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.

Цель: Разработать программу для управления 3D-принтером и нейросеть для распознавания дефектов.

Задачу можно разбить на несколько этапов:

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

ОСНОВНАЯ ЧАСТЬ

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.[1]

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 (см. рис 1) использовались файлы g-code.

Рис. 1 - Anet A8

---

[1] Что такое нейросети и зачем они нужны // СОВКОМБЛОГ URL: https://journal.sovcombank.ru/glossarii/chego-tolko-lyudi-ne-napletut-chto-takoe-neiroseti-i-gde-ih-ispolzuyut

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.

```
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
```

Листинг 1 - Пример кода на языке g-code

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.

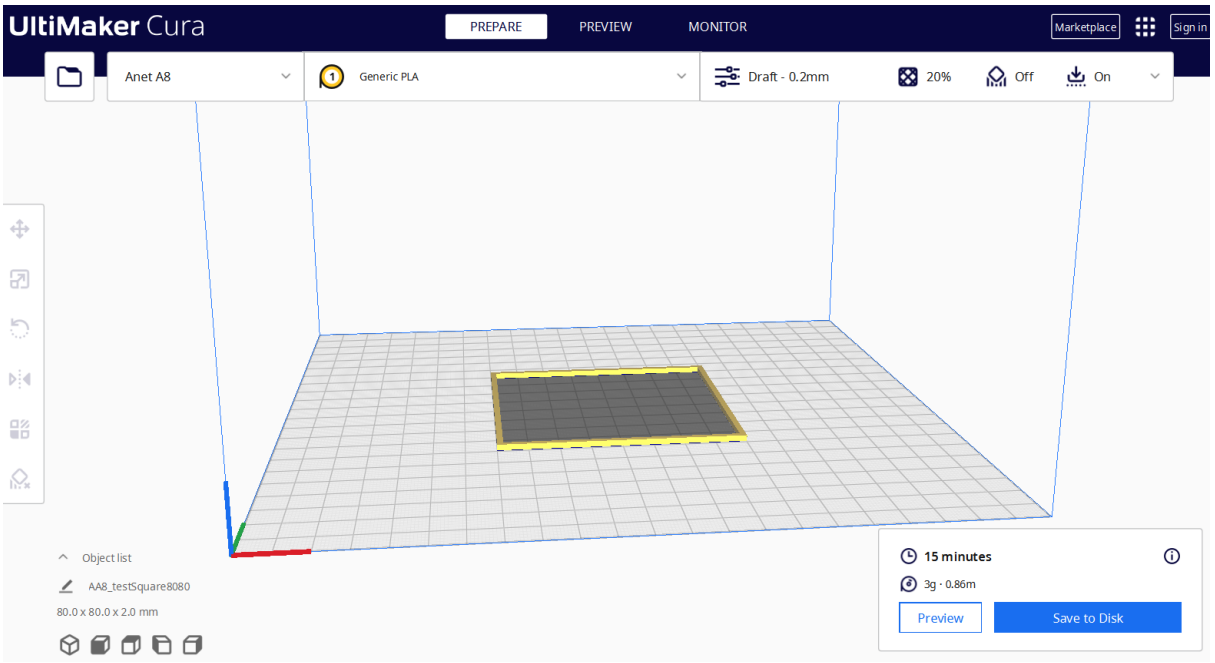

Рис. 2 - Пример изображения 3D-модели в Ultimaker Cura

Для управления и контроля принтера через USB использовалось программное обеспечение Pronterface (см. рис.3).

