

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

Институт информационных технологий Кафедра общей информатики

**ОТЧЕТ О ЗАДАЧЕ: НЕЙРОСЕТЬ ТИПА МНОГОСЛОЙНЫЙ ПЕРЦЕПТРОН**

**по дисциплине**

«ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ»

Выполнил студент группы ИНБО-07-21 Михайлюк Д. С.

Приняла Евстигнеева О. А.

Москва 2021

**СОДЕРЖАНИЕ**

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc90920666)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ 4](#_Toc90920667)

[2.1 ИСПОЛЬЗОВАННЫЕ КЛАССЫ И ИХ МЕТОДЫ 4](#_Toc90920668)

[*2.1.1* КЛАСС MATRIX 4](#_Toc90920669)

[*2.1.2* КЛАСС ACTIVATOR 8](#_Toc90920670)

[*2.1.3* КЛАСС NETWORК 11](#_Toc90920671)

[*2.1.4* СВОБОДНЫЕ МЕТОДЫ И СТРУКТУРЫ 17](#_Toc90920672)

[2.2 КОД ПРОГРАММЫ В MAIN( ) 19](#_Toc90920673)

[3 ОБ ОССОБЕННОСТИ ФОРМАТА ВХОДНЫХ ДАННЫХ 23](#_Toc90920674)

[4 ВЫВОДЫ 27](#_Toc90920675)

[5 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 28](#_Toc90920676)

# ПОСТАНОВКА ЗАДАЧИ

Написать программу которая будет из себя представлять многослойный перцептрон угадывающий цифры. Уметь обучаться таким образом, что при каждой эре обучения ее эффективность увеличивалась.

# ПРИНЦИПИАЛЬНАЯ СХЕМА РАБОТЫ ПРОГРАММЫ

Для того что бы лучше понять как работает нейросеть, предлагаю рассмотреть пример простейшей сети которая определяет, изображен ли кот на картинке или нет? Ее схема изображена ниже.

