МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
И.о. заведующего кафедрой
канд. физ.-мат. наук, доц.
_____ А.В. Письменский
03. 06. 2024 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

Разработка программы с графическим интерфейсом для решения обратных коэффициентных задач теории пороупругости нейросетевыми методами

Работу выполнил _____ О.Ю.Виноградова
(подпись)

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность Технология программирования

Научный руководитель
канд. физ-мат. наук _____ С.И. Фоменко
(подпись)

Нормоконтролер
преподаватель _____ Е.В. Горбачева
(подпись)

Краснодар
2024

РЕФЕРАТ

Выпускная квалификационная работа 61 с., 31 рис., 21 источн.

ЗАДАЧА ПОРОУПРУГОСТИ, НЕЙРОСЕТИ, PYTHON, PYTORCH, ИНТЕРФЕЙС

Целью дипломной работы является изучение теории пороупругости и реализация программного комплекса для решения обратных коэффициентных задач нейросетевыми методами. Необходимо подобрать оптимальную архитектуру, реализовать интерфейс для удобного взаимодействия с программой и осуществить оценку работы нейросети.

В ходе исследования были изучены существующие исследования и разработки на схожие темы и анализ показал, что в данной сфере актуальна дальнейшая работа на основе нейросетевого анализа. Освоены основы теории пороугругости и нейросетевых алгоритмов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире технологического прогресса и научных открытий одним из ключевых направлений является разработка методов и технологий для анализа и прогнозирования свойств материалов. Особенно важным в этом контексте является понимание пористости материалов, которая играет решающую роль в их механических свойствах и способности к взаимодействию с окружающей средой.

В данной работе рассматривается проблема определения пористости материалов с использованием нейросетевых методов. Эта проблема имеет важное практическое значение во многих областях, таких как геология, строительство, нефтегазовая промышленность и другие. Более точное определения степени порообразования в веществах открывает широкие возможности для более продуктивного применения данных материалов в различных направлениях и улучшению уровня качества финального продукта.

Целью данной работы является разработка программы, имеющей графический интерфейс, которая направлена на устранение проблем с решением обратны коэффициентных задач теории пороупругости при помощи методов, применяемых нейросетями.

Результаты, полученные по итогу проведённой работы, могут быть использованы в разных областях науки и промышленности для повышения уровня качества процесса анализа материалов и улучшению эффективности практического применения веществ, обладающих пористой структурой. Стоит подметить, что также результаты исследования могут быть направлены на распространение нейросетевых методов в решении задач, с которыми сталкиваются инженеры.

Можно сказать, что таким образом, данная работа является актуальной и имеет практическую значимость, также она может быть полезна для специалистов, занимающихся анализом и прогнозированием свойств материалов.

1 Анализ актуальности

Актуальность тему исследования формируется из его её значимости в настоящем и её потенциальном вкладе в науку на теоретическом и практических уровнях в будущем. Анализ пористости материалов и их механических свойств может найти огромное количество применений в разных отраслях. Изучения порообразоности веществ является одним из ключевых факторов, влияющих на повышение качества конечного продукта, также способствует развитию новых технологий в данной сфере.

Работы, связанные с этой тематикой, считаются очень важными в современном мире. Пористость материалов оказывает огромное влияние на их свойства, такие как физические, механические и химические.

1.1 Сферы применения

Работа над формированием новых методов прогнозирования характеристик веществ обладает крупным значением в формировании практической базы в различных областях.

Стоит начать с того, что без данных исследований значительно замедляется развитие строительства и гражданской инженерии в целом. Без должных знаний о свойствах материалов, принимающих основное участие в процессе возведения зданий, таких как бетон или кирпич, нельзя точно рассчитать необходимую пористость для обеспечения водонепроцаемости и прочности конструкции.

Также нельзя не упомянуть и про нефтегазовую промышленность, в данной сфере работа над прогнозированием характеристик веществ влияет на эффективность к заполнению и добыче, помимо этого важно осветить, что пористость помогает учёным в работе над оценкой запасов углеводородов и последующей планировке добычи [21].

Кроме того, в качестве примера стоит привезти и исследования, проводимые в сфере геологии, специалисты используют информацию о порообразности горных пород для определения их состава, а также структуры и истории их образования. С помощью данной информации учёные могут лучше изучить геологические процессы, происходящие на месте работ и историю развития региона в целом [20].

Рассматривая преимущества проведения работ над изучением пористости, можно привезти в качества иллюстрации такую область, как агрономия и экология. В настоящие время экологи используют порообразование почвы для регулировки влаго- и воздухопроницаемости, а также определению доступности питательных веществ для растений. Для сельского хозяйства данные параметры помогают значительно сократить расходы на земледелие.

Освещая данную тематику, нельзя не упомянуть про медицину, в этой сфере пористость материалов участвует в изучении костной ткани, играя важную роль при проектировании имплантов и оценке их совместимости с тканями организма пациента [19].

Примеры, представленные выше, демонстрируют важность работ над новыми методами прогнозирования характеристик веществ, в частности, их порообразований.

Тем не менее, необходимо рассмотреть и недостатки, связанные с методами анализа пористости материалов в настоящие время. Стоит начать с того, что способы, которые применяются традиционно в данной сфере, являются достаточно трудоёмкими, требующими дорогостоящего оборудования, а также могут быть недостаточно точными. В связи с этим, учёные начинают обращаться к нейросетям для проведения анализа пористости почвы, так как они обладают способностью к обучению на предоставленной информации и дальнейшему поиску закономерностей на её основе, что может улучшить точность исследований и снизить затраты времени и ресурсов на их проведение.

В целом, можно сказать, что в последние годы наблюдается тенденция к повышению интереса к применению искусственного интеллекта в различных областях науки и техники. Она связана с ростом вычислительной мощности компьютеров и повышением уровня развития новых алгоритмов обучения нейронных сетей.

1.2 Степень решенности проблемы

При сборе информации о данной сфере упор бля сделан на достаточно глубокие исследования, которые освещают физические свойства материалов. Это дало толчок для понимания основ пороупругости веществ, а также специфике проблем в данной области, связанных с распространением волн. Помимо этого, в процессе подборки сведений были выявлены основные условия и ситуации, оказывающие влияния на процессы, происходящие в упругой и пористой среде. Вся собранная информация помогла в постановки точной задачи и выборе методов для её решения с применением искусственного интеллекта [1, 2, 13].

В современном мире вопрос процесс, имеющий волновую структуру и проходящий через пористую и в то же время пористую среду, изучен учёными не в полной мере. Хотя сейчас существует немало работ в данной области среди них совсем немного трудов, освещающих применение нейросетей в этой сфере. Большинство исследований узконаправлены и ограничены по своему объему и глубине анализа. Они часто фокусируются на конкретных аспектах или задачах, не охватывая всего спектра возможных волновых процессов [3, 4]. Также стоит добавить, что большая часть таких текстов представлена на английском языке, что значительно усложняет работу российских учёных, не владеющих им на достаточном уровне.

Проведение работ, направленных на изучение волновых процессов в средах, которые являются упругими и пористыми, с использованием искусственного интеллекта являются довольно-таки перспективным

направлением, требующим дальнейших исследований. Благодаря нейросетевым методам станет возможным создание точных моделей и проведение более точного анализа веществ.

Можно сказать, что активное развития науки в данном направлении имеет большое значение для различных сфер промышленности. Разработки новых методов оценки материалов, что позволит значительно упростить работу специалистам

2 Постановка задачи

Как упоминалось выше: традиционные методы анализа пористости материалов, например оптическая микроскопия или компьютерная томография, считаются дорогостоящими и могут требовать специального оборудования. На сегодняшний день данные способы оценки являются стандартом для неразрушающего контроля характеристик веществ без физических манипуляций с ними, такими как демонтаж.

Ранее уже говорилось, что одним из самых эффективных способов оптимизации анализа порообразности материалов является применение искусственного интеллекта. С их помощью специалисты могут быстро обрабатывать большие объемы информации, после чего выявлять их сложные закономерности.

Главной целью данной работы считается создание программы с интуитивно понятным графическим интерфейсом, которая будет направлена на решение обратных коэффициентных задач теории пороупругости с использованием нейросетевых методов.

На пути к достижению данной цели были поставлены следующие задачи:

1) изучить теоретический материал, связанный с пороупрогостью и искусственным интеллектом;

2) рассмотреть данные, затрагивающие архитектуры программы и методы обработки информации;

3) осветить процесс формирования графического интерфейса, созданного для удобного взаимодействия с приложением;

4) оценить качество обучения нейросетевых моделей на уже имеющихся данных и их точности;

5) провести тестирования и оптимизацию программы для повышения её эффективности.

Решение этих задач позволит создать проект, которые будет направлен на продуктивный анализ пористости веществ с использованием нейросетевых методов.

2.1 Задачи пороупругости

Стоит начать с того, что задачи, направленые на пороупрогость, занимаются изучением механических свойств материалов, обладающих пористой структурой, которые имеют такие характеристики как упругость и проницаемость, подразумевают под собой процесс моделирования взаимодействия между жидкостью или газом, заполняющим поры вещества, и самого материала. Этот метод позволяет учёным спрогнозировать поведение того или иного вещества в различных условиях.

В данной дипломной работе рассматривается обратная задача пороупругости. Для ряда прямых проблем причины известны, и необходимо найти следствие. В качестве источников могут выступать сама математическая модель, начальные значения, коэффициенты дифференциальных операторов, граничные условия, геометрия области.

В различных областях науки, а также социальных моделях и других процессах принцип функционирования объектов задачи обобщается следующим образом:

X обозначает входные данные.

Y подразумевает под собой систему обработки запроса, характеристики и структуры объектов.

Z является выходными данными.

Тогда, прямой задачей можно считать необходимость определения выхода Z, если известны вход X и оператор Y.

На основе этих соображений, в качестве обратного алгоритма нужно поставить задачу идентификации начального объекта изучения. Тогда

неизвестными могут быть оператор Y или часть входной информации X или Y и X.

Для такого вида вопросов известны следствия, требуется найти причины и определить их по некоторой дополнительной информации об объектах исследования.

Задача, рассматриваемая в тексте, направлена на определение пористости материала, которое основывается на данных, полученных от волн. Данный вопрос является обратным, потому что уже известные характеристики источников информации, такие как медленность, подразумевающая обратную скорость волны и частоту, но не известна пористость материала.

Для нахождения ответа на поставленный выше вопросы были применены нейросетевые методы, которые открывают возможности для анализа полученных данных и дальнейшего прогнозирования свойств вещества.

Важно подметить, что теоретическая модель установки является системой, направленное на генерацию и регистрацию упругие волны, которые проходят вкось материалы, обладающие пористой структурой.

