Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Пермский национальный исследовательский
политехнический университет»

На правах рукописи

БАХТИН ВАДИМ ВЯЧЕСЛАВОВИЧ

МЕТОД СИНТЕЗА НЕЙРОСЕТЕВЫХ УСТРОЙСТВ ДЛЯ РЕАЛИЗАЦИИ РЕЖИМА FOG COMPUTING

2.3.2 – Вычислительные системы и их элементы

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Тюрин Сергей Феофентович

Пермь 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования и степень ее разработанности.

На первых этапах развития вычислительных систем обработка данных производилась на одном и том же вычислительном устройстве. По мере развития микропроцессорной техники и сетевых технологий, появилась возможность передавать по сети крупные массивы данных за короткое время, после этого появляется концепция облачных вычислений. Облачные вычисления – это вычисления, которые проводятся на удаленном компьютере, который физически не находится в прямом доступе пользователя [92]. Стало возможным с локальной вычислительной системы низкой производительности осуществлять сложные и объемные вычисления с помощью удаленных серверов. При этом, несмотря на широкое использование «облачных» технологий, наблюдается быстрый рост высоко децентрализованных интеллектуальных решений [90]. Обработка данных в облаке и центрах обработки данных может быть нецелесообразна в тех случаях, когда нужна конфиденциальность и высокое быстродействие, а также когда имеется ограниченная пропускная способность каналов связи для передачи данных, или такие решения характеризуются высокой стоимостью.

Следующим шагом развития распределенных вычислительных технологий становится концепция туманных вычислений. Под туманными вычислениями понимается осуществление обработки данных в «тумане», то есть в группе взаимосвязанных вычислительных устройств без передачи данных удаленным серверам или «облаку». Туманные вычисления – это метод распределения вычислительных задач в виде небольших блоков на небольшие устройства, которые обрабатывают информацию в процессе ее передачи от отправителя к получателю[59].

Существует множество промежуточных и конечных вычислителей с небольшой производительностью: роутеры, коммутаторы, контроллеры, обрабатывающие показания различных электронных датчиков. Большую часть времени работы, центральные процессоры этих устройств простаивают или работают с малой загрузкой, это происходит в связи с их работой в периодическом режиме.

Возможно использование для реализации туманных вычислений не только персональных компьютеров, смарт часов, смартфонов и других гаджетов, но и более простых устройств, содержащих недорогие аппаратные средства [9]. Например, медицинские приборы типа термометров, измерителей давления и пр., поскольку, например, дорогостоящие IoT устройства могут быть не у всех пациентов, да и не у всех медицинских работников, особенно в отдаленных местностях. В то же время предлагаемый подход позволяет в перспективе перевести на новый уровень и некоторые системы, используемые в критически важной технике [24, 26].

Существуют различные подходы к осуществлению крупных вычислений: концентрация вычислительных мощностей в ЦОД (например, центр обработки данных проекта CERN) или распределение вычислительных мощностей на клиентские машины (проект BOINC, который тоже занимается осуществлением научных вычислений). В предлагаемом исследовании предлагается воспользоваться второй парадигмой и осуществлять распределенную реализацию вычислений.

С помощью мейнфреймов сегодня решаются задачи обработки потокового видео, идентификации и кластеризации объектов на них, в рамках видеоаналитики. Но существует класс задач, которые на сегодня нельзя решить, используя только вычислительные мощности ЦОДов, сконцентрированных в одном месте, потому что, для определенных задач важными являются результирующие показатели автономности работы или максимальная скорость получения решения задачи конечным потребителем. Например, задачи распределенной диагностики, как в смысле диагностики оборудования, так и в смысле диагностики состояния человеческого организма в условиях географической или иной удаленности от вычислительных центров или задачи видеоаналитики на потоках видео, полученных с беспилотных летательных аппаратов или иных средств съема видеоинформации с потребностью в автономных вычислениях для принятия решений. Исследования показывают целесообразность [76, 99] создания методов и средств реализации нейросетевых вычислений на устройствах конечного типа, то есть тех, которые находятся максимально близко к конечному пользователю, например, для решения лингвистических задач (обработки естественных языков - NLP), таких как

реализация носимых переводчиков в виде умных очков или через канал восприятия аудио информации.

В настоящее время активно развиваются нейросетевые технологии и их реализация в рамках вычислительных систем и их элементов. Необходимость развития нейросетевых технологий в области медицинской диагностики резко возросла с началом пандемии короновируса. Дальнейшие события привели к возрастанию потребностей совершенствования интеллектуальных видов вооружения и военной техники. Поэтому **объектом исследования является** вычислительная система, её элементы и устройства, используемые для реализации искусственных нейронных сетей, ориентированных на туманные вычисления.

Для решения различных задач, связанных с нейросетевым анализом данных, необходимо иметь возможность строить сложные нейросетевые структуры, обучать их на базах данных предварительно размеченных образцов и использовать обученные нейронные сети для классификации и иных задач [7]. При этом исследователи сталкиваются со следующей трудностью: для обучения и использования нейронных сетей требуются большие вычислительные ресурсы, которые чаще всего имеют высокую стоимость, большое энергопотребление и должны быть физически сконцентрированы в конкретной точке пространства. У данной проблемы могут быть различные решения, например, использование облачных вычислительных ресурсов удаленных дата-центров. Но использование данных ресурсов также чаще всего требует внесения денежных средств в качестве оплаты. Возможным решением данной проблемы, которое мы хотели бы предложить в данной работе, является разделение монолитной нейронной сети на каскад блоков, последовательно выполняемых на связанных между собой вычислителях. Раскроем предполагаемое решение в терминах предметной области.

Нейронная сеть – последовательность из нескольких слоев математических нейронов, соединенных друг с другом. Первый слой нейронов называется входным слоем, он анализирует данные об исследуемом объекте и переводит это в вид коэффициентов, ограниченных определенным интервалом. Последний слой нейронов называется выходным слоем, именно он передает во внешнюю среду информацию о

решении, принятом нейронной сетью. Слои между входным и выходным называются промежуточными или скрытыми, именно они проделывают основную работу по классификации какого-либо объекта [104]. Нейронную сеть, все слои которой производят свои вычисления на одном и том же вычислительном устройстве, принято называть **монолитной нейронной сетью (МНС)** (например, персептрона Розенблатта [87]). Если перед нами стоит задача получения нейронной сети, которая может проводить свои вычисления на нескольких связанных между собой устройствах, то нам потребуется разделить слои этой нейронной сети на блоки последовательных, идущих друг за другом слоев. Каждый из этих блоков будет выполнен на отдельном вычислительном устройстве, а промежуточные результаты будут переданы по сети между ними. Нейронную сеть, разбитую на набор подобных блоков, принято называть **блочной нейронной сетью (БНС)**.

Особую актуальность разрабатываемой теме придает то, что развитие нейронных сетей и туманных вычислений лишь недавно позволило объединить две этих технологии и лишь в последние несколько лет началась разработка данной тематики. Предлагаемый подход ранее не рассматривался в отечественной научной литературе, похожие подходы используются в облачных решениях крупных иностранных компаний, например, Google и Cisco, с использованием фреймворка TensorFlow [84].

В настоящее время возрастает значимость экономии вычислительных ресурсов, оптимизации использования существующих вычислительных систем. Для достижения технологического суверенитета потребуется решить множество вычислительных задач, в том числе осуществлять требовательные к количеству ресурсов нейросетевые вычисления. Расширение обработки данных вычислительной системой зачастую требуется в реальных эксплуатационных задачах на производстве и в организационной деятельности. При этом для увеличения возможностей арифметической, логической, символьной и специальной обработки данных вычислительной системой, как правило, необходимо приобретение и интеграция в систему дополнительных вычислительных узлов, что потребует выделения дополнительных материальных ресурсов. В то же самое время, часто встречаются ситуации, когда вычислительные мощности уже имеющихся устройств задействованы не в полном объеме.

Как следствие, возникает *противоречие в практике* – с одной стороны, тратятся ресурсы на закупку нового оборудования, с другой стороны, имеются не полностью загруженные мощности в рамках вычислительной системы.

Поэтому целесообразно использование метода синтеза нейросетевых устройств для реализации распределенных нейронных сетей, который мог бы обеспечить более оптимальное задействование имеющихся вычислительных устройств таким образом, чтобы не потребовалось устанавливать в существующие системы дополнительные физические вычислители. При этом необходимо подобрать параметры декомпозиции нейронной сети таким образом, чтобы дополнительная нагрузка в виде нейросетевых вычислений, минимально отразилась на исходных параметрах вычислительной системы, таких как общее время обработки получаемых данных и скорость отклика по задачам, которые ранее выполнялись в рамках вычислительной системы.

Таким образом, возможна экономия материальных ресурсов, которые потребовались бы на приобретение дополнительного вычислительного узла, а в сегодняшних реалиях это может стать серьезной проблемой, так как не только цены на вычислительные устройства выросли весьма ощутимо, но и доступность этих устройств снизилась, ввиду логистических осложнений. А возможности настройки декомпозиции нейронной сети позволят добиться того, чтобы распределение вычислительной нагрузки произошло оптимально и с сохранением имеющихся до модификации характеристик вычислительной системы.

Степень разработанности темы исследования. Началом развития нейронных сетей (НС) стала работа F. Rosenblatt, в которой впервые была предложена нейросетевая модель – персептрон. Большой вклад в развитие НС внесли: J. Redmon, A. Farhadi, Потапов А.С., Бурцев М.С. (МФТИ), Южаков А.А., Костарев С.Н., Мильке В.И. (МГТУ), Хижняков Ю.Н., Ясницкий Л.Н. J. Redmon и A. Farhadi[86] предложили архитектуру YOLO для осуществления однопроходной классификации объектов на видео в реальном времени, отечественные ученые также предлагали различные оригинальные архитектуры НС [25, 34]. И развитие архитектур НС и вычислительных сетей привело к созданию класса распределен-

ных НС. Проблемы создания **распределенных НС** в настоящее время рассматриваются в работах таких авторов как: Bradley McDanel, Surat Teerapittayanon, H.T. Kung, Yuchen Liang, W. D. Li, Zhi-Cong Chen, Ионов С.Д., Алексеенко Ю.В. и др. Были предложены различные методы реализации нейросетевых вычислений в облаке, туманных узлах, на конечных устройствах пользователей [49, 58, 75]. Проблемы **синтеза устройств** рассмотрены в работах Дьяченко Ю.Г., Володарского В.Я., Гончарова Д.А, Каменских А.Н, Тюрина С.Ф, Степченкова Ю.А и др. В их работах были продвижения в решении задачи блочного синтеза устройств, заключающиеся в декомпозиции алгоритма на типовые дискретные устройства (блоки) [29]. Однако, задачу блочного синтеза устройств нельзя считать до конца решенной, в том числе и в вопросах синтеза нейросетевых устройств.

Предметом исследования является научно-методический аппарат синтеза вычислительных систем и их элементов, используемых для реализации искусственных нейронных сетей, ориентированных на туманные вычисления.

Цель исследования: решение научной задачи разработки метода синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления.

Для достижения поставленной цели в диссертационной работе поставлены и решены следующие **частные задачи исследования:**

1. Аналитический обзор и анализ моделей, методов и алгоритмов декомпозиции искусственных нейронных сетей в вычислительных системах различных конфигураций и технологий, анализ их недостатков, обоснование актуальности проводимых исследований;

2. Создание математической модели искусственной нейронной сети для синтеза нейросетевых устройств, ориентированных на туманные вычисления;

3. Разработка усовершенствованного метода синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления, и его модификации, позволяющей обеспечить отказоустойчивость;

4. Разработка алгоритма преобразования классической нейронной сети в нейронную сеть, адаптированную для туманных вычислений в устройствах;

5. Разработка алгоритма выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах;

6. Апробация разработанных модели, метода и алгоритмов, реализованных в структуре аппаратного и программного обеспечения элементов вычислительных систем, реализующих распределенные нейронные сети.

Научная новизна заключается в разработанных модели искусственной нейронной сети для синтеза нейросетевых устройств, методе синтеза устройств реализации распределенных искусственных нейронных сетей, ориентированных на туманные вычисления, а также алгоритмах декомпозиции монолитной нейронной сети и выбора оптимального варианта декомпозиции нейронной сети. Новизна научных результатов диссертационного исследования состоит в том, что:

1. Разработана *математическая модель искусственной нейронной сети для синтеза нейросетевых устройств, ориентированных на туманные вычисления.* Она *отличается* от существующих тем, что с ее помощью возможно балансировать размеры декомпозированных блоков нейронной сети в зависимости от характеристик физических устройств, входящих в вычислительный каскад. Это *позволяет* учитывать требуемую загрузку вычислительных узлов при распределении блоков нейронной сети между различными устройствами.

2. Разработан *метод синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления*. Он *отличается* от существующих тем, что учитывает параметры: мощность устройств, оптимальный объем передаваемых между устройствами данных, возможность учета пропорциональности блоков нейронной сети по слоям или по нейронам, а также имеет возможность реализации диагностики и реконфигурации. Это *позволяет* выбрать оптимальный вариант декомпозиции нейронной сети по заранее определенным параметрам и продолжать работу даже в случае отказа или сбоя части устройств в каскаде.

3. Разработан *алгоритм декомпозиции монолитной нейронной сети на каскад блоков блочной нейронной сети, адаптированной для туманных вычислений*. Он *отличается* от существующих тем, что предлагает способ унификации хранения в памяти монолитной нейронной сети и блоков блочной нейронной сети. Это

позволяет проводить многократную декомпозицию в глубину, например, если потребуется декомпозировать отдельный блок еще на несколько блоков.

4. Разработан *алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах*. Он *отличается* от существующих тем, что он реализует многокритериальную оптимизацию путем нахождения Парето-оптимальных вариантов. Это *позволяет* находить оптимальную декомпозицию монолитной нейронной сети сразу по нескольким важным для вычислительной системы параметрам.

Теоретическая значимость заключается в создании модели искусственной нейронной сети, усовершенствованного метода синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления и алгоритмов синтеза и работы устройств, реализующих нейронные сети в режиме fog computing, позволяющих повысить качественные и эксплуатационные характеристики вычислительных систем и их элементов. Разработанные методы и алгоритмы применимы в различных областях, требующих автоматизации с применением нейросетевых методов, в том числе возможна адаптация для реализации нейронных сетей, архитектуры которых отличаются от рассмотренных в работе.

Практическая значимость заключается:

1) в том, что предложенный инструментарий в виде модели, метода, алгоритмов может быть использован в составе аппаратного и программного обеспечения элементов вычислительных систем, реализующих распределенные нейронные сети в режиме туманных вычислений, в том числе в аппаратуре критического применения;

2) в том, что предложенный новый метод декомпозиции монолитной нейронной сети используются в разработках компании «Проминформ», что позволило снизить затраты на создание системы биометрической идентификации на 27 % и сократить энергопотребление прототипа системы биометрической идентификации на 12,7 %. Полученные научные и практические результаты используются в учебном процессе кафедры «Автоматика и телемеханика» Пермского национального исследовательского политехническогоуниверситета;

3) в возможности построения отказоустойчивых вычислительных устройств распределенных нейронных сетей в режиме туманных вычислений с заранее выбранной глубиной адаптации;

4) в применимости разработанных методов и алгоритмов в нейросетевых устройствах без изменений разработанных архитектур и параметров нейронных сетей.

Методология и методы исследования. В диссертационном исследовании используются методы и средства анализа информации о существующих нейронных сетях, математического моделирования нейронной сети, схемотехнического моделирования, анализа и оценки временной сложности алгоритмов. Применяемые методы и средства основаны на положениях дискретной математики, теории булевых функций и автоматов, теории вероятности и математической статистики, теории искусственных нейронных сетей, программирования.

На защиту выносятся следующие научные положения:

1. Существующие методы синтеза нейросетевых устройств позволяют проводить декомпозицию нейронных сетей, однако, они не учитывают при этом важные показатели: стоимость, энергопотребление, время работы нейронной сети и время отклика устройств по существующим задачам.

2. Разработанная модель, метод и алгоритмы решают задачу выбора оптимального набора нейросетевых устройств для реализации туманных вычислений с учетом указанных показателей, при этом формируется множество Парето, из которого возможно получить вариант по заданным ограничениям.

3. Полученные научные и практические результаты позволяют снизить материальные затраты и энергопотребление, сохраняя показатели быстродействия в допустимых пределах, и рекомендуются к использованию в областях критического применения вычислительных систем.

На защиту выносятся следующие новые научные результаты:

1. Математическая модель распределенной искусственной нейронной сети для синтеза нейросетевых устройств, ориентированных на туманные вычисления (п. 7. Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многоядерных, многомашинных и специальных вычислительных систем).

2. Метод синтеза устройств реализации искусственных нейронных сетей для работы в режиме туманных вычислений и его модификация, позволяющая обеспечить отказоустойчивость (п. 7. Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многоядерных, многомашинных и специальных вычислительных систем).

3. Алгоритм декомпозиции монолитной нейронной сети на каскад блоков нейронной сети, адаптированной для туманных вычислений (п. 7. Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многоядерных, многомашинных и специальных вычислительных систем).

4. Алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах (п. 7. Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многоядерных, многомашинных и специальных вычислительных систем).

5. Результат апробации разработанных модели, метода и алгоритмов, реализованных в структуре аппаратного и программного обеспечения элементов вычислительных систем (п. 4. Теоретический анализ и экспериментальное исследование функционирования вычислительных систем и их элементов в нормальных и экстремальных условиях с целью улучшения их технико-экономических и эксплуатационных характеристик).

Достоверность и обоснованность результатов подтверждается соответствием результатов синтеза нейросетевых устройств и схемотехнического моделирования, которые в свою очередь, совпали с результатами прототипирования. Достоверность также подтверждается результатами апробации и внедрения предложенных в диссертации модели, метода и алгоритмов в реальные вычислительные системы. Полученные результаты не противоречат теоретическим и практическим положениям, известным из научных публикаций отечественных и зарубежных исследователей в рассматриваемой предметной области.

Апробация работы. Основные теоретические и практические результаты работы докладывались на научно-технических конференциях: Conference of

Russian Young Researchers in Electrical and Electronic Engineering (ElConRus) 2023, 2022, 2021, 2020, 2019, 14th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE) 2018, IX Международная научная конференция, посвященная 85-летию профессора В.И. Потапова и в других международных и региональных конференциях, Всероссийская научно-техническая конференция «Автоматизированные системы управления и информационные технологии», XIV Международная Интернет-конференция молодых ученых, аспирантов и студентов «Инновационные технологии: теория, инструменты, практика» (InnoTech-2022).

Работы по теме диссертационного исследования выполнялись в рамках научного проекта при поддержке РФФИ на средства гранта № 20-37-90036 Аспиранты «Метод синтеза устройств нейросетевого распознавания для реализации режима Fog computing».