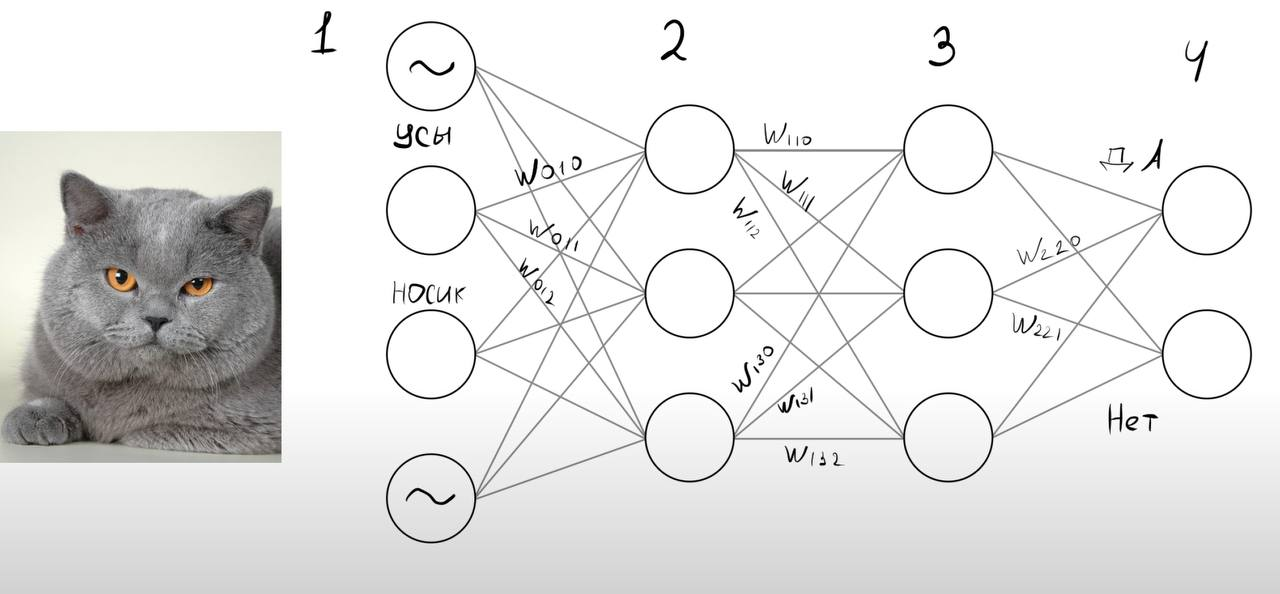


Рисунок 1 – Схема Котовой нейросети

Мы имеем сеть состоящую из 4 слоев. Слой 1 – входной, он состоит из n-ого количества нейронов в которые записываются пиксели изображения. Для лучшего я выделил из них 2 условные группы, 1 – определяет наличие кошачьих ушей, 2 – кошачьего носа.

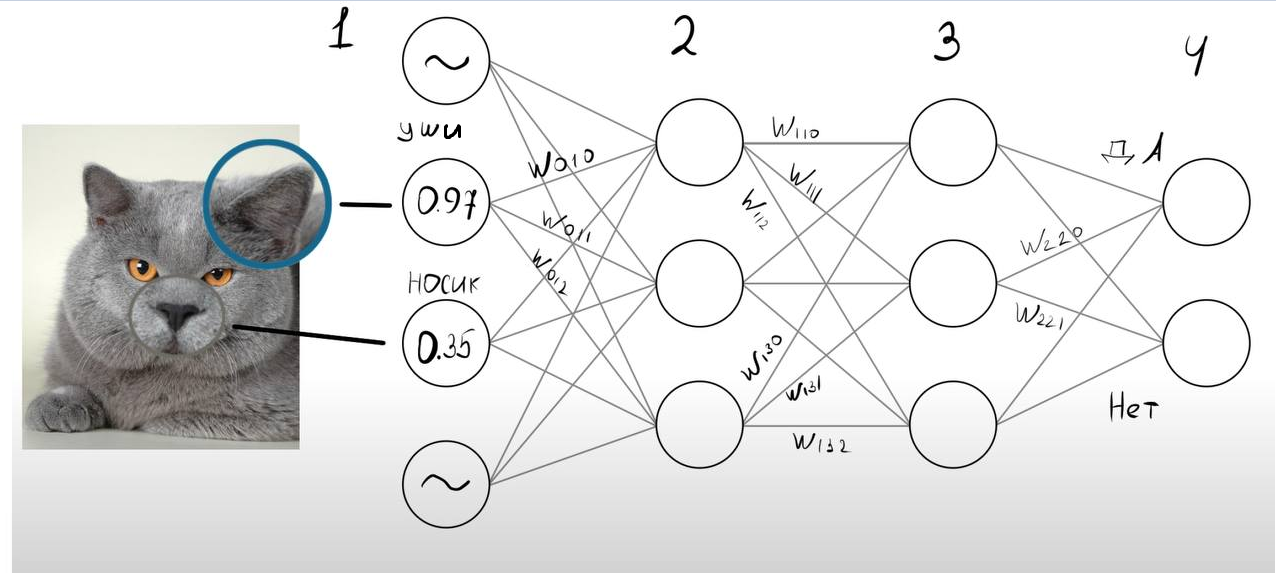


Рисунок 2 – Заполнение входного слоя нейросети

После заполнения входного слоя значениями сеть должна получить значения нейронов на следующих слоях. Для этого из каждого нейрона текущего слоя к каждому нейрону следующего слоя идет “нить”, но это не простая нить на ней располагается вес данной связи. Идя таким образом через нить значение нейрона сначала умножается на вес этой нити и только потом доходит до нейросети. Значение нейрона складывается из всех чисел пришедших ему через нити. Так же перед тем как нейрон передаст свое значение на нити к следующему слою, он должен “пройти” через функцию активации, в зависимости от значения нейрона она решает пускать ли значение нейрона дальше или “обнулить” его, например если значение меньше 0.2 то оно обнуляется. Активационных функций много, в своей программе я использовал функцию ReLu о которой немного подробнее расскажу позже. Меж тем сеть уже досчитала значения на выходном слое, давайте посмотрим.

