианта) или его отсутствие (соответствие бездефектному варианту)

Для управления 3D-принтером Anet A8 использовались файлы g-code.
G-code является стандартным языком программирования, используемым в процессе управления 3D-принтерами и другими машинами с числовым программным управлением (ЧПУ). G-code представляет собой набор инструкций, которые определяют перемещение и операции, выполняемые 3D-принтером во время печати. Команды G-code могут указывать такие параметры, как координаты перемещения (например, X, Y, Z), скорость перемещения, температуру экструдера и платформы печати, время ожидания, команды остановки и многое другое.

Пример кода на языке g-code:

```
G1 F300 Z0.5 ; установка скорости 300 мм/мин , координата Z = 0.5 мм
G1 F1500 E0  ; скорость 1500 мм/мин, всасывание пластика в экструдер
; печать по заданным координатам
G1 X105.216 Y104.041 E0.07341
G1 X105.802 Y103.645 E0.14692
G1 X106.428 Y103.317 E0.22038
G1 X107.087 Y103.062 E0.29382
G1 X107.771 Y102.883 E0.36731
```

Для конвертации 3D-модели в g-code файл использовалось программное обеспечение для 3D-печати Ultimaker Cura. При открытии файла модели и выборе принтера также показывается время печати, вес и количество слоев.

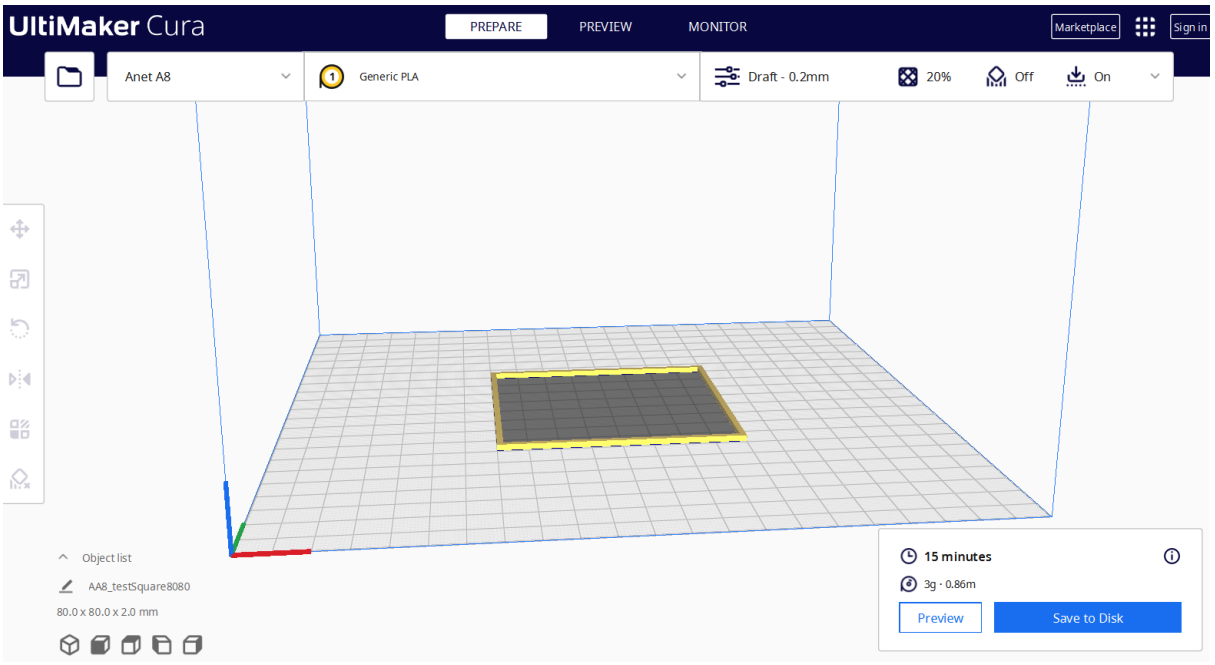
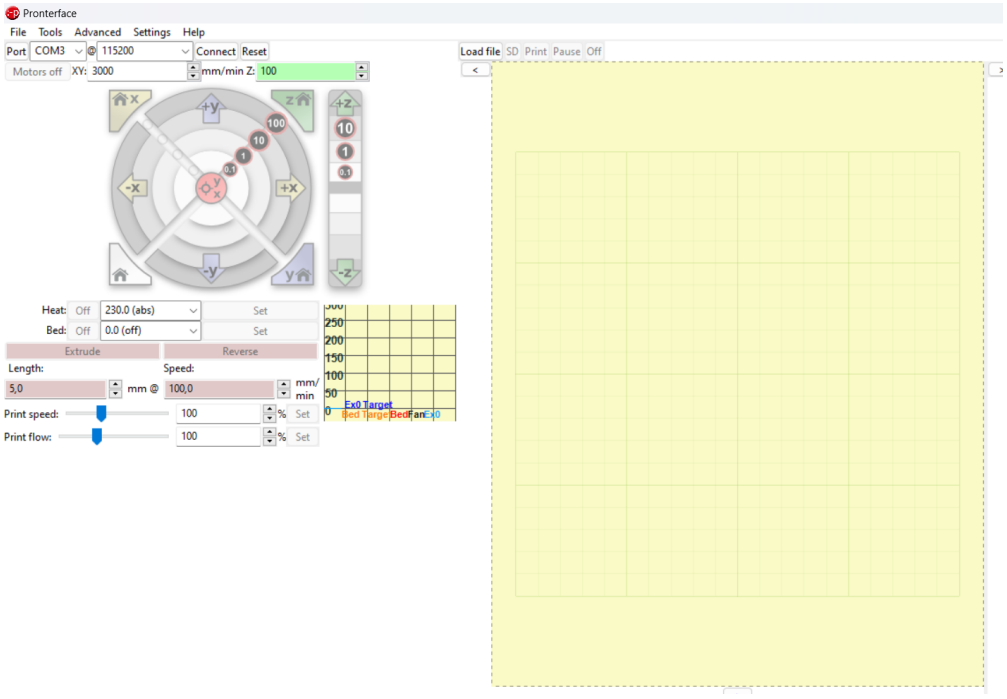


