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
«Перцептрон. Распознавание слов»

Студент Д.О. Можейко

Руководитель Е. В. Богдан

МИНСК 2023

Учреждение образования
«Белорусский государственный университет информатики
и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ
Заведующий кафедрой

_____
(подпись)
_____2023 г.

ЗАДАНИЕ
по курсовому проектированию

Студенту *Можейко Дмитрию Олеговичу*

Тема проекта *«Перцептрон. Распознавание слов»*

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

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

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.
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.
   5.2. Разработка алгоритмов.
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 прямого обхода по нейросети

3. Схема алгоритма обновления весов

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 11.12.22 – 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 нейронные сети стали одной из наиболее важных и перспективных областей исследований в области машинного обучения и искусственного интеллекта. Способность моделировать сложные нелинейные взаимосвязи и обрабатывать большие объемы данных делает нейронные сети не только мощным инструментом для анализа данных, но и ключевым элементом во многих сферах науки, техники и бизнеса. Одним из наиболее важных прототипов нейронной сети является перцептрон, который представляет собой однослойную модель, способную обучаться с учителем и выполнять простые классификационные задачи.

Идея перцептрона была предложена Фрэнком Розенблаттом в 1957 году и относится к пионерским работам в области искусственных нейронных сетей. Он представляет собой модель, имитирующую работу одного нейрона в человеческом мозге. Перцептрон состоит из входов, взвешенной суммы входных сигналов и функции активации, которая определяет выходной сигнал. В своей простейшей форме перцептрон обладает способностью решать только линейно разделяемые задачи, однако современные модификации, такие как многослойные перцептроны, способны решать более сложные задачи, включая распознавание образов и речи, классификацию данных и прогнозирование [4].

Целью данной курсовой работы является глубокое исследование структуры и функционирования перцептрона, его математических основ и возможностей, а также анализ применения данной модели в различных областях, включая компьютерное зрение, обработку естественного языка, управление и другие приложения. В рамках работы будет проведен обзор современных методов обучения перцептронов, включая алгоритмы градиентного спуска, обратного распространения ошибки. Кроме того, будет произведен анализ ограничений и проблем, связанных с применением перцептрона, а также рассмотрены возможные пути их преодоления. В целом, данная работа направлена на расширение понимания принципов работы перцептрона и его вклада в развитие области нейронных сетей и искусственного интеллекта в целом.

1. ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++, овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня C++.Разработать нейронную сеть типа перцептрон по распознаванию латинских слов с использованием среды разработки Qt.

Преимущество Qt:

- Простота использования: Qt имеет хорошо документированную структуру, прост в освоении и имеет простую, однородную структуру.
- Разработка GUI: Qt предоставляет широкий спектр компонентов для создания графических пользовательских интерфейсов, включая Qt Quick, который позволяет быстро и легко создавать интерфейс с использованием специального языка под названием QML [5].

В целом, использование Qt в C++ дает много преимуществ для разработчиков, которые хотят создавать кроссплатформенные приложения с графическим пользовательским интерфейсом. Это мощный и гибкий инструмент, который можно использовать в широком спектре приложений, от настольных и мобильных приложений до специализированного оборудования и встроенных систем.

2. ОБЗОР ЛИТЕРАТУРЫ

2.1. Обзор методов и алгоритмов решения поставленной задачи

2.1.1. Многослойный перцептрон

Многослойный перцептрон (MLP) представляет собой разновидность искусственной нейронной сети, которая состоит из нескольких слоев нейронов, включая входной слой, скрытые слои и выходной слой. Он является расширением простого перцептрона и способен решать сложные задачи, которые не могут быть решены одним линейным классификатором.

Структура многослойного перцептрона:

1. Входной слой:

   Входной слой представляет собой первый слой нейронов, который получает входные данные или признаки. Каждый нейрон в этом слое соответствует одному измерению входных данных. Входные данные могут быть представлены в виде вектора, и каждый нейрон входного слоя принимает соответствующее значение из этого вектора.