Во время измерений был использован источник таких сигналов, который способен формировать волны, имеющие направления в сторону вещества сверху. Под первым материалом находится ещё один, (полупространство, ограниченное сверху первым образцом). В результате, в среде возбуждаются колебания с определёнными скоростям, регистрируемые с помощью сенсора, который располагается на заранее известном расстоянии от источника сигналов. Уровень скорости фиксируются при помощи регистрации времени прихода волны на датчик и дистанции, о которой уже есть информация. После чего специалисты получают данные о медленностях волн, величина, которая являются противоположностью темпа. Сведенья о частоте сигналов представляет собой задаваемую характеристику источника. На рисунке 1 показана модель установки:

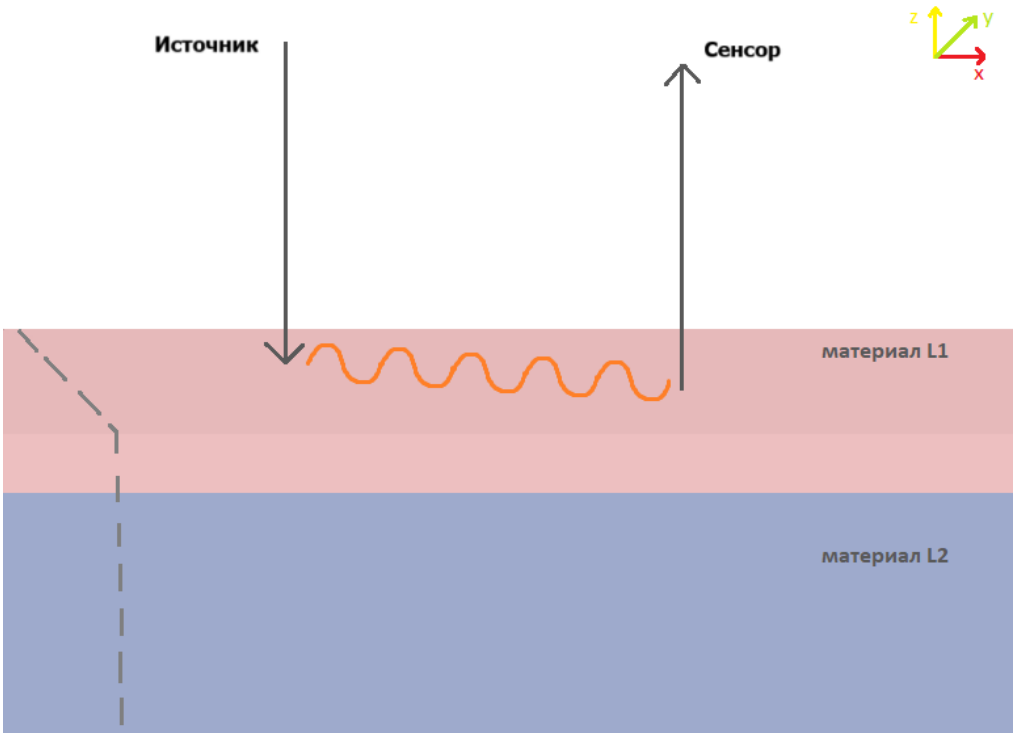

Рисунок 1 — Модель установки

2.2 Датасет

Освещая датасет, важно упомянуть, что для получения данных с экспериментальной установки была использована программа на языке программирования Fortran, которая занималась генерацией значения пористости двух материалов в диапазоне от 0.01 до 0.35, что представляет собой значения объёма пор по отношению ко всей вместимости среды, важно подметить, что после 0.36 среда начинает разрушаться из-за слишком большого количества порообразований, и значения точек на оси x и оси y на графике для пары материалов.

На рисунках 2 и 3 продемонстрированы зависимости медленностей волны от её частоты для разных материалов:

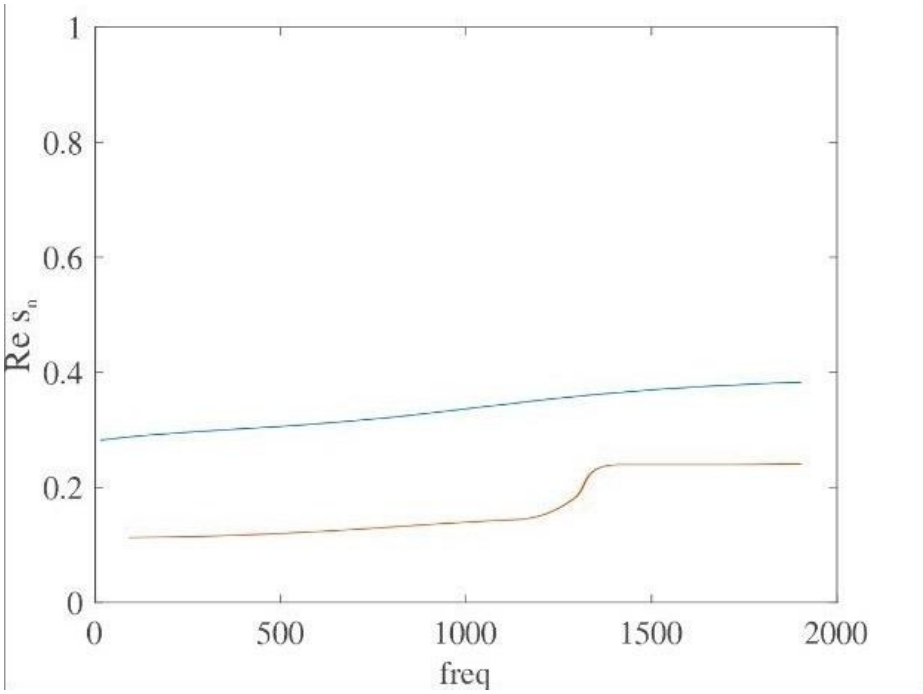

Рисунок 2 – График зависимости медленности волны от её частоты для пористостей $\varepsilon_1 = 0.2$ и $\varepsilon_2 = 0.1$

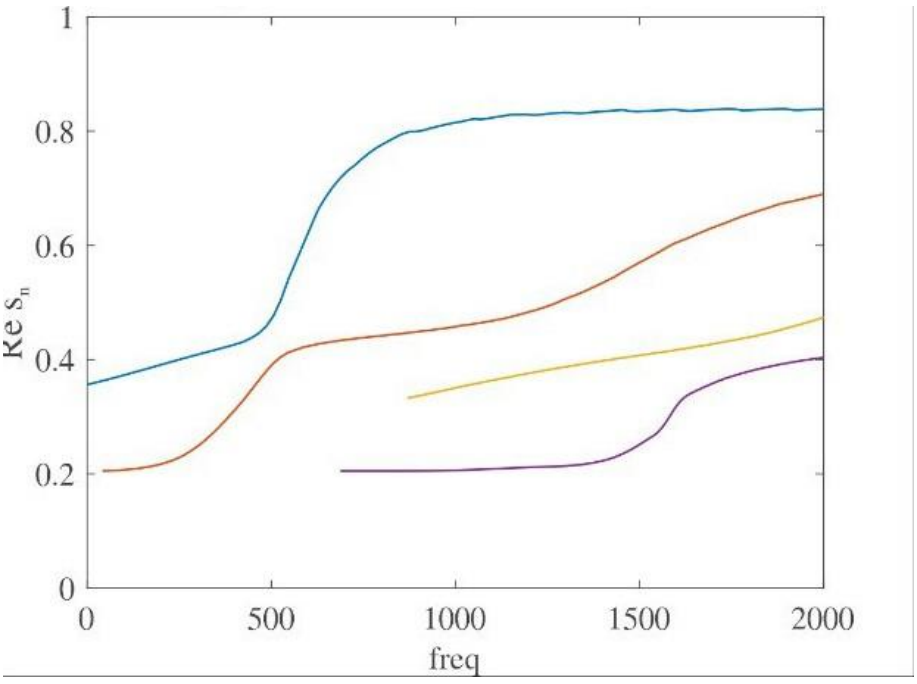

Рисунок 3 – График зависимости медленности волны от её частоты для пористостей $\varepsilon_1 = 0.35$ и $\varepsilon_2 = 0.2$

Для разработки нейросети формируется две выборки: обучающая и тестовая. Первая из них подразумевает под собой набор записей таких характеристик, как медленность и частота волн, полученных в результате проведения экспериментов. Вторая представляет собой информацию, которая не была применена в процессе обучения искусственного интеллекта и служит лишь для оценки эффективности обобщающей способности нейронной сети.

На рисунке 4 представлен пример выборки, которую выдаёт программа:

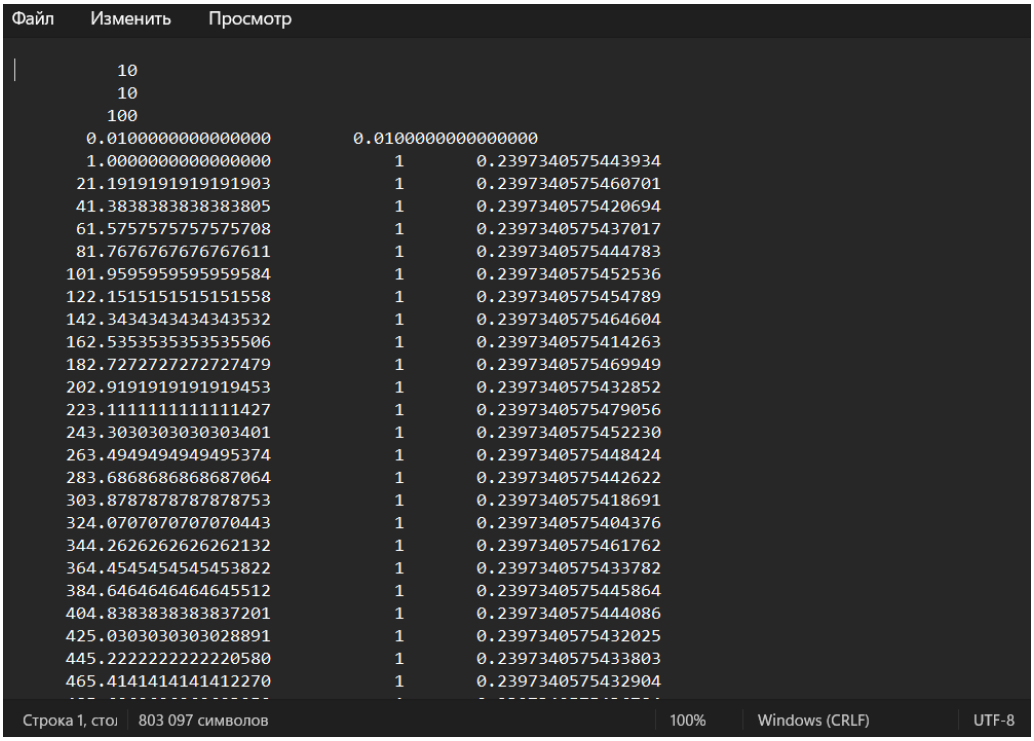

Рисунок 4 — Пример обучающих данных

Первые три строки, продемонстрированные на скриншоте, являются параметрами, которые может менять пользователь, например количество первых пористостей, количество вторых порообразований и число промежуточных частот. Далее представлена пара пористостей, ниже показана точка по оси координат x, а также можно увидеть количество соответствующих ей значений по e и сами эти числа. Перед использованием данной информации проводилась её предварительная очистка от шумов и приведение к нужному формату.

2.3 Нейронные сети

Нейронные сети — это математические модели, вдохновленные работой человеческого мозга, способные обучаться на данных и делать прогнозы или принимать решения на основе этого обучения. Они состоят из нейронов, объединенных в слои, и имеют способность обрабатывать и анализировать сложные данные, выявляя в них закономерности.

Основные принципы работы нейронных сетей включают передачу сигналов между нейронами, веса связей между нейронами, функции активации и обучение с помощью методов оптимизации. Например, сверточные нейронные сети хорошо подходят для анализа изображений, а рекуррентные нейронные сети эффективны для анализа последовательных данных, таких как временные ряды.

В контексте анализа пористости материалов нейронные сети могут быть использованы для обработки данных, полученных измерениями волн, проходящих через материалы. Нейросетевые методы позволяют обучить модель на этих данных и использовать ее для предсказания пористости материала без необходимости проведения дорогостоящих и сложных экспериментов.

Одной из особенностей нейронных сетей является их способность к обучению на больших объемах данных и выявлению сложных зависимостей между переменными. Это позволяет им делать более точные прогнозы, чем традиционные статистические методы, основанные на заранее заданных моделях.

Для применения нейросетевых методов в анализе пористости материалов необходимо разработать специализированные архитектуры нейронных сетей, учитывающие специфику задачи. Также требуется провести обширные экспериментальные исследования для оптимизации параметров модели и обучения нейронной сети на различных типах данных.

Как и биологическая нейронная сеть, искусственная состоит из нейронов, взаимодействующих между собой, однако представляет собой упрощенную модель. Так, например, искусственный нейрон имеет намного более простую структуру: у него есть несколько входов, на которых он принимает различные сигналы, преобразует их и передает другим нейронам. Другими словами, искусственный нейрон — это такая функция, которая преобразует несколько входных параметров в один выходной.