Публикации. Основные результаты диссертационной работы опубликованы в 20 научных работах, из них пять статей в журналах, входящих в перечень ведущих журналов и изданий, рекомендуемых ВАК, три в изданиях, индексируемых в базах SCOPUS, два свидетельства о регистрации программы для ЭВМ, остальные – в тезисах докладов, материалах конференций и прочих источниках.

Объем и структура работы. Диссертация состоит из введения, пяти глав, заключения, списка литературы из 104 наименований и 5 приложений. Полный объем диссертации составляет 153 страницы, из которых 115 страниц занимает основной текст диссертации, включающий 43 рисунка и 10 таблиц.

ГЛАВА 1. АНАЛИЗ ОБЪЕКТА И ПРЕДМЕТА ИССЛЕДОВАНИЯ. ПОСТАНОВКА НАУЧНОЙ ЗАДАЧИ И ЧАСТНЫХ ЗАДАЧ ИССЛЕДОВАНИЯ

1.1. Существующие методы и модели описания нейронных сетей и их классификация

Математический нейрон МакКаллока-Питса [63] представляет из себя элемент, который вычисляет выходной сигнал (по определенному правилу) из совокупности входных сигналов. Между собой нейроны могут быть соединены по-разному, но суть работы нейронной сети остается постоянной. По совокупности поступающих на вход сети сигналов на выходе формируются выходные сигналы. Нейронную сеть можно представить в виде черного ящика, у которого есть входы и выходы [12, 13].

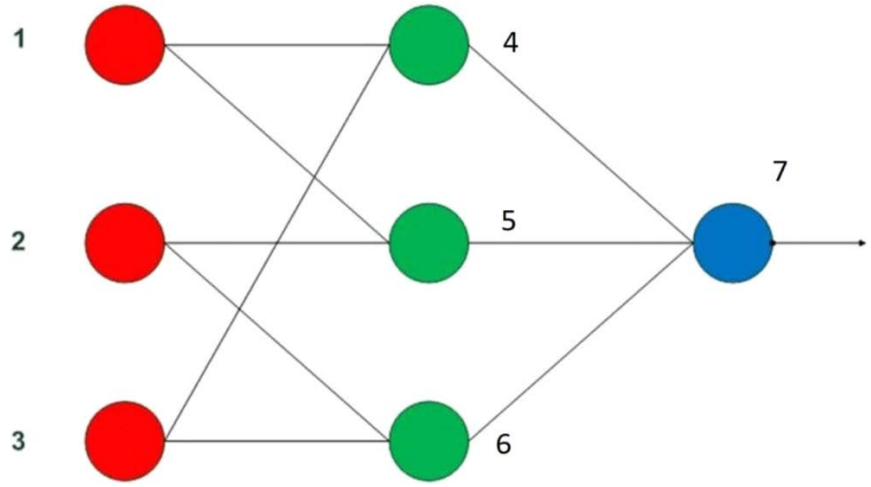


Рисунок 1 – Оптимизированная нейронная сеть TSBuilder

Чаще всего структура связей между нейронами представляется в виде матрицы W, которую называют весовой матрицей. Элемент матрицы w_{ij}, определяет вес связи, идущей от элемента i к элементу j [31]. Для того, чтобы понять, как составляются весовые матрицы, давайте рассмотрим простую нейронную сеть, которая была создана нашим коллективом в рамках предыдущего исследования при создании программного комплекса TSBuilder (представлена на Рисунке 1). Данная сеть использовалась для классификации сложной терминологии, в случае, когда термин состоял из трех слов. Нейроны входного слоя давали информацию о

том, к какому полю относятся слова, входящие в термин. Нейроны скрытого слоя отвечали за категоризацию пар слов в словосочетании, а нейрон выходного слоя осуществлял голосование между попарными сравнениями и принимал решение об общей принадлежности полученного термина [41, 43, 44, 45, 68, 69].

Весовая матрица такой нейронной сети будет иметь следующий вид:

$$W = \begin{bmatrix} 0 & 0 & 0 & W_{14} & W_{15} & 0 & 0 \\ 0 & 0 & 0 & 0 & W_{25} & W_{26} & 0 \\ 0 & 0 & 0 & W_{34} & 0 & W_{36} & 0 \\ W_{14} & 0 & W_{34} & 0 & 0 & 0 & W_{47} \\ W_{15} & W_{25} & 0 & 0 & 0 & 0 & W_{57} \\ 0 & W_{26} & W_{36} & 0 & 0 & 0 & W_{67} \\ 0 & 0 & 0 & W_{47} & W_{57} & W_{67} & 0 \end{bmatrix}$$

Например, от второго элемента к пятому идет связь, вес которой равен W_{25}. В роли функций активации могут выступать абсолютно любые математические или логические функции.

Классификация нейронных сетей по характеру обучения делит их на несколько типов. Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой [93]. Развитая Кохоненом [72] и многими другими, модель обучения без учителя не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т.е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы.

Классификация нейронных сетей по типу настройки весов делит их на сети с фиксированными весами связей (весовые коэффициенты выбираются сразу) и сети с динамическими весами связей (обучение меняет синаптические веса) [66].

Классификация нейронных сетей по типу входной информации делит их на аналоговые (входная информация представлена в форме действительных чисел) и двоичные (входная информация в двоичном виде) [19].

Также можно классифицировать нейронные сети по тому, было ли у них предварительное обучение или они начнут свое обучение с момента начала функционирования [17]. В данной статье описывается работа с уже обученными нейронными сетями. То есть к моменту разделения на блочные нейронные сети веса синапсов уже будут известны и зафиксированы. Как было отмечено ранее, при построении предсказательных моделей исходные данные обычно разбиваются на обучающую и контрольную выборки. Обучающая выборка используется для обучения модели, тогда как контрольная выборка служит для получения оценки прогнозных свойств модели на новых данных[47].

В описываемом исследовании важно, чтобы к моменту синтеза нейросетевого устройства обучение было завершено и веса синапсов стабилизированы [56, 57], это ограничение временное и его можно преодолеть в будущем. Также в приведенном исследовании рассматриваются именно многослойные нейронные сети, так как в полных нейронных сетях слишком много связей и их сложно разделить на независимые кластеры, которые могли бы выполняться на различных физических устройствах [21, 73]. В данной статье будет рассматриваться математическая модель разделения нейронной сети без обратных связей, а также была произведена доработка, которая позволила разделять сети с обратными связями.

В представленном исследовании описывается работа с полностью обученными многослойными нейронными сетями без обратных связей, то есть к моменту разделения на блочные нейронные сети веса синапсов уже будут известны и зафиксированы. Важно, чтобы к моменту синтеза нейросетевого устройства обучение было завершено и веса синапсов стабилизированы, это ограничение временное и его можно преодолеть в будущем. В рамках исследования не имеет значения, для решения каких именно задач будут использованы синтезируемые нейросетевые устройства и блочная нейронная сеть. Декомпозиции с помощью разработанного метода могут подвергаться нейронные сети для распознавания, управления, классификации, генерации и других задач, которые решаются нейросетевыми методами.

1.2. Анализ вычислительных систем и их элементов, используемых для реализации искусственных нейронных сетей и синтеза нейросетевых устройств, ориентированных на туманные вычисления

Использование ресурсов суперкомпьютеров требует большого количества ресурсов (денежных, материальных и человеческих) с учетом передачи этой информации. Экономия ресурсов возможна за счет отсутствия передачи данных и вычислительного времени в удаленный ЦОД.

Для начала проанализируем работу «Применение распределенных вычислений для реализации алгоритмов deep learning» [1]. В ней описывается архитектура программного пакета для реализации нейронных сетей с использованием предоставляемых свободных мощностей ПК. Одной из главных задач, которые ставятся в данной работе, является задача повышения производительности обучения нейронных сетей. В работе были выдвинуты основные требования для программной платформы: высокая скорость обработки данных, многопоточность, объектно-ориентированный дизайн, модульность программного пакета, мультиплатформенность, высокая стабильность.

Архитектура предложенной в работе системы, предназначенной для реализации нейронных сетей, представлена на Рисунке 2.

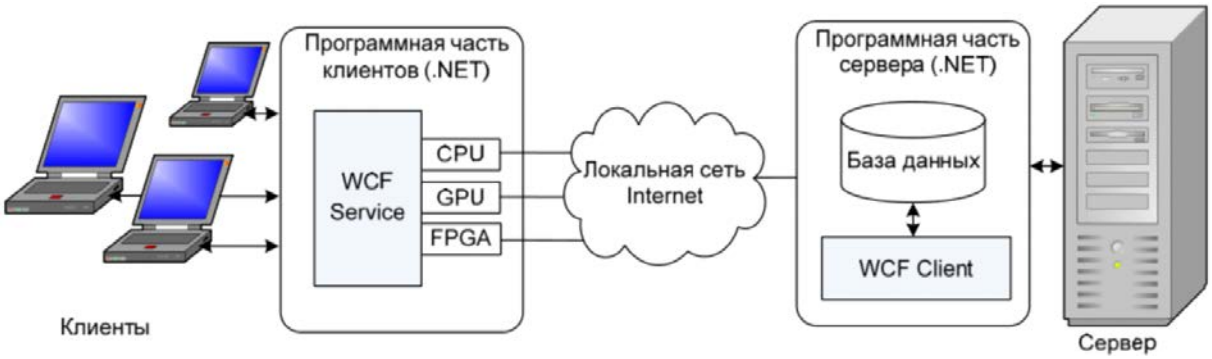


Рисунок 2 – Архитектура гетерогенной системы для распределенных вычислений

Решаемая вычислительная задача состоит из клиентской и серверной части. Предоставляемые клиентами ресурсы, на которые ложится основная нагрузка при вычислении, представлены в качестве WCF-сервиса. Вычислительными блоками могут быть CPU, GPU или FPGA. Достоинствами описанной в данной статье системы по сравнению с существующими аналогами: объединение в единую систему

вычислительных узлов различной архитектуры (CPU, GPU и FPGA); возможность коммуникации на основе различных протоколов, в том числе и с шифрованием; возможность динамического добавления и отключения клиентов; разработка вычислительных задач на современных языках программирования, таких как C#, VB.NET, IronPython и других.

Второй работой для анализа является «DeepFog: Fog Computing-Based Deep Neural Architecture for Prediction of Stress Types, Diabetes and Hypertension Attacks» [85]. В ней описывается создание системы мониторинга состояния здоровья пользователя. Авторы предлагают систему, состоящую из трех уровней. Первый уровень – пользовательский, на котором собирается различная информация о состоянии организма пользователя, о его местонахождении, об условиях окружающей среды. Второй – уровень обработки данных в туманной сети, на котором собранная информация сортируется, обрабатывается. ИНС производит классификацию по типу стресса, классификацию по стадии гипертонии, а также делается прогноз по риску гипертонии и по заболеванию диабетом. Затем получившаяся информация передается на третий – облачный уровень, на котором с полученными данными работают врачи-специалисты, либо информация может передаваться родственникам пользователя, для предупреждения в случае опасной для здоровья ситуации. На Рисунке 3 представлена схема данной системы.

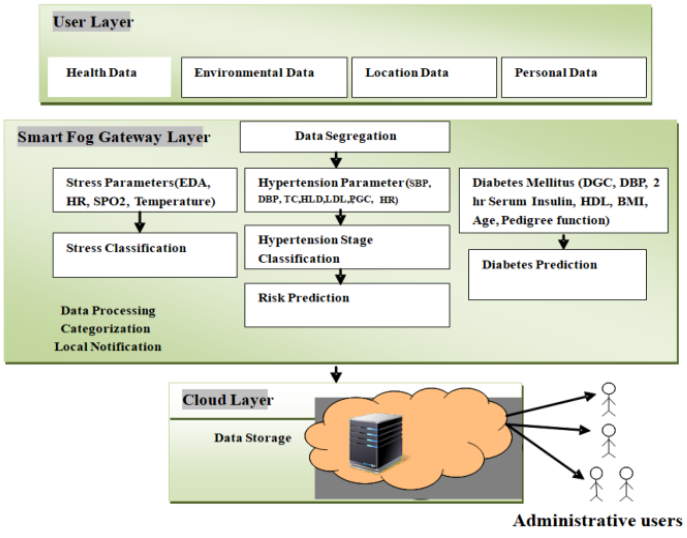


Рисунок 3 – Схема системы мониторинга состояния здоровья пользователя

Следующим рассмотрим метод, предлагаемый в работе «Распределенный запуск нейронных сетей на множестве вычислительных узлов» [18], предлагающий рассмотреть реализацию нейронных сетей на множестве вычислительных узлов в кластерной среде.

В данном исследовании предложена структура воспроизводящейся искусственной нейронной сети (ВИНС), позволяющей, в частности, моделировать ассоциативную ячейку памяти, способную увеличивать свою емкость при поступлении на ее входы новых сигналов. Основной особенностью ВИНС является изменение структуры и настройка внутреннего состояния связей ее внутренними средствами — специализированными нейронами роста и модификации связей.

В работе рассматриваются два графа: граф узлов нейронной сети и граф физически связанных вычислительных устройств. Представлен оригинальный алгоритм декомпозиции нейронной сети на кластеры нейронов. Рассмотрено практическое применение алгоритма – через реализацию сервера, который управляет распределением задач, входные и выходные данные доступны на основном узле, который выделен в рамках архитектуры системы, остальные узлы доступны для взаимодействия только основному узлу, узлы в рассматриваемой работе называются серверами. Итоговая архитектура работающей сети серверов представлена на Рисунке 4.

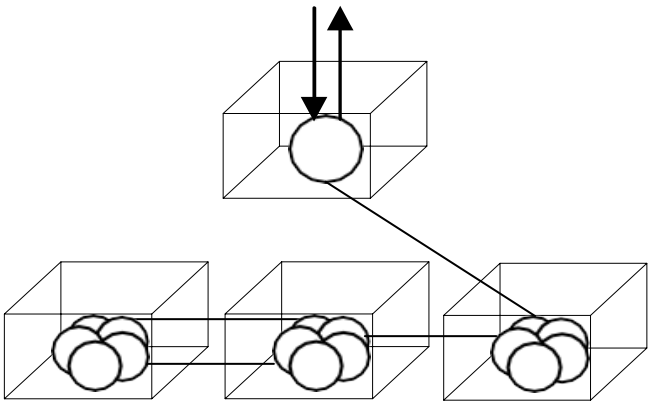


Рисунок 4 – Архитектура сети серверов

В виду заложенного распределения нейронной сети по отдельным вычислительным узлам, а также предполагаемой разнородности производительности уз-

лов возникла задача их синхронизации при обработке проходящих по сети данных. Главная проблема, которую требовалось решить, – это как долго каждый отдельный нейрон должен ожидать сигналов по всем своим входам. Решением принято выставление допустимых задержек при передаче данных на основе анализа вычислительных мощностей устройств и пропускной способности каналов связи.

Следующей рассматриваемой работой является «Data Analytics with Deep Neural Networks in Fog Computing Using TensorFlow and Google Cloud Platform» [84], в ней исследователи описывают алгоритм уменьшения количества данных, которые поступают с конечных устройств, перед размещением их на центральный сервер. В данной работе исследователи предлагают заменить узлы тумана на туманные станции (или туманные сервера). Данные туманные станции представляют собой уменьшенную версию облачных серверов. Данные, поступающие на туманные станции, могут быть получены с различных сенсоров и различных устройств, поэтому требуется фильтрация данных, чтобы избежать их смешивания. В данной статье авторы утверждают, что используют библиотеку TensorFlow, чтобы фильтровать данные и сортировать данные, а затем анализировать, нужны ли данные в дальнейшей работе. Таким образом ненужные данные отбрасываются и тем самым снижается нагрузка на центральный сервер.

На Рисунке 5 показан туманный слой данной архитектуры более подробно. После того, как данные прошли фильтрацию, фильтрующая программа из библиотеки TensorFlow решает, отправить данные прямо на облачный сервер или сначала отправить их на анализ. Существует несколько программ из библиотеки TensorFlow для анализа данных в реальном времени. Данные программы обучены на подходящих наборах данных с использованием машинного обучения. Для обучения модели используется сервис Google Cloud Platform (GCP), который называется Cloud Datalab [84]. Это один из самых мощных инструментов, созданных для анализа данных и построения моделей машинного обучения на GCP.

На Рисунке 6 показано, насколько сильно влияет применение данного алгоритма на пропускную способность центрального сервера.

