Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин
Дисциплина: Программирование на языках
высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
на тему
«Распознавание слов с помощью
нейросети»

БГУИР КП 1-40 02 01 321

Студент А.Э. Мынзул

Руководитель Е.В. Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

_____
(подпись)
_____20
г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Мынзул Александр Эдуардович*

Тема проекта «Распознавание слов с помощью нейросети»

2. Срок сдачи студентом законченного проекта *11 декабря 2023 г.*

3. Исходные данные к проекту *AZ_Handwritten.csv (база данных для нейросети)*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)
1. Лист задания.
2. Введение.
3. Обзор литературы.
 3.1. Обзор методов и алгоритмов решения поставленной задачи.
 3.2. Разработка требований к функционалу.
4. Функциональное проектирование.
 4.1. Структура входных и выходных данных.
 4.2. Разработка диаграммы классов.
 4.3. Описание классов.
5. Разработка программных модулей.
 5.1. Разработка схем алгоритмов(два наиболее важных метода).
 5.2. Разработка алгоритмов (описание алгоритмов по шагам, для двух методов).
6. Результаты работы.
7. Заключение
8. Литература
9. Приложения

5. Перечень графического материала (с точным обозначением обязательных

чертежей и графиков)

1. Диаграмма классов.

2. Схема алгоритма обратного распространения ошибки

3. Схема алгоритма инициализации нейронной сети

6. Консультант по проекту (с обозначением разделов проекта) Е. В. Богдан

7. Дата выдачи задания *15.09.2023г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %;

разделы 2, 3 – 10 %;

разделы 4 к –20 %;

разделы 5 к – 35 %;

раздел 6,7,8 – 5 %;

раздел 9 к – 5%;

оформление пояснительной записки и графического материала к 15.12.23 – 10 %

Защита курсового проекта с 21.12 по 28.12.23г.

РУКОВОДИТЕЛЬ Е.В. Богдан

(подпись)

Задание принял к исполнению_____

(дата и подпись студента)

Содержание

ВВЕДЕНИЕ

В последние десятилетия, развитие компьютерных технологий и искусственного интеллекта привело к революционным изменениям в области обработки естественного языка и анализа текстов. Нейронные сети, включая перцептрон, представляют собой семейство моделей машинного обучения, которые позволяют эффективно анализировать и обрабатывать сложные текстовые данные. Однако, в свете увеличивающейся сложности и разнообразия текстовой информации, важно более детально изучить особенности и ограничения перцептрона в контексте задач распознавания текста.

В этом проекте особое внимание уделяется изучению архитектуры перцептрона, его способности к обучению на основе набора данных, и способности обобщать полученные знания на новые текстовые данные. Анализируя основные компоненты перцептрона, такие как входные слои, скрытые слои и выходные слои, и их взаимодействие в контексте распознавания текста, исследование пытается выявить оптимальные методы настройки параметров модели для достижения наилучших результатов.

Кроме того, в данной работе проводится сравнительный анализ эффективности перцептрона с другими распространенными архитектурами нейронных сетей, такими как рекуррентные нейронные сети и сверточные нейронные сети, с целью выявления преимуществ и ограничений перцептрона в сравнении с более сложными моделями. Такой анализ позволит понять, насколько перцептрон остается актуальным и эффективным инструментом в современной парадигме анализа текста, а также определить области его применения, где он может продемонстрировать наилучшие результаты.

Проект выполенен с помощью фреймворка Qt с использованием языка программирования C++.

1. ЗАДАНИЕ НА КУРСОВУЮ

Цель данной курсовой работы заключается в разработке проекта по распознаванию текста с применением перцептрона в нейронных сетях, с учетом основных принципов объектно-ориентированного программирования (ООП) и использования инструментов Qt для создания графического интерфейса пользователя.

В процессе выполнения работы предполагается изучение и применение принципов ООП при проектировании и реализации нейросетевой модели на основе перцептрона, а также использование Qt для создания интуитивно понятного и удобного пользовательского интерфейса. Рекомендуется уделять внимание аспектам модульности и расширяемости кода, соблюдая принципы ООП в контексте Qt.

При работе с Qt, следует рассмотреть возможности интеграции графического интерфейса с функциональностью нейросетевой модели, предоставляя пользователям возможность обучения нейросети, распознавания слов и загрузки необходимых конфигураций. Это включает в себя создание интуитивно понятных элементов управления для проведения обучения, ввода текстовых данных для распознавания, и управления конфигурациями нейросети.