У нейрона есть n входов x_i, где $i = 1 \dots n$, у каждого из которого есть вес w_i, на который умножается сигнал, проходящий по связи. После этого взвешенные сигналы $x_i * w_i$ направляются в сумматор, который аггрегирует все сигналы во взвешенную сумму. Эту сумму также называют S.

$$S = \sum_{i=1}^{n} x_i * w_i$$

Просто так передавать взвешенную сумму S на выход достаточно бессмысленно — нейрон должен ее как-то обработать и сформировать выходной сигнал. Для этих целей используют функцию активации, которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона. Функция активации обозначается $f(S)$. Таким образом, выходом искусственного нейрона является $f(S)$ [5].

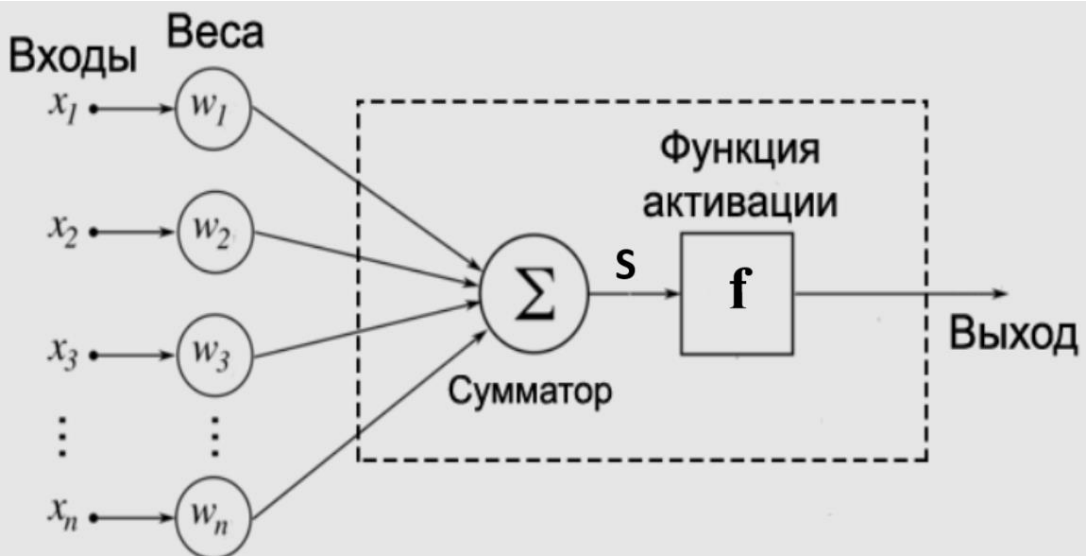

Рисунок 5 – Схема искусственного нейрона

Для разных типов нейронов используют самые разные функции активации, но одними из самых популярных являются [6]:

– Пороговая функция. Проводя аналогии с биологическими нейронными сетями, то функцию активации можно представить как ступенчатую функцию, нейрон может либо возбудиться, либо нет. Для такой активационной используется математическая формула:

$$f(x) = \begin{cases} 1, x \geq a \\ 0, x < a \end{cases},$$

где a – пороговый параметр.

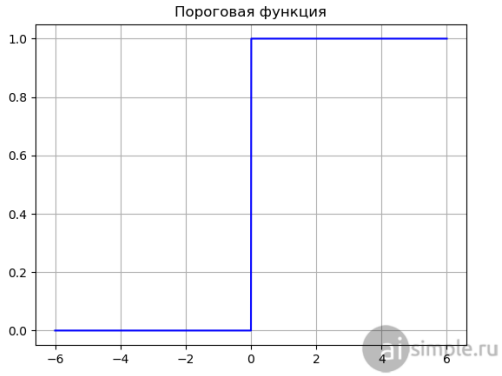

Рисунок 6 – График пороговой функции

– Линейная функция.

$$f(x) = c * x,$$

где c - коэффициент пропорциональности.

Рисунок 7 – График линейной функции с коэффициентом $c = 2$

– Функция ReLU.

$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases}$$

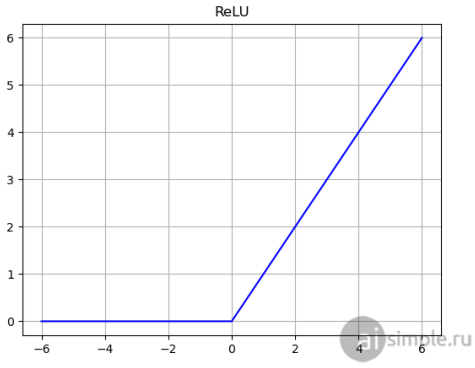

Рисунок 8 – График функции ReLU

– Сигмоидная логистическая функция. Сигмоида очень часто применяется в задачах классификации, поскольку она является гладкой функцией и меняется в диапазоне от -1 до 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Рисунок 9 – График производной сигмоидной функции

– Гиперболический тангенс. Несмотря на название, за этим скрывается лишь модифицированная сигмоида. Поэтому, можно сказать, что все преимущества и недостатки сигмоиды в равной степени относятся и к тангенсу. Различия проявляются лишь в градиенте тангенциальной функции, он выше, это видно из сравнения производных на графике ниже.

$$f(x) = \tan x = \frac{2}{1 + e^{-2x}} - 1$$

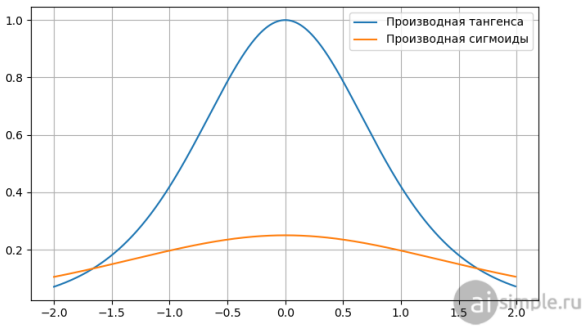

Рисунок 10 – Сравнение графиков функции производной тангенса и производной сигмоиды

Помимо описанных выше формул, существует множество других функций активации, исследователи в области искусственного интеллекта постоянно стремятся найти новые способы улучшения нейронных сетей. Они модифицируют существующие методы обработки входных сигналов и ищут новые подходы. Выбор правильной активационной функции является одной из ключевых задач при построении нейронной сети. Однако нет универсального правила о том, какую функцию следует использовать в конкретной ситуации. Существует лишь опыт, который показывает, что ReLU лучше всего подходит для аппроксимации, в то время как сигмоида и гиперболический тангенс чаще используются в задачах классификации.

При решении сложных задач с использованием многослойных нейронных сетей возможно комбинировать различные функции активации на разных слоях, чтобы достичь наивысшей точности и ускорить процесс обучения.

После выбора архитектуры нейронной сети необходимо правильно настроить параметры в процессе обучения. Обучение нейронных сетей – это процесс, в ходе которого сеть настраивает свои веса и параметры для оптимального выполнения задачи.

Существует два основных типа обучения нейронных сетей: с учителем и без него. Обучение с учителем подразумевает предоставление сети обучающих примеров, где каждый пример сопоставляется с желаемым

выходом. Нейросеть корректирует свои веса таким образом, чтобы минимизировать разницу между своими выходами и желаемыми ответами. Этот тип обучения используется, когда известен требуемый результат, как в задачах классификации и аппроксимации функций [14].

Обучение без учителя не требует предоставления желаемых выходов. Вместо этого алгоритмы обучения нейронных сетей без учителя выявляют внутренние закономерности в данных без явного указания на них. Этот тип обучения применяется для кластеризации, поиска аномалий и создания статистических моделей.

Оба типа обучения имеют свои особенности и применяются в зависимости от конкретной задачи и доступных данных.

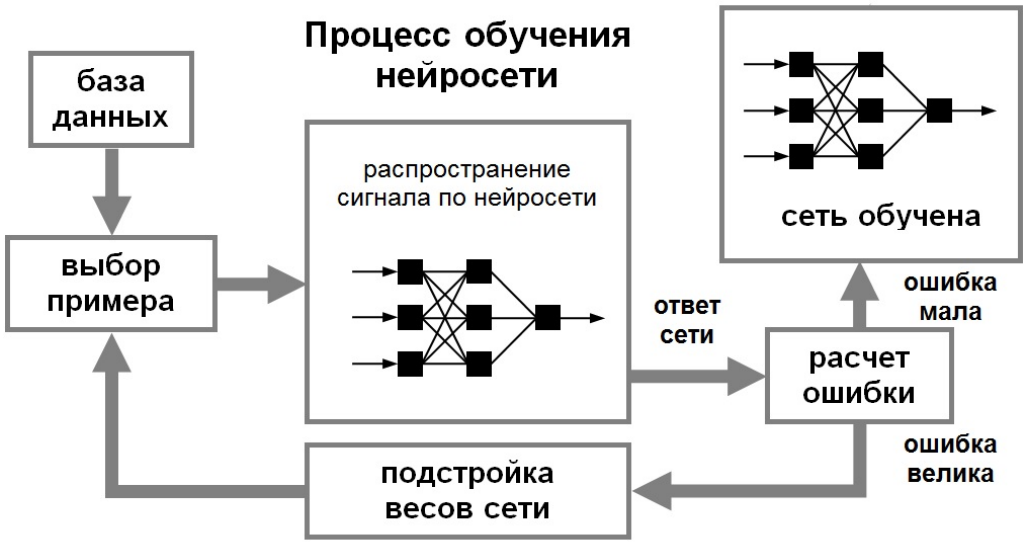

Рисунок 11 – Модель обучения нейросети

И также существует три основных алгоритма обучения нейронных сетей: метод обратного распространения ошибки, метод упругого распространения и генетический алгоритм. В последующей реализации будет использоваться метод обратного распространения ошибки, поэтому кратко обсудим остальные [15].

Метод упругого распространения был создан как более быстрый и удобный способ обучения нейронных сетей по сравнению с методом обратного распространения. Он использует знаки производных для подстройки весовых коэффициентов, при этом обеспечивается быстродействие и эффективность алгоритма. Если производная изменяет свой знак на противоположный, это указывает на слишком большое изменение и сигнализирует об упущении локального минимума. В этом случае вес возвращается к предыдущему значению, а изменение веса уменьшается. Если же знак остается прежним, то изменение веса увеличивается для ускорения сходимости.

Генетический алгоритм обучения – еще один распространенный подход – это обучение нейронной сети генетическим алгоритмом. По своему принципу он схож с эволюционными процессами природы, которые основываются на комбинировании, скрещивании результатов. Завершение алгоритма происходит в тот момент, когда заканчиваются отведенные ему попытки или время на мутацию.

И обратное распространение ошибки, этот метод является ключевым алгоритмом для обучения нейронных сетей. Он позволяет эффективно настраивать веса нейронов сети на основе полученных ошибок предсказания [16, 17].

Процесс обратного распространения ошибки состоит из следующих шагов. Прямой проход: входные данные подаются на вход сети, и значения проходят через все слои от входного до выходного. Каждый нейрон в слое получает взвешенную сумму входных значений (сами входные значения, помноженные на веса), применяет к ней функцию активации и передает результат на следующий слой. Проход продолжается до выходного слоя, где генерируются предсказания сети.

Вычисление функции потерь: сравнивается предсказанный результат с желаемым выходом (целевыми значениями) и вычисляется ошибка или

функция потерь, которая показывает, насколько сильно предсказания отличаются от желаемого результата.

Обратное распространение ошибки: ошибка с выходного слоя распространяется обратно через сеть. Каждый нейрон в слое получает информацию об ошибке и вкладе в нее от нейронов следующего слоя. Затем происходит вычисление градиента функции потерь по весам и нейронов.

Обновление весов: после вычисления градиента функции потерь по весам и смещениям, происходит обновление этих параметров. Обычно применяется градиентный спуск или его модификации, чтобы минимизировать функцию потерь. Веса и смещения корректируются в направлении, противоположном градиенту, чтобы уменьшить ошибку предсказания.

Обратное распространение ошибки позволяет эффективно настраивать веса нейронной сети, обучая ее на обучающем наборе данных и улучшая ее способность к предсказанию целевых значений. Этот процесс повторяется множество раз до достижения оптимальных весов, которые минимизируют функцию потерь и обеспечивают наилучшую производительность модели.