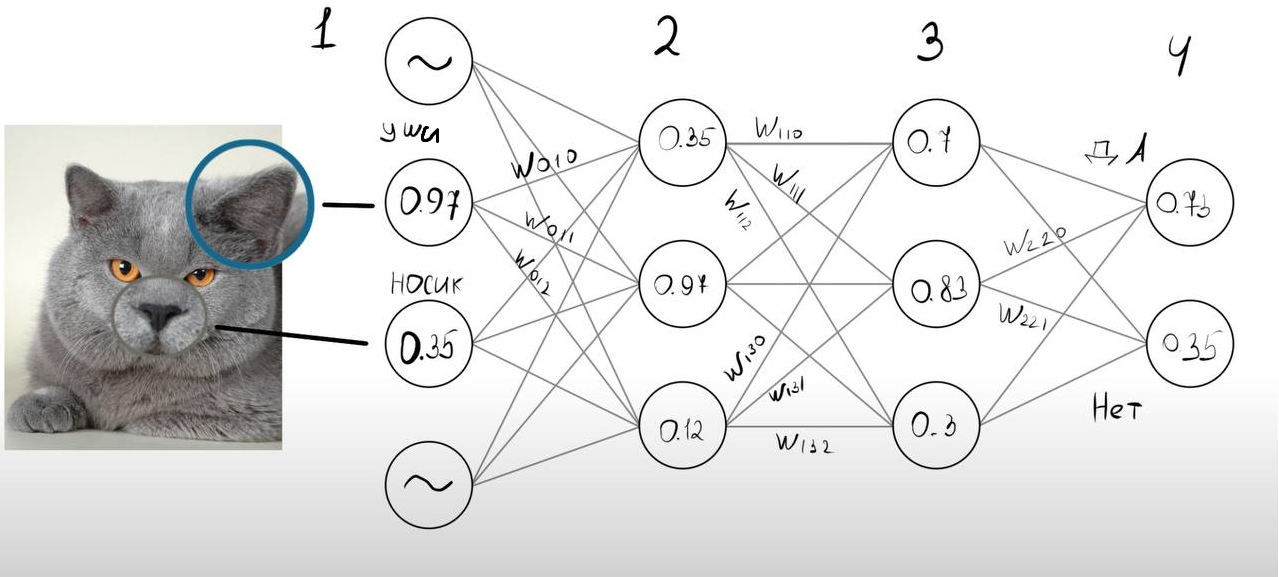


Рисунок 3 – Результат работы Котовой сети

Мы получили значения на выходном слое, где значение больше там, по мнению нейросети и находится ответ. Она считает что это все-таки кот, что ж, похвально. Но всегда можно добиться результата лучше. Давайте ее обучим.

