МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

РАБОТА ПРОВЕРЕНА
Рецензент
_____
«___»_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____Г.И. Радченко
«___»_____ 2019 г.

Разработка веб-приложения для проектирования искусственных нейронных сетей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____И.Л. Надточий
«___»_____2019 г.

Автор работы,
студент группы КЭ-222
_____И.В. Пищаев
«___»_____2019 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Г.И. Радченко
«___»_____2019 г.

ЗАДАНИЕ

на выпускную квалификационную работу магистра

студенту группы КЭ-222
Пищаев Илья Владиславович
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка веб-приложения для проектирования искусственных нейронных сетей» утверждена приказом по университету от 25 апреля 2019 г. №899

2. **Срок сдачи студентом законченной работы:** 1 июня 2019 г.

3. **Исходные данные к работе:** статьи, книги, техническое задание.

– Онлайн курс "Программирование глубоких нейронных сетей на Python" - https://www.asozykin.ru/courses/nnpython. Дата обращения: 25.03.2019.
– Томас, М. Т. React в действии / Питер - Для профессионалов - Отдельное издание, 2019. – 368 с.

– Федоров, Д. Ю. Программирование на языке высокого уровня Python / Д. Ю. Федоров. — 2-е изд., перераб. и доп. — М. : Издательство Юрайт, 2019. — 161 с.
– СУБД MySQL 5.0.12.

4. **Перечень подлежащих разработке вопросов:**

– изучение и сравнение аналогов для работы с искусственными нейронными сетями;
– проектирование веб-приложения для проектирования искусственных нейронных сетей;
– реализация веб-приложения для проектирования искусственных нейронных сетей;
– тестирование веб-приложения для проектирования искусственных нейронных сетей.

5. **Дата выдачи задания:** 1 декабря 2018 г.

Руководитель работы _____/*И.Л. Надточий*/

Студент _____/*И.В. Пищаев*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	01.03.2019	
Разработка модели, проектирование	01.04.2019	
Реализация системы	01.05.2019	
Тестирование, отладка, эксперименты	15.05.2019	
Компоновка текста работы и сдача на нормоконтроль	24.05.2019	
Подготовка презентации и доклада	30.05.2019	

Руководитель работы _____/*И.Л. Надточий*/

Студент _____/*И.В. Пищаев*/

Аннотация

И.В. Пищаев. Разработка веб-приложения для проектирования искусственных нейронных сетей. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2019, 83 с., 53 ил., библиогр. список – 18 наим.

В рамках выпускной квалификационной работы производится анализ технологий построения топологий искусственных нейронных сетей. Организуется разработка веб-приложения для организации обучения, построения и тестирования искусственных нейронных сетей. Производится анализ результатов работы системы, её преимуществ и недостатков.

Во введении рассматривается актуальность данной темы.

В первой главе производится анализ предметной области и обзор аналогов.

Во второй главе производится определение требований для данной системы, определение функциональных требований.

В третьей главе присутствует проектирование системы, ее архитектура и описание данных.

В четвертой главе показана реализация данной системы, ее интерфейсов.

В пятой главе производится тестирование системы.

В заключении описаны итоги проделанной работы.

Библиографический список показывает все дополнительные материалы и статьи, к которым было обращение при создании системы.

В приложении представлен исходный код системы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность.

На сегодняшний день глубокие нейронные сети один из самых популярных методов машинного обучения. Нейронная сеть строится из простых вычислительных элементов – искусственных нейронов. Принцип работы искусственных нейронных сетей схож с принципом работы нейронов головного мозга [1].

Преимущество данного метода машинного обучения над другими методами состоит в том, что в традиционном машинном обучении производится выбор из большого имеющегося объема данных те данные, которые позволяют решить заданную задачу. Выбором этих данных в традиционном машинном обучении занимается человек, и если удалось найти подходящие признаки, то задача будет решена, если не удалось, то задача не решена. Нейронные сети в свою очередь автоматически отделяют нужные данные от ненужных и умеют выбирать правильные признаки.

Последние несколько лет активно ведутся разработки платформ, позволяющих строить НС. Такие платформы позволяют работать с НС без прямого кодирования, значительно упрощая процесс.

Цель исследования.

Целью представленной выпускной квалификационной работы является разработка веб-приложения для построения топологий искусственных нейронных сетей.

Для достижения поставленной цели, необходимо решить следующие поставленные задачи:

1. Изучение и сравнение аналогов.
2. Проектирование веб-приложения.
3. Реализация веб-приложения.
4. Тестирование.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. ОБЗОР АНАЛОГОВ

Существует немало веб-приложений и сервисов для работы с искусственными нейронными сетями. Далее будут рассмотрены некоторые из этих систем.

Веб-сервис «A Neural Network Playground»

Электронный ресурс [2], который позволяет строить различные топологии НС (рисунок 1).

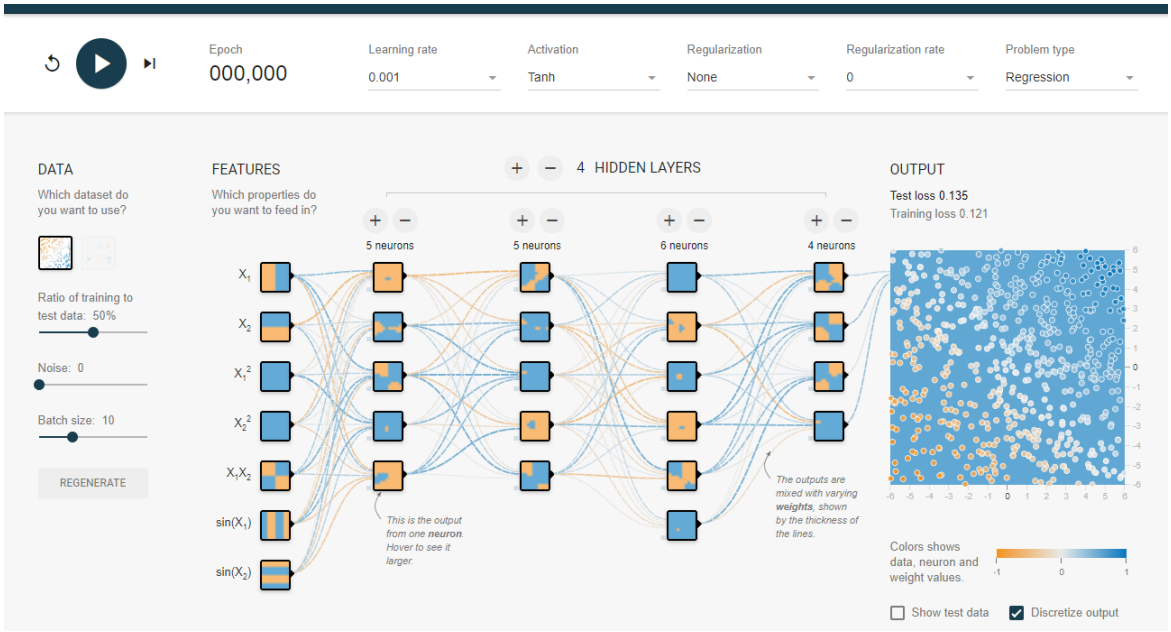

Рисунок 1 – Область работы

Оранжевый и синий используются во всей визуализации немного по-разному, но в целом оранжевый показывает отрицательные значения, а синий показывает положительные значения.

Точки данных (представленные маленькими кружками) изначально окрашены в оранжевый или синий цвета, которые соответствуют положительному и отрицательному.

В скрытых слоях линии окрашены весами связей между нейронами. Синий цвет показывает положительный вес, что означает, что сеть

использует этот выход нейрона, как указано. Оранжевая линия показывает, что сеть набирает отрицательный вес.

В выходном слое точки окрашиваются в оранжевый или синий цвета в зависимости от их исходных значений. Цвет фона показывает, что сеть предсказывает для определенной области. Интенсивность цвета показывает, насколько достоверен этот прогноз.

Плюсы данной платформы:

1. Возможность выбирать параметры для НС:

a) X1.

b) X2.

c) X12.

d) X22.

e) X1X2.

f) sin(X1).

g) sin(X2).

2. Возможность выбирать скрытые слои НС, от 0 до 6.
3. Возможность выбирать количество нейронов в каждом слое, от 1 до 8.
4. Проблемы задачи классификации и регрессии.
5. Выбор типа регуляризации:

a) Отсутствует.

b) L1.

c) L2.

6. Выбор коэффициента регуляризации:

a) 0,001.

b) 0,003.

c) 0,01.

d) 0,03.

e) 0,1.

f) 0,3.

g) 1.

h) 3.

i) 10.

7. Возможность обучения НС, а также выбор коэффициента обучения:

a) 0,00001.

b) 0,0001.

c) 0,001.

d) 0,003.

e) 0,01.

f) 0,03.

g) 0,1.

h) 0,3.

i) 1.

j) 3.

k) 10.

8. Выбор одной из функций активации:

a) Tanh.

b) ReLU.

c) Sigmoid.

d) Linear.

9. Выбор типа набора данных:

a) Circle.

b) Exclusive or.

c) Gaussian.

d) Spiral.

10. Возможность выбора размера выборки, от 1 до 30.

Минусы данной платформы:

1. Нельзя создавать свои параметры, подаваемые на вход НС.
2. Нельзя задавать свою выборку данных.
3. Несмотря на возможность выбора количества слоев в сети и количества нейронов в них, это количество ограниченно. Если нужна будет сеть, у которой будет, например 100 нейронов, данная платформа будет бесполезна в таком случае.
4. Жесткие рамки в выборе коэффициентов регуляризации и обучения.
5. Ограниченный размер выборки.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа является отличным веб-сервисом для работы с НС, так как её плюсы значительно перевешивают минусы.

Веб-сервис «ConvnetJS demo»

Электронный ресурс [3], который позволяет работать с НС (рисунок 2).

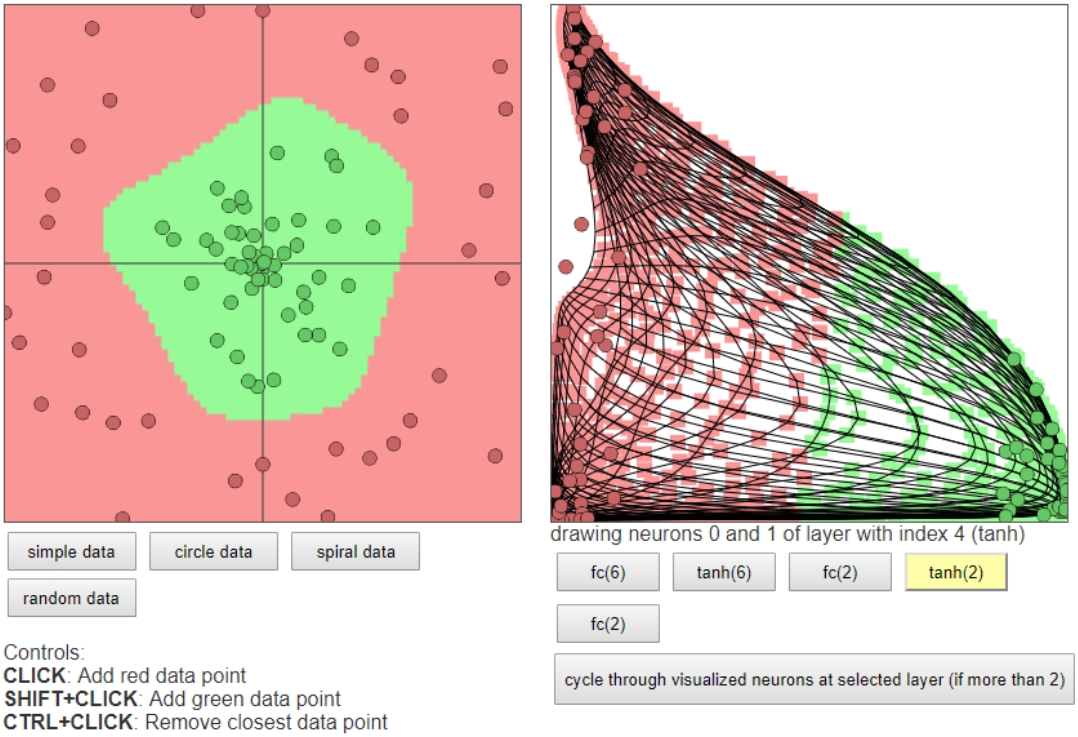

Рисунок 2 – Область работы

Справа визуализируется преобразованное представление всех точек сетки в исходном пространстве и данных для данного слоя и только для 2

нейронов одновременно. Число в скобках показывает общее количество нейронов на этом уровне представления. Если число больше 2, то будет только два визуализированных слоя, но вы можете просмотреть все из них с помощью кнопки цикла.

Плюсы данный платформы:

1. Возможность просмотреть каждый слой сети.
2. Возможность выбора типа данных:

a) Simple data.

b) Circle data.

c) Spiral data.

d) Random data.

Минусы данной платформы:

1. Для того, чтобы изменить количество слоев и нейронов в них, в данной платформе присутствует область редактирования НС (рисунок 3), однако такой способ может быть неудобен людям, которые не умеют кодировать НС напрямую.
2. В режиме визуализации пользователь может увидеть только 5 слоев НС.
3. Нельзя создавать свои параметры, подаваемые на вход НС.
4. Отсутствует обучение НС.

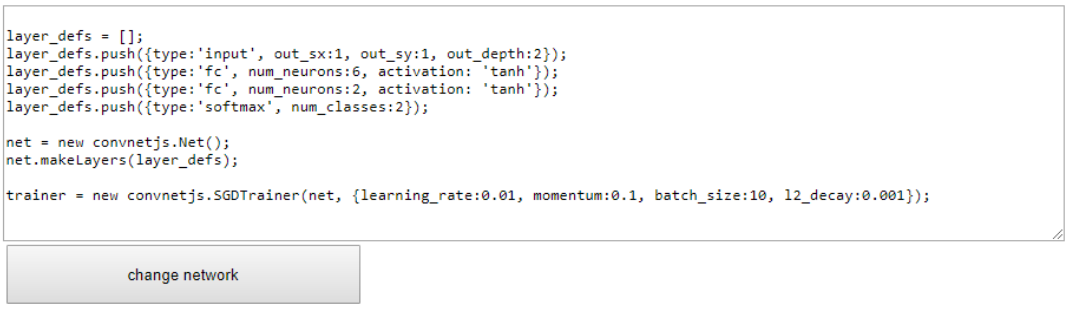

Рисунок 3 – Область редактирования НС

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа не является отличным веб-сервисом для работы с НС, так как её минусы перевешивают плюсы.

Веб-сервис «Machine Learning Playground»

Электронный ресурс [4], которая позволяет работать с НС и её обучением (рисунок 4).

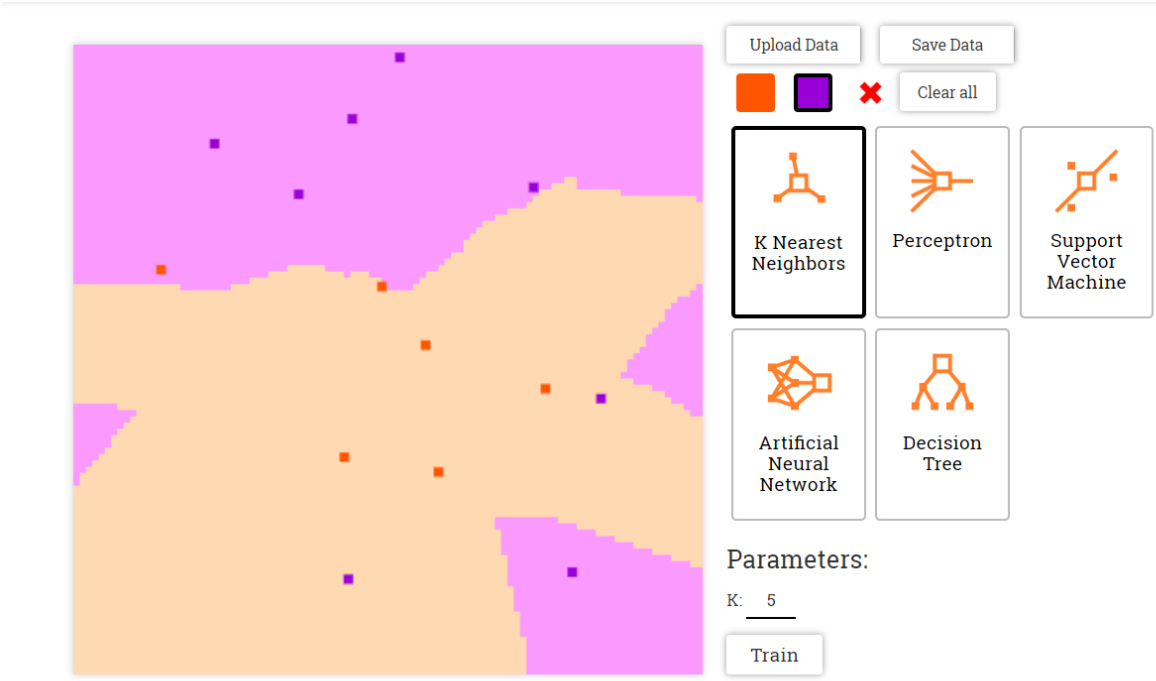

Рисунок 4 – Область работы

Параметры:

1. k (≥ 1): количество ближайших соседей для выбора.
2. Случаи применения:

a. Бинарная классификация.

b. Мультиклассовая классификация.

c. Регрессия.

Плюсы данной платформы:

1. Возможность задавать свою выборку данных для НС.
2. Позволяет работать со следующими НС:

a. Метод k-ближайших соседей, где k может быть любым натуральным числом.
b. Метод опорных векторов.
c. Персептрон, где количество выходов натуральное число от 1 до 99.
d. Количество эпох, натуральное число от 0 до 999.
e. Дерево принятия, с максимальной глубиной дерева равной натуральному числу от 0 до 99.
f. Искусственная НС:

I. Параметры X и Y.
II. Максимальное количество слоев в сети равно 8.
III. Максимальное количество нейронов в каждом слое равно 10.
IV. Коэффициент обучения действительное число от 0 до 999.

Минусы данной платформы:

1. Сложный порог вхождения, сложно понять, что требует данный сайт и какие данный подавать на вход.
2. Несмотря на возможность выбора количества слоев в сети и количества нейронов в них, это количество ограниченно. Если нужна будет сеть, у которой будет, например 100 нейронов, данная платформа будет бесполезна в таком случае.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа является отличным веб-сервисом для работы с НС, так как её плюсы значительно перевешивают минусы.

Веб-сервис «Web-based 3D Neural Network Playground»

Электронный ресурс [5], который позволяет строить различные топологии НС в 3D формате (рисунок 5).