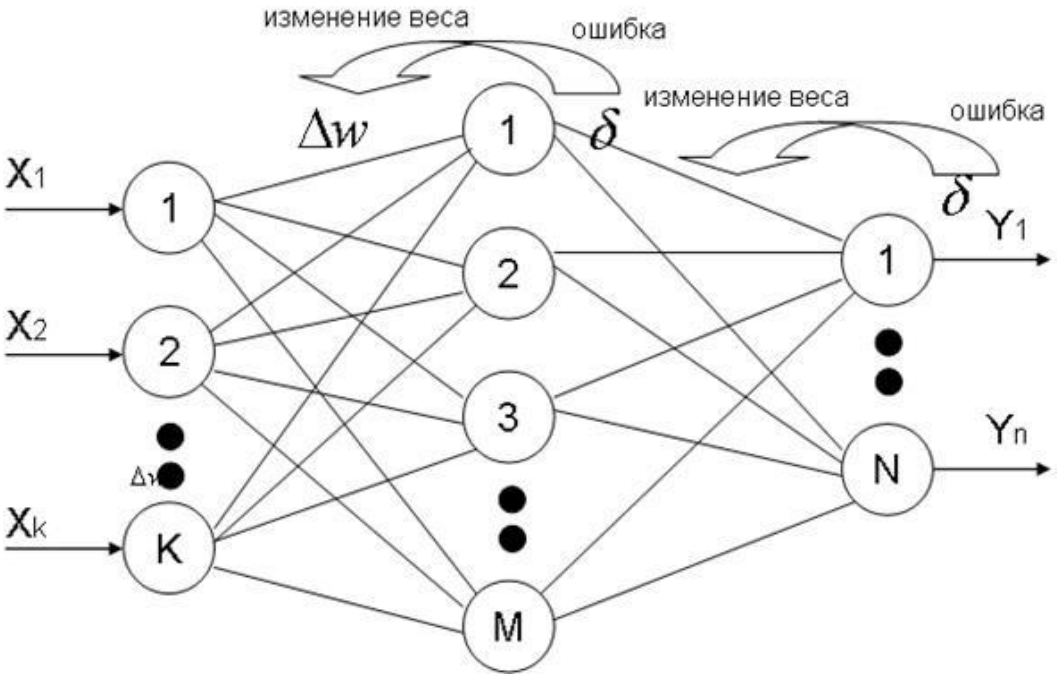

Рисунок 12 – Обратное распространение ошибки

3 Используемые технологии

3.1 Среды разработки

Google Colab.

Google Colab - это облачная среда для выполнения кода на языке Python. Она предоставляет возможность использовать мощные вычислительные ресурсы, включая GPU, для выполнения задач машинного обучения. В моей работе Google Colab использовался для обучения и тестирования модели нейронной сети, так как он обеспечивает быстрый и удобный доступ к вычислительным ресурсам [7].

Google Colab представляет собой бесплатную среду, позволяющий писать код в jupyter notebook. Эта система является облачной, что позволяет вести коллективную работу над одним проектом. С помощью программы можно получить доступ к графическим процессорам GPU и TPU. Их мощности задействуются для анализа искусственного интеллекта и разработки приложений на базе нейросетей.

Colab помогает писать код на Python. Разработка и выполнение осуществляются прямо в браузере. Пользователю нужно лишь предоставить доступ к Google-аккаунту.

С помощью Colab можно использовать в одном файле исполняемый код, html-разметку, изображения. Все эти элементы будут храниться на Google-диске. При этом пользователь может поделиться своими файлами. При помощи настроек доступа можно разрешить просматривать и корректировать данные, а также оставлять комментарии для других разработчиков.

Облачная среда часто используется специалистами по обработке информации и программистами, которые занимаются созданием нейросетей. Colab позволяет решать следующие задачи: сортировка информации, создание визуализаций, машинное обучение, разработка систем для big data, формирование прогнозов, создание руководств.

PyCharm.

PyCharm - это интегрированная среда разработки (IDE) для языка Python, разработанная компанией JetBrains. PyCharm предоставляет широкий набор инструментов для разработки, отладки и тестирования кода. В моей работе PyCharm использовался для разработки и отладки программного кода, использования модели нейросети, а также для создания и настройки графического интерфейса приложения [8].

Среди тех, кто использует язык python эта IDE вторая по популярности после редактора кода Visual Studio Code: как основную её используют 31% разработчиков.

В PyCharm есть все инструменты, чтобы писать, отлаживать и тестировать код. Например, можно быстро исправить программу сразу в нескольких местах, а встроенный форматер приведёт её в соответствие со стандартом PEP 8.

Также IDE позволяет использовать в проектах другие языки программирования, синхронизировать код с системами контроля версий и развёртывать его.

Базовый набор функций PyCharm можно расширять с помощью плагинов, которые позволяют, например, настраивать внешний вид интерфейса и подключать дополнительные инструменты.

3.2 Библиотеки

Следует начать с описанием такой библиотеки, как PyTorch, она представляет собой достаточно эффективный инструмент для машинного обучения с открытым исходным кодом, она используется для создания и дальнейшего обучения нейронных сетей. В этой дипломной работе данная библиотека была применена для формирования модели искусственного интеллекта, загрузки в него информации и прогнозирования пористости материалов. [9].

Как было описано выше PyTorch обладает открытым исходным кодом, который распространяется на безвозмездной основе. Вокруг данного продукты была построена целая экосистема, состоящая из большого количества дополнений, специализирующихся на разных целях.

В качестве фреймворка этот инструмент используется специалистам, ведущими свою деятельность в сфере нейронных сетей на Python, языке программирования, который является одним из самых распространённых в данной отрасли. Конкретно PyTorch ориентирован на такой подвид ML, как глубокое обучение. В данной сфере используются многослойные обучаемые модели. В этой области искусственный интеллект может получать информацию с помощью обработку собственных данных, что открывает возможности для решения широкого спектра задач в то время, как классический вид машинного обучения направлен на работу с моделями, имеющими узкую направленность.[18].

Рассматривая примеры использования PyTorch стоит упомянуть такие отрасли, как распознавание образов на изображениях, компьютерное зрение, также такую технологию, как обнаружение движущихся объектов, поиск закономерностей, анализ данных, в том числе неструктурированных, обработка естественного языка, распознавание речи и машинный перевод, создание машинных описаний для изображений, анализ текстов и поиск в них информации, генерация текстового контента и картинок.

Пример использования данной технологии на практике представлен на рисунке 13:

```
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.fc1 = nn.Linear(100 * 11, 512)
        self.bn1 = nn.BatchNorm1d(512)
        self.fc2 = nn.Linear(512, 256)
        self.bn2 = nn.BatchNorm1d(256)
        self.fc3 = nn.Linear(256, 128)
        self.bn3 = nn.BatchNorm1d(128)
        self.fc4 = nn.Linear(128, 64)
        self.bn4 = nn.BatchNorm1d(64)
        self.fc5 = nn.Linear(64, 2)

        self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.dropout(torch.relu(self.bn1(self.fc1(x))))
        x = self.dropout(torch.relu(self.bn2(self.fc2(x))))
        x = self.dropout(torch.relu(self.bn3(self.fc3(x))))
        x = torch.relu(self.bn4(self.fc4(x)))
        x = self.fc5(x)
        return x
```

Рисунок 13 – Создание архитектуры сети

Scikit-learn (sklearn).

Scikit-learn - это библиотека машинного обучения для языка Python, которая предоставляет простые и эффективные инструменты для анализа данных. В моей работе sklearn используется для нормализации данных и разделения их на обучающие и тестовые выборки [10]. Это один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования или разбивки данных на группы. Sklearn написана на языках Python, C, C++ и Cython. Пример использования в данной работе: нормализация и разделение данных, представленные на рисунке ниже.

```
# Стандартизация признаков
X = StandardScaler().fit_transform(X)

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Рисунок 14 – Нормализация и разделение данных

NumPy — это библиотека для языка Python, добавляющая поддержку больших многомерных массивов и матриц, а также содержащая коллекцию математических функций для работы с этими массивами. NumPy используется для обработки и форматирования данных перед подачей их в нейронную сеть [11].

NumPy не просто работает с многомерными массивами, но и делает это быстро. Интерпретируемые языки производят вычисления медленнее компилируемых, а Python как раз язык интерпретируемый. NumPy же сделана так, чтобы эффективно работать с наборами чисел любого размера в Python. Пример использования в данной работе представлен на рисунке 15:

```
    X.append(np.array(X_block).flatten())  # Преобразуем блок признаков в один вектор
    y.append(y_block)

X = np.array(X)
y = np.array(y)
```

Рисунок 15 — Преобразование данных

Tkinter является стандартной библиотеки для языка Python. Данный инструмент был создан для формирования графических интерфейсов пользователя. В данной дипломной работе Tkinek был применён для разработки пользовательского интерфейса для раздела загрузки файлов, отображения прогресса и вывода полученных результатов.[12].

Стоит подметить, что данная библоитека была разработана в 1988 году профессором математических наук, работающем при университете Беркли, Джоном Остерхаутом. Учёные создал данный инструмент для своего собственного языка программирования Tcl. Тем не мнее, позже библиотека была адаптирована для многих других языков программирования, например, Ruby или Perl. Важно упомянуть, что 1994 данный пакет является частью базового дистрибутива языка программирования Python.

В большинстве пользовательских приложений, которыми мы пользуемся на мобильных и десктопных устройствах, используется графический интерфейс пользователя. Это система интерактивных элементов, обеспечивающих взаимодействие человека с программой. Чтобы программа выполнила нужное вам действие, не нужно вводить соответствующую команду в строку — достаточно кликнуть или коснуться пальцем виртуальной кнопки, поставить флажок или галочку, перетащить ползунок и т.д. Этим программы с графическим интерфейсом отличаются от консольных приложений, в которых любую команду нужно вбивать текстом в командную строку. Пример использования в данной работе: создание интерфейса приложения, представленное на рисунке ниже.

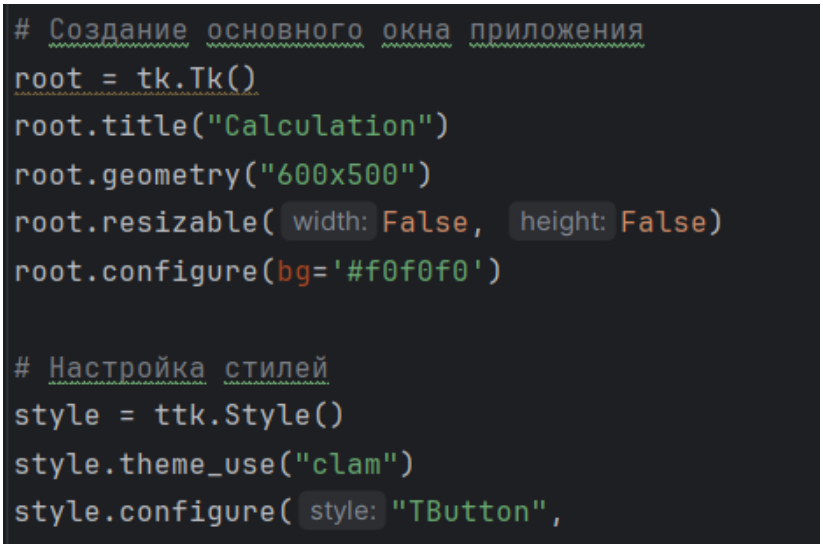

Рисунок 16 – Создание графического интерфейса

Matplotlib.

Matplotlib — это библиотека для создания статических, анимационных и интерактивных визуализаций в языке программирования Python. Она широко используется в научных и аналитических исследованиях благодаря своей гибкости и широкому спектру возможностей для построения графиков.

Основные возможности Matplotlib - создание различных типов графиков: линейные графики для отображения изменений данных во времени

или отслеживания зависимости между переменными, гистограммы для визуализации распределения данных, диаграммы рассеяния для изучения взаимосвязи между двумя переменными, столбчатые диаграммы позволяют сравнивать величины между категориями и другие.

Для настройки графиков можно создавать подписи осей и заголовки, легенды, задавать настройки цветов, стилей линий и маркеров, настройку сеток и шкал, аннотации. Так же библиотека поддерживает анимации и интерактивности.

В работе библиотека Matplotlib использована для задачи визуализации данных с целью сравнения входящих в нейросеть данных с полученными данными для проверки эффективности модели. Пример использования в данной работе: создание графиков, представленное на рисунке ниже.

