Федеральное государственное бюджетное учреждение науки
«Санкт-Петербургский Федеральный исследовательский центр
Российской академии наук»

На правах рукописи

Соболевский Владислав Алексеевич

КОМПЛЕКСНАЯ АВТОМАТИЗАЦИЯ СИНТЕЗА ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ПРЯМОГО РАСПРОСТРАНЕНИЯ

Специальность 2.3.5 – Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
заслуженный деятель науки РФ

Соколов Борис Владимирович

Санкт-Петербург – 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы диссертации. В современном мире всё активнее начинают применяться технологии искусственного интеллекта (ИИ) в совершенно различных областях человеческой деятельности – от анализа больших данных [6, 7, 22] до систем распознавания объектов на фото- и видеоизображениях [122, 129] и автоматических систем контроля и управления, базирующихся на использовании технологий киберфизических систем, промышленного интернета вещей (ИВ), нейросетевых систем [77]. Обращают пристальное внимание на эту область и правительства различных государств [24].

Во многом развитие области ИИ стало возможно благодаря классу таких моделей как искусственные нейронные сети (ИНС). Сам класс алгоритмов зародился более 60 лет назад [115]. Однако, лишь в последние два десятилетия его применение стало возможно благодаря активному развитию компьютерных технологий. Следует отметить, что на данный момент, именно ИНС являются самым активно внедряемым классом алгоритмов ИИ [80].

С каждым годом появляется всё больше алгоритмов ИНС, которые позволяют решать совершенно разные задачи в различных областях человеческой деятельности. Примерами самых бурно развивающихся направлений могут служить ИНС для анализа больших данных, применяющиеся в системах Data Mining [52], а также свёрточные нейронные сети (СНС) [85], использующиеся для анализа фото- и видеоизображений.

Для анализа больших данных применяются совершенно разные архитектуры ИНС [75, 79, 138]. При этом с каждым годом становится не только больше видов таких архитектур, но и совершенствуются и усложняются уже существующие [48, 56, 71]. СНС также активно развиваются. С каждым годом растёт как точность результатов, получаемых с помощью СНС, так и сложность данных систем. Одни из самых успешных и распространённых на данный момент архитектур СНС имеют множество разнородных слоёв [82, 123, 142].

Отдельно стоит отметить следующую особенность ИНС – в случае использования данного класса алгоритмов разделяют термины «архитектура» и «модель». Под моделью ИНС понимают уже обученную сеть, предназначенную для решения конкретной прикладной задачи. А под архитектурой ИНС понимают правила построения модели в процессе обучения и её последующего функционирования при эксплуатации. Иначе говоря, архитектурой называют типовой «чертёж», по которому создаются конкретные модели ИНС. Соответственно, термин «архитектура ИНС» объединяет под собой все модели ИНС, построенные и функционирующие по одинаковым принципам. Данная терминология использовалась в работе Дэвида Румельхарта и Джеймса МакКлелланда ещё в 1987 году [55]. Впоследствии терминология устоялась и на сегодняшний день используется при описании ИНС.

Развитие и усложнение ИНС приводит не только к повышению точности их работы, но и к усложнению процесса создания и обучения таких моделей. С другой стороны, растёт и количество задач, которые возможно решить с их помощью. Такие задачи не всегда требуют применения передовых и самых сложных архитектур ИНС. Но, тем не менее, являются слишком сложными, что бы их можно было решить с помощью простых архитектур ИНС без надлежащей дополнительной настройки. Простые пользователи без знаний методов глубокого обучения и навыков для их применения не смогут корректно создать и обучить ИНС, которые были бы способны решать такие задачи с высокой точностью. Можно сказать, что количество задач растёт быстрее количества профессионалов, способных эти задачи решать.

Всё это приводит к тому, что **становится весьма актуальной задача создания систем автоматизации процесса генерации и обучения ИНС для тех или иных областей** [41, 64, 134]. При этом, **всё острее проявляется потребность в системе, подходящей для решения типовых задач из разных областей.** Существует множество задач одного класса, которые требуется решать в совершенно разных предметных областях. К примеру, распознавание на космических снимках конкретных видов деревьев, особенностей рельефа,

конкретных природных объектов и т.п. Принцип решения подобных задач либо уже известен и они исследуются уже известными линейными и нелинейными моделями [90, 121, 145], либо они решаются на основе создания специализированных ИНС [30, 33, 84], либо они не исследуются вовсе, в связи с отсутствием специалистов, способных создать более точные модели.

При этом, многие ИНС создаются в виде программных прототипов, к примеру с помощью пакета программ MatLab. Такие прототипы требуют доработки, для внедрения в существующее программное обеспечение (ПО), и спроектированы с использованием конкретных прикладных программных языков (C++, Java, Python и т.д.). Это усложняет дальнейшее развитие и последующее внедрение прототипов, поскольку требует переписывания на новый язык и оптимизации уже созданной программы.

Подводя итог, можно сказать, что процесс развития и внедрения ИНС в промышленное ПО и повседневную жизнь человека ограничивают как минимум три серьёзных фактора.

Во-первых, **существует огромное разнообразие архитектур ИНС, которые, на сегодняшний день, могут применяться для совершенно разных задач**, но набор которых при этом слабо систематизированы. Также только начинают появляться источники информации, создатели которых бы на постоянной основе следили за развитием всей области ИНС и актуализировали информацию по существующим решениям [111, 126]. Обычный пользователь или начинающий разработчик, в случае, когда ему потребуется создать модель на базе ИНС, может просто не узнать, что существуют ИНС, способные решить поставленные перед ним задачи. При этом даже узнав об их существовании, он зачастую не сможет с ходу выбрать конкретную ИНС, т.к. не существует комплексного сравнения различных архитектур ИНС, способных решать схожие задачи. В каждом отдельном случае придётся выбирать конкретную архитектуру методом проб и ошибок. Данная проблема также усугубляется очень быстрым развитием теоретической базы и активной разработкой новых архитектур в рассматриваемой научной отрасли знаний.

Во-вторых, следствием первой проблемы является и то, что **не существует универсального подхода к генерации и обучению данных архитектур**. В дальнейшем для обобщённого описания процессов генерации и обучения моделей ИНС будет применяться термин «синтез» моделей ИНС. Разные классы ИНС генерируются и обучаются совершенно разными алгоритмами. Каждый такой алгоритм необходимо дорабатывать под каждую конкретную решаемую задачу. Это всё требует обширных узкоспециализированных знаний. Поэтому модели ИНС имеют высокий порог вхождения и не могут применяться простыми пользователями.

В-третьих, даже **уже имеющиеся решения часто реализованы в виде прототипов**. Внедрение подобных прототипов будет требовать их глубокой модификации под конкретную задачу, а также написания программных интерфейсов. Часто, при модификации подобных прототипов, требуется хорошо разбираться в архитектуре ИНС, которая реализована в данном прототипе, что возвращает ко второй описанной проблеме.

В дополнение к перечисленным выше проблемам, стоит отметить и основное направление развития отрасли создания программного обеспечения. Всё активнее во многих областях начинает осуществляться переход к парадигме Software 2.0 [127]. Данная парадигма нацелена на ускорение разработки программных продуктов за счёт повышения уровня автоматизации их создания. Повышение предлагается осуществлять за счёт перехода от платформ разработки Low-Code [45], которые предоставляют разработчикам инструментарий, автоматизирующий написание программного кода, к платформам разработки No-Code [98], которые полностью автоматизируют работу с программным кодом, благодаря чему пользователь может создавать законченные программные продукты без необходимости писать код.

Указанные проблемы, а также переход к парадигме Software 2.0, обуславливают **актуальность и новизну** избранной темы диссертации, которая нацелена на автоматизацию процессов генерации, обучения и использования моделей ИНС прямого распространения (ПР) различных архитектур, за счёт

создания унифицированного алгоритма синтеза моделей и реализации концепции No-Code разработки. Вторым важным направлением диссертационного исследования является упрощение интеграции моделей ИНС ПР в стороннее ПО, с помощью использования сервис-ориентированной архитектуры (СОА) в процессе генерации программных оболочек для созданных и обученных моделей.

Степень разработанности темы. Подход к автоматизации процессов генерации, обучения и использования моделей ИНС ПР не является новым и уже есть ряд работ на данную тему [43, 49, 57, 135]. Все они сводятся к тому, что автоматизация синтеза моделей машинного обучения (МО) позволит ускорить процесс разработки программных продуктов для решения множества прикладных задач.

Также, на сегодняшний день активно разрабатываются программные комплексы и библиотеки автоматизации обучения ИНС, в том числе и в России [21]. Однако, существующие аналоги зачастую нацелены на решение каких-то узкоспециализированных задач и не рассматривают единообразный и универсальный подход к автоматизации обучения различных архитектур.

В данной диссертационной работе ее автором описано разработанное им программное обеспечение, на базе алгоритма унифицированного подбора гиперпараметров (структурных параметров) модели, которое конструктивно развивает идею автоматизации построения и использования различных классов ИНС прямого распространения и имеет модульную расширяемую структуру [110]. Данная структура позволяет добавлять и комбинировать обучаемые архитектуры, алгоритмы обучения, нормализации данных, валидации и т.д. Также, благодаря предложенному алгоритму унифицированного подбора гиперпараметров, созданное ПО реализует комплексную автоматизацию процессов генерации, обучения и использования моделей ИНС ПР различных архитектур, которая выражается в целенаправленном поиске гиперпараметров сети и в автоматическом создании программных оболочек для синтезированных моделей. Такой подход позволит использовать предложенное ПО при решении типовых задач распознавания непрофессионалам, которые не знают о деталях создания и

настройки нейронных сетей. Под гиперпараметрами в данном случае понимаются как переменные, определяющие структуру моделей ИНС (например, количество скрытых слоёв), так и переменные, определяющие процесс обучения моделей ИНС (например, скорость обучения) [112].

Результатом работы представленного ПО является не просто синтезированная модель, а сгенерированный исполняемый файл, дополненный программными обёртками, поддерживающими интерфейсы REST и SOAP, которые позволят запустить созданную ИНС как сервис и обращаться к ней из других систем и ПО. Данный подход соответствует принципу СОА [74] и позволит использовать достоинства этой парадигмы. В частности, горизонтальную и вертикальную масштабируемость, возможность повторного использования программного кода и модульную расширяемость, а также следование концепции Интернет 2.0. При этом само ПО создано с учётом концепции No-Code разработки, благодаря которой конечный пользователь может создавать законченные программные продукты без необходимости писать код. Всё это позиционирует представленное ПО как инструмент для быстрого и не затратного решения простых типовых задач обычными пользователями. На данный момент существующие аналоги представленного ПО либо решают задачу генерации программных оболочек для созданных моделей только для конкретной архитектуры ИНС, либо не решают ее вовсе.

К моменту начала работы над диссертацией, научная область, связанная с разработкой алгоритмов автоматизации процессов генерации и обучения моделей ИНС ПР, ещё только зарождалась. Этапы создания и эксплуатации моделей МО, требующие автоматизации, а также основные способы автоматизации этих этапов были сформулированы и опубликованы только в 2014 году международной командой во главе с лабораторией машинного обучения Фрайбургского Университета Альберта-Людвига [37]. Представленный в данной диссертации комплексный подход к автоматизации синтеза моделей ИНС ПР является одним из возможных подходов к решению описанных выше научно-технических задач. На данный момент множество групп разработчиков занимаются исследованием

схожих задач [81]. Но многие из них акцентируются на рассмотрении только конкретных задач (к примеру, на свёрточных нейронных сетях [2]) и не используют комплексный подход к автоматизации синтеза моделей ИНС ПР. Другие же научные группы ещё не представили итог своих исследований. Также, схожие работы ведутся научными группами крупных коммерческих компаний. Но данные работы являются частично или полностью закрытыми, в связи с чем можно провести лишь ограниченное сравнение с ними.

Предлагаемый для комплексной автоматизации процессов генерации и обучения моделей ИНС ПР, различных архитектур, алгоритм унифицированного подбора гиперпараметров является модификацией генетического алгоритма (ГА) [42] и входит в класс эволюционных алгоритмов, которые уже успешно используются для решения похожих задач [21, 135]. Однако, в данном случае впервые предлагается использовать реализацию эволюционного алгоритма не для автоматизации процесса генерации и обучения конкретной архитектуры ИНС, а в качестве универсального и унифицированного алгоритма, подходящего для генерации и обучения расширяемого множества архитектур ИНС ПР.

Используемая в предлагаемом ПО модульная расширяемая структура позволила на практическом уровне поддержать заложенную парадигму универсальности. Разработанное ПО может быть расширено новыми архитектурами ИНС ПР без модификации уже созданных программных модулей, что говорит об его открытости и является важнейшим преимуществом данной разработки. Универсальный программный интерфейс, независимость модулей архитектур друг от друга, а также инкапсуляция алгоритмов генерации и обучения моделей ИНС ПР внутри модулей, позволяет созданному ПО единообразно генерировать и обучать модели различных архитектур ИНС.

Цель исследования заключается в повышении степени автоматизации процесса создания, обучения и использования моделей ИНС прямого распространения различных архитектур.

Цель диссертационной работы достигается на основе комплексного решения следующих **научно-технических задач**:

- разработка алгоритма решения задачи автоматизации процессов генерации и обучения моделей ИНС прямого распространения различных архитектур;
- разработка архитектуры и программной системы автоматизации синтеза моделей ИНС прямого распространения с различными архитектурами;
- разработка архитектуры и программной системы автоматической генерации программных оболочек, поддерживающих парадигму сервис-ориентированного подхода и инкапсулирующих алгоритмы работы с созданными моделями ИНС.

Объектом исследования является процесс генерации, обучения и использования созданных моделей ИНС ПР в стороннее ПО.

Предметом исследования является новое программно-математическое обеспечение комплексной автоматизации синтеза моделей ИНС ПР, реализующих различные архитектуры и используемых для решения прикладных задач.

Область исследования соответствует паспорту специальности 2.3.5 - «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» по техническим наукам.

Научная новизна исследования результатов, полученных при решении поставленных задач, состоит в следующем:

1. Разработан алгоритм унифицированного подбора гиперпараметров (структурных параметров) для решения задачи автоматизации процессов генерации и обучения моделей ИНС прямого распространения различных архитектур, отличающийся масштабируемостью по отношению к новым архитектурам ИНС прямого распространения без необходимости модификации самого алгоритма.

2. Разработана архитектура и программная система автоматизации процессов генерации и обучения моделей ИНС прямого распространения с различными архитектурами, отличающаяся возможностью инкапсуляции методов работы с различными архитектурами ИНС, что позволяет упростить и ускорить разработку

моделей за счёт использования перспективной информационной технологии «программирования без программирования» (No-Code разработка).

3. Разработана архитектура и программная система автоматической генерации исполняемых файлов для синтезированных моделей ИНС прямого распространения с различными архитектурами, отличающаяся от существующих конструктивным использованием сервис-ориентированного подхода, а также концепции и технологии No-Code разработки, что позволяет ускорить и упростить интеграцию разработанных моделей в стороннее программное обеспечение.

Теоретическая и практическая значимость работы диссертации заключается в разработке методов, алгоритмов и их программной реализации, которые определяют основное содержание предложенного в диссертации нового подхода к комплексной автоматизации процессов генерации, обучения и использования ИНС ПР различных архитектур. В рамках данного подхода удалось создать архитектуру программной системы синтеза моделей ИНС ПР, которая характеризуется высокой степенью универсальности и унификации и поэтому может использоваться для комплексной автоматизации синтеза моделей различных архитектур ИНС ПР, в том числе не включённых по умолчанию в данное ПО. Также, реализована архитектура программной системы автоматической генерации программных оболочек для синтезированных моделей, которая упрощает и ускоряет внедрение созданных моделей в стороннее ПО, за счёт реализации концепции No-Code разработки и поддержки парадигмы СОА, что подтверждается результатами экспериментальных исследований.

С помощью использования созданного ПО удалось сэкономить ресурсы и сократить время в процессе разработки моделей ИНС ПР за счёт уменьшения потерь на промежуточных этапах их создания (таких этапов, как подбор гиперпараметров модели, генерация исполняемых файлов и т.д.). Также, предложенный комплексный подход к синтезу моделей ИНС ПР позволяет ускорить процесс исследования объекта на основе оценивания влияния новых входных параметров на модель, обученную на ретроспективных данных

(полученных от этого же объекта), за счёт уменьшения времени необходимого на синтез новых моделей.

С практической точки зрения, результаты данной диссертации (прежде всего, созданное ПО) можно использовать для автоматизации процессов создания программных модулей на базе моделей ИНС ПР, позволяющих решать широкий круг прикладных задач. Применимость ПО ограничивается только потенциальными возможностями каждой из архитектур ИНС ПР, включённых в него. При этом, список решаемых задач также может быть впоследствии увеличен, за счёт добавления новых архитектур ИНС ПР, обеспечивающих решение других классов задач.

Методология и методы исследования. Для решения поставленных задач в работе используются теория эволюционных алгоритмов, теории модульного программирования и теории глубокого обучения. При компьютерной реализации использовались методы и алгоритмы глубокого обучения, используемые в программных библиотеках Keras и TensorFlow, а также парадигмы No-Code разработки и сервис-ориентированной архитектуры (реализованной с помощью программной библиотеки Flask).

Положения, выносимыми на защиту:

1) Алгоритм унифицированного подбора гиперпараметров для решения задачи автоматизации процессов генерации и обучения моделей ИНС прямого распространения различных архитектур.

2) Архитектура и программная система автоматизации процессов генерации и обучения моделей ИНС прямого распространения различных архитектур.

3) Архитектура и программная система автоматической генерации исполняемых файлов для синтезированных моделей ИНС прямого распространения различных архитектур.

Степень достоверности результатов диссертации обеспечивается анализом состояния современных исследований по тематикам МО и автоматизированного обучения систем ИИ, подтверждается согласованностью полученных результатов,

успешной апробацией созданного ПО и последующим внедрением результатов его работы, а также 11 выступлениями на международных и российских научных конференциях и публикацией итогов исследований в ведущих рецензируемых изданиях.

Апробация и реализация результатов. Модели ИНС, созданные в результате использования разработанного ПО, были использованы при реализации информационной системы (ИС) ПРОСТОР [143]. Исследования, отражённые в диссертации, проводились в рамках 4 научно-исследовательских работ: 1) грант РФФИ №19-37-90112 «Разработка методов, технологии и программного комплекса автоматизированной генерации и обучения искусственных нейронных сетей на основе сервис-ориентированной архитектуры»; 2) грант РФФИ №19-08-00989 «Разработка и исследование научных основ теории многокритериального оценивания, анализа и управления качеством моделей и полимодельных комплексов, описывающих сложные технические объекты»; 3) грант РФФИ №17-08-00797 «Разработка и исследование методологических основ и технологии комплексного моделирования процессов функционирования системы проактивного управления сложными техническими объектами»; 4) грант РФФИ №16-08-00510 «Разработка и исследование методологии построения и создание прототипа информационной автоматизированной системы прогнозирования состояния растительного покрова Крайнего Севера на основе интегрированной обработки мульти- и гиперспектральных наземно-аэрокосмических данных, а также климатической информации».

Основные результаты работы докладывались и обсуждались на 11 научно-практических конференциях: «32nd European Conference on Modelling and Simulation» (ECMS 2018), г. Вильгельмсхафен, Германия, 2018 г.; «20th International Conference on Harbor, Maritime & Multimodal Logistics Modelling and Simulation» (HMS2018), г. Будапешт, Венгрия, 2018 г.; «Глобальные климатические изменения: региональные эффекты, модели, прогнозы», г. Воронеж, 2019 г.; «33rd European Conference on Modelling and Simulation» (ECMS 2019), г. Вильгельмсхафен, Германия, 2019 г.; «13th IEEE International Conference

“Application of Information and Communication Technologies”» (AICT2019), г. Баку, Азербайджан, 2019 г.; «2nd Euro-Mediterranean Conference for Environmental Integration» (EMCEI-2), г. Сус, Тунис, 2019 г.; «9th IFAC Conference on Manufacturing Modelling, Management and Control» (MIM 2019), г. Берлин, Германия, 2019 г.; «Модели и методы исследования информационных систем на транспорте» (ММРИСТ-2020), г. Санкт-Петербург, 2020 г.; «18 Национальная Конференция по Искусственному Интеллекту с Международным Участием» (КИИ-2020), г. Москва, 2020 г.; «Математическое моделирование в экологии» (ЭкоМатМод-2021), г. Пущино, 2021 г.; «Pattern Recognition and Information Processing» (PRIP'2021), г. Минск, Беларусь.

Публикации. По теме диссертации опубликовано 25 печатных работ, включая 3 публикации в журналах из перечня рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание учёной степени кандидата наук, 10 публикаций в изданиях, индексируемых в WoS/Scopus, 2 свидетельства о регистрации ПрЭВМ в Роспатент. Полный перечень публикаций соискателя по теме исследования представлен в Приложении А диссертационной работы.

Разработанное алгоритмическое и программное обеспечение реализовано в ряде организаций: Государственный природный биосферный заповедник «Таймырский», «Санкт-Петербургский государственный технологический институт (технический университет)» СПбГТИ (ТУ), в СПИИРАН и СЗЦППО входящих в СПб ФИЦ РАН.

Личный вклад автора. Основные научные положения, теоретические выводы и практические решения, результаты тестирования сформулированы и изложены автором самостоятельно.

Структура и объём работы.

Диссертация объемом 144 машинописные страницы содержит введение, четыре главы и заключение, список литературы (148 наименование), 1 таблицу, 32 рисунка, приложение со списком публикаций соискателя.

В первой главе проведён анализ текущего состояния исследований в области практического применения ИНС, имеющиеся проблемы в рассматриваемой прикладной области и возможные пути их решения, проводится формальная постановка задач исследования.

В главе обосновывается актуальность избранной темы диссертации, которая ориентирована на повышение автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур, а также на упрощение интеграции созданных моделей в стороннее ПО. Решать поставленные задачи предлагается за счёт комплексной автоматизации, а также с помощью использования парадигм и технологий СОА, в процессе генерации программных оболочек для созданных моделей, и No-Code разработки, на всех этапах работы.

Во второй главе производится анализ часто используемых моделей ИНС ПР, автоматизация процессов генерации и обучения которых на сегодняшний день наиболее оправдана, описываются подходы к процессам генерации и обучения моделей ИНС, которые требуется автоматизировать.

В главе подробно описываются выбранные архитектуры ИНС ПР и обосновываются причины их выбора для реализации в разрабатываемом ПО. Данные архитектуры предоставляются пользователю на выбор при решении соответствующих задач. В диссертации предлагается почти полностью автоматизировать работы, связанные с генерацией и обучением данных архитектур. При таком подходе от пользователя требуется только наличие размеченной обучающей выборки и указание требуемой точности получения конечных результатов.

Также, в данной главе приводится описание разработанного алгоритма унифицированного подбора гиперпараметров, который используется для комплексного и унифицированного синтеза моделей ИНС ПР различных архитектур и описываются его отличительные особенности.

В третьей главе подробно рассматриваются состав и структура разработанного ПО. Описываются задачи, решаемые каждым из созданных программным модулем, а также способы решения этих задач.

Формулируются базовые принципы СОА и обосновываются причины применения данной парадигмы в разрабатываемом ПО. Приводится подробное описание 11 основных и вспомогательных программных модулей, разработанных диссертантом и вошедших в состав созданного ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур.

В четвёртой главе приводится описание трех прикладных задач, решённых при помощи созданного ПО.

В первой представленной задаче рассматривается процесс весеннего вскрытия льда и последующего половодья на реках. Помимо зависимости от метеорологических и гидрологических параметров, существенное влияние на этот процесс оказывают и географические особенности пространственного размещения самих рек. Для прогнозирования поведения реки в половодье была разработана ИС «ПРОСТОР». Используемые в этой системе модели и методы используются для оперативного прогнозирования состояния речного русла. С помощью представленного в диссертации ПО были сгенерированы модели ИНС, которые успешно применялись для решения задач прогнозирования дня вскрытия льда и последующего прогнозирования уровня воды.

Второй решенной задачей была задача оценивания запасов надземной фитомассы растительных сообществ разных природно-климатических зон и прогнозирование их изменений в зависимости от факторов среды. Данная задача важна, прежде всего, для сельских и природоохранных хозяйств. Используя знания о запасах фитомассы, можно оценить кормовую емкость пастбищ и рассчитать предельную численность домашних или диких животных для длительного устойчивого природопользования. Для решения рассматриваемой задачи была создана модель ИНС (с помощью представленного в диссертации ПО), с использованием которой был осуществлен прогноз запасов надземной фитомассы (максимальной величины фитомассы в период вегетации) растительного сообщества тундры как сложной слабо формализованной системы, в зависимости от погодно-климатических условий. При этом использован прием перевода

статистически слабо представленного материала наземных наблюдений в область более точных и статистически достоверных по объему данных космических снимков одного и того же района исследований.

Третьей задачей была задача по учету численности диких северных оленей тундровых популяций (таймырской, якутских популяций, оленей Чукотки, а также мигрирующих стад северных оленей Канады и Аляски) на основе анализа аэрофотоснимков. В качестве модели распознавания было принято решение использовать Mask R-CNN. Данная модель также была сгенерирована и обучена с помощью представленного в диссертации ПО. После обучения сгенерированная СНС стала успешно решать задачу распознавания и подсчёта северных оленей на основе данных, представленных на аэрофотоснимках.

На основе обобщения результатов выполненных исследований была представленных примерах была также разобрана разработана методика использования ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур для решения прикладных в конкретных предметных областях.

1 СИСТЕМНЫЙ АНАЛИЗ ПРОБЛЕМ ПРАКТИЧЕСКОГО ПРИМЕНЕНИЯ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ. ФОРМАЛЬНАЯ ПОСТАНОВКА ЗАДАЧИ КОМПЛЕКСНОЙ АВТОМАТИЗАЦИИ СИНТЕЗА ДАННЫХ СЕТЕЙ

1.1 Анализ современного состояния исследований в области практического применения ИНС

Отличительной особенностью современного процесса информатизации и автоматизации сложных прикладных областей является активное применение комбинированных алгоритмов МО для решения различных классов важных и актуальных прикладных задач, связанных с анализом и синтезом сложно описываемых объектов и процессов (наличие большого количества формализуемых параметров и/или наличие неявных и комплексных формализуемых параметров). Эта особенность проявляется в условиях существования большого количества важных прикладных задач, решение которых существующими алгоритмами обработки и анализа данных является неоправданно ресурсоёмким. Яркими примерами таких задач может служить многофакторный анализ [78, 95, 107], распознавание объектов на изображениях [29, 83, 96], фильтрация шумов в данных различных форматов [88, 94, 144] и другие задачи.

Существуют следующие наиболее распространённые типовые задачи, решаемые алгоритмами МО (рисунок 1.1):

- регрессионный анализ – предсказание значений одной или нескольких выходных переменных на основание значений одной или нескольких входных переменных;
- классификация – предсказание принадлежности вектора входных данных одному из заранее описанных классов;
- кластеризация – разбиение входного набора данных по однородным группам;
- поиска аномалий – поиск в массиве данных нетипичных векторов входных данных;

- сокращение размерности исходного пространства данных – уменьшение количества переменных в массиве данных с сохранением информационного наполнения.

Данные задачи были сформулированы уже давно и существует множество алгоритмов их решения. Но оставались некоторые прикладные области (к примеру обработка изображений), для которых имеющиеся на тот момент алгоритмы требовали такого объёма вычислительных ресурсов, который бы никогда не окупился по критерию «эффективность-стоимость» при последующем практическом использовании полученных результатов. В указанных условиях особую актуальность приобретают вопросы разработки нового класса моделей и алгоритмов, позволяющих решать указанные задачи с заданной точностью и с меньшими затратами ресурсов на разработку соответствующего ПО и его устойчивое функционирование впоследствии.

Алгоритмы МО стали таким классом алгоритмов. Ключевой особенностью всего класса данных алгоритмов является то, что они не моделируют напрямую какой-либо процесс или объект. Вместо этого они генерируют тем или иным образом, в зависимости от алгоритма, формулы, которые для заданного набора входных данных должны позволять рассчитывать такие выходные данные, которые были бы наиболее близкими по своим значениям к целевым (требуемым) выходным данным. Т.е. алгоритмы МО, в общем случае, на этапе обучения решают задачу генерации (синтеза) функции, восстанавливающей зависимости по эмпирическим данным [50, 69, 87].

Рисунок 1.1 – Примеры использования алгоритмов МО в современном мире

Таким образом, разработанные алгоритмы МО не привязаны к параметрам процесса или объекта, для которых они генерируют функции. Данный класс алгоритмов также не привязан к конкретной прикладной области. Это позволяет использовать одни и те же алгоритмы для создания моделей, используемых для решения принципиально разных задач в процессе человеческой деятельности.

Конечно, стоит упомянуть и то, что данные алгоритмы имеют специфические ограничения в своей работе. Для того, чтобы генерировать функции эти алгоритмы должны иметь информацию об объекте. К примеру, в процессе обучения с учителем (при наличие эталонных выходных значений) для алгоритмов МО такой информацией является набор пар входных и соответствующих им выходных векторов данных, на основании которых алгоритмы настраивают генерируемые функции так, чтобы они рассчитывали требуемые выходные значения для заданных входных. Вследствие чего, алгоритмы МО невозможно применять для генерации функций в тех случаях, когда отсутствует достаточный объём обучающих данных

применительно к конкретному процессу или объекту. Причём, достаточность этого объёма данных сугубо ситуативная и для различных сочетаний задач и алгоритмов будет отличаться.

Тем не менее, набор задач, удовлетворяющих данному требованию, всё равно большой, что и объясняет актуальность и востребованность алгоритмов МО во многих современных отраслях человеческой деятельности.

ИНС, в свою очередь, являются подклассом алгоритмов МО и входят в класс алгоритмов глубокого обучения (ГО) (рисунок 1.2). Принципиальное отличие данного класса алгоритмов заключается в том, что в отличие от других алгоритмов МО, алгоритмы ГО не требуют предварительного анализа обучающих данных [147]. На вход алгоритмы ГО получают массив данных единообразного формата. При этом зависимости между этими данными и степень их влияния на конкретные выходные параметры данный класс алгоритмов устанавливает уже самостоятельно, в процессе обучения.

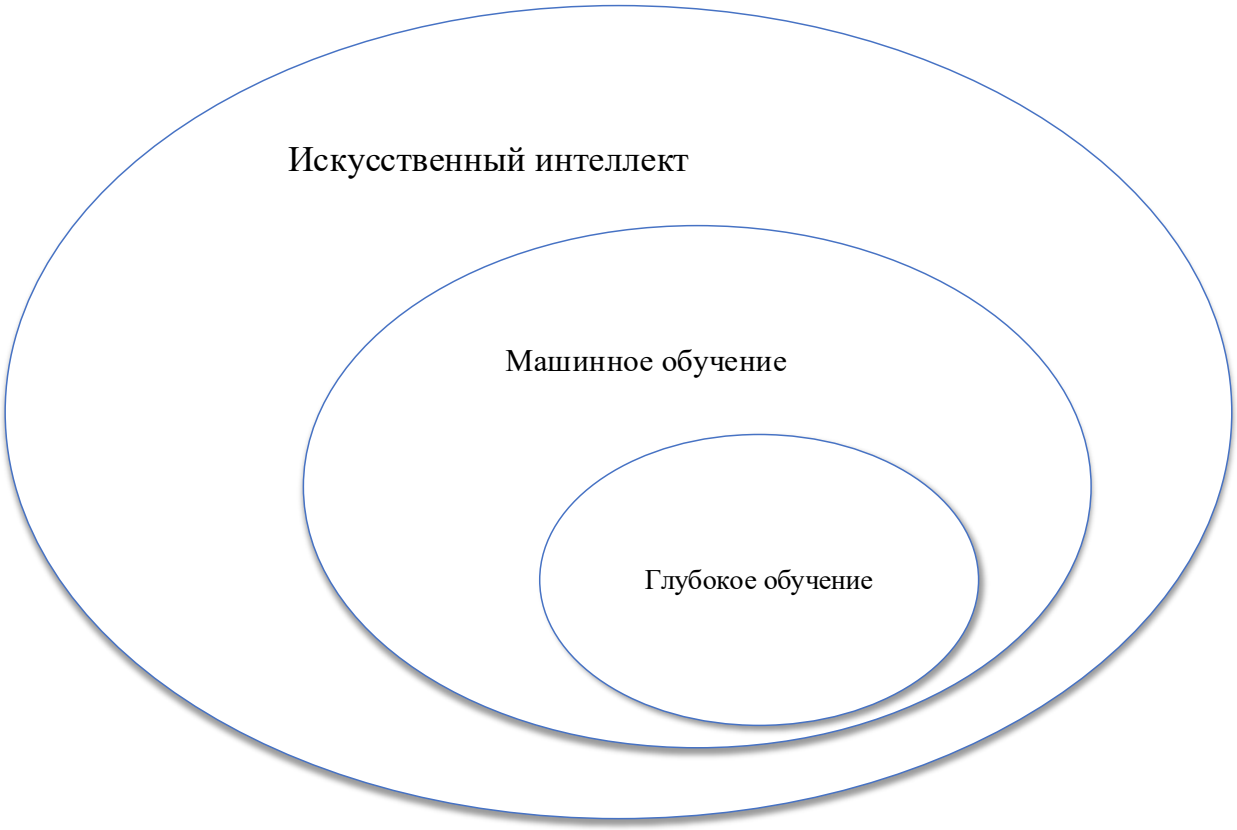

Рисунок 1.2 – Диаграмма Эйлера для ГО

Описанные особенности определяют и конкретную область применения алгоритмов ГО в целом и ИНС в частности. Алгоритмы ГО применяется при решении задач, связанных с «большими данными». Данный класс задач характеризуется наличием большого числа входных параметров (чаще всего, количество таких параметров начинается от нескольких десятков), взаимосвязь которых как между собой, так и относительно выходных параметров заранее не известна. Ярким примером таких задач могут служить:

- задачи анализа сложно описываемых объектов и процессов в режиме реального времени [23, 51, 108, 140], которые являются актуальными практически в любой промышленности;
- задачи, связанные с логистическими операциями любого рода [76, 119, 136];
- задачи финансового и поведенческого анализа [40, 72, 89];
- задачи, связанные с обеспечением информационной безопасности [102, 113];
- задачи моделирования и анализа функционирования систем здравоохранения [47, 91, 137];
- задачи мониторинга в области охраны природы и природопользования [3, 18, 63];
- задачи космической отрасли [8, 9].

Описанные примеры не исчерпывают области приложения ИНС, но иллюстрируют тот факт, что единообразный подход к генерации функций позволяет применять одни и те же архитектуры ИНС в принципиально разных прикладных областях. Таким образом, других алгоритмов, которые бы могли решать существующие сложные прикладные задачи с неструктурированными входными данными с той точностью, которая бы оправдывала практическое применение подобных алгоритмов, на сегодняшний день нет. ИНС зачастую являются безальтернативным способом решения подобных задач.