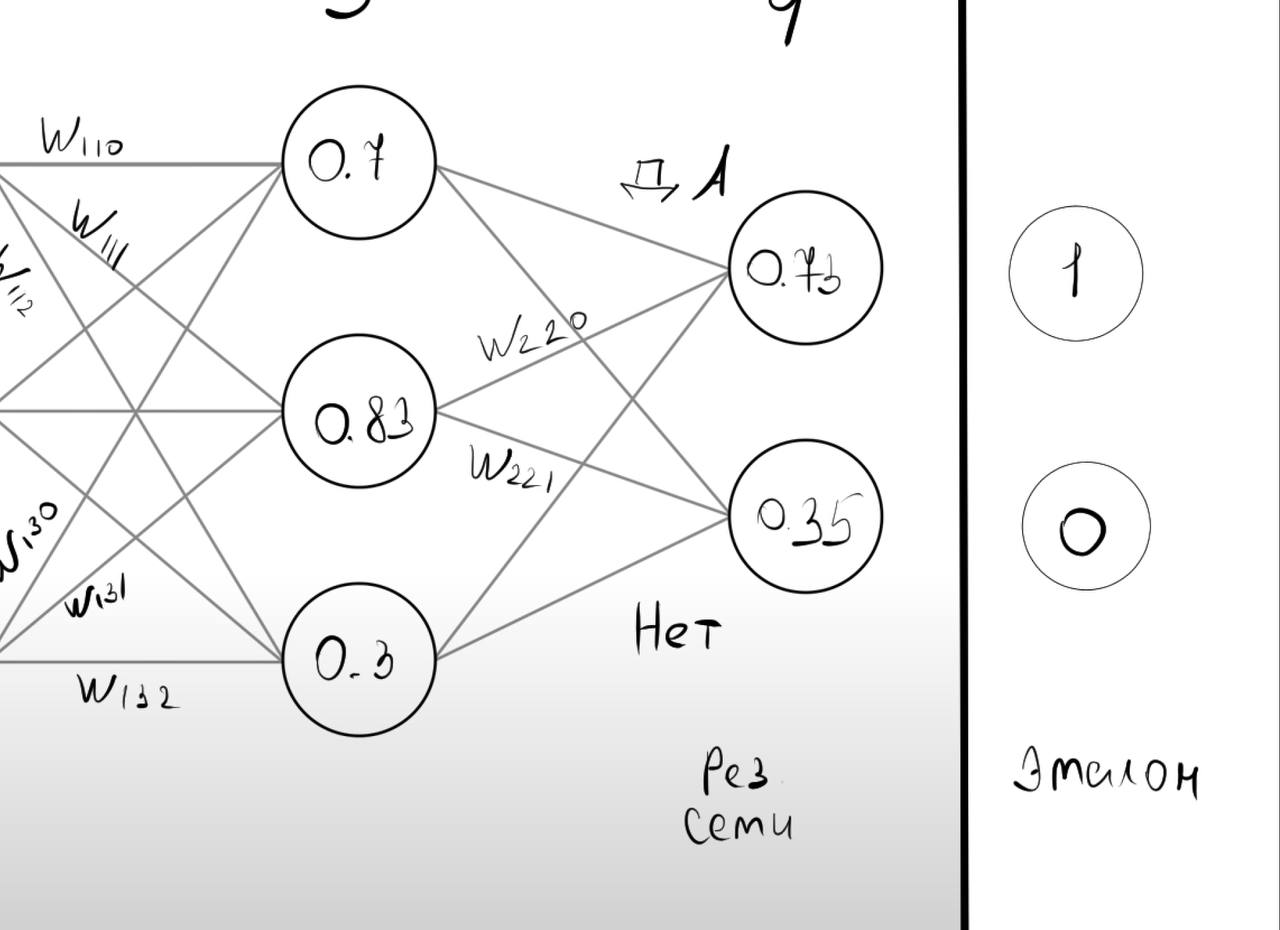


Рисунок 4 – сравнение результатов сети с эталонными

Что бы “показать” нейросети где она ошиблась нужно посчитать ошибку значений, а именно на сколько значения сети отличаются от эталонных и в какую сторону. Эталонными же будем считать следующие значения: 1 – на нейроне с правильным ответом и 0 – на всех остальных.