рис. 1 Пример изображения 3D-модели в Ultimaker Cura

Для управления и контроля принтера через USB использовалось программное обеспечение Pronterface.



Для работы в pronterface нужно подключить принтер через USB, выбрать порт (в нашем случае COM3), нажать кнопку connect. После этого в консоль программы можно писать команды в формате g-кода или управлять стрелками в интерфейсе программы. Принтер будет выполнять данные ему команды.

Чтобы отслеживать дефекты на детали, было решено выдвигать стол принтера после печати каждого слоя и фотографировать деталь. При выявлении дефекта с помощью нейросети печать должна была останавливаться, а иначе продолжаться, но в рамках практики эту задумку пришлось заменить на более простой вариант, а именно - однослойная нейросеть, работающая на простых моделях из «0» и «1».

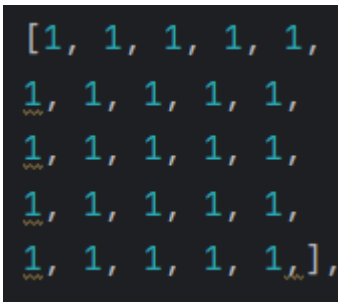


рис.2 “Деталь без дефекта”

В нашем случае “идеальной” моделью (без дефекта) будет последовательность из 25 единиц. Любая другая последовательность будет считаться дефектной.