Итоговый проект должен обеспечивать не только точное распознавание текста с использованием перцептрона, но также предоставлять возможности обучения нейросети, распознавания слов, и удобную загрузку необходимых конфигураций, делая его многофункциональным инструментом в области обработки текстовой информации.

2. ОБЗОР ЛИТЕРАТУРЫ

2.1. Обзор методов и алгоритмов решения поставленной задачи

2.1.1. Многослойный перцептрон

Многослойный перцептрон является важным типом нейронных сетей, состоящим из нескольких слоев, каждый из которых содержит набор нейронов, связанных между собой. Он состоит как минимум из трех типов слоев: входного слоя, скрытых слоев и выходного слоя.

Входной слой: Это первый слой сети, который принимает входные данные и передает их дальше по сети. Количество нейронов в этом слое соответствует размерности входных данных.

Скрытые слои: Эти слои находятся между входным и выходным слоями и выполняют вычисления между ними. Каждый нейрон в скрытом слое принимает на вход взвешенную сумму выходных данных предыдущего слоя и проходит через функцию активации. Количество скрытых слоев и количество нейронов в каждом слое - это параметры, которые определяются конкретной задачей и структурой данных.

Выходной слой: Этот слой принимает данные из последнего скрытого слоя и выдает окончательные результаты. Количество нейронов в выходном слое зависит от типа задачи, которую необходимо решить. Например, в задачах бинарной классификации может быть один нейрон с сигмоидной функцией активации, в то время как в задачах многоклассовой классификации количество нейронов будет соответствовать количеству классов, а функция активации может быть softmax.

Веса и смещения в каждом нейроне настраиваются в процессе обучения сети с использованием алгоритма обратного распространения ошибки (Back Propagation). Этот алгоритм позволяет сети корректировать веса таким образом, чтобы минимизировать ошибку между прогнозами модели и фактическими данными. Для этого используется градиентный спуск, который позволяет найти направление наискорейшего убывания функции потерь и соответствующим образом обновлять веса сети.

Функции активации, такие как ModReLU, Sigmoid или Tanh, добавляют нелинейность в вычисления сети, позволяя ей моделировать более сложные зависимости в данных. Они применяются к взвешенным суммам входов для введения нелинейности в активации нейронов.

Преимущества многослойных перцептронов включают их способность обучаться сложным нелинейным зависимостям в данных, что делает их мощным инструментом для решения разнообразных задач, включая классификацию, регрессию и аппроксимацию сложных функций. Однако важно учитывать проблему переобучения, которая может возникнуть при недостаточном количестве данных или слишком сложной структуре сети. Для борьбы с этой проблемой применяются различные методы регуляризации, такие как

Dropout и L1/L2 регуляризация.

2.1.2. База данных NIST Special Database 19 second edition

Эта база данных является широко признанным набором данных, используемым в области распознавания рукописного текста и символов. Она предоставляет разнообразный и обширный набор образцов рукописных символов, включая цифры, буквы и другие символы, что делает ее ценным ресурсом для обучения и тестирования моделей распознавания текста.

NIST Special Database 19 содержит разнообразные наборы данных, собранные из рукописных документов и различных источников, и предназначена для обучения и оценки эффективности алгоритмов распознавания текста. База данных включает в себя рукописные образцы различных символов, представленных в формате изображений или в виде набора пикселей, что позволяет использовать их напрямую в качестве входных данных для обучения нейронной сети.

Основные особенности и преимущества NIST SD 19 второго издания включают следующее:

- Разнообразие данных: База данных предлагает разнообразные образцы рукописных символов, включая различные стили написания, размеры и формы, что позволяет моделям обучаться на широком спектре вариаций данных и улучшать их обобщающие способности.
- Размер выборки: NIST SD 19 включает большой объем данных, что позволяет исследователям и разработчикам обучать модели на больших обучающих выборках и оценивать их производительность на разнообразных тестовых наборах.
- Признанный стандарт: База данных NIST SD 19 широко признана и используется в научных исследованиях и практических приложениях в области распознавания текста, что делает ее важным и авторитетным ресурсом для оценки производительности различных моделей и алгоритмов.
- Доступность и удобство использования: NIST SD 19 доступна для исследователей и разработчиков в удобном формате, что облегчает ее использование в обучении и тестировании различных моделей машинного обучения, включая многослойные перцептроны для распознавания текста.

