МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное Государственное бюджетное образовательное

учреждение высшего образования

«Поволжский государсТвенный технологический университет»

(ФГБОУ ВО «ПГТУ»)

Факультет (институт)

Радиотехнический факультет

Направление подготовки (специальность) 11.04.01 Радиотехника

(наименование)

Выпускник:

Фамилия Александров\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Имя Илья\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отчество Константинович\_\_\_\_\_\_\_\_\_\_\_

Тема ВКР Веб-сборщик приложений нейронных сетей для решения задач радиолокации

Кафедра Радиотехнических и медико-биологических систем

Заведующий кафедрой д.т.н., профессор Хафизов Ринат Гафиятуллович\_\_\_\_\_

(ф.и.о. уч. степень, звание, должность, подпись)

Руководитель к.т.н., доцент Хафизов Динар Гафиятуллович

(ф.и.о. уч. степень, звание, должность, подпись)

Консультанты

(ф.и.о. уч. степень, звание, должность, подпись)

Рецензент

(ф.и.о. уч. степень, звание, должность, подпись)

№ приказа об утверждении темы ВКР

ВКР начата

ВКР закончена

№ приказа о допуске к защите ВКР

Оценка Государственной экзаменационной комиссии по защите

Декан факультета

(Директор института) Дедов Андрей Николаевич (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Секретарь Государственной

экзаменационной комиссии Охотников Сергей Аркадьевич (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_202\_г.

**Аннотация**

Выпускная квалификационная работа посвящена разработке веб приложения, направленного на создание desktop приложений, которые применяют нейронные сети, сконфигурированные и обученные пользователем.

Разработан алгоритм конфигурации и обучения различных видов нейронных сетей, линейных и сверточных. Реализован API интерфейс для взаимодействие с backend частью приложения через браузер на базе языка программирования Python и инструмента Swagger.

Данная работа содержит введение, \_ разделов, заключение и список использованной литературы. Пояснительная записка изложена на \_\_ страницах.

**Annotation**

The final qualifying work is devoted to the development of a of a web application aimed at creating desktop applications that use neural networks configured and trained by the user.

An algorithm for the configuration and training of various types of neural networks, linear and convolutional, has been developed. An API has been implemented for interacting with the backend part of the application via a browser based on the Python programming language and the Swagger tool.

This work contains an introduction, \_ sections, conclusion and a list of references. The explanatory note is set out on \_\_ pages.

Содержание

[**ВВЕДЕНИЕ** 5](#_Toc199073020)

[**1.** **ЗАДАЧИ И МЕТОДЫ РАДИОЛОКАЦИИ** 8](#_Toc199073021)

[1.1 Задачи радиолокации 8](#_Toc199073022)

[1.1.1 Задача обнаружения сигнала 8](#_Toc199073023)

[1.1.2 Задача распознавания сигналов 8](#_Toc199073024)

[1.1.3 Задача различения сигналов 9](#_Toc199073025)

[1.1.4 Задача оценки параметров сигнала 9](#_Toc199073026)

[1.2 Стандартные методы решения задач радиолокации 10](#_Toc199073027)

[1.2.1 Энергетический детектор 10](#_Toc199073028)

[1.2.2 Критерий Баеса 10](#_Toc199073029)

[1.2.3 Критерий Котельникова 11](#_Toc199073030)

[1.2.4 Критерий Неймона-Пирсона 12](#_Toc199073031)

[1.2.5 Метод спектрального анализа 12](#_Toc199073032)

[1.2.6 Метод наименьших квадратов 13](#_Toc199073033)

[1.3 Виды нейронный сетей 14](#_Toc199073034)

[1.3.1 Линейная нейронная сеть 14](#_Toc199073035)

[1.3.2 Сверточная нейронная сеть 18](#_Toc199073036)

[**2.** **ОПИСАНИЕ РАЗРАБАТЫВАЕМОГО ПО** 20](#_Toc199073037)

[2.1 Инструменты и технологии для разработки 20](#_Toc199073038)

[2.2 Описание архитектуры приложения 22](#_Toc199073039)

[2.3 Описание графического интерфейса 26](#_Toc199073040)

[2.3.1 Основной интерфейс программы 26](#_Toc199073041)

[2.3.2 Загрузка своего датасета 26](#_Toc199073042)

[2.3.3 Обучение линейной нейронной сети 28](#_Toc199073043)

[2.3.4 Обучение сверточной нейронной сети 32](#_Toc199073044)

[2.4 Описание примера работы с результатами работы программы 34](#_Toc199073045)

[**3.** **АПРОБАЦИЯ** 36](#_Toc199073046)

[3.1 Обнаружение сигнала 36](#_Toc199073047)

[3.2 Распознавание сигнала 38](#_Toc199073048)

[**4.** **ИНСТРУКЦИЯ К ПРИМЕНЕНИЮ** 40](#_Toc199073049)

[**5.** **ЗАКЛЮЧЕНИЕ** 41](#_Toc199073050)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 42](#_Toc199073051)

# **ВВЕДЕНИЕ**

Стремительные темпы развития технологий последних десятилетий обусловили значительное увеличение интереса к такому разделу машинного обучения, как глубокое обучение (Deep Learning) основанное на нейронных сетях, которое стало одним из ключевых инструментов анализа и обработки больших объемов данных.

Благодаря своей способности моделировать сложные зависимости между входными параметрами и выходными результатами, нейронные сети нашли широкое применение в различных областях науки и техники, включая решение задач радиолокации.

Радиолокационные системы традиционно характеризуются сложностью и высокой степенью шумов при распространении сигналов, отраженных от объектов различной природы. Применение нейронных сетей позволяет существенно повысить точность распознавания целей, качество оценки характеристик радиосигнала и эффективность обнаружения малозаметных объектов.

Несмотря на очевидные преимущества, широкое внедрение нейронных сетей в радиолокационных системах сдерживается рядом факторов:

Во-первых, обучение нейронных сетей представляет собой ресурсоемкий процесс, требующий значительных вычислительных мощностей и специализированных технических решений.

Во-вторых, создание эффективной модели требует глубоких знаний и опыта, поскольку необходимо понимание принципов функционирования языка программирования, особенностей архитектуры нейронных сетей и специфики используемых библиотек.

Эти сложности делают разработку и настройку нейронных сетей трудоемким процессом даже для опытных специалистов, что ограничивает распространение технологии среди широких кругов пользователей.

Целью данной ВКР является разработка специализированного инструмента для упрощения обучения нейронных сетей. Инструмент позволит регулировать параметры обучения в ручном режиме, не задумываясь о реализации, и обучать модель на конкретных пользовательских данных. Данный инструмент призван устранить препятствия, возникающие перед специалистами, затрудняющие эффективное использование нейронных сетей в повседневной практике радиолокационного мониторинга.

Основные задачи исследования включают:

1. Анализ радиолокационных задач и их решений, выделение задач, которые могут решить нейронные сети.
2. Разработка алгоритма упрощенного создания нейронных сетей под данные радиолокационных сигналов, минимизирующего участие пользователя в процессе разработки и обучения нейронной сети.
3. Создание прототипа программного продукта – веб-сборщика приложений, обеспечивающего автоматизацию основных шагов разработки нейронных сетей и облегчающего эксплуатацию готового продукта конечными пользователями.
4. Проведение экспериментальной проверки предложенного подхода на тестовом наборе радиолокационных данных.

Реализация поставленной цели и выполнение перечисленных задач позволит создать эффективный инструмент поддержки принятия решений в радиолокационном мониторинге, облегчить переход на новые технологии специалистам-практикам и обеспечить повышение точности и надежности результатов обработки радиолокационных данных.

Разработанный инструмент обеспечивает адаптацию нейронной сети под специфику решаемой задачи, уменьшая потребность в экспертизе программистов и исследователей в области нейронных сетей. Программный комплекс способен автоматически формировать готовую модель, которую, при необходимости, можно будет интегрировать в существующие радиолокационные комплексы, обеспечивая комфортную работу конечных пользователей без необходимости углубленного изучения тонкостей архитектуры нейронных сетей.

Практическая значимость состоит в возможности существенного увеличения точности анализа и расширения функциональных возможностей существующих радиолокационных систем.

# **ЗАДАЧИ И МЕТОДЫ РАДИОЛОКАЦИИ**

Традиционно радиолокация решает одни из следующих задач: задача обнаружения сигналов, задача распознавания сигналов, задача разрешения сигналов и задача оценки параметров сигнала.

## Задачи радиолокации

### Задача обнаружения сигнала

Задача обнаружения сигнала сводится к тому, чтобы во входном сигнале наилучшим способом принять решение о наличии или отсутствии искомого образа [1].

Математически постановка задачи выглядит следующим образом: Имеется два возможных сценария сценария. Нулевая гипотеза – сигнал отсутствует и, наблюдаемый сигнал состоит только из шума :

(1.1.1.1)

Альтернативная гипотеза – сигнал присутствует: и наблюдаемый сигнал представляет собой сумму полезного сигнала и шума :

(1.1.1.2)

Цель задачи – принять решение о том: какая из гипотез верна, на основе наблюдений .

### Задача распознавания сигналов

Задача распознавания сигналов сводится к тому, чтобы разработать наилучший алгоритм, согласно которому по наблюдаемому сигналу после выявления всех полезных образов определить их принадлежность к соответствующим объектам – источникам полезных сигналов [1].

Математически постановка задачи выглядит следующим образом: Имеется сигнал , который состоит из образов и шума шума :

, (1.1.3)

Цель задачи – на основе наблюдаемого сигнала идентефицировать и разделить все образы , которые были переданы.

### Задача различения сигналов

Задача различения сигналов сводится к тому, чтобы разработать наилучшее правило или алгоритм, согласно которому будет достигнуто отделение двух сигналов, близким друг другу по определенным параметрам[6].

Математически постановка задачи выглядит следующим образом: Имеется два сигнала и которые являются близкими по временному положению или спектру. На приёмной стороне получается сигнал , которыйпредставляет собой смесь переданны сигналов с шумом :

, (1.1.2)

где – это номер переданного сигнала.

Цель задачи – на основе наблюдаемого сигнала раздельно выделить сигналы и , опираясь на характеристики принятых сигналов и помех.

### Задача оценки параметров сигнала

Задача оценки параметров сигнала сводится к тому, чтобы выбрать параметры таким образом, чтобы сигнал, воссозданный по этим параметрам, был наиболее схож с принятым[7].

Математически постановка задачи выглядит следующим образом: Имеется сигнал , который вычисляется по следующей формуле:

(1.1.4.1)

Где, A – амплитуда сигнала, – циклическая частота, t – значение временного параметра, – фаза сдвига.

Цель задачи – выбор параметров A, , таким образом, чтобы минимизировать ошибку оценивания.

(1.1.4.2)

Формально, для случайного вектора параметров , оценивается такой вектор такой, что критерий точности достигает минимума:

(1.1.4.3)

L – функция потерь, зависящая от выбранного метода оценки.

## Стандартные методы решения задач радиолокации

Для решения каждой из поставленных задач разработаны специальные методы, основанные на: на сравнении общей энергии, корреляционной функции, спектральной характеристики, низкочастотных, высокочастотных, полосовых и режекторных фильтров.

### Энергетический детектор

Энергетический детектор – это метод, основанный на сравнении общей энергии сигнала с некоторым пороговым значением . Энергию сигнала можно оценить с помощью следующего выражения:

(1.2.1.1)

Если энергия сигнала превышает заданный порог, делается вывод о наличии сигнала.

(1.2.1.2)

Энергетический детектор позволяет решить задачу обнаружения сигнала.