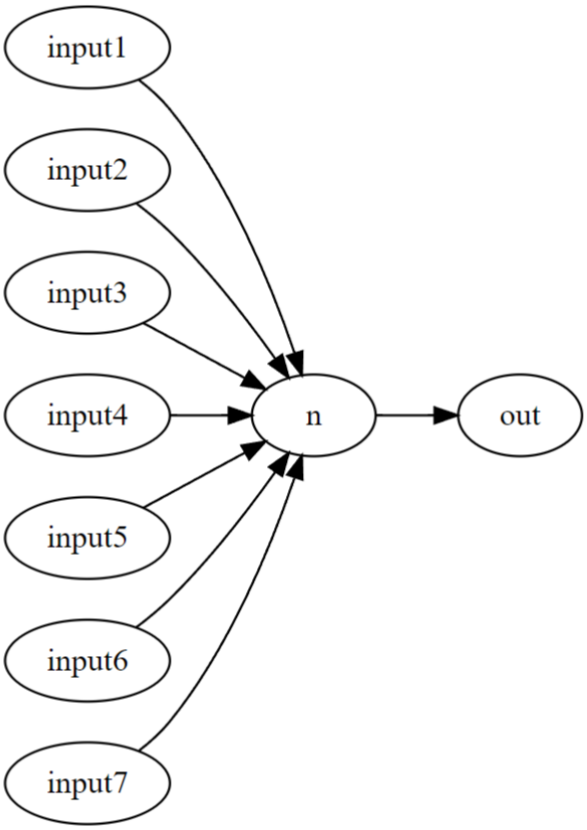


рис.3 схема работы однослойной нейросети

Все файлы для удобства были загружены на Github. Была изучена система контроля версий git.

```
987@Ekaterina MINGW64 ~/neural_practice (main)
$ git add perceptron.py main.py

987@Ekaterina MINGW64 ~/neural_practice (main)
$ git commit -m "добавили файл с однослойной нейросетью"
[main 35884d5] добавили файл с однослойной нейросетью
 1 file changed, 1 deletion(-)
```

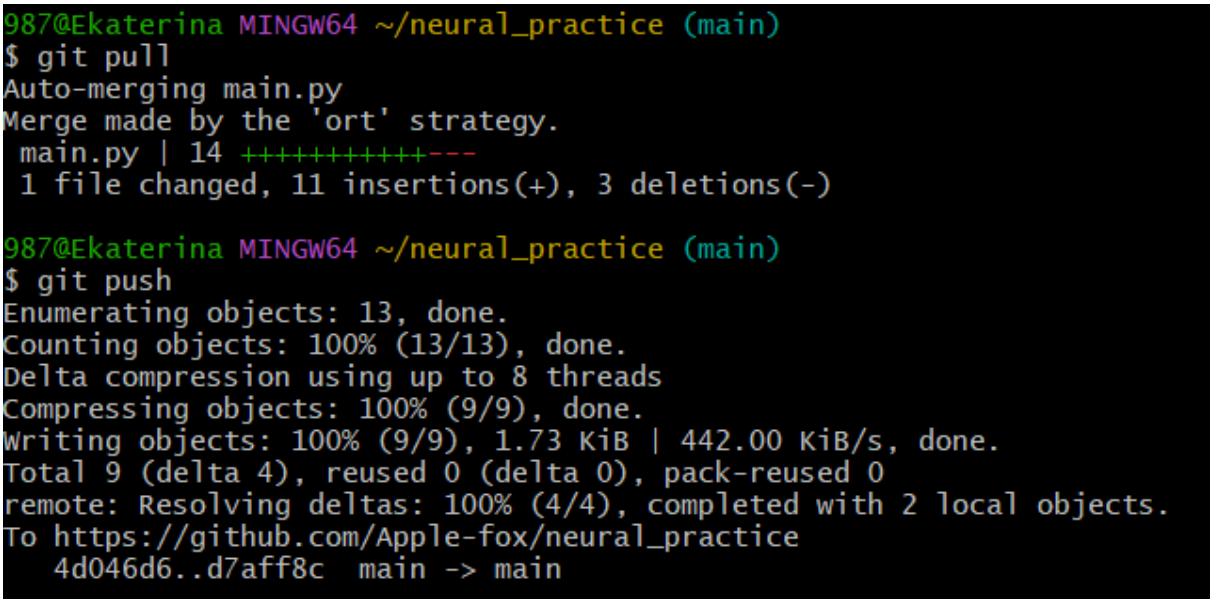


рис. 3,4 - процесс сохранения файла на GitHub с помощью Git Bash

Для отслеживания процесса печати использовалась web-камера Razer Kiyo Pro разрешением 1920x1080.

Технология реализации

Алгоритм печати был реализован на языке программирования Python в программной среде PyCharm.