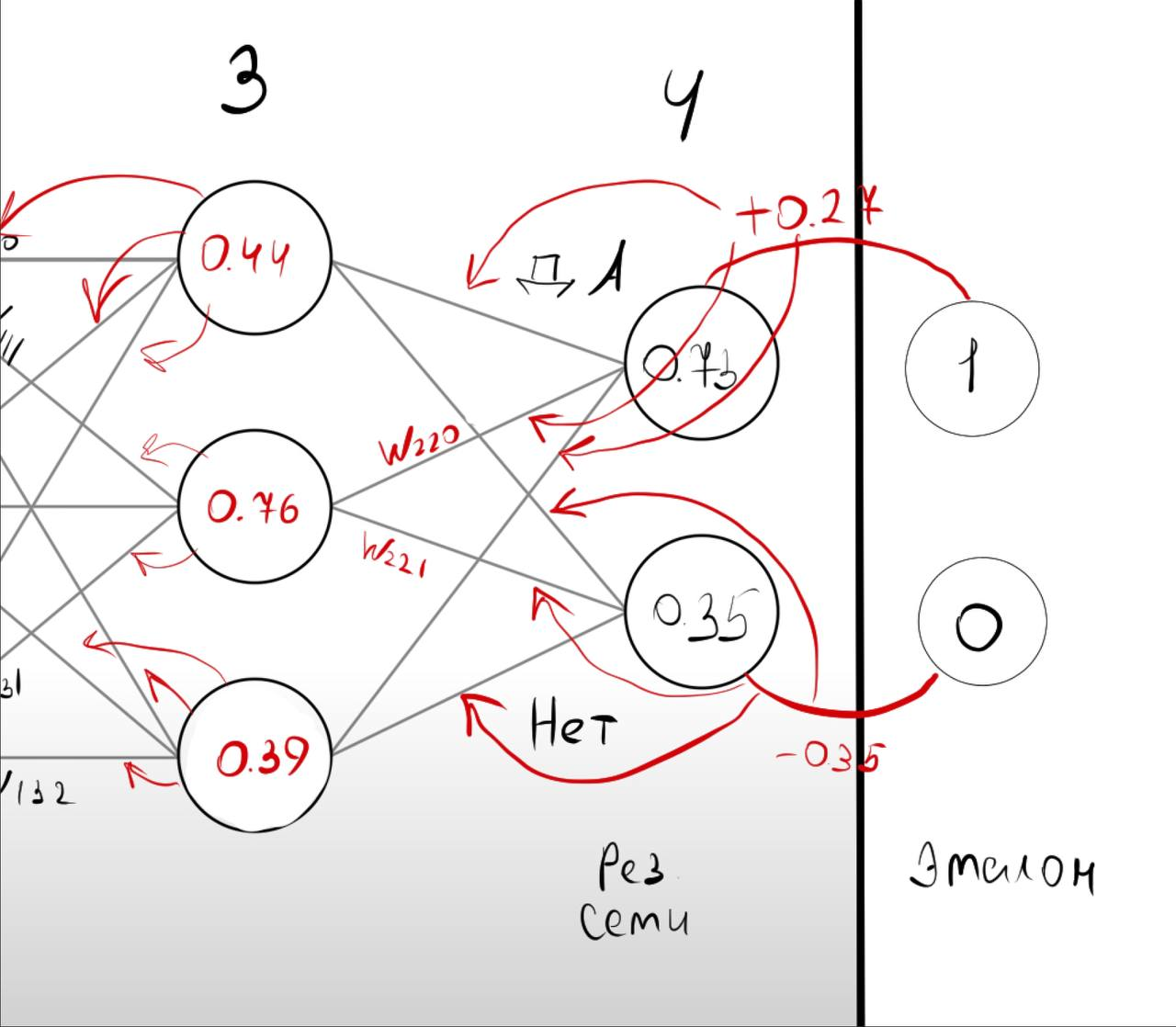


Рисунок 5 – изменение весов на основе посчитанной ошибки

После того как мы полистали ошибку, мы изменяем значения весов на нитях идущих к этим нейронам что бы итоговое значение этих нейронов было ближе к эталонному, аналогичные действия применяем и с предыдущими слоями вплоть до входного слоя. То что мы сейчас сделали, называется методом обратного распространения ошибки и именно так обучаются нейронные сети.

# ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

## ИСПОЛЬЗОВАННЫЕ КЛАССЫ И ИХ МЕТОДЫ

### КЛАСС MATRIX

Рассмотрим класс Matrix*:*

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Класс Matrixи метод initialization

Все достаточно просто – Мы создаем матрицу, выделяя под динамический массив и обнуляем ее.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – метод random

Этот метод заполняет матрицу случайными значениями. Такую формулу я выбрал что бы примерно соответствовать допустимому интервалу значений.

В данном случае это числа от 0 до примерно 0.88.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – метод Multiply

Этот метод умножает матрицу на вектор столбец, таким образом мы в будущем сможем получать значения нейронов скрытого слоя. Здесь const Matrix& m - константная ссылка на объект класса Matrix.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – метод MultiplyTransport

Этот метод аналогичен предыдущему за исключением того, что на вектор столбец умножается не обычная, а транспонированная матрица.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Метод vectorSum и перегрузка оператора ( )

Согласно названию метод возвращает векторную сумму 2-х векторов. Кстати делаю это и предыдущие 2 метода для того что бы к ним ссылаться напрямую а не через объект.

Перегружаю оператор круглых скобок я для того что бы он возвращал конкретный элемент матрицы по ее индексу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – перегрузка операторов << и >>

В обоих случаях идем последовательно по элементам матрицы и записываем эл-ты матрицы в поток. Соответственно при использовании с cout информация будет у нас на экране, а с ofstream в текстовом файле, аналогично для >> с cin и ifstream. Это пригодится нам для сохранения матрицы весов и загрузки матрицы весов из тестового файла.

### КЛАСС ACTIVATOR

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 – Метод activation

Собственно говоря функция активации, которая “решает” будет ли передано значение нейрона на следующий слой основываясь на его значении. Я использовал функцию ReLu из-за ее простоты. Её график можно увидеть ниже:

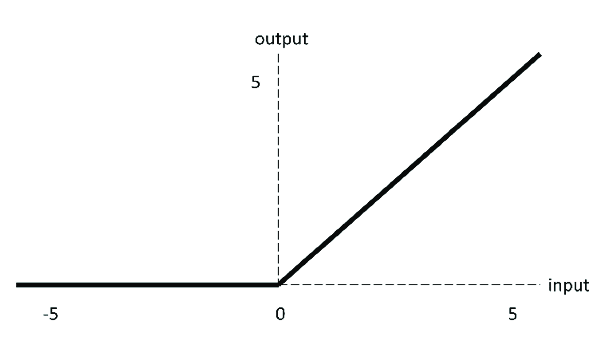


Рисунок 13 – график активационной функции ReLu

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14 – Метод activationDerivative

По сути клон предыдущего метода за исключением того что здесь используется клон активационной функции ReLu.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15 – метод activationDerivative типа double

Отличен от предыдущего метода лишь тем что не имеет цикла т.е. применяется для единственного нейрона а не для целого слоя и возвращает результат в виде числа типа double.

### КЛАСС NETWORК

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 – Данные класса, структура data\_Network и метод initCustomWeights

NetWork – самый важный класс всей программы. Именно в нем записаны все методы которые ответственны за получение результата нашей сети. Метод initCustomWeights позволяет пользователю перед тестом программы не обучать ее заново а загрузить уже откалиброванную матрицу весов что бы уже работать с ней и добиться лучшей точности.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 17 – метод initialization

Используя информацию из структуры data\_Network, а именно количество слоев нейронов (включая входной и выходной) и количество нейронов на каждом слое по сути инициализирует матрицу весов и нейроны ошибки выделяя под них память и заполняя случайными значениями матрицу весов.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 18 – методы CoutInfo, coutValues, setInput и searchMax

Достаточно простые методы по своей струкуре, поэтому рассмотрим их вместе. Метод CoutInfo выводит в консоль информацию о сети, а именно по сути данные структуры data\_NeеWork. Метод coutValues выводит значение нейронов выходного слоя. Метод setInput заполняет входной слой данными полученными из текстового файла, по сути он “вводит” на входной слой сети каждый пиксель картинки с числом, где 1 – это абсолютно закрашенный пиксель, а 0 – абсолютно не закрашенный. Метод searchMax и того проще. Он нужен нам для того что бы искать максимальное значение из нейронов массива и возвращать его индекс, его применение является финальным шагом следующего метода.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 19 – Метод DirectInnings