2. Скрытые слои:

   Скрытые слои находятся между входным и выходным слоями и служат для извлечения признаков из входных данных. Каждый нейрон в скрытых слоях использует взвешенные суммы значений из предыдущего слоя с добавлением смещения (bias) и применяет к ним функцию активации, такую как ReLU или сигмоида. Эти слои позволяют перцептрону изучать сложные нелинейные зависимости в данных.

3. Выходной слой:

   Выходной слой представляет собой последний слой нейронов, который выдает конечный результат. Каждый нейрон в выходном слое связан с предсказанными классами или значением целевой переменной. В зависимости от задачи, функция активации выходного слоя может различаться. Например, для задач классификации часто используется функция Softmax, а для регрессии может использоваться линейная функция или сигмоида.

2.1.2. Метод прямого распространения

Процесс прямого распространения (Forward Feed) является первым шагом в работе нейронной сети и состоит из передачи входных данных через

слои сети для генерации выходных предсказаний [2]. Во время прямого прохода каждый слой принимает входные данные, выполняет операции взвешенной суммы входов и активации, и передает результаты следующему слою.

Алгоритм ForwardFeed:

- Принятие входных данных:

  Входные данные представляют собой сырые значения или признаки, которые подаются на входной слой нейронной сети. Эти данные могут быть представлены в виде вектора или матрицы, в зависимости от структуры и конфигурации сети. Например, для изображений каждый пиксель может быть представлен как отдельный элемент входного вектора.

- Вычисление взвешенной суммы и активации:

  Для каждого нейрона в скрытых слоях вычисляется взвешенная сумма входов путем умножения весов на значения входных данных и последующего добавления смещения (bias). Полученная взвешенная сумма затем передается через функцию активации, такую как ReLU, сигмоид или tanh, для введения нелинейности в выходной сигнал нейрона. Взвешенная сумма вычисляется по формуле:

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

  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)

- Передача данных следующему слою:

  Выходные значения каждого нейрона слоя передаются как входные данные следующему слою, который выполняет аналогичные операции взвешенной суммы и активации. Этот процесс продолжается до тех пор, пока данные не достигнут выходного слоя, где генерируются окончательные выходные предсказания.

- Формирование окончательных предсказаний: В выходном слое выполняется окончательная операция взвешенной суммы и активации, приводящая к формированию окончательных выходных предсказаний или результатов. В зависимости от типа задачи, выходной слой может использовать различные функции активации, такие как softmax для задач многоклассовой классификации или линейную функцию для регрессии.

Процесс прямого распространения является основой для работы нейронных сетей и позволяет сети генерировать предсказания на основе входных дан-

ных, что делает его ключевым шагом в обработке информации нейронной сетью.

2.1.3. Функция активации

Функция активации modRelu, или модифицированная ReLU, является вариацией функции активации ReLU (Rectified Linear Unit). Она представляет собой модифицированную версию ReLU, которая возвращает значение нуля для отрицательных входов и модуля положительных входов. Это позволяет сгладить резкие переходы и избежать некоторых проблем, связанных с исчезающим градиентом, которые могут возникнуть при использовании ReLU. Функция активации modRelu часто используется в нейронных сетях, включая перцептроны, для обеспечения более стабильного и устойчивого обучения. Функция modRelu выглядит следующим образом:

$$f(x) = \max(x, 0.01x)$$

2.1.4. Алгоритм обратного распространения ошибки (Backpropagation) для обновления весов.

Алгоритм обратного распространения ошибки (Backpropagation) является основным методом обучения нейронных сетей, включая перцептроны [1]. Он позволяет эффективно вычислять градиент функции потерь по отношению к весам нейронов в сети. Этот градиент затем используется в методе градиентного спуска для обновления весов, чтобы минимизировать ошибку. Нахождение ошибок и обновление весов происходит по формулам:

$$dE/dw = dE/dz * dz/dw$$

где dE/dw - градиент ошибки по весу, dE/dz - градиент ошибки по входу z нейрона, dz/dw - производная выхода нейрона по весу.

$$w_{new} = w_{old} - \eta * dE/dw$$

где w_{new} - новое значение веса, w_{old} - предыдущее значение веса, η - скорость обучения (learning rate), коэффициент, определяющий величину шага обновления весов, dE/dw - градиент ошибки по весу.