При этом, архитектуры ИНС продолжают активно развиваться и сегодня [61, 116, 124]. Разрабатываются новые алгоритмы и модифицируются уже имеющиеся, что повышает точность решения одних задач и позволяет приступить к решению новых. Всё это приводит к росту количества областей человеческой деятельности, в которых можно применять алгоритмы МО и ГО. Тем не менее, существует и ряд проблем, возникающих на этапе решения прикладных задач.

1.2 Обзор проблем практической реализации алгоритмов ИНС

Несмотря на множество успешных апробаций при решении сложно описываемых задач, применение ИНС для практических задач на сегодняшний день имеет ряд ограничений. Прежде всего, данные ограничения вызваны недостатком квалифицированных специалистов, способных корректно сгенерировать и обучить модель ИНС. Применение ИНС требует глубоких знаний в области математического анализа и статистики. Это делает невозможным их использование людьми, не обладающими такими знаниями. В результате, множество задач, которые можно было бы решать с помощью ИНС, остаются нерешёнными просто потому, что специалисты в данной области не знают о существовании архитектур ИНС, которые они могли бы использовать в своей работе.

С другой стороны, иногда при решении различных классов прикладных задач с использованием ИНС специалисты, применяющие данный математический аппарат и технологию, реализуют их на практике без полного понимания принципов работы выбранных алгоритмов. Они берут, по аналогии, готовую программную реализацию, предназначенную для использования в похожих задачах. Но данный подход, зачастую, не позволяет эффективно использовать выбранные ИНС при глубоких и неявных отличиях в объекте моделирования, которые приводят к отличиям во входных и выходных параметрах, а также ко множеству скрытых проблем, которые проявляются уже на этапе эксплуатации.

При этом, многие модели ИНС создаются в виде программных прототипов (к примеру, с помощью пакета программ MatLab) и такие прототипы требуют доработки для внедрения в существующее ПО, спроектированное на конкретных стеках прикладных программных языков (C++, Java, Python и т.д.). Это усложняет дальнейшее развитие и последующее внедрение создаваемых прототипов.

Подводя итог, можно сказать, что процессы развития и внедрения ИНС в промышленное ПО и повседневную жизнь человека ограничивают как минимум три серьёзные проблемы.

Во-первых, существует огромное разнообразие архитектур ИНС, которые, на сегодняшний день, слабо классифицированы. Имеется множество разнообразных архитектур, которые могут применяться для совершенно разных задач. Однако только начинают появляться платформы, которые бы на постоянной основе отслеживали бы разработку новых решений в области ИНС и актуализировали информацию по существующим алгоритмам. Простой пользователь, зачастую, просто не знает, что существуют архитектуры ИНС, способные решить поставленные перед ним задачи. При этом даже узнав об их существовании, зачастую он не сможет с ходу выбрать конкретную архитектуру. Проведенный анализ показывает, что не существует методического и программного обеспечения для комплексного сравнения всех архитектур ИНС, способных решать схожие задачи. Поэтому на сегодняшний день специалистам нередко приходится выбирать конкретную архитектуру ИНС методом проб и ошибок. В целом рассматриваемая проблема усугубляется очень быстрым развитием теоретической базы и регулярно появляющимися новыми архитектурами.

Во-вторых, не существует унифицированного подхода к реализации и обучению различных архитектур. Разные классы ИНС генерируются и обучаются совершенно разными алгоритмами. При этом довольно часто эти алгоритмы приходится модифицировать при решении новой прикладной задачи. Это всё требует обширных узкоспециализированных знаний. Поэтому область знаний ИНС имеет высокий порог вхождения и простые пользователи не смогут с ходу применять методы и алгоритмы данной области знаний с полной эффективностью.

В-третьих, даже уже имеющиеся решения часто реализованы в виде прототипов. Внедрение подобных прототипов будет требовать их глубокой модификации под конкретную задачу, а также написания программных интерфейсов. Часто, при модификации подобных прототипов, требуется хорошо разбираться в архитектуре ИНС, которая реализована в данном прототипе, что возвращает ко второй описанной проблеме.

Количество проблем, которые возникают в процессе внедрения алгоритмов ИНС, не исчерпываются описанными. Однако, данные проблемы имеют наиболее общий характер и встречаются в большинстве областей человеческой деятельности. Поэтому, решение именно описанных проблем может упростить и ускорить внедрение алгоритмов ИНС во многих областях человеческой деятельности.

1.3 Обзор подходов к решению проблем практической реализации алгоритмов синтеза моделей ИНС

На сегодняшний день многие исследователи и разработчики пришли к идее, что возможно осуществить автоматизацию процессов генерации и обучения моделей ИНС, что позволило бы решить описанные в предыдущем разделе проблемы. К настоящему моменту времени уже имеется ряд работ на данную тему [43, 49, 57, 135]. Все они сводятся к тому, что автоматизация процессов генерации и обучения моделей МО и ГО позволит ускорить процесс разработки программных продуктов для решения множества актуальных прикладных задач. При этом группой учёных уже было сформулировано определение новой научной области [36], базовые задачи которой заключаются в разработке и реализации методологий и технологий создания и использования моделей и алгоритмов автоматизации процессов генерации и обучения моделей МО – automated machine learning (AutoML).

В качестве базовых задач AutoML были выбраны следующие задачи:

- структурно-параметрический синтез модели и оптимизация гиперпараметров;

- обучение представлению и автоматическое извлечение/построение признаков;
- автоматическое создание рабочих процессов машинного обучения;
- мета-обучение и трансферное обучение;
- автоматическое распознавание решаемой задачи в необработанных данных;
- кодирование/преобразование функций в соответствии с требованиями различных алгоритмов обучения;
- автоматическое обнаружение и обработка искаженных данных и/или отсутствующих значений;
- автоматическое обнаружение утечек процесса обучения;
- сопоставление проблем с методами/алгоритмами (помимо регрессии и классификации);
- автоматический сбор новых данных;
- автоматическое написание отчетов (предоставление информации об анализе данных, выполненном автоматически);
- создание пользовательских интерфейсов для AutoML;
- автоматический вывод и дифференцирование результатов работы алгоритмов машинного обучения;
- автоматический выбор показателей оценки;
- автоматическое создание наборов обучения, валидации и тестирования;
- создание безпараметрических алгоритмов;
- автоматический выбор алгоритмов для удовлетворения ограничений по времени/пространству/мощности во время обучения или во время эксплуатации модели;
- создание оболочек защиты во время эксплуатации модели для обнаружения сдвига данных и других причин сбоя прогнозирования.

Поскольку данная область является новой и не так много научных и проектных групп работают в ней, пока что отсутствуют устоявшиеся подходы к описанию и решению различных классов задач AutoML. Каждая из входящих в AutoML задач сама по себе является сложно формализованной и требует индивидуального подхода к поиску решения. На сегодняшний день начинают появляться алгоритмы, позволяющие решать эти задачи по отдельности, однако ПО, с помощью которого можно было осуществить комплексную автоматизацию всех перечисленных процессов, пока не существует.

Имеющиеся на сегодняшний день разработки чаще всего не выходят за рамки автоматизации генерации и обучения моделей отдельных архитектур ИНС, к примеру СНС. Такой подход зачастую не решает первой из описанных в подразделе 1.2 проблем – такие системы возможно задействовать только в тех случаях, когда пользователь уже знает про особенности различных архитектур, может провести их сравнение и выбрать конкретную, для решения своих задач. Тем не менее, представленные примеры показывают возможность применения данного подхода для решения перечисленных задач AutoML и позволяют сделать вывод об оправданности дальнейших разработок в данном направлении.

Для комплексной автоматизации решения задач генерации и обучения моделей ИНС можно применять различные алгоритмы, используемые при синтезе моделей. Однако, в силу существенных различий между разными классами архитектур ИНС, точность работы таких алгоритмов для разных архитектур будет отличаться. К примеру, если в случае с сетями Румельхарта стоит задача подбора таких параметров как количество скрытых слоёв, количество нейронов в скрытых слоях, скорости обучения и т.п., то для самоорганизующихся карт уже необходимо подбирать параметры, связанные с позиционированием нейронов. Всё это, с одной стороны, приводит к невозможности применения одних и тех же алгоритмов к различным архитектурам ИНС без дополнительных модификаций, а с другой накладывает требование к универсальности на сами алгоритмы синтеза.

С другой стороны, развивается и среда, в которой должны функционировать разработанные модели ИНС. Современные методы и подходы к решению задач

мониторинга, прогнозирования, поддержки принятия решений и управления в различных областях человеческой деятельности автоматизируются всё активнее. Автоматизированный мониторинг состояния сложных объектов с участием людей-специалистов на местах всё ещё продолжается использоваться на практике, однако в различных отраслях всё активнее начинают применяться автоматические информационные станции и системы, подключённые к глобальной сети Интернет. Данные системы функционируют полностью без участия человека и при этом способны в режиме реального времени пересылать данные мониторинга в любую точку планеты.

При этом проведенный анализ показывает, что поскольку для пересылки данных используются публичные сети, наиболее продуктивным подходом к проектированию многокомпонентных программных комплексов, использующих данные с автоматических станций и систем мониторинга, являются СОА [132] и ИВ [59]. Оба подхода имеют схожие требования к отдельным сервисам, которые включены в подобные программные комплексы. Эти требования распространяются не только на поставщиков данных, но и на сервисы обработки и анализа данных, которые также должны быть разработаны с учётом работы в публичных сетях.

Всё это приводит к тому, что на ПО, реализующее синтез моделей ИНС, накладываются не только внутренние ограничения, связанные с процессами генерации и обучения самих моделей, но и внешние, которые требуют от созданных моделей возможности функционирования в распределённых многокомпонентных системах. Поэтому, несмотря на то что внутренние и внешние требования порождаются принципиально разными причинами и напрямую друг от друга не зависят, на практике все они должны обязательно выполняться. В противоположном случае (если ряд ограничений не выполняется) созданное ПО будет иметь крайне узко специализированное приложение, либо будет требовать наличия специалиста по алгоритмам ИНС для последующей доработки созданных моделей и их внедрения на практике, что лишит смысла саму комплексную автоматизацию процессов генерации и обучения моделей ИНС.

Стоит также отметить, что в одно и тоже время с моментом начала проведения исследований в рамках данной диссертацией были развернуты сторонние разработки прикладных программных библиотек ПО, решающих схожие задачи. Примерами таких библиотек могут служить auto-sklearn [38], AutoKeras [34] и ряд других. Однако, данные системы либо могут использоваться только на одной конкретной платформе, либо представлены исключительно в виде программной библиотеки, как auto-sklearn и auto-Keras, что не позволяет использовать их как самостоятельные программные решения. Также, в качестве аналога можно привести и расширение для платформы MatLab [35], которое обеспечивает как автоматизацию процессов обучения моделей, так и компиляцию созданных моделей в исполняемые файлы на C++. Но тут появляется еще одна проблема, связанная с созданием модели, поскольку для работы с данной платформой требуется наличие специалиста, имеющего опыт работы с MatLab. К тому же, автоматизация процесса обучения на данной платформе будет всё ещё не полной, а такой специалист должен будет разбираться в базовых концепциях МО для первичной настройки моделей. С другой стороны и компиляция модели на языке C++ далеко не всегда является достоинством. Если для простых моделей ИНС каких-то проблем при решении прикладных задач не будет, то массивные сети, такие как СНС, требуют использования специфических аппаратных ускорителей. Иначе обучение таких моделей может затянуться на месяцы, а то и годы. Решение платформы MatLab на сегодняшний день не имеет интеграции со многими из них. Поэтому несмотря на то, что C++ является низкоуровневым языком, скомпилированные с помощью MatLab модели СНС могут обучаться на порядок дольше, чем модели созданные, к примеру, на языке Python, который имеет программные интеграции со многими аппаратными ускорителями.

Близким аналогом ПО, представленного в данной диссертации, можно считать систему FEDOT [21], разработанную в исследовательском центре университета ИТМО. Несмотря на то, что центром данной системы является программная платформа, решающая задачу автоматизации создания моделей ИНС для различных задач, она имеет и графический пользовательский интерфейс, что

упрощает её эксплуатацию. Также, данная программная платформа построена на принципах масштабируемости, поэтому может быть расширен различными архитектурами ИНС в дальнейшем. Однако, данная система всё ещё является инструментом для профессионалов в области ИНС и с трудом может использоваться не специалистами. Стоит отметить и то, что, на сегодняшний день, результатом работы этой системы является программный прототип, который требуется дорабатывать для интеграции в стороннее ПО.

Подводя итог, можно сказать, что на сегодняшний день уже имеются разработки и исследования, которые точечно решают описанные в подразделе 1.2 проблемы. Однако, всё ещё отсутствуют разработки, которые бы осуществляли ***комплексный подход к автоматизации решения*** всех описанных выше проблем. Поэтому при решение всех описанных проблем в рамках одного ПО можно будет говорить о принципиально новых его возможностях, позволяющих любому пользователю, не обладающему знаниями в области МО, формировать модели и алгоритмы, обеспечивающие решение конкретных прикладных задач, поставленных перед этим пользователем.

1.4 Постановка задачи комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур

На содержательном уровне задача комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур может быть сформулирована следующим образом: для предоставленной на вход обучающей выборки должна быть выбрана, обучена (по принципу обучения с учителем, т.е. на основе эталонных выходных значений) и запрограммирована модель такой архитектуры ИНС ПР с такими значениями варьируемых параметров (гиперпараметров), которая бы на предложенной тестовой выборке показала бы наилучшую точность работы, либо точность работы не ниже заданного порога.

Под варьируемыми параметрами (гиперпараметрами) ИНС подразумеваются те параметры, которые могут быть изменены на этапах генерации и обучения

моделей ИНС, с целью повышения точности её работы. Примером таких параметров может быть: количество скрытых слоёв, количество нейронов в скрытых слоях, функции активации в нейронах, функции потерь и т.п. Поскольку для каждой конкретной архитектуры ИНС прямого распространения количество и набор таких гиперпараметров будет отличаться, на уровне комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур можно говорить только о *наборе гиперпараметров*. Для каждой конкретной архитектуры ИНС такой набор будет отличаться.

Под точностью работы в данном случае понимается точность прогнозирования выходных значений моделируемых величин, предоставленных пользователем в тестовой выборке. Поскольку для различных архитектур формат входных и выходных данных может отличаться (от целочисленного и бинарного до текстового и фото-/видеоформата) на уровне комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур целесообразно говорить только об абстрактной точности прогнозирования выходных значений моделируемых величин, характеризующих исследуемый сложный объект или процесс. При этом, для каждой конкретной архитектуры ИНС ПР подразумевается наличие своего собственного метода оценки точности прогнозирования выходных значений, результаты которого учитываются алгоритмом верхнего уровня.

За основу постановки задачи повышения степени автоматизации процессов создания, обучения и использования моделей ИНС прямого распространения различных архитектур, был взят ГОСТ 23004-78 «Механизация и автоматизация технологических процессов в машиностроении и приборостроении» [4]. С учетом содержания данного документа формальная постановка целевой задачи может быть представлена следующим образом:

$$card\ A\left(a, L^a(p, g^a)\right) \rightarrow\ max_{A \subseteq \acute{\mathrm{P}}(O)}, \tag{1.1}$$

где A – множество автоматизированных первичных частей (например, этапов технологических процессов) процесса создания, обучения и использования моделей ИНС прямого распространения различных архитектур;

a – индекс (номер) архитектуры ИНС ПР, используемой для решения поставленной пользователем задачи;

L^a – функция (алгоритм) автоматического подбора гиперпараметров ИНС ПР архитектуры a;

p – обучающая выборка (включающая в себя данные для валидации моделей), предоставленная пользователем и используемая в процессах генерации и обучения моделей ИНС ПР различных архитектур;

g^a – набор гиперпараметров модели ИНС ПР архитектуры с номером a, варьируемых в процессе обучения;

O – множество всех первичных частей процесса создания, обучения и использования моделей ИНС прямого распространения различных архитектур;

Ṕ(O) – булеан множества всех первичных частей процесса создания, обучения и использования моделей ИНС прямого распространения различных архитектур.

Тогда, определение степени автоматизации процессов создания, обучения и использования моделей ИНС прямого распространения различных архитектур можно вычислить по формуле:

$$D = \frac{card\ A(a,L^a(p,g^a))}{card\ O(a,L^a(p,g^a))}, \quad (1.2)$$

где D – показатель степени автоматизации первичных частей технологического процесса создания, обучения и использования моделей ИНС прямого распространения различных архитектур.

Для достижения поставленной выше задачи необходимо отдельно решить подзадачу автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур. Формальная постановка этой задачи имеет следующий вид:

$$Q^a(N_i^a\left(p, L^a(p, g^a)\right), q) \rightarrow max_{i\in[1,n]}, \quad (1.3)$$

при выполнении условия:

$$P_i(N_i^a) \geq P_{\text{зад}},$$

где p – обучающая выборка (включающая в себя данные для валидации моделей), предоставленная пользователем и используемая в процессах генерации и обучения моделей ИНС ПР различных архитектур;

q – тестовая выборка, предоставленная пользователем как входные данные для разрабатываемой программной системы автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур;

a – индекс (порядковый номер) архитектуры ИНС ПР, используемой для решения поставленной пользователем задачи;

g^a – набор гиперпараметров модели ИНС ПР архитектуры с номером a, варьируемых в процессе обучения;

L^a – функция (алгоритм) автоматического подбора гиперпараметров ИНС ПР архитектуры a;

N_i^a – i-ый экземпляр модели ИНС ПР архитектуры a, который был обучен на выборке p и с гиперпараметрами, полученными в процессе работы соответствующей функции (алгоритма) автоматического подбора гиперпараметров архитектуры ИНС (L^a);

i – индекс экземпляра модели ИНС ПР;

n – заданное количество экземпляров генерируемых и обучаемых моделей ИНС ПР, среди которых производится поиск модели;

Q^a – функция расчёта качества (в том числе, и точности) работы i-ого экземпляра модели ИНС ПР архитектуры a (N_i^a) на тестовой выборке q;

P_i – функция расчёта точности обученной модели;

$P_{зад}$ – заданное пользователем граничное значение требуемой точности обученной модели.

Иными словами, задача комплексной автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур относится к классу задач синтеза моделей сложных объектов или процессов, которые исследуются и решаются с использованием методов и алгоритмов, разрабатываемых в современной теории оценивания качества моделей и полимодельных комплексов (квалиметрии моделей и полимодельных комплексов) [17].

Анализ показывает, что наряду с задачей (1.3) также, требуется решить следующую дополнительную технологическую задачу, которая связана с разрабатываемым ПО и состоит в упрощении процесса интеграции сгенерированных и обученных моделей ИНС ПР в стороннее ПО. Данная задача в диссертации не описана формально, поскольку носит исключительно прикладной характер и её решение заключается в создании требуемых программных оболочек и программных интерфейсов.

1.5 Анализ и обоснование путей решения сформулированных задач

Для решения сформулированных задач требуется, прежде всего, разработать ПО, позволяющее единообразно в автоматизированном режиме на основе предложенной унифицированной информационной технологии генерировать и обучать модели различных архитектур ИНС ПР с использованием предоставленных пользователем обучающей и тестовой выборок, а также обеспечивающее автоматическое создание (генерацию) для синтезированной модели ИНС программной оболочки, упрощающей последующую интеграцию в стороннее ПО.

Подразумевается, что разрабатываемое ПО будет работать с различными архитектурами ИНС ПР. Для комплексной автоматизации процессов их генерации и обучения должен быть разработан унифицированный алгоритм, с помощью которого можно было бы единообразно проводить генерацию и обучение моделей различных архитектур.

Поскольку для различных архитектур отличается и количество, и тип гиперпараметров, для автоматизированного обучения можно рассматривать только алгоритмы способные работать с динамически расширяемым набором переменных. Предполагается, что при внедрении в разрабатываемое ПО новой архитектуры ИНС ПР не потребуется вносить какие-либо модификации в алгоритм. Программа должна будет воспринять новый массив переменных, вне зависимости от их числа и типа данных, и автоматически осуществлять

варьирование этих переменных в процессе автоматизированного обучения модели ИНС.

Удовлетворяющими этому условию оказались алгоритмы, относящиеся к типам эволюционных алгоритмов и алгоритмов роевого интеллекта. Данные алгоритмы были выбраны прежде всего потому, что эффективно и не ресурсоёмко реализуют случайный направленный поиск при решении задачи оптимизации с динамическим набором переменных. Также, достоинством этих групп алгоритмов была их универсальность, поскольку для поставленной задачи условия оптимального поиска будут варьироваться в зависимости от архитектуры ИНС ПР. Наиболее подходящими для поставленной задачи оказались: генетический алгоритм, алгоритм пчелиного роя и клональный алгоритм отбора (относится к классу алгоритмов искусственной иммунной системы).

Таблица 1.1 – Сравнение возможностей алгоритмов

Характеристика алгоритма	Генетический алгоритм	Алгоритм пчелиного роя	Клональный алгоритм отбора
Возможность случайного поиска	+	ограниченная	+
Возможность работы с динамическим набором переменных	+	+	ограниченная
Возможность масштабирования алгоритма	+	+	ограниченная

Из приведённой таблицы видно, что алгоритмы пчелиного роя и клонального отбора имеют определённые ограничения. В частности, алгоритм пчелиного роя начальную точку поиска выбирает случайно, но дальнейший поиск ведёт в области вокруг этой точки (рисунок 1.3). В то время, как клональный и генетический алгоритмы с самого начала работы осуществляют поиск во всём доступном пространстве решений.

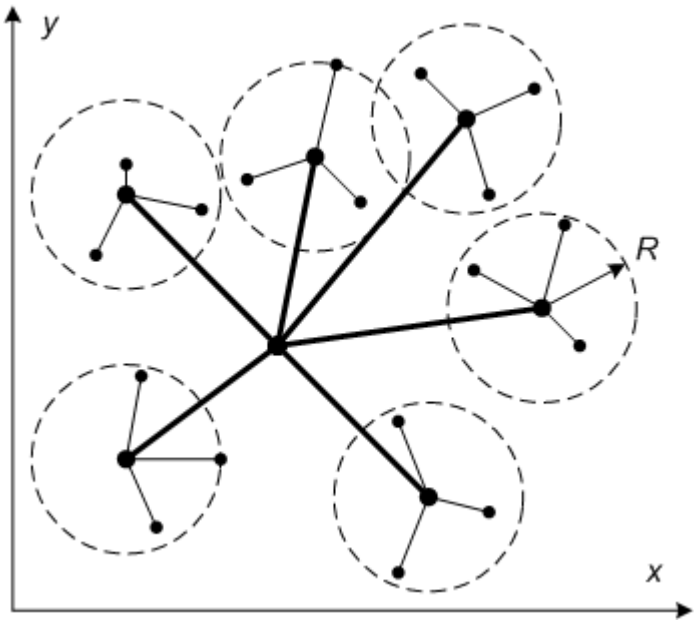

Рисунок 1.3 - Схематичное изображение стратегии разведки двумерного пространства алгоритмом пчелиного роя (жирные линии — вылеты разведчиков, тонкие линии — уточнение решений рабочими пчелами)

С другой стороны, клональный алгоритм отбора, в отличие от алгоритмов пчелиного роя и генетического, более сложен в модификации и масштабировании. Поскольку для работы этого алгоритма требуется описывать взаимодействие изменяющихся параметров (на этапе отбора, при вычислении «аффинности к антигену», т.е. при вычислении степени влияния изменения параметров на точность работы сгенерированной модели) [15]. Клональный алгоритм является усложнённой модификацией генетического, но для рассматриваемой задачи это усложнение приносит больше проблем при реализации, чем пользы при работе. А при попытке решения возникающих проблем, клональный алгоритм будет вырождаться в обычный генетический.

Подводя итог, за основу алгоритма комплексной автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур предложено взять классический генетический алгоритм. Поскольку, с одной стороны, алгоритмы роевого интеллекта имеют ограничения при случайном поиске, а с другой стороны, модификации самого генетического алгоритма, из-за своей специализации, проигрывают ему в универсальности. В свою очередь, универсальность в

рассматриваемой задаче является более важной характеристикой, чем точность при решение конкретных типов задач.

Стоит отметить и то, что в ближайшем аналоге (системе FEDOT) для решения задачи автоматизированного синтеза моделей ИНС ПР также применяется модификация эволюционного алгоритма под названием GPComp [101], что демонстрирует эффективность данного класса алгоритмов при решение подобной задачи. Тем не менее, в данной диссертации модификация эволюционного алгоритма GPComp не была использована по той причине, что имела меньшую универсальность, по сравнению с требуемой.

Со стороны программной части требование к единообразному подходу можно соблюсти лишь в случае модульной структуры разрабатываемого ПО. Поскольку, принципы генерации и обучения моделей различных архитектур ИНС ПР могут принципиально отличаться, требуется инкапсулировать работу с ними в отдельные модули, через которые механизмы межмодульного взаимодействия будут осуществлять всю работу с данными архитектурами. Подобный модульный подход требуется соблюсти и при реализации алгоритмов оценивания точности моделей различных архитектур ИНС ПР, по аналогичным причинам.

Модульная структура, с другой стороны, также решит и проблему расширяемости и масштабируемости ПО. Что позволит, в теории, расширять ПО новыми архитектурами ИНС ПР, которые будут появляться в дальнейшем. Инкапсуляция всей логики работы с новыми архитектурами в модули позволит избежать переписывания уже имеющейся логики работы ПО, что ускорит и упростит в дальнейшем ввод в эксплуатацию новых архитектур ИНС ПР.

Задача создания интегрируемых программных оболочек для сгенерированных моделей ИНС ПР может быть решена за счёт использования логики СОА в создаваемых оболочках. При таком подходе, результатом работы системы будет не просто построенная архитектура, а сгенерированный исполняемый файл, с дополнительными REST и SOAP обёртками, которые позволят без предварительной установки запустить созданную модель ИНС как сервис и обращаться к нему из других систем и ПО. При этом, подобный подход не

исключает и использования созданных моделей в режиме независимых приложений или в качестве программных библиотек. Такой подход является наиболее гибким и позволит с помощью средств встроенных в сгенерированные программные решения удовлетворить потребности большинства пользователей.

1.6 Выводы по разделу 1

1. Проведён анализ современного состояния исследований проблем практического применения моделей ИНС, который показал, что потребность в программных решениях на базе МО в целом и ГО в частности растёт быстрее количества специалистов, способных решать соответствующие прикладные задачи, используя методы и технологии МО. Всё это приводит к необходимости комплексной автоматизации процессов генерации, обучения и использования моделей МО и ГО, а также создания инструментов, упрощающих последующую интеграцию созданных решений в стороннее ПО.

2. Осуществлён обзор проблем, который показал, что на данный момент в области автоматизации процессов генерации, обучения и использования моделей ИНС проблемы имеют разнородный характер. При этом есть проблемы как методологического, так и прикладного характера. Таким образом, лишь комплексное решение всех проблем, в рамках одного ПО, может обеспечить полноценную автоматизации процесса генерации, обучения и использования моделей ИНС.

Существующие же подходы (методологии и методики) и соответствующие инструментальные средства хоть и позволяют решать точечно отдельные из перечисленных проблем, однако они не обеспечивают реализацию комплексного подхода. Их использование все ещё требует наличия у специалистов знаний о специфике генерации и обучения моделей МО и ГО, существует привязка к конкретным платформам или же отсутствует возможность масштабирования, из-за специфичной реализации. Так что, несмотря на имеющиеся достижения в решении отдельных задач, нельзя сказать, что на сегодняшний день проблема автоматизации

процессов генерации, обучения и использования моделей ИНС решена в полной мере.

3. В диссертации предложена формальная постановка задачи повышения степени автоматизации процессов создания, обучения и использования моделей ИНС прямого распространения различных архитектур, а также вспомогательные задачи комплексной автоматизации процесса синтеза моделей ИНС ПР различных архитектур и упрощения интеграции синтезированных моделей в стороннее ПО.

Перечислены и обоснованы требования к алгоритму автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур. Сформулировано требование к модульности создаваемого ПО, что позволило решить проблему унифицированного подхода к различным архитектурам ИНС, с одной стороны, и проблему масштабируемости, с другой стороны. Обосновано применение парадигмы СОА при генерации программных оболочек для созданных моделей, что позволило на конструктивном уровне решить вопрос их интеграции в стороннее ПО.

2 МОДЕЛИ, АЛГОРИТМ УНИФИЦИРОВАННОГО ПОДБОРА ГИПЕРПАРАМЕТРОВ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ИСПОЛЬЗУЕМЫЕ ПРИ КОМПЛЕКСНОЙ АВТОМАТИЗАЦИИ СИНТЕЗА ИНС ПРЯМОГО РАСПРОСТРАНЕНИЯ

Поскольку на сегодняшний день имеется уже не один десяток различных архитектур ИНС ПР, охватить их все в рамках данной работы невозможно. Поэтому, была поставлена цель отобрать наиболее распространённые и универсальные архитектуры. Именно такой подход позволит проверить разрабатываемое ПО на наибольшем числе принципиально различных прикладных задач. На основе успешных апробаций такого рода, можно будет продемонстрировать практическую значимость и применимость предложенного подхода. Поскольку, разработка архитектуры ПО базируется на модульной структуре, то она в последствии может быть расширена новыми архитектурами ИНС ПР. Поэтому, в рамках данной главы будут, прежде всего, рассмотрены и проанализированы различные архитектуры ИНС ПР и обоснована целесообразность их использования в разрабатываемом ПО.

После этого будет представлен разработанный алгоритм унифицированного подбора гиперпараметров, который решает задачу комплексного синтеза моделей ИНС ПР различных архитектур. Отличительной особенностью данного алгоритма является то, что он учитывает обобщённые особенности подхода обучения с учителем к моделям ИНС ПР, что позволяет, за счёт унификации, добиться единообразного синтеза моделей представленных архитектур ИНС ПР.

2.1 Многослойный перцептрон Румельхарта (МПР)

Несмотря на то, что алгоритм МПР был описан уже несколько десятков лет назад [118], он до сих пор находит применение при решении многих задач. Основными достоинствами данной архитектуры ИНС являются её простота и универсальность. Схема МПР представлена на рисунке 2.1.

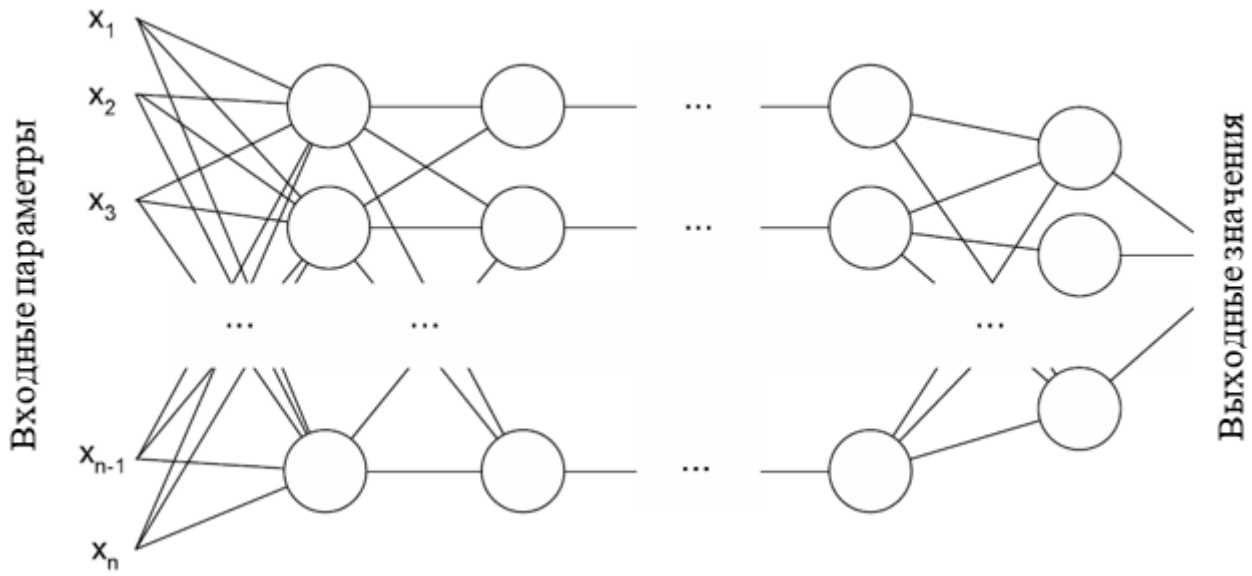

Рис. 2.1 – Схема многослойного перцептрона Румельхарта

МПР, который также называют «многослойным перцептроном», появился в середине 80-х, его авторами являются Дэвид Румельхарт, Джофри Хинтон и Рональд Уильямс. Именно начиная с данной архитектуры ИНС можно уверенно говорить о появлении ГО. От более ранних алгоритмов (к примеру, перцептрона Розенблатта) МПР отличался прежде всего алгоритмом обучения и функциями активации в нейронах. МПР был не просто комбинацией перцептронов, т.е. он не был тождественен связке перцептронов Розенблатта. МПР стал тем алгоритмом, который преодолел ключевое ограничение более ранних ИНС – невозможность решать нелинейные задачи.

Использование в нейронах не пороговых функций активации, а нелинейных (в частности, сигмоидальной) позволило с их помощью решать нелинейные задачи. Поскольку комбинация данных нелинейных функций была также нелинейная функция, то из последовательных комбинаций простых сигмоидальных функций стало возможно строить сложные функции аппроксимации, что подтверждается теоремой Колмогорова [13]. Для обучения моделей МПР авторам пришлось использовать новый алгоритм обучения – алгоритм обратного распространения ошибки [117].

Суть данного алгоритма сводится к тому, что модель ИНС модифицируется исходя из ошибки своей работы, рассчитанной по формуле (2.1):

$$E = 0{,}5 * \sum_{k=1}^{M} (y_k - y'_k)^2, \quad (2.1)$$

где M – количество выходных аппроксимируемых параметров,

y_k – эталонные значения для заданных входных данных,

y'_k –значение выходной переменной ИНС для заданных входных данных.

Вычисленная таким образом ошибка применяется для модификации всех весов связей в ИНС по формуле (2.2):

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}, \quad (2.2)$$

где w_{ij} – вес связи между i-ым и j-ым нейронами ИНС,

η – скорость обучения ИНС.

Данная формула позволяет МПР успешно и единообразно аппроксимировать разнородные функции и, таким образом, решать принципиально разные сложно формализованные задачи.

Также, побочным эффектом многослойной конфигурации ИНС стала возможность формирования сложных внутренних представлений, т.е. возможность создания многомерных аппроксимирующих поверхностей. Персептрон Розенблатта и другие более ранние алгоритмы не обладали такой способностью, поскольку являлись линейными. Именно с появлением этой особенности стало возможно говорить о решении с помощью ИНС таких задач, как интеллектуальный анализ естественного языка, обработка и анализ изображений, прогнозирование поведения сложно формализованных объектов и процессов. МПР позволял уже в первом приближении решать те же задачи, для решения которых используется биологический мозг, поскольку имел схожую логику функционирования.

При этом, алгоритм обратного распространения ошибки позволял полностью автоматизировать сам процесс обучения отдельно взятой модели ИНС. В МПР с нелинейными функциями активации в нейронах, обученными с помощью алгоритма обратного распространения, вся модификация обучаемых параметров сети производится в полностью автоматическом режиме. Поскольку именно от

этих параметров зависит вид аппроксимирующей функции, можно говорить о том, что МПР в автоматическом режиме производит аппроксимацию целевой функции. Именно применение алгоритма обратного распространения в комбинации с нелинейными нейронами стало крупным прорывом в области МО и, по сути, породило ГО.