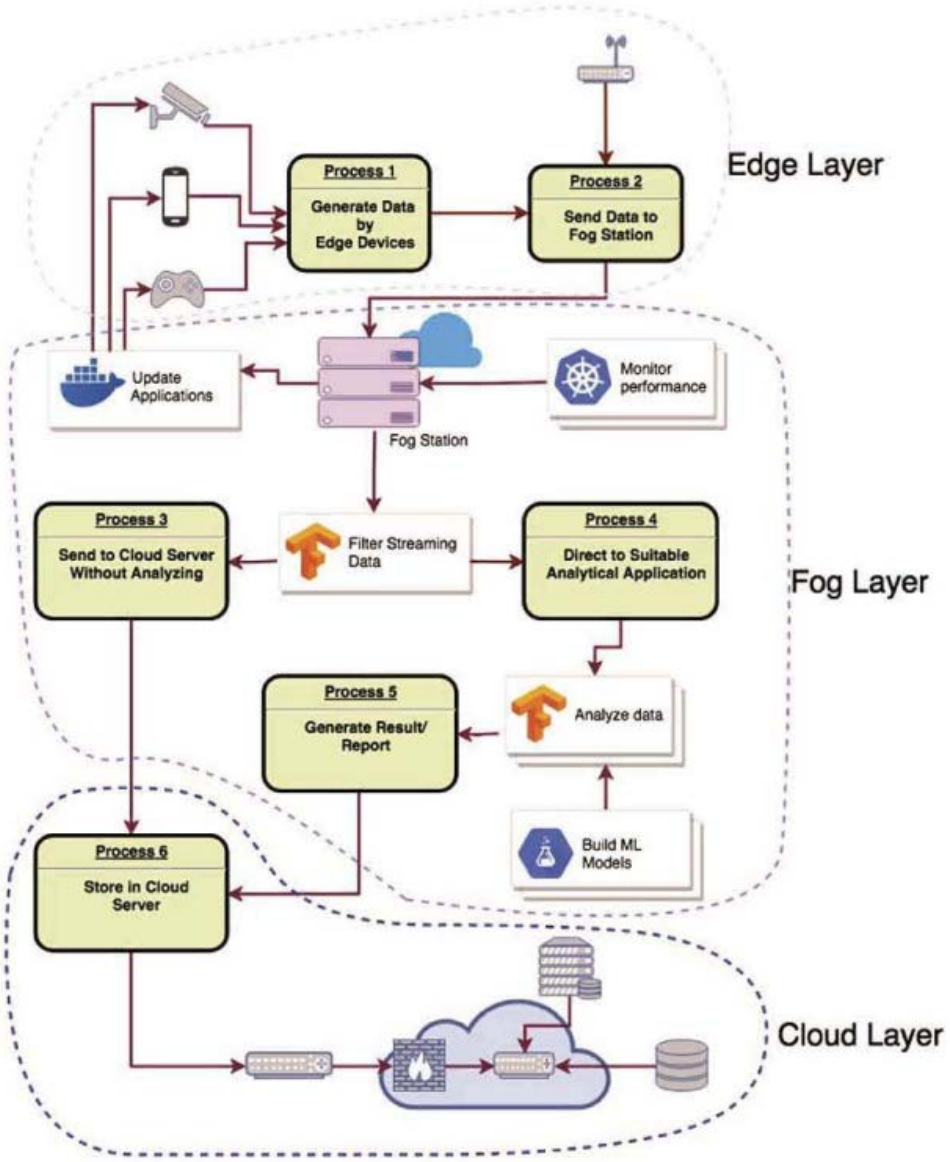


Рисунок 5 – Общая схема архитектуры

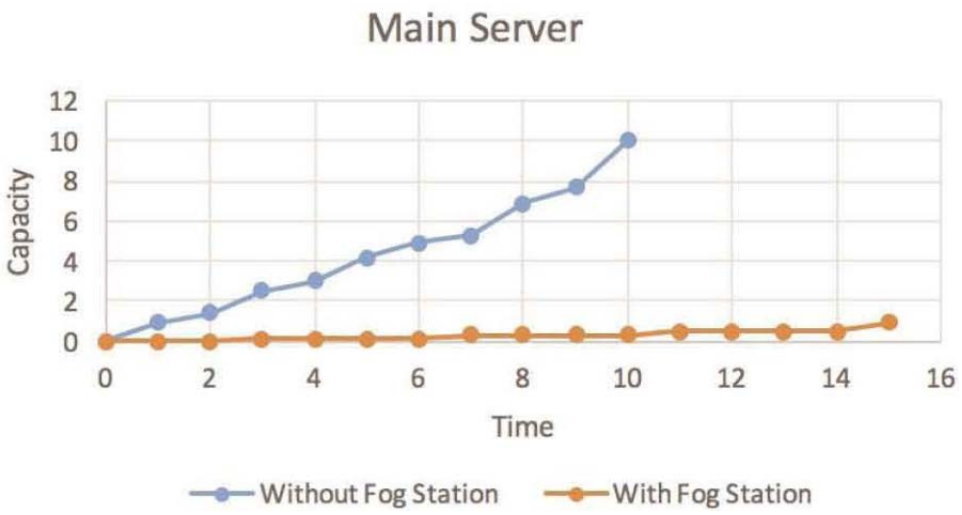


Рисунок 6 – График зависимости пропускной способности центрального сервера от времени для подхода без туманных станций и с ними

Авторы приходят к выводу, что их работа принесет значительную пользу, поскольку они фокусируются на применении туманных устройств и предобработке данных, перед передачей их на сервер. Они считают, что это может помочь идентифицировать проблемы на ранних стадиях, кроме того, анализ производительности туманных станций позволит обеспечить эффективное распределение ресурсов.

Следующая работа, которую следует рассмотреть, называется «An Efficient Binary Convolutional Neural Network with Numerous Skip Connections for Fog Computing» [40]. В данной работе авторы предлагают систему, в которой общий процесс передачи данных аналогичен системе DeepIns [74]. Однако в данной работе есть два отличия. Во-первых, модель, работающая на туманных узлах в этой системе, представляет собой бинарную глубокую нейронную сеть, которая превосходит простую нейронную сеть в DeepIns. На Рисунке 7. можно видеть общую структуру предложенной системы для туманных вычислений. Видно, что при заданных ограничениях, чем выше производительность классификации, тем меньше данных требуется передать в облако, что значительно сократит задержку передачи и улучшит коммуникационный трафик. Во-вторых, вместо промежуточных данных в облако передаются необработанные данные из узлов тумана, что может помочь разработчику оптимизировать всю систему.

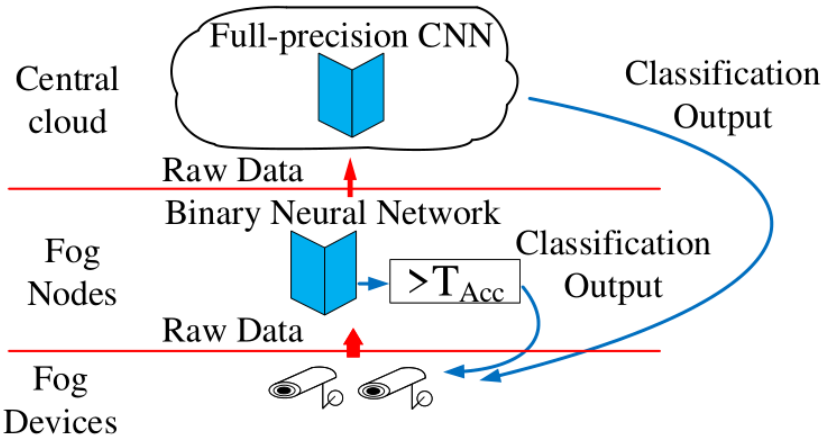


Рисунок 7 – Общая структура предложенной системы для туманных вычислений

В данном исследовании авторы предлагают решить проблемы с задержкой передачи данных в приложениях глубокого обучения в интеллектуальных системах путем внедрения бинарных нейронных сетей (BNN) в туманные вычисления.

Новый BNN под названием BNSC-Net был разработан для повышения точности. Исследователи обнаружили, что точность BNN пропорциональна количеству “пропущенных соединений” в сети. Повторное использование операции конкатенации также может повысить производительность BNN. Кроме того, исследователи обнаружили взаимосвязь между интервалом обновления Straigt-Through-Estimator (STE) и точностью BNN и предложили оптимальный интервал обновления. Чтобы выявить эффективность BNSC-Net, были проведены сравнительные эксперименты между BNSC-Net и современной системой DeepIns. Результаты показывают, что BNSC-Net превосходит DeepIns и эффективен для развертывания в туманных вычислениях.

Следующая работа для рассмотрения называется «Fog Computing and Convolutional Neural Network Enabled Prognosis for Machining Process Optimization» [58]. В представленной работе описывается система распределенных вычислений с поддержкой Fog Computing и сверточных нейронных сетей (CNN) для динамического прогнозирования и оптимизации процессов обработки для решения практических производственных задач. Среди функций и нововведений:

1. Разработана трехуровневая модель туманных и облачных вычислений для оптимизации использования вычислительных ресурсов и минимизации объема передаваемых данных. Благодаря модели туманных вычислений система может обойти распространенную проблему низкой эффективности облачных систем, связанную с высокой задержкой передачи данных в промышленных сетях. Глубокий анализ и вычисления в системе, такие как алгоритмическое обучение, анализ данных, оптимизация планирования/перепланирования, хорошо сбалансированы с рассматриваемой архитектурой НС.

2. Современные обрабатывающие компании, особенно малые и средние предприятия, характеризуются мелкосерийным производством и производством по индивидуальному заказу. Во время процессов обработки могут возникать уникальные ошибки. Чтобы эффективно справляться со сложными условиями, система дополнена сверточной нейронной сетью для динамического извлечения чувствительных признаков неисправности из большого количества сигналов, получен-

ных в производстве. В данной системе применяются механизмы предварительной обработки сигнала для его очищения и повышения производительности CNN в практических производственных ситуациях.

3. Производительность системы была оценена в европейской обрабатывающей компании и показала удовлетворительные результаты. Исследования показали, что система эффективна и подходит для применения на производстве. Система применима и к другим производственным процессам путем обучения CNN с использованием признаков ошибок для этих процессов.

В данной системе на слое терминала датчики мощности устанавливаются на обрабатывающее оборудование. Данные о мощности собираются платой Wemos Wi-Fi [64], которая представляет собой экономичную платформу с аналого-цифровым преобразователем и встроенным модулем Wi-Fi. Отслеживаемые данные о мощности передаются на слой тумана через Wi-Fi. Слой тумана состоит из интернет-маршрутизатора и Raspberry Pi [94], Raspberry Pi расположен рядом с платой Wi-Fi Wemos для приема данных из цеха. Полученные данные обрабатываются обученной CNN, развернутой в Raspberry Pi. Опорные сигналы хранятся на облачном уровне для обучения CNN. Управляемая системным координатором, CNN при необходимости переобучается, и обученная CNN передается обратно на слой тумана для повторного развертывания. На Рисунке 8 показана структура разработанной сверочной нейронной сети.

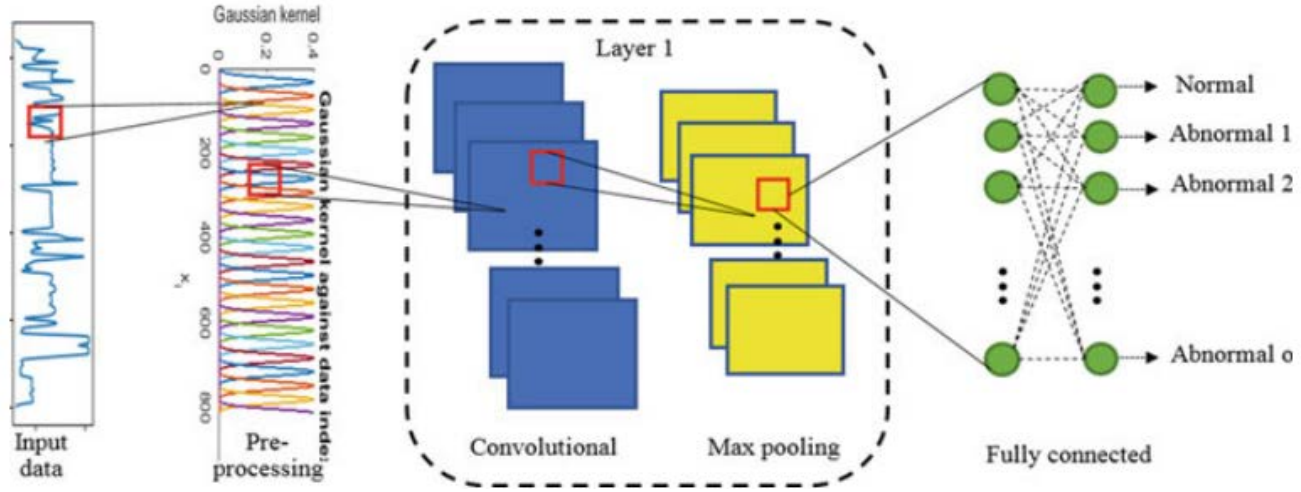


Рисунок 8 – Базовая структура разработанной в работе сверочной нейронной сети (convolutional neural network)

Далее рассмотрим работу под названием «Distributed Deep Neural Networks over the Cloud, the Edge and End Devices» [95]. В данной работе авторы предлагают новый вид нейронных сетей, который называется distributed deep neural network (DDNN), т.е. распределенные глубокие нейронные сети.

DDNN отображает обученную DNN на разнородные физические устройства, распределенные локально, на конечные устройства и в облаке. Поскольку DDNN полагается на совместно обученную структуру DNN во всех частях нейронной сети, как для обучения, так и для вывода, многие сложные инженерные решения значительно упрощаются. На Рисунке 9 представлен обзор архитектуры DDNN. Представленные конфигурации показывают, как DDNN может масштабировать вычисления на разные физические устройства. Облачный DDNN на (a) можно рассматривать как стандартный DNN, работающий в облаке. В этом случае входные данные датчиков, полученные на конечных устройствах, отправляются в облако в исходном формате (формат raw input), где выполняются все слои вывода DNN.

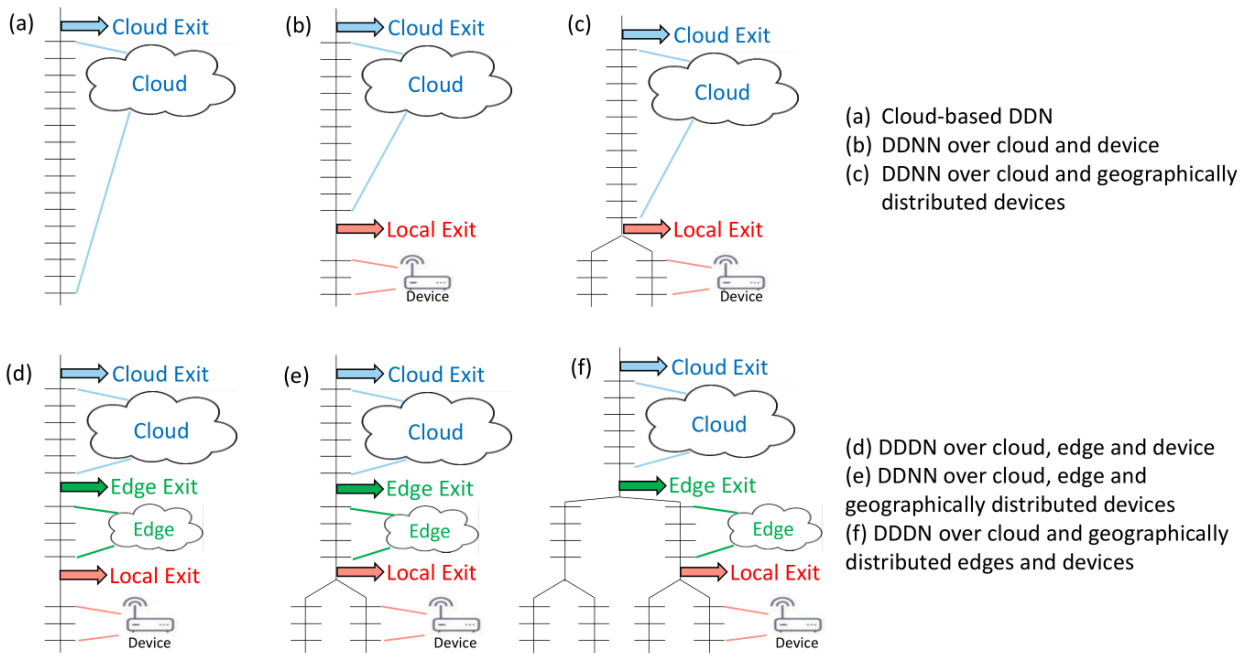


Рисунок 9 – Архитектура распределенной нейронной сети

Авторы утверждают, что могут расширить эту модель, включив в нее одно конечное устройство, как показано на (b), выполнив часть вычисления вывода DNN на устройстве, а не отправив необработанные входные данные в облако. Ис-

пользуя точку выхода после вывода устройства, можно классифицировать те выборки, в отношении которых локальная сеть уверена, без отправки какой-либо информации в облако. Для более сложных случаев промежуточный вывод DNN (вплоть до локального выхода) отправляется в облако, где выполняется дальнейший вывод с использованием дополнительных слоев нейронной сети и принимается окончательное решение о классификации. Промежуточный выходной сигнал может быть сконструирован таким образом, чтобы он был намного меньше входного сигнала датчика (например, необработанное изображение с видеокамеры), и, следовательно, значительно сократить требуемые ресурсы сети для связи между конечным устройством и облаком.

DDNN также может быть распространен на несколько конечных устройств, которые могут быть географически распределены, как показано на (c). Эти устройства работают вместе для принятия решения о классификации. Здесь каждое конечное устройство выполняет локальные вычисления, как на (b), но их выходные данные объединяются вместе перед локальной точкой выхода. Поскольку вся сеть DDNN совместно обучается на всех конечных устройствах и точках выхода, сеть автоматически объединяет входные данные с целью достижения максимальной точности классификации. Это автоматическое объединение данных (sensor fusion) упрощает вывод во время выполнения, устраняя необходимость ручного объединения выходных данных с нескольких конечных устройств. Как и прежде, если локальная точка выхода не приняла решения по классификации, каждое конечное устройство отправляет промежуточные выходные данные в облако, где выполняется еще один цикл агрегирования функций перед принятием окончательного классификационного решения.

DDNN также масштабируется по вертикали, используя конечный уровень в иерархии распределенных вычислений между конечными устройствами и облаком, как показано на (d) и (e). Конечный сервер действует аналогично облаку, принимая выходные данные от конечных устройств, выполняя агрегацию и классификацию, если это возможно, и пересылая свои собственные промежуточные выходные данные в облако, если требуется дополнительная обработка. Таким об-

разом, DDNN естественным образом регулирует время отклика системы на основе каждой выборки. Образцы, которые могут быть правильно классифицированы локально, удаляются без какой-либо связи с конечным устройством или облаком. Образцы, для извлечения которых требуется больше объектов, чем может быть предоставлено локально, отправляются на конечное устройство и, при необходимости, в конечном итоге в облако. Наконец, DDNN также могут масштабироваться географически по конечному слою, что показано на (f).

Экспериментальные результаты авторов показывают, что с помощью данной платформы DDNN один правильно обученный DNN может быть отображен в иерархию распределенных вычислений для удовлетворения требований целевого приложения к точности, качеству связи и задержке, получая при этом неотъемлемые преимущества, связанные с распределенными вычислениями, такие как отказоустойчивость и конфиденциальность.

DDNNs сокращают требуемые ресурсы сети по сравнению со стандартным подходом к разгрузке облака за счет выхода из множества выборок в локальном агрегаторе и отправки компактного двоичного представления объектов в облако, когда требуется дополнительная обработка. Для тестового набора данных требования к сети для DDNN снижается более чем в 20 раз по сравнению с загрузкой необработанных входных данных с датчиков в DNN в облаке, который выполняет все вычисления вывода.

Последняя работа для анализа называется «Deep learning in the fog» [49]. В данной работе авторы выделили три основные проблемы в т.н. революции в передовых системах с искусственным интеллектом:

1. Обеспечение периферийных устройств достаточной вычислительной мощностью для запуска алгоритмов искусственного интеллекта, таких как DNN;

2. Создание библиотек, которые обеспечивают передачу предварительно обученных моделей на периферийные устройства и эффективное их использование;

3. Предоставление платформы для упрощения управления, обновления и синхронизации набора периферийных устройств независимо от системы.

Авторы утверждают, что в настоящее время на рынке видно два противоположных направления развития машинного обучения на конечных устройствах, которые они называют «fat edge devices» и «thin edge devices».

Первое предполагает, что конечные устройства должны быть автономными, а все потенциальные прогнозы или решения должны производится с сокращением времени и на стороне устройства, когда требуется подключение к облаку. Это направление подходит для автономных транспортных средств, интеллектуальных носимых устройств и здравоохранения, а также для промышленных компаний, для которых конфиденциальность производственных данных имеет решающее значение. Второе направление — это «thin edge devices», которые предполагают тесно связанную облачную архитектуру, а это значит, что облако по-прежнему будет играть основную роль в обработке новых данных.