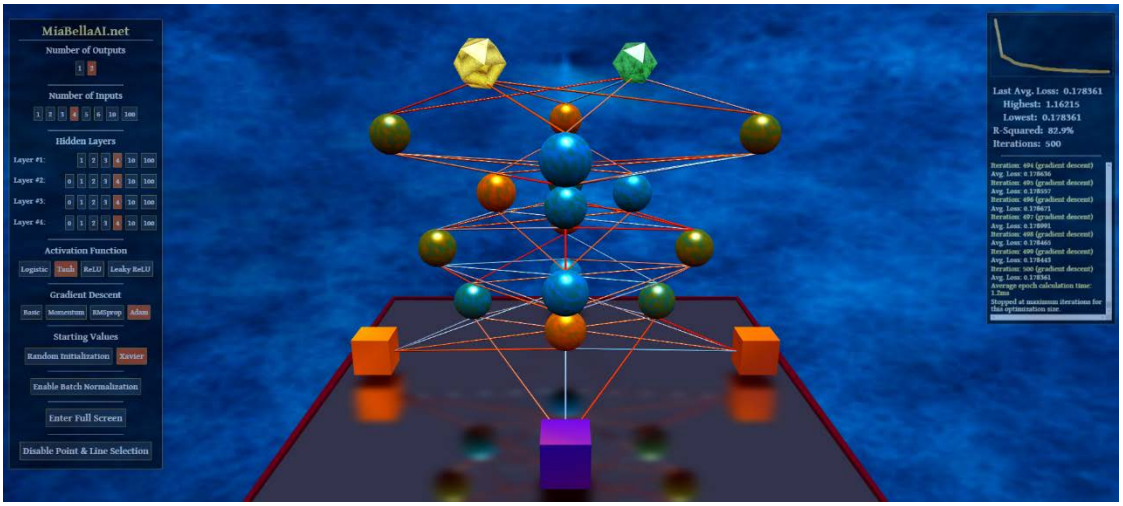

Рисунок 5 – Область работы

MiaBella ANN - это интерактивный веб-инструмент визуализации для изучения внутренней работы искусственных нейронных сетей.

Программа показывает глубокую обучающую нейронную сеть с шестью слоями. Можно изменить слои, входы и даже количество выходов.

Это приложение выполняет оптимизацию градиентного спуска в зависимости от конфигурации нейронной сети, которую вы выберете. Набор данных создается случайным образом на каждом чертеже, а выходные данные генерируются из случайных уравнений на основе входных данных.

По мере оптимизации можно наблюдать в реальном времени, как нейроны и их соединения изменяются в соответствии с данными.

Плюсы данной платформы:

1. Возможность задавать выходные данные в количестве 1 и 2.
2. Возможность задавать входные данные в количестве 1, 2, 3, 4, 5, 6, 10 и 100.
3. Возможность выбирать слои для работы в количестве от 1 до 4.
4. Каждый слой имеет определенное количество нейронов равное 1, 2, 3, 4, 10 и 100.
5. Выбор одной из функций активации:

a) Logistic.

b) Tanh.

c) ReLU.

d) Leaky ReLU.

6. Возможность задать градиентный спуск:

e) Basic.

f) Momentum.

g) RMSprop.

h) Adam.

Минусы данной платформы:

1. Медленная скорость работы.
2. Несмотря на возможность выбора количества слоев в сети и количества нейронов в них, это количество ограниченно. Если нужна будет сеть, у которой будет, например 100 нейронов, данная платформа будет бесполезна в таком случае.
3. Нельзя создавать свои параметры, подаваемые на вход НС.
4. Нельзя задавать свою выборку данных.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа является отличным веб-сервисом для работы с НС, так как её плюсы значительно перевешивают минусы.

Веб-сервис «Nvidia Image Painting».

Электронный ресурс [6], который позволяет редактировать изображения, стирая ненужные элементы и заменяя их фоном или частью изображения, которая находится рядом.

Метод работы данного ресурса состоит в том, что пользователь выбирает нужное ему изображения (рисунок 6), где красным выделена область, которую пользователь хочет заменить.

Рисунок 6 – Оригинальное изображение

И далее, выделяя кисточкой нужную область, он ее меняет, и в результате получается измененное изображение (рисунок 7), где, как видно, выбранная нами область закрашена в соответствии с рядом находящейся областью.

Так в зависимости от выбранной области происходит перекрашивание соседних блоков изображения.

Таким образом, пользователь может затирать шероховатости изображения, либо удалять ненужные ему элементы, как, например, на данном изображении (рисунок 6).

Задачей было убрать белую шерсть с лица кота и закрасить ее монотонным цветом шерсти, которая находится рядом (рисунок 7).

Рисунок 7 – Результат обработки

Плюсы данной платформы:

1. Хорошая точность работы с изображениями.
2. Быстрая обработка изображений.

К минусам данной платформы можно отнести то, что она не позволяет сохранять данные.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа не является отличным веб-сервисом для работы с НС, так как её минусы перевешивают плюсы.

Веб-сервис «AutoDraw»

Данный электронный ресурс [7] позволяет рисовать изображения, предлагая пользователю предложенные варианты готовых изображений, на основе его действий с кистью.

На экране (рисунок 8) пользователь видит чистый лист, на котором он может отрисовывать различные изображения, линии и т.д.

Рисунок 8 – Начальный экран работы

Дале пользователь может нарисовать линию, и программа предложит ему различные варианты изображений, на основе той линии, которую он нарисовал.

Как видно (рисунок 9), была нарисована дуга, и программа стала предлагать различные варианты изображений, которые содержат эту дугу.

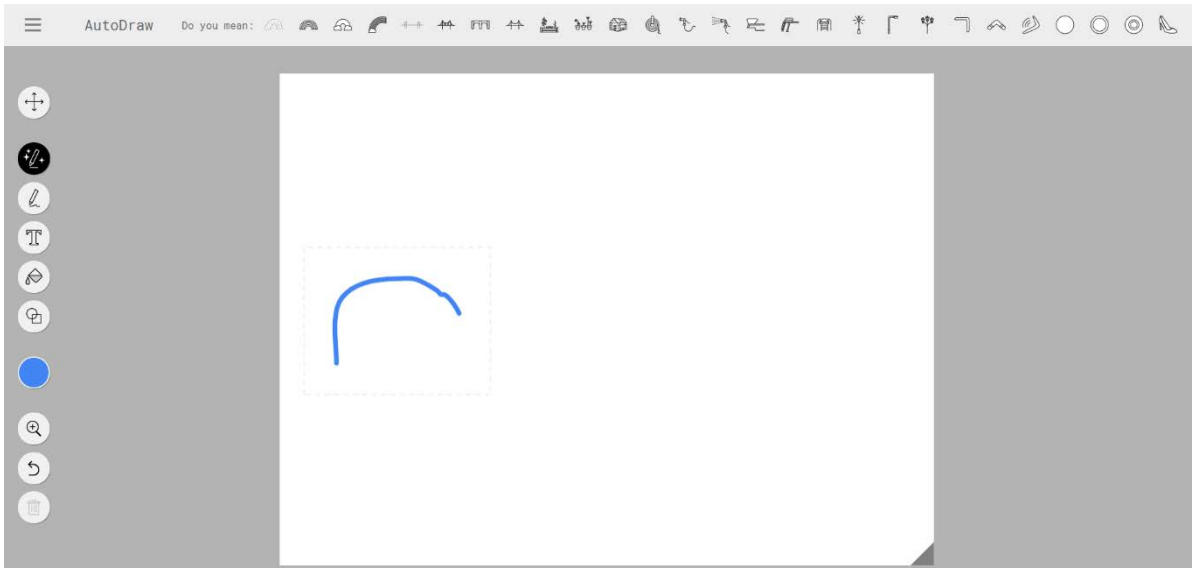

Рисунок 9 – Предложенные варианты изображений

Плюсы данной платформы:

1. Простой и удобный интерфейс.
2. Быстрая обработка изображений и поиск подходящий результатов.
3. Возможность сохранить изображение.

Минусы данной платформы:

1. Иногда программа не может хорошо определить вводное изображение и показывает не тот результат, который ожидается.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа не является отличным веб-сервисом для работы с НС, так как её минусы перевешивают плюсы.

Вебз-сервис «Free logo maker»

Данный электронный ресурс [8] позволяет создать лого компании пользователя на основе его личных предпочтений.

На стартовом экране (рисунок 10) пользователю предлагается выбрать название компании.

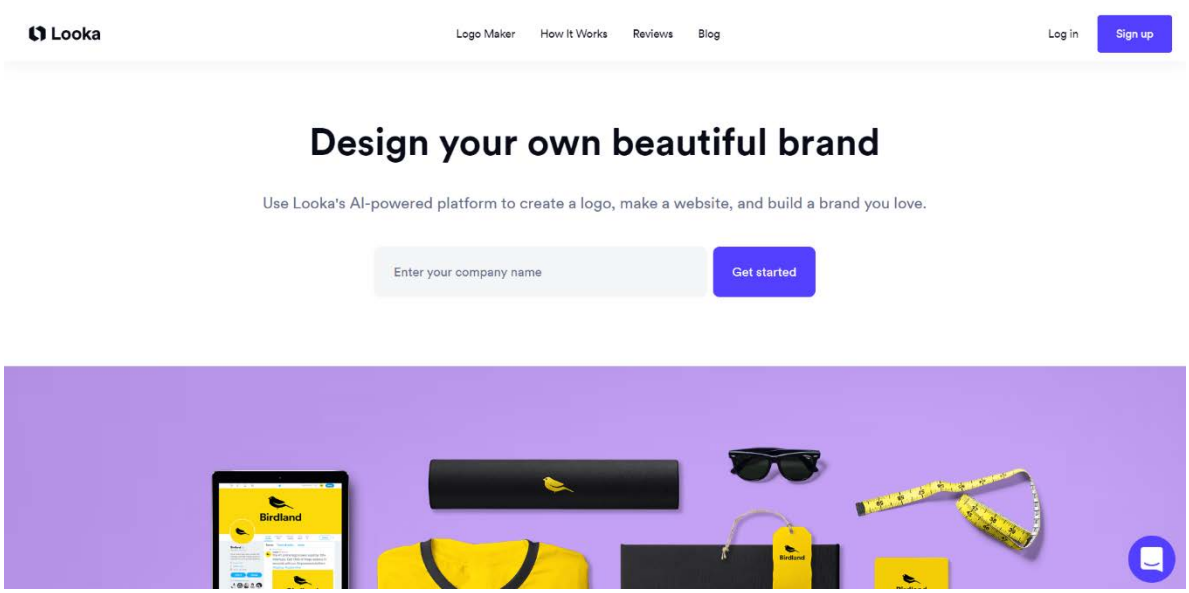

Рисунок 10 – Стартовый экран

Далее предлагается (рисунок 11) выбрать тип компании, такие как:

1. Ресторан.
2. Коффе-бар.

3. Консалтинговая компания и т.д.

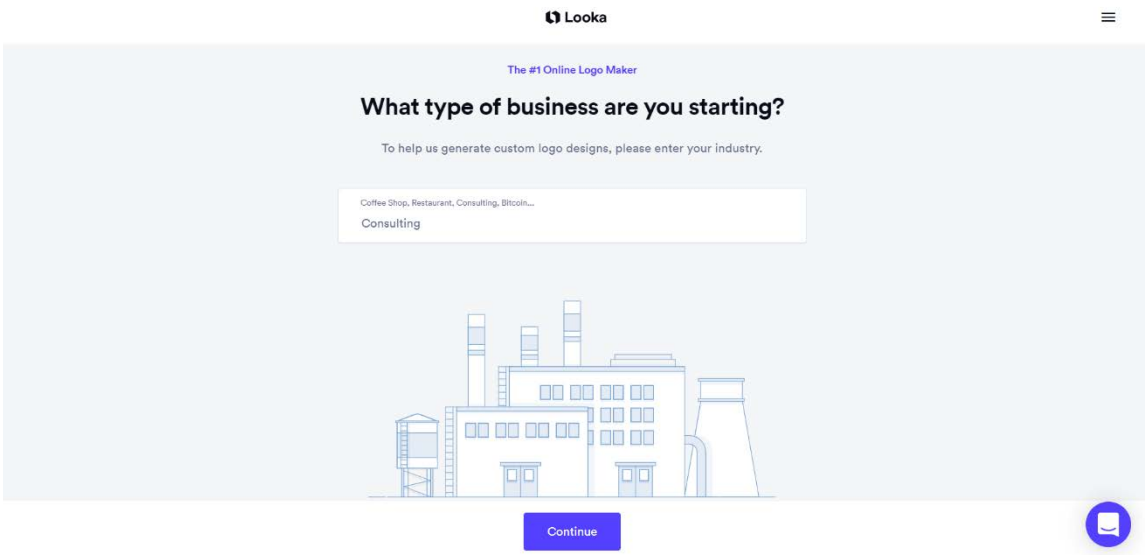

Рисунок 11 – Экран выбора типа компании

Затем пользователю предлагается (рисунок 12) выбрать 5 понравившихся логотипов различных компаний, чтобы определить его вкусовые предпочтения.

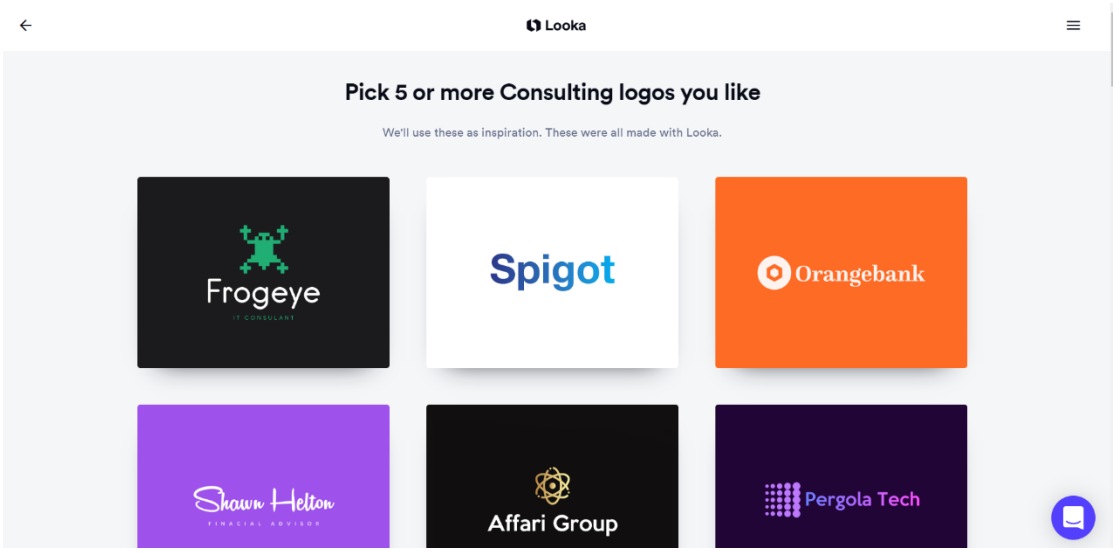

Рисунок 12 – Экран выбора понравившегося логотипа

Далее сайт предлагает выбрать (рисунок 13) два понравившихся цветовых стиля. Данный шаг можно пропустить.

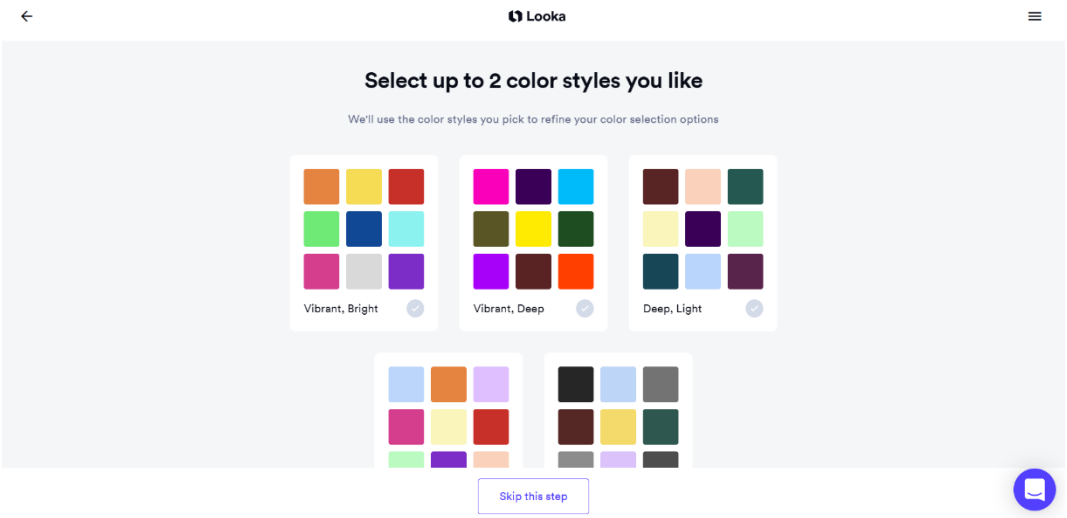

Рисунок 13 – Экран выбора понравившейся цветовой гаммы

Затем предлагается выбрать (рисунок 14) три понравившихся цвета. Данный шаг можно пропустить.

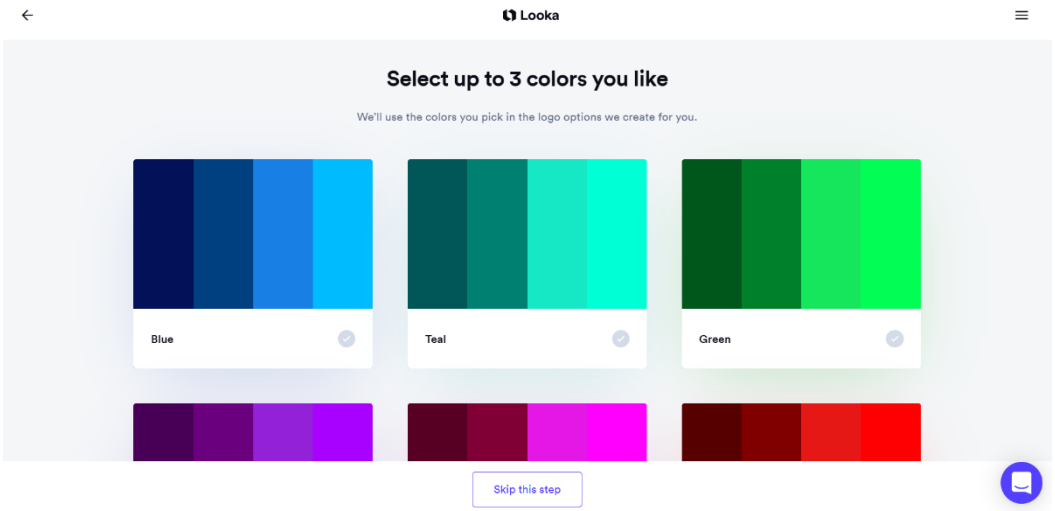

Рисунок 14 – Экран выбора понравившегося цвета

Затем пользователю предлагается написать (рисунок 15) слоган своей компании.