### Критерий Баеса

Критерий Баеса – это вероятностный подход для определения совпадения одной из двух гипотез на основе входных данных, через минимизацию среднего риска (вероятности ошибки) на основе априорных вероятностей и знаний о вероятностных характеристиках наблюдаемых данных.

В контексте радиолокации рассматриваются две гипотезы: – отсутствие сигнала и – наличие сигнала.

Тогда по критерию Баеса верность теории достигается при соблюдении следующего условия:

(1.2.2.1)

где, – отношение правдоподобия, x – входные данные, – априорная вероятность отсутствия сигнала во входных данных, – априорная вероятность наличия сигнала во входных данных, – вероятность получить x при верности теории об отсутствии сигнала и – вероятность получить x при верности теории об наличии сигнала, – цена ошибки первого рода, ложной тревоги, – цена ошибки второго рода, пропуска, и – цены ошибок правильного обнаружения, как правило равные нулю.

Критерий Баеса позволяет решить задачу обнаружения сигнала.

### Критерий Котельникова

Критерий Баеса – это вероятностный подход для определения совпадения одной из двух гипотез на основе входных данных, через знания о вероятностных характеристиках наблюдаемых данных.

В контексте радиолокации рассматриваются две гипотезы: – отсутствие сигнала и – наличие сигнала.

Тогда по критерию Котельникова верность теории достигается при соблюдении следующего условия:

(1.2.3.1)

где, – отношение правдоподобия, x – входные данные, – вероятность получить x при верности теории об отсутствии сигнала и – вероятность получить x при верности теории о наличии сигнала, пороговый коэффициент для вероятности ложной тревоги.

Критерий Котельникова позволяет решить задачу обнаружения сигнала.

### Критерий Неймона-Пирсона

Критерий Баеса – это вероятностный подход для ложной тревоги на основе входных данных, через знания о вероятностных характеристиках наблюдаемых данных и оптимальном пороге вероятности ложной тревоги.

Тогда по критерию Неймона-Пирсона верность теории достигается при соблюдении следующего условия:

(1.2.4.1)

где, – отношение правдоподобия, x – входные данные, – вероятность получить x при верности теории об отсутствии сигнала и – вероятность получить x при верности теории о наличии сигнала, пороговый коэффициент для вероятности ложной тревоги.

Оптимальный пороговый коэффициент выбирается так, чтобы обеспечить желаемый уровень вероятности ложной тревоги , которая рассчитывается по следующей формуле:

где, – вероятность получить x при верности теории об отсутствии сигнала.

Критерий Неймона-Пирсона позволяет решить задачу обнаружения сигнала.

### Метод спектрального анализа

Спектральный анализ позволяет выявить во входном сигнале характерные частоты, которыми описывается искомый образ и на этой основе сделать вывод о наличии эталонного сигнала во входном. Спектральный анализ строится на преобразовании Фурье. Преобразование Фурье – это математическая операция, которая позволяет перевести функцию времени в функцию частоты. Для аналоговых сигналов оно имеет следующий вид:

где, s(t) функция входного сигнала, – круговая частота, -j – мнимая единица.

Для дискретного сигнала преобразование выглядит следующим образом:

(1.2.6.2)

где, – дискретный сигнал, N – размерность дискретного сигнала, j – мнимая единица.

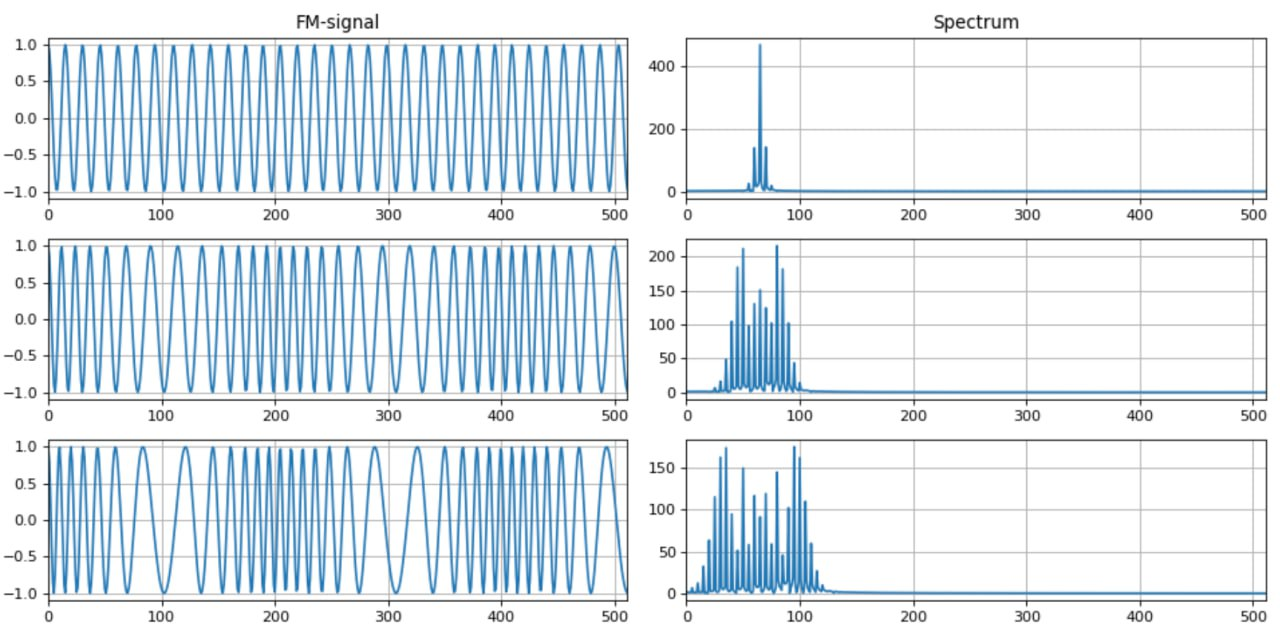


Рисунок .1 - Сигналы с различной степенью частотной модуляции и их спектры.

Как можно заметить на рисунке 1.2.1, спектр имеет ярко выраженные пики на определенных значениях частоты. В сравнении данных значений с эталонными и заключается метод спектрального анализа.

Метод спектрального анализа позволяет решать задачи обнаружения сигнала и распознавания образов в сигнале.

### Метод наименьших квадратов

Задача метода наимеьших квадратов выбрать такие значения параметров , при которых будет достигнута минимальная сумма кв

(1.2.7.1)

где, L(θ) – функция потерь (сумма квадратов отклонений), – набор данных, по которым необхожимо составить функцию, – функция, описывающая зависимость выходной величины y, от независимой переменной при фиксированных значениях параметров .

Метод наименьших квадратов позволяет решать задачу оценки сигнала.

## Виды нейронный сетей

Нейронные сети разделяют на три типа: линейные, сверточные и реккурентные. В контексте решения радиолокационных задач наиболее интересными являются линейные и сверточные.

### Линейная нейронная сеть

Линейная нейронная сеть представляет собой совокупность нейронов, или персептронов, которые организованы в слои и соединены между собой.

Входной слой выполняет функцию распределения данных. Выходной слой обрабатывает информацию, поступающую от предыдущих слоев, и выдает конечные результаты.

Слои, находящиеся между входным и выходным, называются промежуточными или скрытыми, и они также занимаются обработкой данных. Каждый нейрон предыдущего слоя соединен синаптическими связями со всеми нейронами следующего слоя, что создает однородную и регулярную топологию многослойной нейронной сети [5].

Искусственный нейрон выглядит следующим образом:

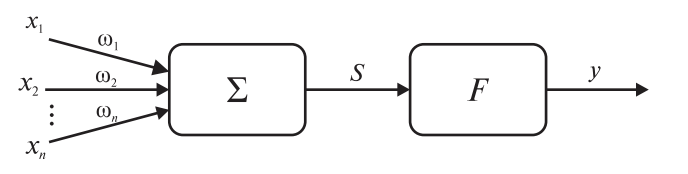


Рисунок .1 – Искусственный нейрон

где, x – входной сигнал, – весовой коэффициент, S – взвешенная сумма, F – оператор нелинейного преобразования, y – выходной сигнал.

Математически расчет искусственного нейрона выглядит следующим образом:

(1.3.1.1)

где, X – вектор входного сигнала, W – весовой вектор, F – оператор нелинейного преобразования, y – выходной сигнал.

Сумма произведений входных сигналов на весовые коэффициенты называется взвешенной суммой. Она представляет собой скалярное произведение вектора весов на входной вектор:

(1.3.1.2)

где, – длины векторов W и X соответственно, угол между векторами W и X.

В качестве оператора нелинейного преобразования используется функция активации. Пусть T – порог нелинейного элемента, который характеризует положение функции активации по оси абцисс. С учетом T взвешенную сумму можно представить следующим образом:

(1.3.1.3)

Одной из наиболее простых и популярных функций активации является сигмоида:

(1.3.1.4)

где, S – сумма C > 0 – коэффициент, характеризующий ширину сигмоидной функции по оси абцисс.

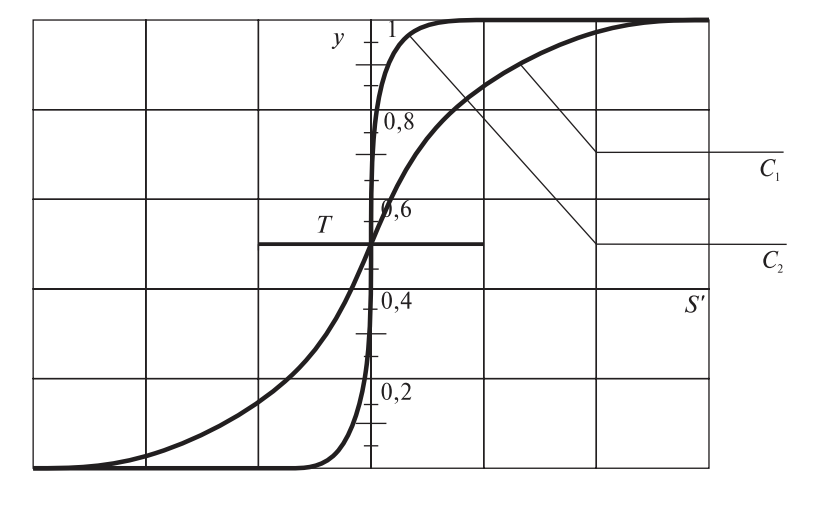


Рисунок .2 – Функция активации сигмоида

Слой нейронной сети – это множество нейронных элементов, на которые в каждый такт времени параллельно поступает информация от других нейронных элементов сети. Однослойная нейронная сеть будет выглядеть следующим образом [5]:

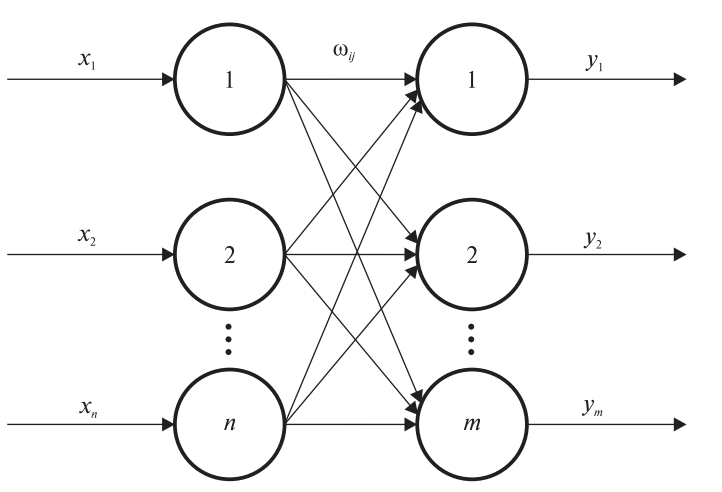


Рисунок . – Топология однослойной нейронной сети

Тогда выходное значение j-го элемента второго слоя будет расчитыватся по следующей формуле:

(1.3.1.5)

где, – порог j-го нейронного элемента выходного слоя, – сила синаптической связи между i-м нейроном распределительного слоя и j-м

нейроном обрабатывающего слоя.