OpenCV

Библиотека компьютерного зрения и машинного обучения с открытым исходным кодом. В неё входят более 2500 алгоритмов, в которых есть как классические, так и современные алгоритмы для компьютерного зрения и машинного обучения.

Основные используемые методы:

- cv2.VideoCapture(0) - захват камеры
- cv2.imwrite(<название файла>, <поток изображения>) - сохранение фотографии с камеры
- release() - закрывает поток камеры
- DestroyAllWindows() - закрывает открытые окна

```
def take_photo(n):
    cap = cv2.VideoCapture(0)  # захват камеры
    o = 0
    while True:  # цикл позволяет снимать видео

        o += 1
        ret, img = cap.read()

        if o == 50:
            cv2.imwrite("camera{}.png".format(n), img)
            break

    cap.release()
    cv2.destroyAllWindows()
    return
```

рис. 5 - функция фото

Данный код представляет функцию take_photo(n), которая использует библиотеку OpenCV для захвата изображения с веб-камеры и сохранения его в файле с именем "camera{n}.png", где n - номер изображения.

Описание кода:

- Создается объект cap, используя функцию cv2.VideoCapture(0), для захвата видео с веб-камеры. Аргумент 0 указывает на использование первой доступной камеры.
- Инициализируется переменная o для отслеживания количества кадров.
- В бесконечном цикле выполняется захват кадров с помощью функции cap.read(). Результат захвата и изображение сохраняются в переменные ret и img соответственно.
- При каждой итерации цикла переменная o увеличивается на 1.
- Когда o достигает значения 50, с помощью функции cv2.imwrite() изображение сохраняется в файле с именем "camera{n}.png".

- После сохранения изображения закрывается захват видео с помощью функции cap.release() и закрываются все окна с помощью функции cv2.destroyAllWindows().

PySerial

Этот модуль инкапсулирует доступ к последовательному порту.

Основные методы:

- ser = serial.Serial('COM3', 9600) - инициировать последовательное устройство
- ser.read() - читать 1 байт с последовательного устройства
- ser.readline() - читать строку с последовательного устройства
- ser.write() - отправить данные через последовательный порт
- ser.close() - закрыть порт

Исходный код программы на Python:

```
ser = serial.Serial('COM3', 115200)
time.sleep(2)

f = open("b.gcode")
a = f.readline()
n = 0
com = ["0", "0", "0", "0"]

while (a):

	if a[0] != ";":
		if "X" in a or "Y" in a or "Z" in a or "E" in a:
			s = a.split(" ")
			for i in range(len(s)):
				if "X" in s[i] and s[i].split("X")[1] != com[0]:
					com[0] = s[i].split("X")[1].strip("\n")
				if "Y" in s[i] and s[i].split("Y")[1] != com[1]:
					com[1] = s[i].split("Y")[1].strip("\n")
				if "Z" in s[i] and s[i].split("Z")[1] != com[2]:
					com[2] = s[i].split("Z")[1].strip("\n")
				if "E" in s[i] and s[i].split("E")[1] != com[3] and
				is_number(s[i].split("E")[1]):
				com[3] = s[i].split("E")[1].strip("\n")
		command = a.strip("\n")
		command = "{}\r\n".format(command)
		ser.write(command.encode())
```

```
            b = ser.readline()

            while b != b'ok 0\r\n' and b != b'ok\r\n' and b != b'wait\r\n':
                  print(b)
                  b = ser.readline()

            print(a)

      else:

            if ";MESH:NONMESH" in a: # если попадается строка означающая
            смену слоя
                  command = "G1 F1500 E{}\r\n".format(str(float(com[3]) -
            9.5))
                  open("filik.txt", "a").write(command)
                  ser.write(command.encode())
                  b = ser.readline()

                  while b != b'ok 0\r\n' and b != b'ok\r\n' and b !=
                  b'wait\r\n':
                  print(b)
                  b = ser.readline()

                  command = "G0 X0 Y220; injected\r\n" # отправляем команду
                  на выдвигание стола
                  ser.write(command.encode())

                  b = ser.readline()

                  while b != b'ok 0\r\n' and b != b'ok\r\n' and b !=
                  b'wait\r\n':
                  print(b)
                  b = ser.readline()

                  take_photo(n) # делаем снимок
                  n += 1

                  command = "G1 X{} Y{} Z{} E{}\r\n".format(com[0], com[1],
                  com[2], com[3])
                  ser.write(command.encode())

                  b = ser.readline()
                  while b != b'ok 0\r\n' and b != b'ok\r\n' and b !=
            b'wait\r\n':
                        print(b)
                        b = ser.readline()

      a = f.readline()

f.close()
ser.close()
```