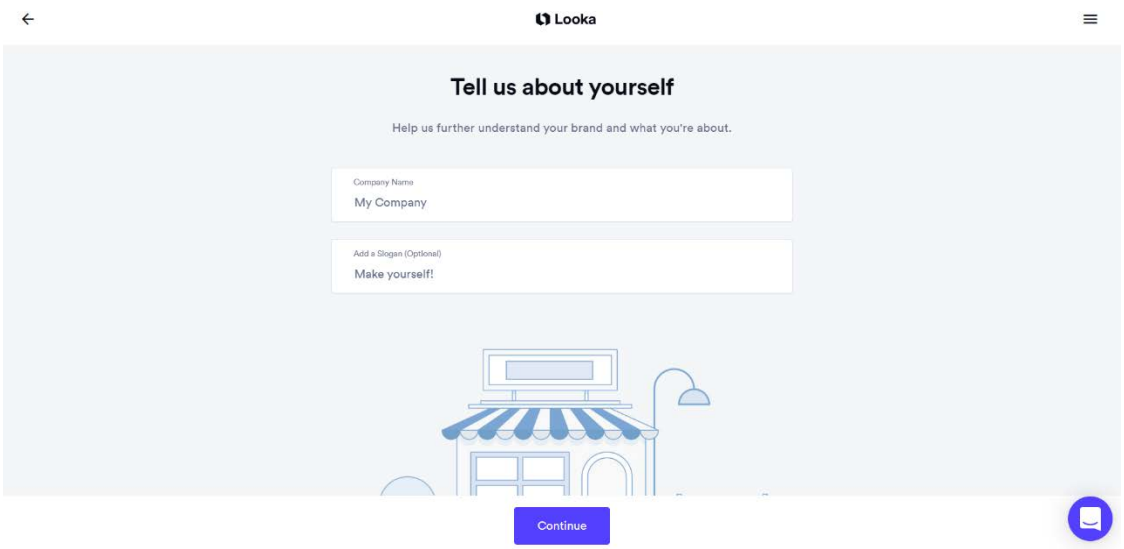

Рисунок 15 – Экран выбора слогана

Далее предлагается выбрать (рисунок 16) понравившиеся символы для своей компании. Данный шаг можно пропустить.

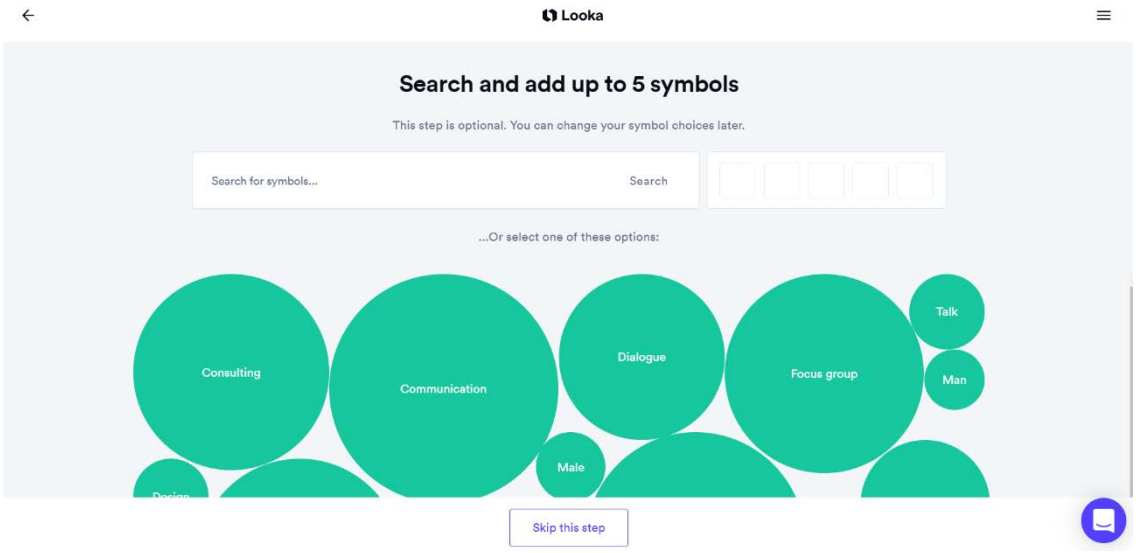

Рисунок 16 – Рисунок выбора понравившихся символов для компании.

В результате работы программы (рисунок 17) на выходе пользователь видит различные варианты логотипов своей компании.

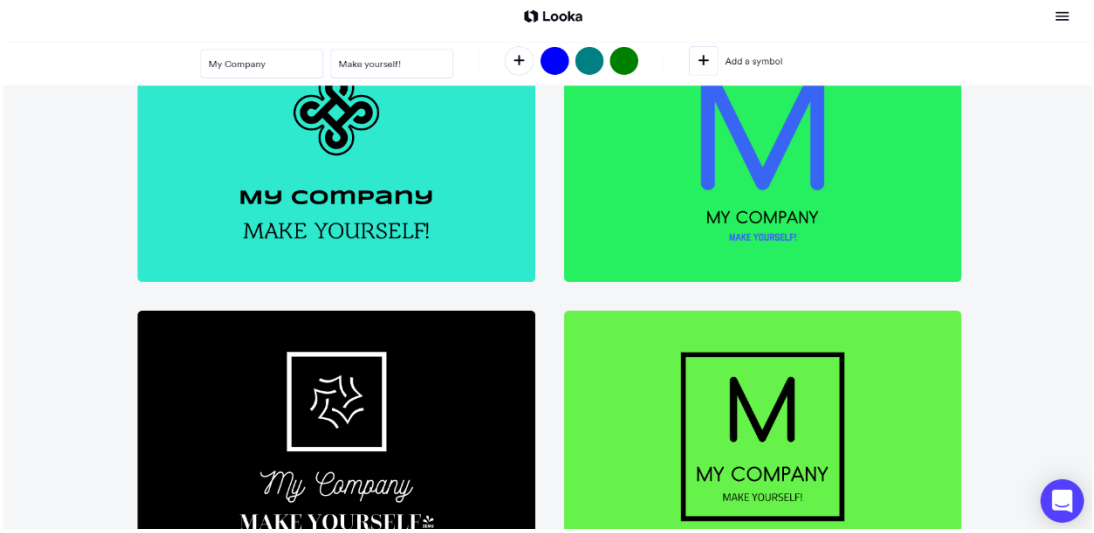

Рисунок 17 – Результат работы

Плюсы данного ресурса:

1. Простой и удобный интерфейс.
2. Быстрая обработка результатов.

Минусы данного ресурса:

1. Нельзя сохранить полученный результат без авторизации или регистрации.

Исходя и вышеназванных плюсов и минусов, можно предположить, что данная платформа является отличным веб-сервисом для работы с НС, так как её плюсы значительно перевешивают минусы.

1.2. ВЫВОД

Вышерассмотренные платформы имеют в своем распоряжении множество как плюсов, так и минусов. Однако не каждая из этих платформ позволяет работать с топологиями НС напрямую, а так же сохранять их и загружать. Работа на данных пользователя тоже отсутствует.

Поэтому принято решение создания Веб-приложения которое позволит:

1. Работать с искусственными нейронными сетями напрямую, позволяя выбирать тип слоев и их различные параметры.
2. Позволит сохранять полученные искусственные нейронные сети.
3. Позволит обучать созданные пользователем искусственные нейронные сети, а так же сохранять их веса.
4. Позволит пользователю создавать свой набор данных для задачи классификации, а так же сохранять его.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Общие требования.

Для реализации данной системы необходим следующий набор подсистем:

1. Сервер на основе Flask (Python) [17] [18], который обеспечивает управление MySQL сервером и управление искусственными нейронными сетями. Flask предоставляет API для связи Front-End с MySQL и с модулем управления искусственными нейронными сетями.
2. MySQL база данных, обеспечивающая хранение данных о пользователях, их наборах искусственных нейронных сетей, а так же наборах данных.
3. Графический интерфейс клиентского приложения – визуальное отображение регистрации и авторизации в системе, данные о сохраненных искусственных нейронных сетях, загруженных данных и управление ими. Данный интерфейс создан в виде React-приложения [10] [11] [12]. С помощью сервера и веб-приложения обеспечивается общее управление системой и работы с искусственными нейронными сетями. Будет возможно создавать искусственные нейронные сети, сохранять и обучать их.

Так же для работы с нейронными сетями было решено взять язык программирования Python [14] [15] [16], так как данный язык обладает удобным и понятным кодом и встроенными библиотеками для работы с нейронными сетями, такими как Keras, TensorFlow, Theano [1] [13].

2.1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Далее будет представлена диаграмма вариантов использования (рисунок 18).

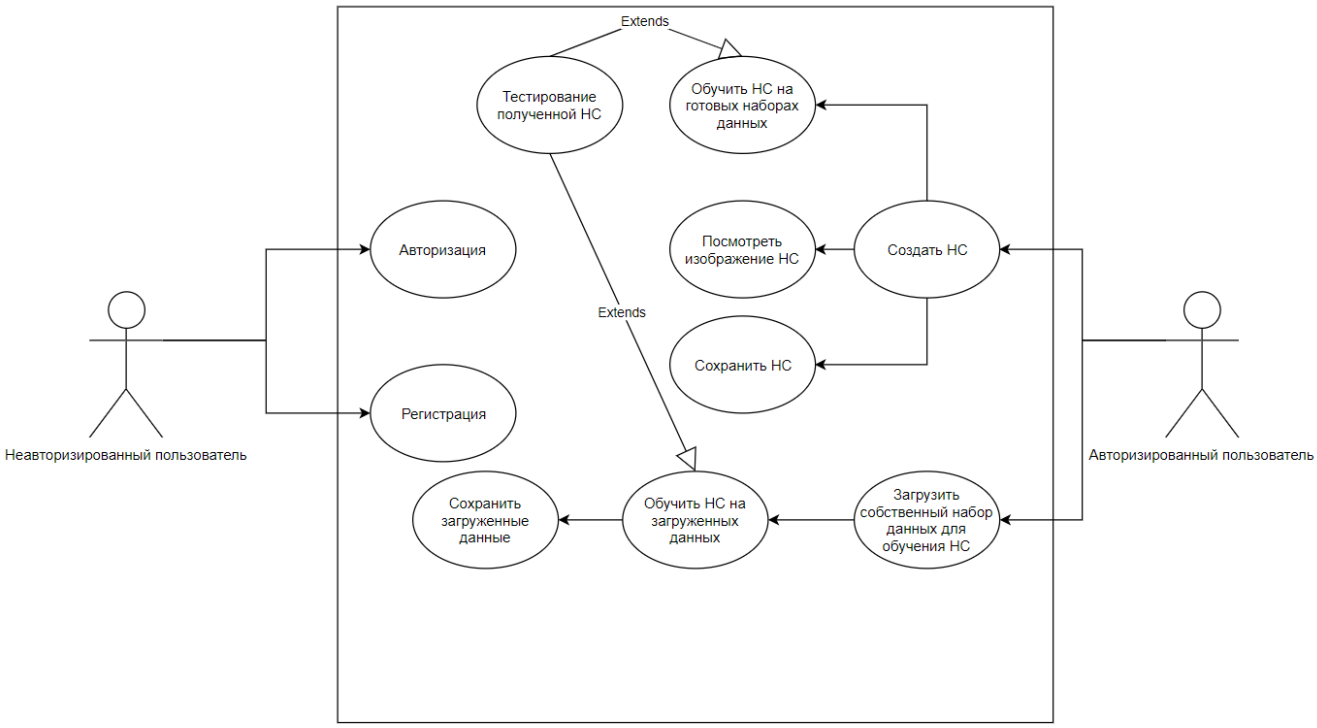

Рисунок 18 – Диаграмма вариантов использования

В данной системе присутствую два типа пользователей:

1. Незарегистрированный пользователь.
2. Зарегистрированный пользователь.

Незарегистрированному пользователю доступны следующие действия:

1. Авторизация. Пользователь авторизируется в системе.
2. Регистрация. Пользователь регистрируется в системе.

Зарегистрированному пользователю доступны следующие действия:

1. Создать искусственную нейронную сеть. Создается новая искусственная нейронная сеть содержащая в себе 0 слоев.
2. Обучить искусственную нейронную сеть на готовых наборах данных. Существуют готовые наборы для обучения искусственных нейронных

сетей, которые можно использовать для обучения собственной искусственной нейронной сети для решения задачи классификации.

3. Посмотреть изображение искусственной нейронной сети. Искусственные нейронные сети состоят из слоев, которые содержат в себе нейроны, которые связаны между собой или другими нейронами других слоев, в зависимости от типа слоя. Данная опция позволяет посмотреть как выглядит данная связь для выбранной искусственной нейронной сети пользователя.
4. Сохранить искусственную нейронную сеть. Пользователь может сохранить свою искусственную нейронную сеть, чтобы потом обучить ее не других данных или протестировать ее работу позже.
5. Загрузить собственных набор данных для обучения искусственной нейронной сети. Пользователь может выбрать собственный набор данных для обучения искусственной нейронной сети для задачи классификации.
6. Обучить искусственную нейронную сеть на загруженных данных. Заданная искусственная нейронная сеть обучается на данных, которые загрузил пользователь и занимается решением задачи классификации, то есть учится определять принадлежность выбранного изображения к классу, из списка классов, созданных пользователем.
7. Сохранить загруженные данные. Пользователь может сохранить данные, которые он загрузил, чтобы потом обращаться к ним для обучения или тестирования работы искусственной нейронной сети по решению задачи классификации.
8. Тестирование полученной искусственной нейронной сети. После того как искусственная нейронная сеть была обучена, или была выбрана из уже обученных, сохраненный искусственных нейронный сетей, пользователь может загрузить изображение, чтобы проверить его

принадлежность к определенному списку классов. Если данная искусственная нейронная сеть работает с высоко точностью, то она сможет верно определить принадлежность изображения к нужному классу изображений, в противном случае, сети будет требоваться дообучение или изменение, либо добавление слоев.

В результате на основе функциональных требований был создан макет веб-приложения (рисунок 19).

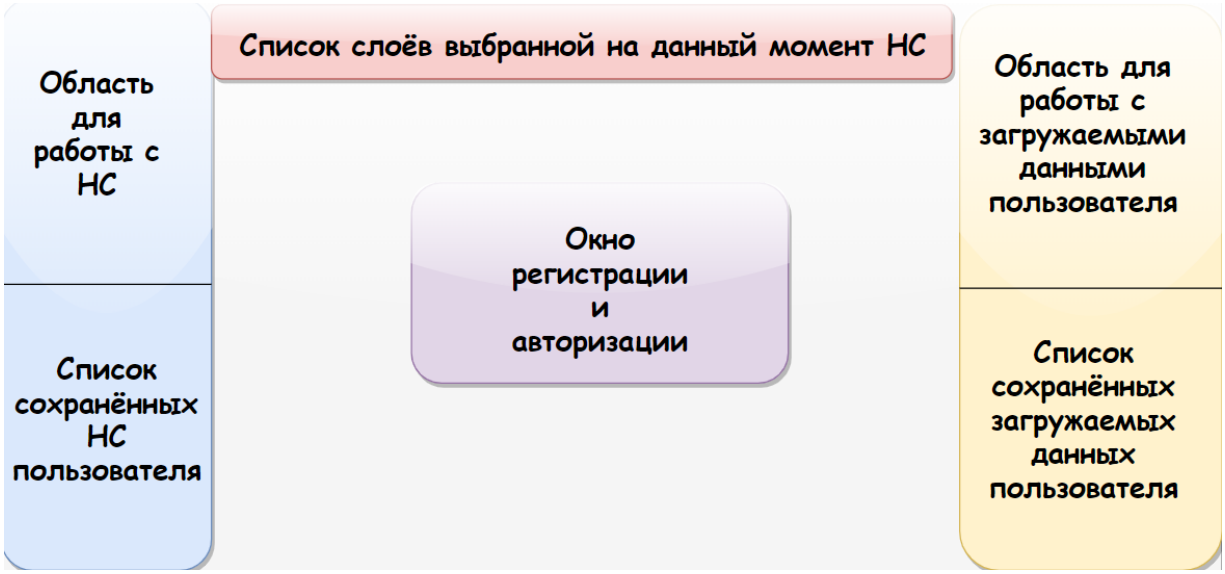

Рисунок 19 – Макет сайта

3. ПРОЕКТИРОВАНИЕ

3.1. АРХИТЕКТУРА ПРЕДЛАГАЕМОГО РЕШЕНИЯ

При описании архитектуры разрабатываемого веб приложения была спроектирована диаграмма компонентов [9] для отображения взаимодействия логических частей приложения, представленная (рисунок 20).

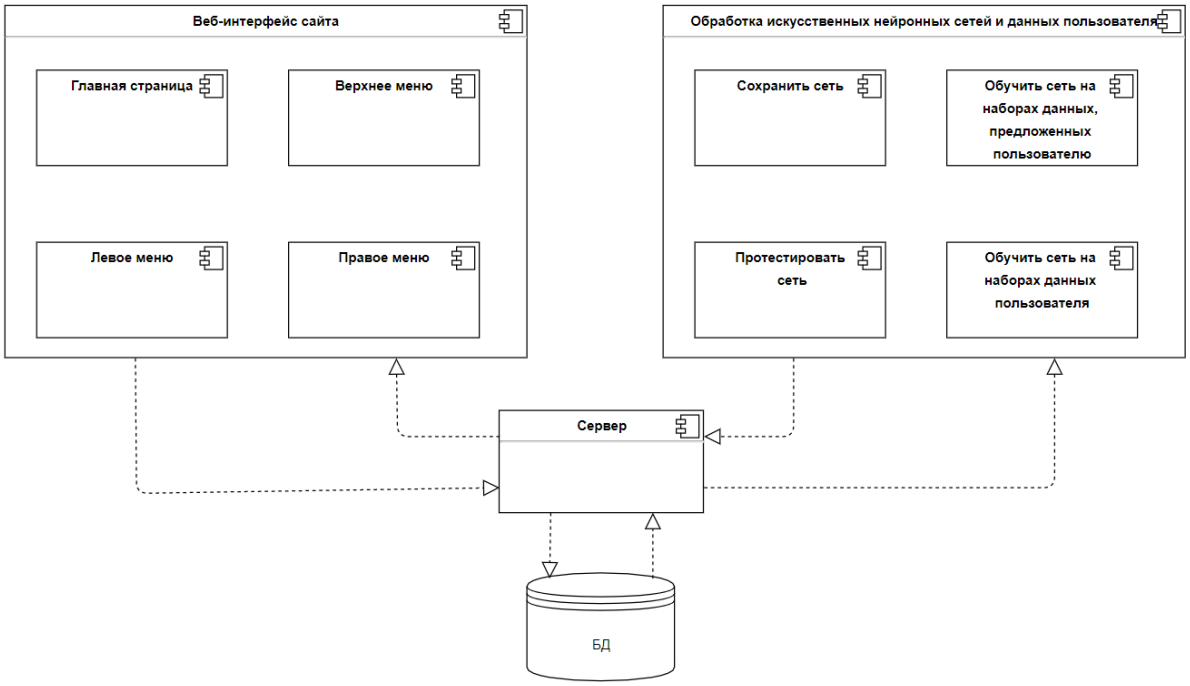

Рисунок 20 – Диаграмма компонентов проектируемой системы

В проектируемой системе предполагается создание четырех главных компонентов системы.

Первым главным компонентом является веб-интерфейс создаваемого приложения. Он будет иметь 4 основных компонента:

1. Главная страница. Главная страница будет содержать в себе регистрацию и авторизацию пользователя, а так же все остальные компоненты Веб-приложения.