Совокупность весовых коэффициентов W можно представить в виде матрицы, размерностью m n:

(1.3.1.6)

Тогда вектор – столбец взвешенной суммы в матричном виде определяется по следующей формуле:

(1.3.1.7)

где, T – вектор-столбец порогов нейронных элементов второго слоя.

На практике чаще всего применяется многослойная нейронная сеть, которая содержит несколько скрытых слоев и выглядит следующим образом [5]:

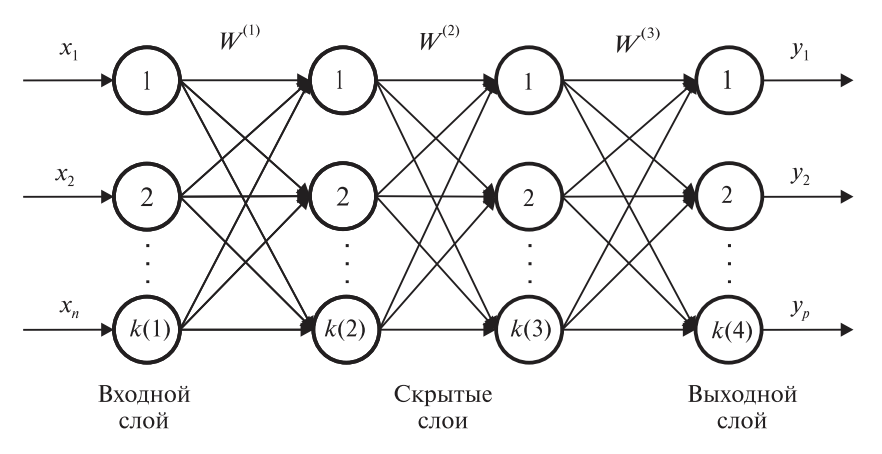


Рисунок . – Топология линейной нейронной сети

Общее количество синаптических связей многослойной линейной нейронной сети определяется по формуле:

(1.3.1.8)

где, p – общее количество слоев нейронной сети, – количество нейронных элементов в i-м слое.

Линейная нейронная сеть позволяет решать задачи обнаружения, распознавания, разрешения и оценки сигнала.

### Сверточная нейронная сеть

Сверточная нейронная сеть (CNN) представляет собой специализированный класс нейронных сетей, разработанный для обработки многомерных данных, таких как изображения.

Основное отличие сверточной нейронное сети от других заключается в применении операции свертки и предвыборки.

Свертка (Convolution) – это процедура фильтрации изображений путём перемножения небольшого фрагмента изображения с особым матрицей-фильтром (ядром свёртки). Итоговая операция создаёт новую версию изображения, акцентируя определённые свойства или детали.

(1.3.2.1)

где, O – выходной признак, I – входное изображение, K – ядро свертки, f – функция активации.

Нелийнейность – после этапа свёртки задействуют специальную нелинейную функцию активации (например, ReLU), которая придаёт модели способность усваивать сложные взаимосвязи и формировать более глубокие представления.

Уменьшение размерности (Pooling) – эта стадия понижает детализацию изображения, фокусируясь лишь на важнейших признаках. Чаще всего применяют технику максимума (max-pooling) либо усреднения (average pooling).

(1.3.2.2)

где, P – выход подвыборки, O – данные после свертки.

Завершающие слои прямого соединения – после череды операций свёртки и уменьшения размерности добавляется один или несколько полносвязных слоёв, предназначенных для выполнения задач классификации или регрессивного анализа.

Топология сверточной нейронной сети представлена на Рисунок 1.6, где выделены два основных этапа, а именно процесс преобразования неструктурированных данных или многомерных таблиц (feature extraction) и процесс классификации (classification), которй напоминает линейную нейронную сеть.

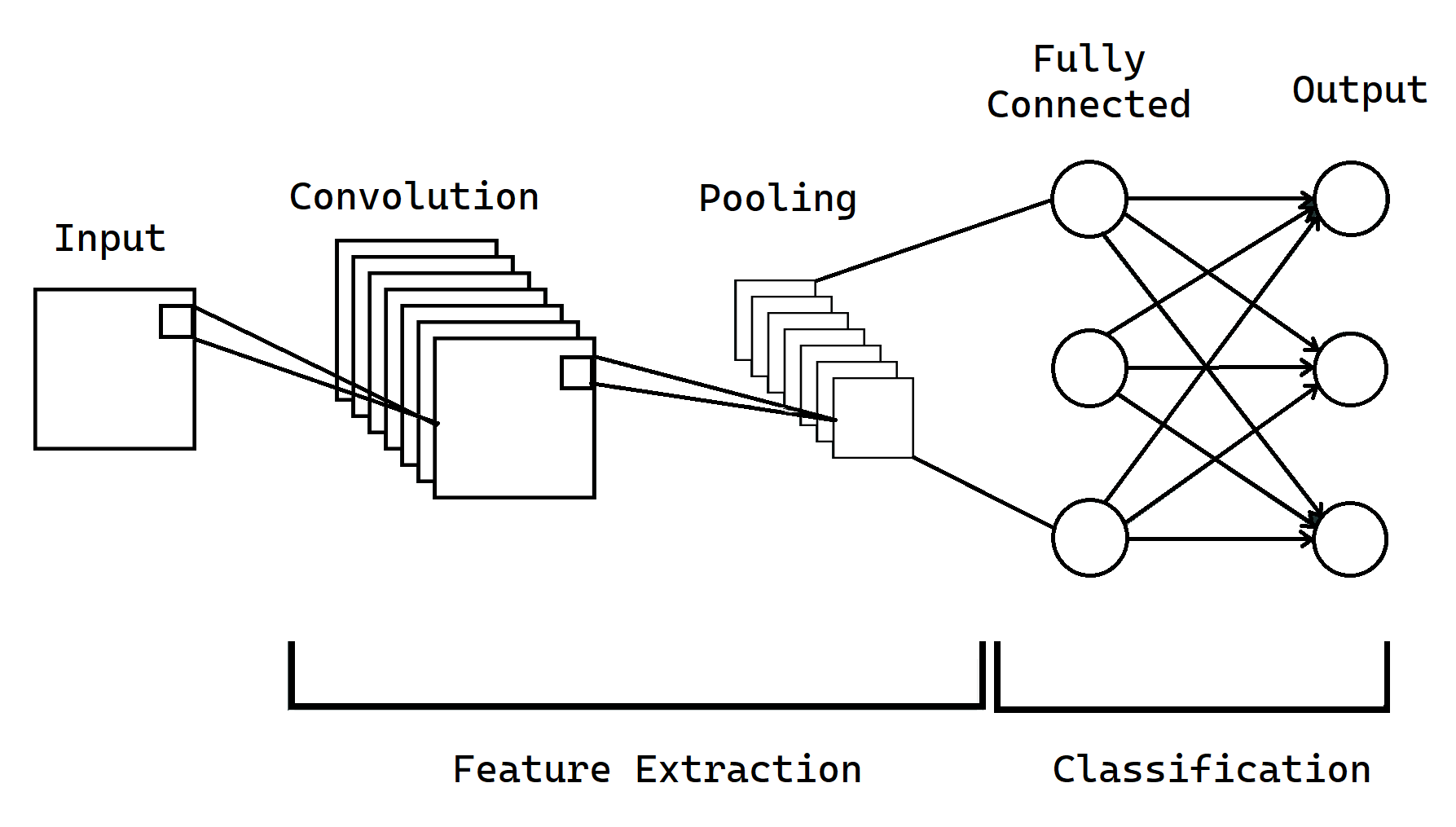


Рисунок .1 – Топология сверточной нейронной сети

Сверточная нейронная сеть позволяет решать задачи обнаружения и распознавания объектов на изображениях и видеозаписях.

# **ОПИСАНИЕ РАЗРАБАТЫВАЕМОГО ПО**

## Инструменты и технологии для разработки

В качестве основного языка программирования был выбран Python. Так-как это один из самых популярных и универсальных языков, который активно используется в области машинного обучения, искусственного интеллекта, обработки больших данных и изображений – именно тех сфер, которые лежат в основе решения задач радиолокации. А также, Python обладает простой и читаемой синтаксисической структурой, что ускоряет процесс разработки и облегчает поддержку кода. К достоинствам Python относятся:

**Богатая экосистема библиотек и фреймворков**, что позволяет быстро интегрировать готовые решения и сосредоточиться на специфике задачи, а не на низкоуровневой реализации.

**Интерактивность и гибкость**, которые важны для экспериментов и отладки моделей нейронных сетей.

**Широкое сообщество и обширная документация**, что упрощает поиск решений и обучение.

**Кроссплатформенность**, позволяющая запускать приложения на различных операционных системах без существенных изменений в коде.

Так же Python, имеет ряд мощных библиотек и инструментов, которые облегчают разработку веб-приложений и работу с нейронными сетями.

**Flask** – легковесный веб-фреймворк для Python, который обеспечивает удобное создание веб-приложений и API. Его простота и расширяемость позволяют быстро разрабатывать серверную часть приложения.

**Flasgger** – расширение для Flask, которое автоматизирует создание документации API на основе спецификаций OpenAPI (Swagger), что улучшает удобство взаимодействия с веб-сервисом.

**Torch и torchvision** – библиотеки для создания и обучения нейронных сетей. PyTorch (torch) обеспечивает гибкий и эффективный инструментарий для построения моделей глубокого обучения, а torchvision содержит наборы данных и инструменты для обработки изображений, что особенно актуально для задач радиолокации.

**OpenCV-python** – библиотека для обработки и анализа изображений, используемая для подготовки данных и предобработки радиолокационных сигналов и изображений.

**TensorFlow** – ещё одна популярная библиотека для машинного обучения и глубокого обучения, предоставляющая мощные средства для разработки и развертывания моделей.

**PyInstaller** – инструмент для упаковки Python-приложений в самостоятельные исполняемые файлы, что облегчает распространение и установку программ без необходимости наличия установленного интерпретатора Python.

**Tkinter** – стандартная библиотека для создания графического интерфейса пользователя (GUI) на Python, используемая для разработки удобных настольных приложений.

В качестве инструмента для автоматического развертывания приложения был выбран Docker. Docker – это платформа для контейнеризации приложений, которая значительно упрощает процесс развертывания и управления программным обеспечением. Его ключевые преимущества в рамках данного проекта:

**Кроссплатформенность** – контейнеры Docker работают одинаково на различных операционных системах (Windows, Linux, macOS), что обеспечивает стабильность и переносимость приложения.

**Автоматизация развертывания**  – Docker позволяет упаковать все зависимости и конфигурации в единый контейнер, избавляя от необходимости вручную устанавливать и настраивать множество библиотек и инструментов.

**Изоляция среды** – контейнеры обеспечивают изоляцию приложений, что уменьшает конфликты между зависимостями и повышает безопасность.

**Упрощение масштабирования и обновления** – благодаря Docker можно быстро разворачивать новые версии приложения и масштабировать сервисы в зависимости от нагрузки.

Использование Docker в проекте позволяет создать единый и воспроизводимый образ веб-сборщика, обеспечив удобство эксплуатации и переносимости между различными средами разработки и продакшена.

## Описание архитектуры приложения

В архитектуре данного приложения реализовано чёткое разделение на два основных контекста: Веб контекст (Web) и контекст бизнес-логики (Application). Такой подход обладает рядом преимуществ:

**Разделение ответственности** – Web отвечает за интерфейс взаимодействия с пользователем, а Application за реализацию логику по работе с нейронными сетями.