Различия, предложенные в этой работе, являются примерами совершенно противоположных направлений. Вместо того чтобы делить рынок, следует ожидать разработки гибридных решений, предназначенных для туманных вычислений, а затем и для периферийных вычислений. Процесс создания архитектуры умного города, увеличение количества интеллектуальных транспортных средств, одежды и других объектов может привести к увеличению числа небольших центров обработки данных, расположенных близко к источникам данных.

Intel, Google, NVIDIA, Apple и Samsung лидируют в разработке аппаратного обеспечения и пытаются создать достаточно мощные устройства, предназначенные для быстрого анализа видеопотоков. С другой стороны, Facebook и Google создают отличное программное обеспечение для высокопроизводительного глубокого обучения и обычного обучения. В настоящее время существуют отличные инструменты для быстрой разработки и прототипирования, такие как TensorFlow и PyTorch, но возможности развертывания ограничены. Например, разработчики сообщают о многих проблемах с запуском TensorFlow Lite на Raspberry Pi, а также на других устройствах. В то же время авторы ждут платформы Caffe2Go от Facebook. Наконец, управлять инфраструктурой конечных устройств непросто, но такие платформы, как EdgeX, AWS IoT Greengrass или Google Cloud Edge, помо-

гают в этом. В настоящее время многие из них, по-видимому, поддерживают только «thin edge devices», требуя отправки данных с конечного устройства в облако для их анализа и хранения.

На Рисунке 10 представлена модель «thin edge devices», а на Рисунке 11 – модель «fat edge devices».

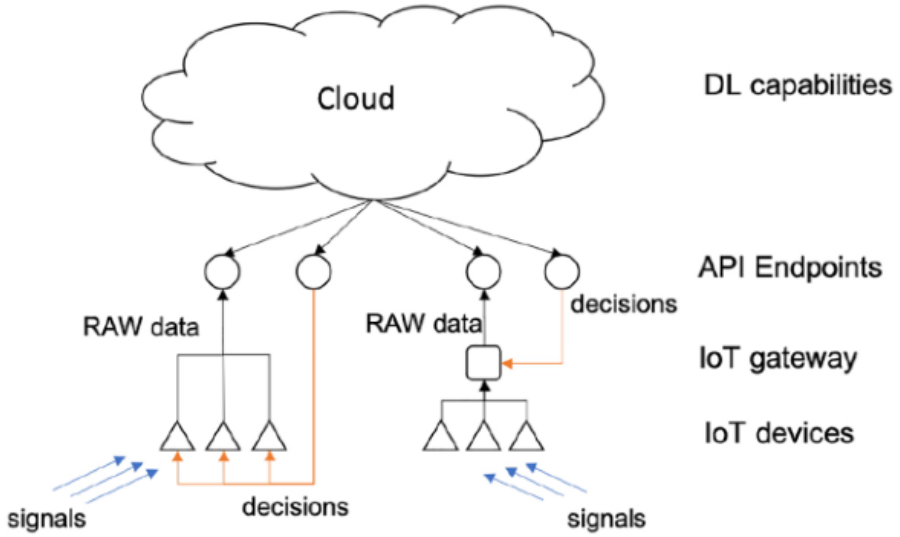


Рисунок 10 – Архитектура «thin edge devices»

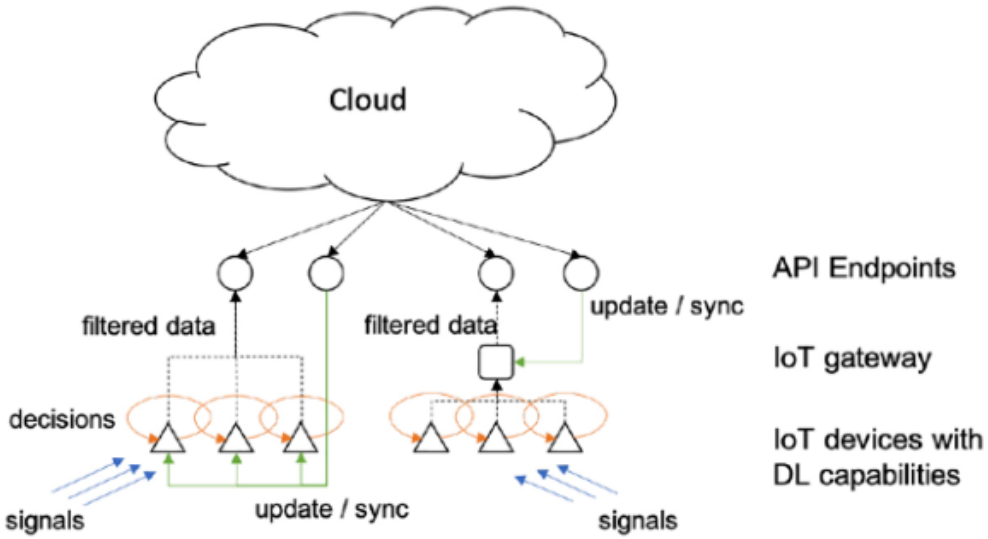


Рисунок 11 – Архитектура «fat edge devices»

Таким образом имеется противоречие в практике – с одной стороны, расширение обработки данных в случае нейронных сетей требует приобретения новых вычислительных устройств, с другой стороны, в вычислительных системах име-

ются не полностью загруженные вычислениями устройства. Следовательно, практическая задача заключается в разработке средств распределения дополнительной вычислительной нагрузки между устройствами вычислительной системы, у которых остался невостребованный вычислительный ресурс, для ее решения выполним анализ существующего научно методического аппарата синтеза вычислительных систем и их элементов, используемых для реализации искусственных нейронных сетей, ориентированных на туманные вычисления.

1.3. Анализ научно-методического аппарата синтеза вычислительных систем и их элементов, используемых для реализации искусственных нейронных сетей, ориентированных на туманные вычисления

В первом из рассмотренных методов есть несомненные достоинства, а именно: возможность задействовать вычислительные устройства с различными архитектурами и способность к динамическому масштабированию. Однако, в представленной схеме имеется главный вычислительный узел, который распределяет задачи между всеми остальными (сервер), без которого не будет возможности функционировать всей распределенной нейронной сети. Предлагаемый же в работе [3] усовершенствованный метод позволяет создать устройства реализации распределенной нейронной сети, которые функционируют самостоятельно, не завися от главного вычислительного узла. Это порождает вопросы отказоустойчивости и надежности, данные проблемы будут решаться в следующих частях представленного исследования.

Следующий рассмотренный метод не подразумевает непосредственно прямой декомпозиции нейронной сети, а выносит ее выполнение в отдельные вычислительные узлы тумана (Fog Node), этот подход действительно снижает нагрузку на сервер, но требует установки дополнительного оборудования, тех самых узлов туманных вычислений. Предложенный же нашим коллективом метод позволяет реализовать распределенную нейронную сеть после декомпозиции на уже имеющемся в вычислительной сети оборудовании, причем использовать для вычислений даже микроконтроллеры, как, впрочем, и классические процессорные системы.

Третий рассмотренный метод предлагает непосредственно декомпозировать нейронную сеть в соответствии с графом вычислительной системы, на которой будут выполняться нейросетевые расчеты. Достоинства метода: возможность распределить вычислительную нагрузку между узлами, учет задержек в передаче данных между ними. Недостатки: наличие основного узла, на котором нужно сосредоточить входные и выходные нейроны, что для некоторых архитектур будет неоптимальным, помимо этого, без него не будет происходить распределение вычислительных задач между остальными серверами.

В работе [84] предлагается алгоритм уменьшения количества данных, которые поступают с конечных устройств, перед размещением их на центральный сервер. В данной работе исследователи предлагают заменить узлы тумана на туманные станции (или туманные сервера) – менее мощные, чем центральные сервера вычислительные устройства на промежуточном слое. Достоинства предлагаемого метода – возможность уменьшения количества данных, приходящих на центральный сервер и использование для реализации вычислений популярного нейросетевого фреймворка TensorFlow. К недостаткам же относятся потребность в затратах на установку туманных станций и необходимость использовать только нейронные сети, описанные в терминах указанного фреймворка, а также потребность в контейнеризации на Docker и конвейеризации на Kubernetes.

В исследовании [40] авторы предлагают трехуровневую систему, в которой выделены три стадии: обработка на оконечных устройствах, на узлах тумана и на облачном сервере. В рассмотренном методе есть два преимущества. Во-первых, модель, работающая на туманных узлах в этой системе, представляет собой бинарную глубокую нейронную сеть, что позволяет осуществить некоторые классификации еще на этапе туманных узлов. Понятно, что чем больше классификаций произойдет на уровне тумана, тем меньше данных потребуется отправить в облако. Во-вторых, вместо промежуточных данных в облако передаются необработанные данные из узлов тумана, что может помочь разработчику оптимизировать всю систему, то есть помимо промежуточных результатов в облако уходят и исходные параметры, принятые первыми слоями нейронной сети. Недостатки во многом

вытекают из предложенной архитектуры. Во-первых, предлагаемый метод подразумевает, что узел тумана должен быть развернут именно на ПЛИС, чтобы использовать преимущества предложенной BNSC-Net по максимуму, то есть без возможности развернуть на процессорных системах. Во-вторых, в настоящее время BNSCNet применяется только к наборам данных изображений, и его применимость к другим областям, таким как обработка речи и естественного языка, нуждается в дальнейшем изучении. Это описывают в исследовании сами авторы.

В работе [58] рассматривается система распределенных вычислений с поддержкой Fog Computing и сверточных нейронных сетей (CNN) для динамического прогнозирования и оптимизации процессов обработки для решения практических производственных задач. Достоинства:

1. Разработана трехуровневая модель туманных и облачных вычислений для оптимизации использования вычислительных ресурсов и минимизации объема передаваемых данных, благодаря этому система может обойти распространенную проблему низкой эффективности облачных систем, связанную с высокой задержкой передачи данных в промышленных сетях.

2. В данной системе применяются механизмы предварительной обработки сигнала для его очищения и повышения производительности CNN в практических производственных ситуациях.

Недостатки: возможность применения только на сверточных нейронных сетях, потребность установки дополнительных устройств – туманных узлов.

В следующей рассмотренной работе [95] авторы предлагают новый вид нейронных сетей, который называется distributed deep neural network (DDNN), т.е. распределенные глубокие нейронные сети. DDNN отображает обученную DNN на разнородные физические устройства, распределенные локально, на конечные устройства и в облаке. Поскольку DDNN полагается на совместно обученную структуру DNN во всех частях нейронной сети, как для обучения, так и для вывода, многие сложные инженерные решения значительно упрощаются. Достоинства рассмотренного метода: возможность уменьшить число данных, которые необходимо передать на уровень облачных вычислений, возможность параллельной об-

работки данных на нескольких оконечных устройствах или нескольких узлах тумана. Недостатки метода, предлагаемого в работе: в настоящее время все слои в DDNN являются двоичными. Хотя двоичные слои являются обязательным требованием для конечных устройств из-за ограниченного пространства на устройствах, в облаке это необязательно, коллектив исследователей хочет в будущем позволить использовать в облаке и в тумане слои со смешанной точностью или с плавающей запятой. Помимо этого, в силу возможности параллельной обработки входящих данных несколькими оконечными устройствами и туманными узлами, может возникнуть узкое место на облачном сервере, что сведет на нет все усилия, которые были направлены на снижение нагрузки, которая приходится на облачный сервер. Поэтому при использовании данного метода требуется проявлять осторожность в процессе увеличения степени параллелизма вычислений.

Работу [49] следует рассматривать скорее, как обзорную, поскольку там не предлагается нового метода создания распределенных нейронных сетей, но в ней рассматриваются основные положения современных подходов к реализации туманных вычислений, а именно «fat edge devices» и «thin edge devices» модели. Первое предполагает, что конечные устройства должны быть автономными, а все потенциальные прогнозы или решения должны производится с сокращением времени и на стороне устройства, когда требуется подключение к облаку. Второе направление — это «thin edge devices», которые предполагают тесно связанную облачную архитектуру, а это значит, что облако по-прежнему будет играть основную роль в обработке новых данных. Достоинство первой модели – возможность принимать решения без взаимодействия с облаком и обрабатывать данные на месте, недостаток – потребность в наличии достаточных вычислительных мощностей для этого на оконечном устройстве. Для второй модели достоинства и недостатки инвертируются, то есть плюс в том, что можно использовать маломощные и дешевые оконечные устройства, в то же время требуется хороший канал связи и возможность ждать принятия решения в облаке. Следует ожидать разработки гибридных решений, предназначенных для туманных вычислений, а затем и для периферийных вычислений. Произойдет увеличение числа небольших центров обработки данных, расположенных близко к источникам данных.

Таким образом, противоречие в науке заключается в том, что несмотря на обилие методов реализации распределенных нейронных сетей и синтеза нейросетевых устройств, не в полной мере разработаны методы синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления, учитывающие параметры: мощность устройств, оптимальный объем передаваемых между устройствами данных, возможность учета пропорциональности блоков нейронной сети по слоям или по нейронам, а также имеет возможность реализации функционального резервирования, что позволило бы снизить нагрузку на те вычислительные узлы, которые требуется разгрузить в рамках задачи и продолжать работу даже в случае отказа или сбоя части устройств в каскаде. В результате исследования планируется получить математическую модель нейронной сети и метод деления нейронной сети на блоки, которые будут работать на оконечных устройствах, также будет проведено испытание данного метода на тестовом каскаде вычислительных устройств. Данный подход важен тем, что позволяет отойти от оптимизации сети в пользу разделения вычислительной нагрузки между устройствами [5]. Поэтому научная задача исследования – разработать метод синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления.

1.4. Постановка научной задачи исследования

Таким образом необходимо разработать метод синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления. Для его реализации необходимо создать математическую модель искусственной нейронной сети, ориентированной на туманные вычисления, на которой возможно предварительно рассчитать параметры будущего каскада вычислительных устройств. Последовательность блоков модели нейронной сети совместно с соответствующими алгоритмами, является результатом синтеза нейросетевых устройств для режима реализации туманных вычислений.

Входными данными для предлагаемого метода синтеза устройств нейросетевых устройств для реализации режима Fog Computing являются исходная обученная монолитная нейронная сеть X, количество устройств, на которые необхо-

димо разделить нейронную сеть D, массив вычислительных мощностей устройств $\{P_0,\ldots,P_{D-1}\}$ и способ разделения на блоки (учитывать пропорциональность по слоям нейронной сети или по количеству нейронов в блоке).

Метод должен обеспечивать поиск оптимального по стоимости, энергопотреблению, времени выполнения нейросетевых вычислений и времени выполнения основных вычислений, решения для декомпозиции нейронной сети на каскад распределенных вычислителей при заданных ограничениях на выбранную архитектуру вычислительной системы (число доступных вычислительных устройств n), их вычислительные мощности $\bar{p}$ и пропускной способности каналов связи q.

1. Оценка стоимости $\boldsymbol{C}$**:**

$$C=\left\{C_{\tau_1}\left(n,\bar{p},q\right),C_{\tau_2}\left(n,\bar{p},q\right),\ldots.C_{\tau_e}\left(n,\bar{p},q\right)\right\} \quad \textbf{(1)}$$

2. Оценка времени выполнения нейросетевых вычислений $\boldsymbol{T^1}$**:**

$$T^1=\left\{T^1_{\tau_1}\left(n,\bar{p},q\right),T^1_{\tau_2}\left(n,\bar{p},q\right),\ldots,T^1_{\tau_e}\left(n,\bar{p},q\right)\right\} \quad \textbf{(2)}$$

3. Оценка среднего времени отклика пульта на входные сигналы $\boldsymbol{T^2}$**:**

$$T^2=\left\{T^2_{\tau_1}\left(n,\bar{p},q\right),T^2_{\tau_2}\left(n,\bar{p},q\right),\ldots,T^2_{\tau_e}\left(n,\bar{p},q\right)\right\} \quad \textbf{(3)}$$

4. Оценка итогового энергопотребления (мощности) $\boldsymbol{W}$:

$$W=\left\{W_{\tau_1}\left(n,\bar{p},q\right),W_{\tau_2}\left(n,\bar{p},q\right),\ldots.W_{\tau_e}\left(n,\bar{p},q\right)\right\} \quad \textbf{(4)}$$

При получении оценок необходимо учесть существующие ограничения современных вычислительных устройств. Для подтверждения работоспособности методов выполнить схемотехническое и физическое моделирование решений (декомпозиций) $\Omega=\{\omega_1,\omega_2,\ldots\omega_u\}$.

Получить оптимальный набор H элементов, используя метод Парето оптимизации:

$$H=\left\langle\langle\omega_1(n,\bar{p},q)\rangle,\langle\omega_2(n,\bar{p},q)\rangle,\ldots\langle\omega_u(n,\bar{p},q)\rangle\right\rangle$$

такой, что $C(H)\to min, T^2(H)\to min$ *без ухудшения* $C(H)$,

$T^1(H)\to \min$ *без ухудшения* $C(H)$ *и* $T^2(H)$,

$W(H)\to \min$ *без ухудшения* $C(H), T^2(H)$ *и* $T^1(H)$.

Необходимо осуществить декомпозицию нейронной сети на несколько устройств с поиском оптимума по параметрам: стоимость, тепловыделение, энергопотребление (при прочих равных приоритет будет отдан варианту, в котором время выполнения будет лучшим).

Находить искомые параметры мы будем по следующим формулам:

1. Оценка стоимости ***C*:**

$$C = \sum_{i=0}^{n-1} C_i \tag{5}$$

2. Оценка времени выполнения нейросетевых вычислений T^1**:**

$$T^1 = \sum_{i=0}^{n-1} T_{i_i}^{exec} + \sum_{j=1}^{n-1} T_j^{send}$$

$$T_{i_i}^{send} = \frac{I}{q}$$

$$T^1 = \sum_{i=0}^{n-1} T_{i_i}^{exec} + \sum_{j=1}^{n-1} \frac{I_j}{q_j}$$

$$T_{i_i}^{exec} = \frac{N}{\nu}$$

$$T^1 = \sum_{i=0}^{n-1} \frac{N_i}{\nu_i} + \sum_{j=1}^{n-1} \frac{I_j}{q_j} \tag{6}$$

3. Оценка среднего времени отклика пульта на входные сигналы $\boldsymbol{T^2}$**:**

$$T^2 = \overline{T_{pult}} + T_{pult}^{exec}$$

4. Оценка итогового энергопотребления (мощности) ***W***:

$$W = \sum_{i=0}^{n-1} W_i \tag{7}$$

Для этого требуется разработать комплекс алгоритмов для декомпозиции нейронной сети на блоки, для работы распределенной нейронной сети и для поиска оптимальной по предложенным оценкам декомпозиции. Для практической реализации требуется разработать программное обеспечение, реализующие эти алгоритмы. Кроме того, необходимо выполнить проверку результатов синтеза с помощью схемотехнического моделирования и прототипирования.