2. Левое меню. Левое меню будет содержать в себе основные опции и функции для работы с искусственными нейронными сетями. Эти сети можно будет создавать, сохранять, обучать, а так же посмотреть как выглядит сеть, то есть посмотреть вид ее слоев и нейронов, и их связей. А так же будет выводить список уже имеющихся, сохраненных сетей пользователя.
3. Верхнее меню. Верхнее меню будет содержать в себе список слоев выбранной на данных момент искусственной нейронной сети.
4. Правое меню. Правое меню будет содержать в себе основные опции и функции для работы с загруженными данными пользователями. Данные можно будет загружать, обучать на них различные искусственные нейронные сети, сохранять эти данные, а так же тестировать сеть. В добавок, в правом меню, будет выводиться список всех уже имеющихся данных пользователя, которые он сохранял до этого.

Вторым главным компонентом является сервер.

Этот компонент получает данные от веб-интерфейса. Эти данные означают действия, которые требуется совершить, а так же параметры искусственных нейронных сетей и загруженных данных пользователя.

Если пользователь авторизируется, то сервер отправляет запрос к Базе Данных и проверяет, есть ли данный пользователь. Затем ответ отправляется на главный компонент веб-интерфейс.

Если пользователь регистрируется, то сервер отправляет запрос к базе данных. Если данный пользователь существует, то на главный веб-интерфейс компонент отправляется ответ, что данных пользователь существует, если пользователь не существует, то данных пользователь сохраняется в Базу Данных, а на главный компонент веб-интерфейс отправляется уведомление о том, что регистрация прошла успешно.

Если пользователь хочет сохранить полученные искусственные нейронные сети, загруженные данные, тогда сервер обращается к базе данных, третьему главному компоненту, куда и сохраняет полученные данные.

Сервер постоянно обращается к базе данных, для того, чтобы проверить какие на данных момент у пользователя имеются сохраненные искусственные нейронные сети или его собственные данные, которые он загрузил для решения задачи классификации. Получив информацию о сохраненных данных пользователя, сервер передает данную информацию главному компоненту веб-интерфейс, который, получив данный список, выводит его в своих компонентах.

Если пользователь хочет загрузить выбранную искусственную нейронную сеть, или свой набор данных, которые он до этого сохранил, то сервер обращается к базе данных для получения информации по выбранной сети или набору данных. Затем эти данные передаются с сервера к главному компоненту веб-интерфейс, где эти данные выводятся в других компонентах, принадлежащих главному компоненту веб-интерфейс.

Третьим главным компонентом является база данных.

В ней хранятся данные о всех зарегистрированных пользователях системы, об их сохраненных искусственных нейронных сетях, а так же об их сохраненных наборах данных для обучения искусственной нейронной сети для решения задачи классификации.

Четвертым главным компонентом является компонент под названием Обработка искусственных нейронных сетей и данных пользователя. Он имеет 4 основных компонента:

1. Сохранить сеть. Искусственная нейронная сеть, которую создал пользователь сохраняется в базу данных со всеми своими параметрами.

2. Обучить сеть на наборах данных, предложенных пользователю. Существуют уже готовые наборы данных, которые позволяют обучать искусственные нейронные сети. Пользователь может выбрать один из наборов данных, на котором хочет обучать свою искусственную нейронную сеть, для решения задачи классификации.
3. Обучить сеть на наборах данных, которые загрузил пользователь. Искусственную нейронную есть, созданную пользователем можно обучить на предварительно загруженных данных, которые были им предоставлены.
4. Протестировать сеть. Пользователь может протестировать результат работы искусственной нейронной сети, которую он создал и обучил, для решения задачи классификации. Для этого ему следует выбрать любое изображение, которое может принадлежать к одному из классов обучения и загрузить в компонент тестирования. В результате искусственная нейронная сеть покажет пользователю к какому классу принадлежит данное изображение. Данное тестирование позволяет проверить точность работы искусственной нейронной сети. Если сеть недостаточно точно определяет принадлежность изображения к какому-нибудь классу, то пользователь может ее до обучить и снова протестировать.

3.2. ОПИСАНИЕ ДАННЫХ

Описание базы данных.

В данной системе была спроектирована [9] база данных (рисунок 21).

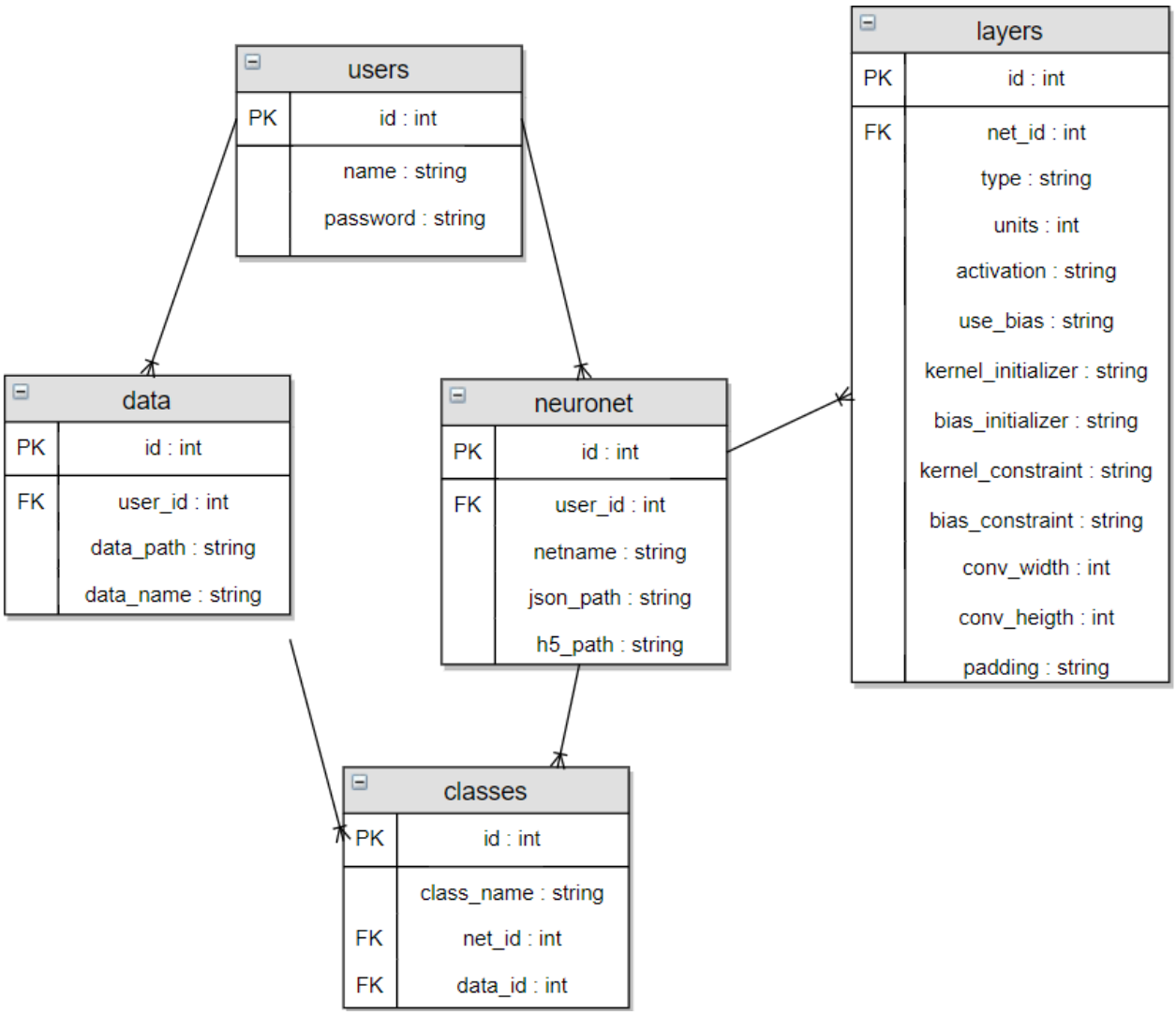

Рисунок 21 – База данных

В базе данных присутствуют 5 таблиц:

1. Users.
2. Data.
3. Neuronet.
4. Layers.
5. Classes.

Таблица «users» содержит в себе информацию о зарегистрированных пользователей системы и имеет следующие поля:

1. Id. Тип integer. Первичный ключ таблицы.
2. Name. Тип string. Имя зарегистрированного пользователя.
3. Password. Тип string. Пароль зарегистрированного пользователя.

Таблица «data» содержит в себе данные, которые пользователь загрузил для обучения своей созданной искусственной нейронной сети, и имеет следующие поля:

1. Id. Тип integer. Первичный ключ таблицы.
2. User_id. Тип integer. Внешний ключ таблицы. Содержит id пользователя, которому эти данные принадлежат.
3. Data_path. Тип string. Путь к загруженным данным пользователя.
4. Data_name. Тип string. Имя загруженных данных пользователя.

Таблица «neuronet» содержит в себе список нейронных сетей, которые принадлежат пользователю и имеет следующие поля:

1. Id. Тип integer. Первичный ключ таблицы.
2. User_id. Тип integer. Внешний ключ таблицы. Содержит id пользователя, которому эти искусственные нейронные сети принадлежат.
3. Netname. Тип string. Имя нейронной сети, которое задал пользователь.
4. Json_path. Тип string. Путь к json файлу, который содержит в себе модель построения искусственной нейронной сети, ее слои, их связи, и тип слоев.
5. H5_path. Тип string. Пусть к h5 файлу, которые содержит в себе веса искусственной нейронной сети.

Таблица «layers» содержит в себе параметры слоев искусственной нейронной сети и содержит в себе следующие поля.

1. Id. Тип integer. Первичный ключ таблицы.

2. Net_id. Тип integer. Внешний ключ таблицы. Содержит id сети, которой эти слои принадлежат.
3. Type. Тип string. Тип слоя искусственной нейронной сети.
4. Units. Тип integer. Количество нейронов в данном слое.
5. Activation. Тип string. Функция активации, которую использует данный слой.
6. Use_bias. Тип string. Данный параметр определяет следует использовать смещение или нет.
7. Kernel_initializer. Тип string. Какой тип инициализации ядра использовать.
8. Bias_initializer. Тип string. Какой тип инициализации смещения использовать.
9. Kernel_initializer. Тип string. Какой тип ограничения ядра использовать.
10. Bias_initializer. Тип string. Какой тип ограничения смещения использовать.
11. Conv_width. Тип integer. Ширина карты признаков.
12. Conv_height. Тип integer. Высота карты признаков.
13. Padding. Тип string. Параметр, который определяет стоит ли использовать дополнение вывода.

Таблица «classes» содержит в себе информацию о классах данных, на которых обучалась искусственная нейронная сеть, или которые были загружены пользователем. Данная таблица содержит в себе следующие поля:

1. Id. Тип integer. Первичный ключ таблицы.
2. Class_name. Тип string. Название класса.
3. Net_id. Тип integer. Внешний ключ таблицы. Содержит id сети, которой эти классы принадлежат.
4. Data_id. Тип integer. Внешний ключ таблицы. Содержит id загруженных данных, которым эти классы принадлежат.

4. РЕАЛИЗАЦИЯ

Диаграммы активности [9]

Авторизация пользователя (рисунок 22).

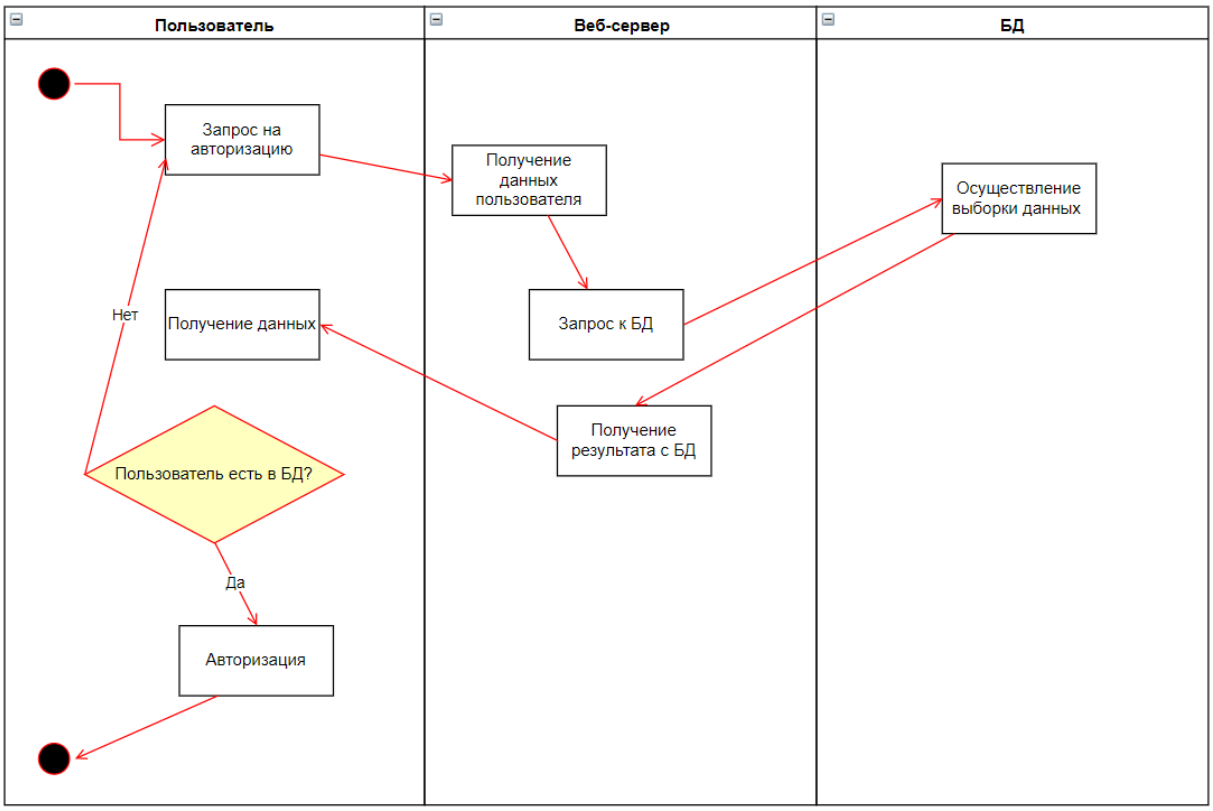

Рисунок 22 – Диаграмма активности авторизации

Пользователь вводит свои данные.

Сервер получает данные пользователя и отправляет запрос в базу данных, для проверки, существует ли данный пользователь в системе. Сервер получает ответ с базы данных и отправляет его обратно на сайт.

В результате, в зависимости от ответа, пользователь будет успешно авторизирован, либо система скажет ему, что данного пользователя не существует в системе.

Регистрация пользователя (рисунок 23).

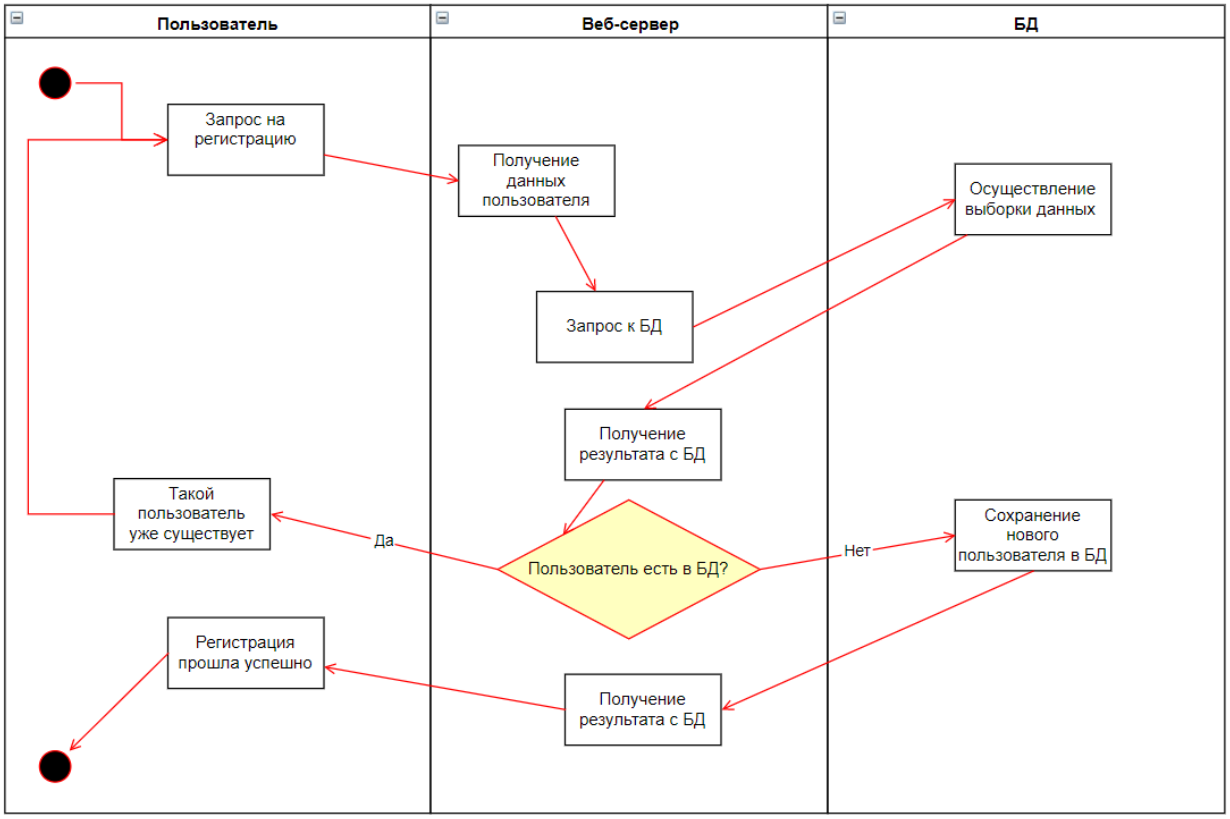

Рисунок 23 – Диаграмма активности регистрации

Пользователь вводит свои данные.

Сервер получает данные пользователя и отправляет запрос в базу данных, для проверки, существует ли данный пользователь в системе. Сервер получает ответ с базы данных.

Далее идет проверка, зарегистрирован данный пользователь в системе или нет. Если пользователь не существует в системе, то сервер отправляет запрос в базу данных на сохранение данного пользователя, если же он уже существует в базе данных, то сервер отправляет ответ на сайт и пользователь получает уведомление, что данный пользователь уже существует.

Работа с искусственными нейронными сетями.

Создание, просмотр внешнего вида и обучение на готовых наборах данных искусственной нейронной сети (рисунок 24).