В контексте задачи распознавания слов алгоритм Backpropagation обеспечивает эффективное и точное обновление весов перцептрона, что способствует улучшению процесса распознавания и улучшению общей производительности модели.

2.1.5. Dropout

Dropout является техникой регуляризации, которая случайным образом удаляет нейроны во время обучения с определенной вероятностью. Это помогает избежать переобучения модели, улучшая ее обобщающую способность на новых данных. В процессе обучения случайно выбранные нейроны исключаются из сети на каждом шаге, что приводит к более устойчивой и надежной модели. Результатом является уменьшение взаимосвязи между нейронами, что позволяет более надежно распознавать слова на новых данных.

2.1.6. Подача картинок в случайном порядке

Подача изображений в случайном порядке помогает улучшить обобщающую способность нейронной сети, предотвращая ее привязку к определенным последовательностям данных. Это обеспечивает более разнообразное обучение, позволяя сети эффективнее обобщать образцы и устойчиво реагировать на новые данные, что важно для успешного распознавания слов в различных контекстах.

2.1.7. Предобработка изображений слов

:

Для распознавания слов изображения разрезаются на отдельные буквы, масштабируются до нужного размера и центрируются, чтобы нейронная сеть могла корректно распознавать буквы в любом месте изображения. Это позволяет повысить точность распознавания и улучшить общую производительность нейронной сети при распознавании слов.

2.1.8. Выборка для обучения

Датасет "NIST Special Database 19" представляет собой набор данных, созданный Национальным институтом стандартов и технологий (NIST) США. Он содержит образцы рукописных символов и текста, включая различные алфавиты, цифры и специальные символы. Этот датасет широко используется для обучения и тестирования систем распознавания текста, оптического распознавания символов (OCR) и других задач, связанных с распознаванием рукописного текста.

Описание использованного датасета "NIST Special Database 19":

- Содержание датасета: Датасет "NIST Special Database 19" содержит большой набор образцов рукописных символов, включая различные алфавиты, такие как латинский, кириллический, греческий и другие, а также цифры и специальные символы. Эти образцы представлены в виде изображений высокого качества, что делает их идеальными для обучения и тестирования систем распознавания текста.

- Разнообразие данных: Датасет обладает разнообразием данных, содержащих различные стили и вариации рукописного текста. Это включает разные шрифты, размеры и стили написания, что позволяет создавать более реалистичные модели распознавания текста и повышает их способность обобщения на различные стили написания.

- Использование в исследованиях: Датасет "NIST Special Database 19" широко используется в исследованиях и разработке систем распознавания текста, включая задачи оптического распознавания символов (OCR), распознавания рукописного текста и других задач, связанных с обработкой естественного языка. Этот датасет является одним из наиболее популярных и широко используемых наборов данных для обучения моделей машинного обучения для распознавания текста.

- Качество данных: Изображения в датасете "NIST Special Database 19" характеризуются высоким разрешением и четкостью, что обеспечивает качественные данные для обучения моделей распознавания текста. Высокое качество изображений способствует эффективному обучению моделей нейронных сетей и других алгоритмов машинного обучения для успешного распознавания рукописного текста.

Датасет "NIST Special Database 19" является важным ресурсом для исследований в области распознавания текста и широко используется для обучения моделей машинного обучения и создания систем распознавания рукописного текста высокой производительности.

3. ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

3.1. Структура входных и выходных данных

1. Структура входных данных:

   Для обучения нейронной сети и распознавания слов изображения со словами размером 128 на 128 пикселей используется в качестве входных данных. Каждое изображение представляет собой синтезированное слово, полученное путем размещения букв в определенном порядке. Входные данные предварительно обрабатываются для подготовки к последующему обучению и распознаванию.