Несмотря на то, что МПР был описан уже давно, он до сих пор активно применяется во многих областях [39, 92, 99]. Это связано, во многом, с двумя ключевыми факторами.

Во-первых, МПР крайне прост в реализации и интерпретации, т.е. в последующем анализе сгенерированной аппроксимирующей функции. Это позволяет использовать его как в прототипировании, когда информации об объекте слишком мало, для обоснованного применения более сложных архитектур, так и для анализа целевых объектов и процессов, посредством анализа аппроксимирующих функций.

Во-вторых, МПР одна из самых универсальных архитектур ИНС ПР. Это позволяет использовать его в принципиально различных областях человеческой деятельности, без необходимости внесения модификаций, а также в качестве составной части более сложных алгоритмов (к примеру, в качестве «классификационной вершины» (classification head) в свёрточной нейронной сети).

Именно эти два фактора и стали причиной выбора МПР в качестве одной из архитектур, реализуемых в рамках ПО комплексной автоматизации процессов генерации, обучения и использования ИНС ПР различных архитектур. Поскольку перед ПО не ставилась задача о применении в какой-то конкретной области человеческой деятельности, возникло требование наличия универсальной архитектуры. МПР полностью удовлетворял этому требованию, позволяя применять ПО для решения разнородных задач.

При генерации и обучении данной архитектуры производятся следующие типовые шаги:

- задаётся входной слой, число нейронов в котором равно числу входных переменных;

- задаётся выходной слой, число нейронов в котором равно числу прогнозируемых переменных;
- выбирается количество скрытых слоёв;
- выбирается количество нейронов в скрытых слоях;
- для нейронов в скрытых слоях задаются функции активации (выбирается конкретная функция из списка возможных функций активации, одинаковая для всех нейронов всех скрытых слоёв);
- выбирается алгоритм оптимизации нейронной сети (из списка возможных алгоритмов);
- выбирается количество эпох обучения;
- производится обучение заданное количество эпох алгоритмом обратного распространения ошибки.

При автоматизации обучения моделей данной архитектуры ИНС ПР предполагается варьировать следующие гиперпараметры: количество скрытых слоёв, количество нейронов в скрытых слоях, количество эпох обучения, функции активации скрытых слоёв, алгоритм оптимизации нейронной сети, скорость обучения, количество шагов в эпохе обучения. Варьирование более специфических гиперпараметров сети (к примеру коэффициент функции активации Leaky ReLU) не предполагается, поскольку это является избыточным для поставленной задачи.

На текущем этапе разработки выбор наиболее точной модели данной архитектуры будет осуществляться на основание значения среднеквадратичной ошибки. Но в дальнейшем предполагается более широкий выбор методов оценки точности модели, в т.ч. пользовательские.

2.2 Свёрточная нейронная сеть (СНС)

СНС является более узкоспециализированным инструментом, подходящим прежде всего для работы с изображениями и иными данными, которые возможно представить в матричном виде. Одной из первых и наиболее известной является модель, разработанная Яном Лекуном [86]. В основу логики функционирования

данного класса моделей были положены исследования зрительной коры головного мозга животных [73] и созданные на их основе алгоритмы распознавания [62]. Данная архитектура предназначена для автоматического и адаптивного изучения многомерных зависимостей в массиве входных данных. Причём СНС учитывает как низкоуровневые зависимости, так и высокоуровневые. На рисунке 2.2 представлена схема СНС Яна Лекуна.

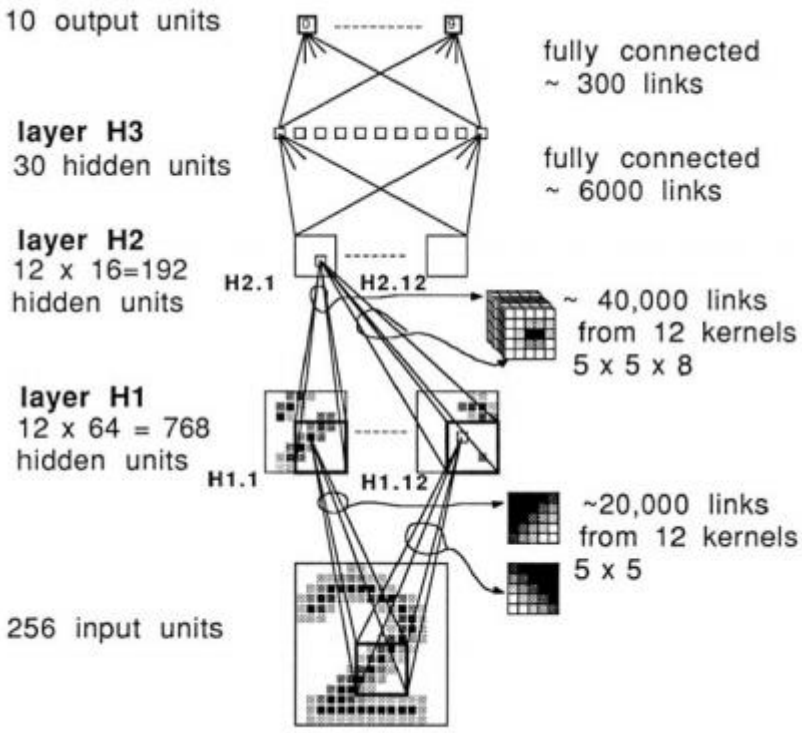

Рис. 2.2 – Схема СНС, разработанной Яном Лекуном (заимствовано из [85])

СНС обычно состоит из трех типов слоев: свёрточных слоёв, объединяющих слоёв и полносвязных слоёв. Первые два типа слоёв решают задачу извлечения признаков, а третий тип занимается интерпретацией промежуточных данных и расчётом выходных значений сети.

В то время, как объединяющие и полносвязные слои характерны и для других архитектур ИНС, свёрточные слои являются ключевой особенностью СНС. Поскольку в изображениях вся информация хранится в виде двумерных матриц (т.е. в виде пикселей), при работе с ними требуется учитывать не только значения, поступающие от самих нейронов, но и связанные с ними значения от группы ближайших нейронов. Для этого в свёрточных слоях, помимо нейронов, имеется

ещё один тип элементов, которые применяют определённые линейные операции ко всем входным данным каждого нейрона слоя – ядро свёртки. Ядро свёртки представляет из себя сетку, которая «скользит» по входной двумерной матрице (или по свёрточному слою предыдущего уровня) и ищет шаблоны и паттерны в данных. Если будет найдена такая часть матрицы, которая совпадёт с шаблоном ядра, то оно передаст расчётному нейрону текущего слоя большое положительное значение. Если совпадения не будет, то ядро передаст небольшое значение или нуль.

Из-за того, что ядро свёртки может применяться к каждой позиции матрицы данных, наличие свёрточного слоя позволяет СНС с высокой точностью решать задачи обработки изображений, поскольку особенности или закономерности в изображениях могут проявляться в любом участке. Т.е. в отличие от прочих архитектур ИНС СНС способна анализировать контекстно-зависимые данные.

При этом, свёрточные слои в СНС могут встречаться несколько раз. Тогда каждый последующий слой обрабатывает данные от предыдущего свёрточного слоя, также, как и от входного, поскольку свёрточный слой, в свою очередь, также является двумерным массивом. Таким образом, каждый последующий свёрточный слой может находить всё более сложные шаблоны и паттерны изначального входного массива.

Подобная организация свёрточного слоя приводит к тому, что в отличие от обычных полносвязных слоёв, которые обучаются за счёт изменения весов связей, в свёрточных слоях «обучаются» ядра свёртки. Поэтому, применять алгоритм обратного распространения ошибки по аналогии с перцептроном Румельхарта в данном случае не получится. Алгоритм должен быть соответствующим образом модифицирован. Это приводит к тому, что программная реализация алгоритма обучения, в случае СНС, будет отличаться.

Исходя из описанных достоинств, СНС была выбрана для использования в разрабатываемой системе. СНС предлагается использовать при решение задач анализа изображений, как наиболее универсальную архитектуру ИНС, способную с высокой точностью решать задачи данного класса. С другой стороны, поскольку

процесс обучения моделей СНС существенно отличается от процесса обучения перцептронов Румельхарта, на примере данной архитектуры можно будет проверить возможность единообразного обучения моделей различных архитектур ИНС ПР.

При генерации и обучение моделей данной архитектуры производятся следующие типовые шаги:

- задаётся входной свёрточных слой, размерность которого равна разрешению входного изображения;
- задаётся выходной слой, число нейронов в котором равно числу прогнозируемых классов;
- выбирается количество скрытых свёрточных слоёв;
- выбирается размерность скрытых свёрточных слоёв;
- выбирается количество скрытых полносвязных слоёв;
- выбирается количество нейронов в скрытых слоях;
- для нейронов в скрытых слоях задаются функции активации (выбираются две конкретные функции из списка возможных функций активации, одна для всех нейронов всех скрытых свёрточных слоёв, вторая для всех нейронов всех скрытых полносвязных слоёв);
- выбирается алгоритм оптимизации нейронной сети (из списка возможных алгоритмов);
- опционально, для свёрточных и/или скрытых слоёв включается функция дропаута;
- выбирается количество эпох обучения;
- производится обучение заданное количество эпох алгоритмом обратного распространения ошибки для свёрточной нейронной сети.

Для автоматизации обучения моделей данной архитектуры предполагается варьировать: количество свёрточных слоёв, количество полносвязных слоёв, количество нейронов в свёрточных слоях, количество нейронов в полносвязных

слоях, включение/выключение дропаута, количество эпох обучения, скорость обучения.

На текущем этапе разработки выбор наиболее точной модели данной архитектуры будет осуществляться на основание значения метрики F-меры. Но в дальнейшем предполагается более широкий набор методов оценки точности модели, в т.ч. пользовательские.

2.3 СНС сегментации и семантического анализа изображения

Для решения задачи сегментации и семантического анализа изображения в ПО было предложено использовать архитектуру Mask Regions with Convolution Neural Networks (MRCNN) [68]. MRCNN наиболее узкоспециализированная из имеющихся в системе архитектур ИНС прямого распространения. Данная архитектура является подвидом классической СНС. Однако, за счёт усложнения архитектуры, она более успешно справляется с задачами семантической и объектной сегментации изображений [63, 146]. Обобщённая схема MRCNN представлена на рисунке 2.3.

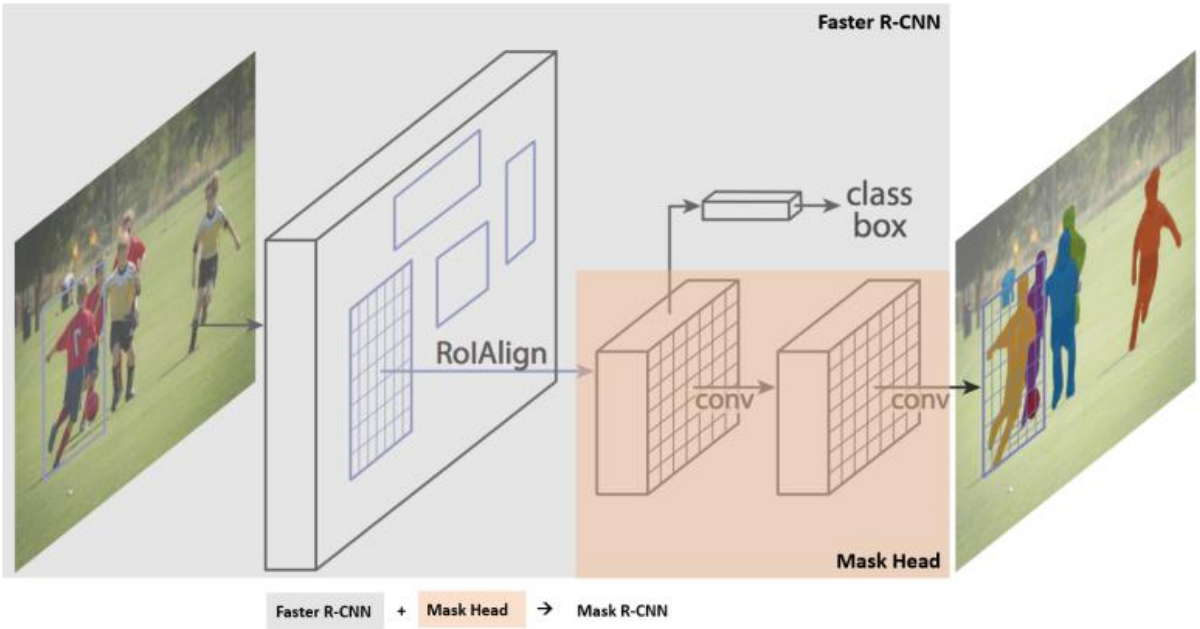

Рис. 2.3 – Схема MRCNN (заимствовано из [68])

Ключевая особенность данной архитектуры заключается в совмещении архитектуры СНС под названием Faster Regions with Convolution Neural Networks

(FRCNN), которая отвечает за решение задачи классификации, и модуля Mask Head, который отвечает за решение задачи сегментации изображения. Результатом работы MRCNN является совмещённый ответ двух данных частей ИНС.

FRCNN это СНС, которая осуществляет поиск на изображении объектов, после чего дополнительно производит классификацию найденного объекта. Результатом работы данной СНС являются ограничивающие прямоугольники для каждого объекта (т.е. прямоугольная граница, которая ограничивается найденный объект) и метка класса найденного объекта с оценкой достоверности.

Первый этап работы FRCNN состоит из одновременного запуска двух включённых ИНС: магистральной (ResNet, VGG, Inception или аналогичных) и сети регионального позиционирования. Эти сети обрабатывают каждое поступающее на вход FRCNN изображение и на выходе предоставляют трёхмерный массив – массив предложенных областей. В данном массиве содержатся координаты регионов на входном изображении, которые содержат какой-либо объект.

На втором этапе работы FRCNN прогнозирует координаты ограничивающих прямоугольников и классы объектов для каждой из предложенных областей, полученных на первом этапе. Каждая предлагаемая область может иметь разный размер, но так как свёрточные слои в СНС всегда требуют вектора фиксированного размера для прогнозирования, на данном этапе также производится масштабирования найденных регионов. Размер регионов масштабируется с помощью либо алгоритма RoI, либо метода RoIAlign.

MRCNN в свою очередь является расширенной версией FRCNN, дополненной ветвью для прогнозирования масок сегментации для каждой области интереса. На втором этапе работы MRCNN уже используется только RoIAlign, который помогает сохранить оригинальные пространственные координаты, которые смещаются в случае использования RoI. Это необходимо затем, чтобы выходные данные RoIAlign можно было совместить с данными, полученными на первом этапе и, с помощью модуля Mask Head (который в свою очередь также реализован на базе свёрточных слоёв), сгенерировать маску для каждого ответа RoIAlign. Такие маски представляют собой двумерную матрицу, с помощью

которой для каждого пикселя, входящего в границы региона объекта, определяется вероятность принадлежности этого пикселя искомому объекту.

Подобный подход позволяет более точно определить границы искомого объекта. В идеальных случаях, MRCNN может точно вычислить все пиксели изображения, на которых находится искомый объект. С другой стороны, в процессе обучения, данные, получаемые в модуле Mask Head, используются для дополнительного обучения FRCNN. Это позволяет повысить точность классификации и определения границ объектов.

Архитектура MRCNN была включена в разрабатываемое ПО по двум причинам. Во-первых, она является одной из наиболее точных на сегодняшний день архитектур, решающих задачи сегментации. Задачи сегментации, в свою очередь, появляются всё чаще, поскольку увеличивается роль контроля за различными процессами человеческой деятельности посредством видеонаблюдения. Во-вторых, на примере данной архитектуры апробируется внедрение в разрабатываемую систему сложно организованных архитектур, требующих специализированного подхода к процессу обучения.

Процессы генерации и обучения моделей данной архитектуры СНС концептуально не отличаются от процессов генерации и обучения, описанных в разделе 2.2. Однако, поскольку данная архитектура СНС является уже фиксированной и оптимизированной, при автоматизированном обучении не предполагается модификация самой архитектуры. Поэтому для варьирования остаются только следующие гиперпараметры – количество эпох обучения, количество шагов в эпохе, скорость обучения.

Тем не менее, из-за сложности обучения данной архитектуры, она имеет уникальную для неё опцию – возможность предобучения. Такой подход распространён при обучение масштабных моделей ИНС и носит название «трансферное обучение» [46]. В случае выбора данной опции обучение сети в разработанном ПО будет производиться не с нуля. Вместо этого за основу будет взята модель MRCNN уже обученная на массиве данных Microsoft COCO Dataset (Microsoft Common Objects in Context) [53]. Данный массив является на

сегодняшний день самым крупномасштабным набором данных, используемым для обучения моделей МО решению задач обнаружения и сегментации. Данный набор данных состоит из 328 тысяч изображений. Все данные изображения уже размечены и сформированы в обучающие выборки. Поэтому использование данного массива для базового обучения MRCNN позволяет задать для неё все основные концепции различных классов объектов.

После базового обучения на основание MS COCO Dataset полученную модель можно будет либо уже использовать, если объекты, которые требуется распознавать, входили в датасет, либо же дообучить для решения конкретной целевой задачи (т.е. для распознавания и сегментации объектов, которые не входили в датасет). В случае дообучения потребуется намного меньшая размерность выборки, чем в случае, если бы сеть пришлось обучать с нуля (тем не менее размер этой выборки индивидуален для каждой задачи и высчитать среднее значение не представляется возможным).

Однако, данная опция подходит далеко не для всех задач. Специфические задачи, к примеру распознавание контуров лесных массивов на спутниковых снимках или клеток на снимках микроскопов, потребуют обучения MRCNN с нуля.

На текущем этапе разработки, из-за специфики данной архитектуры СНС, выбор наиболее точной модели будет осуществляться на основании пользовательского алгоритма оценки точности модели. Поскольку универсальные методы оценки для данной модели отсутствуют.

2.4 Алгоритм унифицированного подбора гиперпараметров для комплексной автоматизации синтеза моделей ИНС ПР различных архитектур

После выбора архитектур ИНС ПР, которые требуется включить в разрабатываемый ПО, закономерным шагом является переход к задаче выбора и интеграции в ПО алгоритма комплексной автоматизации процесса синтеза моделей ИНС ПР различных архитектур. Требования к данному алгоритму формируются исходя как из цели самого разрабатываемого ПО, так и с учётом особенностей

архитектур ИНС ПР, которые будут генерироваться и обучаться данным алгоритмом.

Также, в процессе интеграции данного алгоритма стоит учитывать и принципы модульности, которым должно удовлетворять разрабатываемое ПО при решения поставленных перед ним задач.

В данной главе рассматривается разработанный алгоритм унифицированного подбора гиперпараметров (УПГ), который был выбран для комплексной автоматизации процессов генерации и обучения ИНС ПР, различных архитектур. Обосновываются причины его выбора, а также описываются реализованные особенности, которые были внесены. Также, обосновывается применение СОА при разработке ПО для удовлетворения требований к модульности и для упрощения последующей интеграции в стороннее ПО сгенерированных и обученных моделей ИНС.

На данный момент на практике в разных областях науки и техники часто используется подход, заключающийся в создании индивидуальных мультизадачных ИНС, способных решать целый класс задач [58, 93, 109, 139]. Данный подход имеет ряд преимуществ, в частности более высокую точность для расчёта выбранных целевых показателей. Однако разработка каждой такой модели ИНС является крайне ресурсоёмкой и требует участия профильных специалистов, способных спроектировать уникальные архитектуры таких ИНС. Альтернативным решением, предлагаемым в данной работе, является комплексная автоматизация процессов генерации и обучения моделей ИНС ПР различных архитектур. Данное решение подразумевает одновременное обучение нескольких моделей ИНС ПР, на заранее подготовленной выборке данных, для последующего ситуационного выбора наиболее точной модели. Такой подход приводит к тому, что для решения различных задач из разных областей используется одни и те же архитектуры ИНС, что уменьшает время необходимое на разработку ПО на базе ИНС. Однако, также данный подход приводит и к необходимости решения задачи оценивания качества параметрической адаптации моделей. При этом, задача формирования обучающей выборки, в общем случае, не требует профильных знаний [120].

Для синтеза моделей ИНС ПР различных архитектур в данной работе предлагается использовать алгоритм УПГ. Данный алгоритм является эвристическим алгоритмом поиска и модификацией ГА, но нацелен прежде всего на унифицированный подход к подбору гиперпараметров различных архитектур ИНС ПР. ГА является подвидом класса эволюционных алгоритмов и его отличительной особенностью является использование оператора «скрещивания». В разработанном алгоритме УПГ данный оператор комбинирует новые решения (модели), подбирая значения гиперпараметров из нескольких уже существующих решений (моделей). Причём гиперпараметры выбираются случайным образом.

В терминологии ГА – «фенотип» представляет собой конкретные гиперпараметры ИНС, а «генотип» представлен в виде в виде расширяемого массива, элементы которого могут быть как целочисленными, так и вещественными. Эти элементы соотнесены с соответствующими гиперпараметрами «фенотипа». Выбор описанного формата «генотипа» обусловлен тем, что, во-первых, гиперпараметры ИНС в свою очередь могут быть как целочисленными, так и вещественными. Во-вторых, из-за заявленных требований к универсальности и простоте расширения предполагается, что в дальнейшем в разработанное ПО будут добавляться новые архитектуры ИНС ПР. В этом случае требуется избежать модификации созданного алгоритма, чего сложно было бы добиться (в программной реализации) при ином формате «генотипа». Кроссовер (оператор скрещивания) является случайным и многоточечным (при каждом скрещивании ген разбивается на количество частей равное количеству элементов массива). Подобная модификация ГА является специфичной для поставленной задачи, поскольку заранее не известно, какие гиперпараметры и каким образом нужно изменять, чтобы повысить точность модели ИНС конкретной архитектуры. Гиперпараметры большинства архитектур ИНС не имеют линейной зависимости друг от друга, поэтому задачи генерации и обучения моделей ИНС часто сводятся как раз к подбору комбинаций этих гиперпараметров. Поэтому генерируя, с помощью алгоритма УПГ, новые модели ИНС для которых используются значения гиперпараметров наиболее точных из

уже имеющихся в обучаемой выборке моделей ИНС, можно действительно добиться ещё более высокой точности решения задачи.

Использование алгоритма УПГ в разрабатываемом ПО подразумевает одновременное обучение нескольких моделей ИНС, на заранее подготовленной выборке данных, для последующего ситуационного выбора наиболее точной модели. Таким образом, именно в этом ситуационном выборе и проявляется задача оценивания качества параметрической адаптации моделей.

Алгоритм отбора моделей ИНС был реализован следующим образом:

Шаг 1. В первой родительской популяции генерируется фиксированное число (M) моделей ИНС фиксированной архитектуры (выбранной автоматически на основе представленных пользователем входных данных), со случайно заданными гиперпараметрами.

Шаг 2. Генерируется N_d дочерних моделей ИНС фиксированной архитектуры, гиперпараметры которых выбираются равновероятным случайным образом из гиперпараметров двух случайно подобранных родительских моделей ИНС (для двух родительских моделей – одна дочерняя), а также генерируется N_r моделей ИНС фиксированной архитектуры, гиперпараметры которых задаются полностью случайно, с учётом заданных диапазонов значений для этих гиперпараметров. Оператор мутации (инвертирования генов) для дочерних моделей ИНС не применяется из-за специфического формата «генотипа».

Шаг 3. Далее осуществляется селекция моделей ИНС по методу рулетки (формула 2.3).

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \tag{2.3}$$

где p_i – вероятность выбора i-ой модели ИНС,

f_i – значение функции приспособленности для i-ой модели ИНС,

N – количество моделей ИНС в популяции на текущем шаге (формула 2.4).

$$\mathrm{N} = M + N_d + N_r, \tag{2.4}$$

Метод рулетки выбран из-за его универсальности и простоты программной реализации, поскольку алгоритм подразумевается использовать для разных классов

задач. Использование специфических алгоритмов хоть и повысило бы скорость работы для одних классов задач, но неизбежно понизило бы скорость работы для других классов. В основу функции приспособленности положен расчёт погрешности значения целевого параметра, рассчитанного с помощью моделей ИНС, относительно реального значения тестовой выборки (формула 2.5).

$$f_i = \frac{1}{\sqrt{\frac{\sum_{j=1}^{X}(\varepsilon_{ij}-\omega_j)^2}{X}}}, \quad (2.5)$$

где ε_{ij} – выходное значение целевого параметра спрогнозированное i-ой моделью ИНС в ответ на j-ый входной тестовый вектор,

ω_j – реальное значение тестовой выборки в ответ на j-ый входной тестовый вектор,

X – количество тестовых векторов.

Результатом расчета по данной формуле является значение «уровня приспособленности», которое обратно пропорционально среднеквадратической ошибки i-ой модели ИНС на тестовой выборке. В результате селекции, в текущее поколение, из N моделей ИНС, отбирается M моделей, с наибольшим значением p_i (вероятности выбора i-ой модели ИНС).

Шаг 4. Для всех моделей ИНС, вычисляется (по специфической для каждой архитектуры формуле) ошибка рассчитанного ими значения целевого параметра относительно реального значения тестовой выборки. Если хотя бы одна модель ИНС показывает ошибку ниже заданного значения, цикл прерывается. Модель ИНС с наименьшей ошибкой принимается в качестве «победившей». В противоположном случае, происходит возврат к шагу 2. При этом, популяция моделей каждой итерации отдельно запоминается. Если популяция моделей текущей итерации полностью совпадает с предыдущей популяцией, это означает, что за всю итерацию не было найдено конфигурации модели ИНС с большей точностью и осуществляется безусловный переход к шагу 5.

Шаг 5. Если не найдена модель ИНС с ошибкой меньше заданного значения, цикл запускается с 1 шага с новой родительской популяцией моделей, для которой задаются новые случайные значения параметров. Если за I итераций решение не

найдено, задача признаётся нерешаемой, при заданных настройках, и осуществляется выход из алгоритма.

Ключевые отличия УПГ от исходного генетического алгоритма следующие:

- «Генотип» представлен в виде в виде расширяемого массива, элементы которого могут быть как целочисленными, так и вещественными. Это обусловлено тем, что, во-первых, гиперпараметры ИНС могут быть как целочисленными, так и вещественными, а во-вторых, в разработанное ПО могут добавляться новые архитектуры ИНС ПР и было бы сложно (в программной реализации) избежать модификации созданного алгоритма при ином формате «генотипа».
- Оператор мутации (инвертирования генов) для дочерних моделей ИНС не применяется из-за специфического формата «генотипа».
- Кроссовер (оператор скрещивания) является случайным и многоточечным (при каждом скрещивании ген разбивается на количество частей равное количеству элементов массива). Подобная модификация является специфичной для поставленной задачи, поскольку заранее не известно, какие гиперпараметры и каким образом нужно изменять, чтобы повысить точность модели ИНС конкретной архитектуры.

Обобщённая структурная схема описанного алгоритма представлена на рисунке 2.4.

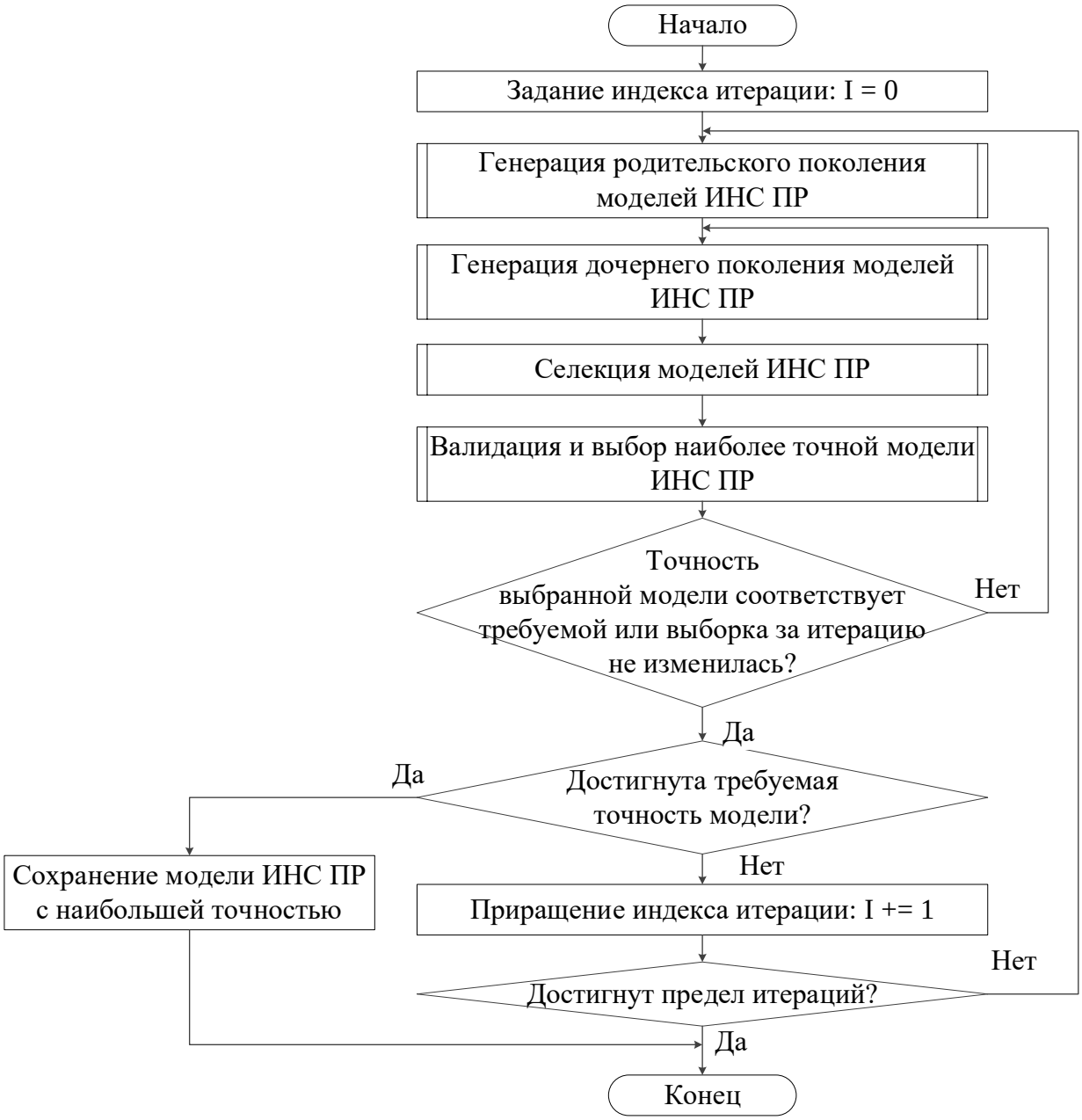

Рис. 2.4 – Структурная схема алгоритма УПГ для автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур

В процессе своей работы разработанный алгоритм УПГ будет отбирать среди ряда обучаемых моделей ИНС наиболее точные, последовательно подстраивая параметры сетей для повышения точности их работы. Количество и тип параметров индивидуальны для различных архитектур ИНС.

Применение описанного алгоритма УПГ к задаче комплексной автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур позволит решить две ключевые задачи. Во-первых, данный алгоритм позволит единообразно обучать различные модели ИНС ПР, поскольку пул

гиперпараметров не фиксирован и может быть адаптирован под разные архитектуры без необходимости модификации самого алгоритма. Так как вся логика работы с конкретными архитектурами ИНС ПР будет инкапсулирована, алгоритм в полностью автоматическом режиме сможет переключаться между различными архитектурами в процессе своей работы. Во-вторых, подход с рекомбинацией гиперпараметров позволит ускорить сам процесс синтеза моделей. Поскольку гиперпараметры новых моделей ИНС будут комбинироваться с гиперпараметрами уже существующих моделей, это позволит быстрее получать модели, обеспечивающие решение поставленной задачи с заданной точностью.

Использование разработанного алгоритма УПГ при решении задачи комплексной автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур актуально в тех случаях, когда разработка специализированной модели ИНС, обеспечивающей решение поставленной задачи с максимально возможной точностью, является нерентабельной. С помощью разработанного алгоритма УПГ возможно дешевле и быстрее создать модель ИНС, способную решать поставленную типовую задачу не с максимально возможной, но с заданной пользователем точностью.

2.5 Выводы по разделу 2

1. Предложены архитектуры ИНС ПР, использование которых оправдано в рамках разрабатываемого ПО. Для каждой архитектуры, во-первых, обоснована причина её использования в разрабатываемом ПО и указана область применения, во-вторых, описаны особенности её генерации, обучения и функционирования, в-третьих, представлены методы её интеграции в разрабатываемый ПО, с учётом описанных особенностей. Представленные в данной главе архитектуры предоставляются пользователю на выбор, при потребности в решении соответствующих задач.

2. Для процессов генерации и обучения выбранных архитектур, предложен новый подход к автоматизации следующих этапов создания моделей: подбор гиперпараметров генерации моделей ИНС ПР, генерация моделей ИНС ПР,

обучение моделей ИНС ПР. При работе с рассмотренными архитектурами от пользователя требуется предоставить размеченную обучающую выборку и указать минимальную целевую точность создаваемой модели ИНС, при достижении которой обучение считается успешным, что позволяет реализовать концепцию No-Code разработки в создаваемом ПО.

3. Разработан алгоритм УПГ для решения задачи автоматизации процесса синтеза моделей ИНС ПР различных архитектур, который предназначен для использования в разрабатываемом ПО в качестве основного алгоритма генерации и обучения моделей ИНС. Инкапсуляция логики функционирования моделей ИНС ПР различных архитектур и модульная структура ПО, предложенные в диссертационной работе, позволяют единообразно обращаться к различным архитектурам и обучать их с помощью одного программного модуля.

3 СОСТАВ, СТРУКТУРА И ТЕХНОЛОГИИ, ИСПОЛЬЗОВАННЫЕ В РАЗРАБОТАННОМ ПРОГРАММНОМ ОБЕСПЕЧЕНИИ КОМПЛЕКСНОЙ АВТОМАТИЗАЦИИ СИНТЕЗА ИНС ПРЯМОГО РАСПРОСТРАНЕНИЯ

После выбора архитектур ИНС ПР, которые требуется включить в ПО, а также решения задачи создания алгоритма комплексной автоматизации процесса синтеза моделей ИНС ПР различных архитектур, закономерным будет переход к задачам, связанным с программной реализацией выбранных алгоритмов. Данные задачи формируются исходя как из планируемого предназначения самого ПО, так и с учётом особенностей архитектур ИНС ПР, на основании которых разработанный алгоритм будет генерировать и обучать модели.

Также, при разработке ПО стоит учитывать и принципы модульности, которым оно должно удовлетворять для решения поставленных задач.

В данной главе описываются особенности разработанных программных модулей, совокупность которых позволяет реализовать парадигму No-Code разработки, а также обосновывается необходимость применения СОА при разработке ПО для удовлетворения требований к модульности и для упрощения последующей интеграции в стороннее ПО сгенерированных и обученных моделей ИНС.