С помощью этого метода мы узнаем от сети, какое же число мы загодали, получая значения нейронов выходного слоя путем умножения значений предыдущих слоев на вес добавление к этому произведению нейроны ошибки и поиску максимального значения на выходном слое – у какого числа максимальное значение присущего ему нейрона – то число и нарисовано на картинке. Все просто.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 20 – Метод BackPropogation

Метод обратного распространения ошибки. Считаем ошибку для нейронов скрытого и выходного слоя путем сравнения их с эталонными значениями, вычисляя квадрат разницы между эталонным и полученным значением нейрона.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21 – Метод weightsUpdate

Имея данные полученные через метод обратного распространения ошибки обновляем веса для улучшения качества последующих результатов складывая значение веса с ошибкой умноженного на коэффициент обучения поданного на вход методу.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22 – Метод saveWeights

Полученную матрицу весов сохраняем в текстовой файл, дабы ей можно было пользоваться повторно.

### СВОБОДНЫЕ МЕТОДЫ И СТРУКТУРЫ

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23 – Структура dataInformation и Метод ReadDataNetwork

Структура dataInformation содержит в себе картинку с цифрой попиксельно и саму цифру которая нарисована на картинке. Метод же ответственен за чтение txt файла с информацией и связывает данные оттуда с ранее упомянутой структурой data\_NetWork.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24 – Метод ReadData

C помощью данного метода функция тащит из файла с картинками их количество а затем для каждой из них связывает значения из файла с переменными структуры dataInformation.

## КОД ПРОГРАММЫ В MAIN( )

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24 – начало main( )

В начале мы инициализируем переменные, объекты разных классов производим начальную настройку сети, выводя информацию о ее текущих характеристиках в консоль, спрашиваем, хочет ли пользователь использовать уже откалиброванную матрицу весов, если да, по просим название файла с весами.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 25 – меню управления программой и обучение сети.

Меню управления программой я реализовал через оператор switch. Что бы пользователю сделать то или иное действие достаточно ввести нужную цифру. Если пользователь хочет обучать нейросеть то программа просит указать количество эр обучения (прогонов файла с рисунками для обучения) затем идет процесс получения значений от сети, путем сравнения их с эталонными получаем значения ошибок нейронов и обновляем веса используя градиентный спуск для коэффициента обучения в методе weightsUpdate. После каждой завершенной эры выводим время потраченное на последний прогон и процент правильных ответов данных сетью, дабы видеть насколько сеть стала точней. После завершения обучения выводим время потраченное на все прогоны.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26 – тест точности сети

Если мы хотим проверить эффективность сети на примерах, не участвующих в обучении сети, то используем другой файл с числами, делаем действия аналогичные обучению сети за исключением того что эра всегда 1 сеть в процессе не обучается и таймера выполнения здесь нет.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27 – Индивидуально загаданное число и конец программы

Если же мы хотим загадать свою цифру нейросети то нужно предварительно загрузить картинку в файл и выбрать 3 вариант действий программы. В консоль она выведет результат своей работы. Если же введено некорректное число в меню программа сообщит об этом и завершит свою работу.

# ОБ ОССОБЕННОСТИ ФОРМАТА ВХОДНЫХ ДАННЫХ

Дабы облегчить работу с перенесением картинки с нарисованным числом в числовой формат с которым сеть должна работать, мною было принято решения использования датасета рукописных чисел Mnist ибо там уже собрано колоссальное количество чисел для обучения и тестирования сети.



Рисунок 28 – Примел рукописных чисел из базы данных Mnist

Более того числа доступные для открытого скачивания хранятся в txt формате а именно в виде двумерного массива чисел где 1 – это полностью закрашенный пиксель а 0 – полностью не закрашенный.