2. Предварительная обработка входных данных

   - Разбиение слов на буквы:

     Синтезированное изображение со словом разбивается на отдельные буквы с использованием алгоритма сегментации изображений. Этот шаг позволяет разделить слово на отдельные компоненты для дальнейшего распознавания.

   - Масштабирование изображений букв:

     Каждая отдельная буква масштабируется до размера 128 на 128 пикселей для обеспечения единообразного входа в нейронную сеть. Этот процесс позволяет подготовить данные для успешного распознавания при любых условиях входных изображений.

3. Представление входных данных:

   Каждая буква представляется в виде массива пикселей размером 128 на 128. Пиксели могут быть представлены в виде оттенков серого или цветовой палитры в зависимости от конкретной реализации.

4. Структура выходных данных

   Выходные данные представляют собой прогнозируемые метки или классы, соответствующие распознанным буквам изображений. Каждая буква на входном изображении ассоциируется с определенным выходным нейроном, представляющим конкретную букву. Общее количество выходных нейронов определяется количеством возможных классов (в данном случае - букв латинского алфавита).

5. Подготовка выходных данных

   - Кодирование меток классов:

     Каждая буква кодируется с использованием соответствующего вектора или индекса, соответствующего определенному классу.

Это позволяет представить выходные данные в форме, пригодной для обучения модели.

- Применение one-hot encoding:

  Для удобства обработки данных и обучения модели используется метод one-hot encoding, который преобразует числовые метки классов в векторы, содержащие 1 на позиции, соответствующей конкретному классу, и 0 на остальных позициях. Это обеспечивает удобство в работе с категориальными данными при обучении нейронной сети.

Структура входных и выходных данных играет ключевую роль в успешном обучении нейронных сетей для распознавания слов на изображениях. Использование датасета "NIST Special Database 19" размером 128 на 128 пикселей позволяет обеспечить достаточное разрешение и детализацию для эффективного обучения модели.

Процесс предварительной обработки данных, такой как разбиение изображений на отдельные буквы, их масштабирование и представление в виде массивов пикселей, существенно влияет на результаты распознавания. Кроме того, правильное кодирование меток классов и применение метода one-hot encoding обеспечивают удобство в обработке и интерпретации выходных данных.

Эффективная структура входных и выходных данных, а также правильная предварительная обработка изображений, играют важную роль в создании надежной модели распознавания слов с помощью нейронной сети. Дальнейшее исследование и углубление в методы обработки изображений и кодирования данных могут значительно повысить точность и эффективность распознавания слов в различных контекстах и условиях.

3.2. Разработка диаграммы классов

Диаграмма классов — это вид диаграммы, используемый в языке моделирования UML (Unified Modeling Language), который предназначен для визуализации структуры классов и отношений между ними в объектно-ориентированных системах программирования. Диаграмма классов представляет собой графическое представление структуры системы, описывающее классы, их атрибуты, методы и отношения между классами.

Основные элементы, которые могут присутствовать на диаграмме классов:

1. Классы: Прямоугольник с разделенной на три части строкой. В верхней части указывается название класса, в середине — атрибуты, а в нижней — методы.

2. Отношения между классами:

    - Ассоциация: Линия, соединяющая два класса, представляющая отношение между объектами этих классов.
    - Наследование (или обобщение): Стрелка, указывающая на базовый класс. Показывает, что один класс наследует от другого.
    - Реализация (или зависимость): Штрихованная линия со стрелкой. Показывает, что один класс реализует интерфейс или зависит от другого класса.

3. Интерфейсы: Прямоугольник, разделенный на три части, подобно классу, но с ключевым словом "interface". Показывает интерфейс, который реализуют классы.

4. Агрегация и композиция: Показывают отношения между целым и его частями. Агрегация обозначается стрелкой с пустым алмазом на конце, а композиция — стрелкой с закрашенным алмазом.

Диаграмма классов помогает разработчикам и архитекторам программного обеспечения лучше понимать структуру системы, визуализировать отношения между классами и обеспечивает основу для дальнейшей разработки кода.