```
# Построение графиков
plt.figure(figsize=(18, 6))

# Первый график
plt.subplot( *args: 1, 3, 1)
plt.scatter(x_values_1, y_values_1, marker='o', color='b')
plt.xlabel('X координаты')
plt.ylabel('Y координаты')
plt.title('График по начальным точкам')
plt.grid(True)

# Второй график
plt.subplot( *args: 1, 3, 2)
plt.scatter(x_values_2, y_values_2, marker='o', color='r')
plt.xlabel('X координаты')
plt.ylabel('Y координаты')
plt.title('График по точкам, соответствующим прогнозу пористостей')
plt.grid(True)
```

Рисунок 17 – Построение графиков

4 Результаты разработки

4.1 Нейросеть

4.1.1 Архитектура

Для дипломной работы была выбрана архитектура многослойного перцептрона (MLP) с пятью полносвязными слоями и использованием Dropout. Эта архитектура обладает несколькими ключевыми преимуществами, которые делают её оптимальной для данной задачи. Многослойный перцептрон способен эффективно моделировать сложные нелинейные зависимости, что обеспечивает высокую точность предсказаний.

Гибкость и адаптивность: многослойный перцептрон (MLP) хорошо справляется с задачами регрессии и классификации, что делает его подходящим для предсказания пористости материалов. Использование нескольких скрытых слоёв позволяет модели извлекать иерархические признаки из данных, что особенно важно для сложных зависимостей в физико-математических задачах.

Обработка многомерных данных: входные данные состоят из 100 строк признаков для каждого из 11 признаков. Полносвязные слои модели могут эффективно обрабатывать такие высокоразмерные данные, трансформируя их в более компактные и информативные представления.

Предотвращение переобучения: использование слоёв Dropout с вероятностью 0.5 после каждого скрытого слоя помогает бороться с переобучением. Dropout улучшает обобщающую способность модели, заставляя её учиться более устойчивым признакам.

Постепенное уменьшение размерности: архитектура модели с последовательным уменьшением размерности скрытых слоёв (256, 128, 64, 32)

позволяет модели постепенно абстрагировать и обобщать информацию. Это также помогает избежать избыточной сложности модели, что могло бы привести к переобучению и ухудшению обобщающей способности.

Архитектура MLP сравнительно проста в реализации и требует меньше вычислительных ресурсов по сравнению с более сложными архитектурами, такими как сверточные или рекуррентные нейронные сети.

Сверточные нейронные сети (CNN) хорошо работают с данными, имеющими пространственные зависимости (изображения или временные ряды). Предоставленные данные являются фиксированным набор признаков, пространственная структура не так важна. Поэтому использование MLP более оправдано.

Рекуррентные нейронные сети (RNN) подходят для последовательных данных (текст, временные ряды), где важен порядок элементов. Исследуемые данные имеют фиксированное количество признаков, и порядок не является критическим фактором, поэтому MLP подходит больше.

Это делает её идеальной для использования в учебных проектах и научных исследованиях, где ресурсы могут быть ограничены. Выбранная архитектура многослойного перцептрона обеспечивает баланс между точностью, стабильностью и вычислительной эффективностью.

```python
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.fc1 = nn.Linear(100 * 11, 512)
        self.bn1 = nn.BatchNorm1d(512)
        self.fc2 = nn.Linear(512, 256)
        self.bn2 = nn.BatchNorm1d(256)
        self.fc3 = nn.Linear(256, 128)
        self.bn3 = nn.BatchNorm1d(128)
        self.fc4 = nn.Linear(128, 64)
        self.bn4 = nn.BatchNorm1d(64)
        self.fc5 = nn.Linear(64, 2)

        self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.dropout(torch.relu(self.bn1(self.fc1(x))))
        x = self.dropout(torch.relu(self.bn2(self.fc2(x))))
        x = self.dropout(torch.relu(self.bn3(self.fc3(x))))
        x = torch.relu(self.bn4(self.fc4(x)))
        x = self.fc5(x)
        return x
```

Рисунок 18 – Архитектура нейронной сети

Модель, представленная в коде, представляет собой многослойную нейронную сеть с использованием полносвязных слоёв, нормализации пакетов (Batch Normalization) и регуляризации с помощью Dropout. В модели пять полносвязных слоёв, вход в первый слой размером 1100 (по количеству признаков), далее 512, 256, 128, 64 и 2 (так как прогнозируемых параметра два). Более широкая архитектура в начале, то есть увеличенное количество нейронов в первых слоях, оказалась выгодной (по сравнению, например, с последовательностью количества нейронов 1100 – 256 – 128 – 64 – 32 – 2), так как исходные данные имеют сложные и многочисленные взаимосвязи. В конструкторе класса определяются слои нейронной сети, количество нейронов в них Batch Normalization и Dropout. В методе forward определяется порядок выполнения слоёв и операции в модели.

Batch Normalization улучшает стабильность и скорость обучения за счёт нормализации активаций и снижает зависимость от начальных значений параметров и помогает предотвратить переобучение.

Dropout предотвращает переобучение путём случайного отключения нейронов во время обучения, это улучшает обобщающую способность модели, делая её более устойчивой к шуму в данных. Такие слои вставлены после каждого полносвязного слоя, кроме последнего, чтобы улучшить обобщающую способность модели.

ReLU (Rectified Linear Unit) — популярная функция активации, которая помогает избежать проблемы затухающих градиентов и ускоряет обучение, так как она проста и вычислительно эффективна. ReLU отключает нейрон, если входное значение меньше или равно нулю и пропускает положительные значения без изменений.

4.1.2 Обучение

Для обучения нейронной сети необходимо подготовить данные. Данные загружаются из файла DispMatr.txt, обрабатываются и форматируются для использования в модели. Были созданы пустые списки X и y, в которые записывались признаки и целевые значения соответственно. Данные обрабатывались блоками по 101 строке. Каждый блок состоял из 100 строк признаков и одной строки целевых значений. Код для загрузки и подготовки данных представлен ниже:

```python
# Загрузка данных
with open("DispMatr.txt", "r") as f:
    lines = f.readlines()[3:]

# Обработка данных
X = []
y = []
for i in range(0, len(lines), 101):
    lines_t = lines[i: i + 101]

    X_block = []
    try:
        por1, por2 = (float(val) for val in lines_t[0].split())
        for j in range(1, 101):
            points_t = [float(val) for val in lines_t[j].split()]
            n_y = points_t.pop(1)
            points_t += [0 for _ in range(10 - int(n_y))]
            X_block.append(points_t)

        y_block = [por1, por2]
    except ValueError as e:
        print(e)
        continue
    if len(X_block) != 100:
        print(f"Unexpected number of rows in block starting at line {i + 4}: {len(X_block)}")
        continue
    X.append(np.array(X_block).flatten())
    y.append(y_block)

X = np.array(X)
y = np.array(y)
```

Рисунок 19 – Обработка датасета

После подготовки данных, они стандартизируются и разделяются на обучающую и тестовую выборки. Определяется функция потерь и оптимизатора, используется MSE в качестве функции потерь и Adam в качестве оптимизатора. Для изменения скорости обучения используется CyclicLR. Adam — это алгоритм оптимизации, который используется для обновления весов нейронной сети во время обучения. CyclicLR — это стратегия изменения скорости обучения, при которой значение скорости обучения циклически изменяется между минимальным и максимальным значениями в течение тренировки. Далее на протяжении 1000 эпох модель обучается на тренировочных данных. Для каждой эпохи модель переводится в режим обучения, для каждого батча данных вычисляется ошибка, обновляются веса модели. После обучения модель переводится в режим оценки, и вычисляется ошибка на валидационных данных. Если текущая ошибка на валидационных данных меньше минимальной ранее

зафиксированной, сохраняются параметры модели. Модель сохраняется с параметрами, которые соответствуют минимальной достигнутой ошибке.

4.2 Использование модели и графический интерфейс

Для выполнения задач по определению пористости материалов в рамках моей дипломной работы используется модель нейронной сети, реализованная с помощью библиотеки PyTorch. Дополнительно разработан графический интерфейс на языке Python с использованием библиотеки Tkinter, обеспечивающий удобное взаимодействие пользователя с программой и проводится графическое сравнение результатов в виде графиков, созданных с помощью библиотеки Matplotlib.

Для использования обученной модели нейронной сети необходимо загрузить модель и её параметры. Далее нужно обработать входной файл и загрузить его. Построение модели и обработка входящих данных должна проводиться так же, как проводилась и при обучении модели.

Интерфейс программы включает в себя кнопки для загрузки файла и начала обработки, а также индикатор прогресса и текстовое поле для вывода результатов. Результат запуска программы представлен на картинках ниже.

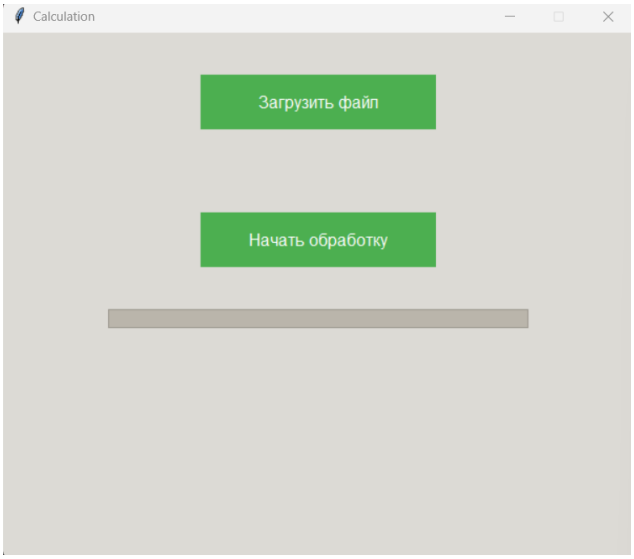

Рисунок 20 – Начальное окно графического интерфейса

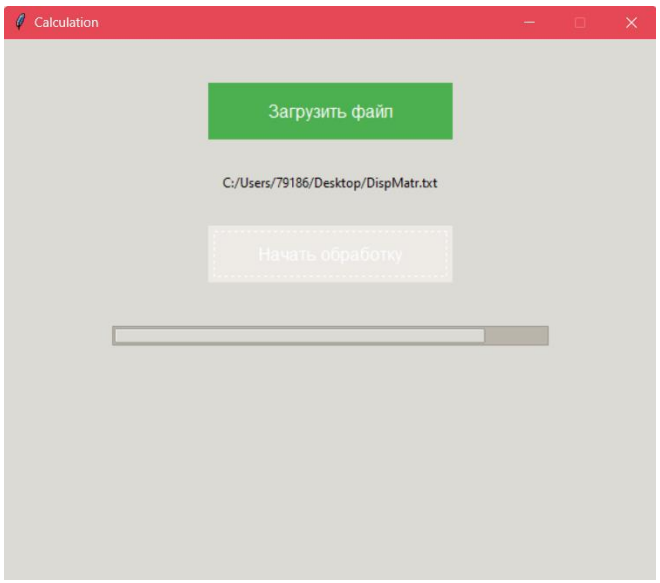

Рисунок 21 – Процесс загрузки файла

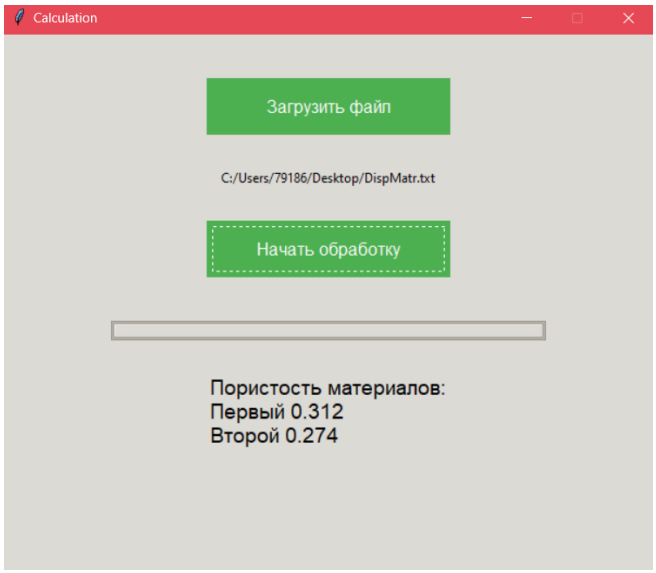

Рисунок 22 – Результат, отображающийся в интерфейсе

Для наглядности работы программы можно сравнивать графики зависимости, которые являются данными для обучения. Для этого мы после получения прогноза нейросетью сохраняем эти два параметра пористости и создаем запрос программе на Fortran, отправляя ей эти показатели.

Файл Изменить Просмотр

DispMatr.txt