**Улучшенная тестируемость** – каждый контекст можно тестировать отдельно, что облегчает выявление и исправление ошибок.

**Повышенная модульность и масштабируемость** – можно независимо расширять или заменять бизнес-логику без влияния на веб-интерфейс и наоборот.

**Возможность многократного использования бизнес логики** – например, её можно использовать не только в веб-приложении, но и в других интерфейсах (CLI, API, мобильных приложениях).

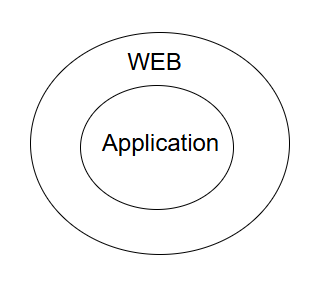


Рисунок .1 – Описание вложенности контекстов приложения

Внутри веб контекста (Web) находится логика по взаимодействию с пользователем, а именно:

**Модуль обработки HTTP сообщений** – отвечает за приём, маршрутизацию и обработку HTTP-запросов от клиентов. Его основные функции: Приём входящих запросов (GET, POST, PUT, DELETE и др.) через веб-сервер (например, Flask). Валидация и парсинг данных из запросов для последующей передачи в бизнес-логику. Маршрутизация запросов к соответствующим контроллерам, которые реализуют необходимую логику. Формирование и отправка корректных HTTP-ответов клиенту, включая обработку ошибок и исключений.

**Модуль пользовательского графического интерфейса** – отвечает за создание и документирования пользовательского интерфейса API с применением инструмента **Flasgger**, который интегрируется с Flask и автоматически генерирует интерактивную документацию на основе спецификаций OpenAPI (Swagger).

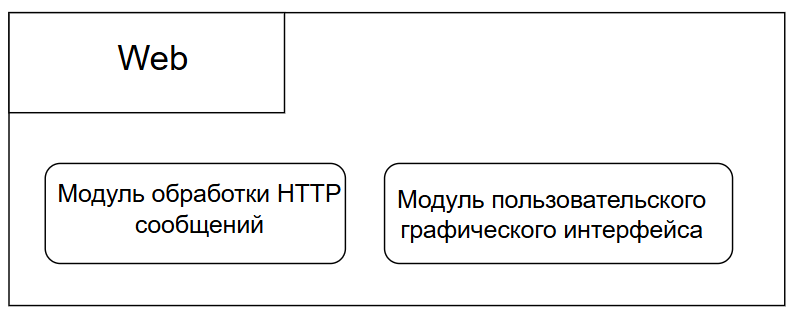


Рисунок . – Модульная схема контекста Web

Внутри контекста бизнес-логики (Application) выделены четыре отдельных модуля:

**Модуль работы с данными для обучения** – отвечает за управление данными пользователей. Функции включают создание датасета для конкретного пользователя, удаление датасета, получение списка доступных датасетов и получение конкретного датасета. Это обеспечивает персонализацию и удобство работы с данными.

**Модуль работы с моделями нейронных сетей** – аналогично модулю работы с данными для обучения, этот модуль позволяет создавать, удалять, получать список и конкретные модели нейронных сетей, что обеспечивает управление жизненным циклом моделей.

**Модуль обучения линейных нейронных сетей** – специализированный модуль, который отвечает за процесс обучения линейных моделей, оптимизированных под задачи радиолокации.

**Модуль обучения сверточных нейронных сетей** – отвечает за обучение более сложных моделей – сверточных нейронных сетей, которые эффективны для обработки изображений и радиолокационных данных.

Такое разделение позволяет сосредоточить специфичные алгоритмы и методы обучения в отдельных модулях, улучшая читаемость, повторное использование и поддержку кода.

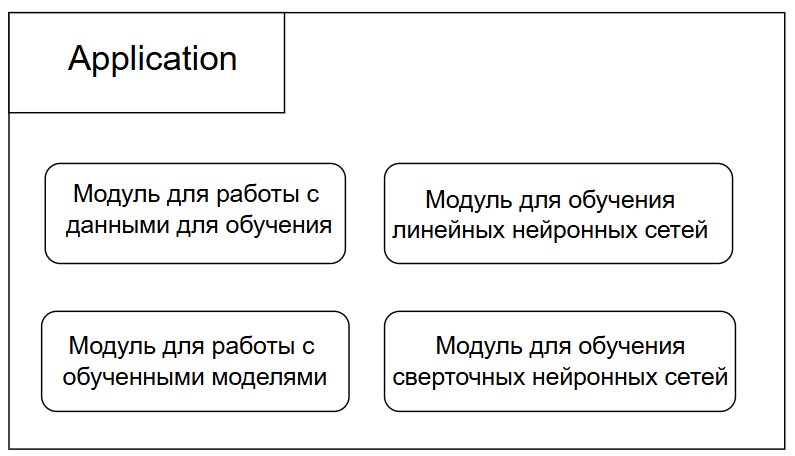


Рисунок .3 – Модульная схема контекста Application

Блок схема взаимодействия пользователя с приложением выглядит следующим образом:

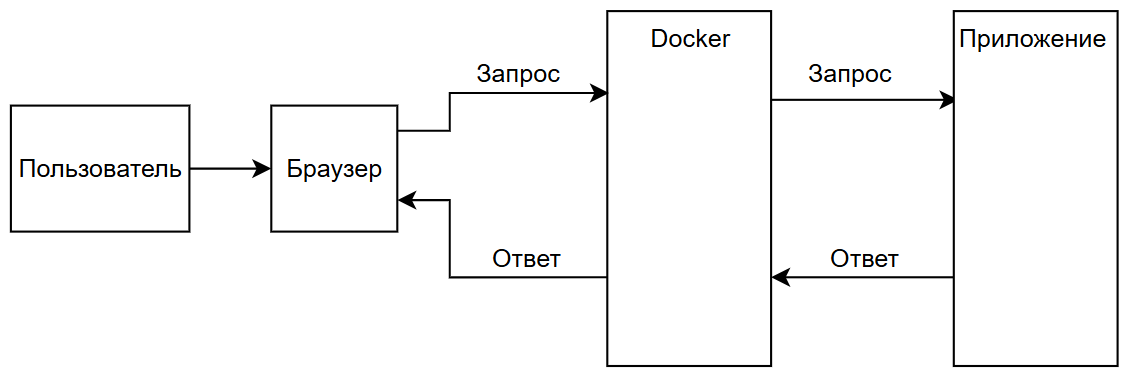


Рисунок .4 – Блок схема взаимодействия пользователя с приложением

## Описание графического интерфейса

### Основной интерфейс программы

Для работы с программой необходимо загрузить свой набор данных и запустить обучение линейной, либо сверточной нейронной сети. В интерфейсе программы эти действия выглядят следующим образом:

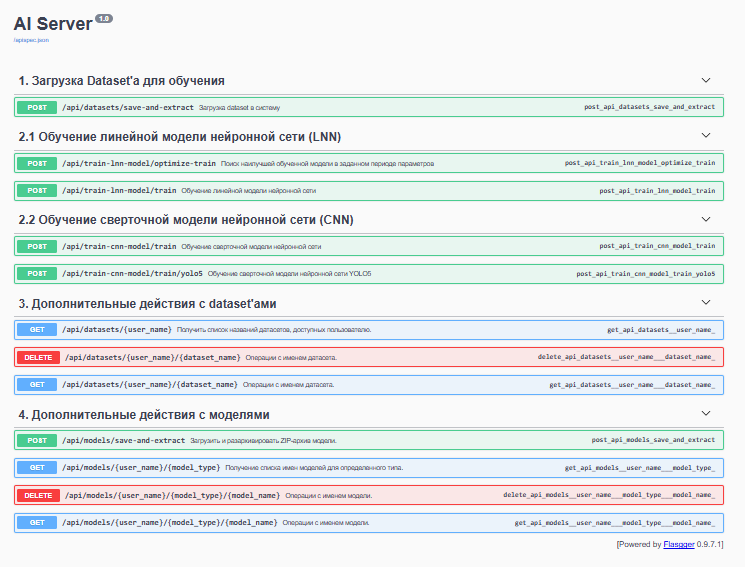


Рисунок .1 – Основной интерфейс программы

### Загрузка своего датасета

Для корректной загрузки своей выборки данных необходимо предоставить имя (username) и архивный файл (.zip) с данными.

В описании к методу указано требование к формату и структуре файлов в зависимости от типа нейронной сети, для которой предназначены данные: линейная сеть, сверточная сеть. А также указана организация файлов для корректного обнаружения или распознавания:

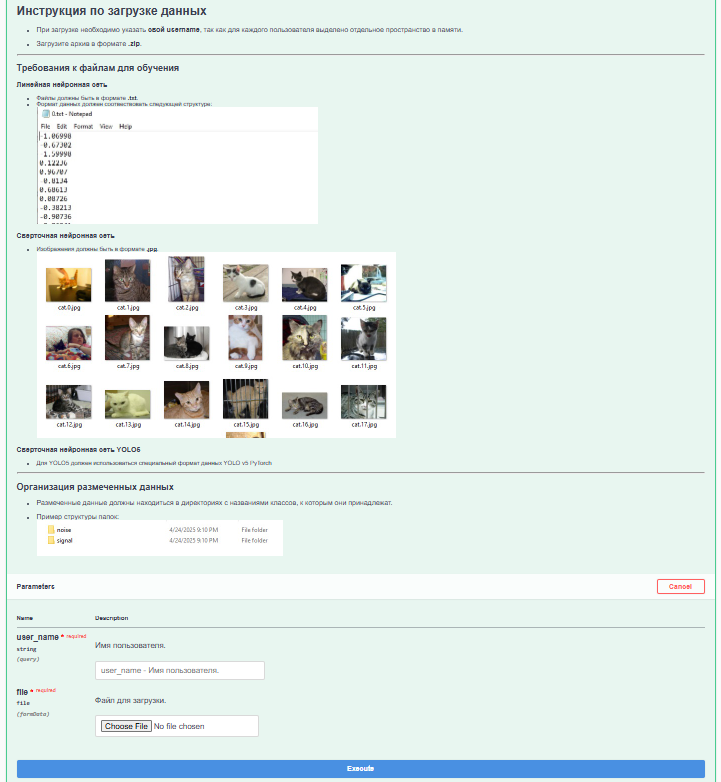


Рисунок .1 – Метод предоставляющий возможность загрузки своего датасета

После успешного выполнения метода появится следующая информация:

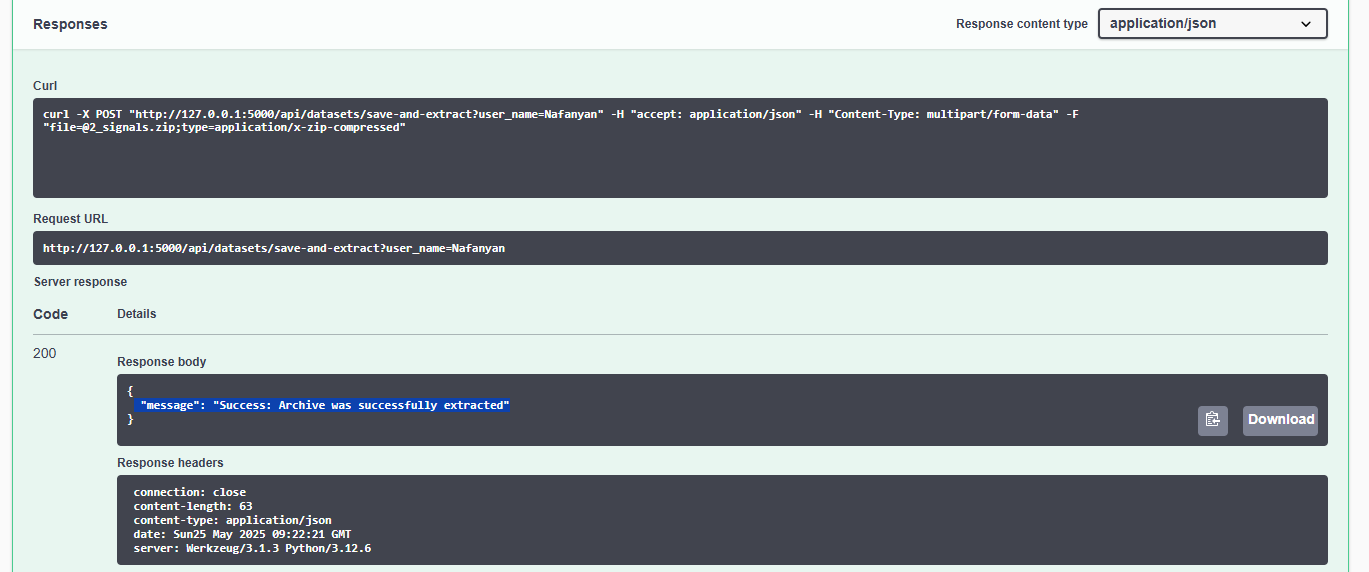


Рисунок 2.3.2. – Успешный ответ сервера на загрузку своего датасета

### Обучение линейной нейронной сети

Для корректного обучения линейной нейронной сети требуется указать следующие параметры:

* **ai\_model**: Тип модели – 'binary\_classification' для бинарной классификации или 'multiple\_classification' для многоклассовой.
* **epochs**: Количество эпох обучения, определяющее, сколько раз модель пройдет по всему набору данных.
* **batch\_size**: Размер батча – количество примеров, обрабатываемых одновременно за одну итерацию.
* **neurons\_in\_layers**: Массив с количеством нейронов в каждом скрытом слое (без учета входного и выходного слоя).
* **activations**: Массив функций активации для каждого скрытого слоя, из возможных: ['relu', 'leaky\_relu', 'sigmoid', 'softmax', 'softplus', 'softsign', 'exponential', 'swish'].
* **optimizer**: Функция оптимизации, выбираемая из ['sgd', 'rmsprop', 'adam', 'adadelta', 'adamax', 'nadam', 'ftrl'].
* **train\_percentage**: Процент тренировочной выборки от общего набора данных (число от 0 до 100).
* **test\_percentage**: Процент проверочной выборки от общего набора данных (число от 0 до 100).
* **user\_name**: Имя пользователя, под которым ведется обучение.
* **dataset\_name**: Название датасета, используемого для обучения.
* **trained\_model\_name**: Имя, которое будет присвоено обученной модели.
* **is\_create\_app**: Опция создания приложения на основе обученной модели – 'yes' или 'no'.

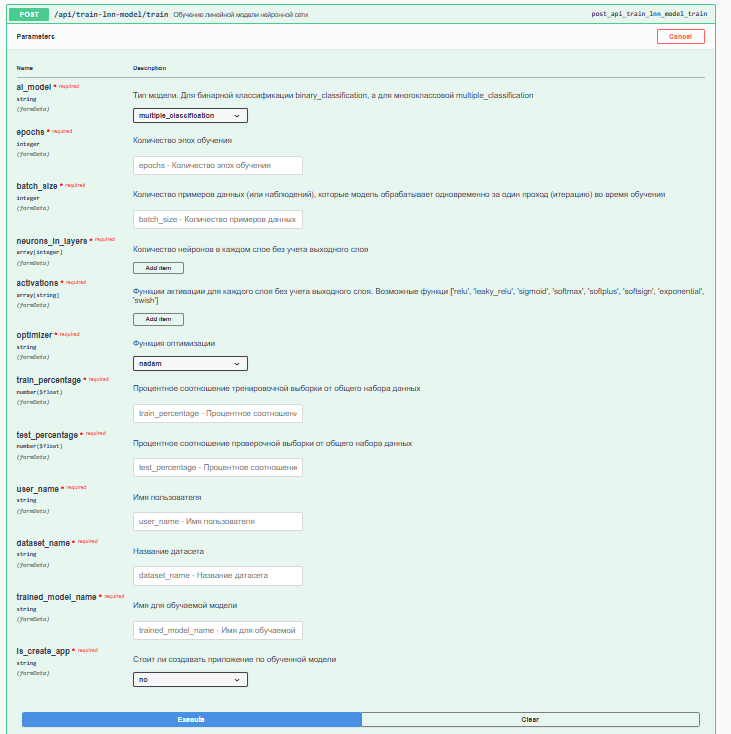


Рисунок .1 – Метод для обучения линейной нейронной сети

После выполнения запроса в ответе будет предоставлен .zip архив, доступный к скачиванию:

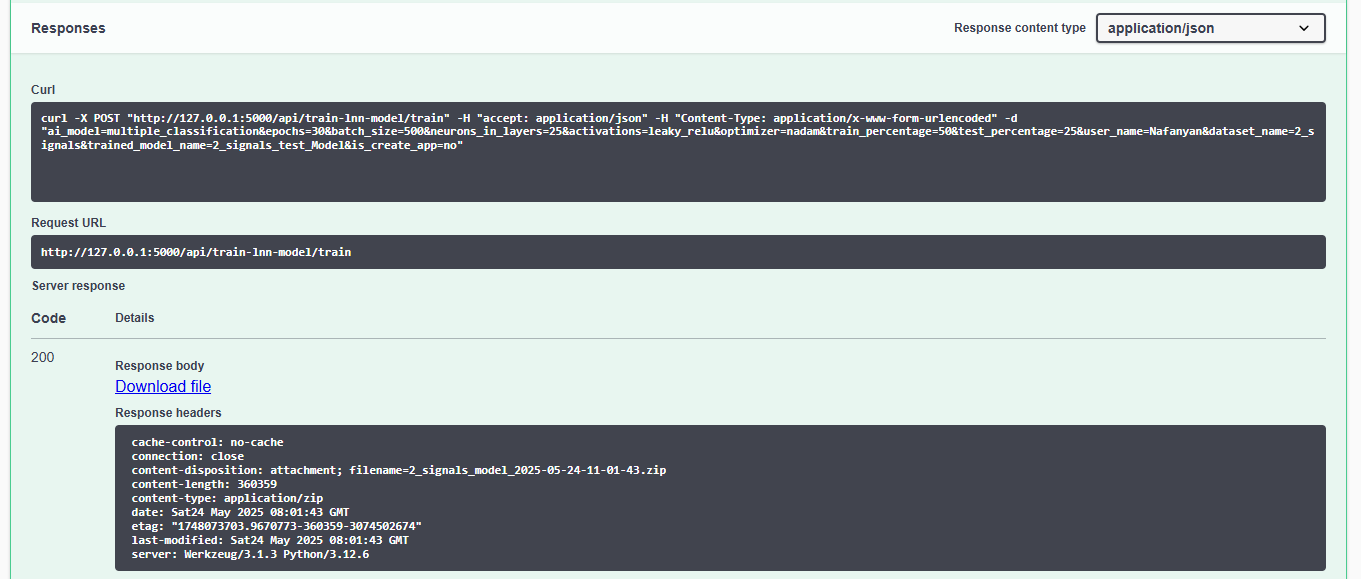


Рисунок .2 – Успешный ответ сервера на обучение линейной нейронной сети

Для корректного поиска наиболее точной обученной модели в заданном диапазоне параметров требуется указать следующие параметры:

* **ai\_model**: Тип модели – 'binary\_classification' для бинарной классификации или 'multiple\_classification' для многоклассовой.
* **epochs**: Массив допустимых значений количества эпох обучения, среди которых будет вестись поиск оптимального.
* **hidden\_layers**: Массив допустимых вариантов количества скрытых слоев для выбора оптимальной архитектуры.
* **batch\_sizes**: Массив допустимых значений размера батча – количества примеров, обрабатываемых за одну итерацию.
* **neurons\_per\_layers**: Массив допустимых вариантов количества нейронов в скрытых слоях, среди которых будет выбран лучший.
* **activation\_functions**: Массив возможных функций активации для скрытых слоев из списка ['relu', 'leaky\_relu', 'sigmoid', 'softmax', 'softplus', 'softsign', 'exponential', 'swish'].
* **optimizers**: Массив возможных функций оптимизации из ['sgd', 'rmsprop', 'adam', 'adadelta', 'adamax', 'nadam', 'ftrl'].
* **user\_name**: Имя пользователя, под которым ведется поиск и обучение.
* **dataset\_name**: Название датасета, используемого для обучения.
* **trained\_model\_name**: Имя, которое будет присвоено найденной лучшей модели.
* **is\_create\_app**: Опция создания приложения на основе найденной модели – 'yes' или 'no'.

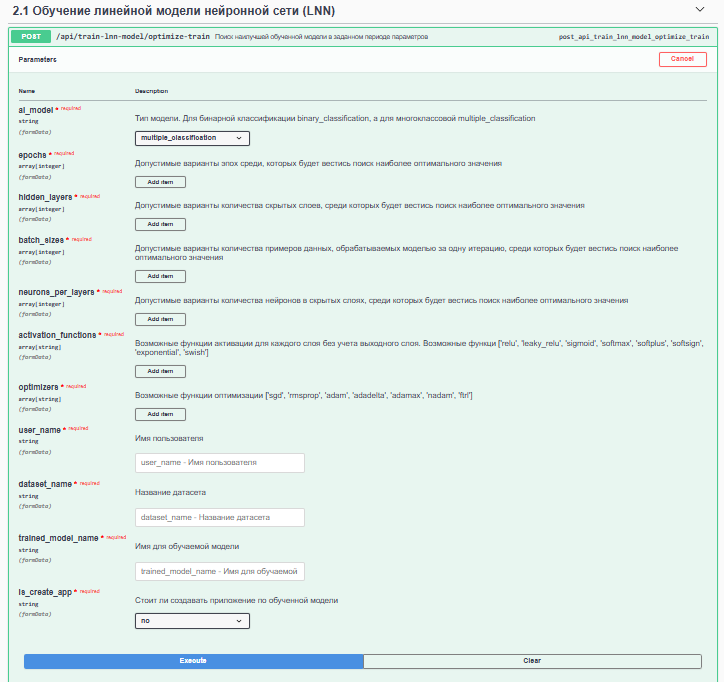


Рисунок .3 – Метод поиска наиболее точной модели линейной нейронной сети

После выполнения запроса в ответе будет предоставлен .zip архив, доступный к скачиванию:

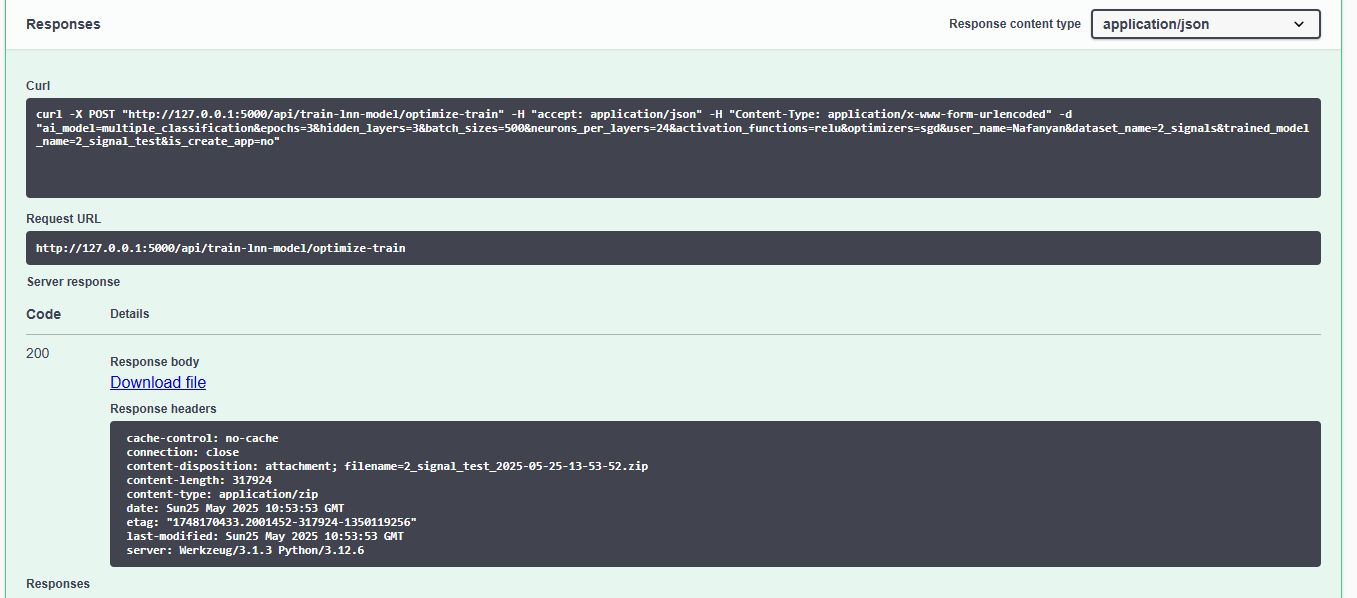


Рисунок .4 – Успешный ответ сервера на поиск наиболее точной модели линейной нейронной сети

### Обучение сверточной нейронной сети

Для корректного обучения сверточной модели нейронной сети требуется указать следующие параметры:

* **ai\_model**: Тип модели – 'binary\_classification' для бинарной классификации или 'multiple\_classification' для многоклассовой.
* **img\_size**: Размер, к которому будет приводиться изображение. Указывается одно число, так как форма изображения квадратная.
* **epochs**: Количество эпох обучения, определяющее, сколько раз модель пройдет по всему набору данных.
* **batch\_size**: Размер батча – количество примеров, обрабатываемых одновременно за одну итерацию.
* **filters**: Массив с числом фильтров (ядер свертки) для каждого сверточного слоя; каждый фильтр извлекает определённые признаки.
* **kernel\_sizes**: Массив размеров фильтров (ядер свертки), задаваемых как одно число (для квадратных фильтров, например 3 означает 3x3) или кортеж.
* **pool\_sizes**: Массив размеров окон для операции подвыборки (MaxPooling), задаваемых числом или кортежем (например 2 или (2,2)).
* **activations**: Массив функций активации для каждого слоя без учета выходного слоя, из возможных: ['relu', 'leaky\_relu', 'sigmoid', 'softmax', 'softplus', 'softsign', 'exponential', 'swish'].
* **optimizer**: Функция оптимизации, выбираемая из ['sgd', 'rmsprop', 'adam', 'adadelta', 'adamax', 'nadam', 'ftrl'].
* **train\_percentage**: Процент тренировочной выборки от общего набора данных (число от 0 до 100).
* **test\_percentage**: Процент проверочной выборки от общего набора данных (число от 0 до 100).
* **user\_name**: Имя пользователя, под которым ведется обучение.
* **dataset\_name**: Название датасета, используемого для обучения.
* **trained\_model\_name**: Имя, которое будет присвоено обученной модели.

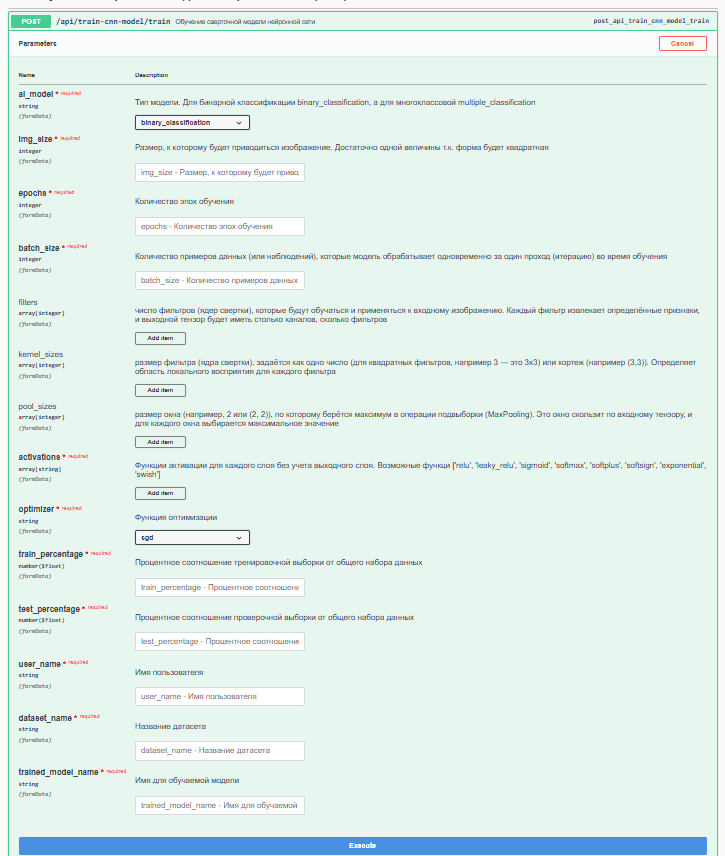


Рисунок .1 – Метод для обучения сверточной нейронной сети

После выполнения запроса в ответе будет предоставлен .zip архив, доступный к скачиванию:

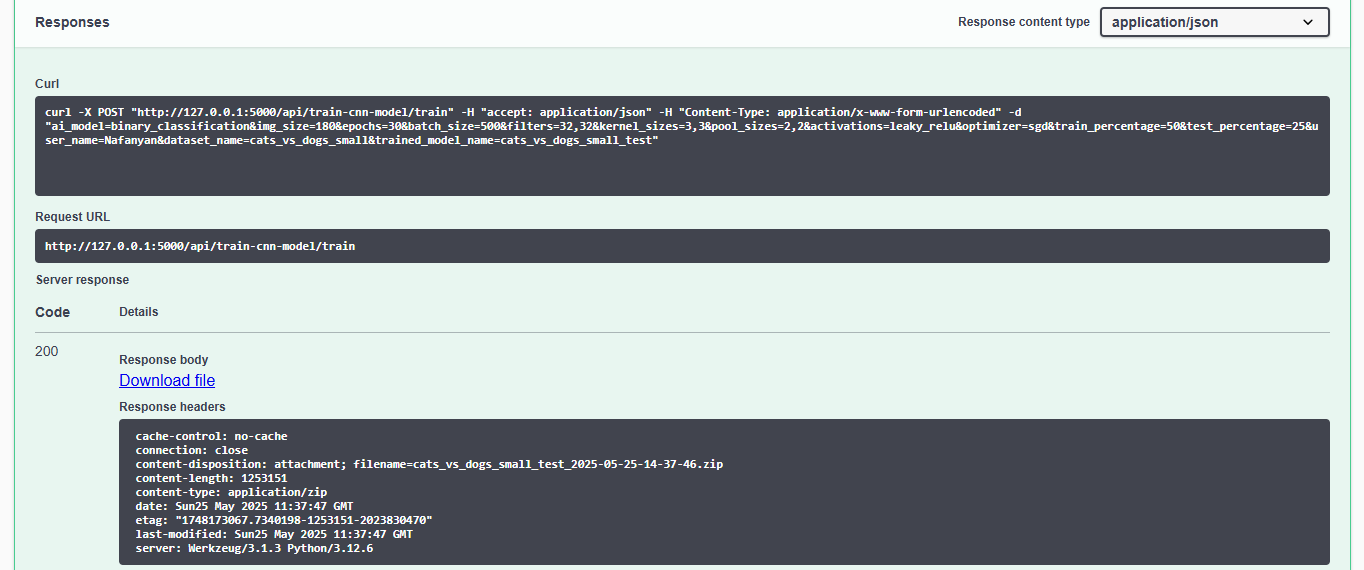


Рисунок .2 – Успешный ответ сервера на обучение сверточной нейронной сети

## Описание примера работы с результатами работы программы

Результатом работы программы являются .zip архив, который содержит в себе:

* Графики изменения точности и потерь в процессе обучения и проверки;
* Обученную модель;
* Пример приложения для проверки обученной модели на имеющихся данных.

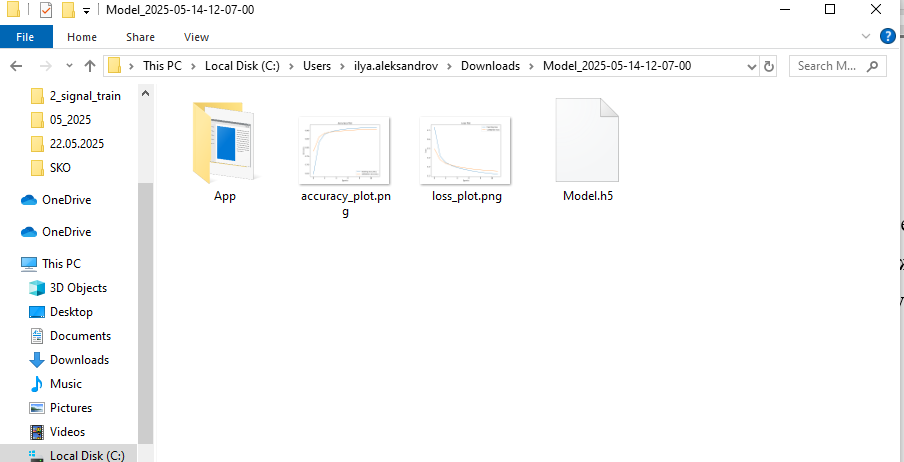


Рисунок .1 – Результат работы программы

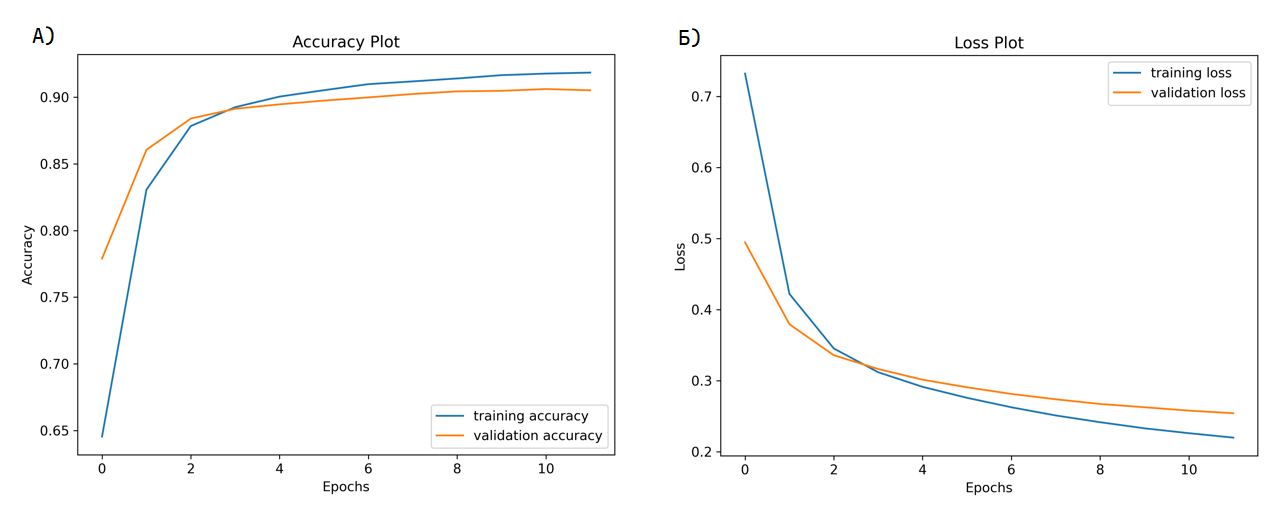


Рисунок .2 – Примеры графиков изменения точности и потерь модели нейронной сети: а) график изменения точности в процессе обучения и проверки; б) график изменения потерь в процессе обучения и проверки

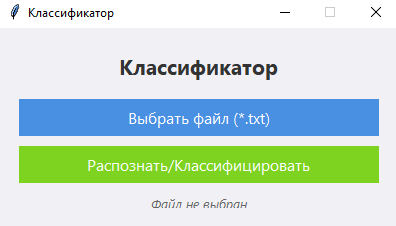


Рисунок .3 – Приложения для проверки качества работы линейной нейронной сети

Для дальнейших разработок и интеграций обученной модели в различные системы, необходимо использовать файл с расширением .h5, который содержит в себе все расчитанные весовые коэффициенты.