3.1 Сервис-ориентированная архитектура (СОА)

СОА это парадигма разработки программного обеспечения, при которой программные модули предоставляют возможность работать с реализованными в них методами и функциями другим программным модулям посредством сервисной оболочки через протоколы связи по сети Интернет. Парадигма СОА не зависит от используемых программных библиотек и конкретных протоколов связи. В СОА несколько сервисов взаимодействуют друг с другом одним из двух способов: непосредственно передавая данные друг другу через сервисный программный интерфейс или с помощью сервисов-администраторов, которые координируют деятельность всех остальных сервисов.

СОА подразумевает модульный подход к разработке ПО. Данный подход базируется на принципах прозрачного доступа к функциям и масштабируемости, что позволяет расширять функциональность программного продукта не модификацией уже существующих модулей, а добавлением новых сервисов. Под прозрачным доступом (или прозрачностью) здесь и далее понимается такой доступ к объектам и функциям различных модулей, который осуществляется с помощью одних и тех же операций, независимо от того, являются ли они локальными или удаленными. То есть программный интерфейс для доступа к конкретному объекту или функции должен быть согласованным для этого объекта, независимо от того, где он фактически хранится в системе и как он модифицируется в процессе разработки или сопровождения программного продукта [31]. Также, СОА следует концепции Интернета 2.0, что является существенным достоинством на сегодняшний день.

Для следования заданной СОА парадигме, требуется, чтобы генерируемые программные оболочки для созданных моделей ИНС имели типовой программный или веб-интерфейс и использовали распространённые протоколы обмена данными. В разработанном ПО генерируется автономный сервис, содержащий модель ИНС ПР, созданную и обученную для решения конкретной задачи по заданным пользователем данным. Сервис состоит из самой модели ИНС ПР, а также программных оболочек, реализующих интерфейс REST и SOAP. Данный сервис кроссплатформенный и может быть без предварительной установки и настройки дополнительного программного обеспечения запущен на ряде операционных систем (что возможно благодаря кроссплатформенности языка Python, с использованием которого были созданы данные модули). Кроме того, реализация нескольких программных оболочек позволяет обращаться к модели ИНС ПР как к сервису, через наиболее распространённые веб-интерфейсы. При этом, созданную модель ИНС ПР всё ещё можно использовать как простую программную библиотеку, обращаясь к ней через соответствующие программные интерфейсы. Также, сгенерированные разработанным ПО сервисы не обязательно должны быть установлены на одном и том же компьютеры. Они могут быть распределены между

различными компьютерами или же располагаться в облачных хранилищах. Соответственно, подобные сервисы могут быть использованы в системах, поддерживающих как парадигму СОА, так и концепцию ИВ.

Само ПО внутри себя также построено по принципам СОА. Это является необходимым условием, учитывая, что коммерческое использование разработанного ПО потребует больших вычислительных ресурсов. Наиболее простым и удобным в таких условиях будет распределённый подход к архитектуре ПО. Дополнительным достоинством СОА является и то, что расширение функциональности ПО посредством добавления новых архитектур ИНС ПР происходит быстро и удобно, без модификации существующего программного кода. Обобщенная структура технологии комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур представлена на рисунке 3.1.

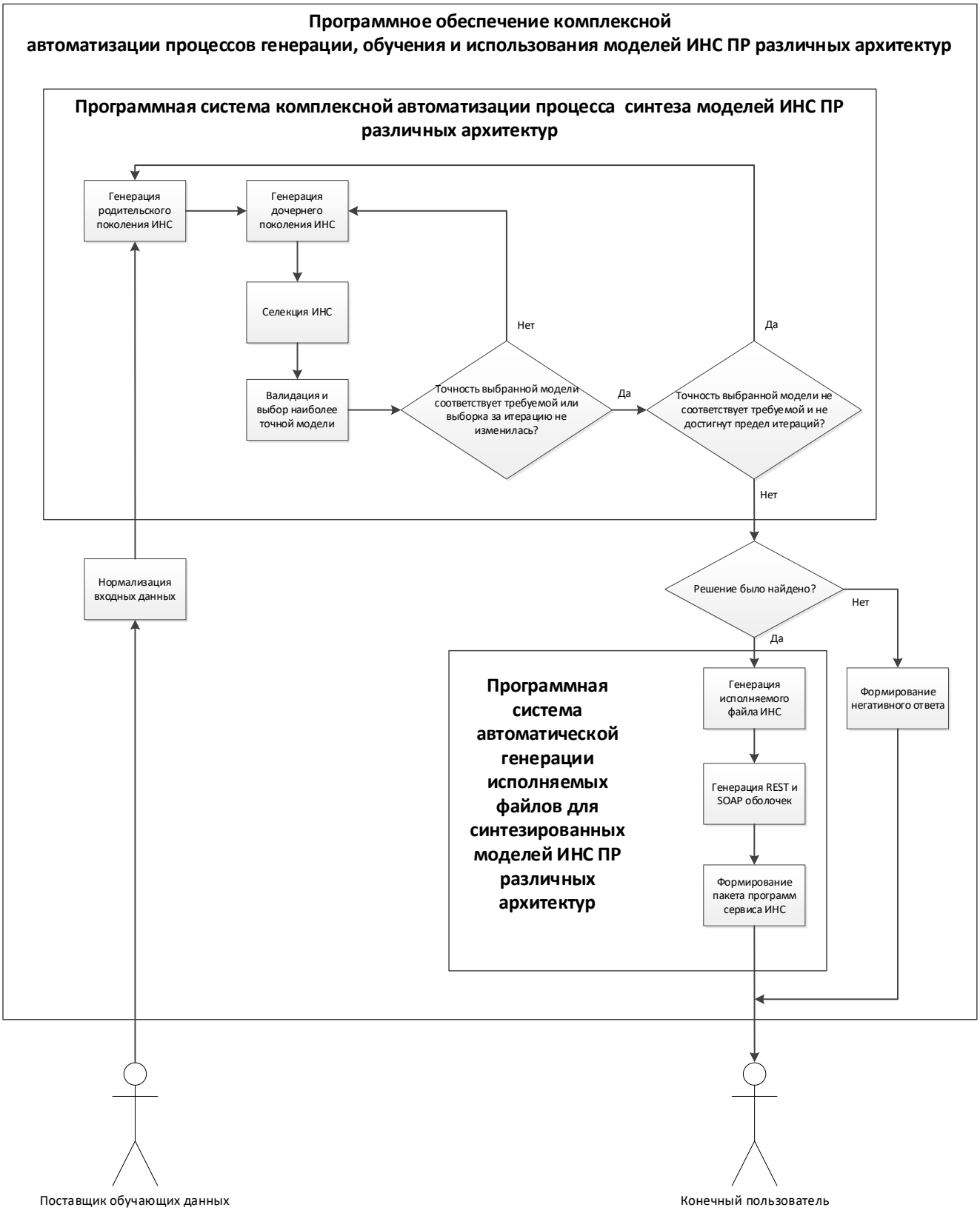

Рисунок 3.1 – Обобщенная структура технологии комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур

3.2 Состав и архитектура ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур

3.2.1 Общий состав разработанного ПО

Как было описано в предыдущей главе, разработанное ПО имеет модульную структуру. Такая структура ориентируется на максимально возможную инкапсуляцию функциональности, чтобы, впоследствии, можно было с минимальными затратами ресурсов и времени масштабировать ПО.

Помимо разбиения на модули отдельных архитектур ИНС ПР, ПО также условно делится на внешнюю и внутреннюю части. Внешняя часть отвечает за взаимодействие с пользователем и форматирование входных и выходных данных. Внешняя часть ПО подразделяется на следующие модули:

- модуль предобработки входных данных;
- модуль генерации исполняемого файла сгенерированной модели ИНС ПР;
- модуль генерации REST и SOAP оболочек для сгенерированной модели ИНС ПР;
- модуль формирования пакета программ для сгенерированной модели ИНС ПР;
- рекомендательный модуль (отвечает за предоставление рекомендаций пользователю в случае невозможности создания модели ИНС ПР для заданных входных данных);
- пользовательский интерфейс.

Внутренняя часть ПО ориентирована исключительно на генерацию и обучение моделей ИНС ПР на основе входных данных, предоставленных пользователем. Внутренняя часть ПО подразделяется на следующие модули:

- модуль инициализации (отвечает за инициализацию исходных условий решаемой задачи, в т.ч. родительского поколения моделей ИНС ПР);
- модуль генерации итеративного поколения моделей;
- модуль валидации и отбора моделей;

- модуль итоговой валидации и верификации модели;
- модули описывающие архитектуры ИНС ПР (в т.ч. все функции необходимые для взаимодействия с моделью конкретной архитектуры и варьирования её параметров).

Детальная схема разработанного ПО изображена на рисунке 3.2.

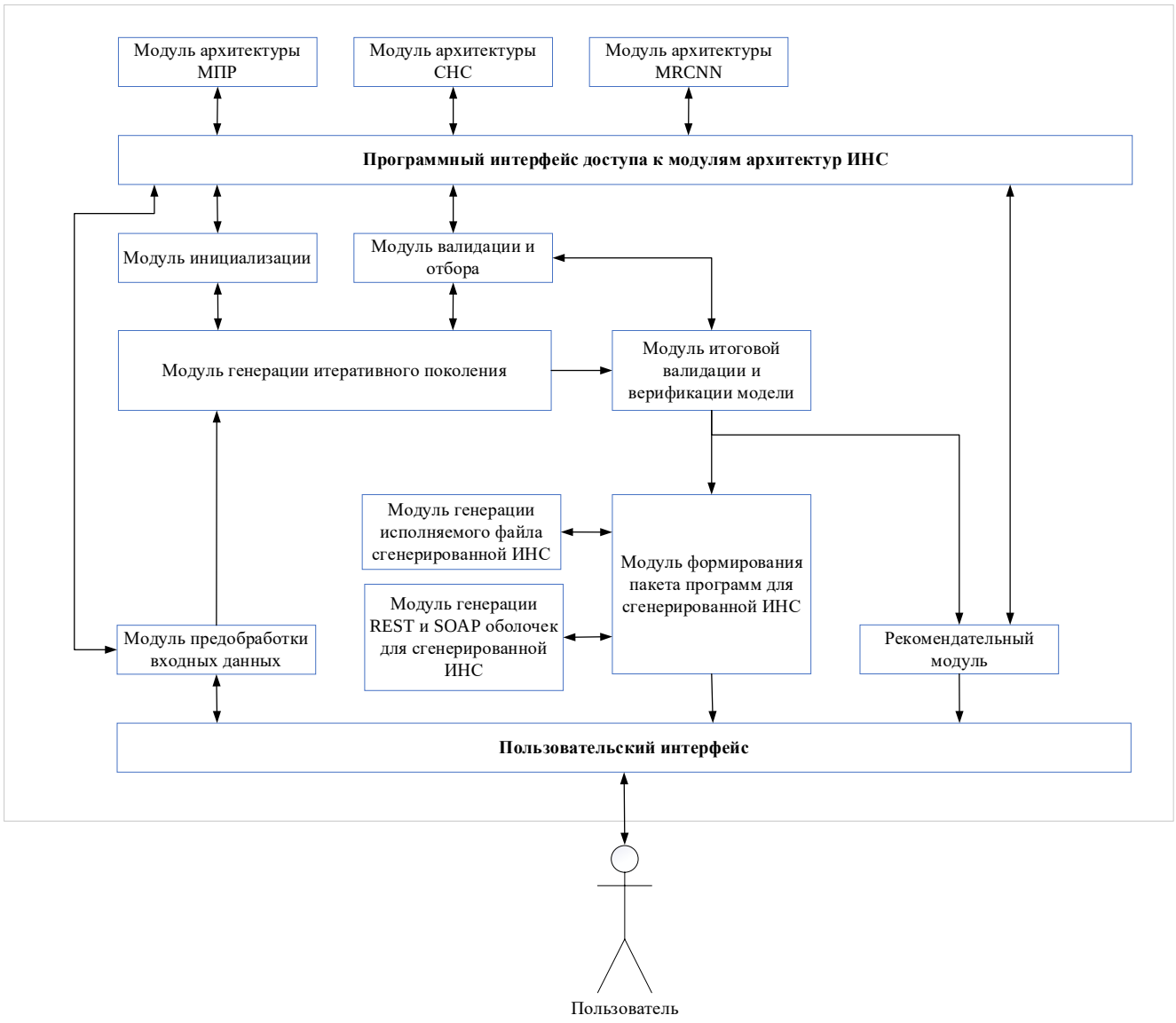

Рисунок 3.2 – Детальная схема ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур

Как видно из схемы, в разработанное ПО были введены механизмы автоматизации процессов интеграции создаваемых моделей ИНС в стороннее ПО на основе использования парадигмы СОА. Данный подход не привязан к конкретному ПО и сетевому окружению, поэтому может быть реализован с

помощью различных технологий. При этом к созданным моделям ИНС ПР можно будет обращаться единообразно из различных программных сред.

Согласно ГОСТ Р 57412-2017 «Компьютерные модели в процессах разработки, производства и эксплуатации изделий» [5] этапы разработки компьютерной модели ИНС ПР можно сформулировать следующим образом:

1) содержательная постановка задачи на концептуальном уровне;

2) построение модели ИНС ПР;

3) выбор метода решения с учётом знаний и предпочтений пользователя и разработчика;

4) программная реализация модели ИНС ПР;

5) применение полученной модели ИНС ПР при решение прикладных задач;

6) контроль и анализ полученных результатов.

Поскольку большинство существующих средств AutoML являются системами автоматизации работы специалистов по машинному обучению, они автоматизируют только пункты 2 и 4. Поэтому на практике показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур у таких систем оказывается D=0.33.

С другой стороны, в разработанном ПО автоматизируются в том числе этапы 3, 5 и 6. Поэтому для ряда прикладных задач показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур разработанного ПО может достигать D=0.83.

3.2.2 Модуль предобработки входных данных

Данный модуль полностью отвечает за всю предобработку входных данных и параметров. Список решаемых данным модулем задач, следующий:

- оценка входного массива данных на наличие ошибок в данных (несоответствия форматов и/или типов данных, несогласованность данных, пропуски и/или пустые значения и т.п.);

- оценка соответствия входного массива данных заданной архитектуре ИНС ПР и её требованиям;
- в случае прохождения проверок, данные форматируются в соответствии с требованиями конкретной архитектуры ИНС ПР.

Для осуществления всех необходимых проверок модулю необходимо знать требования, которые ставят перед входными и обучающими данными конкретные архитектуры ИНС ПР. Поэтому для корректного функционирования данный модуль должен обращаться к модулям соответствующих архитектур ИНС ПР, входящих во внутреннюю часть ПО. Данное обращение осуществляется через унифицированный программный интерфейс. При выборе пользователем на интерфейсе конкретной архитектуры ИНС ПР, в начале работы данного модуля инициализируется соответствующей этой архитектуре класс. Далее приведён пример такой инициализации:

```
# Инициализация текущей модели
curModelClass = None

if (inputSelectedModel == "DNN"):
    curModelClass = SelectedModel.DNN
elif (inputSelectedModel == "CNN"):
    curModelClass = SelectedModel.CNN
elif (inputSelectedModel == "MRCNN"):
    curModelClass = SelectedModel.MRCNN
```

После выбора архитектуры все дальнейшие проверки и обработки данных осуществляются единообразно, поскольку вся логика таких проверок инкапсулирована в функциях класса конкретной архитектуры. Возвращают эти функции либо булеву переменную (в случае функций проверки), либо типовой массив обработанных данных (в случаях функций форматирования).

3.2.3 Модуль генерации исполняемого файла сгенерированной модели ИНС ПР

Данный модуль отвечает за генерацию исполняемых файлов на основе созданной и обученной модели ИНС ПР. Эта задача вынесена в отдельный модуль

по той причине, что для генерации исполняемых файлов требуется подключение ряда сторонних библиотек, которые не требуются во всех остальных модулях.

Итогом работы этого модуля является .exe файл без пользовательского интерфейса. Принципы и технологии обмена данными для сгенерированного исполняемого файла описываются в документации, генерируемой модулем формирования пакета программ (см. далее подпараграф 3.2.4). Подобный принцип работы объясняется тем, что процесс генерации исполняемых файлов унифицирован. Подключение модулей архитектур ИНС ПР только для получения текстовой переменной избыточно, учитывая наличие модуля формирования пакета программ.

Также, в дальнейшем, планируется расширение функциональности данного модуля возможностью генерации исполняемых файлов для операционных систем Unix.

3.2.4 Модуль генерации REST и SOAP оболочек для сгенерированной модели ИНС ПР

Данный модуль отвечает за генерацию REST и SOAP скриптов на языке Python для созданной модели ИНС ПР, которые могут быть запущены автономно. Сгенерированные скрипты позволяют обращаться к программе, реализующей логику работы модели ИНС ПР через веб-интерфейсы по локальной сети или сети Интернет. Эта задача вынесена в отдельный модуль по той же причине, что и для случая использования модуля генерации исполняемых файлов – поскольку на данном этапе работы требуется подключение программных библиотек, которые нигде более не используются. Однако, также при генерации веб-интерфейсов требуется знать специфику работы конкретной архитектуры ИНС ПР и, соответственно, требуется выгружать дополнительную информацию (доступ к которой осуществляется через модуль формирования пакета программ, описанный в подпараграфе 3.2.4). Поэтому данный модуль отделён от модуля генерации исполняемых файлов.

Итогом работы данного модуля являются два сгенерированных скрипта. Спецификация данных веб-интерфейсов описана в документации, генерируемой модулем формирования пакета программ (см. подпараграф 3.2.4).

Поскольку язык программирования Python кроссплатформенный, сгенерированные на нём скрипты могут быть запущены как на операционной системе Windows, так и на Unix. Программы установки интерпретатора, который необходим для запуска скриптов, также предоставляются модулем формирования пакета программ.

3.2.5 Модуль формирования пакета программ для сгенерированной модели ИНС ПР

Данный модуль отвечает за формирования всех сопроводительных материалов и программ установки необходимых приложений и библиотек. Список файлов, формирующих выходной пакет, следующий:

- скрипт, реализующий сгенерированную и обученную модель ИНС ПР, на языке Python;
- исполняемый файл модели ИНС ПР формата .exe;
- скрипты реализующие веб-интерфейсы REST и SOAP для созданной модели ИНС ПР, написанные на языке Python;
- набор требуемых программ установки интерпретатора Python и сопутствующих библиотек, необходимых для работы модели ИНС ПР (в соответствии с конкретной сгенерированной архитектурой);
- инструкции для разворачивания и запуска модели ИНС ПР (отдельно для Windows и Unix систем), а также спецификации программных и веб-интерфейсов (в соответствии с конкретной сгенерированной архитектурой).

Поскольку, разные архитектуры ИНС ПР для своей работы требуют разные наборы программных библиотек, данный модуль должен получать информацию о том, какая архитектур ИНС выбрана. Однако, все программы установки стороннего ПО хранятся в данном модуле и при формировании выходного пакета программ среди них выбирается нужная комбинация. Поэтому, в последствии, при

масштабировании разработанного ПО данный модуль возможно потребуется модифицировать, добавив в него необходимые программы установки стороннего ПО.

Результатом работы данного модуля является архив со всеми необходимыми файлами. Данный архив выкладывается на сервере в открытый доступ и пользователю предоставляется ссылка для его скачивания. Для того, чтобы не перегружать сервер, архив выкладывается временно. При этом вне зависимости от того, скачал пользователь его или нет, через заданное время он будет удалён с сервера.

3.2.6 Рекомендательный модуль

Поскольку не для всех ситуаций созданное ПО может сгенерировать решение, предусматривается также случай отрицательного ответа пользователю. Если сгенерировать модель ИНС ПР с требуемой точностью работы не вышло, ПО обращается именно к рекомендательному модулю и передаёт ему код ошибки. Ошибки могут быть как общие для всех архитектур ИНС ПР, так и специфичные для конкретной архитектуры.

В первом случае рекомендательный модуль по коду ошибки находит текст рекомендации среди хранящихся у него заготовок. Во втором случае обращается за текстом рекомендации к модулю соответствующей архитектуры ИНС ПР. В обоих случаях, полученный текст выводится пользователю.

В зависимости от ситуации, пользователю может быть рекомендовано как перезапустить обучение с другими настройками, так и сформировать новую обучающую выборку. К данному модулю напрямую обращается модуль предобработки входных данных (см. подпараграф 3.2.2), в случае если не были пройдены какие-либо проверки.

3.2.7 Пользовательский интерфейс

Для разработанного ПО был реализован исключительно веб-интерфейс. Поскольку данное ПО предполагается запускать на серверах, а не на клиентских компьютерах, наличие десктопного интерфейса не оправдано.

Веб-интерфейс был реализован на связке из языка программирования Python (с использованием библиотеки Flask) и языка JavaScript (для генерации веб-страниц).

Использование библиотеки Flask обусловлено тем, что генерация и обучение некоторых моделей ИНС ПР (в частности свёрточных) может затянуться на десятки минут, а то и на часы. В связи с этим, потребовалось создать сокеты, которые будут отслеживать и хранить сессии различных пользователей. В этом случае при окончании создания конкретной модели ИНС ПР, сгенерированный на её основе пакет файлов будет предоставлен именно тому пользователю, который инициировал создание этой модели.

Тем не менее, стоит отметить, что на данный момент интерфейс реализован на уровне рабочего прототипа. Он не прорабатывался глубоко и не учитывает многие непредвиденные ситуации, которые могут возникнуть в процессе работы. Также, на данный момент в ПО отсутствует возможность заводить профили пользователей. Поэтому, если конкретный пользователь закроет окно браузера с запущенной программой, ссылка на сгенерированный пакет программ ему предоставлена не будет.

Кроме того, на данный момент разработанный ПО никак не решает вопросы сетевой безопасности и защиты от DDoS-атак. Пример разработанного пользовательского интерфейса представлен на рисунке 3.3.

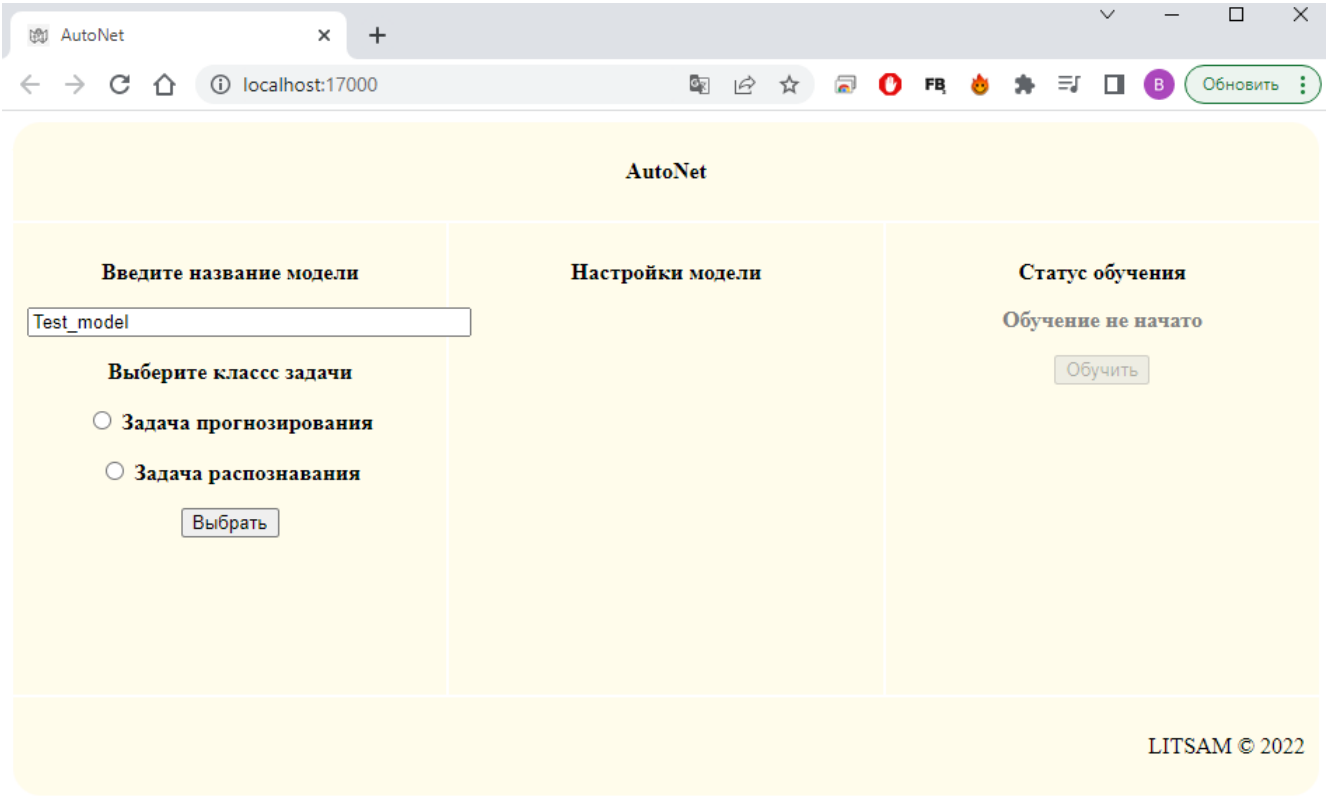

Рисунок 3.3 – Пример пользовательского интерфейса разработанного ПО

3.2.8 Модуль инициализации

Поскольку для инициализации ИНС ПР различных архитектур требуется совершить различный набор действий (для разных архитектур будет отличаться даже количество необходимых операций), данный процесс вынесен в отдельный модуль. Основная задача модуля инициализации заключается в том, чтобы вызвать модуль выбранной пользователем архитектуры ИНС ПР, задать все необходимые стартовые настройки и сгенерировать родительское поколение моделей для работы разработанного алгоритма УПГ. Причём, сгенерированный родительский массив моделей ИНС ПР должен иметь унифицированный вид, вне зависимости от выбранной архитектуры. Это необходимо как для прозрачной работы самого алгоритма УПГ, так и позволяет избежать его изменения под каждую конкретную архитектуру.

Инициализация и генерация моделей родительской выборки происходит по тому же принципу, который был описан в подпараграфе 3.2.2. В этом случае через унифицированный программный интерфейс происходит обращение к модулю

конкретной выбранной архитектуры ИНС ПР и вызов требуемых функций. Далее приведён пример реализации процесса инициализации класса-генератора и вызова его функции генерации выборки моделей ИНС ПР:

```
# Перечисление доступных классов-генераторов
SelectedGenerator = { 'DNN' : DNNGenerator(),
                      'CNN' : CNNGenerator(),
                      'MRCNN' : MRCGenerator() }

# Выбор класса-генератора
workGenerator = SelectedGenerator[inputSelectedModel]

# Инициализация параметров генерации
workGenerator.startInit()
# Генерация первого поколения
curGen = workGenerator.generation(inputGenerationNum)
```

Поскольку каждый из модулей архитектур ИНС ПР наследуется от абстрактного родительского класса, который содержит описание всего необходимого программного интерфейса, работа с классом-генератором осуществляется за счёт вызова унифицированных функций (каждая из которых переопределяется в модулях архитектур ИНС ПР).

Также, инициализированный класс-генератор передаётся в последующие модули, в которых с его помощью осуществляется генерация дочерних поколений.

3.2.9 Модуль генерации итеративного поколения

Полученный от предыдущего модуля класс-генератор задействуется в дальнейшей работе алгоритма. Есть две основные функции, которые в обязательно порядке используются на каждой итерации разработанного алгоритма УПГ – генерация моделей ИНС дочернего поколения на основе моделей ИНС родительского поколения, а также случайная генерация новых моделей ИНС.

Для решения этих двух задач отводится отдельный модуль. Он отделён от модуля инициализации, поскольку в данном случае не требуется выбор конкретной архитектуры ИНС ПР. Данный модуль осуществляет работу с классом-

генератором, в рамках реализации алгоритма УПГ, и не зависит от инкапсулированной логики функционирования конкретных архитектур ИНС ПР.

Оба вида генерации новых моделей также реализуются модулями архитектур ИНС ПР, поскольку логика генерации различных архитектур существенно отличается. Тем не менее, имеется некий общий сценарий такой генерации.

При генерации моделей ИНС дочернего поколения на основе моделей ИНС родительского поколения вызывается функция, с помощью которой на основе предоставленного массива попарно осуществляется выбор модели ИНС и создание новых, гиперпараметры которых будут взяты случайным образом у одного из двух «родителей». Поскольку типы и число гиперпараметров отличаются для различных архитектур ИНС ПР, этот выбор инкапсулируется в модулях соответствующей архитектуры.

При случайной генерации новых моделей ИНС ПР вызывается такая же функция, которая использовалась для генерации родительской выборки. Только в данном случае задаётся меньшее число генерируемых моделей.

Два полученных массива экземпляров моделей ИНС ПР объединяются в один, который становится новым родительским поколением в следующей итерации работы разработанного алгоритма УПГ.

3.2.10 Модуль валидации и отбора

С помощью данного модуля осуществляется оценивание моделей ИНС ПР, входящих в текущую родительскую выборку, и отсеивание наименее точных. Поскольку оценка точности моделей ИНС ПР различных архитектур также существенно отличается, для решения данной задачи вызываются функции модулей архитектур ИНС ПР.

На данный момент для каждой архитектуры имеется лишь один метод оценки точности модели. Тем не менее, в дальнейшем планируется вынести методы оценки точности моделей в отдельный модуль. Это обуславливается тем, что периодически в прикладных задачах требуется специфическая оценка точности, привязанная к

конкретным требованиям данной задачи. Поэтому, для решения это проблемы требуется как расширить набор методов оценки, так и добавить возможность формирования пользовательских оценок.

Однако, при разработке ПО задача реализации подобной возможности не ставились в рамках данной диссертационной работы. Поэтому, добавление этой возможности предполагается при дальнейшей разработке ПО.

3.2.11 Модуль итоговой валидации и верификации модели

Данный модуль отвечает за выбор итоговой модели, которая обеспечивает наибольшую точность решения поставленной прикладной задачи. Выбор осуществляется с помощью того же метода оценки точности моделей, что и в предыдущем подпараграфе (см. подпараграф 3.2.9). Модель, обеспечившая наибольшую точность, считается наиболее предпочтительной (валидированной) и передаётся в последующие модули для генерации всех необходимых программных оболочек.

Также, данный модуль отвечает за верификацию победившей модели. Во-первых, при работе модуля осуществляется оценка корректности аппроксимации моделью искомой функции. В частности, осуществляется проверка сгенерированной модели на предмет ее переобучения. Во-вторых, определяется достигает ли модель заданной пользователем точности.

В зависимости от результатов верификации, данный модуль вызывает либо модули, отвечающие за генерацию исполняемых файлов и документации, либо рекомендательный модуль.

3.2.12 Модули архитектур ИНС ПР

Для обеспечения прозрачности и масштабируемости ПО классы, реализующие функции, необходимые для генерации и обучения конкретных экземпляров ИНС ПР различных архитектур, было решено инкапсулировать в отдельные модули, в соответствии с выбранными архитектурами.

Работа с данными модулями в ПО была реализована с помощью шаблона проектирования «Фасад» [60]. В данном шаблоне обращение к модулям осуществляется через общий программный интерфейс, который перенаправляет вызов, в зависимости от входных параметров. В ПО перенаправление осуществляется в зависимости от выбранной пользователем архитектуры ИНС ПР. Обобщённая схема реализации «Фасада» представлена на рисунке 3.4.

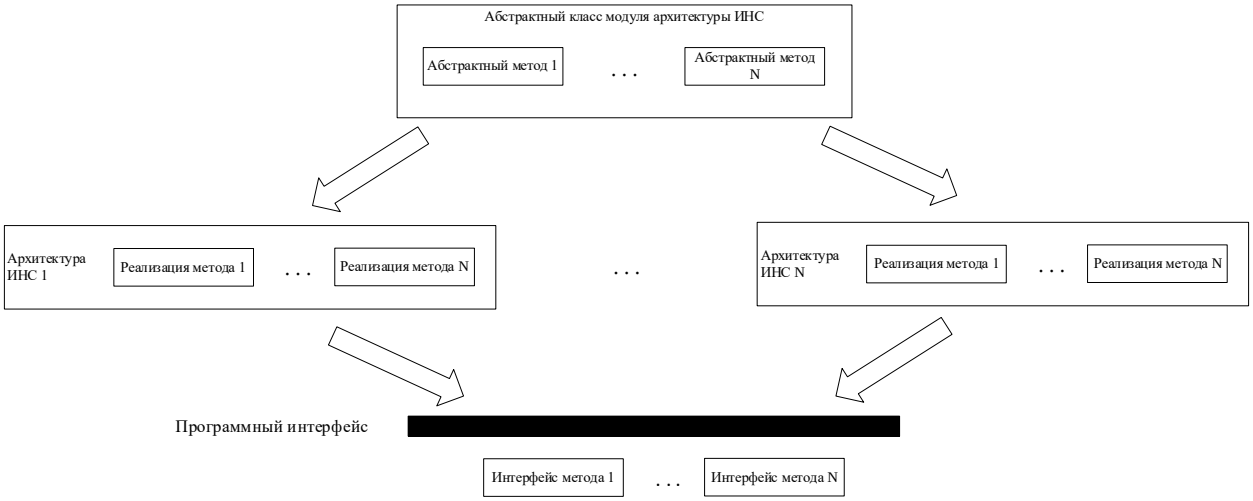

Рисунок 3.4 – Обобщённая схема реализации шаблона проектирования «Фасад» в разработанном ПО

Как видно из данной схемы, модуль каждой архитектуры ИНС ПР наследуется от общего абстрактного класса и реализует все методы, которые необходимы для комплексной автоматизации процессов генерации и обучения модели ИНС ПР соответствующей архитектуры. Внешние модули ПО обращаются к модулям этих архитектур через общий программный интерфейс, который в зависимости от выбранной архитектуры, вызывает конкретную реализацию соответствующего метода.

Подобная организация позволяет легко масштабировать ПО, добавляя в него новые архитектуры. Ведь в случае добавления новой архитектуры достаточно лишь создать для неё класс, наследуемый от абстрактного класса архитектур ИНС ПР данного ПО, и реализовать все необходимые методы. На практике бывает достаточно лишь подключить стороннюю программную библиотеку, реализующую новую архитектуру ИНС ПР, и переопределить вызываемые функции, поскольку зачастую такие библиотеки уже содержат всю требуемую

функциональность. В итоге, вся интеграция новой архитектуры ИНС ПР в разработанный ПО может свестись лишь к адаптации сторонних программных библиотек под соответствующий программный интерфейс. Именно таким образом в ПО была интегрирована архитектура Mask R-CNN.

3.3 Используемые программные библиотеки и программно-аппаратные ускорители

Описываемое в диссертации ПО было разработано на языке программирования Python [28], основные достоинства которого связаны с его кроссплатформенностью, расширяемостью и большим количеством сторонних программных библиотек, используемых для решения конкретных задач.