```
1  0.312 0.312  !n_eps1,eps1_1,eps1_2;  eps1_2 <= 0.36; eps1_1>0
1  0.274 0.274  !n_eps2,eps2_1,eps2_2;  eps2_2 <= 0.36; eps2_1>0
100 1.00 2e3   !n_freq,freq1,freq2
```

Рисунок 23 – Загрузка данных в программу

После запуска мы получаем файл с набором точек и теперь можем сравнить его с тем, что мы загружали в нейросеть. После построения двух графиков мы можем их сравнить. Если графики будут не совпадать, это будет означать, что прогноз неверный.

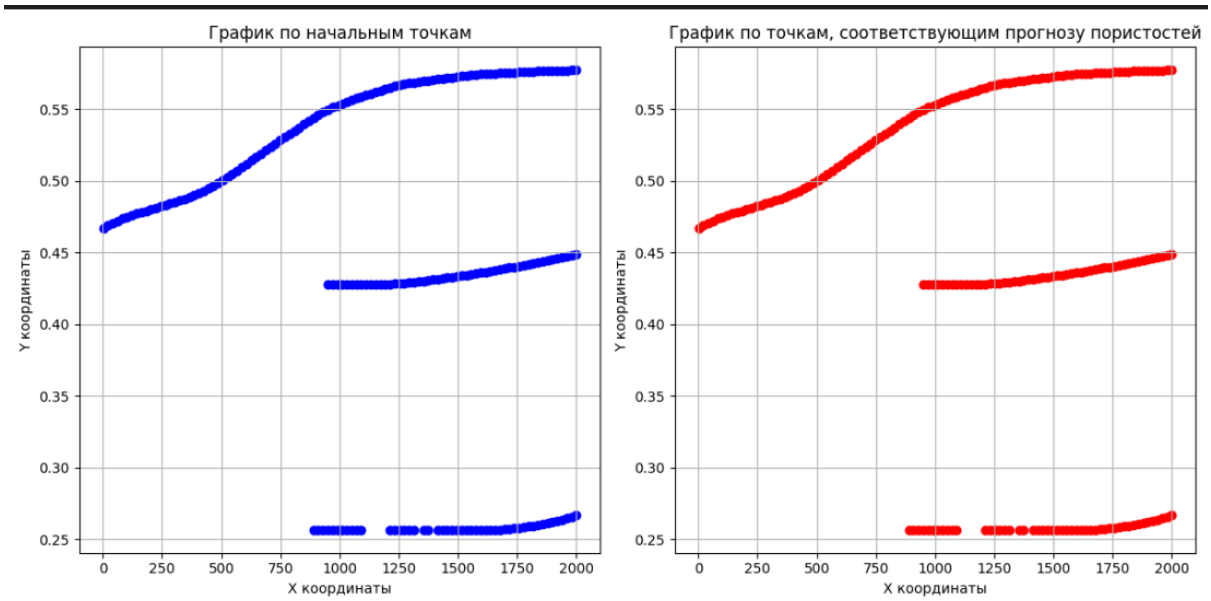

Рисунок 24 – Сравнение результатов в первом примере

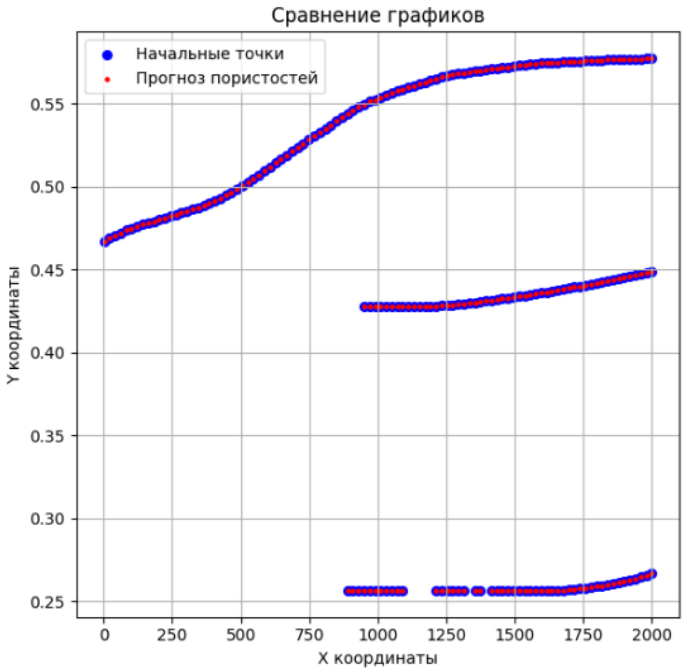

Рисунок 25 – Наложение результатов первого примера

В данном эксперименте графики совпали, что видно при наложении графиков друг на друга, данные были спрогнозированы точно, но так происходит не всегда. Протестировав систему множество раз, я выделила ещё два интересных случая, которые привожу ниже. На примере ниже видно, что данные будто точно совпадают.

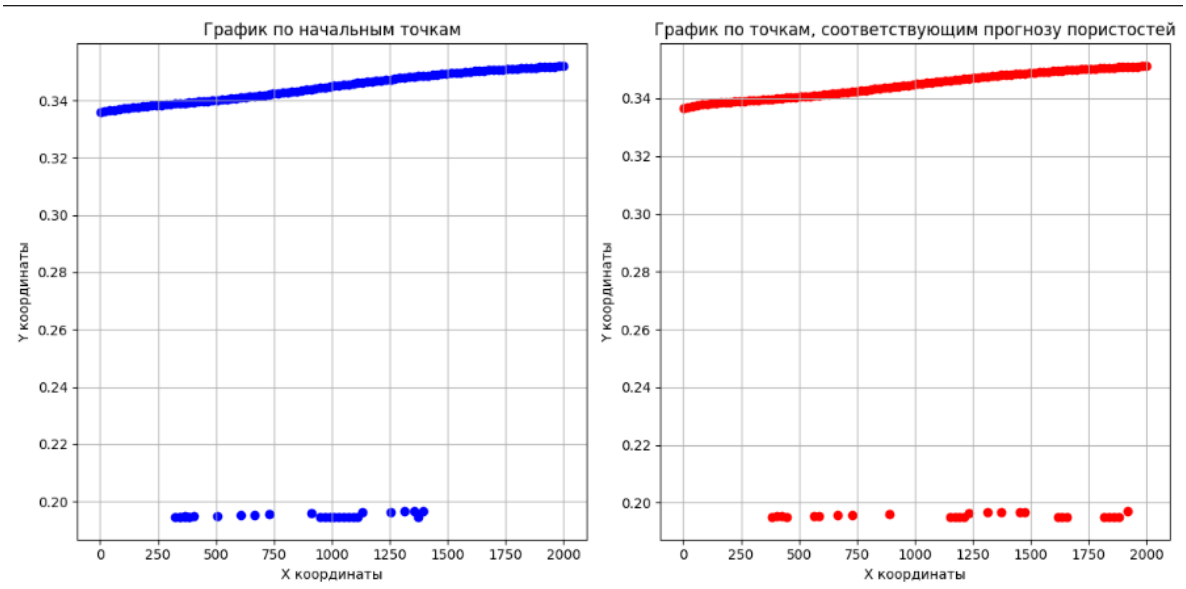

Рисунок 26 – Сравнение результатов во втором примере

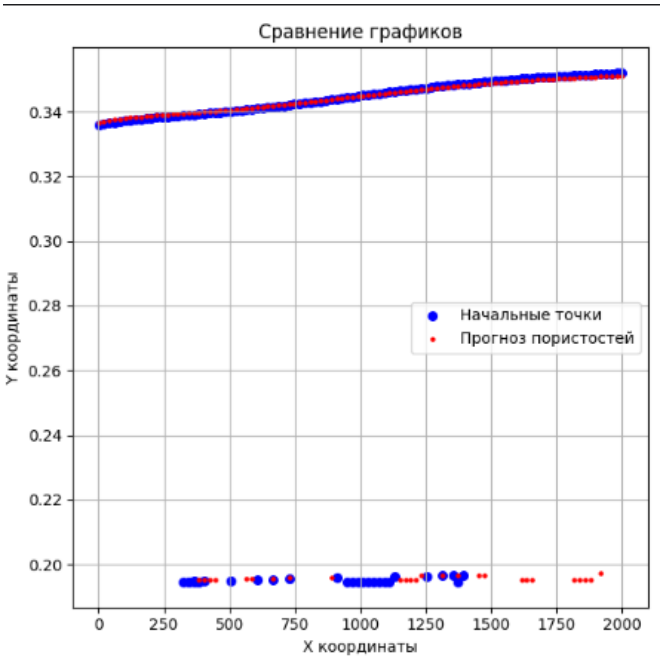

Рисунок 27 – Наложение результатов второго примера

Но при наложении графиков видно, что несовпадения есть, хоть и достаточно незначительные. На следующем примере без наложения графиков друг на друга создается впечатление, что графики не совпадают совсем.

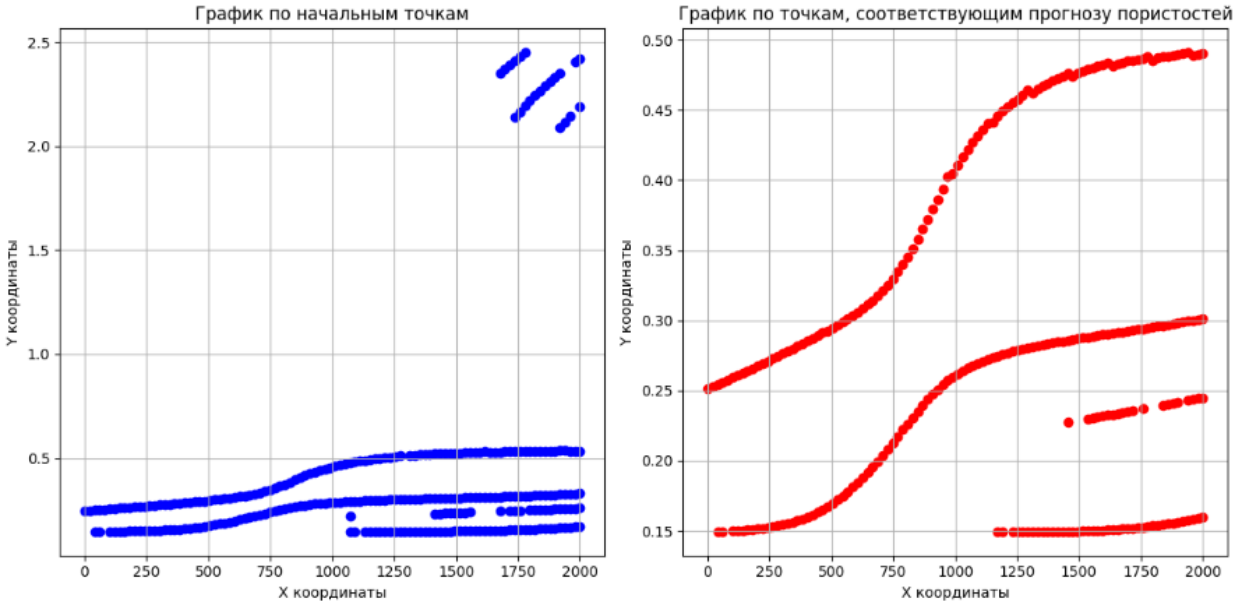

Рисунок 28 – Сравнение результатов в третьем примере

Однако, при наложении, хоть мы и видим существенные расхождения, всё же понятно, что величины спрогнозированы верно, хоть и с ошибкой.

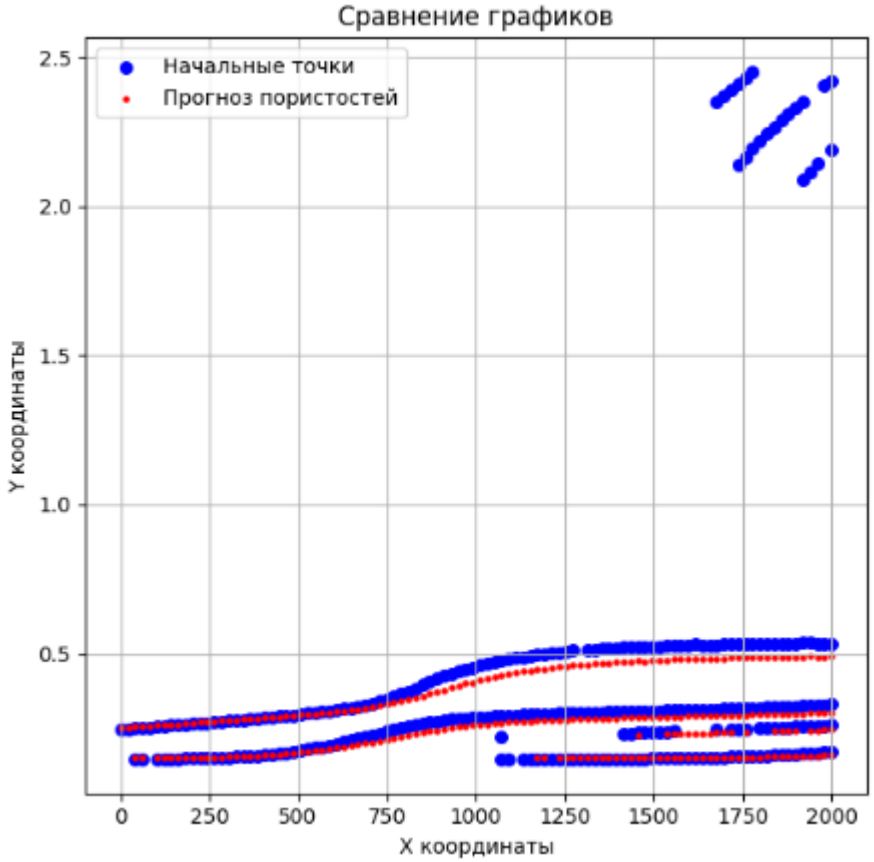

Рисунок 29 – Наложение результатов третьего примера

4.3 Анализ ошибок модели

Mean Squared Error (MSE) (среднеквадратическая ошибка) — это метрика, которая используется для оценки качества предсказательной модели, особенно в задачах регрессии. Она вычисляет среднее значение квадратов отклонений предсказанных значений от фактических. Чем меньше значение MSE, тем точнее модель.

В работе MSE использовалась для оценки точности модели, которая предсказывает пористость двух материалов на основе данных о прохождении волн через материалы. После обучения отслеживались значения MSE на

каждой эпохе. Минимальная ошибка MSE равна 0.0022. Это хорошее значение, так как оно близко к нулю, что указывает на высокую точность модели. С учетом того, что предсказываемые значения находятся в диапазоне от 0.01 до 0.35, значение MSE нужно интерпретировать с учетом этого диапазона.

Если рассматривать среднеквадратическое отклонение (RMSE), которое является корнем из MSE, то отклонения будет равняться 0.0469 – корень из 0.0022. Это означает, что в среднем предсказания модели отклоняются от фактических значений на 0.0469, что примерно составляет 13.4% от среднего значения диапазона.

Для диапазона 0.01-0.35 значение RMSE около 0.0469 можно считать хорошим результатом, так как ошибка составляет около 13.4% от среднего значения диапазона. Это указывает на то, что модель делает достаточно точные предсказания.

Рассмотрим изменение ошибки на протяжении обучения модели. На первых этапах обучения модели значение ошибки было довольно высоким. Например, на первой эпохе значение составило 0.0311, а на второй эпохе оно резко увеличилось до 0.5673. Это вполне ожидаемо для начала обучения, так как модель только начинает подстраиваться под данные и искать оптимальные параметры.

По мере обучения, в процессе которого модель прошла через 1000 эпох, значение ошибки постепенно уменьшалось. Мы видим, что уже на восьмой эпохе ошибка снизилась до 0.0216, что свидетельствует о том, что модель начинает находить правильные зависимости в данных. К пятнадцатой эпохе ошибка продолжала снижаться и составила 0.0185.

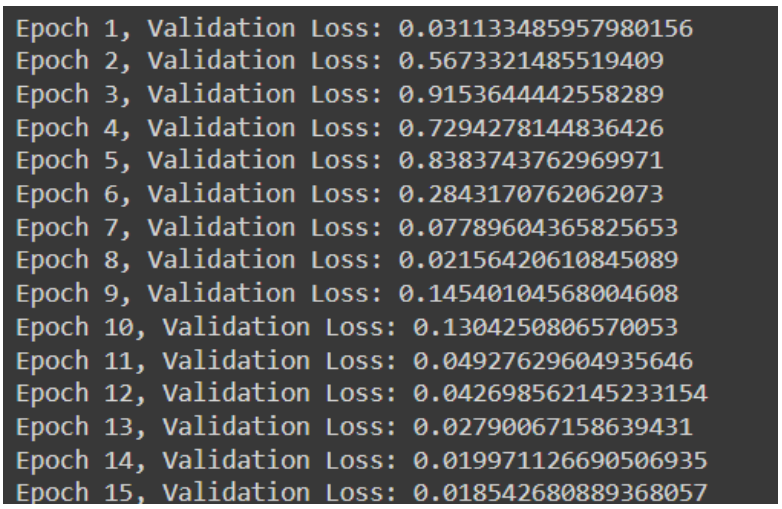

Рисунок 30 – Ошибка в начале обучения

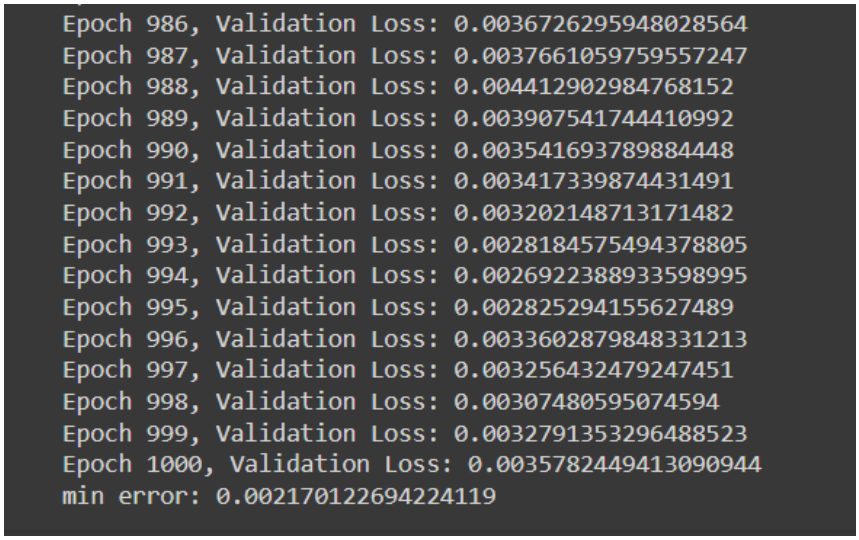

Рисунок 31 – Ошибка в конце обучения и минимальная ошибка

Однако наиболее заметное улучшение происходило в течение последних 15 эпох. Например, на 986-й эпохе ошибка составляла 0.0037, а на 994-й эпохе она еще уменьшилась до 0.0027. Минимальная ошибка, достигнутая за все время обучения, составила 0.0022. Модель с такими параметрами была сохранена, чтобы можно было использовать её для дальнейших предсказаний.

ЗАКЛЮЧЕНИЕ