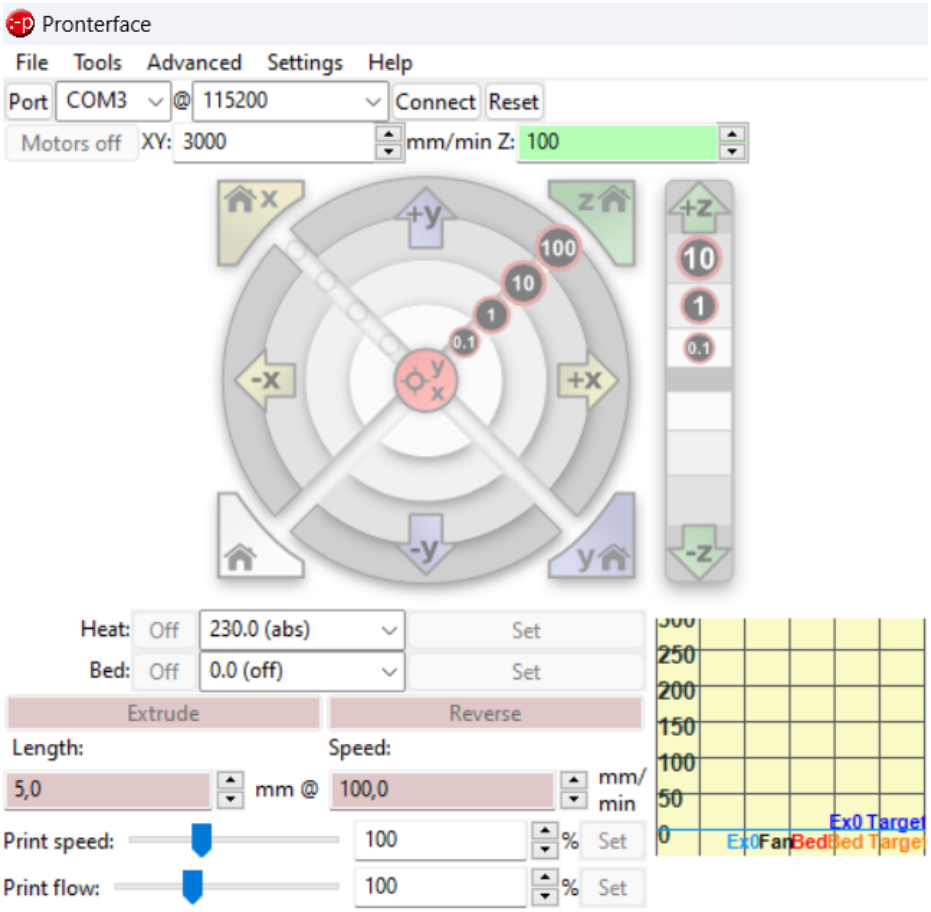

Рис. 3

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».

```
[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,],
```

Листинг 2 - “Деталь без дефекта”

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

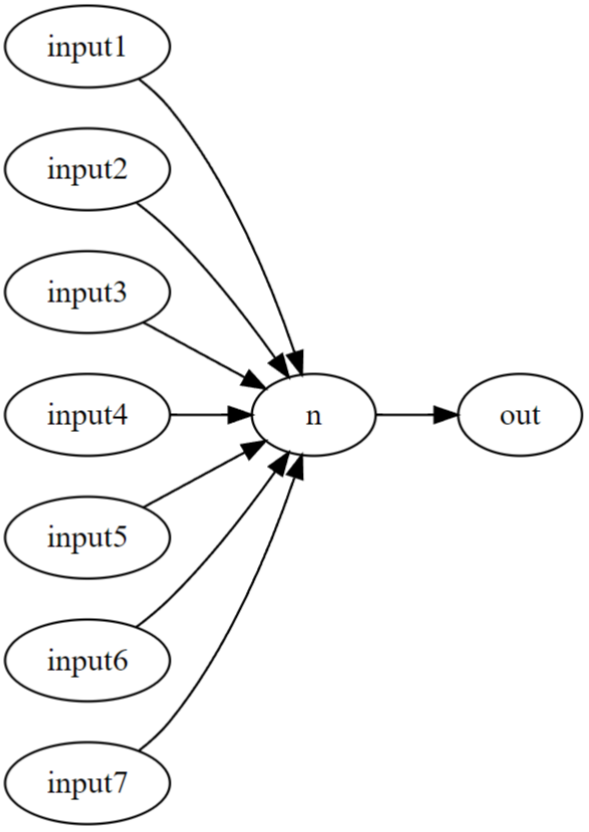

Рис.4 - Схема работы однослойной нейросети

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(-)
```

Листинг 2 - процесс сохранения файла на GitHub с помощью Git Bash

```
987@Ekaterina MINGW64 ~/neural_practice (main)
$ git pull
Auto-merging main.py
Merge made by the 'ort' strategy.
 main.py | 14 ++++++++++---
 1 file changed, 11 insertions(+), 3 deletions(-)