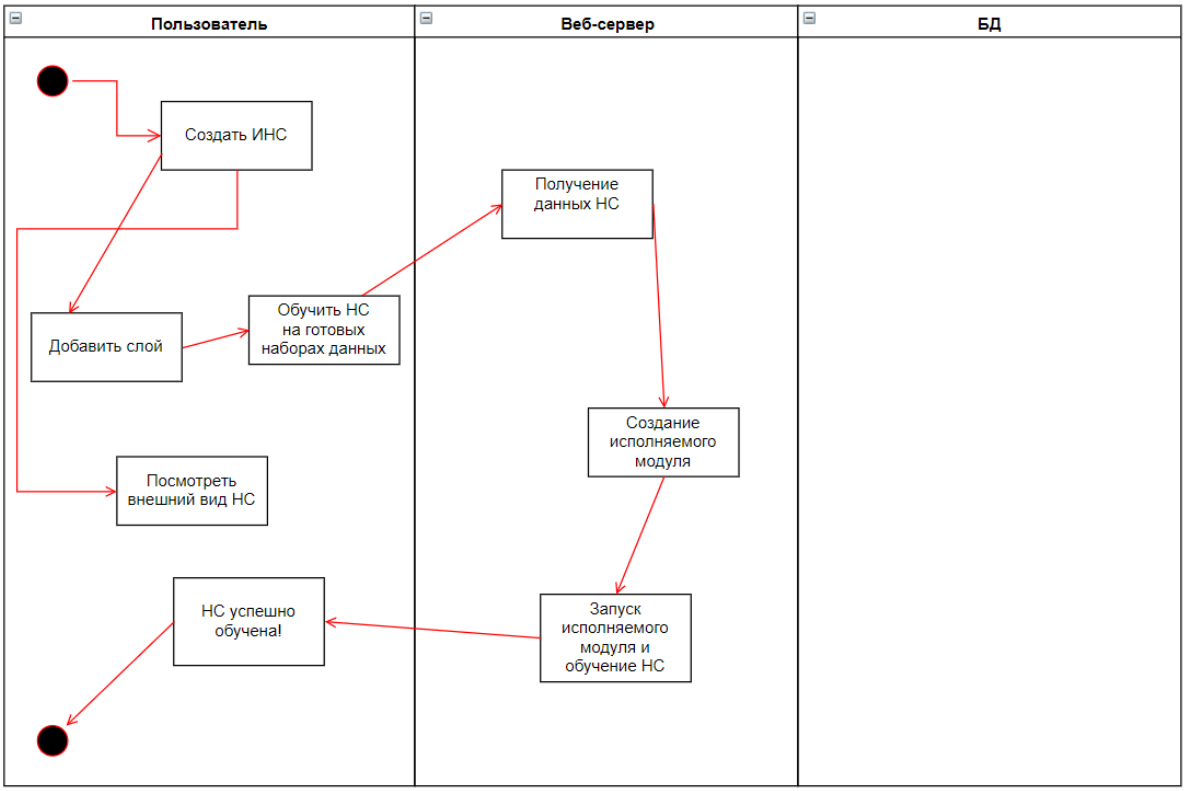

Рисунок 24 – Диаграмма активности создания, просмотра внешнего вида и обучение на готовых наборах данных искусственной нейронной сети

Пользователь может создать искусственную нейронную сеть. Далее он может добавлять для нее слои, или посмотреть как выглядит сеть, ее нейроны и связи между ними.

Далее пользователь может обучить данную сеть на готовых наборах данных. При выборе этого действия на сервер отсылается информация об искусственной нейронной сети, ее слоях.

После получения данных сервер создает модуль, при запуске которого произойдет обучение искусственной нейронной сети пользователя. После создания модуля, сервер производит его запуск, ждет ответа от модуля, когда обучится сеть и затем отправляет ответ на сайт.

При получении ответа от сервера пользователю выводится сообщение, что нейронная сеть успешно обучена.

Сохранение искусственной нейронной сети, а так же получение списка всех сохраненных нейронных сетей пользователя (рисунок 25).

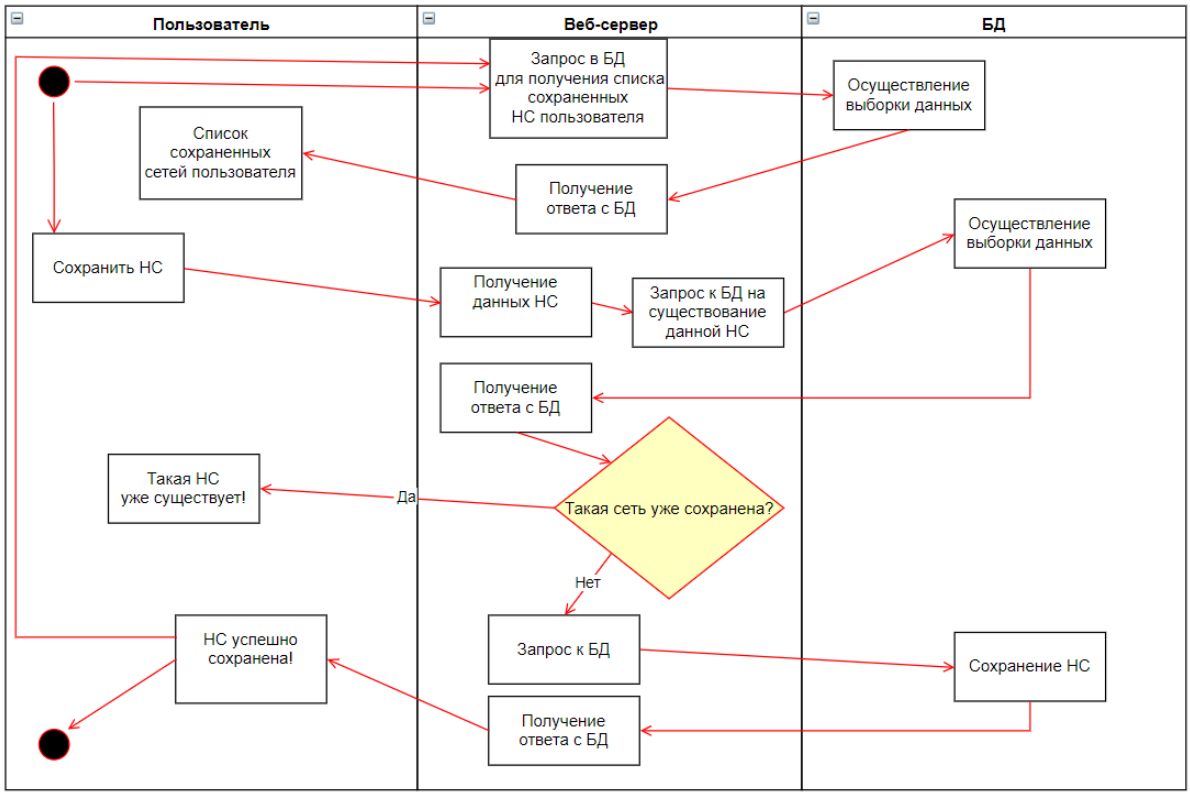

Рисунок 25 – Диаграмма получения списка сохраненных искусственных нейронных сетей и их сохранение

При заходе на сайт, сервер отправляет запрос в базу данных для получения списка сохраненных искусственных нейронных сетей пользователя. После получения списка, сервер передает ответ на сайт.

Пользователь может выбрать сохранение своей искусственной нейронной сети. После этого имя и параметры данной сети передаются на сервер. Сервер отправляет запрос в базу данных на проверку, сохранена ли данная сеть. Если оказывается, что данная искусственная нейронная сеть уже существует, то на сайт отправится соответствующее уведомление и пользователю будет сказано, что такая сеть уже существует.

Если данной сети нету в базе данных, то сервер отправляет запрос в базу данных на сохранение данной искусственной нейронной сети. Далее

сервер отправляет ответ на сайт, что сеть сохранена и пользователь получает уведомление, что данная сеть успешно сохранена.

После каждого успешного сохранения сети, сервер делает запрос в базу данных на получение нового списка сохраненных искусственный нейронных сетей пользователя и отправляет список на сайт.

Данный список затем обновляется и отображается на сайте вместе с новой сохраненныо сикусственной нейронной сетью.

Работа с загруженными данными.

Загрузка и сохранение данных пользователя (рисунок 26).

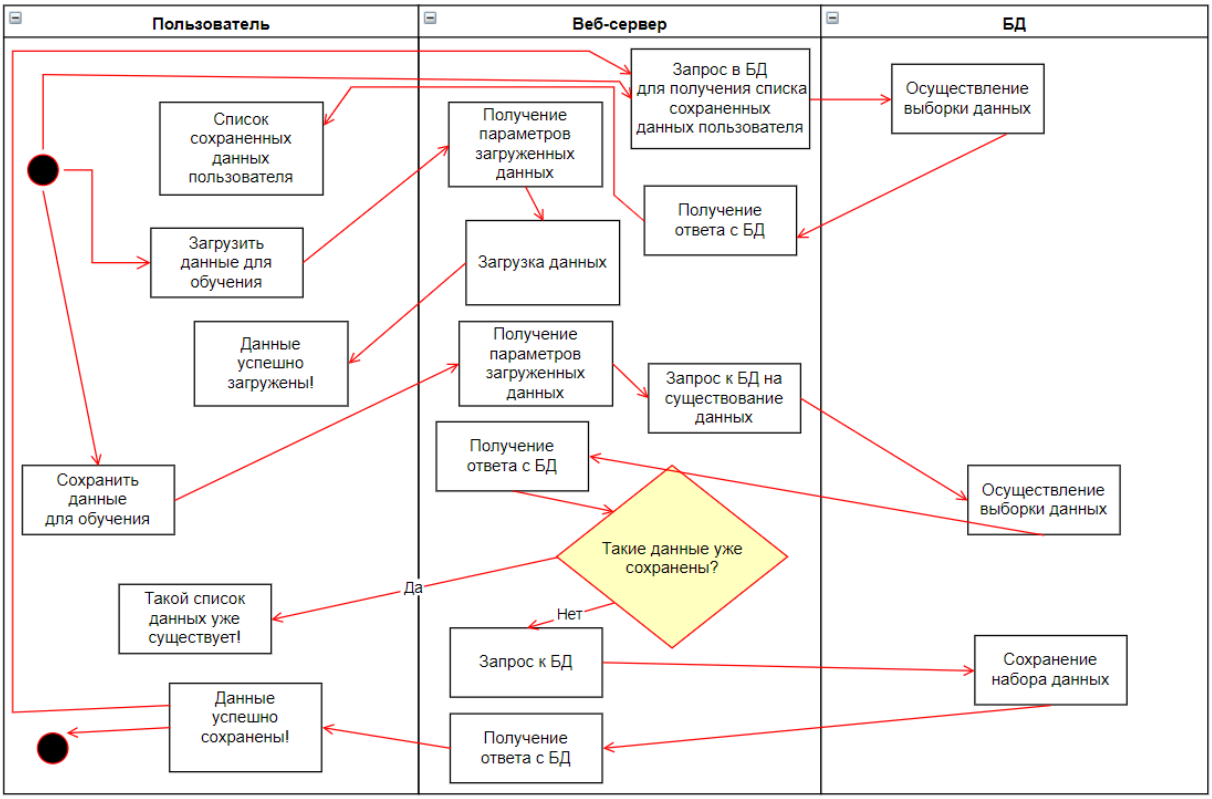

Рисунок 26 – Загрузка и сохранение данных пользователя

Пользователь может загрузить данные для обучения. После того как он выбрал данные, которые хочет загрузить, сервер получает эти данные, загружает их и передает уведомление на сайт, что данные успешно загружены.

При заходе на сайт, сервер отправляет запрос в базу данных для получения списка сохраненных данных пользователя. После получения списка, сервер передает ответ на сайт.

Пользователь может выбрать сохранение своих данных. После этого имя и параметры данных передаются на сервер. Сервер отправляет запрос в базу данных на проверку, сохранены ли выбранные данные.

Если оказывается, что выбранные данные уже существует, то на сайт отправится соответствующее уведомление и пользователю будет сказано, что такой набор данных уже существует.

Если выбранного набора данных нету в базе данных, то сервер отправляет запрос в базу данных на сохранение выбранного набора данных. Далее сервер отправляет ответ на сайт, что данные сохранены и пользователь получает уведомление, что выбранные данные успешно сохранены.

После каждого успешного сохранения данных, сервер делает запрос в базу данных на получение нового списка сохраненных данных пользователя и отправляет список на сайт.

Данный список затем обновляется и отображается на сайте вместе с новым сохраненным набором данных.

Обучение искусственной нейронной сети на загруженных данных пользователя, а так же ее тестирование (рисунок 27).

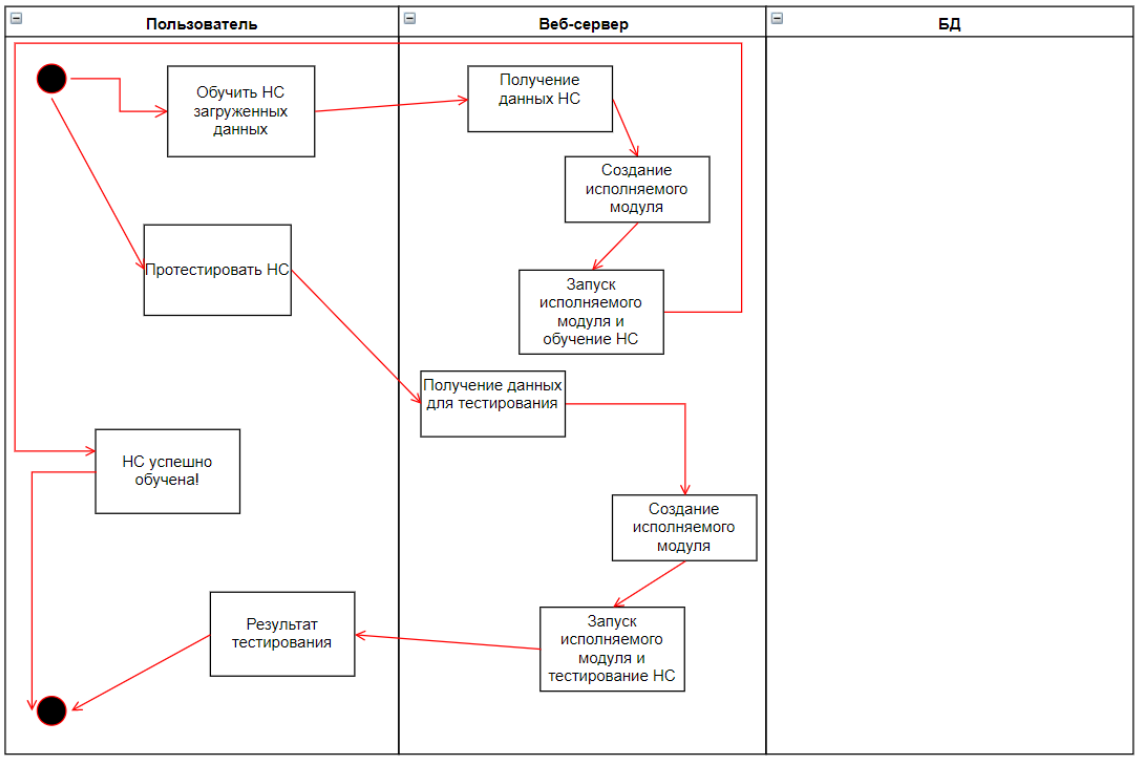

Рисунок 27 – Обучение искусственной нейронной сети на загруженных данных пользователя, а так же ее тестирования

Далее пользователь может обучить свою искусственную нейронную сеть на своем собственном наборе данных. При выборе этого действия на сервер отсылается информация об искусственной нейронной сети, ее слоях, а так же набора данных, который пользователь загрузил.

После получения данных сервер создает модуль, при запуске которого произойдет обучение искусственной нейронной сети пользователя. После создания модуля, сервер производит его запуск, ждет ответа от модуля, когда обучится сеть и затем отправляет ответ на сайт.

При получении ответа от сервера пользователю выводится сообщение, что нейронная сеть успешно обучена.

При тестировании сети пользователь выбирает данные для теста. После выбора данных на сервер отправляются данные для тестирования. Затем сервер генерирует модель для тестирования искусственной нейронной сети

пользователя. После создания модуля, сервер производит его запуск, ждет ответа от модуля, какой в итоге получился результат тестирования.

Затем сервер отправляет ответ на сайт и пользователю выводится сообщение с результатами тестирования его искусственной нейронной сети.

4.1. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСОВ

Веб-приложение было написано с помощью React (JavaScript) и Flask (Python). Используемая база данных MySQL. Общее количество строк кода приблизительно равно 4000.

Главная страница сайта.

Попадая на сайт, пользователь видит окно с авторизацией и регистрацией (рисунок 28).

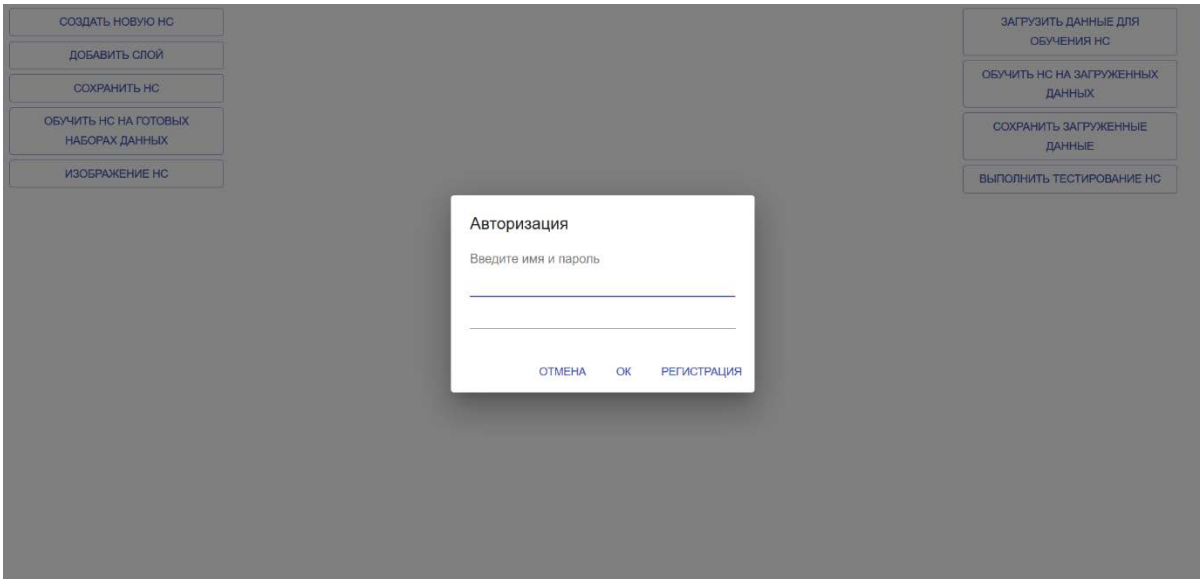

Рисунок 28 – Главная страница сайта, окно регистрации и авторизации

Далее, пройдя регистрацию или авторизацию пользователь попадает на сайт (рисунок 29), где видит различные опции для работы с искусственными нейронными сетями, список сохраненных сетей.

Так же пользователь видит область для работы с собственными наборами данных для решения задачи классификации и список своих сохраненных данных.

Рисунок 29 – Главная страница сайта, после прохождения авторизации или регистрации.

Пример кода главного элемента сайта (рисунок 30), которые содержит в себе другие элементы, такие как левое меню, верхнее меню, правое меню.

Эти меню, в свою очередь, содержат в себе различные опции для работы с искусственными нейронными сетями, их слоями и данными пользователя.

Однако основные параметры, необходимые для работы с данными и искусственными нейронными сетями, такие как:

- Список сетей.
- Слои сетей.
- Путь к данный пользователя.
- Классы данных пользователя.