2.1.3. Метод обратного распространения ошибки

Метод обратного распространения ошибки (Back Propagation) - это метод обучения нейронных сетей, который заключается в вычислении градиента функции потерь по весам сети с целью корректировки их значений для минимизации ошибки [1]. Он состоит из нескольких шагов:

- Вычисление ошибки:

После прямого распространения сеть делает предсказания и вычисляет ошибку, используя функцию потерь, например, среднеквадратичную ошибку для задач регрессии или кросс-энтропию для задач классификации.

• Обратное распространение ошибки:

Градиент функции потерь вычисляется с помощью правила цепного дифференцирования. Градиент показывает, как изменится ошибка при изменении весов. Он вычисляется для каждого слоя сети, начиная с выходного слоя и двигаясь к входному слою.

$$\frac{dE}{dw} = \frac{dE}{dz} \cdot \frac{dz}{dw} \tag{2.1}$$

где z - взвешенная сумма входных данных, E - функция потерь, w - вес нейрона

• Градиентный спуск:

После вычисления градиентов происходит обновление весов сети с использованием метода градиентного спуска. Веса корректируются в направлении, противоположном градиенту, с учетом скорости обучения, которая определяет величину шага оптимизации.

$$w_{\text{new}} = w_{\text{old}} - \eta \cdot \frac{dE}{dw} \tag{2.2}$$

где η - скорость обучения, определяющая шаг гардиентного спуска Этот процесс повторяется для каждого обучающего примера в наборе данных и на каждой эпохе обучения. Он позволяет модели улучшать свои предсказания путем корректировки весов в направлении, уменьшающем ошибку предсказаний.

2.1.4. Метод прямого распространения

Прямое распространение - это процесс, при котором данные проходят через нейронную сеть от входного слоя к выходному слою. Во время прямого распространения каждый нейрон в сети получает входные данные, умножает их на соответствующие веса, складывает результаты и применяет функцию активации для генерации выходного сигнала [2].

• Взвешенная сумма: Для каждого нейрона j в слое l, сначала вычисляется взвешенная сумма его входов. Это происходит путем умножения каждого входного значения на соответствующий ему вес и суммирования результатов вместе с добавлением смещения (bias):

$$z_j^l = \sum_i w_{ji}^l a_i^{l-1} + b_j^l \tag{2.3}$$

где z_j^l - взвешенная сумма в слое l, w_{ji}^l - вес между нейронами, a_i^{l-1} - выход

предыдущего слоя,b_j^l - смещение (bias)

• Функция активации: После вычисления взвешенной суммы, результат подается на функцию активации, которая добавляет нелинейность к выходному сигналу нейрона. Функции активации, такие как ModReLU, сигмоида и гиперболический тангенс, позволяют модели изучать более сложные зависимости между входными данными и выходными предсказаниями.

• Прохождение через слои: Этот процесс происходит для каждого слоя сети, начиная с входного слоя и двигаясь к выходному слою. Каждый слой принимает выходной сигнал предыдущего слоя как вход и вычисляет свой собственный выход, который затем передается следующему слою.

• Выходной слой: В конце прямого распространения вычисляется окончательный выходной сигнал сети, который представляет собой предсказание или результат работы нейронной сети для заданного входа.

2.1.5. Функция активации ModReLU

Функция ModReLU, выглядит следующим образом:

$$f(x) = \max(0.01x, x) \tag{2.4}$$

Эта функция является вариантом ReLU с параметром наклона, который в данном случае равен 0.01. Это означает, что для отрицательных значений функция действует как линейная функция с наклоном 0.01, а для положительных значений - как обычная функция ReLU. Такое определение функции помогает избежать проблемы "мёртвых нейронов" (dead neurons) при использовании стандартной ReLU, когда нейрон перестает обновляться на этапе обратного распространения ошибки.

2.1.6. Dropout

Метод Dropout представляет собой технику регуляризации, при которой в процессе обучения случайно выбранные нейроны и их связи временно исключаются из сети с определенной вероятностью. Это означает, что на каждой итерации обучения некоторые нейроны не будут участвовать в процессе обновления весов, и в результате модель обучается более устойчиво и становится менее склонной к переобучению [3].

Во время применения Dropout вероятность исключения каждого нейрона обычно устанавливается на уровне от 0.2 до 0.5, хотя оптимальное значение может варьироваться в зависимости от конкретной задачи и архитектуры сети. Этот метод также способствует уменьшению взаимосвязей между нейронами и препятствует развитию сложных зависимостей, что позволяет модели обобщать лучше на новые данные.