Данный код выполняет чтение файла с именем "b.gcode" и последовательную обработку строк в файле. Каждая строка файла представляет собой команду для 3D-принтера, записанную в формате G-code. В процессе обработки команды отправляются на 3D-принтер с помощью метода ser.write().

Описание кода:

- Открывается файл "b.gcode" для чтения с помощью функции open(), и объект файла сохраняется в переменной f.
- Читается первая строка файла с помощью метода f.readline() и сохраняется в переменной a.
- Инициализируется переменная n для отслеживания номера снимка.
- Инициализируется список com, который содержит текущие координаты X, Y, Z и E.
- Вход в цикл while (a), который будет выполняться, пока строка a не будет пустой (конец файла).
- Проверяется, является ли первый символ строки a комментарием (строка начинается с символа ";"). Если нет, выполняется следующий код:
-
  - Проверяется, содержит ли строка a символы "X", "Y", "Z" или "E". Если да, строка разделяется на отдельные части, используя разделитель " ", с помощью метода split(" "), и результат сохраняется в переменной s.
  - В цикле for i in range(len(s)) проверяется каждая часть s:
  - Если часть содержит символ "X" и значение X отличается от текущего значения в com[0], текущее значение в com[0] обновляется.
  - Аналогично проверяются части, содержащие символы "Y", "Z" и "E", и соответствующие значения обновляются в com.

- Строка a очищается от символов новой строки (\n) с помощью метода strip("\n").

- Команда a отправляется на 3D-принтер с помощью метода ser.write(), предварительно преобразовав ее в байтовый формат с помощью метода encode().
- Читается ответ от 3D-принтера с помощью метода ser.readline() и результат сохраняется в переменной b.
- В цикле while b != b'ok 0\r\n' and b != b'ok\r\n' and b != b'wait\r\n' читается ответ от 3D-принтера.

- Если строка a является комментарием и содержит ";MESH:NONMESH", выполняется следующий код:

- Формируется команда для выдвижения стола и сохраняется в переменной command.
- Команда command отправляется на 3D-принтер с помощью метода ser.write().
- Читается ответ от 3D-принтера с помощью метода ser.readline() и результат сохраняется в переменной b.
- В цикле while b != b'ok 0\r\n' and b != b'ok\r\n' and b != b'wait\r\n' читается ответ от 3D-принтера и проверяется, равен ли он "ok 0\r\n" или "ok\r\n" или "wait\r\n".
- Вызывается функция take_photo(n), которая делает снимок.
- Затем формируется команда для возврата печатной головки на предыдущие координаты X, Y, Z и E и отправляется на 3D-принтер.
- Опять же, читается ответ от 3D-принтера и проверяется, равен ли он "ok 0\r\n" или "ok\r\n" или "wait\r\n".

- Читается следующая строка файла с помощью метода f.readline() и процесс повторяется.

NumPy

NumPy — это библиотека языка Python, добавляющая поддержку больших многомерных массивов и матриц, вместе с большой библиотекой

высокоуровневых (и очень быстрых) математических функций для операций с этими массивами.