Диаграмма классов для приложения «Перцептрон» приведена в приложении А.

3.3. Описание классов

3.3.1. *ActivateF*

ActivateF - это класс, предназначенный для работы с функциями активации, используемыми в нейронных сетях. Он содержит различные методы для вычисления и применения функций активации:

- *ModR* - это метод, который принимает массив значений нейронов *NeuronValue* и их количество *NeuronsOnLayer*. Он применяет функцию активации ModReLU (Modified Rectified Linear Unit) к каждому значению нейрона в массиве, обеспечивая нелинейность в выходных значениях. Функция ModReLU очень похожа на ReLU, но с добавленной дополнительной функциональностью или модификациями.

- *ModRDer* - это перегруженный метод, который также принимает массив значений нейронов *NeuronValue* и их количество *NeuronsOnLayer*. Он вычисляет производную функции активации *ModR* для каждого значения нейрона и возвращает результат. Этот метод полезен при обратном распространении ошибки (backpropagation) в процессе обучения нейронной сети.

- *ModRDer* (второй по счету) - это перегруженная версия метода, которая принимает одно значение нейрона *NeuronValue*. Она вычисляет производную функции активации *ModR* для конкретного значения нейрона и возвращает результат. Этот метод также используется при обратном распространении ошибки (backpropagation) для обновления весов в процессе обучения.

3.3.2. *Matrix*

Matrix - это класс, который представляет собой матрицу, используемую в операциях матричного умножения и транспонирования, необходимых для обработки данных в нейронных сетях. Он содержит переменные для хранения значений матрицы, а также методы для инициализации, генерации случайных значений и выполнения различных операций над матрицами:

- *matrix* - это двумерный массив, который используется для хранения значений элементов матрицы.

- *row* и *column* - это переменные, которые указывают количество строк и столбцов в матрице соответственно.

- *Multipl* - это метод, который выполняет умножение матрицы на вектор neurons или другую матрицу *Mult_Matrix* в зависимости от переданных аргументов. Этот используется для вычислений при прямом и обратном распространении ошибки в нейронных сетях.

- *MultiTrans* - это метод, который выполняет операцию умножения транспонированной матрицы на вектор neurons или другую матрицу *Mult_Matrix*. Этот метод используется в процессе обучения нейронных сетей.

- *Sum* - это метод, который выполняет операцию суммирования матрицы *Mult_Matrix* с вектором *bias* в соответствии с переданными аргументами. Он также может использоваться при выполнении различных вычислений в нейронных сетях.

- *GetElement* - это метод, которые позволяют получить доступ к элементам матрицы по указанным индексам.

- *Init* - это метод, который инициализирует матрицу с заданным количеством скрытых и входных нейронов.

- *Rand* - это метод, который генерирует случайные значения для элементов матрицы. Нужен для инициализации случайными значениями

3.3.3. *Network*

Network - представляет нейронную сеть, содержащую основные методы для прямого и обратного распространения ошибки, обновления весов, инициализации сети, ввода данных, поиска максимального индекса, а также сохранения и чтения весов сети:

- *layers* - это переменная, которая хранит количество слоев в нейронной сети.

- *neurons_on_layer* - это массив, который хранит количество нейронов на каждом слое.

- *function* - объект класса *ActivateFunction*, который предоставляет функции активации для использования в сети.

- *weight* - это массив матриц, который хранит веса между нейронами на различных слоях.

- *neurons* - это двумерный массив, который хранит значения нейронов на каждом слое.

- *error_of_neurons* - это двумерный массив, который хранит значения ошибок нейронов на каждом слое.

- *bias* - это массив, который хранит смещения (biases) для каждого нейрона.

