Автономная некоммерческая общеобразовательная организация

"Физтех-лицей" им. П. Л. Капицы

(АНОО «Физтех-лицей» им. П.Л. Капицы)

Международная научно-практическая конференция

**«СТАРТ В ИННОВАЦИИ»**

**Исследовательская работа**

**«КОНСТРУКТОР НЕЙРОННЫХ СЕТЕЙ»**

**Авторы работы:**

Супенко Иван

Чернышев Артем

Боймель Максим

Сапронова Маргарита

Подлягин Данил

10 класс «К»

**Научный руководитель:**

Сальников Владислав Владимирович

Московская область, г. Долгопрудный

2024 г.

**Разработка программного продукта**

1. **Актуальность, описание целевой аудитории (или наличие заказчика)**

Нейронная сеть — это компьютерная система, созданная по образу и подобию человеческого мозга. Она обучается на загруженных в нее текстовых базах данных и принимает обоснованные решения. Она состоит из взаимосвязанных узлов или «нейронов», которые сотрудничают для обработки информации и решения разнообразных проблем. Конструктор нейронных сетей — программа для создания нейронных сетей.

Конструктор предоставляет возможности:

* **Визуализация прогресса обучения.** Одной из ключевых функций нашего конструктора является визуализация прогресса обучения. На каждой эпохе вы сможете следить за динамикой обучения, а главное - рассматривать данные каждого нейрона и его связей. Это позволяет более детально анализировать, какие участки сети требуют дополнительной настройки, делая процесс обучения более прозрачным и эффективным.
* **Удобный конструктор нейронных сетей.** Наш конструктор обеспечивает интуитивно понятное создание нейронных сетей. Вы сможете легко выбирать типы слоев, определять количество нейронов, настраивать функции активации и многое другое. Графический интерфейс позволяет вам легко перемещаться между слоями и связями, делая процесс конструирования нейронной сети увлекательным и эффективным.
* **Текстовые уроки для понимания нейронных сетей.** Чтобы сделать процесс обучения еще более доступным, мы предоставляем текстовые уроки. Они детально разъясняют основы нейронных сетей, ключевые концепции и практические аспекты. Текстовые уроки встроены в конструктор, что делает их доступными и удобными в использовании.

Последнее время нейронные сети стали набирать популярность. Тем не менее, создание и проектирование нейросети осложнено недостатком информации, либо же слишком сложным её преподнесением. Упрощение процесса создания нейронных поспособствовало бы их популяризации. Благодаря понятной визуальной составляющей в дальнейшем пользователям нашей программы проще начать обучать свои нейросети и позже использовать их в своих проектах.

1. **Анализ аналогов на рынке.**

Сам процесс создания нейронной сети зависит от множества факторов, включая их структуру, количество слоев и нейронов, объем и тип данных для обучения. Разработка и настройка эффективной нейронной сети подразумевает определенные сложности, требующие глубоких знаний в области искусственного интеллекта, машинного обучения и статистики. Иными словами, создание нейронных сетей представляет собой комплексный и трудоемкий процесс, который подразумевает длительное обучение и углубленные знания.

У существующих и доступных на данных момент конструкторов нейронных сетей куча недостатков.

* Функциональные и удобные конструкторы всегда платные.
* Неподготовленный пользователь может запутаться в связи с неудобностью интерфейса по сравнению со сложностью самого процесса создания.
* Отсутствие встроенной возможности обучения. Необходимую информацию приходится искать в других источниках, что не всегда легко.
* Ограниченный функционал, что может не удовлетворять потребностям пользователя и не давать полной картины процесса создания и обучения нейронной сети.

Сравнительный анализ существующих конструкторов нейронных сетей:

* **NeuroNet** – имеет ограниченный функционал, в котором возможно собрать только полносвязную сеть, а также неудобный интерфейс.
* **Tensorflow playground** – ограниченные вариации конструкций моделей, а также невозможность обучать модель на своем наборе данных
* **Exponenta Deep Learning Toolbox** – невозможность наблюдать за прогрессом обучения.

1. **Описание функционала конечного продукта**

Конечным результатом проекта является программа – конструктор нейронных сетей, позволяющая визуально по частям собрать модель нейросети, а после посмотреть прогресс ее обучения и ее состояние на каждой итерации обучения. Также в приложении можно изучить текстовые уроки о работе нейросетей.

В приложении есть четыре вкладки:

* **Главная.** На этой вкладке есть возможность выбора инициализации проекта. Например, пользователь может создать новый проект, открыть проект из файла, либо же открыть один из недавних проектов, представленных в меню.
* **Редактор**. Здесь пользователь может собрать модель по слоям, а также настроить каждый слой отдельно, выбрав подходящие настройки для каждого из них. Также на этой вкладке можно выбрать набор данных для обучения и разметить, как модель должна его обрабатывать.
* **Обучение**. На этой вкладке пользователь может посмотреть прогресс обучения, изучив состояние модели на каждой итерации обучения. Также на этой вкладке представлены графики, описывающие модель (например, графики loss и accuracy, описывающие функцию ошибки и долю правильных ответов соответственно)
* **Лекторий.** Здесь пользователь может изучить текстовые уроки про нейросети, их виды, способы обучения и т. д.м Уроки поделены на разделы для их структуризации и упрощения навигации по ним