2.1.7. Предварительная обработка изображения

Обработка изображений перед использованием нейронной сетью при распознавании текста - важный этап.

Когда мы подготавливаем изображения для использования нейронной сетью в задаче распознавания текста, первый этап - сегментация текста. Это означает разделение изображения на более мелкие области или символы, что облегчает последующее распознавание текста.

Далее идет этап нормализации, где изображения приводятся к общему стандарту. Разные изображения могут иметь разные размеры, освещение и углы съемки, и нормализация помогает унифицировать их параметры, что упрощает дальнейший анализ.

Чтобы улучшить качество распознавания, применяется фильтрация шума. Изображения могут содержать шум, который может мешать правильному выделению текста. Фильтрация шума включает в себя удаление нежелательных элементов или артефактов с изображения с целью повышения четкости и качества текста.

Центрирование и выравнивание текста - еще один важный шаг. Центрирование помогает достичь однородности положения текста на изображении, а выравнивание обеспечивает единообразную ориентацию и расположение текста, что снова способствует лучшему распознаванию.

Наконец, преобразование формата изображения может потребоваться, поскольку нейронные сети могут требовать данные определенного формата. Таким образом, изображения могут подвергаться преобразованию, например, из формата BMP в CSV, чтобы быть подходящими для входа в нейронную сеть.

2.1.8. Сохранение весов

Сохранение весов нейросети после обучения является важным этапом для сохранения полученных знаний и параметров модели. Это позволяет в будущем загружать и использовать обученную модель без необходимости повторного обучения, что упрощает процесс инференса и применения модели на новых данных.

3. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1. Структура входных и выходных данных

Входные данные:

• Изображения слов размером 28 на 28 пикселей: Входные данные представляют собой матрицы пикселей, где каждый пиксель содержит информацию о яркости или цвете соответствующей точки на изображении. Размер изображения составляет 28x28 пикселей, что обеспечивает достаточное разрешение для захвата деталей текста.

• Разделение слова на буквы: После получения изображения размером 28x28 пикселей каждое слово разбивается на отдельные буквы для дальнейшей обработки. Это позволяет нейросети обрабатывать каждую букву независимо и предсказывать соответствующий символ.

• Масштабирование изображений: Каждая буква подвергается процессу масштабирования, чтобы убедиться, что они имеют одинаковый размер и соответствуют стандарту для подачи на вход нейронной сети. Это помогает обеспечить однородность входных данных и упрощает процесс обучения.

• Подача каждой буквы на вход нейронной сети: Каждая буква, представленная в виде матрицы пикселей, подается на входной слой нейронной сети, который состоит из 784 нейронов, соответствующих каждому пикселю входного изображения размером 28x28.

Выходные данные:

• Для задачи распознавания текста выходные данные могут быть представлены в формате, подходящем для классификации символов или букв. Это может быть представление в виде One-Hot Encoding, где каждый символ представлен вектором с одной активной (значением 1) позицией, соответствующей индексу символа в алфавите.

Такая структура данных позволяет нейронной сети принимать изображения слов в качестве входных данных, обрабатывать их для извлечения информации о каждой букве, и предсказывать соответствующий символ или букву на выходе с помощью One-Hot Encoding или подобного метода классификации.

3.2. Разработка диаграмм классов

Диаграмма классов в объектно-ориентированном проектировании программного обеспечения занимает важное положение, предоставляя разработчикам наглядное представление структуры системы. Суть этого инструмента заключается в выделении ключевых классов, их атрибутов и взаимосвязей, что существенно облегчает понимание и проектирование архитектуры программы.

При ближайшем рассмотрении диаграммы классов можно выделить три основных компонента для каждого класса. В верхней части размещается имя класса, выделенное полужирным шрифтом и центрированное для наглядности. Средняя часть области отведена для перечисления атрибутов (полей) класса, предоставляя полезную информацию о его характеристиках. В нижней части области перечисляются методы класса, определяя тем самым функциональность данного класса.

Цель создания диаграммы классов - предоставить визуальное представление структуры системы, выделить ключевые классы, их атрибуты и взаимосвязи.

Этот инструмент играет ключевую роль в повышении читаемости, поддерживаемости и расширяемости кодовой базы, что существенно влияет на качество проектирования программного продукта.