Python является высокоуровневым языком программирования общего назначения. Основными особенностями Python являются: наличие динамической строгой типизации; автоматическое управление памятью; ориентированность на повышение производительности разработчика за счёт повышения читаемости программного кода; обеспечение переносимости программного кода между различными программными средами и платформами. Python является объектно-ориентированным и интерпретируемым языком программирования. С одной стороны, это приводит к тому, что программы, написанные на нём, зачастую имеют низкую скорость работы и более высокое потребление памяти, по сравнению с компилируемыми языками программирования. С другой стороны, у Python имеется возможность использовать программные библиотеки написанные на языках программирования C и C++. Применение библиотек, которые реализуют низкоуровневую вычислительную логику и работу с аппаратной платформой и при этом написаны на компилируемых языках программирования, позволяет минимизировать различие в скорости работы для многих задач. В том числе, подобные библиотеки на Python имеются и для решения прикладных задач, где используются МО и СОА.

Также, Python является мультипарадигменным языком программирования, что позволяет эффективно реализовывать на нём как логику сетевого

взаимодействия между сервисами, в рамках парадигмы СОА, так и функциональные вычисления, связанные с МО. Поэтому в разрабатываемом ПО он использовался как для написания сервисных оболочек для сгенерированных и обученных моделей ИНС ПР, так и для реализации разработанного алгоритма УПГ, который и занимался созданием этих моделей.

Для решения задач, связанных с интеграцией парадигмы СОА в разрабатываемый ПО, использовалась программная библиотека Flask [131]. Flask является программной библиотекой языка Python, которая реализует в себе концепцию микрофреймворка, подразумевающую под собой создание минималистичных каркасов веб-сервисов, реализующих лишь базовые возможности, но которые впоследствии могут легко быть расширены различными сторонними библиотеками и инструментами.

По умолчанию Flask не включает в себя обработку на уровне абстракции базы данных, не осуществляет проверку формы или другие специализированные функции веб-приложения, которые реализованы в рамках уже существующих программных библиотек на Python. Но Flask предоставляет программный интерфейс для подключения подобных программных библиотек к создаваемому ПО и даёт возможность обращаться к ним единообразно, без необходимости внесения дополнительных модификаций.

Выбор Flask для использования в разрабатываемом ПО обусловлен как раз его простотой, универсальностью, расширяемостью и масштабируемостью. Поскольку предполагается генерация и обучения моделей ИНС ПР для решения задач в совершенно разных прикладных областях нельзя заранее описать весь список требований к реализации сервисной программной оболочки для созданной модели. Этот список, которому должна будет удовлетворять создаваемая программная оболочка, будет отличаться как для разных прикладных задач, так и для различных задач даже в одной прикладной области. Всё это приводит к тому, что использование нерасширяемых программных библиотек противоречит одной из главных задач, поставленных перед ПО, даже если подобные нерасширяемые библиотеки показывают большую скорость работы, по сравнению с Flask.

На сегодняшний день имеется ряд примеров реализации ПО, поддерживающего парадигму СОА на языке программирования Python [70, 148]. Эти примеры демонстрируют как концептуальную возможность реализации парадигмы СОА на Python, так и эффективность данной реализации для ряда задач.

Для реализации алгоритмов МО в целом и ИНС в частности использовались программные библиотеки TensorFlow [133] и Keras [27].

TensorFlow является открытой программной библиотекой языка программирования Python, которая реализует алгоритмы МО. Внутренние функции программной библиотеки написаны на языке C++, что делает её крайне эффективной в многопроцессорных матричных вычислениях. Первоначальная версия TensorFlow была разработана командой Google Brain для внутреннего использования в Google, однако в 2015 году к программной библиотеке был открыт свободный доступ по лицензии Apache 2.0.

У библиотеки TensorFlow имеются следующие особенности:

- Данная библиотека реализует большое количество алгоритмов МО, помимо ИНС. Это позволяет использовать её не только для реализации самих моделей ИНС ПР, генерируемых и обучаемых ПО, но и для реализации разработанного алгоритма УПГ.
- Библиотека имеет широкий инструментарий для работы с моделями МО. Это позволяет как тонко настраивать алгоритм УПГ (используя уже оптимизированные программные функции соответствующих архитектур ИНС), так и более тонко настраивать генерируемые и обучаемые модели ИНС ПР. Также, это позволяет автоматизировать перебор большего числа параметров, в процессе синтеза моделей ИНС ПР с помощью алгоритма УПГ.
- В дополнение к реализации алгоритмов МО, TensorFlow также включает собственную систему работы с матричными данными, что позволяет реализовать единообразный подход к работе с массивами данных, используемых как в обучении моделей ИНС ПР, так и в процессе их последующей эксплуатации.
- Библиотека имеет интеграцию с рядом программно-аппаратных платформ, повышающих эффективность обучения и работы алгоритмов МО. В том

числе TensorFlow имеет интеграцию с программно-аппаратной платформой CUDA, что является дополнительным достоинством в рамках разрабатываемого ПО.

- Программная библиотека реализует множество архитектур ИНС ПР в своей базовой версии. При этом, с одной стороны, количество поддерживаемых программной библиотекой архитектур расширяется с каждым годом, с другой сама программная библиотека имеет программный интерфейс, позволяющий на её основе реализовывать пользовательские архитектуры ИНС.

Также, у данной библиотеки есть и ряд небольших достоинств, которые выделяют её на фоне конкурентов, к примеру на фоне того же PyTorch. Для TensorFlow на сегодняшний день написана и выложена в открытый доступ наиболее подробная документация и больше всего руководств от сторонних специалистов. При этом, программный интерфейс TensorFlow поддерживает множество сторонних разработчиков ИНС, что позволяет интегрировать разработанные ими архитектуры в ПО, работающее на TensorFlow.

Все перечисленные достоинства, как и в случае с программной библиотекой Flask, делают TensorFlow наиболее подходящим кандидатом для решения поставленных перед разрабатываемым ПО задач. Не являясь самой эффективной во многих случаях программной библиотекой, она при этом является самой гибкой, масштабируемой и распространённой. Всё это упростит как разработку уже выбранных архитектур ИНС ПР, так и внедрение новых архитектур впоследствии.

Вторая выбранная программная библиотека Keras является надстройкой над программными платформами Deeplearning4j, Theano и, в том числе, TensorFlow. Концептуально Keras является не автономной программной библиотекой МО, а программным «фасадом» над существующими библиотеками. Данная программная библиотека предназначена для реализации высокоуровневого набора программных абстракций, которые на более низком уровне реализуются конкретными программными платформами. Основной целью Keras является инкапсуляция логики работы с конкретными программными библиотеками МО, что должно упростить формирование различных архитектур ИНС, вне зависимости от выбранной программной платформы.

Keras является программным интерфейсом, который был разработан прежде всего для использование людьми. Он ориентирован на удобство использования программистами, на автоматизацию их работы с моделями МО. Keras реализует согласованный и простой в использовании программный интерфейс, что позволяет минимизировать количество действий разработчика, необходимых для построения моделей МО. В контексте разрабатываемого ПО это достоинство упрощает создание модулей, реализующих работу с архитектурами ИНС ПР, которые были выбраны в рамках рассматриваемой диссертации. Также, в дальнейшем это упростит разработку и внедрение новых архитектур ИНС.

Также Keras имеет модульную расширяемую структуру, которая продолжает парадигму разработки заложенную в TensorFlow. В Keras реализован набор автономных, сконфигурированных модулей, которые могут быть подключены к разрабатываемому ПО без каких-либо ограничений. Эти модули также можно комбинировать между собой различным образом, что позволяет тонко настраивать создаваемые модели ИНС. Благодаря этому реализуется заложенная в разрабатываемый ПО идея модульности и инкапсуляции работы с различными архитектурами ИНС ПР.

Отдельно стоит отметить расширяемость Keras, которая позволит впоследствии легко интегрировать в разрабатываемый ПО новые модули, классы и функции, и, следовательно, работать с новыми архитектурами ИНС. Возможность добавлять новые модули, реализованная в библиотеке Keras, позволит наиболее полно решить задачу универсального и единообразного подхода к генерации и обучению моделей ИНС ПР различных архитектур. И, поскольку разработка Keras продолжается, и данная библиотека расширяется новыми архитектурами ИНС, её использование позволит и в дальнейшем расширять разрабатываемый ПО наиболее быстрым и простыми образом.

Выбор описанного стека технологий объясняется тем, что перед ПО не ставится задача реализации не типовых специализированных решений. Напротив, требуется быстрая реализация уже известных архитектур. Использование уже разработанных, протестированных и оптимизированных библиотек полностью

удовлетворяет поставленной задаче. Также, ключевыми требованиями являются расширяемость и масштабируемость. Соответственно, реализация ПО на базе постоянно дорабатываемых и расширяемых программных библиотек позволит добавлять новые архитектуры ИНС ПР и средства работы с ними, за счёт единого программного интерфейса. При этом кроссплатформенность описанного стека и поддержка парадигмы СОА позволит масштабировать ПО на различные аппаратные платформы.

В разрабатываемом ПО предполагается автоматизировать обучение моделей различных архитектур ИНС ПР, в том числе и свёрточных, поэтому следует учитывать и высокие требования к вычислительным системам, на которых это обучение должно будет проводиться. Разработка собственных программно-аппаратных ускорителей для поставленной задачи не является оправданной. Предполагается создание универсальных программных решений, поэтому потребуется использование соответствующих универсальных программных библиотек, которые бы не накладывали никаких дополнительных ограничений на процесс эксплуатации созданных решений. Исходя из этих предпосылок, было решено использовать программно-аппаратную платформу CUDA (Compute Unified Device Architecture) [54], созданную и распространяемую компанией NVidia.

CUDA является программно-аппаратной платформой, реализующей распараллеливание вычислений пользовательских программ на аппаратном уровне, что позволяет существенно увеличить скорость вычислений за счёт использования графических процессоров NVidia [65, 106]. CUDA представляет из себя набор программных библиотек, имеющих низкоуровневую реализацию и способных напрямую работать с графическими процессорами компании Nvidia. Использование этих библиотек позволяет программным модулям напрямую обращаться к аппаратной платформе, в обход операционной системы, на которой выполняется программа. При этом, сами программные библиотеки CUDA реализуют наиболее эффективное вычисление заложенных в них алгоритмов и их распараллеливание. Также CUDA позволяет через свои программные библиотеки напрямую обращаться к инструкциям GPU и управлять его памятью. Программные

библиотеки CUDA поддерживают обращение к ним посредством нескольких языков программирования, в том числе и Python. Базовый принцип работы CUDA представлен на рисунке 3.5.

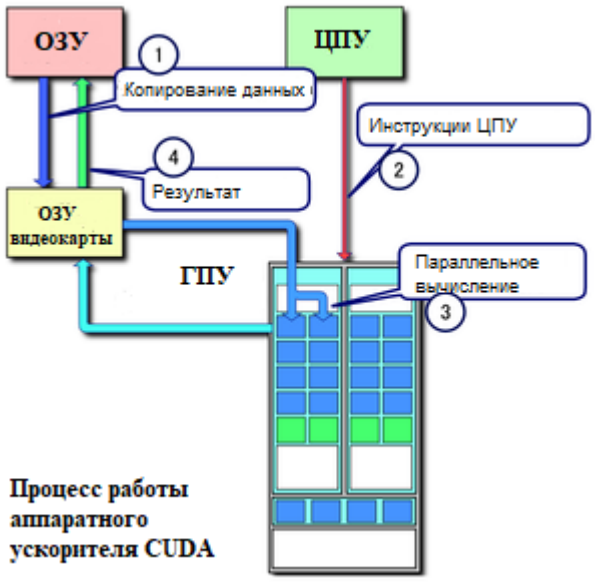

Рисунок 3.5 – Базовый принцип работы CUDA (заимствовано из [100])

Как видно из рисунка, основной принцип распараллеливания вычислений и ускорения работы с приложениями заключается в том, чтобы инструкции и команды, которые должны выполняться на центральном процессоре, обрабатывались на многоядерном графическом процессоре. При этом наибольший прирост в скорости вычислений будет при выполнении массива однотипных команд, поскольку сама архитектура графического процессора, из-за специфики работы с изображениями, ориентирована прежде всего на параллельные однотипные вычисления. Логика работы пользовательских программ с платформой CUDA представлена на рисунке 3.6.

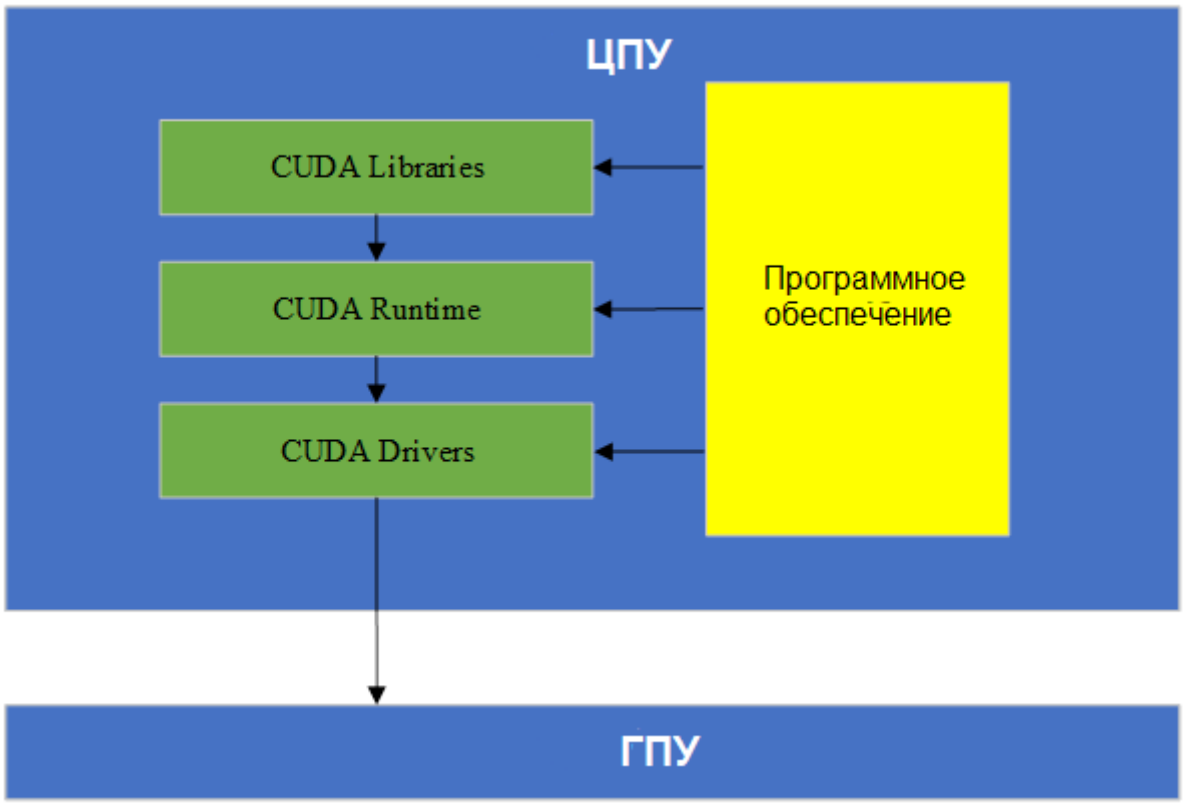

Рисунок 3.6 – Логика взаимодействия пользовательских приложений с платформой CUDA

На рисунке видно, что взаимодействие с процессором осуществляется исключительно с помощью CUDA, но при этом пользовательское приложение может осуществлять это взаимодействие на трёх уровнях абстракции.

CUDA Libraries это основные программные библиотеки, адаптированные для использования высокоуровневыми языками программирования. Они полностью инкапсулируют логику работы с драйверами и предоставляют только функции для конкретных высокоуровневых вычислений.

CUDA Runtime является программной обёрткой над драйверами видеокарты и предоставляет возможность создания пользовательских функций. Данный уровень позволяет пользователям из базовых функций драйверов создавать собственные высокоуровневые функции для специфических вычислений, которые отсутствуют в стандартных библиотеках. Данный уровень уже не инкапсулирует логику работы с драйверами, а лишь предоставляет программный интерфейс для работы с ними.

CUDA Driver являются непосредственно видеодрайверами, которые занимаются всем низкоуровневым взаимодействием с графическим процессором и внутренний графической видеопамятью. CUDA Runtime, и CUDA Libraries осуществляют взаимодействие с видеокартой исключительно с помощью функций драйверов. При этом сами драйвера реализуют взаимодействие с видеокартой наиболее эффективным образом.

Реализованное, в рамках платформы CUDA, ускорение параллельных вычислений позволяет непосредственно говорить о повышении скорости обучения моделей ИНС. Поскольку параллельными являются те вычисления, которые можно разбить на множество небольших, однотипных задач, выполнение каждой из которых не зависит от других, работу большинства моделей ИНС можно полностью реализовать на данном виде архитектур. Почти все операции, которые выполняются процессе работы модели ИНС легко разбиваются на более мелкие асинхронные операции. Самым заметным примером, работа которого наиболее ускоряется за счёт использования параллельных вычислений, является процесс свёртки в СНС (рисунок 3.7).

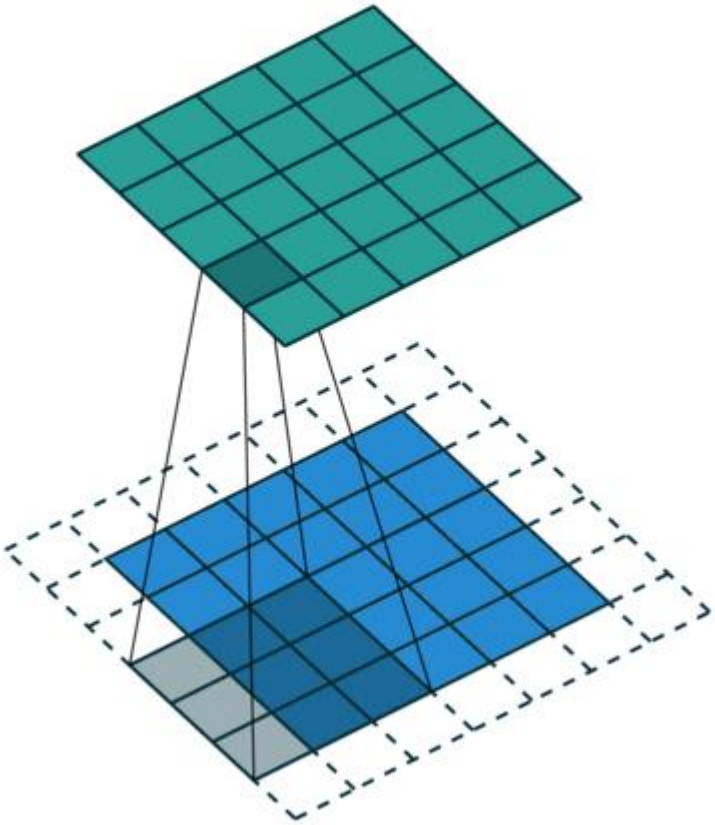

Рисунок 3.7 – Иллюстрация процесса свёртки в СНС

На этом рисунке синим цветом отмечен входной канал свёртки (который может быть как исходным изображением, так и предыдущим свёрточным слоем), зелёными выходной канал свёртки (текущий свёрточный скрытый слой), серым отмечен свёрточный фильтр, так же называемый «скользящим окном» [86].

Для каждой ячейки входного канала (которая хранит информацию о пикселе входного изображения или является выходом нейрона предыдущего слоя) свёрточный фильтр производит операции суммирования и преобразования, для их последующей обработки в нейроне выходного канала, которые никак не зависят друг от друга. При этом, вычисления свёрточного фильтра для каждого конкретного нейрона выходного канала никак не зависят от вычислений для других нейронов. Поэтому в случае свёрточного слоя СНС распараллелить можно как вычисления в рамках одной итерации свёртки свёрточного фильтра, так и в целом вычисления свёрточного фильтра для всех нейронов одного свёрточного слоя. Это

приводит к большой востребованности разнообразных инструментов параллельных вычислений в целом и CUDA в частности.

При этом, для данной платформы компания NVidia создала и надстройку, которая повышает эффективность работы с различными библиотеками, связанными с МО – CUDA-X AI [104]. Данное расширение имеет глубокую интеграцию с одними библиотеками и программный интерфейс для работы с другими [103].

Использование данного расширения в разрабатываемом ПО позволит упростить взаимодействие с различными программными библиотеками, реализующими различные архитектуры ИНС ПР. Такие библиотеки имеют как уникальные архитектуры, так и модификации, повышающие точность и скорость работы уже существующих архитектур, что делает их применение оправданным при решении различных специфических задач. Поэтому возможность работать с различными библиотеками и обеспечение их доступа к программно-аппаратному ускорению помогает достичь цели разрабатываемого ПО - единообразного подхода к генерации и обучению моделей различных архитектур ИНС ПР.

С другой стороны, CUDA-X AI так же реализует и возможность работы с различными аппаратными и облачными платформами, что в перспективе может быть использовано в процессе эксплуатации разрабатываемого ПО.

Дополнительно стоит учитывать, что с 2017 года компания NVidia внедряет в свои видеокарты технологию тензорных ядер [125], которая ещё больше повышает скорость работы и обучения моделей ИНС. Тензорные ядра являются вычислительным блоком графического процессора, который за один такт вычисляет не одну операцию, а производит вычисление матрицы. Т.е. тензорное ядро применяет оператор матричного вычисления к массиву данных, поэтому использование тензорных ядер позволяет в разы уменьшить время матричных вычислений. При этом выгода от использования тензорных ядер растёт экспоненциально, в зависимости от размеров самих матриц.

С другой стороны, как уже говорилось выше, все вычисления, связанные с нейронами в ИНС, носят типовой характер и не зависят друг от друга. Проще

говоря, веса нейронов каждого слоя можно представить в виде матрицы и далее последовательно производить матричные вычисления над всем слоем, а не рассчитывать отдельно значения для каждого нейрона. Вместе с распараллеливанием вычислений, использование тензорных ядер позволит дополнительно уменьшить время, затрачиваемое на обучение и работу различных моделей ИНС ПР. К примеру, время вычисления операций СНС уменьшается на большую долю, по сравнению с вычислением операций МПР, поскольку слои СНС имеют большую размерность.

Представленные выше достоинства программно-аппаратной платформы CUDA позволяют говорить о необходимости её использования в разрабатываемом ПО. Ведь, помимо универсальности и возможности работать с различными архитектурами, разрабатываемый ПО также должен давать возможность параллельной работы в системе нескольким пользователям. При росте числа пользователей экспоненциально растут требования к вычислительным ресурсам. Учитывая, что большинство ИНС ПР можно представить в виде матриц, при увеличении количества матриц, количество вычислений на обычных центральных процессорах будет расти экспоненциально, а не линейно. При этом стоит отметить, что программно-аппаратных решений аналогичных CUDA на сегодняшний день нет, поэтому проводить сравнение не с чем.

Отдельно стоит отметить, что в разрабатываемом ПО используется программная библиотека CUDA SDK, которая позволяет при обучении моделей ИНС использовать аппаратное ускорение видеокарт компании NVidia [66, 128]. Данной технологии особенно существенно ускоряет обучение СНС [130].

Конкретно, в разрабатываемом ПО использовалась программная библиотека NVidia CUDA Deep Neural Network (cuDNN) [105]. cuDNN является библиотекой примитивов ИНС, функции базовых вычислений которых реализованы с помощью программно-аппаратной платформы CUDA, что позволяет ускорять эти вычисления за счёт работы графического процессора. cuDNN реализует работу многих стандартных функций и алгоритмов, применяемых в области ГО, таких как алгоритмы прямой и обратной свертки, методы объединения и нормализация

векторов данных, а также функций активации. Поскольку cuDNN инкапсулирует всю логику работы с аппаратной платформой, её использование решает все вопросы, связанные с оптимизацией программного кода для выполнения на конкретных аппаратных решениях.

По сути, cuDNN является программным интерфейсом доступа к технологиям CUDA-X AI, которые были описаны выше. И основная причина использования этой библиотеки заключается именно в возможности доступа к этим технологиям и их применения в разрабатываемом ПО.

При этом данная программная библиотека также имеет глубокую интеграцию с библиотекой TensorFlow. Смысл этой интеграции заключается в решении следующей проблемы - поскольку обе библиотеки используют для основных вычислений модули написанные на C++, наибольшие затраты по времени, при их взаимодействии, занимает процесс передачи данных между программными библиотеками. Для уменьшения этих затрат между этими двумя программными библиотеками реализован прямой программный интерфейс, который позволяет им напрямую обращаться к низкоуровневым вычислительным модулям друг друга.

Подводя итог по описанному стеку технологий, можно сказать, что все выбранные библиотеки уже имеют тесную интеграцию друг с другом, которая во многом обусловлена достоинствами выбранного языка программирования. При этом, все представленные библиотеки используют для вычислений как низкоуровневые программные модули, написанные на языке C++, так и программно-аппаратное ускорение за счёт использования платформы CUDA. Всё это приводит к решению множества проблем, которые могут возникнуть при эксплуатации разрабатываемого ПО, за счёт инкапсуляции всей низкоуровневой логики в выбранных программных библиотеках.

При этом за счёт кроссплатформенности как самого языка программирования, так и всех используемых программных библиотек, можно говорить и о кроссплатформенности генерируемых программных оболочек для создаваемых моделей ИНС ПР. Все это расширяет круг решаемых

разрабатываемым ПО задач и позволяет более полно соответствовать выбранной парадигме СОА.

3.4 Выводы по разделу 3

1. Разработаны и обоснованы состав, структура и технологии использования созданного ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур. При этом разработана и реализована как общая архитектура спроектированного ПО, так и составляющие его модули. В состав ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур вошло одиннадцать программных модулей, реализующих на конструктивном уровне существующие парадигмы СОА и модульной структуры. Разработанные программные модули использует шаблон проектирования «Фасад», который позволяет свободно масштабировать и расширять созданное ПО, в том числе за счёт добавления новых архитектур ИНС ПР.

2. Принципиальная отличительная особенность разработанного ПО заключается в комплексной автоматизации всех этапов генерации и обучения моделей ИНС ПР различных архитектур, а также этапов генерации программных и сервисных оболочек для созданных моделей. Созданные программные модули в автоматизированном режиме, согласно указанным пользователем настройкам, обеспечивают реализацию полного цикла проектирования и генерации моделей ИНС: от базовой нормализации обучающих данных и выбора гиперпараметров модели до настройки параметров процесса обучения. В результате работы ПО формируется пакет кроссплатформенных программ, который позволяет свободно использовать созданную модель ИНС для решения практических задач напрямую или интегрировать её в существующее ПО и обращаться к ней через программный интерфейс. Пользовательских интерфейс ПО предоставляет необходимый минимум инструментов для автоматизации процесса создания моделей ИНС ПР различных архитектур. Таким образом, вся сложность процессов генерации и

обучения моделей ИНС скрыта от пользователя за счёт автоматизации данных процессов. За счёт этого в созданном ПО реализуется концепция No-Code разработки, благодаря которой пользоваться им смогут не только специалисты с высоким уровнем знаний в области МО, но и специалисты из различных прикладных областей, не имеющие компетенций по вопросам использования МО и ГО, а также другие пользователи, заинтересованные в создании моделей ИНС ПР.

3. Были решены задачи, связанные с упрощением интеграции создаваемых моделей ИНС в стороннее ПО на основе использования парадигмы СОА. Данный подход не привязан к конкретному ПО и сетевому окружению, поэтому может быть реализован с помощью различных технологий. При этом к созданным моделям ИНС ПР можно будет обращаться единообразно из различных программных сред.

4. В разработанное ПО были интегрированы сторонние программные библиотеки и программно-аппаратные ускорители, позволяющие реализовать выбранные во 2 главе алгоритмы и методы, которые требуют больших вычислительных ресурсов. Без использования предложенных программных библиотек и программно-аппаратных ускорителей решение практических задач могло бы занимать месяцы, что полностью лишало бы смысла использование представленного ПО. Всё это повышает универсальность создаваемого программного обеспечения и существенно упрощает процесс интеграции синтезированных моделей ИНС ПР.

5. Новым практическим результатом, полученным в данной главе, стало конкретное доказательство реализуемости концепции комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур, предложенной в предыдущих главах диссертации. Было продемонстрировано, что за счёт использования модульной архитектуры и парадигмы СОА можно создать ПО, которое бы единообразно осуществляло взаимодействие с принципиально различными архитектурами ИНС ПР.

4 МЕТОДИКА ИСПОЛЬЗОВАНИЯ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРИ РЕШЕНИИ ПРИКЛАДНЫХ ЗАДАЧ ИЗ РАЗЛИЧНЫХ ПРЕДМЕТНЫХ ОБЛАСТЕЙ

Благодаря заложенной в разрабатываемое ПО универсальности, оно успешно использовалось на практике для решения широкого спектра прикладных задач из совершенно разных прикладных областей. С помощью данного ПО были созданы и обучены модели ИНС, которые были успешно интегрированы в стороннее ПО. Количество модификаций, которые пришлось внести в автоматически сгенерированные программные оболочки, отличаются для каждой конкретной прикладной задачи. При этом их модификации были вызваны исключительно особенностями конкретных прикладных областей. Однако, ни в одном случае не пришлось вносить модификации в саму модель и редактированию подверглись в различной степени только программные интерфейсы.

Отдельно стоит отметить, что на сегодняшний день интерфейс ПО ещё находится в процессе разработки, поэтому конечным пользователям пока что ещё требуются знания в области МО, чтобы можно было работать с данным ПО. К настоящему моменту времени часть прикладных задач была решена на уровне создания отдельных программных модулей, содержащих созданные ИНС. Также есть пример практического использования разработанного ПО для создания автономного ПО, которое было введено в эксплуатацию и успешно используется сторонними организациями.

4.1 Базовая методика использования разработанного программного обеспечения

Поскольку в разработанном ПО реализована концепция No-Code разработки, сама методика работы с ним не требует написания программного кода. Также высокая степень автоматизации не требует непосредственного участия пользователя в генерации и обучении моделей ИНС. Тем не менее, в зависимости от поставленной перед пользователем задачи, процесс работы с ПО может отличаться. На основе результатов выполненных исследований и многочисленных

машинных экспериментов была сформирована обобщённая методика работы с ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур. Она включает в себя следующие этапы:

Этап 1. Пользователь должен сформировать входную выборку данных, на основание которой будет осуществляться обучение и тестирование создаваемых моделей (разбиение на обучающую, тестовую и проверочную выборку будет осуществлено разработанным ПО автоматически). Тем не менее, поскольку в рамках диссертации не ставилась задача автоматической подготовки, нормализации и разметки обучающих данных, все эти операции пользователю требуется осуществить самостоятельно.

Этап 2. Пользователь должен загрузить в ПО подготовленные и размеченные данные, выбрать класс решаемой задачи и дать название модели.

Этап 3. Данный шаг является опциональным. Для выбранного класса задач автоматически будут выбраны по умолчанию начальные значения соответствующих гиперпараметров. Однако, пользователь, при желании, может выбрать некоторые значения самостоятельно. Для разных архитектур ИНС ПР набор варьируемых гиперпараметров и их значения будут отличаться.

Этап 4. После выбора всех начальных настроек пользователь должен запустить процесс автоматического синтеза модели ИНС ПР.

Этап 5. После окончания синтеза модели отображаемый на пользовательском интерфейсе статус задачи изменится, и пользователь сможет скачать архив, содержащий программную реализацию готовой модели и сопутствующее ПО.

Этап 6. Получив программную реализацию созданной модели ИНС ПР, пользователь может интегрировать её в стороннее ПО или использовать напрямую.

Поскольку описанная методика является обобщённой, на практике набор действий на этапах 1 и 6 может отличаться для различных прикладных задач. Примеры таких ситуаций будут рассмотрены далее при решении трех различных прикладных задач.

4.2 Модель и программа прогнозирования уровня воды при возникновении ледовых заторов

В настоящее время, несмотря на достижения в области формального описания и исследования сложных объектов, остаётся большой класс объектов и процессов, которые всё ещё крайне сложно моделировать с целью дальнейшего решения соответствующих задач анализа и прогнозирования их поведения. Процесс весеннего вскрытия льда и последующего половодья на реках как раз является примером подобного класса процессов. Помимо зависимости от метеорологических и гидрологических параметров, существенное влияние на этот процесс оказывают и географические особенности самих рек. Для прогнозирования поведения реки в половодье сотрудниками СПИИРАН была разработана ИС «ПРОСТОР» [143]. Используемые в этой системе модели и методы применяются для оперативного прогнозирования состояния речного русла.

В ходе разработки данной ИС стало ясно, что для обеспечения требуемого уровня адекватности указанных моделей необходима их адаптация к динамически изменяющимся условиям. Предварительный анализ показал, что использование достоинств существующих алгоритмов МО позволяет успешно применять их для решения задачи прогнозирования дня вскрытия льда и последующего прогнозирования уровня воды [32, 97, 141].

Поскольку гидрологические и метеорологические наблюдения применительно к каждой реке РФ проводятся уже много десятилетий, имеется большой объём исторических статистических данных, используя которые можно проанализировать процессы ледохода в различных условиях, и уже на основании этих данных провести генерацию и обучение модели ИНС для решения задач прогнозирования наводнений.

В последние годы создано множество крупных ИС на базе ИНС, прогнозирующих уровень воды на реках, которые продемонстрировали свою определенную эффективность. Однако все эти системы использовались для прогнозирования уровня воды на реках, которые либо не замерзают, либо уровень их промерзания зимой незначителен. Кратковременное прогнозирование уровня

воды на реках с глубоким промерзанием и прогнозирование дня вскрытия льда является на данный момент уникальной особенностью моделей ИНС, сгенерированных и обученных с помощью разработанного ПО. В качестве примера, иллюстрирующего возможности созданных моделей ИНС, приводятся результаты решения с их помощью задач прогнозирования даты вскрытия льда и уровня воды на участке реки Северная Двина, возле города Великий Устюг.

Модели ИНС генерировались и обучались на данных, которые были получены с участка реки Северная Двина в районе города Великий Устюг на 1 этапе методики использования разработанного ПО. Данный участок, из-за особых характеристик русла, имеет повышенную опасность образования ледовых заторов. Требовалось создать две отдельные модели ИНС: для долгосрочного прогнозирования дня вскрытия льда и определения максимального уровня воды в течение весеннего сезона, а также для кратковременного прогнозирования вероятности вскрытия льда и уровня воды в ближайшие 3 дня относительно даты расчёта прогноза. Данные для обучения моделей ИНС были получены из открытых источников. Выбор параметров предсказания вероятности заторов обуславливался известными существующими закономерностями зажоро- и заторообразования [1].

Далее, для решения задачи долгосрочного прогнозирования дня вскрытия льда и максимального уровня воды за весну, на 2-5 этапах методики использования разработанного ПО, была сгенерированы модели ИНС, архитектура и настройки которых подбирались исходя из значений параметров объекта прогнозирования, в рамках автоматизированного обучения. Разработанным ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур была автоматизировано создана модель ИНС архитектуры МПР, имеющая следующие характеристики:

- входной слой – 8 нейронов;
- первый скрытый слой – 50 нейронов, функция активации softsign;
- второй скрытый слой – 50 нейронов, функция активации softsign;
- выходной слой – 3 нейрона, функция активации softsign;

- функция потерь – logcosh;
- алгоритм оптимизации – adaptive moment estimation (adam);
- количество эпох обучения – 900.