В рамках данной дипломной работы был разработан программный комплекс с графическим интерфейсом для решения обратных задач коэффициентов теории пороупругости с использованием методов нейронных сетей. Созданный программный продукт позволяет определять пористость материалов на основе данных о прохождении волн через два пористых материала, расположенных друг над другом.

В ходе работы были рассмотрены и применены основы теории пороупругости и нейронных сетей. Был проведен анализ существующих методов решения подобных задач, определены их преимущества и недостатки, что позволило выбрать наиболее подходящий подход для реализации поставленной задачи.

Разработанный программный комплекс является инструментом, который может значительно облегчить процесс определения пористости материалов. Его использование может быть полезным в различных научных и промышленных приложениях, где требуется точное определение характеристик материалов.

Таким образом, данная работа представляет собой шаг вперед в области исследований пороупругости и имеет потенциал для широкого применения. В дальнейшем возможны улучшения и дополнения программы, что позволит еще более эффективно использовать ее возможности в различных сферах.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Глушков, Е. В. Влияние пористоски на Характеристики волн релеевского типа в многослойном полупространстве / Е. В. Глушков, Н. В. Глушкова, С. И. Фоменко – 2011 г. – 12 с.

2. Fomenko, S. I. The influence of water evaporation on surface acoustic waves in soils / S.I. Fomenko, R.B. Jana // 2023 Days on Diffraction (DD). – 2023, P. 1-4, – DOI 10.1109/DD58728.2023.10325755.

3. Vasilyeva, M. Machine learning for accelerating effective property prediction for poroelasticity problem in stochastic media / M. Vasilyeva, A. Tyrylgin. – город:издательство, 2018 г. – 21 с.

4. Bliyeva, D. Computer Simulation of the Seismic Wave Propagation in Poroelastic Medium / D. Bliyeva, D. Baigereyev, K. Imomnazarov. – город 2022 г. – 20 с.

5. Нейронные сети, перцептрон сайт. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Нейронные_сети,_перцептрон (дата обращения: 20.05.2024).

6. Функции активации в искусственных нейронных сетях сайт. – URL: https://aisimple.ru/8-funkcii-aktivacii-v-iskusstvennyh-nejronnyh-setjah.html#:~:text=Пожалуй%2C%20наиболее%20популярная%20функция%20активационная,диапазоне%20от%20-1%20до%201. (дата обращения: 01.06.2024).

7. Работа в Google Colab сайт. – URL: https://gb.ru/blog/rabota-v-google-colab/ (дата обращения 01.05.2024).

8. IDE для Data Science сайт. – URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения 05.05.2024).

9. PyTorch GET STARTED сайт. – URL: https://pytorch.org/ (дата обращения 05.05.2024).

10. scikit-learn Machine Learning in Python сайт. – URL: https://scikit-learn.org/stable/ (дата обращения 10.05.2024).

11. NumPy сайт. – URL: https://numpy.org/ (дата обращения 10.05.2024).

12. Руководство по Tkinter сайт. – URL: https://metanit.com/python/tkinter/ (дата обращения 10.05.2024).

13. Fomenko, S.I. Numerical Modeling of Elastic Wave Propagation in Porous Soils with Vertically Inhomogeneous Fluid Contents Due to Infiltration. Mathematics/ S.I. Fomenko , R.B. Jana, M.V. Golub // Mathematics, MDPI, – 2023. - Vol. 11, №19. – P. 1-17.

14. Саттон, Р. С. Обучение с подкреплением: Введение. 2-е изд./пер. с англ. АА Слинкина //Москва: ДМК Пресс. – 2020. – 310. - ISBN 9789001796808.

15. Абу-Мостафа Я. Малик Магдон-Исмаил Learning From Data /Абу-Мостафа Я. Малик Магдон-Исмаил, Сюань-Тянь Линь – 2012. – 124. – ISBN 9781789534160

16. Попова, Ю. Б. Обучение искусственных нейронных сетей методом обратного распространения ошибки/ Ю. Б. Попова, С. В. Яцынович – 2016. – 237. - ISBN 9785020066977.

17. Короткий, С. Нейронные сети: алгоритм обратного распространения / С. Короткий – 2002. –125. – ISBN 9785457387683

18. Stevens, E. Deep learning with PyTorch/ E. Stevens, L. Antiga, T. Viehmann. – 2020. – 522. – ISBN 9781638354079.

19. Кучеренко, Д. В. Пороупругая модель в задаче деформирования элемента миокарда / Д.В. Кучеренко //Неделя науки СПбПУ: материалы научной конференции с международ-ным участием, 18–23 ноября 2019 г.

20. Гарипов, Т. Т. Моделирование процесса гидроразрыва пласта в пороупругой среде / Т.Т. Гарипов //Математическое моделирование. – 2006. – Т. 18. – №. 6. – С. 53-69.

21. Мельничук, Д. А. Математическое моделирование деформационных изменений в окрестности нефтяной скважины/ Д. А. Мельничук, В. В. Стрельченко //Секция № 5. – 2016. – С. 75.

ПРИЛОЖЕНИЕ А