С целью построения активно отказоустойчивых вычислительных устройств нейронных сетей в режиме туманных вычислений с заранее выбранной глубиной адаптации G требуется модифицировать метод синтеза нейросетевых устройств для реализации туманных вычислений.

Метод должен позволять осуществить декомпозицию нейронной сети и синтез нейросетевых устройств, учитывая вычислительные возможности каждого из устройств, а также особенности архитектуры и структуры слоев разделяемой нейронной сети, за счет разделения с учетом нагрузки, распределяемой по числу нейронов на устройство, а не просто слоев нейронной сети.

1.5. Выводы по главе

Как видно из анализа предметной области, различные научные коллективы ведут актуальные исследования в данной области, у рассматриваемой проблемы уже существуют различные методы решения, но они позволяют решить поставленную нами задачу не в полной мере. Однако, с учетом ограничений, которые выставляются в нашей работе, и необходимых для этого усовершенствований, рассматриваемая задача еще не была решена.

Исходя из представленных данных было решено предложить усовершенствованный метод синтеза устройств нейросетевого распознавания для реализации режима Fog Computing. Предлагаемым усовершенствованием существующего метода декомпозиции будет расширение гибкости настройки получаемой декомпозированной блочной нейронной сети, за счет различных способов декомпозиции, которые определяются входными параметрами декомпозиции, изменение этих параметров будет приводить к получению распределенных нейронных сетей с различными свойствами, а именно: различным объемом проводимых на конкретных устройствах нейросетевых вычислений и объемом данных, передаваемых между устройствами.

Предлагаемое усовершенствование полезно не только для случая, когда вычислительная система создается с нуля, в этом случае степеней свободы у того, кто ее проектирует достаточно много, но и в случае, когда требуется усовершен-

ствовать и расширить обработку данных уже существующей вычислительной системы с функционирующей архитектурой, в первую очередь физического уровня. Когда встает задача дополнить нейросетевыми вычислениями существующую систему с законченной архитектурой и требуется развернуть на имеющих физических устройствах распределенную нейронную сеть, но большинство современных систем гетерогенны и мультивендорны, то есть у входящих в систему вычислительных устройств различные вычислительные мощности, скорость обработки данных и другие параметры. За счет изменения параметров декомпозиции нейронной сети у разработчика появится возможность вынести больше слоев или нейронов на определенный вычислительный узел, тем самым устранив узкое место в системе потокового обслуживания, которой по сути и является каскад устройств, реализующий нейросетевые вычисления в рассматриваемых случаях. Помимо этого, предлагаемый метод обеспечит отказоустойчивость вычислений на всем каскаде благодаря тому, что позволяет реализовать вычисления узла, на котором произошел отказ или сбой, на соседнем узле, на который заранее будет занесена информация о блоке блочной нейронной сети, которую выполняет предыдущий узел. Похожим образом может быть организована диагностика и реконфигурация каскада устройств, для замещения более чем одного соседнего вычислительного узла.

Рассматриваемый метод позволит найти оптимальное по стоимости, энергопотреблению, времени выполнения нейросетевых вычислений и времени выполнения основных вычислений, решение для декомпозиции нейронной сети на каскад распределенных вычислителей при заданных ограничениях на выбранную архитектуру вычислительной системы (число доступных вычислительных устройств n), их вычислительные мощности $\bar{p}$ и пропускную способность каналов связи q. Для демонстрации этого потребуется решить практическую задачу, где необходимо применение усовершенствованного метода.

ГЛАВА 2. РАЗРАБОТКА МАТЕМАТИЧЕСКОЙ МОДЕЛИ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ, ОРИЕНТИРОВАННОЙ НА ТУМАННЫЕ ВЫЧИСЛЕНИЯ

2.1. Разработка модели работы искусственной нейронной сети с архитектурой FFNN, ориентированной на туманные вычисления

В первую очередь были сформулированы ограничения, в рамках которых будет работать предлагаемая математическая модель. Предлагаемая математическая модель отображает многослойные нейронные сети, полная нейронная сеть может быть описана с ее помощью, но для этого часть связей потребуется искусственно назначить обратными. Разработаны математические модели нейронных сетей со следующими архитектурами: FFNN, сверточные и рекуррентные нейронные сети. Математическая модель способна отображать нейронные сети на различных этапах их жизненного цикла, но для корректной работы метода синтеза устройств требуется, чтобы к моменту декомпозиции нейронные сети уже были обучены. Математическая модель может отображать нейронные сети независимо от того, для решения каких именно задач они будут использованы, например, для распознавания, управления, классификации, генерации и других задач, которые решаются нейросетевыми методами. Подробнее о предлагаемой математической модели можно прочитать в работе [5].

Дано: монолитная многослойная нейронная сеть X, результат работы которой – последовательность сигналов $\left\{y_0^K,...,y_{H_K}^K\right\}$.

Найти: последовательность нейронных сетей $\left\{\overline{X}_0,...,\overline{X}_{D-1}\right\}$, где результат вычисления $\overline{X}_0 \rightleftarrows \overline{X}_1 \rightleftarrows \ldots \rightleftarrows \overline{X}_{D-1}$ совпадает с результатом работы сети X.

Процесс преобразования монолитной ИНС в каскад блочных ИНС, результат вычисления которого совпадает с результатами монолитной нейронной сети назовем **декомпозицией** искусственной нейронной сети.

Подробнее построение математической модели описано в [2, 5, 83]. Начнем создание математической модели искусственной нейронной сети, ориентирован-

ной на туманные вычисления, с того, что воспользуемся общепринятыми определениями функционирования нейронных сетей и обозначим их за начальные точки исследования.

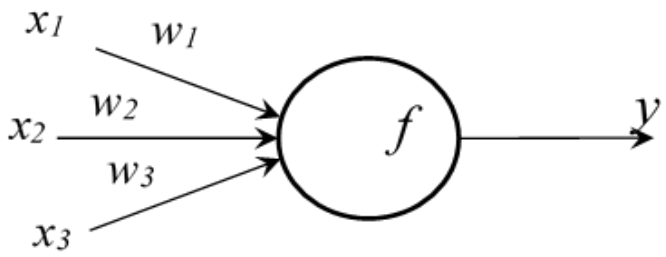


Рисунок 12 – Математический нейрон

Математический нейрон представляет собой взвешенный сумматор (Рисунок 12), единственный выход которого определяется через его входы и матрицу весов следующим образом [102]:

$$y = f(u),\ u = \sum_{j=0}^{n} w_j \cdot x_j, \tag{8}$$

где y – выход математического нейрона, $f(u)$ – передаточная функция, u – взвешенная сумма сигналов на входах нейрона, x_j – вход синапса, w_j – вес синапса.

Тогда функция работы ИНС на примере многослойного персептрона [35]:

$$\begin{cases} y_i^{(k)} = f_i^{(k)}\left(\sum_{j=0}^{H_{k-1}} w_{ij}^{(k)} \cdot y_i^{(k-1)}\right) \\ y_i^{(0)} = x_i^{(0)} \end{cases} \tag{9}$$

где $x_i^{(0)}$ – входные данные ИНС, $y_i^{(k)}$ – выход i-ого нейрона k-ого слоя, H_{k-1} – число нейронов на слое $k-1$, $f_i^{(k)}$ – функция активации i-ого нейрона k-ого слоя.

Математическая модель монолитной нейронной сети в обобщенном схематическом виде представлена на Рисунке 13.

Разделение на подсети может осуществляться с различными параметрами. Начнем с того, что рассмотрим простой пример деления монолитной нейронной сети на блочные нейронные сети с одинаковым количеством слоев в каждой из

них. Количество слоев K, количество устройств D. Количество подсетей: D, по числу устройств. Для случая, когда $K \geq D$ и K кратно D, получим следующие рекуррентные формулы блочных нейронных сетей для каждого из устройств:

$$\overline{X_0}: \begin{cases} y_i^{\left(\frac{K}{D}\right)} = f_i^{\left(\frac{K}{D}\right)} \left(\sum_{j=0}^{H_{\frac{K}{D}-1}} w_{ij}^{\left(\frac{K}{D}\right)} \cdot y_i^{\left(\frac{K}{D}-1\right)} \right) \\ \dots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

$$\overline{X_1}: \begin{cases} y_i^{\left(\frac{2K}{D}\right)} = f_i^{\left(\frac{2K}{D}\right)} \left(\sum_{j=0}^{H_{\frac{2K}{D}-1}} w_{ij}^{\left(\frac{2K}{D}\right)} \cdot y_i^{\left(\frac{2K}{D}-1\right)} \right) \\ \dots \\ y_i^{\left(\frac{K}{D}+1\right)} = y_i^{\left(\frac{K}{D}\right)} \end{cases}$$

$$\overline{X_{D-1}}: \begin{cases} y_i^{(K)} = f_i^{(K)} \left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)} \right) \\ \dots \\ y_i^{\left(\frac{(D-1)K}{D}+1\right)} = y_i^{\left(\frac{(D-1)K}{D}\right)} \end{cases}$$

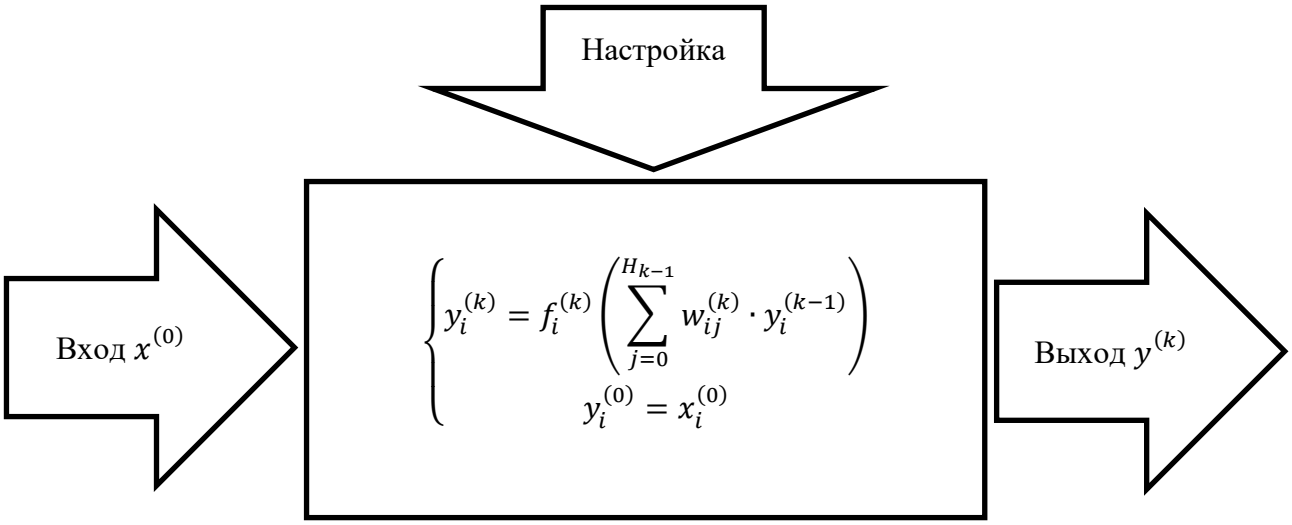


Рисунок 13 – Математическая модель монолитной нейронной сети

Обобщив данные формулы, получим общую рекуррентную формулу описания каждой из полученных блочных нейронных сетей:

$$\overline{X_N}:\begin{cases} y_i^{\left(\frac{(N+1)K}{D}\right)} = f_i^{\left(\frac{(N+1)K}{D}\right)}\left(\sum_{j=0}^{H_{\frac{(N+1)K}{D}-1}} w_{ij}^{\left(\frac{(N+1)K}{D}\right)} \cdot y_i^{\left(\frac{(N+1)K}{D}-1\right)}\right) \\ \dots \\ \begin{cases} y_i^{\left(\frac{NK}{D}+1\right)} = y_i^{\left(\frac{NK}{D}\right)}, if\ N>0 \\ y_i^{(0)} = x_i^{(0)}, if\ N=0 \end{cases} \end{cases}$$

Ответ: общая рекуррентная формула описания каждой из полученных блочных нейронных сетей, где массив $\{L_0,...,L_{D-1}\}$ – количество слоев нейронной сети, которые должны быть переданы на устройство с номером *N*:

$$\overline{X}_N:\begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right), N = D-1 \\ y_i^{\left(\sum_{k=0}^{N} L_k\right)} = f_i^{\left(\sum_{k=0}^{N} L_k\right)}\left(\sum_{j=0}^{H_{\sum_{k=0}^{N} L_k - 1}} w_{ij}^{\left(\sum_{k=0}^{N} L_k\right)} \cdot y_i^{\left(\sum_{k=0}^{N} L_k - 1\right)}\right), N < D-1 \end{cases} \\ \dots \\ \begin{cases} y_i^{(L_{N-1}+1)} = y_i^{(L_{N-1})}, N>0 \\ y_i^{(0)} = x_i^{(0)}, N=0 \end{cases} \end{cases} \quad (10)$$

Математическая модель монолитной нейронной сети в обобщенном схематическом виде представлена на Рисунках 14 и 15.

На Рисунках 14 и 15 видно, что в зависимости от входных параметров и выбранного способа декомпозиции, объем блоков в слоях или в нейронах может варьироваться, что позволяет получить наиболее оптимальный вариант декомпозиции для каждой рассматриваемой задачи.

Предлагаемой формулой также возможно пользоваться для отображения сверточных нейронных сетей, для этого необходимо представить сверточную

часть нейронной сети как последовательность одинаковых слоев обработки данных, которые древовидно сужаются от полной размерности рассматриваемой области данных к итоговой размерности конечного ядра свертки [36, 78]. Однако, разработанная математическая модель требует уточнения для альтернативной архитектуры нейронной сети – рекуррентной нейронной сети.

Предложенная математическая модель является аналитической. Научной новизной предлагаемой модели является возможность заменить модель одной монолитной нейронной сети на каскад моделей нескольких блочных нейронных сетей, которые связаны между собой, и возможность балансировать размеры блоков в зависимости от исходных параметров каскада вычислительных устройств, который будет реализовывать распределенную нейронную сеть.

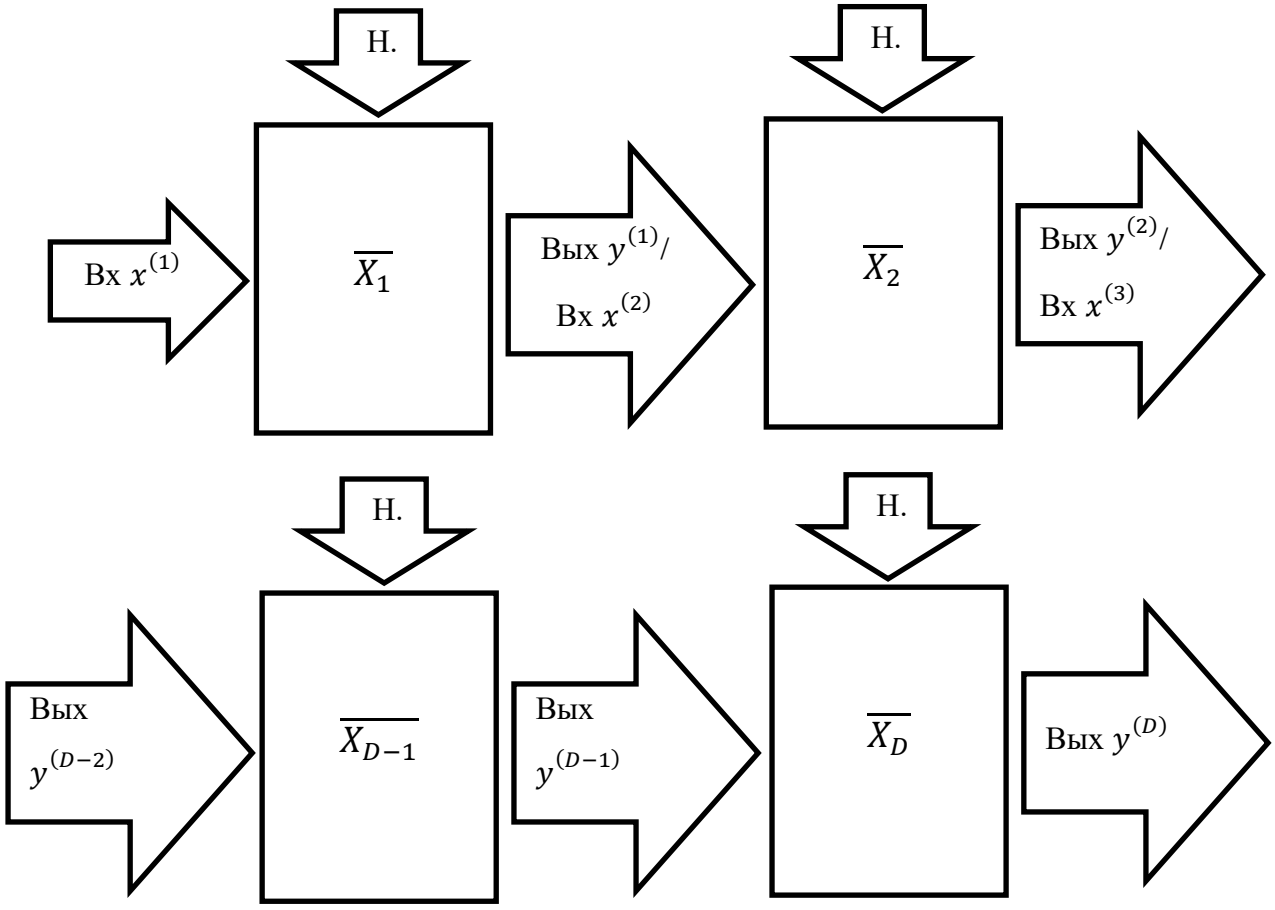


Рисунок 14 – Математическая модель блочной нейронной сети, декомпозированной способом с равномерным распределением слоев по блокам