Подробную диаграмму классов можно найти в приложении А, где предоставлена детальная структура системы, что обеспечивает необходимый уровень абстракции для полного понимания статической архитектуры программы.

Диаграмма классов представлена в приложении А.

3.3. Описание классов

3.3.1. Класс `ActivateFunction`

Класс `ActivateFunction` обеспечивает реализацию активационных функций для нейронов в нейронной сети и позволяет эффективно применять функции активации ModReLU и их производные в процессе обучения и прямого распространения в нейронной сети.

Методы:

• `ModRelu(double* NeuronValue, int NeuronsOnLayer)`: Этот метод реализует функцию активации ModReLU. Он принимает указатель на массив `NeuronValue`, представляющий значения нейронов на слое, и количество нейронов на слое `NeuronsOnLayer`. Для каждого нейрона в массиве применяется функция активации ModReLU.

• `ModReluDerivate(double* NeuronValue, int NeuronsOnLayer)`:Этот метод рассчитывает производную функции активации ModReLU для каждого нейрона на слое. Он принимает указатель на массив `NeuronValue`, представляющий значения нейронов на слое, и количество нейронов на слое `NeuronsOnLayer`. Для каждого нейрона в массиве вычисляется производная функции ModReLU.

• `ModReluDerivate(double NeuronValue)`: Этот метод представляет собой перегруженную версию функции для вычисления производной функции ModReLU. Он принимает значение `NeuronValue` в качестве аргумента и возвращает производную функции для данного значения нейрона.

3.3.2. Структура `Pixel`

Атрибуты:

• `letter`: представляет букву или символ, соответствующий пикселю.

• `black`: представляет значение яркости пикселя.

• `id`: идентификатор пикселя.

3.3.3. Класс `NeuralNetwork`

Поля:

• `layers`: количество слоев в нейронной сети.

• `neurons_on_layer`: массив, содержащий количество нейронов на каждом слое.

• `function`: объект класса ActivateFunction, отвечающий за активационные функции нейронов.

• `weight`: объект класса Matrix, представляющий веса между нейронами.

• `neurons`: двумерный массив, представляющий значения нейронов на каждом слое.

• `error_of_neurons`: двумерный массив, представляющий ошибку нейронов на каждом слое.

• `bias`: одномерный массив, содержащий смещения для каждого слоя.

• `bias_weight`: двумерный массив, представляющий веса смещений.

Методы:

• `double ForwardFeed()`: Этот метод отвечает за прямое распространение данных через нейронную сеть.

• `void BackPropogation(double predict)`: Этот метод реализует алгоритм обратного распространения ошибки в нейронной сети.

• `void UpdateWeights(double lr)`: Этот метод обновляет веса нейронной сети с учетом заданного коэффициента обучения.

• `void ShowValues(int L)`: Этот метод печатает значения нейронов для заданного слоя.

• `void NetworkInitialization()`: Этот метод инициализирует нейронную сеть с заданными параметрами.

• `void InputData(double* picture, int train)`: Этот метод отвечает за ввод данных в нейронную сеть.

• `double MaxIndex()`: Этот метод выполняет поиск индекса нейрона с максимальным значением.

• `void WeightSaver()`: Этот метод сохраняет веса нейронной сети.

• `void WeightReader()`: Этот метод считывает сохраненные веса нейронной сети.

3.3.4. Класс `Matrix`

Поля:

• `matrix`: двумерный массив, представляющий матрицу.

• `row`: количество строк в матрице.

• `column`: количество столбцов в матрице.

Методы:

• `static void Multipl(Matrix& mas, double* neurons, double* Mult_Matrix, int numb_neurons, int start, int end)`:Этот метод отвечает за умножение матрицы на вектор нейронов с определенным диапазоном индексов. Входные параметры включают матрицу `mas`, вектор `neurons`, массив `Mult_Matrix`, представляющий результат умножения, количество нейронов `numb_neurons`, а также индексы start и end, определяющие диапазон обрабатываемых элементов.

• `static void MultiTrans(Matrix& mas, double* neurons, double* Mult_Matrix, int numb_neurons)`:Этот метод отвечает за умножение транспонированной матрицы на вектор нейронов. Входные параметры

включают матрицу mas, вектор neurons, и массив Mult_Matrix, представляющий результат умножения. Количество нейронов передается как параметр numb_neurons.

• static void Sum(double* bias, double* Mult_Matrix, int hid_neurons):Этот метод отвечает за суммирование смещений и умноженной матрицы. Входные параметры включают массив смещений bias, массив Mult_Matrix, представляющий результат умножения, и количество скрытых нейронов hid_neurons.