- *bias_weight* - это двумерный массив, который хранит веса для смещений (biases) каждого нейрона.
- *ForwardFeed* - метод для выполнения прямого распространения, где входные данные передаются через сеть для генерации выходных предсказаний.
- *BackPropogation* - метод для обратного распространения ошибки, используемый для обновления весов в соответствии с ошибками на выходном слое.
- *WeightsUpdater* - метод для обновления весов сети в процессе обучения с использованием метода градиентного спуска.
- *PrintValues* - метод для печати значений на определенном слое L, что может быть полезным для отладки и анализа процесса обучения.
- *NetworkInit* - метод для инициализации нейронной сети, включающий в себя начальное определение числа слоев, числа нейронов на каждом слое и других параметров сети.
- *DataInput* - метод для ввода данных (картинок) в сеть, готовящий данные для прямого распространения.
- *SearchMaxIndex* - метод для поиска максимального индекса, определяет предсказанные классы или категории.
- *SaveWeight* - метод для сохранения весов сети, позволяющий сохранить состояние весов для дальнейшего использования или анализа.
- *ReadWeight* - метод для чтения сохраненных весов сети, позволяющий загрузить предыдущее состояние весов для продолжения обучения или использования.

4. РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1. Разработка схем алгоритмов

Метод double Network::ForwardFeed() предназначен для выполнения прямого прохода (forward pass) или процесса прямого распространения (forward propagation). Этот процесс важен в контексте обучения нейронных сетей.

Прямой проход представляет собой передачу входных данных через сеть от входного слоя к выходному слою. В процессе прямого прохода каждый нейрон в каждом слое выполняет следующие шаги:

1. Умножение на веса: Входные данные, которые поступают на нейрон, умножаются на соответствующие веса.

2. Суммирование: Результат умножения суммируется с другими входами и смещением (bias).

3. Применение функции активации: Результат суммирования передается через функцию активации, которая вводит нелинейность в выход нейрона.

4. Передача к следующему слою: Полученные значения передаются в качестве входных данных следующему слою, и процесс повторяется до достижения выходного слоя.

В итоге, метод ForwardFeed выполняет прямой проход для входных данных через все слои нейронной сети, схема метода ForwardFeed показана в приложении Б.

Метод void Network::WeightsUpdater(double learning_rate) предназначен для обновления весовых коэффициентов (весов) и смещений (bias) в процессе обучения. Этот процесс известен как обратное распространение ошибки (backpropagation), и он является ключевой частью обучения нейронных сетей.

Функции WeightsUpdater:

1. Обновление весов:

    - Для каждого слоя (кроме последнего) и каждого нейрона в следующем слое выполняется двойной вложенный цикл. Внешний цикл проходится по слоям, а внутренний — по нейронам в следующем слое.
    - Для каждого веса между текущим нейроном и нейроном в следующем слое выполняется следующая операция: текущий вес увеличивается на произведение входа нейрона и ошибки в нейроне следующего слоя, умноженной на скорость обучения (learning_rate).

2. Обновление смещений (bias):

    - Аналогично, для каждого слоя (кроме последнего) и каждого нейрона в следующем слое выполняется цикл.
    - Каждое смещение в слое увеличивается на ошибку в соответствующем нейроне следующего слоя, умноженную на скорость обучения.

Эти операции направлены на минимизацию ошибки между предсказанными значениями нейронной сети и фактическими значениями в процессе обучения. Обновление весов и смещений происходит на каждом шаге обучения и осуществляется с использованием градиентного спуска. схема метода WeightsUpdater показана в приложении В.

4.2. Разработка алгоритмов

4.2.1. Алгоритм прямого обхода по нейросети

1. Инициализация:

    - Начинаем с инициализации переменной i равной 1, так как прямой проход начинается со второго слоя (первый слой — входной).
    - Определяем переменную neuronsonlayer как количество нейронов в текущем слое (neurons_on_layer[i]).

2. Прямой проход по слоям:

    - Начинаем цикл, в котором проходим по слоям сети, начиная со второго и заканчивая последним (layers).
    - Внутри цикла:
        - Вычисляем количество нейронов в текущем слое (neuronsonlayer).
        - Выполняем умножение матрицы весов текущего слоя на выходные данные (активации) предыдущего слоя. Результат сохраняется в neurons[i].
        - Суммируем смещения (bias) текущего слоя с выходами предыдущего слоя. Результат также сохраняется в neurons[i].
        - Применяем функцию активации (*ModR*) к выходам текущего слоя, модифицируя значения в neurons[i].