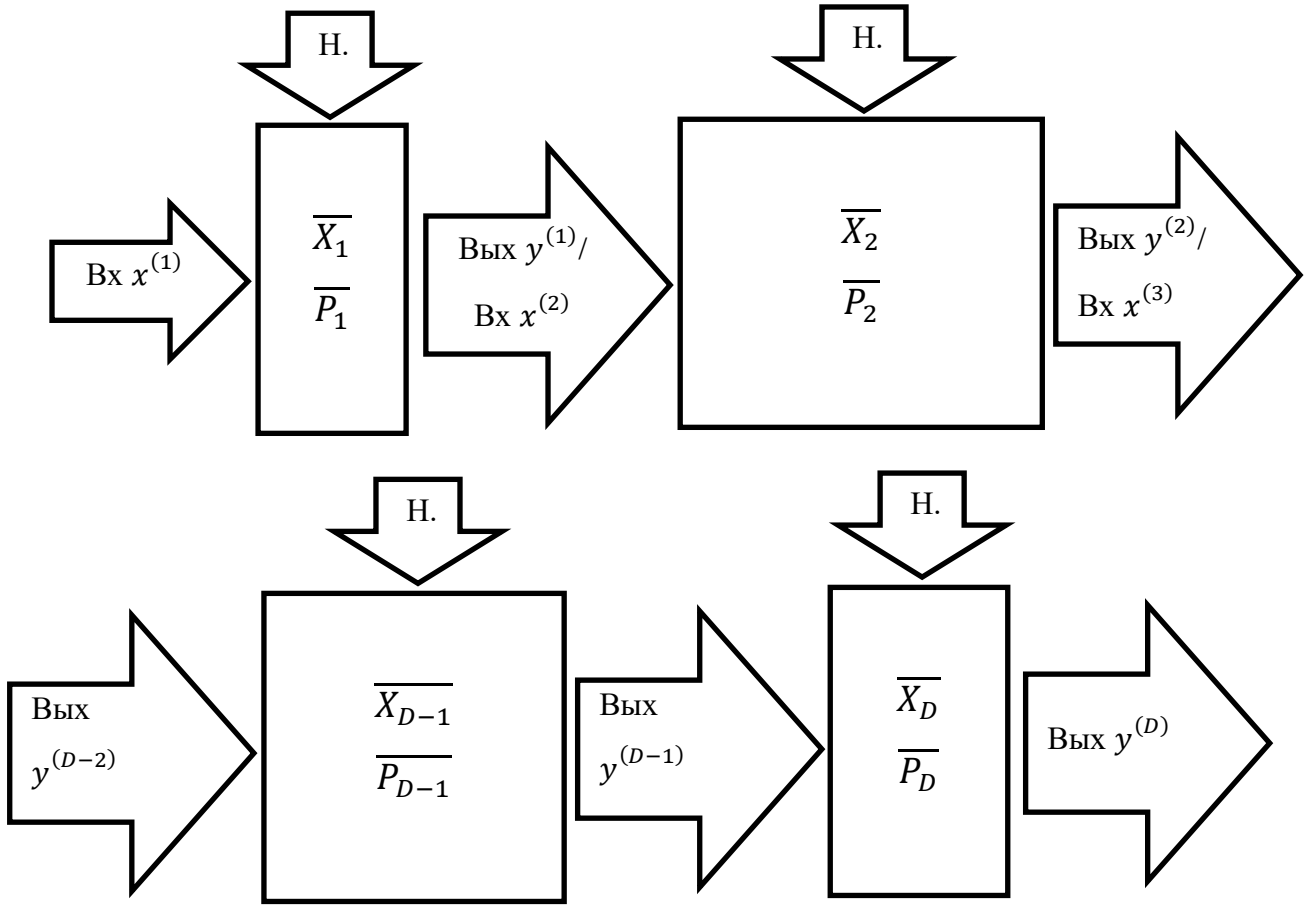


Рисунок 15 – Математическая модель блочной нейронной сети, декомпозированной способом с пропорциональным распределением слоев по блокам

2.2. Моделирование работы рекуррентной нейронной сети, ориентированной на туманные вычисления

Функция работы рекуррентной нейронной сети непосредственно следует из функции работы нейронной сети с прямым распространением сигнала (1). Добавляется обратная связь в виде произведения матрицы весов для обратных связей и выходных сигналов с последнего цикла работы нейросети:

$$\begin{cases} y_i^{(K)} = f_i^{(K)}\left(\begin{array}{l}\sum\limits_{j=0}^{H_{K-1}-R_K}(w_{ij}^{(K)}\cdot y_i^{(K-1)})+ \\ \sum\limits_{m=0}^{R_K}(w_{im}^{(K)}\cdot y_i^{(K+r_{im})})\end{array}\right), \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

где $x_i^{(0)}$ – входные данные нейронной сети, $y_i^{(K)}$ – выход нейрона с номером i на слое с номером K, H_{K-1} – число нейронов на слое $K-1$, $f_i^{(K)}$ – функция активации нейрона с номером i на слое с номером K, R_K – число рекуррентных связей на слое K, r_{im} – количество слоев на которое нужно сместиться для получения обратной связи относительно слоя K, $y_i^{(K+r_{im})}$ – выход нейрона с номером i на слое номер $K + r_{im}$.

Рассмотрим деление рекуррентной нейронной сети на блоки так, чтобы в каждом блоке было равное количество слоев. Число слоев нейронной сетиK, число устройств D. Количество подсетей: D, по числу устройств. Для случая, когда $K \geq D$ и K кратно D, получаем следующие формулы блочных рекуррентных нейронных сетей для каждого из устройств:

$$\overline{X_0}: \begin{cases} y_i^{\left(\frac{K}{D}\right)} = f_i^{\left(\frac{K}{D}\right)} \left(\begin{array}{l} \sum\limits_{j=0}^{H_{\frac{K}{D}-1}-R_{\frac{K}{D}}} w_{ij}^{\left(\frac{K}{D}\right)} \cdot y_i^{\left(\frac{K}{D}-1\right)} + \\ \sum\limits_{m=0}^{R_{\frac{K}{D}}} (w_{im}^{\left(\frac{K}{D}\right)} \cdot y_i^{\left(\frac{K}{D}+r_{im}\right)}) \end{array} \right) \\ \cdots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

$$\overline{X_1}: \begin{cases} y_i^{\left(\frac{2K}{D}\right)} = f_i^{\left(\frac{2K}{D}\right)} \left(\begin{array}{l} \sum\limits_{j=0}^{H_{\frac{2K}{D}-1}-R_{\frac{2K}{D}}} w_{ij}^{\left(\frac{2K}{D}\right)} \cdot y_i^{\left(\frac{2K}{D}-1\right)} + \\ \sum\limits_{m=0}^{R_{\frac{2k}{D}}} (w_{im}^{\left(\frac{2K}{D}\right)} \cdot y_i^{\left(\frac{2K}{D}+r_{im}\right)}) \end{array} \right), \\ \cdots \\ y_i^{\left(\frac{K}{D}+1\right)} = y_i^{\left(\frac{K}{D}\right)} \end{cases}$$

$$\overline{X_{D-1}}: \begin{cases} y_i^{(K)} = f_i^{(K)} \left(\begin{array}{l} \sum\limits_{j=0}^{H_{K-1}-R_K} w_{ij}^{(K)} \cdot y_i^{(K-1)} + \\ \sum\limits_{m=0}^{R_k} (w_{im}^{(K)} \cdot y_i^{(K+r_{im})}) \end{array} \right), \\ \cdots \\ y_i^{\left(\frac{(D-1)K}{D}+1\right)} = y_i^{\left(\frac{(D-1)K}{D}\right)} \end{cases}$$

Обобщим эти формулы и получим общую формулу для описания каждой из полученных блочных рекуррентных нейронных сетей:

$$\overline{X_N}:\begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum\limits_{j=0}^{H_{K-1}-R_K} w_{ij}^{(K)} \cdot y_i^{(K-1)} + \sum\limits_{m=0}^{R_k}(w_{im}^{(K)} \cdot y_i^{(K+r_{im})})\right), \\ \cdots \\ y_i^{\left(\frac{NK}{D}+1\right)} = y_i^{\left(\frac{NK}{D}\right)} \end{cases} \tag{11}$$

Тогда, если обозначить послойные размерности блочных рекуррентных нейронных сетей в виде массива $\{L_0,\ldots,L_{D-1}\}$, то общая формула описания каждой из полученных блочных нейронных сетей имеет вид:

$$\overline{X_N}:\begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum\limits_{j=0}^{H_{K-1}-R_K} w_{ij}^{(K)} \cdot y_i^{(K-1)} + \sum\limits_{m=0}^{R_K}(w_{im}^{(K)} \cdot y_i^{(K+r_{im})})\right), \\ if\ N = D-1 \end{cases} \\ \begin{cases} y_i^{\left(\sum_{k=0}^{N} L_k\right)} = f_i^{\left(\sum_{k=0}^{N} L_k\right)}\left(\sum\limits_{j=0}^{H_{\sum_{k=0}^{N} L_k - 1} - R_{\sum_{k=0}^{N} L_k}} w_{ij}^{\left(\sum_{k=0}^{N} L_k\right)} \cdot y_i^{\left(\sum_{k=0}^{N} L_k - 1\right)} + \sum\limits_{m=0}^{R_{\sum_{k=0}^{N} L_k}} (w_{im}^{\left(\sum_{k=0}^{N} L_k\right)} \cdot y_i^{\left(\sum_{k=0}^{N} L_k + r_{im}\right)})\right), \\ if\ N < D-1 \end{cases} \\ \cdots \\ \begin{cases} y_i^{(L_{N-1}+1)} = y_i^{(L_{N-1})}, if\ N > 0 \\ y_i^{(0)} = x_i^{(0)}, if\ N = 0 \end{cases} \end{cases} \tag{12}$$

Для случая, когда K не кратно D, число слоев L_i, которые нужно перенести на устройство D_i рассчитывается по формуле:

$$L_i = \frac{K}{D},$$

Так как для получения целого используется округление в меньшую сторону, то количество слоев нейронной сети, которое будет на последнем вычислительном устройстве с номером $D-1$:

$$L_{D-1} = K - (D-1)L_0.$$

Таким образом, мы получаем общую формулу описания каждой из полученных блочных рекуррентных нейронных сетей:

$$\overline{X_N}: \begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum\limits_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)} + \sum\limits_{m=0}^{R_K} (w_{im}^{(K)} \cdot y_i^{(K+r_{im})})\right), \ if \ N = D-1 \\ y_i^{\left((N+1)\frac{K}{D}\right)} = f_i^{\left((N+1)\frac{K}{D}\right)}\left(\sum\limits_{j=0}^{H_{(N+1)\frac{K}{D}-1}} w_{ij}^{\left((N+1)\frac{K}{D}\right)} \cdot y_i^{\left((N+1)\frac{K}{D}-1\right)} + \sum\limits_{m=0}^{R_{(N+1)\frac{K}{D}}} (w_{im}^{\left((N+1)\frac{K}{D}\right)} \cdot y_i^{\left((N+1)\frac{K}{D}+r_{im}\right)})\right) \\ \quad if \ N < D-1 \end{cases} \\ \begin{cases} y_i^{\left(N\frac{K}{D}+1\right)} = y_i^{\left(N\frac{K}{D}\right)}, \ if \ N > 0 \\ y_i^{(0)} = x_i^{(0)}, \ if \ N = 0 \end{cases} \end{cases} \quad (13)$$

2.3. Доказательство адекватности предлагаемой математической модели искусственной нейронной сети, ориентированной на туманные вычисления

Существование блочной нейронной сети, модель которой разработана в параграфе 2.1, требует отдельного доказательства. В процессе диссертационного исследования была сформулирована и доказана следующая теорема.

В рамках подтверждения адекватности предлагаемой математической модели потребуется доказать определенные теоретические предположения, в первую очередь требуется доказать следующую теорему.

Теорема 1. О существовании блочной нейронной сети. Для любого $D \leq K$ существует последовательность нейронных сетей $\{\bar{X}_0, \ldots, \bar{X}_{D-1}\}$, называемых блочными нейронными сетями, такая, что результат последовательного их вычисления, т.е. когда выходное значение $y_i^{(n)}$ нейронной сети $\bar{X}_n$ является входным значением $x_i^{(n+1)}$ нейронной сети $\bar{X}_{n+1}$ для $n \in [1, \mathrm{D} - 1]$, совпадает с результатом работы исходной сети X для всех $x_i^{(0)} \in \mathbb{R}$.

База индукции. D = 1. То есть исходная нейронная и есть единственный элемент последовательности, т.е. вырождается сама в себя. Соответственно результат работы такой последовательности из одного блока равен результату работы исходной нейронной сети.

Индукционный переход. Пусть утверждение справедливо для D = n. Тогда исходная сеть X декомпозируется на последовательность нейронных сетей $\{\bar{X}_0, \ldots, \bar{X}_{n-1}\}$.

Докажем справедливость утверждения для D = n+1. Возьмем последовательность $\{\bar{X}_0, \ldots, \bar{X}_{n-1}\}$ и разделим в ней нейронную сеть, в которой не менее двух слоев, на две нейронные сети. Таким образом получаем последовательность $\{\bar{X}_0, \ldots, \bar{X}_{n+1-1}\}$, которая в конечном итоге будет записана в виде $\{\bar{X}_0, \ldots, \bar{X}_n\}$. Если же среди нейронных сетей последовательности $\{\bar{X}_0, \ldots, \bar{X}_{n-1}\}$ не нашлось ни одной нейронной сети с числом слоев k > 1, то это означает, что n = K. Из этого следует то, что достигнуто предельно допустимое состояние декомпозиции нейронной сети.

Таким образом, блочная нейронная сеть существует в любом случае, кроме тривиального случая нейронной сети из одного слоя.

Однако, существование блочной нейронной сети не гарантирует ее эквивалентности монолитной нейронной сети, что доказывается предлагаемой Теоремой 2. Выполним доказательство методом от противного.

Процесс преобразования каскада блочных ИНС в монолитную ИНС, результат вычисления которой совпадает с результатами блочной нейронной сети назовем **композицией** искусственной нейронной сети.

Теорема 2. Об эквивалентности монолитной и блочной нейронных сетей. Результаты работы монолитной нейронной сети X и последовательности блочных нейронных сетей $\{\bar{X}_0, \dots, \bar{X}_{D-1}\}$, полученной декомпозицией монолитной нейронной сети X, равны для всех входных значений $x_i^{(0)} \in \mathbb{R}$.

Доказательство: в соответствии с алгоритмом декомпозиции последний слой монолитной нейронной сети X с номером K и последний слой нейронной сети $\bar{X}_{D-1}$ представляют собой один и тот же слой, что, очевидно, позволяет говорить об одинаковой размерности выходных векторов $\left\{y_0^K, \dots, y_{H_K-1}^K\right\}$ и $\left\{y_0^{L_{D-1}}, \dots, y_{H_K-1}^{L_{D-1}}\right\}$. Аналогичным образом можно утверждать, что входные вектора для блочной и монолитной нейронной сети имеют одинаковую размерность H_0.

Предположим противное, что на одинаковых исходных данных $\left\{x_0^0, \dots, x_{H_0-1}^0\right\}$ монолитная нейронная сеть X и последовательность нейронных сетей $\{\bar{X}_0, \dots, \bar{X}_{D-1}\}$ могут дать различающиеся выходные векторы $\left\{y_0^K, \dots, y_{H_K-1}^K\right\}$ и $\left\{y_0^{L_{D-1}}, \dots, y_{H_K-1}^{L_{D-1}}\right\}$.

Общая рекуррентная формула монолитной нейронной сети выглядит следующим образом:

$$X: \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right) \\ \dots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

Количество слоев K, количество устройств D. Количество подсетей: D, по числу устройств. Для случая, когда $K \geq D$ и K кратно D, получим следующие рекуррентные формулы блочных нейронных сетей для каждого из устройств представлены выше.

Далее начнем осуществлять композицию блочных нейронных сетей, объединим $\bar{X}_0$ и $\bar{X}_1$, в результате композиции получим нейронную сеть:

$$\overline{X_{01}}: \begin{cases} y_i^{\left(\frac{2K}{D}\right)} = f_i^{\left(\frac{2K}{D}\right)}\left(\sum_{j=0}^{H_{\frac{2K}{D}-1}} w_{ij}^{\left(\frac{2K}{D}\right)} \cdot y_i^{\left(\frac{2K}{D}-1\right)}\right) \\ \dots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

Тогда композиция последовательности блочных нейронных сетей $\{\bar{X}_0,\dots,\bar{X}_{D-1}\}$ будет представлять собой следующую нейронную сеть:

$$\overline{X_{0(D-1)}}:\begin{cases} y_i^{\left(\frac{DK}{D}\right)} = f_i^{\left(\frac{DK}{D}\right)}\left(\sum_{j=0}^{H_{\frac{DK}{D}-1}} w_{ij}^{\left(\frac{DK}{D}\right)} \cdot y_i^{\left(\frac{DK}{D}-1\right)}\right) \\ \dots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

После несложных преобразований получим следующий вид нейронной сети $\overline{X_{0(D-1)}}$:

$$\overline{X_{0(D-1)}}:\begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right) \\ \dots \\ y_i^{(0)} = x_i^{(0)} \end{cases}$$

Таким образом, очевидно, что исходная монолитная нейронная сеть X и результат композиции последовательности блочной нейронной сети $\overline{X_{0(D-1)}}$ представляют собой одну и ту же нейронную сеть. На одном наборе входных данных представленные нейронные сети дадут одинаковые результаты вычислений, но по нашему предположению эти сети могут дать различные результаты вычислений. Возникает противоречие, а значит верно обратное, что и требовалось доказать.

2.4. Адекватность математической модели

Для проверки адекватности математических моделей применяются различные методики, например, критерий Фишера, который позволяет сравнивать величины выборочных дисперсий двух рядов наблюдений [16, 33], критерий Пирсона, критерий Колмогорова и другие статистические критерии. Для предлагаемой математической модели указанные критерии не подходят, поскольку сравнивается не реальный объект и его математическая модель, а в сущности – две математические модели, ведь монолитная нейронная сеть изначально является математической абстракцией, эвристическим алгоритмом. Поэтому для проверки адекватности предлагаемой математической модели было решено использовать другую методику.

Возьмем монолитную нейронную сеть, у которой существуют входные параметры, которые могут принимать конечное число значений для каждого из параметров. Декомпозируем нейронную сеть в каскад блоков блочной нейронной сети, используя предлагаемую математическую модель. И для проверки соответствия блочной и монолитной нейронных сетей подадим на вход каждой из них все возможные наборы входных параметров. Тогда, если выходные значения исходной нейронной сети и сети, полученной декомпозицией с использованием предлагаемой математической модели совпадут для всех возможных наборов параметров, подаваемых на вход, это и станет для предлагаемой модели подтверждением ее адекватности.

Для доказательства адекватности математической модели возьмем небольшую нейронную сеть из девяти слоев и декомпозируем ее для выполнения на три вычислительных блока, используем способ декомпозиции с равномерным распределением числа слоев на каждый из вычислителей, получим три блока по три слоя исходной нейронной сети в каждом.