• double& operator()(int i, int j): Этот метод перегружает оператор круглых скобок для доступа к элементу матрицы по индексам строки и столбца.

• double& GetElement(int i, int j): Этот метод возвращает элемент матрицы по заданным индексам строки и столбца.

• void Init(int numb_of_hid_neurons, int numb_of_input_neurons): Этот метод инициализирует матрицу с заданными размерами, принимая количество скрытых нейронов numb_of_hid_neurons и количество входных нейронов numb_of_input_neurons

• void Rand(): Этот метод заполняет матрицу случайными значениями. Он генерирует случайные числа для каждого элемента матрицы, обеспечивая тем самым случайную инициализацию.

4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1. Разработка схем алгоритмов

Метод `void Network::NetworkInitialization(int conf)` готовит структуру нейронной сети для обучения с заданными параметрами, включая количество слоев, количество нейронов на каждом слое и начальные веса. Блок схема метода `void Network::NetworkInitialization(int conf)` приведена в приложении Б.

Метод `void Network::BackPropogation(double predict)` реализует алгоритм обратного распространения ошибки для обновления весов в нейронной сети в процессе обучения. Блок схема метода `void Network::BackPropogation(double predict)` приведена в приложении В.

4.2. Разработка алгоритмов

4.2.1. Алгоритм обратного распространения ошибки

В данном методе представлен алгоритм обратного распространения ошибки.

Шаг 1. Инициализация ошибок на выходном слое.

Шаг 2. Проходим по каждому нейрону на выходном слое `(layers - 1)`.

Шаг 3. Если индекс нейрона не совпадает с предсказанным числом `(predict)`, устанавливаем ошибку как отрицательное произведение значения активации на модифицированную производную функции активации (ReLU).

Шаг 4. Если индекс нейрона совпадает с предсказанным классом, устанавливаем ошибку как разность единицы и значения активации, умноженной на модифицированную производную функции активации.

Шаг 5. Начиная с предпоследнего слоя и двигаясь в обратном порядке (от `layers - 2` до 1)

Шаг 6. Ошибок следующего слоя `(error_of_neurons[k + 1])`, результат записываем в вектор ошибок текущего слоя `(error_of_neurons[k])`.

Шаг 7. Для каждого нейрона на текущем слое умножаем значение ошибки нейрона на модифицированную производную функции активации.

4.2.2. Алгоритм инициализации нейронной сети

В данном методе производиться инициализация нейронной сети.

Шаг 1. Устанавливаем начальное значение генератора случайных чисел с использованием текущего времени `(srand(time(NULL)))`.

Шаг 2. Если `conf` равен 1, устанавливаем параметры для нейронной сети с тремя слоями и заданным количеством нейронов на каждом слое.

Шаг 3. Если `conf` равен 2, устанавливаем параметры для сети с четырьмя слоями.

Шаг 4. Если `conf` равен 3, устанавливаем параметры для сети с шестью слоями.
Шаг 5. Создаем массив `neurons_on_layer` для хранения количества нейронов на каждом слое.
Шаг 6. Выделяем память для массивов весов, весов смещения, смещений, активаций и ошибок нейронов.
Шаг 7. Создаем двумерные массивы neurons и `error_of_neurons` для хранения значений активаций и ошибок нейронов на каждом слое.
Шаг 8. Инициализируем значениями по умолчанию.
Шаг 9. Для каждого слоя создаем массив `neurons[i]` для хранения активаций нейронов.
Шаг 10. Если текущий слой не последний `(i < layers - 1)`, инициализируем матрицу весов `(weight[i])`, веса смещения `(bias_weight[i])`, и значение смещения `(bias[i])`.
Шаг 11. Инициализируем случайные значения весов и смещений.
Шаг 12. Завершаем инициализацию, готовя нейронную сеть для обучения с установленными параметрами и начальными значениями весов.

5. РЕЗУЛЬТАТ РАБОТЫ

В данном разделе представлены результаты работы разработанной системы распознавания с использованием нейронной сети. Проект включает в себя графический пользовательский интерфейс, который предоставляет удобный способ взаимодействия с функционалом системы.