# **АПРОБАЦИЯ**

## Обнаружение сигнала

Для оценки эффективности алгоритма обнаружения сигнала была сформирована экспериментальная выборка, включающая эталонный полезный сигнал (Рисунок 3.1.2) и шумовые помехи (Рисунок 3.1.3) с различными уровнями Гауссовского шума, наложенного на полезный сигнал.

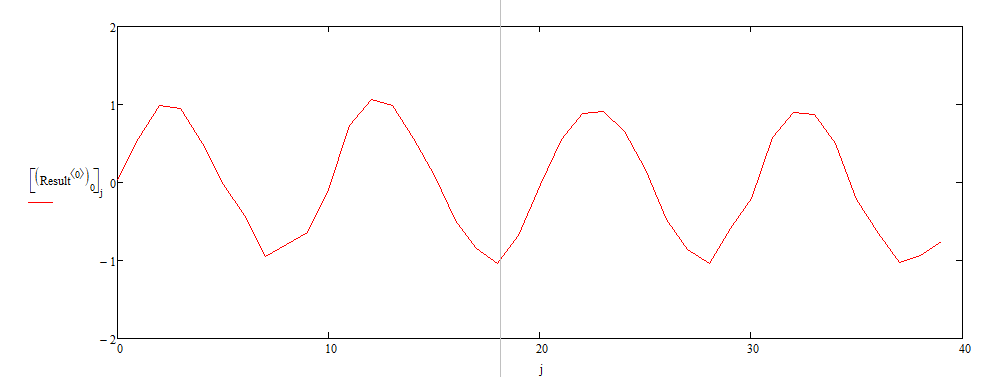


Рисунок 2.3.4. – Полезный сигнал в виде синусоиды

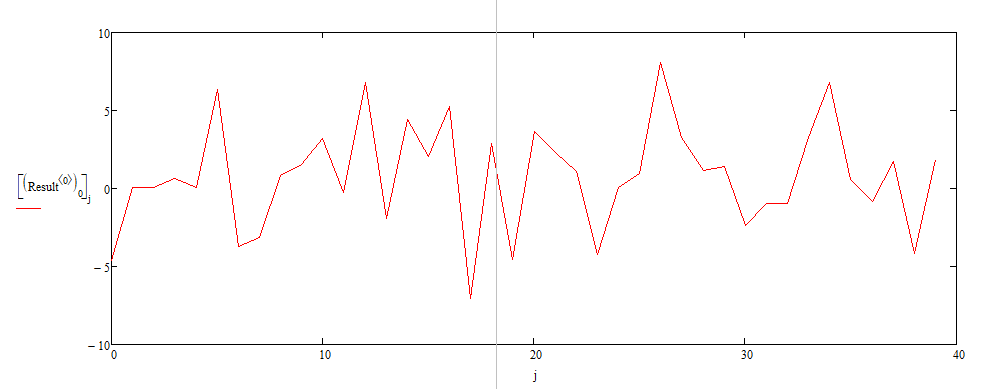


Рисунок .2 – Шум

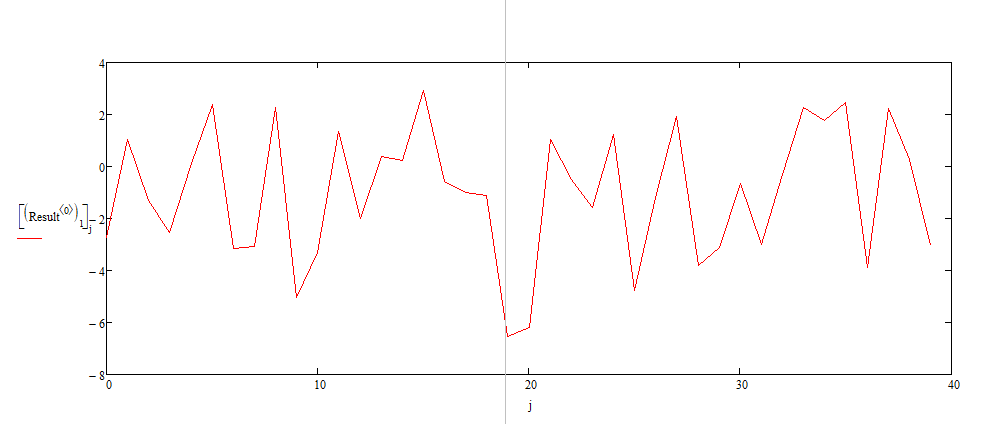


Рисунок .3 – Сигнал, совмещенный с Гауссовским шумом при СКО = 3

После проведения тестирования на обнаружения сигнала во входных данных в зависимости от их зашумленности были определены вероятности правильного обнаружения, ложной тревоги и пропуска:

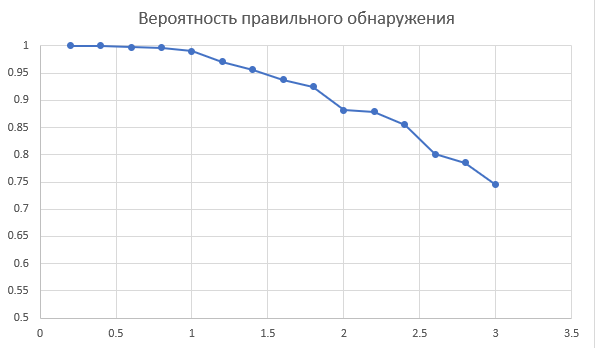


Рисунок .4 – Вероятность правильного обнаружения сигнала в зависимости от величины СКО накладываемого шума



Рисунок .5 – Вероятность ложной тревоги в зависимости от величины СКО накладываемого шума

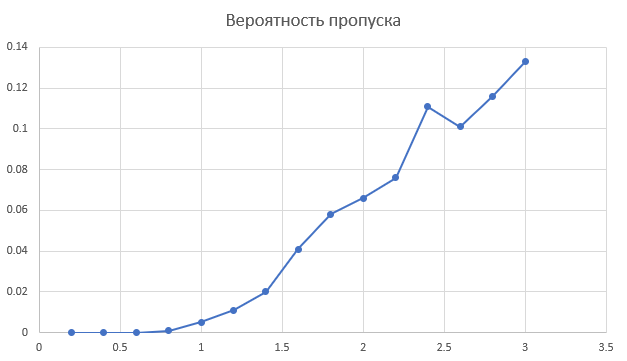


Рисунок .6 – Вероятность пропуска сигнала в зависимости от величины СКО накладываемого шума

## Распознавание сигнала

Для проверки решения задачи распознавания в исходную выборку был добавлен ещё один вид сигнала (Рисунок 3.2.1), который тоже намеренно искажался при помощи Гуссовского шума (Рисунок 3.2.2).

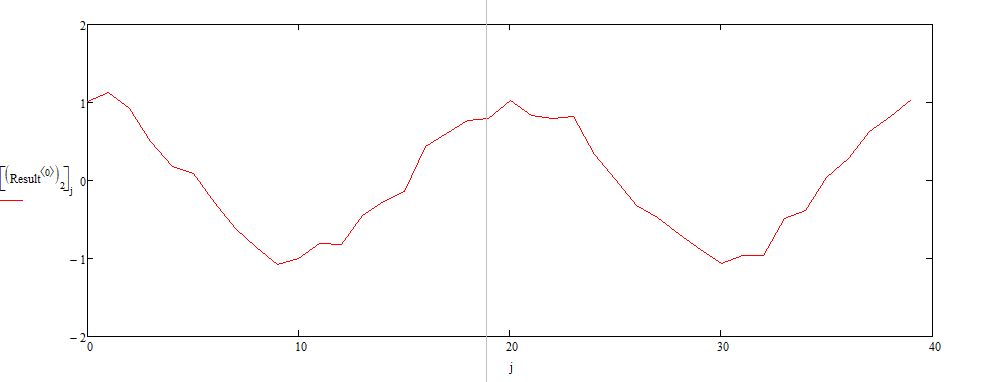


Рисунок 2.3.4. – Полезный сигнал второго класса

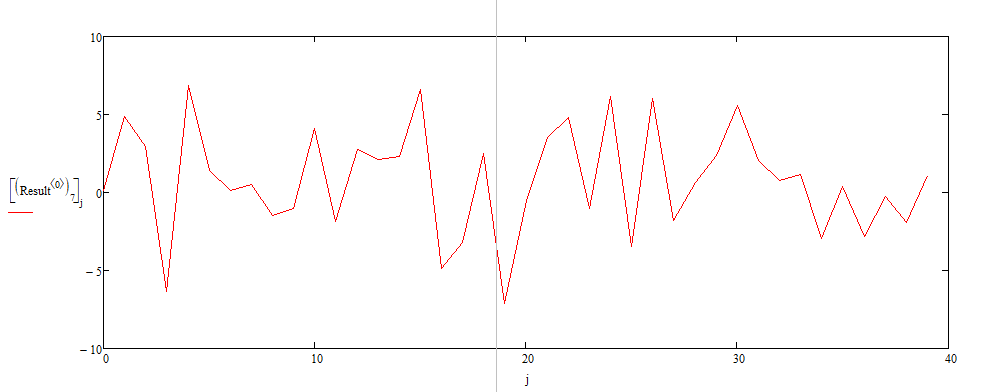


Рисунок .2 – Сигнал второго класса, совмещенный с Гауссовским шумом при СКО = 3

После проведения тестирования на распознавание сигнала во входных данных в зависимости от их зашумленности были определены вероятности правильного обнаружения, ложной тревоги и пропуска:

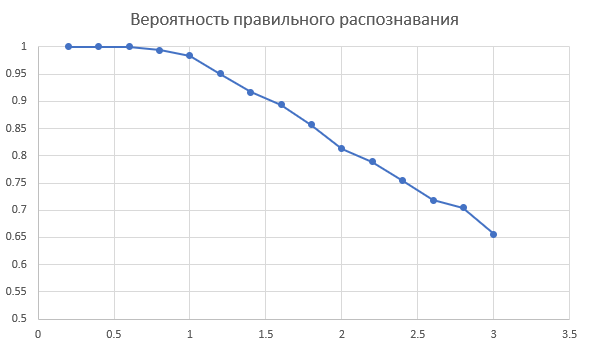


Рисунок 2.3.4.3 – Вероятность правильного распознавания сигнала в зависимости от величины СКО накладываемого шума



Рисунок 2.3.4.4 – Вероятность ложной тревоги в зависимости от величины СКО накладываемого шума

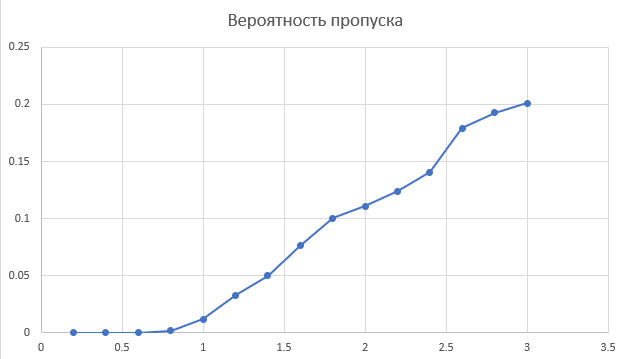


Рисунок 2.3.4.5 – Вероятность пропуска сигнала в зависимости от величины СКО накладываемого шума

# **ИНСТРУКЦИЯ К ПРИМЕНЕНИЮ**

Для развертывания приложения на своем устройстве необходимо иметь следующее программное обеспечение:

* Git – для дальнейшей разработки и полу-автоматической установки обновлений;
* Docker – для автоматического развертывания приложения.