987@Ekaterina MINGW64 ~/neural_practice (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.73 KiB | 442.00 KiB/s, done.
Total 9 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 2 local objects.
To https://github.com/Apple-fox/neural_practice
   4d046d6..d7aff8c  main -> main
```

Листинг 3 - процесс сохранения файла на GitHub с помощью Git Bash

Для отслеживания процесса печати использовалась web-камера Razer Kiyo Pro разрешением 1920x1080 (см. рис. 5).

Рис. 5

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

Используемые библиотеки Python

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:

1. cv2.VideoCapture(0) - захват камеры
2. cv2.imwrite(<название файла>, <поток изображения>) - сохранение фотографии с камеры
3. release() - закрывает поток камеры
4. 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
```

Листинг 4 - 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:

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

PySerial - модуль, инкапсулирующий доступ к последовательному порту.

Основные методы:

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

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 на Python представлен на GitHub в файле main.py.

Исходный 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:

1. Открывается файл "b.gcode" для чтения с помощью функции open(), и объект файла сохраняется в переменной f.
2. Читается первая строка файла с помощью метода f.readline() и сохраняется в переменной a.
3. Инициализируется переменная n для отслеживания номера снимка.
4. Инициализируется список com, который содержит текущие координаты X, Y, Z и E.
5. Вход в цикл while (a), который будет выполняться, пока строка a не будет пустой (конец файла).
6. Проверяется, является ли первый символ строки a комментарием (строка начинается с символа ";"). Если нет, выполняется следующий код:
    1) 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.
    2) В цикле for i in range(len(s)) проверяется каждая часть s:
    3) Если часть содержит символ "X" и значение X отличается от текущего значения в com[0], текущее значение в com[0] обновляется.
    4) Аналогично проверяются части, содержащие символы "Y", "Z" и "E", и соответствующие значения обновляются в com.

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

8.Команда a отправляется на 3D-принтер с помощью метода ser.write(), предварительно преобразовав ее в байтовый формат с помощью метода encode().

9.Читается ответ от 3D-принтера с помощью метода ser.readline() и результат сохраняется в переменной b.

10.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.

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

1)Формируется команда для выдвижения стола и сохраняется в переменной command.

2) Команда command отправляется на 3D-принтер с помощью метода ser.write().

3) Читается ответ от 3D-принтера с помощью метода ser.readline() и результат сохраняется в переменной b.

4)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".

5) Вызывается функция take_photo(n), которая делает снимок.

6)Затем формируется команда для возврата печатной головки на предыдущие координаты X, Y, Z и E и отправляется на 3D-принтер.

7) Опять же, читается ответ от 3D-принтера и проверяется, равен ли он "ok 0\r\n" или "ok\r\n" или "wait\r\n".

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

Исходный код программы нейросети на Python взят с сайта https://github.com/daniel-kurushin/modern_data_course, и представлен на GitHub в файле preceptron.py.

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

Описание кода:

1. Инициализация нулевого вектора весов w0 размером (25).
2. Определение набора данных D, который содержит 45 последовательностей с пятью признаками каждая.
3. Инициализация вектора желаемых результатов Y0 размером (45).
4. Задание скорости обучения α и параметра смещения β.
5. Определение функции активации σ, которая возвращает 1, если вход больше 0, и 0 в противном случае.

6. Определение функции f(x, _w), которая вычисляет выход нейрона путем суммирования взвешенных признаков и применения функции активации.
7. 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.
8. В цикле while train(w0, D, Y0), пока обучение продолжается (веса меняются), выводятся текущие значения весов.
9. 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 1 0 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
```

Листинг 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:

1. Добавление дополнительных слоев, увеличение количества нейронов.
2. 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 используемых источников:

pySerial // pySerial 3.4 documentation. URL: https://pyserial.readthedocs.io/en/latest/pyserial.html(дата обращения: 06.07.2023).
OpenCV-Python Tutorials // OpenCV Open Source Computer Vision. URL: https://pyserial.readthedocs.io/en/latest/pyserial.html (дата обращения: 06.07.2023).

Приложения:

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