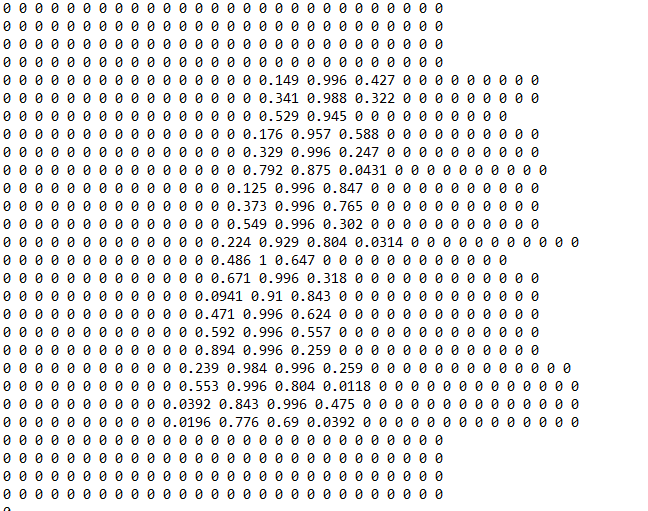


Рисунок 29 – Пример того как выглядит число, в данном случае это единица

А поэтому для того что бы проверить нейросеть дав ей “угадать” свою цифру, в текстовый файл мы должны положить данные аналогичного формата. Ниже представлен пример “загадывания” нейросети числа 9 созданного самим.

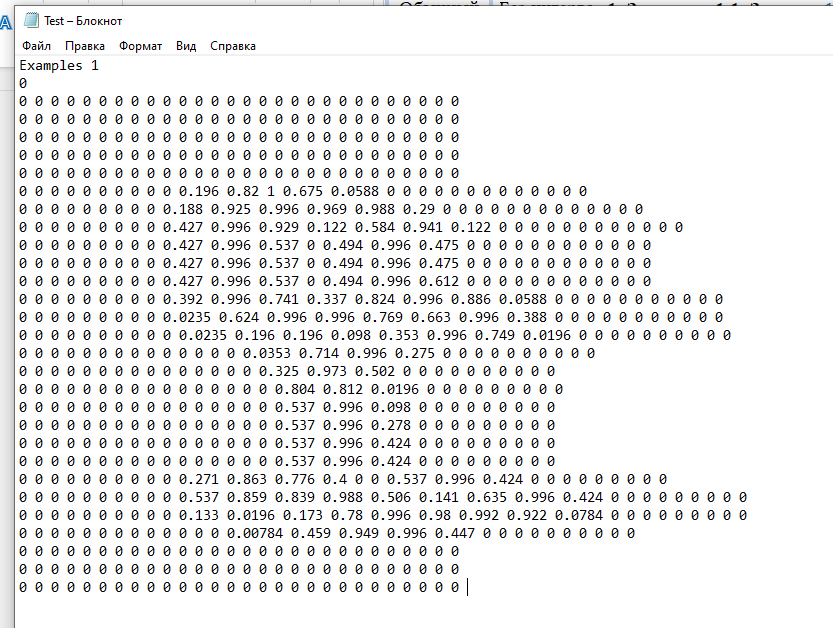


Рисунок 30 – Помещение “рисунка” числа 9 в тестовый файл

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 31 – Результат работы программы

# ВЫВОДЫ

Была разработана программа, которая из себя представляет многослойный перцептрон угадывающий цифры. Сеть умеет обучаться таким образом, что при каждой эре обучения ее эффективность увеличивается.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Лекции по процедурному программированию. Лектор: Каширская Е. Н. МИРЭА. 2021
2. Видео: “But what is a neural network? | Chapter 1, Deep learning” автор: 3blue1brown
3. Лекция Михаила Бурцева «Искусственный интеллект: что могут нейронные сети и как они изменят нашу жизнь?»
4. Mnist Database of Handwritten Digits