3. Поиск предсказанного значения:

    - После завершения цикла по слоям, вызываем функцию *SearchMaxIndex*, которая находит индекс нейрона с максимальным выходом. Результат сохраняется в predict_letter.

4. Возвращение результата:
    - Возвращаем предсказанное значение (predict_letter).

4.2.2. Алгоритм обновления весов

1. Начинаем цикл по всем слоям сети (кроме последнего).
2. Внутри первого цикла, проходим по всем нейронам в следующем слое.
3. Внутри второго цикла, запускаем третий цикл, проходя по всем нейронам в текущем слое.
4. Внутри третьего цикла, обновляем веса между текущим и следующим нейронами на основе градиента и скорости обучения.
5. После завершения первого цикла, начинаем второй цикл по слоям (кроме последнего).
6. Внутри второго цикла, запускаем третий цикл по нейронам в следующем слое.
7. Внутри третьего цикла, обновляем смещения (bias) для каждого нейрона следующего слоя на основе градиента и скорости обучения.
8. Завершаем метод WeightsUpdater.

5. РЕЗУЛЬТАТ РАБОТЫ

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была разработана и реализована многослойная нейронная сеть на основе перцептрона с целью распознавания слов. Этот проект представляет собой важный шаг в области обработки естественного языка, позволяя автоматически распознавать и классифицировать слова на основе обученных весов и архитектуры сети.

Основными компонентами проекта является многослойная архитектура перцептрона, которая была обучена на соответствующих данных для достижения высокой точности распознавания. Обученные веса и параметры сети позволяют создавать предсказания с высокой степенью уверенности, что является важным элементом в областях, таких как обработка речи и распознавание слов.

Кроме того, в рамках проекта был разработан интуитивно понятный пользовательский интерфейс с использованием фреймворка Qt. Этот интерфейс предоставляет удобные средства взаимодействия с нейронной сетью, что существенно улучшает пользовательский опыт и упрощает процесс взаимодействия с разработанным решением.

Одним из ключевых результатов работы стало успешное интегрирование нейронной сети с пользовательским интерфейсом, что создает простую и эффективную систему для распознавания слов. Данный проект может иметь широкий спектр применений в областях автоматического распознавания речи, систем обработки данных и других сферах, где распознавание слов играет важную роль.

Обобщая, выполненная курсовая работа представляет собой ценный вклад в область разработки нейронных сетей и пользовательских интерфейсов, демонстрируя потенциал применения подобных технологий для повышения эффективности и удобства в повседневных задачах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] How the backpropagation algorithm works: [Электронный ресурс]. Режим доступа: http://neuralnetworksanddeeplearning.com/chap2.html. (Дата обращения: 25.10.2023).

[2] Basic Introduction to Feed-Forward Network in Deep Learning: [Электронный ресурс]. Режим доступа: https://www.analyticsvidhya.com/blog/2022/03/basic-introduction-to-feed-forward-network-in-deep-learning/. (Дата обращения: 26.10.2023).

[3] Object Oriented Programming in C++: [Электронный ресурс]. Режим доступа: https://www.geeksforgeeks.org/object-oriented-programming-in-cpp/(Дата обращения: 20.10.2023).

[4] Что такое нейронная сеть?: [Электронный ресурс]. Режим доступа: https://aws.amazon.com/ru/what-is/neural-network/. (Дата обращения: 21.10.2023).

[5] Qt for Beginners: [Электронный ресурс]. Режим доступа: https://wiki.qt.io/Qt_for_Beginners. (Дата обращения: 21.10.2023).

ПРИЛОЖЕНИЕ А

Диаграмма классов

ПРИЛОЖЕНИЕ Б

Схема метода ForwardFeed

ПРИЛОЖЕНИЕ В

Схема метода WeightUpdater

ПРИЛОЖЕНИЕ Г

Код программы