Главное меню:

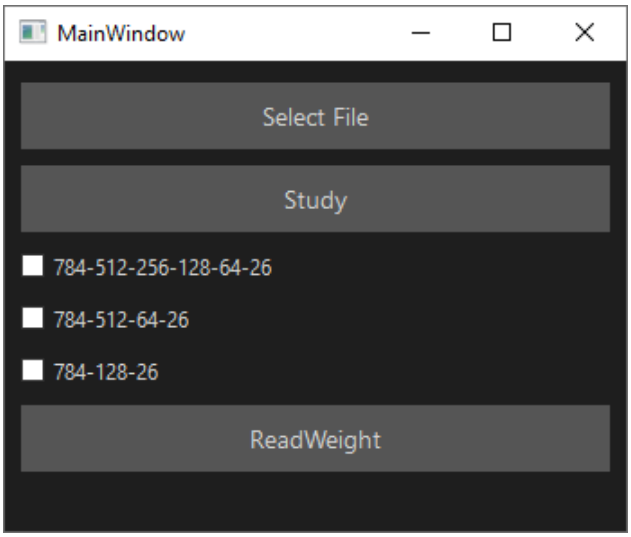

Рисунок 5.1. — Главное меню

На главном экране приложения пользователь встречается с интуитивно понятным интерфейсом. Здесь доступны следующие функциональности:

• Study: Функция, позволяющая запустить процесс обучения нейронной сети. Обучение направлено на улучшение навыков распознавания системы.

• ReadWeight: Опция, которая обеспечивает считывание весов нейронной сети из предварительно сохраненного файла. Это позволяет быстро загрузить предварительно обученные параметры и ускоряет процесс подготовки сети к работе.

• CheckBoxes: Три CheckBox предоставляют пользователю гибкость в настройке конфигурации нейронной сети. Это удобное средство для подбора оптимальных параметров в зависимости от поставленных задач.

• LoadFile: Кнопка, предназначенная для загрузки изображения в формате BMP. Это необходимо для последующего распознавания текста на загруженном изображении.

Считывание Весов:

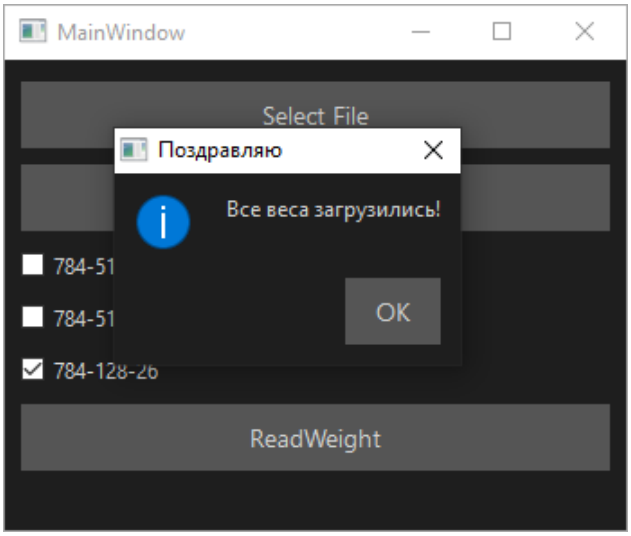

Рисунок 5.2. — Загрузка весов нейросети

Рисунок 5.2 подробно демонстрирует процесс считывания весов нейронной сети из файла. На данном этапе мы предоставляем пользователю детальный обзор процесса считывания весов нейронной сети из файла. Этот функционал важен, поскольку позволяет быстро загрузить предварительно обученные параметры, что является ключевым аспектом в оптимизации времени подготовки сети к работе.

После выполнения операции считывания весов, пользователь получает визуальное подтверждение в виде сообщения об успешном завершении процесса. Это уведомление не только информирует о корректности операции, но также создает уверенность в том, что нейронная сеть готова к последующей работе.

Введение этого функционала сопровождается работой над улучшением интерфейса, что обеспечивает пользователю максимальную удобочитаемость и понимание процесса считывания весов.

Распознавание Текста:

На рисунке 5.3 проиллюстрирован момент после загрузки изображения (рисунок 5.4), когда система успешно распознала текст. Пользователь видит сообщение с разгаданным текстом, что подчеркивает точность и эффективность работы системы распознавания.