```
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import torch
from torch.utils.data import Dataset
import torch.nn as nn
from torch.optim.lr_scheduler import CyclicLR


# Загрузка данных
with open("DispMatr.txt", "r") as f:
    lines = f.readlines()[3:]

# Обработка данных
X = []
y = []
for i in range(0, len(lines), 101):
    lines_t = lines[i: i + 101]

    X_block = []
    try:
        por1, por2 = (float(val) for val in lines_t[0].split())
        for j in range(1, 101):
            points_t = [float(val) for val in lines_t[j].split()]
            n_y = points_t.pop(1)
            points_t += [0 for _ in range(10 - int(n_y))]
            X_block.append(points_t)
```

```
            y_block = [por1, por2]
        except ValueError as e:
            print(e)
            continue
        if len(X_block) != 100:
            print(f"Unexpected number of rows in block starting at line {i + 4}: {len(X_block)}")
            continue
        X.append(np.array(X_block).flatten())
        y.append(y_block)

X = np.array(X)
y = np.array(y)

# Убедимся, что данные успешно считаны
print(f"Shape of X: {X.shape}")
print(f"Shape of y: {y.shape}")

# Стандартизация признаков
X = StandardScaler().fit_transform(X)

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)

# Определение набора данных
class CustomDataset(Dataset):
    def __init__(self, X, y):
        self.X = X
        self.y = y
```

```
    def __len__(self):
        return len(self.X)

    def __getitem__(self, idx):
        return torch.tensor(self.X[idx], dtype=torch.float32), torch.tensor(self.y[idx], dtype=torch.float32)

train_dataset = CustomDataset(X_train, y_train)
val_dataset = CustomDataset(X_test, y_test)

# Определение загрузчиков данных
train_dataloader = torch.utils.data.DataLoader(train_dataset, batch_size=32, shuffle=True)
val_dataloader = torch.utils.data.DataLoader(val_dataset, batch_size=128, shuffle=True)

# Устройство для вычислений
device = "cuda" if torch.cuda.is_available() else "cpu"

# Определение модели
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.fc1 = nn.Linear(100 * 11, 512)
        self.bn1 = nn.BatchNorm1d(512)
        self.fc2 = nn.Linear(512, 256)
        self.bn2 = nn.BatchNorm1d(256)
        self.fc3 = nn.Linear(256, 128)
        self.bn3 = nn.BatchNorm1d(128)
```

```
        self.fc4 = nn.Linear(128, 64)
        self.bn4 = nn.BatchNorm1d(64)
        self.fc5 = nn.Linear(64, 2)

        self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.dropout(torch.relu(self.bn1(self.fc1(x))))
        x = self.dropout(torch.relu(self.bn2(self.fc2(x))))
        x = self.dropout(torch.relu(self.bn3(self.fc3(x))))
        x = torch.relu(self.bn4(self.fc4(x)))
        x = self.fc5(x)
        return x

model = Model().to(device)

# Определение функции потерь и оптимизатора
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
scheduler = CyclicLR(optimizer, base_lr=0.001, max_lr=0.01, step_size_up=5,
mode="triangular")

# Количество эпох
EPOCHS = 1000

# Цикл обучения
min_val_loss = float('inf')
for epoch in range(EPOCHS):
    model.train()
    for X_batch, y_batch in train_dataloader:
```

```
            X_batch, y_batch = X_batch.to(device), y_batch.to(device)
            optimizer.zero_grad()
            predictions = model(X_batch)
            loss = criterion(predictions, y_batch)
            loss.backward()
            optimizer.step()
            scheduler.step()

        model.eval()
        val_loss = 0
        with torch.no_grad():
            for X_batch, y_batch in val_dataloader:
                X_batch, y_batch = X_batch.to(device), y_batch.to(device)
                predictions = model(X_batch)
                val_loss += criterion(predictions, y_batch).item()

        print(f"Epoch {epoch + 1}, Validation Loss: {val_loss / len(val_dataloader)}")


        if val_loss < min_val_loss:
            min_val_loss = val_loss


print("min error:", min_val_loss)
torch.save(model.state_dict(), f"models/model_loss_{min_val_loss:.3f}.pt")
```

ПРИЛОЖЕНИЕ Б

```
import numpy as np
import torch
from sklearn.preprocessing import StandardScaler
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
import time


# Определение модели
class Model(torch.nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.fc1 = torch.nn.Linear(100 * 11, 512)
        self.bn1 = torch.nn.BatchNorm1d(512)
        self.fc2 = torch.nn.Linear(512, 256)
        self.bn2 = torch.nn.BatchNorm1d(256)
        self.fc3 = torch.nn.Linear(256, 128)
        self.bn3 = torch.nn.BatchNorm1d(128)
        self.fc4 = torch.nn.Linear(128, 64)
        self.bn4 = torch.nn.BatchNorm1d(64)
        self.fc5 = torch.nn.Linear(64, 2)
```

```
        self.dropout = torch.nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.dropout(torch.relu(self.bn1(self.fc1(x))))
        x = self.dropout(torch.relu(self.bn2(self.fc2(x))))
        x = self.dropout(torch.relu(self.bn3(self.fc3(x))))
        x = torch.relu(self.bn4(self.fc4(x)))
        x = self.fc5(x)
        return x

# Устройство для вычислений
device = "cuda" if torch.cuda.is_available() else "cpu"

# Создание модели и загрузка сохранённых весов
model = Model().to(device)
model.load_state_dict(torch.load("model_loss_0.002.pt"))

# Установка модели в режим предсказания
model.eval()

# Функция для загрузки и обработки данных из выбранного файла
```

```
def load_and_process_file(file_path):
    with open(file_path, "r") as f:
        lines = f.readlines()

    X_block = []
    for j in range(100):
        points_t = [float(val) for val in lines[j].split()]
        if len(points_t) < 2:
            raise ValueError(f"Недостаточно данных в строке {j + 1}: {lines[j]}")
        n_y = points_t.pop(1)
        points_t += [0 for _ in range(10 - int(n_y))]
        X_block.append(points_t)

    example_data = np.array(X_block).flatten().reshape(1, -1)

    scaler = StandardScaler()
    example_data = scaler.fit_transform(example_data)

    example_data_tensor = torch.tensor(example_data,
dtype=torch.float32).to(device)

    with torch.no_grad():
        prediction = model(example_data_tensor)
```

```
    prediction = prediction.cpu().numpy().flatten()
    return prediction

# Создание основного окна приложения
root = tk.Tk()
root.title("Calculation")
root.geometry("600x500")
root.resizable(False, False)
root.configure(bg='#f0f0f0')

# Центрирование окна на экране
window_width = 600
window_height = 500
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
position_top = int(screen_height / 2 - window_height / 2)
position_right = int(screen_width / 2 - window_width / 2)
root.geometry(f'{window_width}x{window_height}+{position_right}+{position_top}')

# Настройка стилей
style = ttk.Style()
```

```
style.theme_use("clam")
style.configure("TButton",
                font=('Helvetica', 12),
                background='#4CAF50',
                foreground='#FFFFFF',
                relief="flat",
                width=20,
                borderwidth=5,
                focuscolor='#FFFFFF',
                focusthickness=2)
style.configure("TScale", troughcolor='#f0f0f0', sliderlength=20)

# Функция для выбора файла
def select_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        file_label.config(text=file_path)

# Функция для выполнения предсказания
def make_prediction():
    file_path = file_label.cget("text")
    if not file_path:
```

```
        result_label.config(text="Please select a file first.")
        return

    # Progress bar
    progress['value'] = 0
    root.update_idletasks()
    for i in range(100):
        time.sleep(0.01)
        progress['value'] += 1
        root.update_idletasks()

    try:
        prediction = load_and_process_file(file_path)
        result_text = f"Пористость материалов:\nПервый
{prediction[0]:.3f}\nВторой {prediction[1]:.3f}"
        result_label.config(text=result_text)
    except Exception as e:
        result_label.config(text=f"Error: {str(e)}")

# Создание фрейма для содержимого
content_frame = ttk.Frame(root, padding="20")
content_frame.pack(expand=True, fill=tk.BOTH)
```

```
# Кнопка для выбора файла
select_button = ttk.Button(content_frame, text="Загрузить файл",
command=select_file, style="TButton")
select_button.pack(pady=20, ipadx=10, ipady=5)

# Метка для отображения пути к файлу
file_label = ttk.Label(content_frame, text="", wraplength=500)
file_label.pack(pady=10)

# Кнопка для выполнения предсказания
predict_button = ttk.Button(content_frame, text="Начать обработку",
command=make_prediction, style="TButton")
predict_button.pack(pady=20, ipadx=10, ipady=5)

# Шкала загрузки
progress = ttk.Progressbar(content_frame, orient=tk.HORIZONTAL, length=400,
mode='determinate')
progress.pack(pady=20)

# Метка для отображения результата
result_label = ttk.Label(content_frame, text="", wraplength=500, font=('Helvetica',
14))
result_label.pack(pady=10)
```

```
# Запуск основного цикла приложения
root.mainloop()
```

ПРИЛОЖЕНИЕ В

```
import matplotlib.pyplot as plt

# Задаем путь к файлам
file_path_1 = 'DispMatr_model_1.txt'
file_path_2 = 'DispMatr_model_2.txt'

# Инициализация списков для хранения значений x и y
x_values_1, y_values_1 = [], []
x_values_2, y_values_2 = [], []

# Чтение файла и обработка строк
def read_file(file_path, x_values, y_values):
    with open(file_path, 'r') as file:
        for line in file:
            parts = line.strip().split()
            x = float(parts[0])
            y_count = int(parts[1])
            for i in range(y_count):
                y = float(parts[2 + i])
                x_values.append(x)
                y_values.append(y)

# Чтение данных для первого графика
read_file(file_path_1, x_values_1, y_values_1)

# Чтение данных для второго графика
read_file(file_path_2, x_values_2, y_values_2)
```

```
# Построение графиков
plt.figure(figsize=(18, 6))

# Первый график
plt.subplot(1, 3, 1)
plt.scatter(x_values_1, y_values_1, marker='o', color='b')
plt.xlabel('X координаты')
plt.ylabel('Y координаты')
plt.title('График по начальным точкам')
plt.grid(True)

# Второй график
plt.subplot(1, 3, 2)
plt.scatter(x_values_2, y_values_2, marker='o', color='r')
plt.xlabel('X координаты')
plt.ylabel('Y координаты')
plt.title('График по точкам, соответствующим прогнозу пористостей')
plt.grid(True)

# Третий график с наложением
plt.subplot(1, 3, 3)
plt.scatter(x_values_1, y_values_1, marker='o', color='b', label='Начальные точки',
s=30)
plt.scatter(x_values_2, y_values_2, marker='o', color='r', label='Прогноз
пористостей', s=5)
plt.xlabel('X координаты')
plt.ylabel('Y координаты')
plt.title('Сравнение графиков')
plt.grid(True)
```

```
plt.legend()

plt.tight_layout()
plt.show()
```