Содержатся в главном компоненте.

```
<div>
    <Grid container direction="row">
        <Grid item style={{width:"20%"}} >
            <Grid>
                <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.newNet}>
                    Создать новую НС
                </Button>
                <LeftMenu _neuronet={this.state.neuronet} _addLayer={this.addLayer}/>
                <SaveNet _NetSave={this.NetSave} _NeuralSave={this.NeuralSave}/>
                <LearnNet _NetLearn={this.NetLearn}/>
                <LeftBottom layers={this.state.layer}/>
            </Grid>
            <Grid>
                <NeuralList _neuronet={this.state.neuronet}/>
            </Grid>
        </Grid>
        <Grid item style={{width:"60%"}} >
            <Grid container direction="column" >
                <Grid item style={{height:"10%"}} >
                    <TopMenu layers={this.state.layer} _setLayerParams={this.setLayerParams}/>
                </Grid>
                <Grid item style={{height:"70%"}} >
                    <AppWindow />
                </Grid>
            </Grid>
        </Grid>
        <Grid item style={{width:"20%"}}>
            <LoadUserData _user_id={this.state.user_id} _DataSave={this.DataSave} _NetLearnData={this.NetLearnData}/>
            <Grid>
                <DataList _dataLoads={this.state.dataLoads}/>
            </Grid>
        </Grid>
        <ModalLogin openWindow={this.state.ModalLogin}  _setUserId={this.setUserId} openSignIn={this._openSignIn}/>
        <ModalSignIn openWindow={this.state.ModalSignIn}  _getUserId={this.getUserId} />
    </Grid>
</div>
```

Рисунок 30 – Реализация главной страницы

Левое меню (рисунок 31) содержит в себе опции для работы с искусственными нейронными сетями.

Можно создать новую сеть, добавить слой, обучить ее, посмотреть внешний вид сети, а так же посмотреть список всех сохраненных искусственных нейронных сетей пользователя.

```
class LeftMenu extends React.Component {
  state = {
    openImage: false,
    open: false,
    selectedValue: neurons[1],
  };

  handleClickOpen = () => {
    this.setState({ open: true, });
  };

  handleClose = value => {
    this.setState({ selectedValue: value, open: false });
        console.log("value--", this.state.selectedValue, neurons, value);
                    // (value, untis, activation, use_bias, kernel_initializer, bias_initializer, kernel_constraint, bias_constraint)
    this.props._addLayer(value, '10' , 'relu'    , 'False' , 'None'            , 'None'          , 'None'           , 'None');
  };

  render() {

    const {_neuronet} = this.props

    return (
      <div>
        <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.handleClickOpen}>
          Добавить слой
        </Button>

        <ChooseNeuronDialogWrapped
          selectedValue={this.state.selectedValue}
          open={this.state.open}
          onClose={this.handleClose}
        />

      </div>
    );
  }
}
```

Рисунок 31 – Реализация левого меню

Верхнее меню (рисунок 33) содержит в себе список всех слоев выбранной искусственной нейронной сети.

Данные слои представляются в виде горизонтального списка иконок (рисунок 32), при нажатии на которые появляется возможность настраивать параметры данного слоя.

Рисунок 32 – Список слоев сети

```
class TopMenu extends Component {

  state = {
    layer: 0,
    layerProps: {},
    openLayerProps: false,
  };

  handleClickOpen = (item, i) => {
    this.setState({ openLayerProps: true, });
    this.setState({ layer: i, });
    this.setState({ layerProps: item, });
    console.log('i--', i);
  };


  render() {

  const { layers } = this.props;
  const { classes } = this.props;

  //console.log('layers--', layers);

  const listLayers = layers.map((item, i) =>
            (<IconButton aria-label="Layer" onClick={()=>this.handleClickOpen(item, i)}>
                <Badge badgeContent={i+1} color="primary" classes={{ badge: classes.badge }}>
                    {icons[item.icon]}
                </Badge>
            </IconButton>)
        );

  return (
    <div>

        {listLayers}

        <ModalSetLayerProps openWindow={this.state.openLayerProps}
                            closeWindow={()=>this.setState({ openLayerProps: false, })}
                            __setLayerParams={this.props._setLayerParams}
                            item={this.state.layer}
                            layerProps={this.state.layerProps} />

    </div>
    );
  }
}
```

Рисунок 33 – Реализация верхнего меню

Правое меню (рисунок 34) содержит опции для работы с наборами данных пользователя.

Данные можно загружать, сохранять, обучать на них собственную искусственную нейронную сеть. Так же в правом меню присутствует опция для тестирования искусственной нейронной сети пользователя.

В добавок ко всему, правое меню содержит в себе список всех сохраненных наборов данных пользователя.

```
return (
    <div>
        <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.handleDialogOpenLoad}>
            Загрузить данные для обучения НС
        </Button>
        <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.handleDialogOpenLearn}>
            Обучить НС на загруженных данных
        </Button>
        <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.openSaveData}>
            Сохранить загруженные данные
        </Button>
        <Button variant="outlined" color="primary" style={{margin: "1%", width:"90%"}} onClick={this.handleDialogOpenTest}>
            Выполнить тестирование НС
        </Button>
        <Dialog
            open={this.state.openWindow}
            onEnter={this.handleOpen}
            onClose={this.handleCloseCancel}
            aria-labelledby="form-dialog-title">
            <DialogTitle id="form-dialog-title">Параметры загружаемых данных</DialogTitle>
            <DialogContent>
                <DialogContentText>
                    Введите название данных.
                </DialogContentText>
                <TextField
                    autoFocus
                    margin="dense"
                    id="units"
                    type="text"
                    fullWidth
                    value={this.state.dataName}
                    onChange={e => this.setState({ dataName: e.target.value })}
                />
                <DialogContentText>
                    Выберите количество классов обучения.
                </DialogContentText>
                <label>
                    <select
                        value={this.state.classes}
                        onChange={e => this.setState({classes: e.target.value})}>
                        <option value="2">2</option>
                        <option value="3">3</option>
                        <option value="4">4</option>
                        <option value="5">5</option>
                        <option value="6">6</option>
                        <option value="7">7</option>
                        <option value="8">8</option>
                        <option value="9">9</option>
                        <option value="10">10</option>
                    </select>
                </label>
                <DialogContentText>
                    Какой процент данных будет использоваться для обучения?.
                </DialogContentText>
                <label>
```

Рисунок 34 – Реализация правого меню

Авторизация и регистрация (рисунок 35).

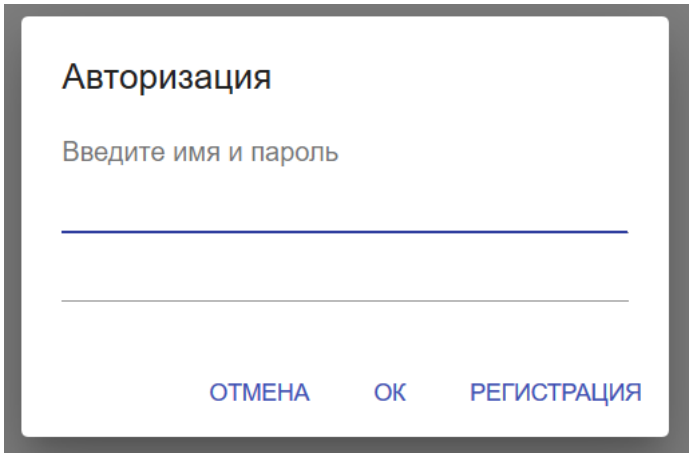

Рисунок 35 – Окно авторизации и регистрации

При авторизации отправляется запрос на сервер (рисунок 36), который в свою очередь отправляет запрос к базе данных и получает ответ, есть ли данный пользователь в системе.

```
@app.route("/login")
def login():
    query="SELECT id FROM `users` WHERE `name`=%s AND `password`=%s"
    password = request.args['password']
    username = request.args['user']
    args = (username, password)
    try:
        cursor = CONN.cursor()
        cursor.execute(query, args)
        row = cursor.fetchone()
#        while row is not None:
#            row = cursor.fetchone()
#            return(row)

    except Error as e:
        print(e)

    finally:
#        cursor.close()
        if row is None:
           return('Error')
        else:
           return str(row[0])
```

Рисунок 36 – Реализация авторизации

При регистрации отправляется запрос на сервер (рисунок 37), который в свою очередь отправляет запрос к базе данных и получает ответ, есть ли данный пользователь в системе. Если пользователь присутствует, то сервер отправляет ответ на сайт и регистрация не проходит. Если такого пользователя нет в системе, то сервер отправляет запрос к базе данных и сохраняет там пользователя, а на сайт передает информацию об успешной регистрации пользователя.

```
@app.route("/adduser")
def adduser():
    query = "INSERT INTO `users` (`name`, `password`) VALUES(%s, %s)"
    password = request.args['password']
    username = request.args['user']
    args = (username, password)
    user_id = 0

    try:
        cursor = CONN.cursor()
        cursor.execute(query, args)

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
            user_id = cursor.lastrowid
        else:
            print('last insert id not found')

        CONN.commit()
    except Error as error:
        print(error)

    finally:
#        cursor.close()
        return str(user_id)
```

Рисунок 37 – Реализация регистрации

Работа с искусственными нейронными сетями.

Для работы с нейронными сетями было решено взять язык программирования Python, так как данный язык обладает удобным и

понятным кодом и встроенными библиотеками для работы с нейронными сетями, такими как Keras, TensorFlow, Theano.

Добавление слоя (рисунок 38).

При добавлении нового слоя предлагается выбрать какой тип слоя использовать. Доступны два типа слоя:

1) Полносвязный.
2) Сверточный.

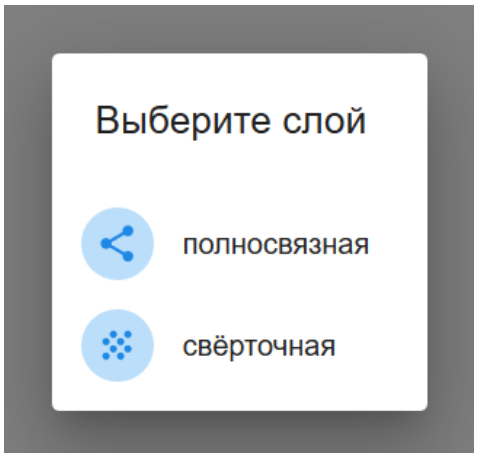

Рисунок 38 – Экран выбора типа слоя

Изменение параметров слоя (рисунок 39).

При изменения параметров слоя, для изменения доступны следующие параметры слоя:

1) Units. Количество нейронов в данном слое. Доступно полносвязной и сверточной сети. Основной параметр.
2) Activation. Функция активации, которую использует данный слой. Доступно полносвязной и сверточной сети. Основной параметр.
3) Use_bias. Данный параметр определяет следует использовать смещение или нет. Доступно полносвязной и сверточной сети. Дополнительный параметр.
4) Kernel_initializer. Какой тип инициализации ядра использовать. Доступно полносвязной и сверточной сети. Дополнительный параметр.

5) Bias_initializer. Какой тип инициализации смещения использовать. Доступно полносвязной и сверточной сети. Дополнительный параметр.
6) Kernel_initializer. Какой тип ограничения ядра использовать. Доступно полносвязной и сверточной сети. Дополнительный параметр.
7) Bias_initializer. Какой тип ограничения смещения использовать. Доступно полносвязной и сверточной сети. Дополнительный параметр.
8) Conv_width. Ширина карты признаков. Доступно только сверточной сети. Основной параметр.
9) Conv_height. Высота карты признаков. Доступно только сверточной сети. Основной параметр.
10) Padding. Параметр, который определяет стоит ли использовать дополнение вывода. Доступно только сверточной сети. Дополнительный параметр.

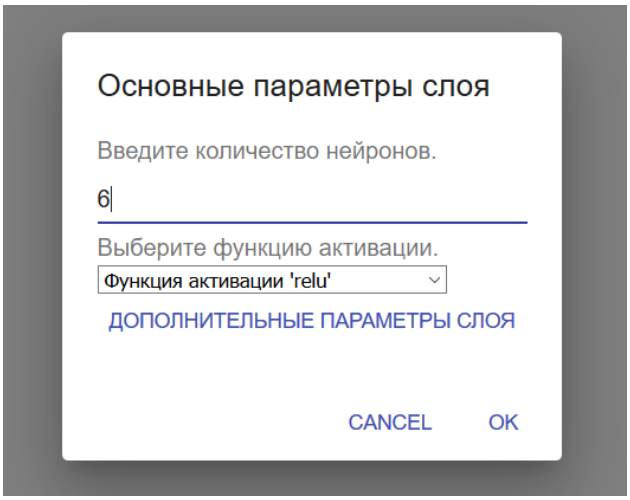

Рисунок 39 – Экран настройки параметров слоя

Обучение искусственной нейронной сети на готовых наборах данных.

На данный момент сеть можно обучить на наборе данных MNIST для распознавания рукописных цифр.

При выборе этого действия на сервер отсылается информация об искусственной нейронной сети, ее слоях.

После получения данных сервер создает модуль (рисунок 40), при запуске которого произойдет обучение искусственной нейронной сети пользователя. После создания модуля, сервер производит его запуск, ждет ответа от модуля, когда обучится сеть и затем отправляет ответ на сайт.

```
@app.route("/netLearn", methods=["GET", "POST"])
def netLearn():

    global mas

    lengthStr = request.args['length']
    iStr = request.args['i']

    units = int(request.args['units'])
    activation = request.args['activation']
    use_bias = request.args['use_bias']
    kernel_initializer = request.args['kernel_initializer']
    bias_initializer = request.args['bias_initializer']
    kernel_constraint = request.args['kernel_constraint']
    bias_constraint = request.args['bias_constraint']

    i = int(iStr)
    length = int(lengthStr)

    if (i==0) :
      mas = []

    d = dict()

    if (activation != "None"):
      d['activation']=activation
    d['use_bias'] = use_bias
    if (kernel_initializer != "None"):
      d['kernel_initializer'] = kernel_initializer
    if (bias_initializer != "None"):
      d['bias_initializer'] = bias_initializer
    if (kernel_constraint != "None"):
      d['kernel_constraint'] = kernel_constraint
    if (bias_constraint != "None"):
      d['bias_constraint'] = bias_constraint

    mas.append([])  # заносим пустую строку в матрицу
    # записываем в строку под номером i параметры слоя
    mas[i].append(request.args['type'])
    mas[i].append(int(request.args['units']))
    mas[i].append(d)
```

Рисунок 40 – Создание модуля для обучения искусственной нейронной сети

Сохранение искусственной нейронной сети (рисунок 41).

Пользователь может выбрать сохранение своей искусственной нейронной сети. После этого имя и параметры данной сети передаются на сервер. Сервер отправляет запрос в базу данных на проверку, сохранена ли данная сеть. Если оказывается, что данная искусственная нейронная сеть уже

существует, то на сайт отправится соответствующее уведомление и пользователю будет сказано, что такая сеть уже существует.

Если данной сети нету в базе данных, то сервер отправляет запрос в базу данных на сохранение данной искусственной нейронной сети. Далее сервер отправляет ответ на сайт, что сеть сохранена и пользователь получает уведомление, что данная сеть успешно сохранена.

```
@app.route("/neuralSave", methods=["GET", "POST"])
def neuralSave():

  import sys
  sys.path.append('C:/Work/Python/Keras/')
  import NeuralSave

  global mas

  user_id = request.args['user_id']
  netname = request.args['netname']

  print("neuralSavePython")

  result = NeuralSave.modelSave(user_id,netname,mas)

  CONN = mysql.connector.connect(host='localhost', database='tensor', user='root', password='')
  # создаем запрос в базу данных
  query = "INSERT INTO `layers` (`net_id`, `type`, `units`, `activation`,`use_bias`,`kernel_initializer`,`bias_initializer`,`kernel_constraint`,`bias_constraint`) VALUES(%s, %
  args_net_id = result

  print("neuralSavePython")

  for i in range(len(mas)):
    args_type = mas[i][0]
    args_unints = mas[i][1]
    args_activation = mas[i][2]['activation']
    args_use_bias = mas[i][2]['use_bias']
    if ('kernel_initializer' in mas[i][2]):
      args_kernel_initializer = mas[i][2]['kernel_initializer']
    else:
      args_kernel_initializer = 'None'
    if ('bias_initializer' in mas[i][2]):
      args_bias_initializer = mas[i][2]['bias_initializer']
    else:
      args_bias_initializer = 'None'
    if ('kernel_constraint' in mas[i][2]):
      args_kernel_constraint = mas[i][2]['kernel_constraint']
    else:
      args_kernel_constraint = 'None'
    if ('bias_constraint' in mas[i][2]):
      args_bias_constraint = mas[i][2]['bias_constraint']
    else:
      args_bias_constraint = 'None'
    args = (args_net_id, args_type, args_unints, args_activation, args_use_bias, args_kernel_initializer, args_bias_initializer, args_kernel_constraint, args_bias_constraint)
    cursor = CONN.cursor()
    cursor.execute(query, args)
```

Рисунок 41 – Сохранение искусственной нейронной сети

Список сохраненных искусственных нейронных сетей пользователя (рисунок 42).

Сервер отправляет запрос в базу данных (рисунок 43) для получения списка сохраненных данных пользователя. После получения списка, сервер передает ответ на сайт.

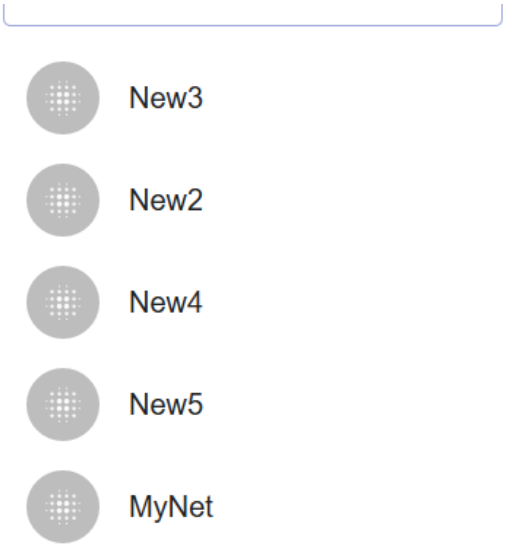

Рисунок 42 – Список сохраненных искусственных нейронных сетей пользователя

```
@app.route("/listnet")
def listnet():

    user_id = request.args['user_id']

    CONN = mysql.connector.connect(host='localhost', database='tensor', user='root', password='')
    query = "SELECT id, netname FROM `neuronet` WHERE `user_id` = %s" % user_id
    args = (user_id)

    print ('1')

    cursor = CONN.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()

    print (rows)
```

Рисунок 43 – Получение списка сохраненных искусственных нейронных сетей пользователя

Изображение искусственной нейронной сети (рисунок 44).

Показывает как выглядит сеть, ее нейроны и связи между ними.

Рисунок 44 – Внешний вид искусственной нейронной сети

Получение списка искусственных нейронных сетей (рисунок 45).

Сервер отправляет запрос в базу данных для получения списка сохраненных искусственных нейронных сетей пользователя. После получения списка, сервер передает ответ на сайт.

```
@app.route("/listnet")
def listnet():

    user_id = request.args['user_id']

    CONN = mysql.connector.connect(host='localhost', database='tensor', user='root', password='')
    query = "SELECT id, netname FROM `neuronet` WHERE `user_id` = %s" % user_id
    args = (user_id)

    print ('1')

    cursor = CONN.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()

    print (rows)

#    print('Total Row(s):', cursor.rowcount)
#    for row in rows:
#      print(row)

#    cursor.close()
    if rows is None:
      return('Error')
    else:
      return jsonify(rows)
```

Рисунок 45 – Получения списка искусственный нейронных сетей

Работа с наборами данных пользователя.