Для решения задачи кратковременного прогнозирования величины процентной вероятности вскрытия льда и уровня воды в ближайшие три дня относительно даты расчёта прогноза, по тем же принципам, была построена следующая модель МПР:

- входной слой – 16 нейронов;
- первый скрытый слой – 66 нейронов, функция активации softsign;
- второй скрытый слой – 66 нейронов, функция активации softsign;
- выходной слой – 6 нейронов, функция активации sigmoid;
- функция потерь – logcosh;
- алгоритм оптимизации – adaptive moment estimation (adam);
- количество эпох обучения – 1200.

Результаты тестирования созданных моделей показали, что прогноз даты вскрытия льда, рассчитанный с использованием ИНС, оказался не хуже прогноза Росгидромета (с точки зрения конечного результата прогнозирования). Однако на данных за период с 2000 года, прогноз, выполненный с помощью модели ИНС, имеет большую погрешность, т.к. он не учитывал субъективный фактор, вызванный влиянием человека на естественный ледоход (в частности подрыв льда). Поэтому на данных в период с 2000 года с использованием ИНС осуществлялся прогноз естественного ледохода, который был бы в отсутствие влияния человека.

Также созданная модель ИНС, помимо даты вскрытия льда, рассчитывает прогноз вероятности затора. Прогнозы данной вероятности в сводном виде Росгидромет не даёт, а только предоставляет сведения о предполагаемом затоплении, при наступлении затора, в связи с чем сравнить их не представляется возможным. Результаты кратковременного прогнозирования уровня воды, полученные с использованием модели ИНС, отличаются большей точностью, по сравнению с долгосрочным прогнозом дня вскрытия льда. В свою очередь,

аналогичные централизованные системы оперативного прогнозирования даты вскрытия и характеристик ледохода у Росгидромета отсутствуют.

Результаты тестирования моделей ИНС долгосрочного и краткосрочного прогнозирования характеристик ледохода (рисунки 4.1 – 4.8) демонстрируют, что созданные в рамках данной диссертационной работы модели ИНС имеют сопоставимую с прогнозом Росгидромета точность, а также обладают широким спектром прогнозируемых параметров и возможностью получения уточнённого прогноза в течение всей весны.

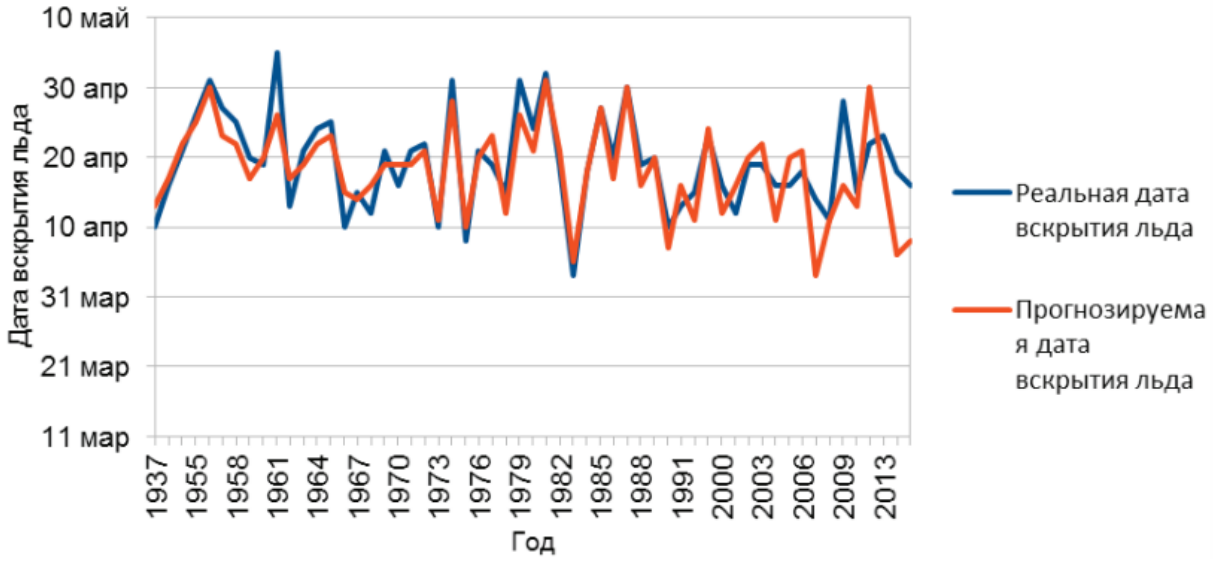

Рисунок 4.1 – Результаты тестирования долгосрочного прогноза даты вскрытия льда на диапазоне данных за 1937 – 2015 года

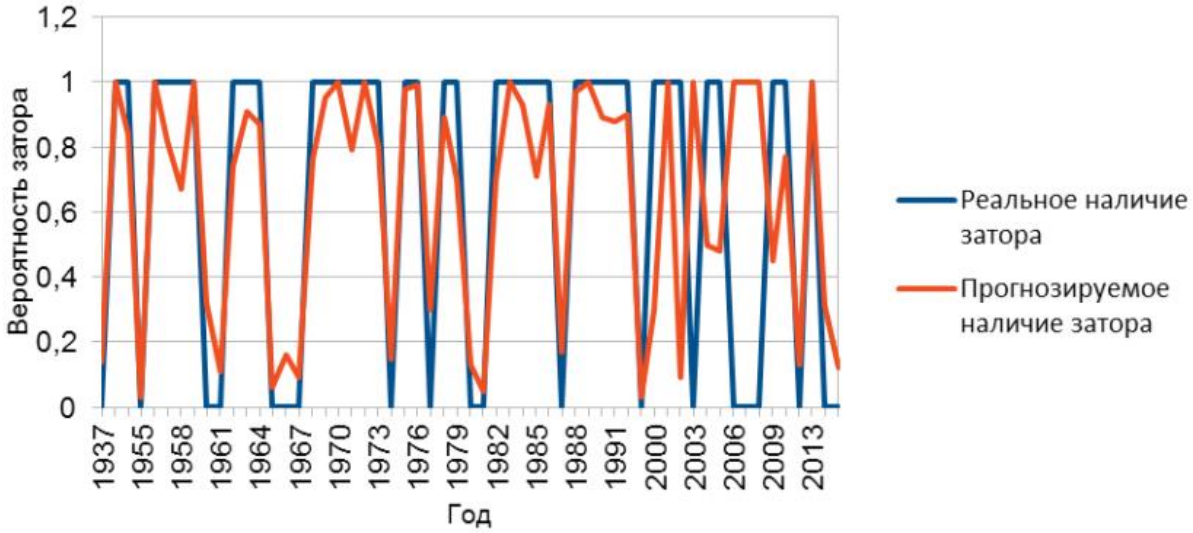

Рисунок 4.2 – Результаты тестирования прогноза вероятности затора на диапазоне данных за 1937 – 2015 года

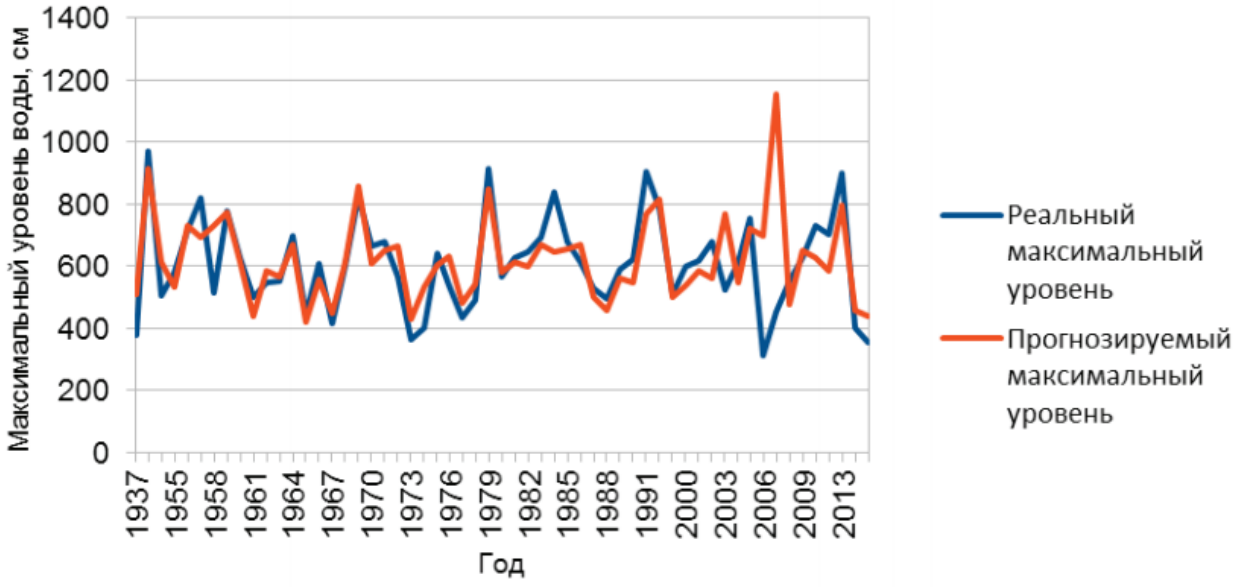

Рисунок 4.3 – Результаты тестирования максимального уровня воды на диапазоне данных за 1937 – 2015 года

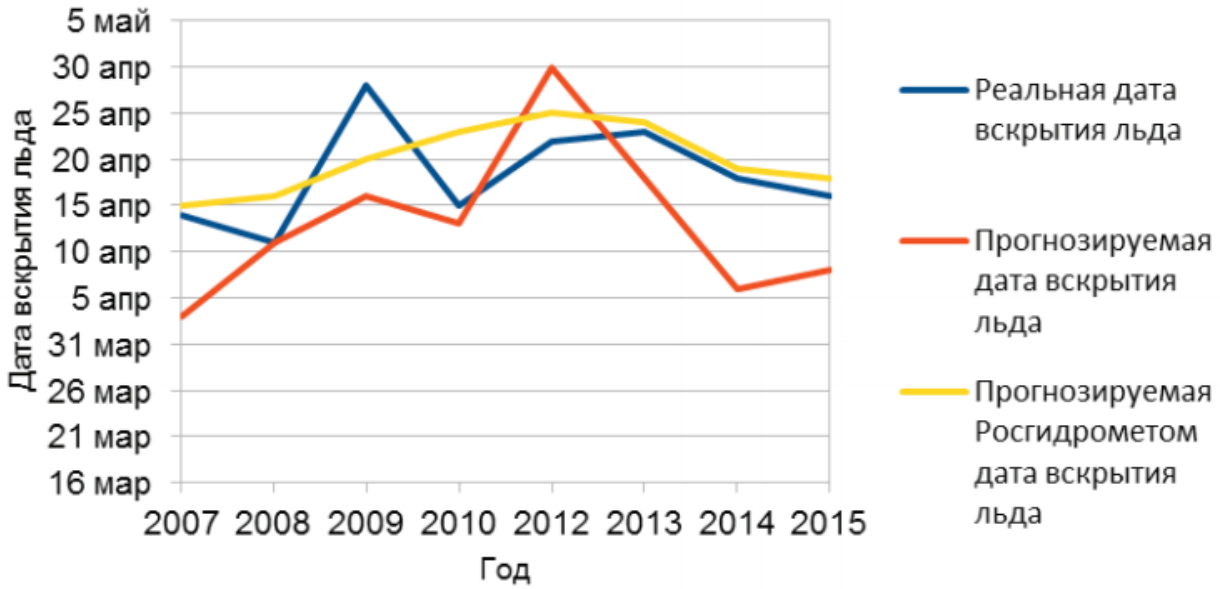

Рисунок 4.4 – Сравнение прогноза модели ИНС с прогнозом Росгидромета, на диапазоне данных за 2007 – 2015 года.

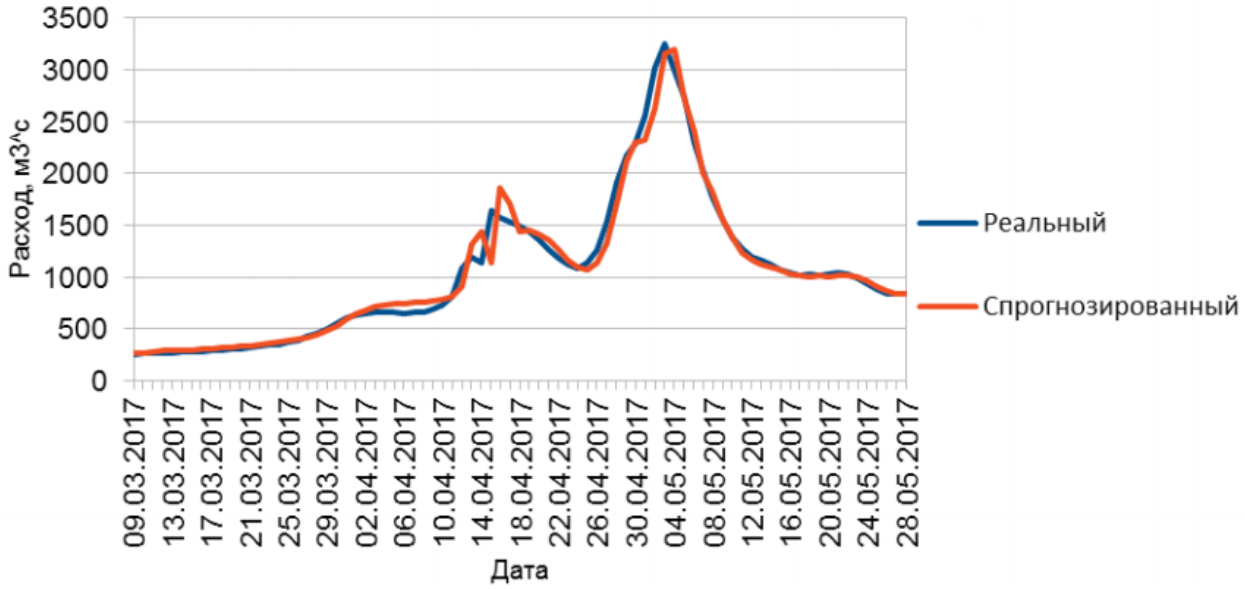

Рисунок 4.5 – Краткосрочный прогноз расхода воды на завтра в период весеннего ледохода 2017 года

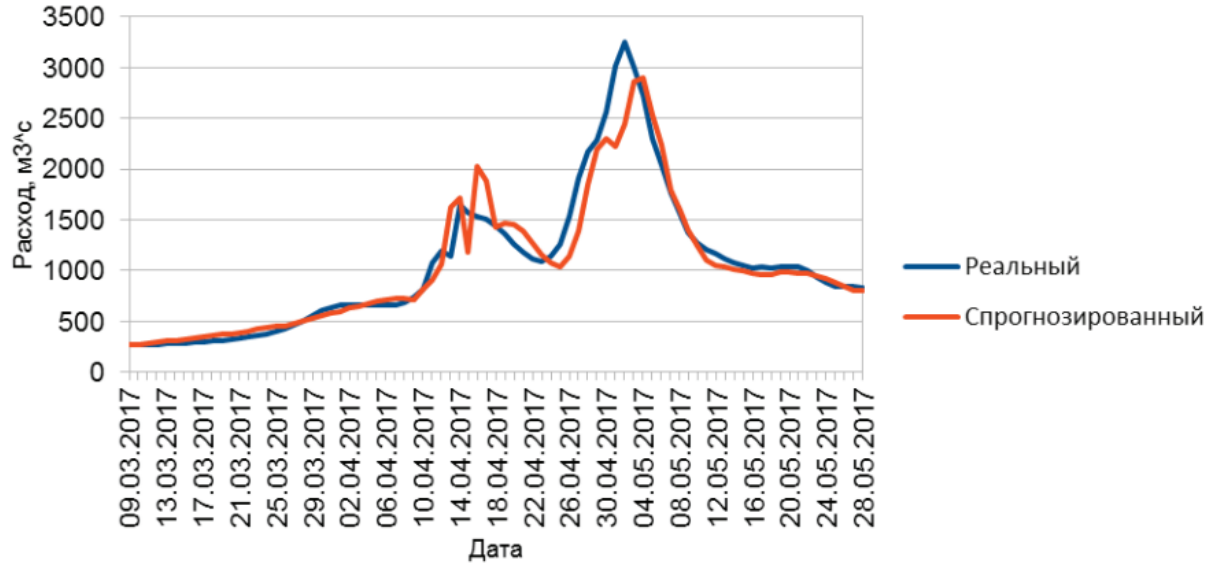

Рисунок 4.6 – Краткосрочный прогноз расхода воды на послезавтра в период весеннего ледохода 2017 года

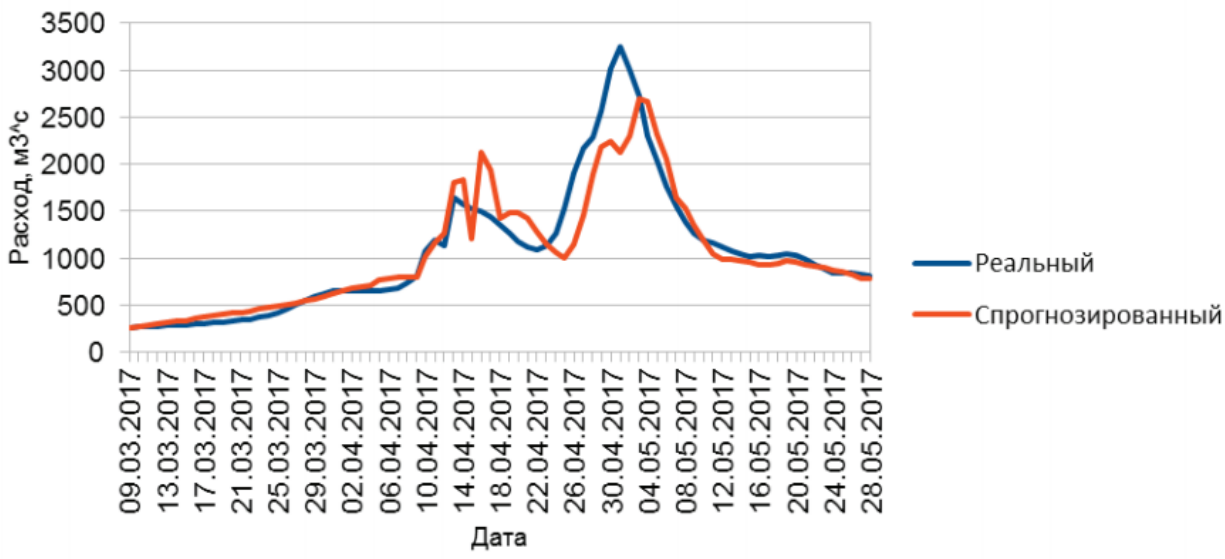

Рисунок 4.7 – Краткосрочный прогноз расхода воды на 3 дня вперёд в период весеннего ледохода 2017 года

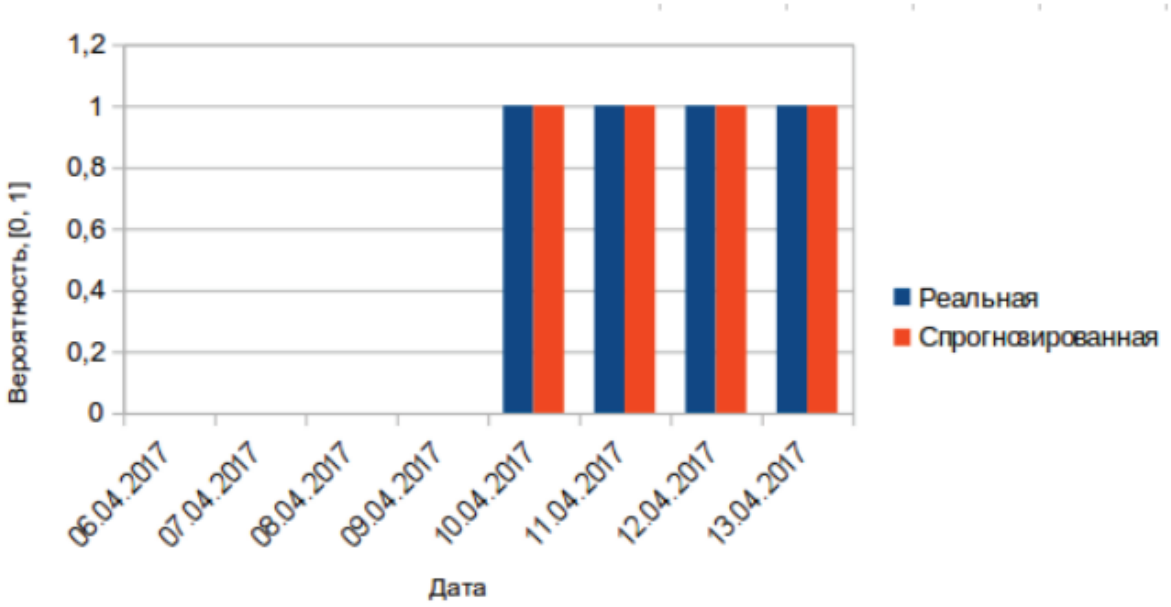

Рисунок 4.8 – Краткосрочный прогноз вероятности вскрытия льда в следующие 3 дня в период весеннего ледохода 2017 года

Основное же преимущество представленных моделей заключается в том, что при существенных изменениях русла реки или режима ледохода, эти ИНС можно переобучить на новых данных, с помощью созданного ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур. Это является важным отличием от используемых в данное время моделей, которые приходится разрабатывать заново. Данный подход, в перспективе, потребует меньше времени и ресурсов для адаптации ИС

мониторинга и прогнозирования сложно формализуемых динамических объектов, в т.ч. наводнений, при естественных и искусственных изменениях характеристик объекта прогнозирования.

На 6 этапе методики применения разработанного ПО созданные модели ИНС использовались не автономно, а были внедрены в сам ИС «ПРОСТОР» как одна из подсистем прогнозирования [10]. Схема ИС простор представлена на рисунке 4.9.

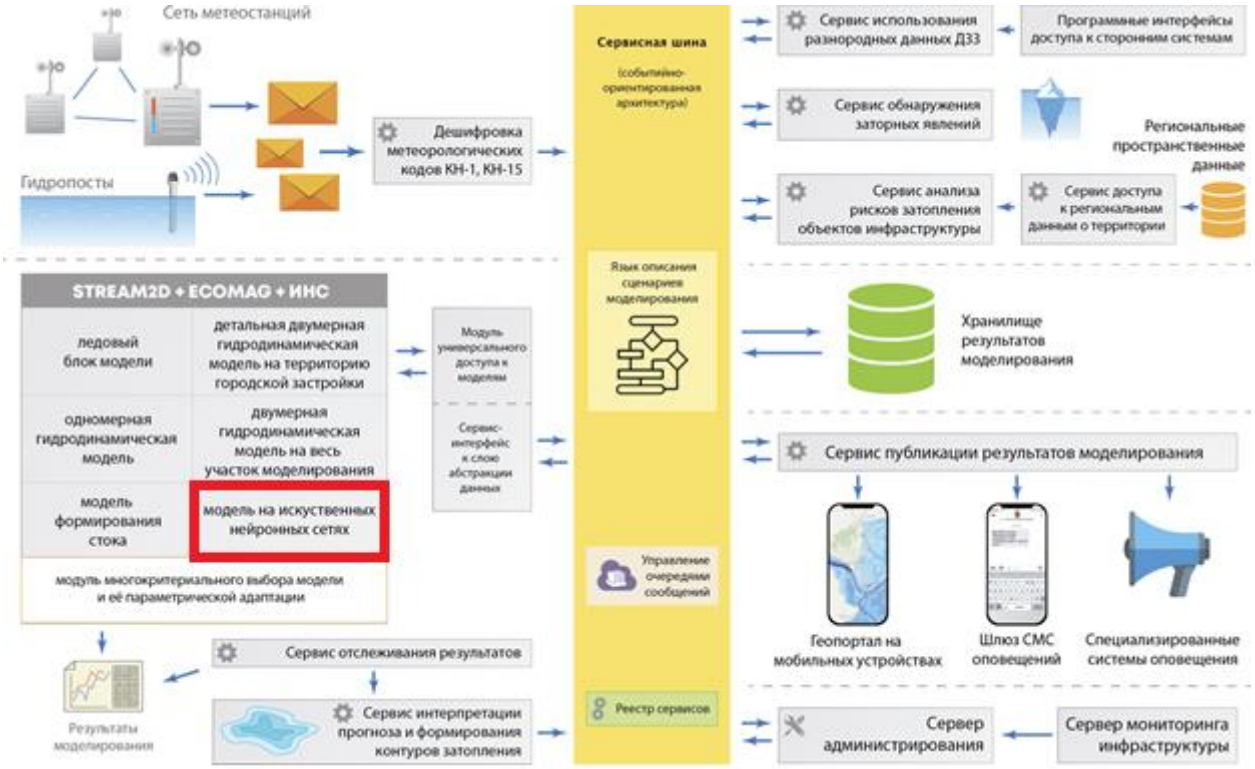

Рисунок 4.9 – Детальная схема ИС «ПРОСТОР» с указанием местоположения разработанной модели ИНС в общей структуре

Как видно из схемы, ИС была разработан на основе СОА. Поэтому сгенерированные для созданных моделей ИНС сервисные оболочки практически без изменений были использованы в ней.

Тестирование моделей ИНС в составе ИС проводилось во время весеннего ледохода 2018 года. Во время этого тестирования модели показали свою точность и эффективность, а программная оболочка успешно прошла проверку на надёжность. Пример интерфейса ИС «ПРОСТОР» с результатами работы моделей ИНС представлен на рисунке 4.10. Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.83.

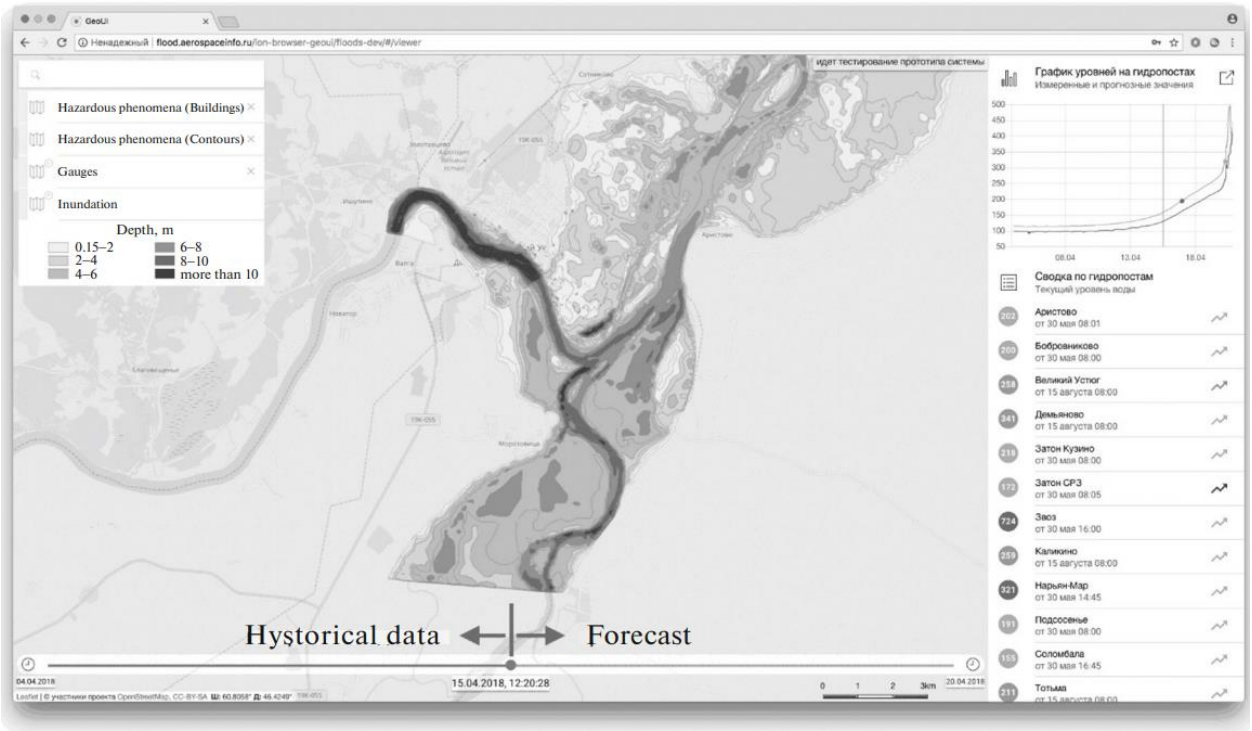

Рисунок 4.10 – пример интерфейса ИС «ПРОСТОР» с результатами краткосрочного прогнозирования уровня воды для конкретного гидропоста (правый верхний угол интерфейса)

4.3 Модель и программа прогнозирования динамики изменения фитомассы растительных сообществ тундры

Проблема оценивания запасов надземной фитомассы растительных сообществ разных природно-климатических зон и прогнозирование их изменений в зависимости от факторов среды является важной практической задачей в области сельского и природоохранных хозяйств. Знания о запасах фитомассы позволяют оценить кормовую емкость пастбищ и рассчитать предельную численность домашних или диких животных для длительного устойчивого природопользования.

С научных позиций информация о фитомассе важна для исследования круговоротов биогенных элементов и установления границ природно-климатических зон в условиях происходящего потепления Арктики. Прогнозирование позволяет оценить величину запасов фитомассы и смещение границ природных зон в перспективе в условиях изменяющегося климата. При решении подобных задач возникает проблема создания моделей, эффективность и

полезность которых во многом зависит от применяемых математических методов их построения. Следует также учитывать достоверность, точность, объективность, нечеткость, однородность или неоднородность, объем и другие характеристики анализируемых данных и информации.

Решаемая в данном исследовании задача исходно определена как слабо формализованная. Такая формулировка связана с тем, что для природных, общественных и техносферных систем характерны [20]:

- взаимозависимость свойств и организации;
- бесперспективность применения линейных аппроксимаций;
- стохастическое (недетерминированное) поведение;
- не выводимость свойств системы как целого из свойств ее элементов;
- не воспроизводимость поведения по начальным данным;
- неопределимость и логическая недоказуемость законов причинности;
- самоподобие;
- саморазвитие.

Для задач оценивания динамики фитомассы растительных сообществ тундры характерны все перечисленные особенностями с присущими внутренней неопределенностью, эмпирическим отбором факторов, ошибками их определения, неполнотой учитываемых внешних возмущающих воздействий, наличием локальных микроклиматических и почвенных условий произрастания сообществ и многих других причин. Таким образом, можно констатировать, что для такого слабо формализованного природного объекта построить адекватную модель методами детерминированной математики невозможно.

При это имеются исходные эмпирические данные, которые можно использовать для разработки моделей прогнозирования. Это сведения о сезонной и межгодовой динамике фитомассы видов и сообществ растительного покрова. Для оценивания сезонной и межгодовой динамики продуктивности растительных сообществ часто используются различные вегетационные индексы, такие, как:

NDVI (Normalized Difference Vegetation Index), улучшенный вегетационный индекс EVI (Enhanced Vegetation Index) и листовой индекс LAI (Leaf Area Index) [26].

Для решения задачи прогнозирования состояния описанного объекта было создано несколько моделей, обеспечивающих прогнозирование запасов надземной фитомассы (максимальной величины фитомассы в период вегетации) растительного сообщества тундры как сложной слабо формализованной системы, в зависимости от погодно-климатических условий. При этом использован прием перевода статистически слабо представленного материала наземных наблюдений в область более точных и статистически достоверных по объему данных космических снимков одного и того же района исследований. Модели создавались на основе данных о растительных группировках острова Колгуев. Использовались архивные данные об NDVI, результаты наземных измерений фитомассы, а также обобщенные данные о связи NDVI с фитомассой для биома тундры [114]. Использование NDVI как промежуточного параметра связано с невозможностью построения прямой связи между входными данными и прогнозируемыми параметрами из-за объективной невыполнимости условий создания репрезентативной выборки (как по объему, так и по однородности наземных исследований).

При решении задачи использовались три независимые модели: классическая регрессионная модель, нечетко-возможностный подход и ИНС, синтезированная с помощью ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур.

В качестве конкретного объекта моделирования межгодовой динамики запасов фитомассы выбран арктический остров Колгуев (подзона типичных тундр), расположенный в Баренцевом море. На острове в 2005-2013 гг. проводились геоботанические исследования [16], благодаря которым был накоплен материал по картированию зональных и интразональных растительных сообществ с их геоботаническим описанием на площадках, спектральными характеристиками и точной географической привязкой. Именно эти данные и использовались на 1 этапе методики использования разработанного ПО.

На 2-5 этапах методики использования разработанного ПО была синтезирована модель ИНС архитектуры МПР решающая задачу моделирования взаимосвязи NDVI растительного сообщества с факторами среды [18]. Оценка точности модели МПР проводилась по среднеквадратическому отклонению рассчитанных на сети и полученных по спутниковым данным значениям NDVI. Обучение происходило на 1000 эпохах, в качестве функции потерь (regression loss functions) применялась функция logcosh (logarithm of the hyperbolic cosine of the prediction error), в качестве алгоритма оптимизации (optimizer) использовался алгоритм adam (adaptive moment estimation). В состав вариантов обучения входило:

- обучение по полному набору аргументов для всего сезона вегетации и для укороченного интервала (от начала вегетации до момента определения NDVI);
- обучение по расширенному набору с добавлением в состав аргументов значение NDVI предыдущего года;
- обучение по ограниченному набору с удалением одного из аргументов.

Результаты сравнения показали, что наивысшей точностью обладает модель ИНС, обученная на полном наборе аргументов. Именно эта модель была принята в качестве основной при решении задачи прогнозирования динамики NDVI. Выбранная модель ИНС имела следующую конфигурацию:

- входной слой сети состоит из 6 нейронов, соответствующих входному вектору;
- скрытый слой был создан один и состоял из 19 нейронов;
- функцией активации для скрытых слоёв была выбрана linear;
- выходной слой состоял из 3 нейронов с функцией активации selu (scaled exponential linear units).

Поскольку результат работы модели ИНС для данной задачи сложно представить аналитически, оценка адекватности моделирования проводилась по корреляционным полям (рисунок 4.11). На рисунке 4.11б отчетливо просматривается систематическая погрешность, причину которой можно объяснить малыми объемами как обучающей, так и тестовой выборок.