Рассматриваемая нейронная сеть использовалась для определения уровня субъективного одобрения погодных условий пользователем исходя из объективных показателей данных о текущих погодных условиях. Входными данными для рассматриваемой нейронной сети выступали: температура воздуха, влажность, скорость ветра, атмосферное давление, и облачность. Каждый из входных параметров был формализован таким образом, чтобы принимать конечное число значений. Температура попадала в один из девяти диапазонов, давление в один из шести, скорость ветра – пяти диапазонов, влажность – семи диапазонов и параметр «облачность и осадки» мог принимать одно из одиннадцати состояний. Таким образом, общее число всех наборов входных данных, которые потребовалось перебрать при тестовых запусках нейронных сетей – 20 790 наборов данных. Выходным результатом работы нейронной сети было целое число, которое принимало значения от 1 до 5, тем самым показывая предполагаемую реакцию пользователя на погодные условия (very_sad, sad, neutral, smile, happy). Небольшая часть рассмотренных входных наборов данных и полученных выходных результатов (монолитная и блочная нейронные сети) представлена в Таблица 1.

Таблица 1 – Результаты работы монолитной нейронной сети и математической модели на одинаковых наборах входных данных

Входные данные	Результат исходной НС	Результат математической модели
+22·C(7), 745 mm(3), малооблачно(2), 1 м\с(1), 60%(5)	5	5
+16·C(6), 743 mm(3), малооблачно(2), 2 м\с(1), 91%(7)	4	4
-19·C(3), 733 mm(3), пасмурно(4), 4 м\с(2), 45%(4)	3	3
-7·C(4), 736 mm(3), пасмурно(4), 2 м\с(1), 75%(5)	4	4
-16·C(3), 742 mm(3), снег(9), 2 м\с(1), 81%(6)	4	4
+9·C(6), 754 mm(4), дождь(6), 6 м\с(2), 72%(5)	2	2
+3·C(6), 758 mm(3), гроза(7), 12 м\с(4), 97%(7)	1	1
+34·C(8), 737 mm(3), ясно(1), 0 м\с(1), 40%(4)	3	3
+44·C(8), 732 mm(3), ясно(1), 3 м\с(2), 32%(3)	1	1
-32·C(2), 756 mm(4), малооблачно(2), 1 м\с(1), 64%(5)	2	2
-40·C(1), 745 mm(3), снег(9), 7 м\с(3), 28%(2)	1	1

Как видно из результатов, представленных в таблице, для одинаковых входных данных исходная сеть и предлагаемая математическая модель получают одинаковые выходные сигналы. Такая картина происходит на всех возможных наборах данных, которые были поданы обеим сетям на вход, то есть выходные сигналы совпали на абсолютно всех возможных наборах входных данных, допустимых для исходной нейронной сети. На основании результатов работы монолитной и блочной нейронных сетей справедливо утверждение, что разработанные модели блочных нейронных сетей являются адекватными моделями исходных нейронных сетей.

2.5. Выводы по главе

Разработанная математическая модель подходит для использования в рамках создания усовершенствованного метода синтеза нейросетевых устройств и реализации алгоритмов декомпозиции нейронных сетей и функционирования

блочных нейронных сетей. Адекватность представленной математической модели подтверждается полным перебором всех возможных входных векторов и сравнением их с результатами, полученными исходной монолитной нейронной сетью, такая операция была проведена с несколькими нейронными сетями, результаты всех проверок подтвердили адекватность представленной модели. Это означает, что полученная модель отражает процесс декомпозиции нейронной сети в блочную нейронную сеть достаточно точно, чтобы использовать ее в разработке усовершенствованного метода синтеза устройств и алгоритма декомпозиции нейронных сетей с различными входными параметрами, которые позволят осуществить оптимальное распределение нагрузки по предлагаемым вычислительным узлам существующей системы с устоявшейся архитектурой.

ГЛАВА 3. РАЗРАБОТКА МЕТОДА СИНТЕЗА УСТРОЙСТВ РЕАЛИЗАЦИИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ, ОРИЕНТИРОВАННЫХ НА ТУМАННЫЕ ВЫЧИСЛЕНИЯ, И ЕГО ОТКАЗОУСТОЙЧИВОЙ МОДИФИКАЦИИ

3.1. Формулировка метода синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления

В первую очередь были сформулированы ограничения, в рамках которых функционирует и дает положительные результаты предлагаемый усовершенствованный метод. Обучение монолитной нейронной сети должно быть завершено к моменту использования метода, то есть метод дает возможность декомпозировать уже обученные нейронные сети. Производится декомпозиция именно многослойных нейронных сетей, поскольку полные сети потребуют других механизмов декомпозиции и последующего функционирования. Предлагаемый усовершенствованный метод декомпозирует нейронные сети как без обратных связей, так и с обратными связями, необходимые математические модели FFNN и рекуррентных нейронных сетей представлены в Главе 2. Метод применим для декомпозиции нейронных сетей независимо от того, для решения каких именно задач будут использованы синтезируемые нейросетевые устройства и блочная нейронная сеть. Декомпозиции с помощью представленного метода могут подвергаться нейронные сети для распознавания, управления, классификации, генерации и других задач, которые решаются нейросетевыми методами. Подробнее о предлагаемом методе можно прочитать в работе [6].

Именно метод преобразования монолитной нейронной сети к виду блочной нейронной сети, адаптированной для выполнения туманных вычислений [38] и является ключом к созданию каскадов из нескольких не очень мощных вычислительных устройств на ПЛИС и микроконтроллерах, которые будут сопоставимы с более производительными вычислительными устройствами и будут иметь достаточную производительность для работы нейронных сетей. Поэтому актуальной задачей становится разработка и реализация усовершенствованного метода синте-

за нейросетевых устройств для реализации распределенных нейронных сетей, который сможет сбалансировать нагрузку на уже имеющиеся устройства таким образом, чтобы нагрузка требуемым образом была распределена между узлами, имеющими запас производительности, при этом не вызвав на них существенного ухудшения технико-экономических и эксплуатационных характеристик выполнения уже существующей обработки данных.

Некоторые подходы, примененные в усовершенствованном методе декомпозиции нейронной сети, были взяты из работы [18], например, то, что результатом работы метода должно быть отображение графа нейронной сети на граф физических узлов, а также использование разрезов графа нейронной сети для получения блоков блочной нейронной сети. Однако предлагаемый усовершенствованный метод декомпозиции позволяет разделить на блоки многослойную нейронную сеть, а предлагаемый в работе [18] метод может декомпозировать только нейронные сети с архитектурой ВИНС (воспроизводящейся искусственной нейронной сети). Предлагаемый нами усовершенствованный метод позволяет убрать зависимость от основного сервера, который раздает всем остальным вычислительные задачи. Для этого заранее отчертим для создаваемого вычислительного узла ту часть вычислений, которые он будет выполнять. В предлагаемом методе будет осуществляться разрезы графа нейронной сети в зависимости от исходных параметров монолитной нейронной сети и задаваемых пользователем входных параметров декомпозиции.

Входными данными для реализации метода являются МНС, число устройств D, их вычислительные мощности $\overline{p}$ и флаг минимизации передаваемых данных. Метод формирует оптимальную по выбранным параметрам последовательность нейросетевых устройств для реализации режима fog computing размерности D. Сущность предлагаемого усовершенствованного метода синтеза нейросетевых устройств представлена на Рисунке 16. На первом шаге определяется нужная МНС и входные параметры для работы метода: число устройств в каскаде, их вычислительные мощности и конфигурация и пропускная способность каналов связи. Далее с помощью алгоритма декомпозиции монолитной нейронной сети осуществить все декомпозиции различными способами для всех потенциальных архитектур вычис-

лительных каскадов, предложенный алгоритм декомпозиции основан на оригинальной модели блочной нейронной сети. На следующем этапе используется алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах, который позволяет осуществить Парето-оптимизацию [50, 89] всех полученных вариантов декомпозиции по параметрам стоимости, энергопотребления, времени выполнения нейросетевых вычислений и времени отклика критически важных узлов вычислительного каскада. Последний этап подразумевает под собой использование полученного оптимального варианта синтеза каскада нейросетевых устройств с помощью предложенного алгоритма функционирования блочной нейронной сети.

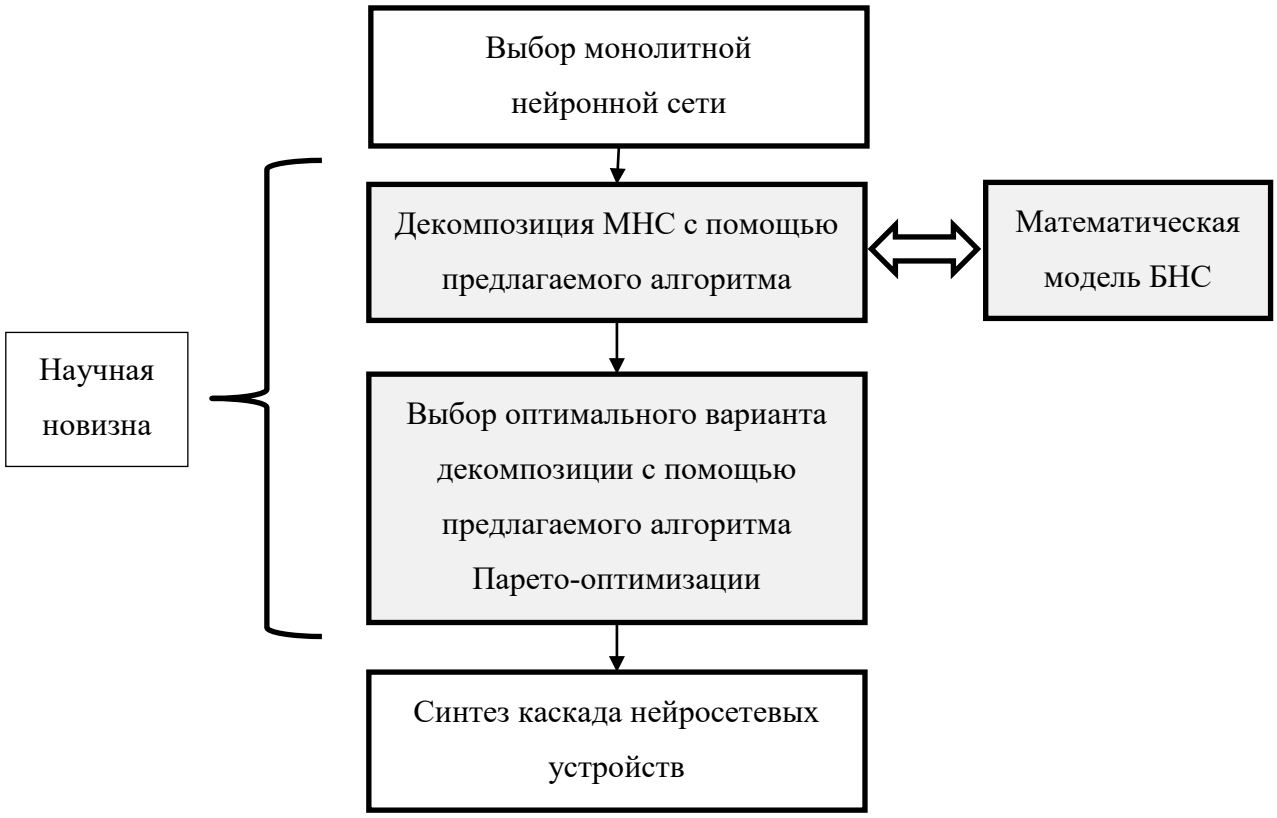


Рисунок 16 – Сущность предлагаемого метода синтеза нейросетевых устройств для реализации распределенных нейронных сетей

Более детально шаги, которые потребуются для реализации метода синтеза нейросетевых устройств, представлены ниже, они детализируют сущность метода до последовательности понятных действий, которые необходимы для реализации каскада нейросетевых устройств:

1. Выбрать монолитную нейронную сеть, которую требуется декомпозировать

2. Перевести ее в предлагаемый формат обработки нейронных сетей *.ann;

3. Использовать алгоритм декомпозиции МНС, задав входные параметры: количество устройств, на которые необходимо разделить нейронную сеть D, массив вычислительных мощностей устройств $\{P_0, \dots, P_{D-1}\}$ и флаг предпочитаемого способа разделения на блоки;

4. Получить D блоков БНС в файлах предлагаемого формата;

5. Перенести содержимое файла в программу для микроконтроллера;

6. Скомпилировать файл *.hex и загружаем его как прошивку для устройства;

7. Запустить в работу каскад контроллеров, соединенных между собой.

Для реализации метода на процессорных устройствах с операционной системой вторая половина шагов видоизменяется:

5. Перенести файл на устройство в соответствующую директорию файловой системы;

6. Запустить программу выполнения блока БНС, написанную на языке программирования высокого уровня;

7. Запустить в работу каскад устройств, соединенных между собой.

Возможен комбинированный вариант реализации каскада вычислительных устройств, содержащего как микроконтроллеры, так и процессорные устройства, в случае, когда между устройствами различного типа существует канал связи.

Данные между вычислительными устройствами, соединёнными в каскад должны передаваться в следующем формате: байт начала передачи, номер пакета передаваемых данных (2 байта), полезная нагрузка (количество байт определяется решаемой задачей), байт окончания передачи данных. Рассмотрим подробнее каждый из элементов, входящих в пакет передаваемых данных подробнее. За байт начала передачи был принят 37h, все пришедшие далее по каналу аналогичные биты считаются полезной нагрузкой либо до получения байта окончания передачи (42h), либо до превышения максимальной размерности полученного пакета (которая определяется как число байт полезной нагрузки + 4). Номер пакета сравнивается с номером предыдущего полученного пакета, если такое требуется условиями зада-

чи, счетчик номера пакета вполне может обнуляться в соответствии с переполнением двухбайтного целого числа. Полезная нагрузка – либо передает сразу все данные, которые требуется передать от одного блока нейронной сети к другому, либо дробит данные на порции, в этом случае как раз и требуется контроль номера передаваемого пакета. Байт окончания передачи данных – 42h, если он встретился и размер данных полезной нагрузки полностью передан – пакет считается оконченным. В различных задачах может быть установлен различный таймаут ожидания следующего пакета, в рассмотренной в нашей работе задачи устанавливалось бессрочное ожидание передачи данных (неограниченный таймаут).

Свойство коммутативности для рассматриваемого метода не соблюдается, поскольку замена первого вычислителя на второй поменяют последовательность выполнения нейросетевых операций, которые были заложены в каскад нейросетевых устройств в целом, что приведет к получению неверного результата нейросетевых вычислений всем каскадом нейросетевых устройств.

Структура пакета данных, которые передаются между устройствами, реализующими блоки нейронной сети представлена в Таблице 2.

Таблица 2 – Структура пакета данных, передаваемых между вычислительными устройствами в каскаде

Стартовый байт	Номер устройства	Номер пакета	Передаваемая информация	Байт окончания
1 байт	1 байт	2 байта	n байт	1 байт

На Рисунке 17 представлена обобщенная схема передачи данных между блоками и пример содержимого пакетов, которые передаются от одного вычислительного узла к другому.

Однако, для реализации предложенного метода необходимо учитывать различные входные параметры, которые будут напрямую влиять на результаты декомпозиции.

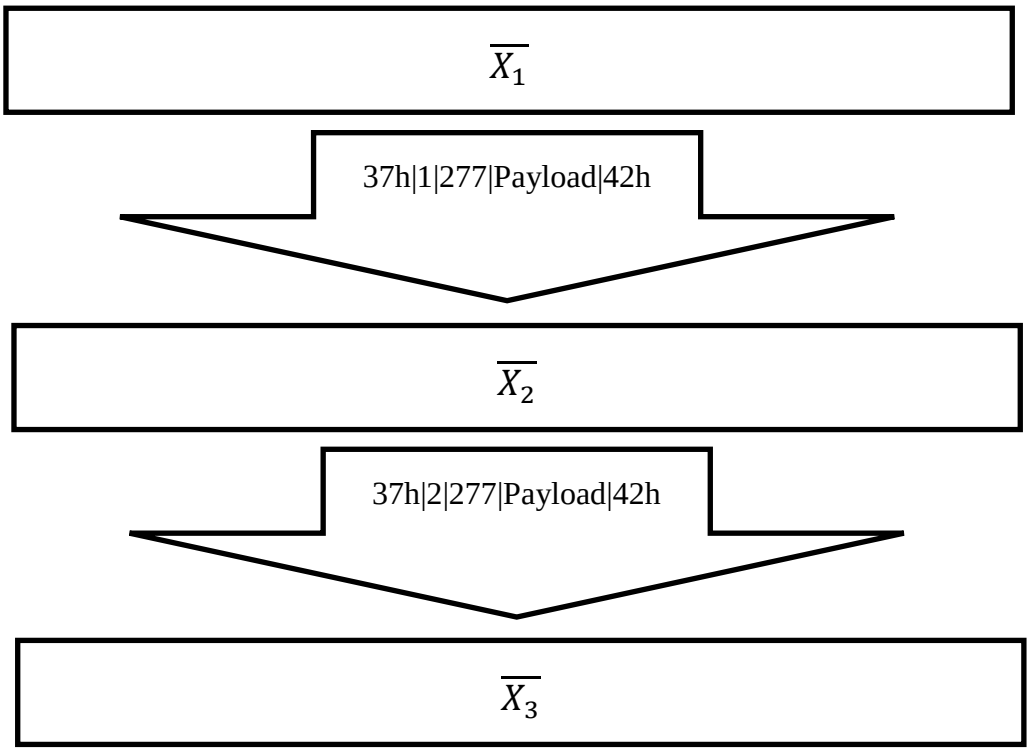


Рисунок 17 – Предлагаемая структура пакета передаваемых данных

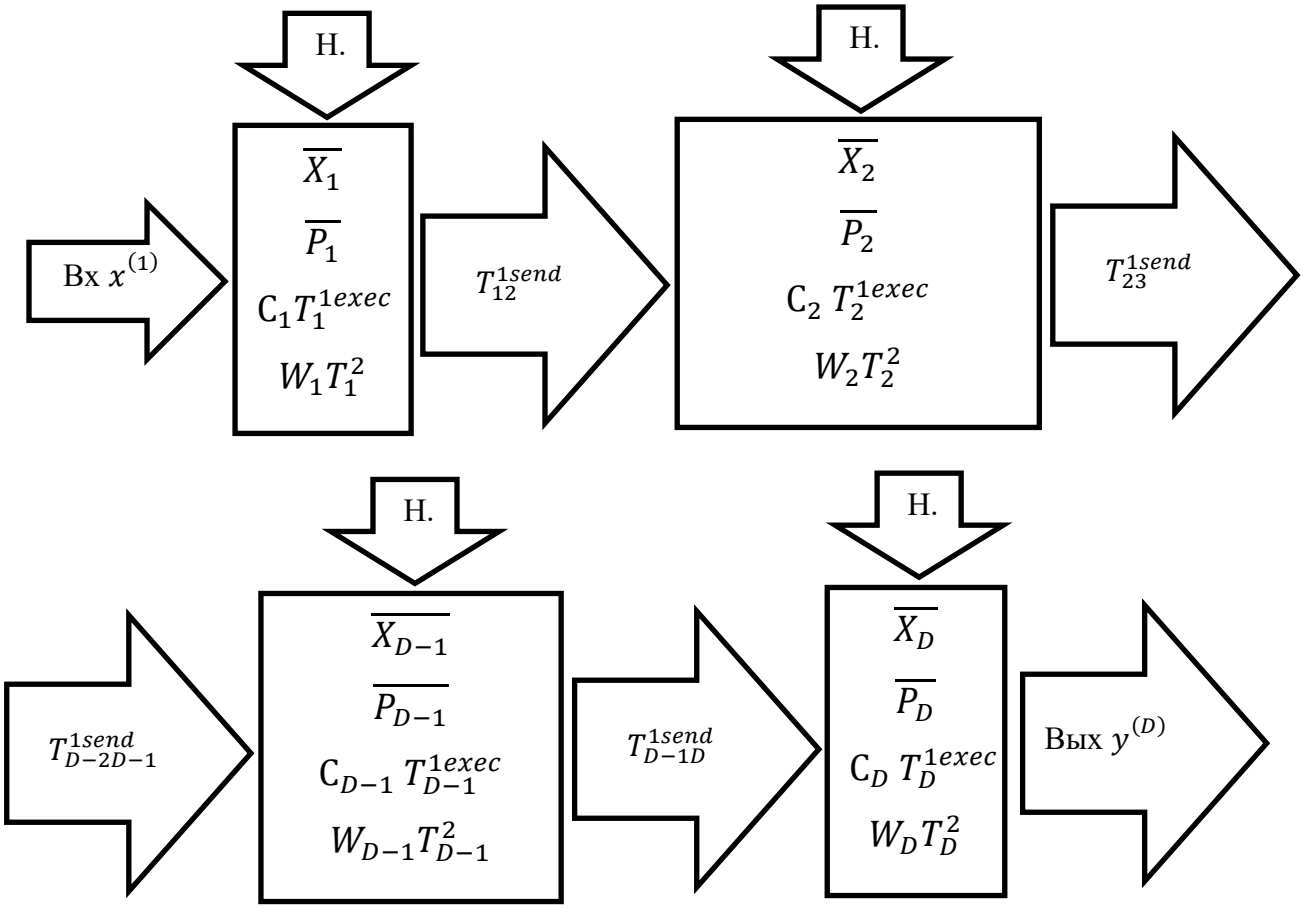


Рисунок 18 – Модель блочной нейронной сети с указанием параметров, по которым будет производиться оптимизация по Парето

Таким образом, после осуществления различных вариантов декомпозиции, полученные результаты будут сравниваться по 4 параметрам: стоимости C, энергопотреблению W, времени выполнения нейросетевых вычислений T^1 и времени выполнения основных вычислений T^2 (Рисунок 18).