Загрузить данные для обучения искусственной нейронной сети (рисунок 46).

При загрузке собственных данных пользователь может выбрать следующие параметры:

1) Имя данных.
2) Количество классов изображений.
3) Какой процент данных будет использоваться для обучения.

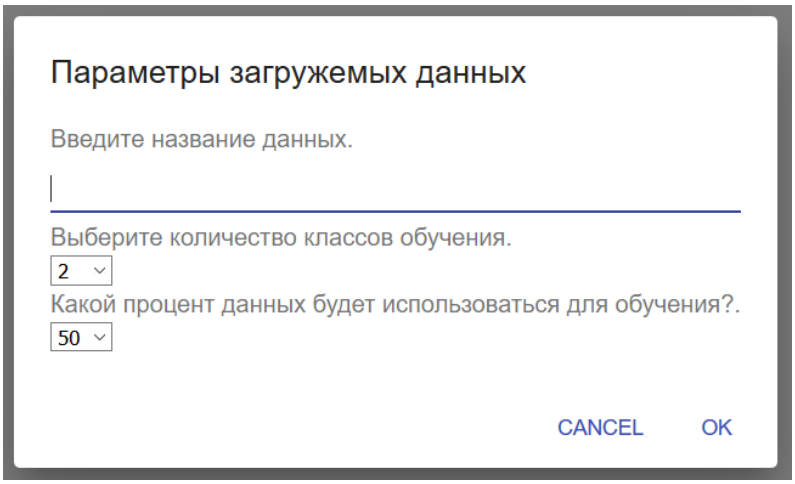

Рисунок 46 – Экран загрузки данных

Далее, в зависимости от количества выбранных классов, пользователю будет предложено назвать класс (например класс Cats, который означает, что на изображении присутствуют кошки) (рисунок 47) загружаемых изображений и после этого, пользователь может загрузить изображения в этот класс (например 50 изображений, 50% из которых буду использованы для обучения) (рисунок 48). Затем переходит к следующему классу и так далее, пока не будут загружены изображения для всех классов.

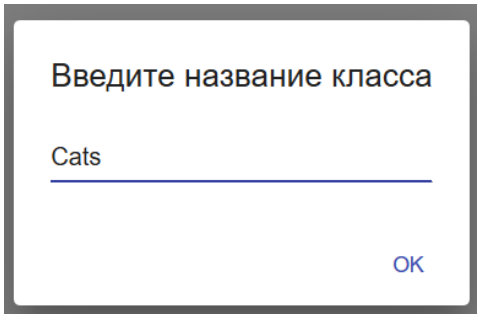

Рисунок 47 – Экран выбора названия класса

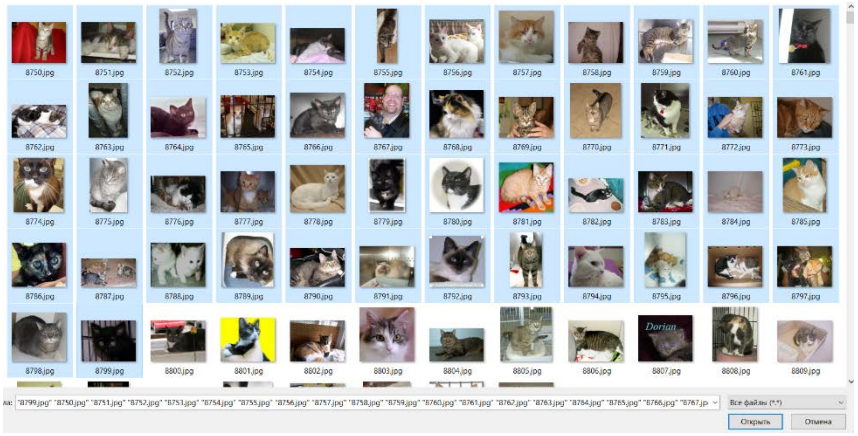

Рисунок 48 – Выбор изображений для загрузки в класс

Обучить искусственную нейронную сеть на загруженных данных (рисунок 49).

Если пользователь хочет обучить свою сеть на созданном им наборе данных, то он может выбрать пункт «Обучить искусственную нейронную сеть на загруженных данных», после чего ему будет предложено указать следующие параметры:

1) Ввести размер выборки, которая будет использоваться для обучения.
2) Ввести количество эпох обучения.
3) Ввести ширину изображений на которой будет учиться искусственная нейронная сеть.
4) Ввести высоту изображения на которой будет обучаться сеть.

Далее программа начнет обучение искусственной нейронной сети пользователя на его собственных наборах данных.

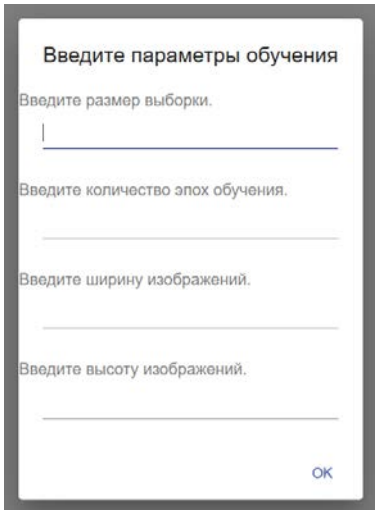

Рисунок 49 – Экран параметров обучения сети на наборе данных

Сохранение собственных наборов данных пользователя (рисунок 50).

Пользователь может выбрать сохранение своих данных. После этого имя и параметры данных передаются на сервер. Сервер отправляет запрос в БД на проверку, сохранены ли выбранные данные.

Если оказывается, что выбранные данные уже существует, то на сайт отправится соответствующее уведомление и пользователю будет сказано, что такой набор данных уже существует.

Если выбранного набора данных нету в БД, то сервер отправляет запрос в БД на сохранение выбранного набора данных. Далее сервер отправляет ответ на сайт, что данные сохранены и пользователь получает уведомление, что выбранные данные успешно сохранены.

Выбранные сохраненные данные заносятся в список сохраненных наборов данных пользователя, которые он видит в правом нижнем углу.

```
@app.route('/DataSave', methods=['POST','GET'])
def DataSave():

    global directiory

    user_id = request.args['user_id']
    dataName = request.args['dataName']

    print("DIRECTORY-----", directory)
    print("user_id-----", user_id)
    print("dataName-----", dataName)
    CONN = mysql.connector.connect(host='localhost', database='tensor', user='root', password='')
    query = "INSERT INTO `data` (`user_id`, `data_path`, `data_name`) VALUES(%s, %s, %s)"
    args = (user_id, directory, dataName)


    cursor = CONN.cursor()
    cursor.execute(query,args)
    CONN.commit()

    cursor.close()
```

Рисунок 50 – Сохранение набора данных пользователя

Тестирование искусственной нейронной сети (рисунок 51).

При тестировании сети пользователь выбирает данные для теста. После выбора данных на сервер отправляются данные для тестирования. Затем сервер генерирует модель для тестирования искусственной нейронной сети пользователя.

После создания модуля, сервер производит его запуск, ждет ответа от модуля, какой в итоге получился результат тестирования.

Затем сервер отправляет ответ на сайт и пользователю выводится сообщение с результатами тестирования его искусственной нейронной сети.

Результатом теста является определение к какому классу изображений принадлежит изображение, выбранное пользователем

```
# Загрузка файла для тестирования
@app.route('/uploadTest', methods=['POST','GET'])
def uploadTest():
    file = request.files['file']

    filename = file.filename
    global fileTestPath
    file.save(os.path.join(fileTestPath, filename))

    return filename
# Тестирование работы НС
@app.route('/uploadTestWork', methods=['POST','GET'])
def uploadTestWork():
    global fileTestPath
    image_width = request.args['image_width']
    image_heigth = request.args['image_height']
    user_id = request.args['user_id']

    import sys
    sys.path.append('C:/Work/Python/Keras/')
    import DataTest

    print("TESTING DataLearn!!!!")
    result = DataTest.userDataTest(fileTestPath,image_width,image_heigth,user_id)
    return result
```

Рисунок 51 – Тестирование искусственной нейронной сети пользователя

Список сохраненных данных пользователя (рисунок 52).

Сервер отправляет запрос в базу данных (рисунок 53) для получения списка сохраненных данных пользователя. После получения списка, сервер передает ответ на сайт.

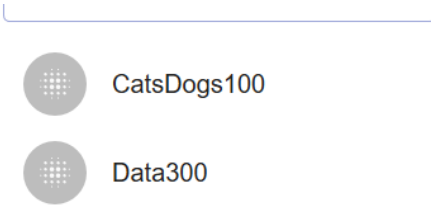

Рисунок 52 – Список сохраненных наборов данных пользователя

```
@app.route('/DataList', methods=['POST','GET'])
def DataList():
    user_id = request.args['user_id']

    CONN = mysql.connector.connect(host='localhost', database='tensor', user='root', password='')
    query = "SELECT data_path, data_name FROM `data` WHERE `user_id` = %s" % user_id
    args = (user_id)

    print ('1')

    cursor = CONN.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()

    print (rows)

#    print('Total Row(s):', cursor.rowcount)
#    for row in rows:
#      print(row)

#    cursor.close()
    if rows is None:
      return('Error')
    else:
      return jsonify(rows)
```

Рисунок 53 – Получение списка сохраненных наборов данных пользователя

5. ТЕСТИРОВАНИЕ

Далее будет представлена таблица (таблица 1) с функциональным тестированием системы.

Таблица 1 – Тестирование системы

Тест	Результат
Авторизация пользователя	Успешно пройден
Регистрация пользователя	Успешно пройден
Если, при регистрации, введенное имя пользователя уже есть в системе, уведомить об этом пользователя и не регистрировать его	Успешно пройден
Создание новой нейронной сети	Успешно пройден
Добавление слоя	Успешно пройден
Возможность выбрать тип слоя при добавлении	Успешно пройден
Возможность изменять параметры слоя	Успешно пройден
Каждый тип слоя имеет свои уникальные параметры, не доступные другому слою	Успешно пройден
Сохранение слоя	Успешно пройден
Если, при сохранении искусственной нейронной сети, введенное имя пользователя уже есть в системе, уведомить об этом пользователя и не сохранять сеть	Успешно пройден
Обучение нейронной сети на готовых наборах данных	Успешно пройден
Повторное обучение	Успешно пройден
Обучить нейронную есть несколько раз подряд	Успешно пройден
Изображение нейронной сети	Успешно пройден
Запретить обучение нейронной сети без слоев	Успешно пройден
Запретить обучение нейронной сети с 1 слоем	Успешно пройден
Запретить сохранение нейронной сети без слоев	Успешно пройден

Продолжение таблицы 1

Запретить сохранение нейронной сети с 1 слоем	Успешно пройден
Изображение нейронной сети корректно отображает вид этой сети	Успешно пройден
Список сохраненных нейронных сетей выводится корректно	Успешно пройден
После авторизации пользователя выводится список его сохраненных искусственных нейронных сетей	Успешно пройден
При сохранении нейронной сети, она тут же добавляется в список сохраненных сетей	Успешно пройден
При нажатии на значок сохраненной нейронной сети, она выгружается на экран	Успешно пройден
Слои сети отображаются в верхней части экрана	Успешно пройден
Значок слоя сети корректно отображается	Успешно пройден
Значок слоя зависит от типа слоя	Успешно пройден
При нажатии на значок слоя выводится диалоговое окно с настройкой параметров этого слоя	Успешно пройден
При загрузке данных появляется диалоговое окно с настройкой параметров загружаемых данных	Успешно пройден
Загрузка одновременно нескольких изображений	Успешно пройден
Загрузка одновременно несколько сотен изображений	Успешно пройден
В зависимости от количества классов, происходит загрузка нескольких типов изображений	Успешно пройден
При обучении нейронной сети на загруженных данных появляется диалоговое окно с настройкой параметров обучения	Успешно пройден
Возможность обучить два раза подряд	Успешно пройден

Продолжение таблицы 1

Обучение несколько раз подряд	Успешно пройден
Запретить обучение сети без слоев	Успешно пройден
Запретить обучение сети с 1 слоем	Успешно пройден
Запретить обучение сети, если количество нейронов выходного слоя сети не соответствует количеству классов изображений для обучения	Успешно пройден
Сохранение слоя	Успешно пройден
Если, при сохранении набора данных, введенное имя пользователя уже есть в системе, уведомить об этом пользователя и не сохранять сеть	Успешно пройден
Список сохраненных наборов данных выводится корректно	Успешно пройден
После авторизации пользователя выводится список его сохраненных наборов данных	Успешно пройден
При сохранении набора данных, он тут же добавляется в список сохраненных наборов данных	Успешно пройден
При нажатии на значок сохраненного набора данных, он загружается	Успешно пройден
При регулярном изменении параметров слоя или их количества, изображение этой сети каждый раз выводится корректно	Успешно пройден

6. ЗАКЛЮЧЕНИЕ

В ходе работы было создано веб-приложения для проектирования искусственных нейронных сетей.

Был произведен разбор аналогов, веб-сервисов по работе с искусственными нейронными сетями.

Так же при работе с веб-приложением для проектирования искусственных нейронных сетей были выполнены следующие задачи:

- Было спроектировано данное приложение;
- Была выполнена его реализация;
- Было выполнено тестирование работоспособности данного приложения.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Онлайн курс "Программирование глубоких нейронных сетей на Python" - https://www.asozykin.ru/courses/nnpython. Дата обращения: 25.03.2019.
2. Neural network playground - https://playground.tensorflow.org. Дата обращения: 02.04.2019.
3. ConventJsDemo - https://cs.stanford.edu/people/karpathy/convnetjs/. Дата обращения: 02.04.2019.
4. Machine Learning Playground - http://ml-playground.com/. Дата обращения: 02.04.2019.
5. Web-based 3D Neural Network Playground - https://hergott.github.io/neural-network-playground/. Дата обращения: 02.04.2019.
6. Nvidia Image Painting - https://www.nvidia.com/research/inpainting/. Дата обращения: 02.04.2019.
7. AutoDraw - https://www.autodraw.com. Дата обращения: 02.04.2019.
8. Free logo maker - https://looka.com. Дата обращения: 02.04.2019.
9. DrawIo - https://www.draw.io/. Дата обращения: 07.05.2019.
10. React, JavaScript-библиотека для создания пользовательских интерфейсов - https://ru.reactjs.org/. Дата обращения: 15.04.2019.
11. Основы ReactJs - https://learn.javascript.ru/screencast/react. Дата обращения: 15.04.2019.
12. Томас, М. Т. React в действии / Питер - Для профессионалов - Отдельное издание, 2019. – 368 с.
13. Keras Documentation - https://keras.io. Дата обращения: 05.05.2019.

14. Федоров, Д. Ю. Программирование на языке высокого уровня Python / Д. Ю. Федоров. — 2-е изд., перераб. и доп. — М.: Издательство Юрайт, 2019. – 161 с.
15. Марк Л. Изучаем Python, 3-е издание / Пер. с англ. – СПб.: Символ-Плюс, 2009. – 848 с.
16. Марк Л. Изучаем Python, 4-е издание / Пер. с англ. – Москва: Символ-Плюс, 2011. – 1272 с.
17. Flask web development, one drop at a time - http://flask.pocoo.org/. Дата обращения: 15.04.2019.
18. Гринберг, М. Разработка веб-приложений с использованием flask на языке Python / пер. с англ. А. Н. Киселева. – М.: ДМК, пресс 2016. – 272 с.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