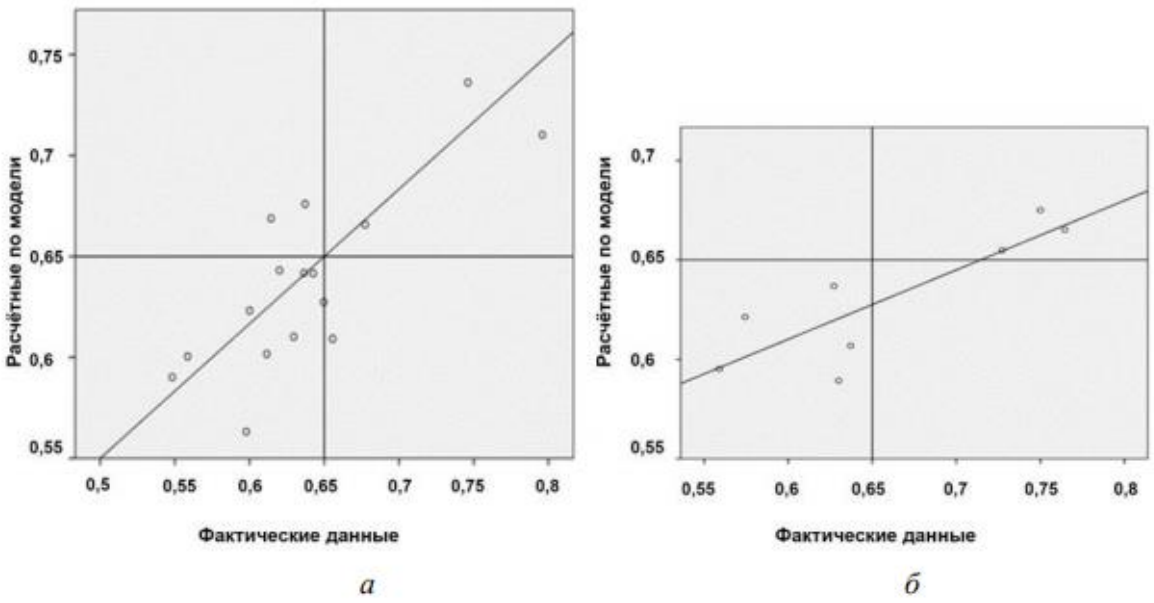

Рисунок 4.11 – Корреляционные поля точек: а – по обучающей выборке за 16 лет, б – по тестовой выборке за 8 лет

Результаты экспериментов показали, что ошибка оценивания NDVI по модели ИНС, обученной на данных, усредненных для сезона вегетации, составляет около 5%. Результаты тестирования модели ИНС по величине ошибки оказались выше ожидаемых для исходных рядов данных длиной 25 лет, поскольку считается [67], что такой объем информации может оказаться недостаточным для эффективного обучения модели ИНС ПР. При обучении модели ИНС по укороченному ряду данных ошибка оценивания возрастает почти в 1.5 раза. Учет предыстории путем включения в состав аргументов NDVI предыдущего года почти не влияет на величину ошибки.

Эксперименты с обучением на ограниченных выборках проводились для оценки влияния внешних факторов на адекватность модели. Результаты показали, что наибольшую роль в изменениях NDVI играет сумма температур. При исключении этого фактора ошибка увеличивается с 5 до 8.3%. Затем идет сумма осадков – увеличение ошибки до 7.6%, период вегетации и время начала вегетации – увеличение ошибки до 6.6%. При исключении ветра и облачности из состава обучающей выборки ошибка не увеличивается, что можно объяснить слабой межгодовой изменчивостью этих факторов. Такое распределение факторов

соответствует ранжированию их значимости согласно имеющимся исследованиям [11, 44].

Таким образом, использование модели МПР при решении данной задачи подтвердило принципиальную возможность их применения для решения слабо формализованной задачи прогнозирования динамики изменения фитомассы растительных сообществ тундры. Но, поскольку МПР относится к моделям обучающимся с учителем, точность результатов прогнозирования целиком зависит от размера и релевантности обучающей выборки. Поскольку в данном случае обучающих данных было слишком мало, прогнозирование показателей многих аномальных лет оказалось крайне неточным. Однако, данная модель в большинстве случаев всё же рассчитывала результат более близкий к реальному, по сравнению с другими двумя моделями. Результаты прогнозирования реальных данных представлены на рисунке 4.12.

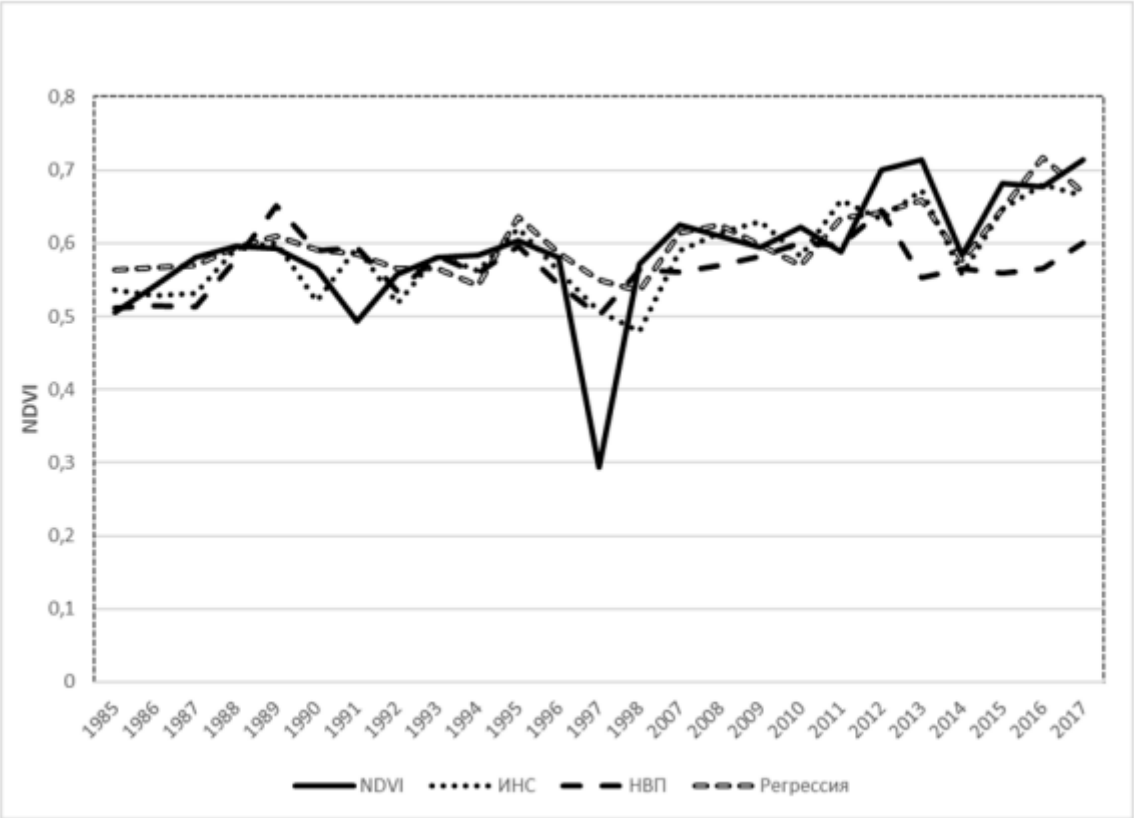

Рисунок 4.12 – Графики фактических и рассчитанных по моделям значений межгодовой динамики NDVI. Кривая NDVI – фактические значения индекса; кривая ИНС – модель, построенная с помощью разработанного ПО; кривая НВП – модель, построенная на основе нечетко-возможностного подхода; кривая Регрессия – линейная регрессионная модель.

Поскольку рассматриваемая работа носила исследовательский характер и на данный момент пока нет практического прикладного внедрения, потребность в разработанной программной оболочке не возникла. Поэтому на 6 этапе созданной методики, сгенерированное программное обеспечение на базе синтезированной модели ИНС ПР применялось автономно для решения поставленной научно-исследовательской задачи. С другой стороны, данная апробация продемонстрировала возможность применения разработанного ПО не только для решения прикладных, но и исследовательских задач.

Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.5.

4.4 Модель и программа распознавания и подсчёта числа северных оленей по аэрофотоснимкам

Используемая в настоящее время методика учетов численности диких северных оленей тундровых популяций (таймырской, якутских популяций, оленей Чукотки, а также мигрирующих стад северных оленей Канады и Аляски) основывается на экологической особенности видов, состоящей в том, что олени в жаркую погоду во время вылета кровососущих насекомых собираются в многотысячные скопления на ограниченной территории в северной части летнего ареала (подзоны арктических и типичных тундр) [12, 14]. Стада в скоплениях фотографируются и количество животные в них подсчитываются вручную «по головам».

Преимущество такого подхода перед чисто аппроксимационными методами оценки численности [25] состоит в существенно большей точности результатов, поскольку подавляющее количество животных в популяции (до 90%) подсчитываются напрямую на фотографиях стад и лишь небольшое число оценивается путем аппроксимации по территории. Однако ручная обработка результатов съемок для крупных популяций занимает около трех месяцев, в то время как для экологически обусловленного управления динамикой численности популяции, для неистощительного использования биологических ресурсов вида и определения норм промыслового изъятия оленей требуется, чтобы данные о численности оленей имелись бы во второй половине августа, то есть через 10-15 дней после окончания авиаучета. В связи с этим особую актуальность приобретает задача автоматизации процедур обработки аэрофотоснимков для сокращения времени получения результатов авиаучетов.

В качестве модели распознавания при решение данной задачи была выбрана СНС (конкретно архитектура MRCNN) [19]. Поскольку данных для обучения в выбранной прикладной области оказалось недостаточно, было предложено использовать подход трансферного обучения на массиве данных, включающем изображение других животных. Это было необходимо для того, чтобы MRCNN научилась распознавать животных как класс объектов. После этого, для решения

конкретной прикладной задачи, сеть дообучалось на специфическом массиве данных, включающем аэрофотоснимки стад оленей. Специфика задачи состоит в том, что фотографирование стад производится с разного расстояния, на разных ландшафтах, при различных условиях освещённости, сами животные на снимках имеют различную окраску и могут находиться под разными углами к камере, а также могут перекрывать друг друга. Эти особенности аэрофотоснимков создают дополнительные сложности при решении задачи распознавания оленей. Поэтому представленная задача является нетривиальной и применение MRCNN обученной только на распространённых массивах данных невозможно.

На 1 этапе методики использования разработанного ПО основным массивом обучения для MRCNN стал массив изображений MS COCO dataset (Microsoft Common Objects in Context). Данный массив является на сегодняшний день одним из самых крупномасштабных наборов данных, который используется для обучения моделей МО решению задач обнаружения и сегментации. Набор данных состоит из 328 тысяч изображений. Все изображения размечены и сформированы в обучающие выборки. Поэтому использование данного массива для базового обучения модели MRCNN позволяет задать для неё все основные концепции различных классов объектов, в том числе и животных. Однако, изображения оленей не входят в MS COCO и модель MRCNN по умолчанию не способна отличить их от ряда других животных (овец, газелей, коров, лошадей). Это также было причиной того, что для решения задачи распознавания северных оленей модель MRCNN должна была быть дообучена с использованием массивов аэрофотоснимков с изображением этих животных. Пример работы недообученной модели MRCNN представлен на рисунке 4.13.

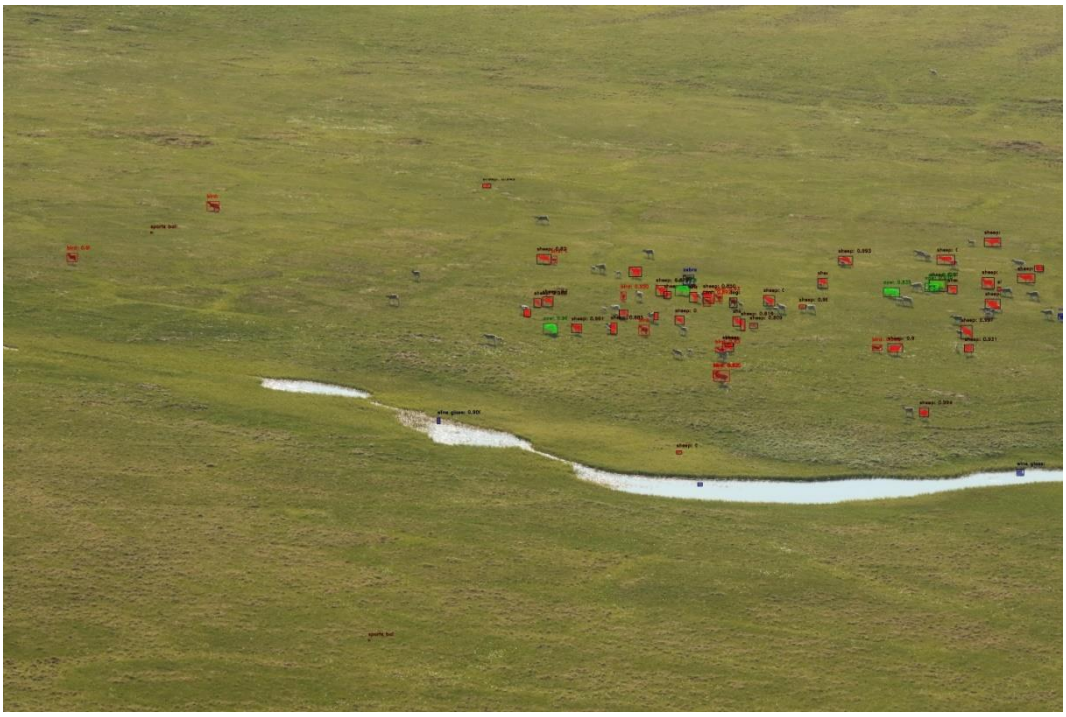

Рисунок 4.13 – Распознавание оленей с помощью модели MRCNN обученной только на MS COCO dataset

Для дообучения модели MRCNN был подготовлен входной массив данных, содержащий обучающую выборку из 100 аэрофотоснимки стад и тестовую выборку из 30 исходных снимков стад.

На 2-5 этапах методики использования разработанного ПО модель была обучена на 20 эпохах, с 60 шагами обучения в эпохе, со скоростью обучения 0,0058 и с порогом пропуска обнаружения 0.7. Обученная модель для той же фотографии корректно распознала 70 оленей из 93 (количество распознанных оленей составило 75% от общего числа) и при этом не совершила ни одной ошибки второго рода (рисунок 4.14).

Рисунок 4.14 – Результат распознавания модели MRCNN дообученной на аэрофотоснимках

На всём тестовом массиве данных обученная модель корректно распознавала в среднем 82% оленей. Точность распознавания оленей может быть повышена путем дообучения модели MRCNN на расширенном наборе обучающего массива данных. Дополнительные примеры работы сгенерированной и обученной модели MRCNN можно увидеть на рисунках 4.15 – 4.17.

Рисунок 4.15 – Пример распознавания высоко детализированного снимка

На рисунок 4.15 видно, что модель MRCNN совершила 4 ошибки – 2 оленя были распознаны дважды и 2 теленка не были распознаны. Всего из 41 оленя корректно было распознано 35 особей. Ошибка подсчета составили всего 15%, что является высоким результатом для автоматических систем распознавания животных в естественных условиях. Причиной этого являются высокое качество снимка и высокое разрешение самих оленей, а также сильная контрастность фона с объектами распознавания.

Рисунок 4.16 – Пример распознавания снимков стад высокой плотности

На рисунке 4.16 видно, что модель MRCNN хорошо работает с большими стадами. По снимку заметно, что несмотря на то, что на снимке есть олени на разном удалении от фотографа, при увеличении расстояния до них точность распознавания падает незначительно. Таким образом, сеть не привязана к конкретному масштабу объектов и способна работать с искаженной перспективой. Ошибка распознавания на данном снимке составила около 3%

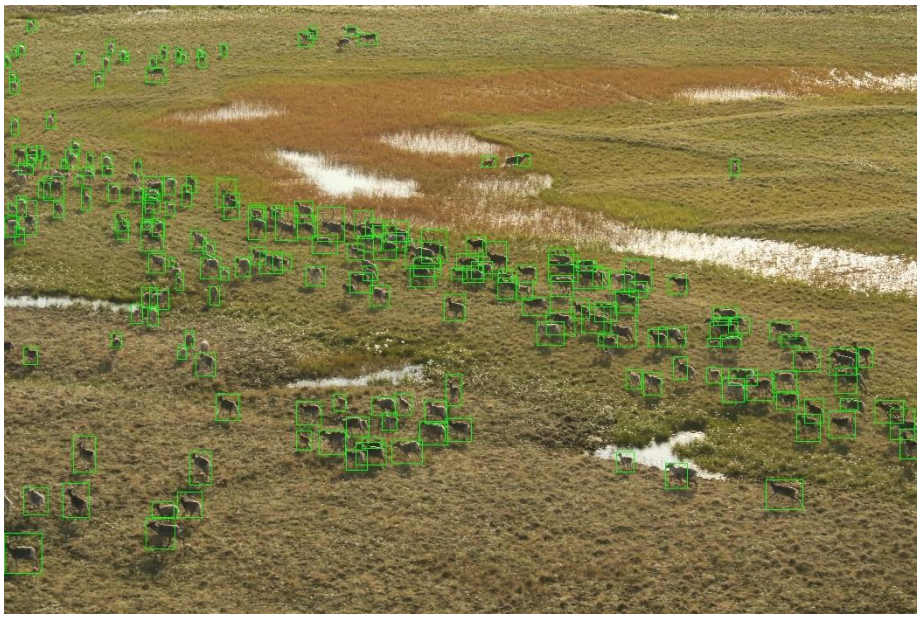

Рисунок 4.17 – Пример распознавания зашумлённого снимка

На рисунке 4.17 видно, что модель MRCNN может работать со снимками, которые зашумлены фоновыми объектами – лужайками, озерами, бугорками, полигонами и т.п. Также, заметно, что модель продолжает хорошо работать и со стадами, в которых олени собираются в очень плотные группы. Точность распознавания на данном снимке составила около 83%.

Отдельно стоит отметить, что данная разработка имеет прикладной характер и уже начинает применяться на практике профильными организациями. На 6 этапе созданной методики на основе разработанной модели было создано автономное ПО. Оно предполагает серверное развёртывание и для работы с ним отдельно создан веб-интерфейс. К текущему времени ПО введено в ограниченную эксплуатацию. Пример интерфейса представлен на рисунке 4.18.

Рисунок 4.18 – Пример веб-интерфейса разработанного ПО автоматического распознавания и подсчёта количества северных оленей

Для работы с созданным ПО автоматического распознавания и подсчёта количества северных оленей пользователь должен загрузить в программу через веб-интерфейс снимки стад в форматах JPEG (.jpg) или GIF (.gif) и запустить расчёт. После того, как обработка снимка будет завершена, пользователю будет предоставлена информация о количестве оленей на снимке и возможность скачать сам размеченный снимок.

Веб-интерфейс ПО содержит набор оконных форм, обеспечивающих:

- загрузку аэрофотоснимков с компьютера пользователя для их обработки,
- запуск модели СНС распознавания и подсчета оленей на снимках,
- представление результатов работы модели СНС с отображением на экране снимка, на котором помечены распознанные изображения оленей, и общее число подсчитанных животных,
- скачивание результатов на компьютер пользователя.

После просмотра скачанного размеченного снимка, если пользователя не удовлетворит точность работы ПО, он может продолжить дальнейшую обработку

снимка вручную в любом графическом редакторе, поддерживающем расширение файла с изображением .jpg.

На данный момент разработанное ПО автоматического распознавания и подсчёта количества северных оленей находится в ограниченной эксплуатации по той причине, что работа даже обученной модели MRCNN требует больших вычислительных ресурсов, которые на сегодняшний день не может предоставить аппаратная часть имеющегося сервера. Поэтому обработка даже одного аэрофотоснимка может занимать более 5 минут. Из-за этих ограничений сейчас с программой может работать только один пользователь в один момент времени.

В дальнейшем, для ввода данного ПО в режим штатной эксплуатации, потребуется его перенос на высокопроизводительные сервера. Коммерческое использование разработанного ПО подразумевает параллельное работу нескольких пользователей, что может существенно нагрузить систему. Несмотря на заложенную в архитектуру ПО параллельность вычислений (посредством СОА), при последующим его совершенствовании потребуется также провести дополнительные исследования и стресс-тесты, чтобы определить конкретные требования к аппаратному обеспечению, необходимому для функционирования рассматриваемого ПО.

Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.67.

4.5 Выводы по разделу 4

1. В главе рассмотрены примеры практического использования разработанного ПО комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур. На данных примерах была продемонстрирована не только концептуальная универсальность, но и программно-инженерная реализуемость используемого подхода к комплексной автоматизации процессов генерации и обучения моделей ИНС ПР.

2. Была разработана методика использования разработанного программного обеспечения комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур, с помощью которой были решены конкретные задач из принципиально разных областей. В автоматизированном режиме были синтезированы различные модели ИНС ПР, каждая из которых отличалась как процессами генерации и обучения, так и процессом их интеграции. При этом все эти модели генерировались и обучались с помощью одного и того же ПО, на основе разработанного алгоритма УПГ для комплексной автоматизации процесса синтеза моделей ИНС ПР различных архитектур. Каждая из созданных моделей показала высокую точность работы, а две из них были внедрены в прикладные программные комплексы и в настоящее время эффективно используются на практике.

3. Результаты экспериментальных исследований подтвердили повышение оперативности создания моделей ИНС ПР, достигаемые при использовании разработанного ПО. Поскольку участие человека при генерации и обучении моделей ИНС ПР было исключено, данные процессы были полностью автоматизированы, что, по сравнению с существующими подходами, привело к сокращению общего времени разработки моделей ИНС ПР для решения конкретных прикладных задач. Также были существенно снижены требования к специальной квалификации пользователей за счет автоматизированного режима работы системы.

При создании моделей прогнозирования уровня воды, за счёт использования разработанного ПО, было снижено общее время их разработки с нескольких недель до нескольких суток (по результатам экспериментальной оценки). Практически до нуля было снижено время необходимое для интеграции созданных моделей в ИС «ПРОСТОР», поскольку с помощью разработанного ПО сразу были сгенерированы сервисные оболочки необходимые для подключения к общей сервисной шине ИС. Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.83.

При решении задачи прогнозирования динамики изменения фитомассы растительных сообществ тундры было сокращено время необходимое на прототипирование модели. Был осуществлён автоматизированный перебор гиперпараметров моделей ИНС ПР, благодаря чему рабочий прототип был получен за сутки, вместо нескольких рабочих дней (по результатам экспериментальной оценки). Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.5.

При синтезе моделей ИНС ПР распознавания и подсчёта числа северных оленей по аэрофотоснимкам разработанное ПО позволило больше всего сэкономить время на указанный синтез. Поскольку модели СНС в процессе обучения являются одними из наиболее требовательных к затратам вычислительных ресурсов, предложенный подход к комплексной автоматизации процесса этого обучения позволил созданному ПО надежно функционировать круглые сутки без остановки. При этом время на создание модели было сокращено с нескольких месяцев до нескольких суток (по результатам экспериментальной оценки). Показатель степени автоматизации первичных частей процесса создания, обучения и использования моделей ИНС ПР различных архитектур при решении данной задачи составил D=0.67.

4. В главе наглядно продемонстрировано конструктивное использование СОА при генерации программных оболочек для созданных моделей ИНС ПР. Результаты интеграции в ИС «ПРОСТОР» наиболее ярко показали необходимость и оправданность выбранного подхода для упрощения процесса интеграции созданных моделей ИНС ПР в стороннее ПО.

ЗАКЛЮЧЕНИЕ

В представленной диссертации сформулирована и решена новая научно-техническая задача комплексной автоматизации процессов генерации, обучения и использования моделей ИНС ПР различных архитектур. Решённая задача имеет важное значение для совершенствования методов и алгоритмов автоматизированного машинного обучения, используемых для упрощения и ускорения процесса разработки моделей ИНС ПР, что в свою очередь приводит к удешевлению и ускорению разработки программных комплексов на базе моделей ИНС, решающих прикладные задачи в различных областях человеческой деятельности. Результаты апробации разработанного ПО подтверждают достижение цели диссертационной работы – повышения степени автоматизации процесса создания, обучения и использования моделей ИНС ПР различных архитектур за счёт автоматизации их разработки, базирующейся на целенаправленном выборе архитектур и гиперпараметров создаваемых моделей ИНС ПР с использованием алгоритма УПГ, а также автоматической генерации программных оболочек для синтезированных моделей ИНС ПР. На примере решения трёх различных прикладных задач (рассмотренных в главе 4) было показано, что сокращение затрат времени было достигнуто за счёт повышения уровня автоматизации разработки и внедрения моделей ИНС ПР. Поэтому конкретные числовые показатели этого сокращения индивидуальны для каждой задачи. При создании модели распознавания и подсчёта числа северных оленей на аэрофотоснимках использование созданного ПО позволило сэкономить больше месяца работы (по результатам экспериментальной оценки). С другой стороны процесс интеграции созданной модели в ИС «ПРОСТОР» был полностью автоматизирован, что снизило затраты времени на этот процесс до нуля. Поскольку прямых аналогов создаваемой системы не существует, прямое сравнение было невозможно провести.

Основные научные результаты, составляющие **итоги** исследования:

1. Разработан алгоритм УПГ для комплексной автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур. Предложенный алгоритм обеспечивает унифицированный подход к генерации и обучению моделей ИНС ПР различных архитектур, который отличается от существующих подходов к решению рассматриваемой задачи синтеза тем, что при его реализации отсутствует необходимость внесения модификаций в сам алгоритм при работе с каждой конкретной архитектурой ИНС ПР. Эта возможность достигнута за счёт внесения модификаций, учитывающих обобщённые особенности подхода обучения с учителем к ИНС ПР.

2. Разработана архитектура и программная система автоматизации процессов генерации и обучения моделей ИНС ПР различных архитектур, отличающаяся модульной расширяемой структурой, что позволяет использовать её для решения широкого спектра практических задач. Программная система в составе ПО была апробирована на ряде прикладных задач и показала свою вычислительную эффективность и удобство при её использовании непрограммирующими пользователями. Благодаря реализации концепции No Code в разработанном ПО, конечный пользователь способен создавать полностью работоспособные программные реализации моделей ИНС ПР различных архитектур, без необходимости самостоятельно писать программный код.

3. Разработана архитектура и программная система автоматической генерации программных оболочек для созданных моделей, с учётом парадигмы СОА. Данная программная система позволяет ускорить и упростить интеграцию созданных моделей в стороннее ПО. Благодаря реализации концепции No Code в разработанной программной системе, конечный пользователь способен создавать интегрируемые программные модули для созданных моделей ИНС ПР реализующих различные архитектуры, без необходимости самостоятельно писать программный код.

Перспектив дальнейшей разработки темы включают вопрос развития пользовательского программного интерфейса, который бы позволил работать с созданным ПО неспециалистам в области ГО (в данный момент реализован лишь

рабочий прототип интерфейса). Кроме этого, требуется ввести разработанное ПО в ограниченную эксплуатацию, для оценки вычислительных ресурсов, необходимых для его штатной работы, а также для проведения стресс-тестов и тестов производительности. После проведения всех необходимых тестов и оценки разработанного ПО, планируется его дальнейшая полноценная эксплуатация, с последующим расширением набора архитектур ИНС ПР, комплексную автоматизацию процессов генерации, обучения и использования которых он сможет обеспечить. Ограничения разработанного ПО, на сегодняшний день, связаны с небольшим количеством реализованных в ней архитектур ИНС ПР. Несмотря на то, что на данный момент разработанное ПО обеспечивает решение наиболее востребованных на практике прикладных задач, при дальнейшей разработке необходимо будет увеличить количество доступных архитектур ИНС ПР.

Полученные результаты соответствуют специальности 2.3.5 – «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей».

Разработанное алгоритмическое и программное обеспечение реализовано в ряде организаций: Государственный природный биосферный заповедник «Таймырский», «Санкт-Петербургский государственный технологический институт (технический университет)» СПбГТИ (ТУ), в СПИИРАН и СЗЦППО входящих в СПб ФИЦ РАН.

СПИСОК СОКРАЩЕНИЙ

ГА	генетический алгоритм
ГО	глубокое обучение
ИВ	интернет вещей
ИИ	искусственный интеллект
ИНС	искусственная нейронная сеть
ИС	информационная система
МО	машинное обучение
МПР	многослойный перцептрон Румельхарта
ПО	программное обеспечение
ПР	прямое распространение
СНС	свёрточная нейронная сеть
СОА	сервис-ориентированная архитектура
УПГ	унифицированный подбор гиперпараметров
AutoML	automated machine learning
CUDA	compute unified device architecture
cuDNN	CUDA deep neural network
FRCNN	faster regions with convolution neural networks
MRCNN	mask regions with convolution neural networks
NDVI	normalized difference vegetation index

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бузин В.А. Опасные гидрологические явления. Учебное пособие / В.А. Бузин. – СПб.: изд. РГГМУ, 2008. - 228 с.
2. Визильтер Ю. В. Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей / Ю. В. Визильтер, В. С. Горбацевич, С. Ю. Желтов // Компьютерная оптика. – 2019. – №5. – С. 886–900.
3. Голев А.В. Система мониторинга и обеспечения безопасхности природных ресурсов / Материалы 28-й Международной конференции «Проблемы управления безопасностью сложных систем» (ПУБСС'2020, Москва). – М.: ИПУ РАН, 2020. – С. 294 - 297.
4. ГОСТ 23004-78 Механизация и автоматизация технологических процессов в машиностроении и приборостроении: утв. и введ. в действие постановлением Государственного комитета стандартов Совета Министров СССР от 9 марта 1978 г. №634, дата введения с 01.01.1979. М.: Издательство стандартов, 1978.
5. ГОСТ Р 57412-2017 Компьютерные модели в процессах разработки, производства и эксплуатации изделий: утв. и введ. в действие приказом Федерального агентства по техническому регулированию и метрологии от 10 марта 2017 г. №110-ст, дата введения 2017-07-01. М.: Стандартинформ, 2018.
6. Дейтел П. Python. Искусственный интеллект, большие данные и облачные вычисления / П. Дейтел, Х. Дейтел. – СПб.: Издательский дом «Питер», 2020. – 864 с.
7. Дэвенпорт Т. Внедрение искусственного интеллекта в бизнес-практику: Преимущества и сложности / Т. Дэвенпорт. – М.: Альпина Паблишер, 2020. – 320 с.
8. Ефимов В.В. Нейроподобные сети в бортовых информационно-управляющих комплексах космических аппаратов. – СПб.: ВИКА им. А.Ф. Можайского, 1996. – 113 с.
9. Ефимов В.В., Яковкин В.А. Нейросетевое формирование оптимальной последовательности обслуживания объектов // IX Всероссийская НТК

“Экстремальная робототехника” 14-16 апреля 1998 г. - СПб.: ЦНИИ робототехники и технической кибернетики. - С. 326-331.

10. Зеленцов В.А., Алабян А.М., Крыленко И.Н., Пиманов И.Ю., Пономаренко М.Р., Потрясаев С.А., Семёнов А.Е., Соболевский В.А., Соколов Б.В., Юсупов Р.М. Модельно-ориентированная система оперативного прогнозирования речных наводнений // Вестник Российской академии наук. 2019. Т. 89. № 8. С. 831-843. DOI: 10.31857/S0869-5873898831-843.
11. Зуев В.В. Климатически обусловленные изменения растительного покрова тайги и тундры Западной Сибири в 1982-2015 гг. по данным спутниковых наблюдений / В.В. Зуев, Е.М. Короткова, А.В. Павлинский // Исследование Земли из космоса. – 2019. – №6. – С. 66-76.
12. Зырянов В.А. Экологические основы учета численности промысловых животных в тундровой зоне Таймыра / В.А. Зырянов, Б.М. Павлов, Г.Д. Якушкин // Проблемы охотничьего хозяйства Красноярского края. – 1971. – С. 70-72.
13. Колмогоров А. О представлении непрерывных функций нескольких переменных суперпозициями непрерывных функций меньшего числа переменных / А. Колмогоров // Известия АН СССР. – 1956. – Том 108. – с. 179—182.
14. Колпащиков Л.А. Методика авиаучета и определения норм опромышления таймырской популяции диких северных оленей: методические рекомендации / Л.А. Колпащиков, Б.М. Павлов, В.В. Михайлов. – СПб., 1999. – 25 с.
15. Кушнир Н.В. Искусственные иммунные системы: обзор и современное состояние / Н.В. Кушнир, А.В. Кушнир, Е.В. Анацкая, П.А. Катышева, К.Г. Устинов // Научные труды КубГТУ, №12. – 2015.
16. Лавриненко И. А. Влияние климатических изменений на растительный покров островов Баренцева моря / И. А. Лавриненко, О. В. Лавриненко // Тр. Карельского НЦ РАН. – 2013. – № 6 – С. 4–16.

17. Микони С. В., Соколов Б. В. Юсупов Р. М. Квалиметрия моделей и полимодельных комплексов: монография С. В. Микони, Б. В. Соколов, Р. М. Юсупов. - М.: РАН, 2018. - 314 с.
18. Михайлов В. В., Спесивцев А. В., Соболевский В. А., Карташев Н. К., Лавриненко И. А., Лавриненко О. В., Спесивцев В. А. Многомодельное оценивание динамики фитомассы растительных сообществ тундры на основе спутниковых снимков // Исследование Земли из Космоса, 2021. №2. С. 15-30. DOI: 10.31857/s0205961421020056.
19. Михайлов В.В., Соболевский В. А., Колпащиков Л. А., Соловьев Н. В., Якушев Г. К. Методологические подходы и алгоритмы распознавания и подсчета животных на аэрофотоснимках // Информационно-управляющие системы. 2021. №5 (114). С. 20-32. DOI: 10.31799/1684-8853-2021-5-20-32.
20. Моисеев Н.Н. Математические задачи системного анализа / Н.Н. Моисеев. – М.: Наука, 1981. – 328 с.
21. Программный комплекс для решения задач автоматического машинного обучения машинного обучения FEDOT.CORE [Электронный ресурс]. – 2022. – Режим доступа: https://actcognitive.org/platformy/freymvork-generativnogo-avtomaticheskogo-mashinnogo-obucheniya-fedot .
22. Рассел М. Data mining. Извлечение информации из Facebook, Twitter, LinkedIn, Instagram, GitHub / М. Рассел, М. Классен. – СПб.: Издательский дом «Питер», 2020. – 464 с.
23. Станкевич Л.А. Интеллектуальные роботы и системы управления. Кн. 20. Сборник статей под ред. А.А. Харламова. – М.: Радиотехника, 2006. – 144 с., с. 44-66.
24. Указ Президента РФ от 10.10.2019 №490 «О развитии искусственного интеллекта в Российской Федерации» [Электронный ресурс] // Сайт Администрации Президента. – 2019. – Режим доступа: http://www.kremlin.ru/acts/bank/44731 .
25. Челинцев Н.Г. Математические основы учета животных / Н.Г. Челинцев. – М., 2000. – 431 с.