### Установка необходимых зависимостей

Последнюю версию ПО Git можно скачать с сайта <https://git-scm.com/downloads> для любой операционной системы. После скачивания исполняемого файла, необходимо установить приложение на рабочее устройство.

Последнюю версию ПО Docker можно скачать с сайта <https://www.docker.com/products/docker-desktop/> для любой операционной системы. После скачивания исполняемого файла, необходимо установить приложение на рабочее устройство.

### Установка приложения

Код приложения находится в облачном хранилище Github по ссылке: <https://github.com/Nafanyan/AI_Server>. Для скачивания с помощью системы Git необходимо открыть командную консоль, перейти в директорию, удобную для расположения приложения и ввести команду: «git clone [https://github.com/Nafanyan/AI\_Server»](https://github.com/Nafanyan/AI_Server)

Если нет необходимости в дальнейшей разработке и удобном поддержании обновлений приложения, то можно скачать .zip архив с кодом и разархивировать его в удобное место:

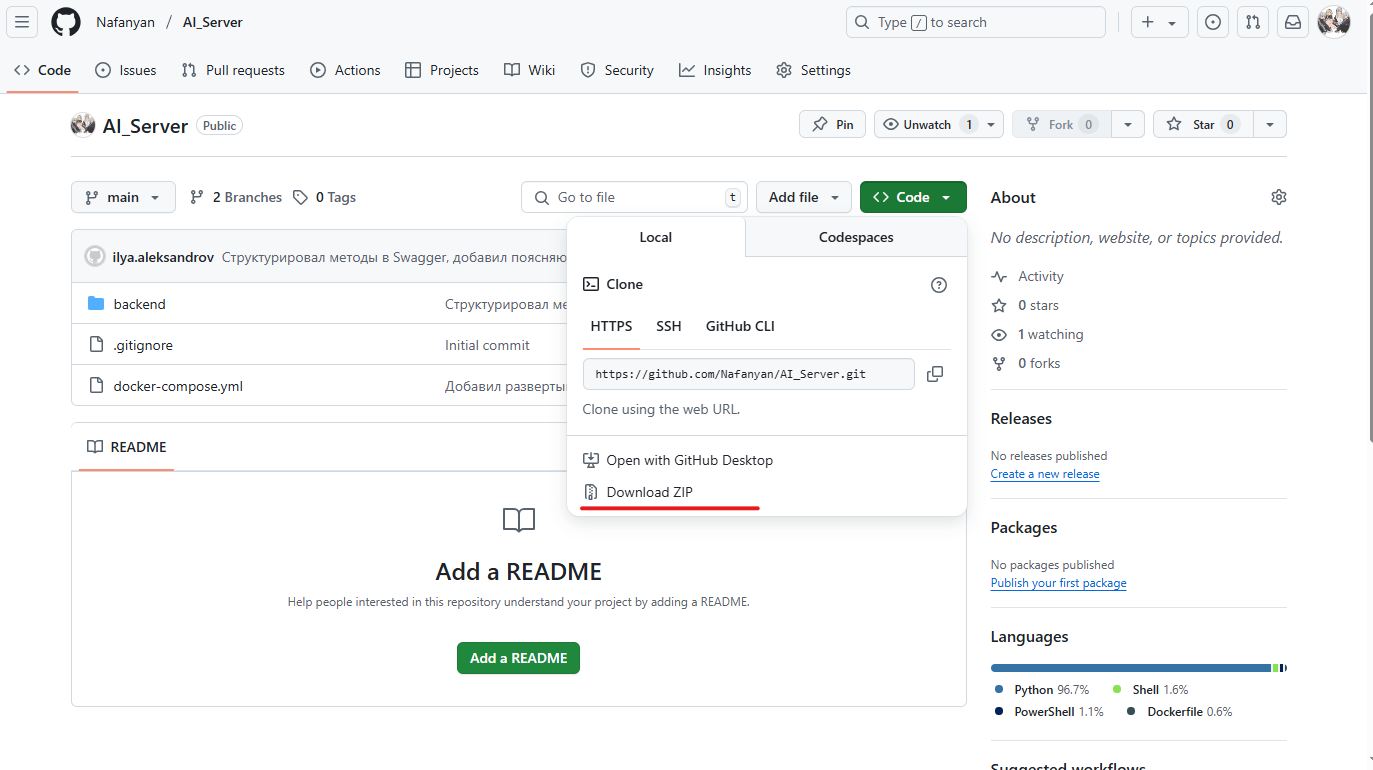


Рисунок .1 – Хранилище Github с кодом приложения

### Запуск приложения

Для запуска приложения необходимо настроить Docker на работу с Linux контейнерами.

Для запуска приложения необходимо через консоль перейти в корень проекта и запустить сборку образа через команду «docker-compose build».

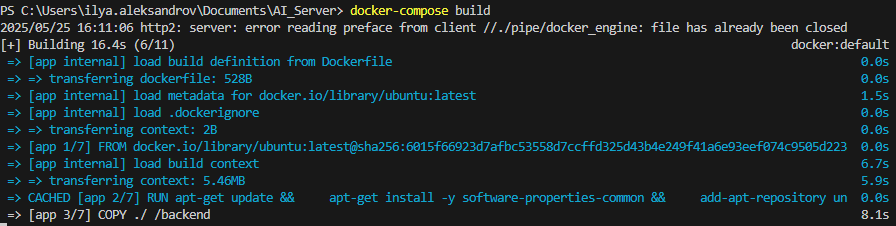


Рисунок 4.1.3.1 – Запуск команды docker-compose build

После завершения команды необходимо создать контейнер и запустить его через команду «docker-compose up». После завершения команды в консоль выведется сообщение о запуске сервера:

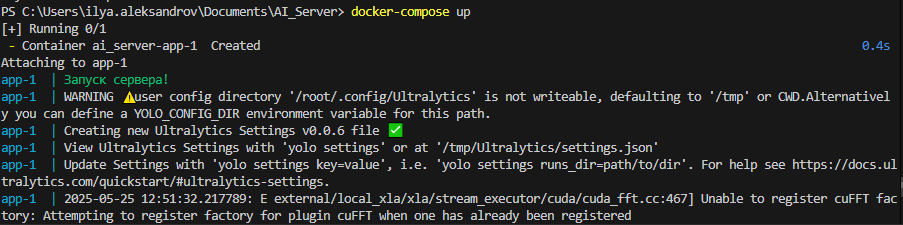


Рисунок 4.1.3.2 – Запуск команды docker-compose up

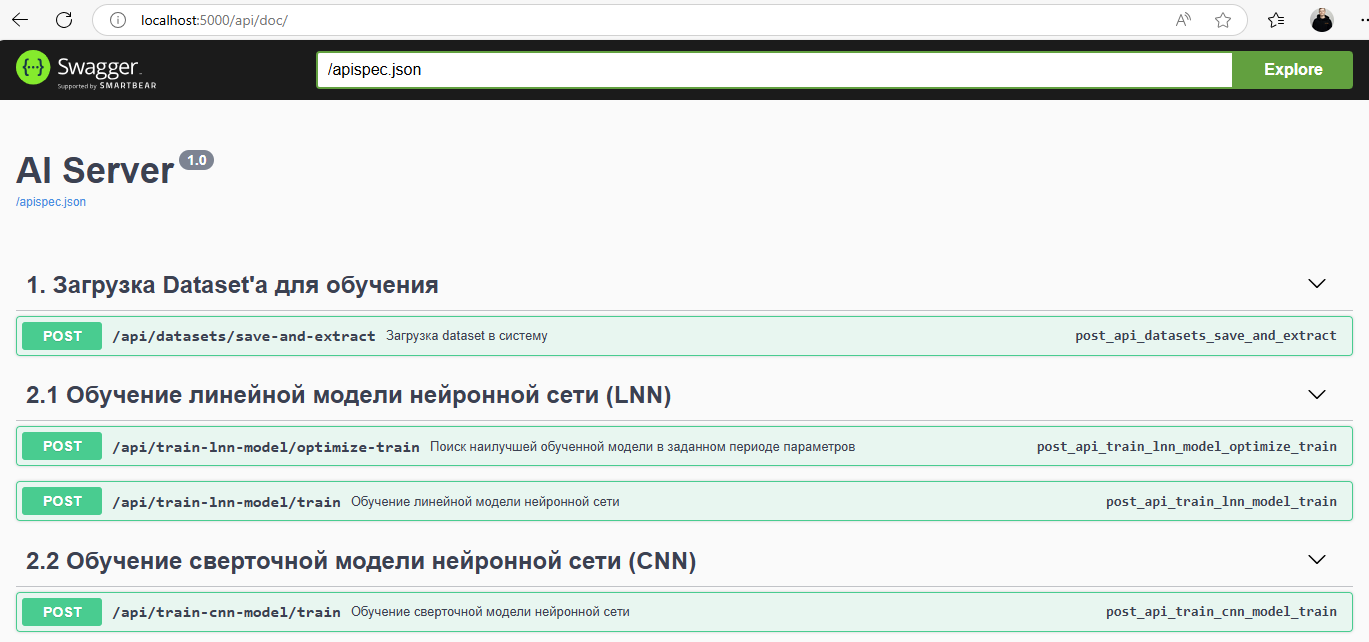


Рисунок . – Запущенное приложение, открытое через браузер

# **ЗАКЛЮЧЕНИЕ**

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Тисленко В.И. Статистическая теория радиотехнических систем: Учеб. пособие. – Томск: Томский государственный университет управления и радиоэлектроники, 2003. – 153 с.
2. Тихонов В.И. Статистическая радиотехника. 2е изд., перераб. и доп. – М.: Радио и связь, 1982. – 624 с.
3. Яневич Ю. М. Задачи приема сигналов и определения их параметров на фоне шумов учебное пособие / Ю.М. Яневич; Санкт-Петерб. гос. ун-т. – Санкт-Петербург : С.-Петербургский государственный университет, 2004. – 86 с.
4. Осадченко, В. Х. Фильтры высоких и низких частот : [учеб.-метод. пособие] / В. Х. Осадченко, Я. Ю. Волкова, Ю. А. Кандрина; [под общ. ред. В. Х. Осадченко] ; М-во образования и науки Рос. Федерации, Урал. федер. ун-т. – Екатеринбург : Изд-во Урал. ун-та, 2015. – 80 с.
5. Головко, В. А. Нейросетевые технологии обработки данных : учеб. пособие / В. А. Головко, В. В. Краснопрошин. – Минск : БГУ, 2017. – 263 с.
6. Ширман Я. Д. Разрешение и сжатие сигналов. – М.: Сов. радио, 1974. – 360 с.
7. Сейдж Э., Мелс Дж. Теория оценивания и её применение в связи и управлении. – М.: Связь. - 1976. 496 с.