1. **Описание алгоритмов и структур данных, используемых при создании программного продукта, и их сложности.**

В нашем проекте мы используем такие алгоритмы и структуры данных, как:

* Обучение нейросети методом обратного распространения ошибки и минимизация функции ошибки стохастическим градиентным спуском.
* Различные функции активации нейронов (см. Приложение 1):
  + **Sigmoid** – одна из первых функций активации. Из плюсов этой функции активации можно отметить гладкость градиента, а также ограниченность значений отрезком [0, 1]. Из минусов – близкое к нулю значение производной при больших по модулю значениях аргумента.
  + **ReLU** – сравнительно новая функция активации (представлена в 2015 году). Из минусов можно отметить большое значение функции при больших значениях аргумента, что во время обучения методом обратного распространения ошибки может привести к взрыву коэффициентов.
  + **Tanh** – похожа на сигмоиду, но в отличие от нее может принимать отрицательные значения.
  + **Softmax** – функция активации, которую обычно используют для того, чтобы при задаче классификации оценить вероятность того, что данные принадлежат какому-то классу.  
    
* Различные типы слоев:
  + **Input layer** – входной слой, в котором значения нейронов выставляются напрямую из входных данных.
  + **Fully-connected layer** – полносвязный слой, в котором значения нейронов задаются суммой смещения нейрона и произведений всех нейронов прошлого слоя на соответствующие веса связей.
  + **Convolutional layer** – сверточный слой, совершающий операцию свертки данных прошлого слоя с ядром параметров, и добавляющий к результату значение смещения нейрона.
  + **Pooling layer** – слой пулинга, уменьшающий размер данных предыдущего слоя, при этом почти не изменяю смысл этих данных.
* Различные методы нормализации для борьбы с затуханием градиента:
  + **BatchNorm** – нормализация данных в каждом мини-батче таким образом, чтобы для каждого канала среднее значение было равно нулю, а дисперсия единице. Главный минус такого подхода заключается в том, что каждый мини-батч нормируется независимо от остальных, а также непонятно, как нормализовать данные при использовании нейросети в будущем, потому что на вход подается лишь один экземпляр входных данных, и непонятно, как его нормализовывать.
  + **LayerNorm** – это метод нормализации, который нормализует входные данные по каждому слою, а не по каждому батчу. Он нормализует среднее значение и дисперсию для каждого слоя.
  + **MinMax** – нормализация данных, путем приведения их к отрезку [min, max]. Из недостатков – часто мы не можем определить, в каких границах могут лежать данные до нормализации, поэтому привести их к отрезку [min, max] может быть проблематично.

1. **Описание использованных технологий и обоснование их выбора (например: библиотеки написания интерфейса, выбор конкретного языка программирования и платформы и т.д.)**

Наш конструктор разрабатывается на языке C++, потому что он довольно быстрый, а для нашего проекта это очень важно, потому что, в отличие от большинства других проектов, он должен обучать нейросеть параллельно с работой графической оболочки. Также наша команда изучает C++ как основной язык программирования, поэтому нам будет проще и интереснее написать проект на нем.

Для реализации графического интерфейса мы будем использовать библиотеку QT6, потому что она популярна, довольно проста в использовании, но при этом имеет много модулей и возможностей их кастомизации.

Для создания и обучение моделей нейросетей мы не будем использовать какие-либо специфические библиотеки, кроме STL, потому что нам нужно полностью контролировать состояние всех компонентов нейросети на каждой итерации обучения, что не позволяют сделать такие библиотеки, а также, потому что нам самим интересно изучить работу нейросетей изнутри.

1. **Ссылки на источники**
2. С++ documentation [Электронный ресурс]. URL: <https://ru.cppreference.com/w/>
3. Конструктор Deep learning toolbox [Электронный ресурс]. URL: <https://exponenta.ru/neural-network-toolbox>
4. Конструктор NeuroNet [Электронный ресурс]. URL: <https://habr.com/ru/articles/351922>
5. Конструктор Tensorflow Playground [Электронный ресурс]. URL: <https://playground.tensorflow.org>
6. QT documentation [Электронный ресурс]. URL: <https://wiki.qt.io/Main>
7. С. Николенко, А. Кадурин, Е. Архангельская Глубокое обучение // Погружение в мир нейронных сетей, 2020
8. Activation function [Электронный ресурс]. URL: <https://en.wikipedia.org/wiki/Activation_function>

**Приложение 1**

* **Sigmoid**  
  
* **ReLU**  
    
  
* **Tanh**  
    
  
* **Softmax**  
  

**Приложение 2**

| Название | Создатель | Преимущества | Недостатки |
| --- | --- | --- | --- |
| NeuroNet | Артём Музагафаров  habr: [@smile\_artem](https://habr.com/ru/users/smile_artem/) | * Наглядное представление состояния нейросети в текущий момент | * Неудобный интерфейс * Ограниченность вариаций конструкций моделей |
| Tensorflow Playground | Google Brain Team <https://www.tensorflow.org/> | * Открытый исходный код * Удобный интерфейс | * Невозможность использования своего набора данных для обучения * Ограниченность вариаций конструкций моделей |
| Deep Learning Toolbox | The MathWorks, Inc.  <https://exponenta.ru/> | * Удобный интерфейс | * Невозможность наблюдать за прогрессом обучения |