26. Шевырногов А.П. Сезонная динамика растительности залежных земель красноярской лесостепи по наземным и спутниковым данным / А.П. Шевырногов, Т.И. Письман, Н.А. Кононова, И.Ю. Ботович, А.А. Ларько, Г.С. Высоцкая // Исследование Земли из космоса. – 2018. – №6. – С. 39 – 51.
27. About Keras [Электронный ресурс] // Keras.io. – 2021. – Режим доступа: https://keras.io/about/
28. About Python [Электронный ресурс] // Python.org. – 2021. – Режим доступа: https:// www.python.org/about/ .
29. Ahmed T. Drone Detection by Neural Network Using GLCM and SURF Features / T. Ahmed, T. Rahman, B.B. Roy, J. Uddin // Journal of Information Systems and Telecommunication. – 2021. – Vol. 9, issue 33. – P. 15 – 23.
30. Alizadeh Z. Assessment of machine learning techniques for monthly flow prediction / Z. Alizadeh, J. Yazdi, J. H. Kim, A. K. Al-Shamiri // Water (Switzerland). – 2018. – Vol. 10, issue 11, article № 1676.
31. Anthony R. J. Systems Programming Designing and Developing Distributed Applications Book / R. J. Anthony. – Burlington, USA: Morgan Kaufmann Publishers, 2016. – 548 p.
32. Arsene C. Decision Support System for Water Distribution Systems Based on Neural Networks and Graphs Theory for Leakage Detection / C. Arsene, D. Al-Dabass, B. Gabrys //Expert Systems with Application – 2012. – Vol. 39, issue 18. – P. 13214 – 13224.
33. Ateeq-ur-Rauf Performance assessment of artificial neural networks and support vector regression models for stream flow predictions / Ateeq-ur-Rauf, A. R. Ghumman, S. Ahmad, H. N. Hashmi // Environmental Monitoring and Assessment. – 2018. – Vol. 190, issue 12, article № 704.
34. AutoKeras 1.0 Tutorial Overview [Электронный ресурс] // AutoKeras. – 2021. – Режим доступа: https://autokeras.com/tutorial/overview/ .
35. AutoML (Automated Machine Learning) Explained [Электронный ресурс] // MathWorks. – 2022. – Режим доступа: https://www.mathworks.com/discovery/automl.html .

36. AutoML [Электронный ресурс] // AutoML.org. – 2022. – Режим доступа: https://www.automl.org/automl/ .
37. AutoML workshop @ ICML'14 [Электронный ресурс]. – 2022. - Режим доступа: https://sites.google.com/site/automlwsicml14/home .
38. Auto-sklearn documentation [Электронный ресурс] // Github. – 2021. – Режим доступа: https://automl.github.io/auto-sklearn/master/ .
39. Aydin Temel F. A multilayer perceptron-based prediction of ammonium adsorption on zeolite from landfill leachate: Batch and column studies / F. Aydin Temel, O. Cagcag Yolcu, A. Kuleyin // Journal of Hazardous Materials. – 2021. – Vol. 410, article №124670.
40. Baghdasaryan V. Comparison of econometric and deep learning approaches for credit default classification / V. Baghdasaryan, H. Davtyan, A. Grigoryan, K. Khachatryan // Strategic Change. – 2021. – Vol. 30, issue 3. – P. 257 – 268.
41. Baker B. Designing neural network architectures using reinforcement learning / B. Baker, O. Gupta, N. Naik, R. Raskar // 5th International Conference on Learning Representations, ICLR 2017. – 2017.
42. Barricelli N. A. Numerical testing of evolution theories - Part II preliminary tests of performance. symbiogenesis and terrestrial life / N. A. Barricelli // Acta Biotheoretica. – 1963. – Vol. 16, issue 3-4. – P. 99-126.
43. Bello I. Neural optimizer search with Reinforcement learning / I. Bello, B. Zoph, V. Vasudevan, Q. V. Le // 34th International Conference on Machine Learning, ICML 2017. – 2017. – Vol. 1. – P. 712-721.
44. Bhatt U.S. Recent declines in warming and vegetation greening trends over pan-Arctic tundra / U.S. Bhatt, D.A. Walker, M.K. Raynolds, P.A. Bieniek, H.E. Epstein, J.C. Comiso, J.E. Pinzon, C.J. Tucker, I.V. Polyakov // Remote Sensing. – 2018. – Vol. 4. – P. 4229-4254.
45. Blokdyk G. Low-code development platforms A Complete Guide / G. Blokdyk. – Plano, USA: 5STARCooks, 2021. – 311 p.

46. Bozinovski S. The influence of pattern similarity and transfer learning upon the training of a base perceptron B2 / S. Bozinovski, F. Ante // Proceedings of the Symposium Informatica, (3-121-5). – 1976.
47. Bressem K.K. Deep learning for detection of radiographic sacroiliitis: achieving expert-level performance / K.K. Bressem, J.L. Vahldiek, L. Adams, S.M. Niehues, H. Haibel, V.R. Rodriguez, M. Torgutalp, M. Protopopov, F. Proft, J. Rademacher, J. Sieper, M. Rudwaleit // Arthritis Research and Therapy. – 2021. – Vol. 23, issue 1, article №106.
48. Burlina P. MRCNN: A stateful Fast R-CNN: Using temporal consistency in R-CNN for video object localization and classification / P. Burlina // 23rd International Conference on Pattern Recognition, ICPR 2016. – 2016. – P. 3518-3523.
49. Cai H. Efficient architecture search by network transformation / H. Cai, T. Chen, W. Zhang, Y. Yu, J. Wang // 32nd AAAI Conference on Artificial Intelligence, AAAI 2018. – 2018. – P. 2787-2794.
50. Chen G. An unsupervised machine-learning checkpoint-restart algorithm using Gaussian mixtures for particle-in-cell simulations / G. Chen, L. Chacon, T.B. Nguyen // Journal of Computational Physics. – 2021. – Vol. 436, article №110185.
51. Chou J.-S. Automated sensing system for real-time recognition of trucks in river dredging areas using computer vision and convolutional deep learning / J.-S. Chou, C.-H. Liu // Sensors (Switzerland). – 2021. – Vol. 21, issue 2. – P. 1 – 31.
52. Clifton C. Data Mining [Электронный ресурс] / C. Clifton // Encyclopædia Britannica. – 2009. – Режим доступа: https://www.britannica.com/technology/data-mining.
53. COCO Common Objects in Context [Электронный ресурс] // COCODataset. – 2022. – Режим доступа: https://cocodataset.org/#home .
54. CUDA Zone [Электронный ресурс] // NVidia Developer. – 2021. – Режим доступа: https://developer.nvidia.com/cuda-zone .
55. David E. Rumelhart Parallel Distributed Processing / David E. Rumelhart, James L. McClelland. – New York, USA: Random House, 1987. – 570 p.

56. Dijkstra K. CentroidNetV2: A hybrid deep neural network for small-object segmentation and counting / K. Dijkstra, J. van de Loosdrecht, W. A. Atsma, L. R. B. Schomaker, M. A. Wiering // Neurocomputing. – 2021. – Vol. 423. – P. 490-505.
57. Dong J.-D. DPP-Net: Device-Aware Progressive Search for Pareto-Optimal Neural Architectures / J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, M. Sun // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – 2018. – Vol. 11215 LNCS. – P. 540-555.
58. Dutta A. An efficient convolutional neural network for coronary heart disease prediction / A. Dutta, T. Batabyal, M. Basu, S. T. Acton // Expert Systems with Applications. – 2020. – Vol. 159.
59. Evans D. The Internet of Things. How the Next Evolution of the Internet is Changing Everything [Электронный ресурс] / D. Evans // Cisco White Paper. – 2011. – Режим доступа: https://www. cisco.com/c/dam/en_us/about/ac79/docs/innov/ IoT_IBSG_0411FINAL.pdf .
60. FAÇADE [Электронный ресурс] // w3sDesign. – 2022. – Режим доступа: http://w3sdesign.com/?gr=s05&ugr=proble .
61. Fernandes F.E.Jr. Pruning Deep Convolutional Neural Networks Architectures with Evolution Strategy / F.E.Jr. Fernandes, G.G. Yen // Information Sciences. – 2021. – Vol. 552. – P. 29 – 47.
62. Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position / K. Fukushima. // Biological Cybernetics. – 1980. – Vol. 36. – P. 193–202.
63. Ganesh P. Deep Orange: Mask R-CNN based Orange Detection and Segmentation / P. Ganesh, K. Volle, T.F. Burks, S.S. Mehta // IFAC-PapersOnLine. – 2019. – Vol. 52, issue 30. – P. 70-75.
64. Geng Z. Automated design of a convolutional neural network with multi-scale filters for cost-efficient seismic data classification / Z. Geng, Y. Wang // Nature Communications. – 2020. – Volume 11, Issue 1.

65. Ghorpade J. GPGPU Processing in CUDA Architecture / J. Ghorpade, J. Parande, M. Kulkarni, A. Bawaskar // Advanced Computing: An International Journal. – 2012. – Vol. 3, №1. – P. 105 – 120.
66. Han J. Learn CUDA Programming: A beginner's guide to GPU programming and parallel computing with CUDA 10.x and C/C++ / J.Han, B. Sharma. – Birmingham, UK: Packt Publishing, 2019. – P. 508.
67. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition / T. Hastie, R. Tibshirani, J. Friedman. – Berlin: Springer, 2013. – 745 p.
68. He K. Mask R-CNN / He K., Gkioxari G., Dollar P., Girshick R.Mask // Computer Vision and Patter Recognition. – 2017.
69. He Y. Predicting body weight in growing pigs from feeding behavior data using machine learning algorithms / Y. He, F. Tiezzi, J. Howard, C. Maltecca // Computers and Electronics in Agriculture. – 2021. – Vol. 184, article №106085.
70. Hillar G.C. Internet of Things with Python / G.C. Hillar. – Birmingham, UK: Packt Publishing, 2016. – P. 388.
71. Huang Q. CoDeNet: Efficient deployment of input-adaptive object detection on embedded fpgas / Q. Huang, D. Wang, Z. Dong, Y. Gao, Y. Cai, T. Li, B. Wu, K. Keutzer, J. Wawrzynek. // International Symposium on Field-Programmable Gate Arrays, ACM/SIGDA 2021. – 2021. – P. 206 – 216.
72. Huang S.-C. Intelligent FinTech Data Mining by Advanced Deep Learning Approaches / S.-C. Huang, C.-F. Wu, C.-C. Chiou., M.-C. Lin // Computational Economics. – 2021. – Published online: 17 April 2021.
73. Hubel D.H. Receptive fields and functional architecture of monkey striate cortex / D.H. Hubel, T.N. Wiesel. // The Journal of Physiology. – 1968. – Vol. 195. – P. 215 – 243.
74. IBM Cloud Education - SOA (Service-Oriented Architecture) [Электронный ресурс] // IBM Cloud Learn Hub. – 2019. – Режим доступа: https://www.ibm.com/cloud/learn/soa.

75. Ibrahim N. Datamining Using Artificial Neural Networks Techniques / N. Ibrahim. – Lambert Academic Publishing (LAP), 2016. – 76 p.
76. Iriondo A. Pick and place operations in logistics using a mobile manipulator controlled with deep reinforcement learning / A. Iriondo, E. Lazkano, L. Susperregi, J. Urain, F. Fernandez, J. Molina // Applied Sciences (Switzerland). – 2019. – Vol. 9, issue 2, article №348.
77. Jagannathan S. Neural Network Control of Nonlinear Discrete-Time Systems / S. Jagannathan. – Boca Raton: CRC Press, 2006. – 624 p.
78. Jin J.-Q. Multifactor analysis of patients with oral sensory complaints in a case-control study / J.-Q. Jin, H.-M. Cui, Y. Han, S. Su., H.-W. Liu // Chinese medical journal. – 2020. – Vol. 133, issue 23. – P. 2822 – 2828.
79. Kelleher J. D. Fundamentals of Machine Learning for Predictive Data Anayltics (Algorithms, Worked Examples, and Case Studies) / J. D. Kelleher. – Cambridge: The MIT Press, 2015. – 624 p.
80. Keller J. M. Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation / J. M. Keller, D. Liu, D. B. Fogel. – Hoboken: Wiley-IEEE Press, 2016. – 378 p.
81. KerasTuner [Электронный ресурс]. – 2022. – Режим доступа: https://keras.io/keras_tuner/
82. Krizhevsky A. ImageNet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton // Communications of the ACM. – 2017. – Vol. 60, issue 6. – P. 84-90.
83. Kumar D. Projection-mapping-based object pointing using a high-frame-rate camera-projector system / D. Kumar, S. Raut, K. Shimasaki, T. Senoo, I. Ishii // ROBOMECH Journal. – 2021. – Vol. 8, issue 1, article №8.
84. Lantrip J. Results of near-term forecasting of surface water supplies / J. Lantrip, M. Griffin, A. Aly // 2005 World Water and Environmental Resources Congress. – 2005. – P. 436.

85. LeCun Y. Backpropagation applied to handwritten zip code recognition / Y. Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel // Neural Computation. – 1989. – Vol. 1, issue 4. – P. 541-551.
86. LeCun Y. Object Recognition with Gradient-Based Learning / Y. LeCun, P. Haffner, L. Bottou, Y. Bengio // Shape, Contour and Grouping in Computer Vision. – 1999. – P. 319 – 345.
87. Lee J. A comparison and interpretation of machine learning algorithm for the prediction of online purchase conversion / J. Lee, O. Jung, Y. Lee, O. Kim, C. Park // Journal of Theoretical and Applied Electronic Commerce Research. – 2021. – Vol. 16, issue 5, article №16.
88. Li J. Atmospheric PM2.5 concentration prediction and noise estimation based on adaptive unscented Kalman filtering / J. Li, X. Li, K. Wang, G. Cui // Measurement and Control (United Kingdom). – 2021. – Vol. 54, issue 3-4. – P. 292 – 302.
89. Li Y. The role of news sentiment in oil futures returns and volatility forecasting: Data-decomposition based deep learning approach / Y. Li, S. Jiang, X. Li, S. Wang // Energy Economics. – 2021. – Vol. 95, article №105140.
90. Li Y.-L. Fuzzy C-means cluster segmentation algorithm based on hybridized particle swarm optimization / Y.-L. Li, Y. Shen // 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010. – 2010. – P. 811-815.
91. Lin S. Feasibility of using deep learning to detect coronary artery disease based on facial photo / S. Lin, Z. Li, B. Fu, S. Chen, X. Li, Y. Wang, X. Wang, B. Lv, B. Xu, X. Song, Y.-J. Zhang, X. Cheng // European Heart Journal. – 2021. – Vol. 41, issue 46. – P. 4400-4411.
92. Lin S.-K. Classification of patients with Alzheimer`s disease using the arterial pulse spectrum and a multilayer-perceptron analysis / S.-K. Lin, H. Hsiu, H.-S. Chen, C.-J. Yang // Scientific reports. – 2021. – Vol. 11, issue 1. – P. 8882.
93. Lu T. Identification, classification, and quantification of three physical mechanisms in oil-in-water emulsions using AlexNet with transfer learning / T. Lu, F. Yu, C. Xue, B. Han // Journal of Food Engineering. – 2021. – Volume 288.

94. Matolcsy B. Common-mode noise filtering with space-divided differential 2x2 VLC for V2V application / B. Matolcsy, E. Udvary, A. Schranz // Optical and Quantum Electronics. – 2021. – Vol. 53, issue 4, article №182.
95. Matygulina V.N. Dependence of strength of fiberboards from terms of preparation of wood fiber semi-finished products / V.N. Matygulina, N.G. Chistova, A.Yu. Vititnev // Khimiya Rastitel`nogo Syr`ya. – 2020. – Vol. 4. – P. 467 – 474.
96. Medeiros A.C.S. 3D pointing gestures as target selection tools: guiding monocular UAVs during window selection in an outdoor environment / A.C.S. Medeiros, P. Ratsamee, J. Orlosky, Y. Uranishi, M. Higashida, H. Takemura // ROBOMECH Journal. – 2021. – Vol. 8, issue 1, article №14.
97. Morley M. S. Neptune DSS: A decision support system for near-real time operations management of water distribution systems / M. S. Morley, J. Bicik, L. S. Vamvakeridou-Lyroudia, Z. Kapelan, D. A. Savic // Proceeding of 10th International Conference on Computing and Control for the Water Industry: Integrating Water Systems, CCWI 2009. – 2009. – P. 249 – 255.
98. Mote D. S. No Code App Development: Learn To Build Apps Without Code / D. S. Mote. - Chennai, India: Notion Press, 2022. – 162 p.
99. Mukesh Kumar P.C. Prediction of nanofluid viscosity using multilayer perceptron and Gaussian process regression / P.C. Mukesh Kumar, R. Kavitha // Journal of Thermal Analysis and Calorimetry. – 2021. – Vol. 144, issue 4. – P. 1151 – 1160.
100. Nayak S. R. Mixed-mode database miner classifier: Parallel computation of graphical processing unit mining / S. R. Nayak, S. Sivakumar, A. K. Bhoi, G. S. Chae // International Journal of Electrical Engineering Education. – 2021. – P. 1-26.
101. Nikitin N. O. Structural Evolutionary Learning for Composite Classification Models / N. O. Nikitin, I. S. Polonskaia, P. Vychuzhanin, I. V. Barabanova, A. V. Kalyuzhnaya // Procedia Computer Science. – 2020. – Volume 178. – P. 414-423.
102. Ning Z. Intelligent resource allocation in mobile blockchain for privacy and security transactions: a deep reinforcement learning based approach / Z. Ning, S.

Sun, X. Wang, L. Guo, G. Wang, X. Gao, R.Y.K. Kwok // Science Chine Information Sciences. – 2021. – Vol. 64, issue 6, article №162303.

103. Nvidia CUDA-X [Электронный ресурс]. – 2022. – Режим доступа: https://www.nvidia.com/ru-ru/technologies/cuda-x/

104. NVidia CUDA-X GPU-Accelerated Libraries [Электронный ресурс] // NVidia Developer. – 2021. – Режим доступа: https://developer.nvidia.com/gpu-accelerated-libraries .

105. NVidia cuDNN [Электронный ресурс] // NVidia Developer. – 2021. – Режим доступа: https:// developer.nvidia.com/cudnn .

106. Oancea B. GPGPU Computing / B. Oancea, T. Andrei, R.M. Dragoescu // Proceedings of the «Challenges of the knowledge society». – 2014. – P. 2026 – 2035.

107. Oleynikov V.E. Early Predictors of heart Failure Progression in Patients After Myocardial Infarction / V.E Oleynikov., E.V. Dushina, A.V. Golubeva, J.A. Barmenkova // Kardiologiia. – 2020. – Vol. 60, issue 11. – P. 84 – 93.

108. Ong S.-Q. Implementation of a deep learning model for automated classification of Aedes aegypti (Linnaeus) and Aedes albopictus (Skuse) in real time / S.-Q. Ong, H. Ahmad, G. Nair, P. Isawasan, A.H.A. Majid // Scientific Reports. – 2021. – Vol. 11, issue 1, article №9908.

109. Ozkaya U. GPR B scan image analysis with deep learning methods / U. Ozkaya, F. Melgani, M. Belete Bejiga, L. Seyfi, M. Donelli // Measurement: Journal of the International Measurement Confederation. – 2020. – Vol. 165.

110. Parnas D. L. On the criteria to be used in decomposing systems into modules / D. L. Parnas // Communications of the ACM. – 1972. – Vol. 15, issue 12. – P. 1053 – 1058.

111. Pretrained AI Models [Электронный ресурс]. – 2022. – Режим доступа: https://developer.nvidia.com/ai-models

112. Raschka S. Python Machine Learning / S. Raschka. – Birmingham, UK: Packt Publishing, 2015. – 452 p.

113. Rathore S. A Blockchain-Based Deep Learning Approach for Cyber Security in Next Generation Industrial Cyber-Physical Systems / S. Rathore, J. H. Park // IEEE Transactions on Industrial Informatics. – 2021. – Vol. 17, issue 8. – P. 5522 – 5532.

114. Raynolds M. A new estimate of tundra biom phytomass from trans-Arctic field data and AVHRR NDVI / M. Raynolds, D. Walker, H. Epstein, J. Pinzon, C. Tucker // Remote Sensing Letters. – 2012. – Vol. 3, N 5. – P. 403 – 411.

115. Rosenblatt F. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain / F. Rosenblatt // Psychological Review. – 1958. – Vol. 65, No. 6. – P. 386 – 408.

116. Ruiz L. Graph Neural Networks: Architectures, Stability, and Transferability / L. Ruiz, F. Gama, A. Ribeiro // Proceedings of the IEEE. – 2021. – Vol. 109, issue 5. – P. 660 – 682.

117. Rumelhart D.E. Learning representations by back-propagating errors / D.E. Rumelhart, G. E. Hinton, R. J. Williams // Nature. – 1986. – Vol. 323. – P. 533 – 536.

118. Rumelhart D.E. Parallel distributed processing: explorations in the microstructure of cognition / D.E. Rumelhart, J.L. McClelland, PDP Research Group. – Cambridge: MIT Press, 1986. – 516 p

119. Savic M. Deep Learning Anomaly Detection for Cellular IoT with Applications in Smart Logistics / M. Savic, M. Lukic, D. Danilovic, Z. Bodroski, D. Bajovic, I. Mezei, D. Vukobratovic, S. Skrbic, D. Jakovetic // IEEE Access. – 2021. – Vol. 9. – p. 59406 – 59419.

120. Sewak M., Practical convolutional neural networks: implement advanced deep learning models using Python / M. Sewak, M.R. Karim, P. Pujari. – Birmingham, UK: Packt Publishing, 2018. – P. 218.

121. Sheeba F. Segmentation and reversible watermarking of peripheral blood smear images / F. Sheeba, T. Hannah Mary Thomas, J. J. Mammen // 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010. –2010. – P. 1373 – 1376.

122. Simon J.D. Prince Computer Vision: Models, Learning, and Inference / J.D. Simon. – Cambridge: Cambridge University Press, 2012. – 598 p.

123. Simonyan K. Very deep convolutional networks for large-scale image recognition / K. Simonyan, A. Zisserman // 3rd International Conference on Learning Representations, ICLR 2015. – 2015.

124. Syed T. Exploring optimized spiking neural network architectures for classification tasks on embedded platforms / T. Syed, V. Kakani, X. Cui, H. Kim // Sensors. – 2021. – Vol. 21, issue 9, article 3240.

125. Tensor Cores [Электронный ресурс] // NVidia Developer. – 2021. – Режим доступа: https:// developer.nvidia.com/tensor-cores .

126. TensorFlow Hub [Электронный ресурс]. – 2022. – Режим доступа: https://tfhub.dev/

127. The road to Software 2.0 [Электронный ресурс]. – 2022. – Режим доступа: https://www.oreilly.com/radar/the-road-to-software-2-0/

128. Tuomanen D.B. Hands-On GPU Programming with Python and CUDA: Explore high-performance parallel computing with CUDA / D.B. Tuomanen. – Birmingham, UK: Packt Publishing, 2018. – 310 p.

129. V. Kishore Ayyadevara, Yeshwanth Reddy Modern Computer Vision with PyTorch: Explore deep learning concepts and implement over 50 real-world image applications / V Kishore Ayyadevara, Reddy Yeshwanth. – Birmingham: Packt Publishing, 2020. – 824 p.

130. Vaidya B. Hands-On GPU-Accelerated Computer Vision with OpenCV and CUDA: Effective techniques for processing complex image data in real time using GPUs / B. Vaidya. – Birmingham, UK: Packt Publishing, 2019. – 380 p.

131. Welcome to Flask [Электронный ресурс] // Flask. – 2021. – Режим доступа: https://flask. palletsprojects.com/en/2.0.x/# .

132. What is SOA? [Электронный ресурс] // SOA Source Book. – 2021. – Режим доступа: https://collaboration.opengroup.org/projects/soa-book/pages.php?action=show&ggid=1314 .

133. Why TensorFlow [Электронный ресурс] // TensorFlow.org. – 2021. – Режим доступа: https:// www.tensorflow.org/about .

134. Wistuba M. Automation of deep learning / M. Witsuba, A. Rawat, T. Pedapati // 2020 International Conference on Multimedia Retrieval, ICMR 2020. – 2020. – P. 5 – 6.

135. Wistuba M. Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations / M. Wistuba // Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). – 2019. – Vol. 11052 LNAI. – P. 243 – 258.

136. Woschank M. A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics / M. Woschank, E. Rauch, H. Zsifkovits // Sustainability (Switzerland). – 2020. – Vol. 13, issue 9, article №3760.

137. Wulczyn E. Interpretable survival prediction for colorectal cancer using deep learning / E. Wulczyn, D.F. Steiner, M. Moran, M. Plass, R. Reihs, F. Tan, I. Flament-Auvigne, T. Brown, P. Regitnig, P.-H.C. Chen, N. Hegde., A. Sadhwani // Npj Digital Medicine. – 2021. – Vol. 4, issue 1, article №71.

138. Xin-She Yang Introduction to Algorithms for Data Mining and Machine Learning / Yang Xin-She. – Cambridge: Academic Press, 2019. – 175 p.

139. Xu D. Deep learning based emotion analysis of microblog texts / D. Xu, Z. Tian, R. Lai, X. Kong, Z. Tan, W. Shi // Information Fusion. – 2020. – Volume 64. – P. 1 – 11.

140. Yip M.C. Deep learning-based real-time detection of neurons in brain slices for in vitro physiology / M.C. Yip, M.M. Gonzalez, C.R. Valenta, M.J.M. Rowan, C.R. Forest // Scientific Reports. – 2021. – Vol. 11, issue 1, article №6065.

141. Yong Z. Development of a web-basedd decision support system for supporting integrated water resources management in Daegu city, South Korea / Z. Yong, C. Yanpeng, J. Peng, J. Hoogkee // Expert Systems with Application. – 2012. – Vol. 39, issue 11. – P. 10091 – 10102.

142. Zeiler M. D. Visualizing and understanding convolutional networks / M. D. Zeiler, R. Fergus // 3th European Conference on Computer Vision, ECCV 2014. – 2014. – Vol. 8689, issue 1. – P. 818 – 833.

143. Zelentsov V. A. A Model-Oriented System for Operational Forecasting of River Floods / V. A. Zelentsov, A. M. Alabyan, I. N. Krylenko, I. Yu. Pimanov, M. R. Ponomarenko, S. A. Potryasaev, A. E. Semenov, V. A. Sobolevskii, B. V. Sokolov, R. M. Yusupov. // Herald of the Russian Academy of Sciences. – 2019 – Vol. 89, issue 4. – P. 405 – 417.

144. Zhang Z. Non-local means based Rician noise filtering for diffusion tensor and kurtosis imaging in human brain and spinal cord / Z. Zhang, D. Vernekar, W. Qian, M. Kim // BMC Medical Imaging. – 2021. – Vol. 21, issue 1, article №16.

145. Zhang Z. X. A novel segmentation algorithm for complex 3D mesh model in computer vision / Z. X. Zhang, Y. X. Feng, I. R. Hagiwara // 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010. – 2010. – P. 869 – 873.

146. Zhao G. A mask R-CNN based method for inspecting cable brackets in aircraft / G. Zhao, J. Hu, W. Xiao, J. Zou // Chinese Journal of Aeronautics. – 2020.

147. Zheng A. Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists / A. Zheng, A. Casari. – Sebastopol: O`Reilly Media, 2018. – 218 p.

148. Ziade T. Python Microservices Development / T. Ziade. – Birmingham, UK: Packt Publishing, 2017. – 340 p.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ ПЕРЕЧЕНЬ ПУБЛИКАЦИЙ СОИСКАТЕЛЯ ПО ТЕМЕ ИССЛЕДОВАНИЯ

Публикации в журналах из перечня рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты диссертаций на соискание учёной степени кандидата наук, на соискание учёной степени доктора наук:

1. **Соболевский В.А.** Автоматизированная система генерации, обучения и использования искусственных нейронных сетей // Информатизация и связь. 2019. №3. С. 100-107. DOI:10.34219.2078-8320-2019-10-3-100-107.

2. **Соболевский В.А.** Сервис-ориентированный подход к разработке систем на базе свёрточных нейронных сетей // Информатизация и связь. 2020. №5. С.34–40. DOI: 10.34219/2078-8320-2020-11-5-34-40.

3. Михайлов В.В., **Соболевский В. А.,** Колпащиков Л. А., Соловьев Н. В., Якушев Г. К. Методологические подходы и алгоритмы распознавания и подсчета животных на аэрофотоснимках // Информационно-управляющие системы. 2021. №5 (114). С. 20-32. DOI: 10.31799/1684-8853-2021-5-20-32.

В изданиях, включенных в базу данных Scopus:

1. Mikhailov, V.V., **Sobolevskii, V.A.,** Kolpaschikov, L.A. Mask R-CNN-Based System for Automated Reindeer Recognition and Counting from Aerial Photographs // Communications in Computer and Information Sciencethis. 1562. 2022. Стр. 137–151. DOI: 10.1007/978-3-030-98883-8_10.

2. Mikhailov V.V., Spesivtsev A.V., **Sobolevsky V.A.,** Kartashev N.K., Spesivtsev V.A., Lavrinenko I.A., Lavrinenko O.V. Multimodel evaluation of phytomass dynamics of tundra plant communities based on satellite images // Izvestiya, Atmospheric and Oceanic Physics. 57(9). 2021. Стр. 1198-1210. DOI: 10.1134/S0001433821090553.

3. Mikhailov V., Ponomarenko M., **Sobolevsky V.** Simulation of phytomass dynamics of plant communities based on artificial neural networks and NDVI // Recent Advances in Environmental Science from the Euro-Mediterranean and Surrounding

Regions (2nd Edition). EMCEI 2019. Environmental Science and Engineering. 2021. Стр. 1335-1339. DOI: 10.1007/978-3-030-51210-1_211.

4. **Sobolevskii V.A.** The system of convolution neural networks automated training // CEUR Workshop Proceedings. 2803. 2020. Стр. 100-106. DOI 10.24412/1613-0073-2803-100-106.

5. Gnidenko, A., **Sobolevsky, V.,** Potriasaev, S., Sokolov, B. Methodology and integrated modeling technologies for synthesis of cyber-physical production systems modernization programs and plans // IFAC-PapersOnLine. 52(13). 2019. Стр. 642–647. DOI: 10.1016/j.ifacol.2019.11.305.

6. Rostova, E.N., Rostov, N.V., **Sobolevsky, V.A.** Synthesis and simulation of biotechnical position-force control system of a robot manipulator with reconfigurable structure // IFAC-PapersOnLine. 52(13). 2019. Стр. 1097–1101. DOI: 10.1016/j.ifacol.2019.11.342.

7. Zelentsov V.A., Alabyan A.M., Krylenko I.N., Pimanov I.Yu., Ponomarenko M.R., Potryasaev S.A., Semenov A.E., **Sobolevskii V.A.,** Sokolov B. V., Yusupov R.M. A model-oriented system for operational forecasting of river floods // Herald of the Russian Academy of Sciences, 89(4): 405–417, 2019. DOI: 10.1134/S1019331619040130.

8. Rostova, E., Rostov, N., **Sobolevsky, V.,** Zakharov, V. Design and simulation of biotechnical multidimensional motion control systems of a robot manipulator // Proceedings - European Council for Modelling and Simulation, ECMS. 33(1). 2019. Стр. 145–150. DOI: 10.7148/2019-0145.

9. Sokolov, B., Mikoni, S., **Sobolevsky, V.,** Zakharov, V., Rostova, E. Quality evaluation of models and polymodel complexes: Subject-object approach // Proceedings - European Council for Modelling and Simulation, ECMS. 2018. Стр. 305–310. DOI: 10.7148/2018-0305.

10. Petrovskiy, D., Barashkov, A., **Sobolevsky, V.,** Sokolov, B., Pjatkov, V. On the real time logistics monitoring system development using artificial neural network // International Conference on Harbour, Maritime and Multimodal Logistics Modelling and Simulation. 2018. Стр. 14–20.

В изданиях РИНЦ:

1. Михайлов В.В., **Соболевский В.А.,** Колпащиков Л.А. Подсчет северных оленей в скоплениях с использованием сверточной нейронной сети архитектуры Mask R-CNN // Материалы седьмой конференции «Математическое моделирование в экологии» ЭкоМатМод-2021. Г. Пущино, Россия. 2021. С. 76-78.

2. Михайлов В.В., Пономаренко М. Р., **Соболевский В.А.** Моделирование влияния климатических факторов на динамику надземной фитомассы растительных сообществ тундры // Глобальные климатические изменения: региональные эффекты, модели, прогнозы: Материалы международной научно-практической конференции (г. Воронеж, 3-5 октября 2019г.) / Под общ. редакцией С.А. Куролапа, Л.М. Акимова, В.А. Дмитриевой. – Воронеж: Издательство «Цифровая полиграфия», 2019. – Том 2. – С. 106-109.

3. Михайлов В. В., Спесивцев А. В., **Соболевский В. А.,** Лавриненко И. А., Спесивцев В. А. Интеллектуализация процесса моделирования динамики фитомассы растительных сообществ тундры на основе спутниковых снимков // 18 Национальная Конференция по Искусственному Интеллекту с Международным Участием КИИ-2020. Москва. 2020. С. 239-248.

4. **Соболевский В.А.** Программный комплекс автоматизированной генерации сервисов на базе искусственных нейронных сетей // Информационные технологии в управлении (материалы конференции). Санкт-Петербург. 2020. С. 184-186.

В прочих изданиях:

1. Михайлов В. В., Спесивцев А. В., **Соболевский В. А.,** Карташев Н. К., Лавриненко И. А., Лавриненко О. В., Спесивцев В. А. Многомодельное оценивание динамики фитомассы растительных сообществ тундры на основе спутниковых снимков // Исследование Земли из Космоса, 2021. №2. С. 15-30. DOI: 10.31857/s0205961421020056.

2. Зеленцов В.А., Алабян А.М., Крыленко И.Н., Пиманов И.Ю., Пономаренко М.Р., Потрясаев С.А., Семёнов А.Е., **Соболевский В.А.,** Соколов

Б.В., Юсупов Р.М. Модельно-ориентированная система оперативного прогнозирования речных наводнений // Вестник Российской академии наук. 2019. Т. 89. № 8. С. 831-843. DOI: 10.31857/S0869-5873898831-843.

3. Крылов А.В., Охтилев М.Ю., **Соболевский В.А.,** Соколов Б.В., Ушаков В.А. Методологические и методические основы создания и использования интегрированных систем поддержки принятия решений // Изв. вузов. Приборостроение. 2020. №11. Т. 63. С.963–974. DOI: 10.17586/0021-3454-2020-63-11-963-974.

4. Петровский Д. В., **Соболевский В. А.** Сравнение методов искусственной генерации данных для глубокого обучения системы мониторинга // Логистика и управление цепями поставок. 2018. №3. С.86–93.

5. Mikhailov V., Spesivtsev A., **Sobolevsky V.,** Kartashev N. Multi-model estimation of the dynamics of plant community phytomass // 13th IEEE International Conference “Application of Information and Communication Technologies” (AICT2019) (23–25 October 2019, Baku). P.322–328.

6. Mikhailov V., **Sobolevskii V.** Reindeer recognition and Counting System Based on Aerial Images and Convolutional Neural Networks // Pattern Recognition and Information Processing (PRIP'2021): Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. Minsk: UIIP NASB, 2021. P. 89-91.

Патенты и свидетельства о регистрации программ для ЭВМ:

1. Соколов Б.В., **Соболевский В.А.** Программа автоматизированной генерации и обучения искусственных нейронных сетей. Свидетельство №2021668925. Зарегистрировано в реестре программ для ЭВМ 22.10.2021.

2. **Соболевский В.А.** Программа автоматизированного распознавания и подсчёта северных оленей на аэрофотоснимках. Свидетельство № 2022665074. Зарегистрировано в реестре программ для ЭВМ 09.08.2022.