Исходный код программы нейросети на Python:

```
import numpy as np

w0 = np.zeros((25))

# набор данных для обучения
D = np.array([
[1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,],

[1, 1, 1, 1, 1,
1, 0, 0, 0, 1,
1, 1, 1, 0, 1,
1, 0, 0, 0, 1,
1, 1, 1, 1, 1,],

.  .  . # здесь еще 40 обучающих последовательностей

[1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 0,
1, 1, 1, 1, 1],

[1, 0, 1, 0, 1,
0, 1, 0, 1, 0,
1, 1, 1, 1, 1,
1, 1, 1, 1, 0,
1, 1, 1, 1, 1]
])
print(len(D))
# желаемые результаты от входных данных для обучения
Y0 =
np.array([1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,0,0])
print(len(Y0))

α = 0.3 # скорость обучения
β = -0.4
σ = lambda x: 1 if x > 0 else 0 # функция активации нейрона

def f(x, _w):
      s = β + np.sum(x @ _w)
```

```
	return σ(s)

def train(w, D, Y):
	_w = w.copy()
	for x, y in zip(D, Y):
		w += α * (y - f(x, w)) * x
	return (w != _w).any()

while train(w0, D, Y0) :
	print(w0)

D = np.array([


[1, 1, 1, 1, 1,
1, 0, 0, 0, 1,
1, 1, 1, 0, 1,
1, 0, 0, 0, 1,
1, 1, 1, 1, 1, ],

[1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 0,
1, 1, 1, 1, 1, ],

[1, 1, 1, 1, 1,
1, 0, 0, 0, 1,
1, 1, 1, 1, 1,
1, 0, 0, 0, 1,
1, 1, 1, 1, 1, ],

[1, 1, 1, 1, 1,
1, 1, 1, 0, 1,
1, 1, 1, 0, 1,
1, 1, 1, 0, 1,
1, 1, 1, 1, 1],

[1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
1, 1, 1, 1, 1],

[1, 0, 1, 0, 1,
0, 1, 0, 1, 0,
1, 1, 1, 1, 1,
1, 1, 1, 1, 0,
1, 1, 1, 1, 1]

])
print("вход результат")
for x in D:
```

```
print(x, " ", f(x, w0))
```

Нейросеть обучается на наборе данных D и соответствующих желаемых результатов Y0.

Описание кода:

- Инициализация нулевого вектора весов w0 размером (25).
- Определение набора данных D, который содержит 45 последовательностей с пятью признаками каждая.
- Инициализация вектора желаемых результатов Y0 размером (45).
- Задание скорости обучения α и параметра смещения β.
- Определение функции активации σ, которая возвращает 1, если вход больше 0, и 0 в противном случае.
- Определение функции f(x, _w), которая вычисляет выход нейрона путем суммирования взвешенных признаков и применения функции активации.
- Определение функции train(w, D, Y), которая выполняет обучение нейрона на наборе данных D с использованием желаемых результатов Y. Внутри функции происходит обновление весов на основе разницы между желаемым результатом и текущим выходом нейрона.
- В цикле while train(w0, D, Y0), пока обучение продолжается (веса меняются), выводятся текущие значения весов.
- Определение нового набора данных D, которые нейросеть должна будет распознать сама.

```
вход                                                        результат
[1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1]    0
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1]    0
[1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1]    0
[1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1]    0
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]    1
[1 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1]    0
```

рис.6 - вывод программы

Деталь без дефекта определяется последовательностью из единиц и при выводе дает результат 1. Все остальные отличные от “идеальной” последовательности дают результат 0.

Нейросеть распознает простые последовательности, но требует доработки

Заключение

В ходе практики было выявлено, что текущая нейросеть способна успешно распознавать простые последовательности, но ее производительность ограничена в более сложных сценариях. Также данная нейросеть работает лишь с последовательностью цифр. В дальнейшем необходимо улучшить нейросеть для работы с изображениями, например перевод изображения в набор чисел или непосредственная работа с пикселями фотографии.

Также нужно рассмотреть следующие доработки:

- Добавление дополнительных слоев, увеличение количества нейронов.

- Сбор дополнительных примеров последовательностей с различными вариациями и дефектами.

В итоге, практика по доработке и улучшению нейросети для распознавания дефектов в процессе печати 3D-принтера с использованием камеры позволяет стремиться к повышению точности, эффективности и надежности модели. Это может привести к более автоматизированному и надежному контролю качества печати и улучшению производственных процессов.

Список источников:

https://pyserial.readthedocs.io/en/latest/pyserial.html - документация библиотеки PySerial
https://docs.opencv.org/3.4/d6/d00/tutorial_py_root.html - документация библиотеки OpenCV

Приложения:

GitHub-репозиторий - https://github.com/Apple-fox/neural_practice/tree/main