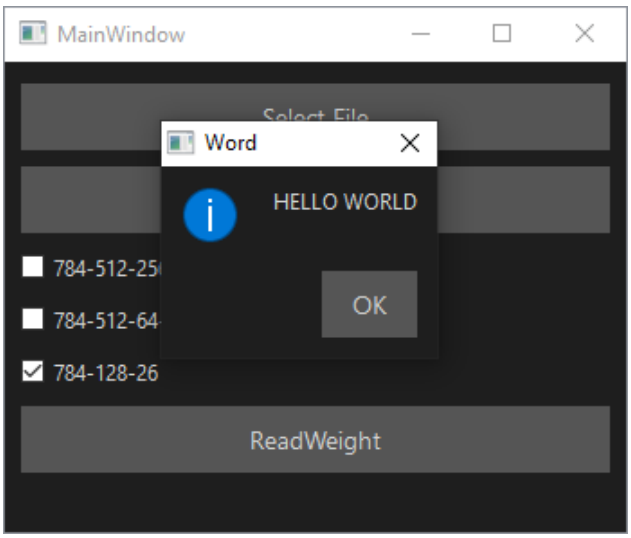

Рисунок 5.3. — Результат работы нейросети

Для демонстрации функционала распознавания текста, мы представляем пользователю изображение с фразой "Hello, World!", сохраненное в формате BMP. Этот этап подчеркивает способность нейронной сети к успешному распознаванию текста на изображениях.

Рисунок 5.4. — Исходный файл

Этот раздел подчеркивает не только успешное функционирование системы на различных этапах, но и обеспечивает пользователю визуальные подтверждения корректности выполненных операций. Разработанный графический интерфейс делает процесс взаимодействия с приложением понятным и удобным для пользователя, а полученные результаты подтверждают эффективность использования нейронной сети в задаче распознавания текста.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы был разработан и реализован программный продукт, представляющий собой систему распознавания текста с использованием нейронной сети. Проект охватывает широкий спектр аспектов, начиная от проектирования архитектуры программы до применения методов машинного обучения для обучения нейронной сети.

Одним из ключевых моментов в разработке является использование объектно-ориентированного подхода при проектировании кода. Применение принципов инкапсуляции, наследования и полиморфизма позволило создать структурированный и легко модифицируемый код.

Разработка нейронной сети включала в себя инициализацию, обучение и использование для распознавания текста. Выбор различных конфигураций нейронных сетей и возможность загрузки весов из файла предоставляют пользователю гибкость и удобство в использовании программы.

Применение библиотеки Qt для создания графического пользовательского интерфейса добавило проекту эстетичность и удобство взаимодействия с пользователем. Возможность выбора конфигурации нейронной сети, загрузка весов и файла с текстом, а также наглядное отображение процесса обучения и распознавания текста делают программу удобной и функциональной.

Завершая работу, можно отметить, что разработанный программный продукт успешно решает поставленную задачу распознавания текста. Он предоставляет пользователям эффективный и удобный инструмент, который может быть использован в различных областях, таких как распознавание рукописного текста или автоматизированное чтение изображений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Vaughan, J. Backpropagation algorithm / J.Vaughan [Электронный ресурс].–Режим доступа: https://www.techtarget.com/searchenterpriseai/definition/backpropagation-algorithm#:~:text=Backpropagation.–Дата доступа: 10.10.2023.

[2] Nicholson, C. A Beginner's Guide to Neural Networks and Deep Learning / C.Nicholson [Электронный ресурс].–Режим доступа:https://wiki.pathmind.com/neural-network.–Дата доступа: 10.10.2023.

[3] Pradeep,J. DIAGONAL BASED FEATURE EXTRACTION FOR HANDWRITTEN ALPHABETS RECOGNITION SYSTEM USING NEURAL NETWORK /J.Pradeep. International Journal of Computer Science & Information Technology (IJCSIT), 2011.

[4] Руководство по языку программирования C++ [Электронный ресурс].–Режим доступа: https://metanit.com/cpp/tutorial/.–Дата доступа: 10.10.2023.

[5] Князев, А.А. Qt для начинающих / А.А.Князев [Электронный ресурс].–Режим доступа: http://knzsoft.ru/qt-bgr-ls1/.–Дата доступа: 25.10.2023.

ПРИЛОЖЕНИЕ А

(обязательное)

Диаграмма классов

ПРИЛОЖЕНИЕ Б

(обязательное)

Схема метода `void Network::NetworkInitialization(int conf)`

ПРИЛОЖЕНИЕ В

(обязательное)

Схема метода `void Network::BackPropogation(double predict)`

ПРИЛОЖЕНИЕ Г

(обязательное)

Код программы

ПРИЛОЖЕНИЕ Д
(обязательное)
Ведомость документов