Листинг А.1 – исходный код.

```
from flask import Flask, render_template, jsonify, request, session, Response
from flask import send_file
from datetime import datetime
import shutil

import time
import os
import sys
import json
import logging
import shutil
from werkzeug.utils import secure_filename
from flask_cors import CORS, cross_origin
import mysql.connector
from mysql.connector import Error

mas = [] # глобальная переменная которая содержит слои сети

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger('HELLO WORLD')

path =  os.path.abspath(os.path.dirname(sys.argv[0]))
conf_file = path + '/tensorshow.conf'

# директория пользователя с сохр данными обучения
directory = ""
# процент обучения
trainPer = 0;
# кол-во передаваемых файлов
filesLen = 0
# индекс
i = 0
# количество для тренировки
iTrain = 0
# количество для валидации
iVal = 0
# количество для тестов
iTest = 0
# название класса
className = ""
# путь, где лежит файл для теста
fileTestPath = ''

Position = 1

with open(conf_file, 'r') as f:
    data = json.loads(f.read())
directory = data["directory"]
os.chdir(directory)
curr_dir = directory
```

```
home_dir = directory
f.close()

app = Flask(__name__, static_folder="./static/dist", template_folder="../public")
app.secret_key = os.urandom(24)

# Переименовывание имя файла, на вход получает имя файла и на выход измененное имя файла
def replacer(str):
  return str.replace("%20", " ")

# Функция которая выводит текущую директорию и выводит список различных параметров, тип,
наименования и т.д
def lsDir(dir):
  d = {}
  m = []
  for entry in os.scandir(replacer(dir)):
    timestamp = os.stat(entry.name).st_mtime
    dt = datetime.fromtimestamp(timestamp)
    df = "{:%Y %b %d, %H:%M:%S}".format(dt)
    if entry.is_dir():
       d = { "checked": bool(), "type": "dir", "name": entry.name, "date": df }
    if entry.is_file():
       filesize  = os.stat(entry.name).st_size
       d = { "checked": bool(), "type": "file", "name": entry.name, "date": df, "size":
KMGTbytes(filesize) }
    m.append(d)
  return jsonify(m)

# Задает точку маршрута
@app.route("/")
def index():
    return render_template("index.html")

# Тестовая функция, маршрут
@app.route("/hello")
def hello():
    return "Hello World!"

# Вывод текущего каталога
@app.route("/ls", methods=["GET", "POST"])
def ls():
    return lsDir(curr_dir)

# Смена каталога
@app.route("/cd", methods=["GET", "POST"])
def cd():
    curr_dir = os.getcwd() + '/' + request.args['dir']
    os.chdir(replacer(curr_dir))
    return lsDir(curr_dir)

# Печать содержимого текущего каталога
@app.route("/pwd", methods=["GET", "POST"])
def pwd():
    return os.getcwd()

# Возвращение в домашний каталог
@app.route("/home", methods=["GET", "POST"])
def home():
    os.chdir(replacer(home_dir))
    return lsDir(home_dir)
```

```
# Вернуться из директории на уровень наверх
@app.route("/back", methods=["GET", "POST"])
def back():
    if len(os.getcwd()) > len(directory):
      os.chdir("..")
      curr_dir = os.getcwd()
      return lsDir(curr_dir)
    else:
      return lsDir(home_dir)

# Соединение с базой данных
@app.route("/connect")
def connect():
    """ Connect to MySQL database """
    try:
        global CONN
        CONN = mysql.connector.connect(host='localhost', database='tensor', user='root',
password='')
        if CONN.is_connected():
            print('Connected to MySQL database')
            return('OK')
    except Error as e:
        print(e)
    finally:
        print('Connection to %s closed' % CONN)

# Авторизация пользователя в системе. Получает на вход имя и пароль пользователя,
возвращает id пользователя
@app.route("/login")
def login():
    query="SELECT id FROM `users` WHERE `name`=%s AND `password`=%s"
    password = request.args['password']
    username = request.args['user']
    args = (username, password)
    try:
        cursor = CONN.cursor()
        cursor.execute(query, args)
        row = cursor.fetchone()

    except Error as e:
        print(e)

    finally:
        if row is None:
           return('Error')
        else:

           return str(row[0])

# Регистрация пользователя в системе. Получает на вход имя и пароль пользователя,
возвращает id пользователя.
@app.route("/adduser")
def adduser():
    query = "INSERT INTO `users` (`name`, `password`) VALUES(%s, %s)"
    password = request.args['password']
    username = request.args['user']
    args = (username, password)
    user_id = 0

    try:
```

```
        cursor = CONN.cursor()
        cursor.execute(query, args)

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
            user_id = cursor.lastrowid
        else:
            print('last insert id not found')

        CONN.commit()
    except Error as error:
        print(error)

    finally:
        return str(user_id)

# Создает директорию с данным пользователя. Получает на вход имя директории и id пользователя. На выходе дает уведомление, что директория успешно создана.
@app.route("/addDirPer")
def addDirPer():
    dataName = request.args["dataName"];
    trainPersent = request.args["trainPersent"]
    user_id = request.args["user_id"]

    global trainPer
    global i
    i = 1

    trainPer = int(trainPersent)
    print ("TRAIN PER-------", trainPer)

    create_dir_path = "D:/tmp/Keras"
    if (os.path.exists(create_dir_path)):
      print('This dir exists!')
    else:
      os.mkdir(create_dir_path)
    create_dir_path = "D:/tmp/Keras/" + user_id
    if (os.path.exists(create_dir_path)):
      print('This dir exists!')
    else:
      os.mkdir(create_dir_path)

    global directory
    directory = "D:/tmp/Keras/" + user_id + "/" + dataName
    os.mkdir(directory)
    os.mkdir(directory + "/train")
    os.mkdir(directory + "/test")
    os.mkdir(directory + "/val")
    print('DirCreated')

    return "Good"

# Создание директорий классов пользователя в общей директории данных. На вход получает имена классов, путь общей директории и id пользователя. На выходе дает уведомление, что директории для классов успешно созданы.
@app.route("/addDirDataClassName")
def addDirDataClassName():
    className1 = request.args["className"]
    dataName1 = request.args["dataName"]
    user_id1 = request.args["user_id"]
```

```
    global className
    className = className1

    global i
    i = 1

    global directory
    directory1 = ''
    directory1 = "D:/tmp/Keras/" + user_id1 + "/" + dataName1 + "/"
    os.mkdir(directory1 + "/train/" + className)
    os.mkdir(directory1 + "/test/" + className)
    os.mkdir(directory1 + "/val/" + className)
    print('DirClassNameCreated')

    return "Good"

# Получение количества загружаемых файлов и создание коэффициентов для распределения их
по собственным директориям. На входе получает количество загруженных файлов, на выходе
дает уведомление, что коэффициента распределения успешно созданы.
@app.route("/addFileLength")
def addFileLength():
    fileLength = request.args["fileLength"]

    global fileLen
    global iTrain
    global iVal
    global iTest
    fileLen = int(fileLength)
    print ("FILE LENGTH--------", fileLen)

    coeff = trainPer/100
    iTrain = int(fileLen*coeff)
    iVal = iTrain + int((fileLen-iTrain)/2)
    iTest = int((fileLen-iTrain)/2)

    return "Good"

# Загрузка файлов в директорию train. На вход получает файл для загрузки, на выходе
получает уведомление, что файл загружен
@app.route('/upload', methods=['POST','GET'])
def fileUpload():
    target=os.getcwd()
    if not os.path.isdir(target):
       os.mkdir(target)
    global directory
    global trainPer
    global fileLen
    global i
    global iTrain
    global iVal
    global Position
    global className
    Position = 1

    curr_dir = target
    file = request.files['file']
    filename = file.filename
    destination=directory + "/"
    logger.info("welcome to upload - %s" % filename)
```

```
    destination = destination + "train/" + className + "/"
    file.save(os.path.join(destination, filename))
    session['uploadFilePath']=destination
    data={"Pischaeff": filename}
    js = json.dumps(data)
    print('HERE                                    WORKING                                    I1111
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!', i)
    print('dest--------------------------' + destination)
    i+=1
    resp = Response(js, status=200, mimetype='application/json')
    return resp

# Создание пути файла для тестирования. Получает на вход id пользователя, на выход что создание прошло успешно.
@app.route('/uploadTestPath', methods=['POST','GET'])
def uploadTestPath():
    user_id = request.args['user_id']

    global fileTestPath
    fileTestPath = "D:/tmp/Keras/" + user_id + "/"

    return 'good'

# Загрузка файла для тестирования. На вход получает файл, на выход, что файл успешно загружен.
@app.route('/uploadTest', methods=['POST','GET'])
def uploadTest():
    file = request.files['file']

    filename = file.filename
    global fileTestPath
    file.save(os.path.join(fileTestPath, filename))

    return filename

# Тестирование работы НС. Получает на вход параметры размера изображения, id пользователя и список классов изображений. Запускает модуль тестирования. Возвращает ответ, результат тестирования, которые равен названию класса изображения.
@app.route('/uploadTestWork', methods=['POST','GET'])
def uploadTestWork():
    global fileTestPath
    classImg = reques.arsg[‘classes’]
    image_width = request.args['image_width']
    image_heigth = request.args['image_height']
    user_id = request.args['user_id']

    import sys
    sys.path.append('C:/Work/Python/Keras/')
    import DataTest

    print("TESTING DataLearn!!!!")
    result = DataTest.userDataTest(fileTestPath,image_width,image_heigth,user_id)
    return result

# Перенос файов в директории test и val. Перенос части файлов в директорию для тестирования и валидации. Использует ранее заданные коэффициенты переноса. Возвращает уведомление, что перенос прошел успешно.
@app.route('/fileMove', methods=['POST','GET'])
def fileMove():
```

```
    global directory
    global iTest
    global iTrain
    global Position
    global className
    destinationTrain = directory + "/" + "train/" + className + "/"
    destinationTest =  directory + "/" + "test/" + className + "/"
    destinationVal =  directory + "/" + "val/" + className + "/"
    print("destinationTrain",destinationTrain)
    print("destinationTest",destinationTest)
    print("destinationVal",destinationVal)

    files = os.listdir(destinationTrain)
    files.sort()

    print('iTrain', iTrain)
    print('iTest', iTest)

    for f in files:
      if (Position<=iTest):
        src = destinationTrain+f
        dst = destinationTest+f
        print("Position",Position)
        print('TEST')
        shutil.move(src, dst)
        Position += 1
      elif ((Position>iTest)and(Position<=iTrain)):
        src = destinationTrain+f
        dst = destinationVal+f
        print("Position",Position)
        print('VAL')
        shutil.move(src, dst)
        Position += 1
      elif (Position>iTrain):
        print("Position",Position)
        break
    return "Good"

# Сохранение собственно набора данных пользователя для обучения. На вход получает id пользователя, имя данных и список классов этих данных. На выходе возвращает уведомления, что сохранение прошло успешно.
@app.route('/DataSave', methods=['POST','GET'])
def DataSave():

    global directiory

    user_id = request.args['user_id']
    dataName = request.args['dataName']
    classImg = reques.arsg[‘classes’]

    print("DIRECTORY-----", directory)
    print("user_id-----", user_id)
    print("dataName-----", dataName)
    CONN  =  mysql.connector.connect(host='localhost',  database='tensor',  user='root', password='')
    query = "INSERT INTO `data` (`user_id`, `data_path`, `data_name`) VALUES(%s, %s, %s)"
    args = (user_id, directory, dataName)
```

```
    cursor = CONN.cursor()
    cursor.execute(query,args)
    CONN.commit()

    cursor.close()
    return "Good"

# Выводит список сохраненных данных пользователя. На вход получает id пользователя. На
выход список название его сохраненных данных.
@app.route('/DataList', methods=['POST','GET'])
def DataList():
    user_id = request.args['user_id']

    CONN  =  mysql.connector.connect(host='localhost',  database='tensor',  user='root',
password='')
    query = "SELECT data_path, data_name FROM `data` WHERE `user_id` = %s" % user_id
    args = (user_id)

    print ('1')

    cursor = CONN.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()

    print (rows)

    if rows is None:
      return('Error')
    else:
      return jsonify(rows)

# Выводит список сохраненных искусственных нейронных сетей пользователя. На вход получает
id пользователя. На выход список название его сохраненных сетей.
@app.route("/listnet")
def listnet():

    user_id = request.args['user_id']

    CONN  =  mysql.connector.connect(host='localhost',  database='tensor',  user='root',
password='')
    query = "SELECT id, netname FROM `neuronet` WHERE `user_id` = %s" % user_id
    args = (user_id)

    print ('1')

    cursor = CONN.cursor()
    cursor.execute(query)
    rows = cursor.fetchall()

    print (rows)

    if rows is None:
      return('Error')
    else:
      return jsonify(rows)

## Удаление временного файла весов H5 и хранения моделей Json
@app.route("/DeleteH5", methods=["GET", "POST"])
def DeleteH5():
  import os
```

```
  import os.path

  user_id = request.args['user_id']
  if (os.path.exists("C:/Work/Python/Keras/model" + user_id + ".h5")):
    os.remove("C:/Work/Python/Keras/model" + user_id + ".h5")
  if (os.path.exists("C:/Work/Python/Keras/model" + user_id + ".json")):
    os.remove("C:/Work/Python/Keras/model" + user_id + ".json")

  return "File removed"

# Обучение нейронной сети на стандартных наборах данных. На вход получает параметры
искусственной нейронной сети и записывает его в список данных. На выходе уведомление, что
список составлен.
@app.route("/netLearn", methods=["GET", "POST"])
def netLearn():

    global mas

    lengthStr = request.args['length']
    iStr = request.args['i']

    units = int(request.args['units'])
    activation = request.args['activation']
    use_bias = request.args['use_bias']
    kernel_initializer = request.args['kernel_initializer']
    bias_initializer = request.args['bias_initializer']
    kernel_constraint = request.args['kernel_constraint']
    bias_constraint = request.args['bias_constraint']

    i = int(iStr)
    length = int(lengthStr)

    if (i==0) :
      mas = []

    d = dict()

    if (activation != "None"):
      d['activation']=activation
    d['use_bias'] = use_bias
    if (kernel_initializer != "None"):
      d['kernel_initializer'] = kernel_initializer
    if (bias_initializer != "None"):
      d['bias_initializer'] = bias_initializer
    if (kernel_constraint != "None"):
      d['kernel_constraint'] = kernel_constraint
    if (bias_constraint != "None"):
      d['bias_constraint'] = bias_constraint

    mas.append([])  # заносим пустую строку в матрицу
    # записываем в строку под номером i параметры слоя
    mas[i].append(request.args['type'])
    mas[i].append(int(request.args['units']))
    mas[i].append(d)

    return "Good"

# Запуск модуля для обучения нейронной сети на стандартных наборах данных.
@app.route("/neuralTraining", methods=["GET", "POST"])
def neuralTraining():
```

```
  import sys
  sys.path.append('C:/Work/Python/Keras/')
  import MnistLearn

  global mas

  user_id = request.args['user_id']

  print("Learning!!!!")
  result = MnistLearn.NeuralLearnMnist(mas,user_id)
  return result

## Обучение нейронной сети на загруженных данных. На вход получает параметры
искусственной нейронной сети и записывает его в список данных. На выходе уведомление, что
список составлен.
@app.route("/netLearnData", methods=["GET", "POST"])
def netLearnData():

    global mas

    lengthStr = request.args['length']
    iStr = request.args['i']

    units = int(request.args['units'])
    activation = request.args['activation']
    use_bias = request.args['use_bias']
    kernel_initializer = request.args['kernel_initializer']
    bias_initializer = request.args['bias_initializer']
    kernel_constraint = request.args['kernel_constraint']
    bias_constraint = request.args['bias_constraint']

    i = int(iStr)
    length = int(lengthStr)

    if (i==0) :
      mas = []

    d = dict()

    if (activation != "None"):
      d['activation']=activation
    d['use_bias'] = use_bias
    if (kernel_initializer != "None"):
      d['kernel_initializer'] = kernel_initializer
    if (bias_initializer != "None"):
      d['bias_initializer'] = bias_initializer
    if (kernel_constraint != "None"):
      d['kernel_constraint'] = kernel_constraint
    if (bias_constraint != "None"):
      d['bias_constraint'] = bias_constraint

    mas.append([])  # заносим пустую строку в матрицу
    # записываем в строку под номером i параметры слоя
    mas[i].append(request.args['type'])
    mas[i].append(int(request.args['units']))
    mas[i].append(d)

    return "Good"
```

```
#Запуск модуля для обучения нейронной сети на загруженных данных.
@app.route("/neuralTrainingData", methods=["GET", "POST"])
def neuralTrainingData():

  import sys
  sys.path.append('C:/Work/Python/Keras/')
  import DataLearn

  global mas
  global directory
  global iTrain
  global iTest
  global iVal
  global fileLen

  test_path = directory + "/test/"
  train_path = directory + "/train/"
  val_path = directory + "/val/"

  iVal = int((fileLen-iTrain)/2)
  iTest = int((fileLen-iTrain)/2)

  user_id = request.args['user_id']
  classCount = int(request.args['classCount'])
  epochs = int(request.args['epochs'])
  batch_size = int(request.args['batch_size'])
  img_width = int(request.args['img_width'])
  img_height = int(request.args['img_height'])

  print("Learning!!!!")
  result                                                                    =
DataLearn.NeuralDataLearn(mas,user_id,classCount,train_path,val_path,test_path,iTrain,iVa
l,iTest,epochs,batch_size,img_width,img_height)
  return result

# Сохранение Нейронной сети. На вход получает параметры сети и ее слоев и сохраняет ее в
список. На выходе уведомление об успешной работе функции.
@app.route("/netSave", methods=["GET", "POST"])
def netSave():

    global mas
    lengthStr = request.args['length']
    iStr = request.args['i']

    neural_type = request.args['type']
    units = int(request.args['units'])
    activation = request.args['activation']
    use_bias = request.args['use_bias']
    kernel_initializer = request.args['kernel_initializer']
    bias_initializer = request.args['bias_initializer']
    kernel_constraint = request.args['kernel_constraint']
    bias_constraint = request.args['bias_constraint']

    i = int(iStr)
    length = int(lengthStr)

    d = dict()

    if (activation != "None"):
```

```
      d['activation']=activation
    d['use_bias'] = use_bias
    if (kernel_initializer != "None"):
      d['kernel_initializer'] = kernel_initializer
    if (bias_initializer != "None"):
      d['bias_initializer'] = bias_initializer
    if (kernel_constraint != "None"):
      d['kernel_constraint'] = kernel_constraint
    if (bias_constraint != "None"):
      d['bias_constraint'] = bias_constraint

    mas.append([])  # заносим пустую строку в матрицу
    # записываем в строку под номером i параметры слоя
    mas[i].append(request.args['type'])
    mas[i].append(int(request.args['units']))
    mas[i].append(d)

    return "Worked!"

# Запуск модуля для сохранения искусственной нейронной сети пользователя. На вход id
пользователя и имя нейронной сети, на выходе уведомление, что сохранение прошло успешно.
@app.route("/neuralSave", methods=["GET", "POST"])
def neuralSave():

  import sys
  sys.path.append('C:/Work/Python/Keras/')
  import NeuralSave

  global mas

  user_id = request.args['user_id']
  netname = request.args['netname']

  print("neuralSavePython")

  result = NeuralSave.modelSave(user_id,netname,mas)

  CONN   =   mysql.connector.connect(host='localhost',   database='tensor',   user='root',
password='')
  query     =     "INSERT     INTO     `layers`     (`net_id`,     `type`,     `units`,
`activation`,`use_bias`,`kernel_initializer`,`bias_initializer`,`kernel_constraint`,`bias
_constraint`) VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s)"
  args_net_id = result

  print("neuralSavePython")

  for i in range(len(mas)):
    args_type = mas[i][0]
    args_unints = mas[i][1]
    args_activation = mas[i][2]['activation']
    args_use_bias = mas[i][2]['use_bias']
    if ('kernel_initializer' in mas[i][2]):
      args_kernel_initializer = mas[i][2]['kernel_initializer']
    else:
      args_kernel_initializer = 'None'
    if ('bias_initializer' in mas[i][2]):
      args_bias_initializer = mas[i][2]['bias_initializer']
    else:
      args_bias_initializer = 'None'
    if ('kernel_constraint' in mas[i][2]):
```

```
      args_kernel_constraint = mas[i][2]['kernel_constraint']
    else:
      args_kernel_constraint = 'None'
    if ('bias_constraint' in mas[i][2]):
      args_bias_constraint = mas[i][2]['bias_constraint']
    else:
      args_bias_constraint = 'None'
    args = (args_net_id, args_type, args_unints, args_activation, args_use_bias,
args_kernel_initializer,          args_bias_initializer,          args_kernel_constraint,
args_bias_constraint)
    cursor = CONN.cursor()
    cursor.execute(query, args)
    CONN.commit()

  cursor.close()
  return result

# Получение списка слоев нейронной сети. На вход получает id сети, на выходе, что список
получен успешно.
def neuralList(net_id):

  CONN = mysql.connector.connect(host='localhost', database='tensor', user='root',
password='')
  query = "SELECT units, activation FROM `layers` WHERE `net_id`=%s" % net_id
  cursor = CONN.cursor()
  cursor.execute(query)
  row = cursor.fetchall()


  print('FROM DATA BASE--------------' + str(row[0][0]))
  print('FROM DATA BASE--------------' + str(row[1][1]))
  print('FROM DATA BASE--------------' + str(row[2][0]))

  cursor.close()

  return "Good"

# Удаление директории
@app.route("/rmdir", methods=["GET", "POST"])
def rmdir():
    curr_dir = os.getcwd()
    curr_dir1 = curr_dir  + '/' + replacer(request.args['dir'])
    for root, dirs, files in os.walk(curr_dir1, topdown=False):
      for name in files:
        os.remove(os.path.join(root, name))
      for name in dirs:
        os.rmdir(os.path.join(root, name))
    os.rmdir(curr_dir1)
    return lsDir(curr_dir)

# Удаление файла
@app.route("/rmfile", methods=["GET", "POST"])
def rmfile():
    curr_dir = os.getcwd()
    filename = replacer(request.args['file'])
    if os.path.exists(filename):
       os.remove(filename)
    else:
        print("Can not delete the %s as it doesn't exists" % filename)
    try:
```

```
        os.remove(filename)
    except:
        print("Error while deleting %s " % filename)
    return lsDir(curr_dir)

# Загрузка файла
@app.route('/download', methods=["GET", "POST"])
def fileDownload():
    #For windows you need to use drive name [ex: F:/Example.pdf]
    source=os.getcwd()
    data = request.data.decode('utf8')
    print("data-- %s" % request.data)
    data2 = json.loads(data)
    #data = request.form.to_dict()
    print("request-- %s" % data2['item'])
    filename = replacer(data2['item'])
    path="/".join([source, filename])
    print("path for download %s" % path)
    return send_file(path, as_attachment=True, attachment_filename=filename)
```