Предложенный метод синтеза нейросетевых устройств может быть использован для реализации двух различных подходов к организации вычислительной системы: реализация нейросетевых вычислений как побочной нагрузки на остаточные вычислительные ресурсы мощных устройств в вычислительной сети, либо реализация сложной нейронной сети с большим количеством слоев и нейронов на нескольких устройствах с невысокой вычислительной мощностью.

3.2. Разработка вариантов декомпозиции нейронной сети, в зависимости от входных параметров

В процессе исследования был выполнен анализ различных наборов входных параметров, в зависимости от которых будут реализовываться конкретные варианты разделения монолитной ИНС на части. Предлагается следующая классификация вариантов декомпозиции монолитной нейронной сети в зависимости от входных параметров:

1. Разделение на блоки с равным количеством слоев на узле

Выше рассматривался случай, когда количество слоев нейронной сети *K* кратно числу устройств *D*, которые будут задействованы в цепочке вычисления результатов блочной нейронной сети. Для случаев, когда *K* не кратно *D*, количество слоев L_i, которые должны быть переданы на устройство D_i, вычисляется по формуле:

$$L_i = \left| \frac{K}{D} \right| \tag{14}$$

Поскольку применяется округление в меньшую сторону для получения целого числа, количество слоев нейронной сети, которое будет на последнем вычислителе с номером *D-1*, вычисляется по формуле:

$$L_{D-1} = K - (D-1)L_0 \tag{15}$$

Таким образом, получим общую рекуррентную формулу описания каждой из полученных блочных нейронных сетей:

$$\bar{X}_N : \begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right), N = D-1 \\ y_i^{\left((N+1)\left\lfloor \frac{K}{D} \right\rfloor\right)} = f_i^{\left((N+1)\left\lfloor \frac{K}{D} \right\rfloor\right)}\left(\sum_{j=0}^{H_{(N+1)\left\lfloor \frac{K}{D} \right\rfloor - 1}} w_{ij}^{\left((N+1)\left\lfloor \frac{K}{D} \right\rfloor\right)} \cdot y_i^{\left((N+1)\left\lfloor \frac{K}{D} \right\rfloor - 1\right)}\right), N < D-1 \end{cases} \\ \dots \\ \begin{cases} y_i^{\left(N\left\lfloor \frac{K}{D} \right\rfloor + 1\right)} = y_i^{\left(N\left\lfloor \frac{K}{D} \right\rfloor\right)}, N > 0 \\ y_i^{(0)} = x_i^{(0)}, N = 0 \end{cases} \end{cases} \tag{16}$$

Тогда общая рекуррентная формула описания каждой из полученных блочных нейронных сетей, где массив $\{L_0, ..., L_{D-1}\}$ – количество слоев нейронной сети, которые должны быть переданы на устройство с номером N выглядит следующим образом:

$$\bar{X}_N : \begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right), N = D-1 \\ y_i^{\left(\sum_{k=0}^{N} L_k\right)} = f_i^{\left(\sum_{k=0}^{N} L_k\right)}\left(\sum_{j=0}^{H_{\sum_{k=0}^{N} L_k - 1}} w_{ij}^{\left(\sum_{k=0}^{N} L_k\right)} \cdot y_i^{\left(\sum_{k=0}^{N} L_k - 1\right)}\right), N < D-1 \end{cases} \\ \dots \\ \begin{cases} y_i^{(L_{N-1}+1)} = y_i^{(L_{N-1})}, N > 0 \\ y_i^{(0)} = x_i^{(0)}, N = 0 \end{cases} \end{cases} \tag{17}$$

2. Разделение пропорционально производительности устройств (слои)

Для разделения пропорционально производительности (по количеству слоев нейронной сети) нужно сделать массив мощностей $\{P_0, ..., P_{D-1}\}$, выраженных в некоторых абсолютных единицах (результатах тестов производительности, тактовых частотах, объемах оперативной памяти).

Процент от всех слоев сети, который должен быть передан на устройство D_i для выполнения вычислений, рассчитывается по формуле:

$$\gamma_i = \frac{P_i}{\sum_{h=0}^{D-1} P_h} \tag{18}$$

Следовательно, количество слоев L_i, которые должны быть переданы на устройство D_i, вычисляется по формуле:

$$L_i = \left\lfloor \frac{K \cdot P_i}{\sum_{h=0}^{D-1} P_h} \right\rfloor \tag{19}$$

Поскольку применяется округление в меньшую сторону для получения целого числа, количество слоев нейронной сети, которое будет на последнем вычислителе с номером *D-1*, вычисляется по формуле:

$$L_{D-1} = K - \sum_{h=0}^{D-2} L_h \tag{20}$$

Обобщив формулы (12), (13) и (10), получим общую рекуррентную формулу описания каждой из полученных блочных нейронных сетей:

$$\overline{X}_N : \begin{cases} \begin{cases} y_i^{(K)} = f_i^{(K)}\left(\sum_{j=0}^{H_{K-1}} w_{ij}^{(K)} \cdot y_i^{(K-1)}\right), N = D-1 \\ y_i^{\left(\sum_{m=0}^{N}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor\right)} = f_i^{\left(\sum_{m=0}^{N}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor\right)}\left(\sum_{j=0}^{H_{\sum_{m=0}^{N}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor - 1}} w_{ij}^{\left(\sum_{m=0}^{N}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor\right)} \cdot y_i^{\left(\sum_{m=0}^{N}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor - 1\right)}\right), N < D-1 \end{cases} \\ \cdots \\ \begin{cases} y_i^{\left(\sum_{m=0}^{N-1}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor + 1\right)} = y_i^{\left(\sum_{m=0}^{N-1}\left\lfloor \frac{K \cdot P_m}{\sum_{h=0}^{D-1} P_h} \right\rfloor\right)}, N > 0 \\ y_i^{(0)} = x_i^{(0)}, N = 0 \end{cases} \end{cases} . \tag{21}$$

3. Разделение на блоки с равным количеством нейронов на узле

Берем сумму всех нейронов на всех слоях монолитной нейронной сети. На каждом слое всего H_b нейронов, тогда желаемое число нейронов в каждой из блочных ИНС будет равно

$$Neur_N = \frac{\sum_{b=1}^{K} H_b}{D} \tag{22}$$

В целочисленном варианте:

$$Neur_N = \left\lfloor \frac{\sum_{b=1}^{K} H_b}{D} \right\rfloor \quad (23)$$

И в последней нейронной сети:

$$Neur_{D-1} = \sum_{b=1}^{K} H_b - (D-1)\left\lfloor \frac{\sum_{b=1}^{K} H_b}{D} \right\rfloor \quad (24)$$

Но данный вариант тоже не подходит: нейроны разделены на слои, а делить слой, разнося его на различные устройства, нецелесообразно, так как это увеличит объем данных, пересылаемых между физическими устройствами. По этой причине разделять сеть будем послойно, а определять число нейронов и слоев на каждой из блочных ИНС – с помощью сравнения с желаемым значением $Neur_N$. Чтобы получить массив $\{L_0,...,L_{D-1}\}$, проведем предварительный шаг оценки, используя жадный алгоритм:

1. На очередной ИНС нет слоев нейронов.

2. Если количество нейронов на слоях суммарно со слоем-претендентом меньше, чем $\left\lfloor \frac{\sum_{b=1}^{K} H_b}{D} \right\rfloor$, то добавить слой к ИНС, то есть увеличить L_N на единицу.

3. Иначе – зафиксировать ИНС как завершенную, перейти к составлению следующей ИНС.

4. На ИНС с номером *D-1* перенести все оставшиеся слои.

Тогда после получения массива $\{L_0,...,L_{D-1}\}$ достаточно подставить их в (10).

4. Разделение пропорционально производительности устройств (нейроны)

В способе с распределением нейронов по блочным ИНС пропорционально мощности устройств желаемое число нейронов для ИНС будет равно $Neur_N$:

$$Neur_N = \frac{\sum_{b=1}^{K} H_b \cdot P_N}{\sum_{h=0}^{D-1} P_h}, N \neq D-1 \quad (25)$$

В целочисленном варианте:

$$Neur_N = \left\lfloor \frac{\sum_{b=1}^{K} H_b \cdot P_N}{\sum_{h=0}^{D-1} P_h} \right\rfloor, N \neq D-1 \quad (26)$$

Тогда последняя ИНС:

$$Neur_N = \sum_{b=1}^{K} H_b \cdot (D-1) \left\lfloor \frac{\sum_{b=1}^{K} H_b \cdot P_N}{\sum_{h=0}^{D-1} P_h} \right\rfloor, N = D-1 \quad (27)$$

По той же причине, что и в предыдущем способе, делить нейронную сеть будем послойно. Определять число нейронов и слоев в сети будем с помощью сравнения с L_N по следующему алгоритму:

1. На очередной ИНС нет слоев нейронов.
2. Если количество нейронов на слоях суммарно со слоем-претендентом меньше, чем $\left\lfloor \frac{\sum_{b=1}^{K} H_b \cdot P_N}{\sum_{h=0}^{D-1} P_h} \right\rfloor$ то добавить слой к ИНС, то есть увеличить L_N на единицу.
3. Иначе – зафиксировать ИНС как завершенную, перейти к составлению следующей ИНС.
4. На ИНС с номером *D-1* перенести все оставшиеся слои

Тогда после получения массива $\{L_0,...,L_{D-1}\}$ достаточно подставить их в (10).

5. Разделение с условием минимизации передаваемых по сети данных

Для того, чтобы минимизировать объем данных, передаваемых по сети между устройствами, которые будут выполнять вычисления блочных ИНС, необходимо разделять монолитную нейронную сеть на части, рассекая ее в тех местах, где число нейронов на соответствующем слое минимально. В этом случае будет минимальное число параметров, которые потребуется передавать по сети между вычислительными устройствами.

Для того, чтобы получить массив послойных размерностей блочных ИНС $\{L_0,...,L_{D-1}\}$ потребуется осуществить предварительную подготовку. В первую очередь нам потребуется отобрать *D-1* слоев нейронной сети с наименьшим коли-

чеством нейронов. Отсортируем для этого массив слоев нейронной сети по количеству нейронов на каждом слое, сортировка по возрастанию. Первые *D-1* слоев и есть искомые нами слои, отделим их в новый массив и отсортируем уже по порядковому номеру слоя в монолитной нейронной сети, по возрастанию. Увеличим размерность массива *Numb* до *D+1*, на нулевое место поставим *0*, на последнее – номер последнего слоя *K*. Полученный массив $\{Numb_0,...,Numb_D\}$ будет использован нами для вычисления массива послойных размерностей блочных ИНС $\{L_0,...,L_{D-1}\}$ по следующей формуле:

$$L_N = Numb_{N+1} - Numb_N \tag{28}$$

Далее от нас требуется подставить полученные значения массива $\{L_0,...,L_{D-1}\}$ в (10).

Предложенная классификация обеспечивает детализацию предлагаемого метода в зависимости от входных параметров для нейросетевых устройств не совсем критического применения. В настоящее время чрезвычайно актуально использование нейросетевых устройств в вооружении и военной технике, в которых очень важно обеспечение надежности [11, 20]. Поэтому необходима отказоустойчивая модификация метода синтеза нейросетевых для реализации режима туманных вычислений.

3.3. Разработка модификации метода синтеза отказоустойчивых нейросетевых устройств для реализации режима туманных вычислений

Предлагаемый метод синтеза может позволить получить каскад вычислительных устройств, который будет обладать возможностью продолжать работу даже после выхода одного или более вычислительных узлов. Если на одном из устройств, включенных в каскад, происходит отказ, то необходимо, чтобы выполнение его функций взял на себя другой вычислительный узел системы[6]. В рамках предложенного исследования рассматриваются два подхода к повышению отказоустойчивости: структурное резервирование [15] и адаптация.

Прикладной смысл структурного резервирования заключается в добавлении дополнительного устройства к каскаду параллельно с основным. Таким образом, дополнительное устройство дублирует вычисления, выполняемые основным. Положительный эффект этого метода заключается в том, что неисправное устройство заменяется другим и не происходит перераспределения вычислительной нагрузки между другими узлами каскада. Отрицательным эффектом этого метода является значительное увеличение стоимости системы, поскольку требуется ввести в систему дополнительные вычислительные узлы [32].

Второй рассмотренный подход – это адаптация [10]. Суть этого метода заключается в том, что между узлами создаются дополнительные соединения. Это означает, что соединение устанавливается не только с соседними узлами каскада, но и со следующими и предыдущими узлами. Программное обеспечение каждого вычислительного устройства включает в себя не только программу его собственной блочной нейронной сети, но и блочные нейронные сети соседних с ним устройств. Серийный номер узла отправляется вместе с данными, передаваемыми подключенным узлам. Если узел перестает получать информацию от соседнего в течение определенного количества циклов, то этот узел принимает входные данные от предпоследнего узла. Затем узел подключает другую блочную нейронную сеть в программе и начинает вычислять свою собственную блочную нейронную сеть и блочную нейронную сеть своего соседа по каскаду. Очевидной положительной стороной этого метода является то, что он не требует покупки и установки дополнительных узлов для дублирования каскадных элементов. Отрицательной стороной является то, что это увеличивает вычислительную нагрузку на определенные узлы. Чтобы сбалансировать это, могут быть введены обратные связи. Это позволило бы системе выбирать, должен ли предыдущий или следующий вычислительный узел взять на себя функции неисправного.

После анализа обоих вариантов реализации, предпочтение было отдано варианту с адаптацией, поскольку он позволяет добавить отказоустойчивость практически без изменения стоимости системы. Чтобы нивелировать недостатки этого подхода, в будущем потребуется система балансировки нагрузки [82].

Рассмотрим подход, который лег в основу отказоустойчивой модификации метода синтеза нейросетевых устройств подробнее: предлагаемый подход относится к средствам активной отказоустойчивости [30], поскольку в нем требуется фиксация факта отказа или сбоя, определение места отказа, все это можно обобщенно именовать – диагностика [14]. Далее происходит реконфигурация, то есть перераспределение функций в рамках всего вычислительного каскада, в процессе адаптации замещающий узел системы начнет выполнять дополнительно функции отказавшего узла. Это дешевле структурного резервирования, но производительность всей системы снижается, относительно рассчитанных изначально показателей. При этом необходимо следить за нахождением параметров времени выполнения и времени отклика устройств в допустимых диапазонах, в случае выхода за их границы может потребоваться вмешательство человека и повторный поиск оптимальной декомпозиции для новой конфигурации вычислительного каскада. В каждом устройстве, которое сможет выступить замещающим, должны быть копии блока сети узла, который оно страхует от отказа или сбоя. Диагностика отказа или сбоя производится с помощью контроля передаваемых данных, если устройство, которое диагностируется, не подает сигналы в течение определенного таймаута (в наших экспериментах – 250 мс), то происходит реконфигурация устройства и оно начинает выполнять работу по реализации своего блока нейронной сети и реализации блока замещаемого устройства [22].

На Рисунке 19 видно, что добавлено дополнительное соединение. Дополнительное соединение представляет собой бордовую линию от первого устройства к третьему в каскаде. Кроме того, добавлена обратная связь от четвертого устройства к третьему (бордовая линия). Отсутствие информации, передаваемой по линии обратной связи, определяет момент, в который третье устройство должно взять на себя дополнительные функции четвертого устройства.

Данные между вычислительными устройствами передаются в том же формате, что и раньше, но требуется добавить еще одно поле – номер вычислительного устройства, с которого отправляется пакет данных, в предыдущей схеме это выглядит как: байт начала передачи, номер устройства в вычислительном каскаде

(1 байт), номер пакета передаваемых данных (2 байта), полезная нагрузка, байт окончания передачи данных. Номер вычислительного устройства устанавливается на каждом устройстве в код вычислительного блока на этапе компиляции, также эта информация представлена в файле описания нейронной сети *.ann. В различных задачах может быть установлен различный таймаут ожидания следующего пакета с определенным номером вычислителя, в рассмотренной в нашей работе задачи время между приходом пакетов могло быть разным, поскольку рассматриваемый функционал не был активен всегда, поэтому было выставлено следующее ограничение: если на адаптирующееся устройство не приходит три пакета подряд от отказавшего устройства, то адаптирующееся устройство берет на себя его функции.

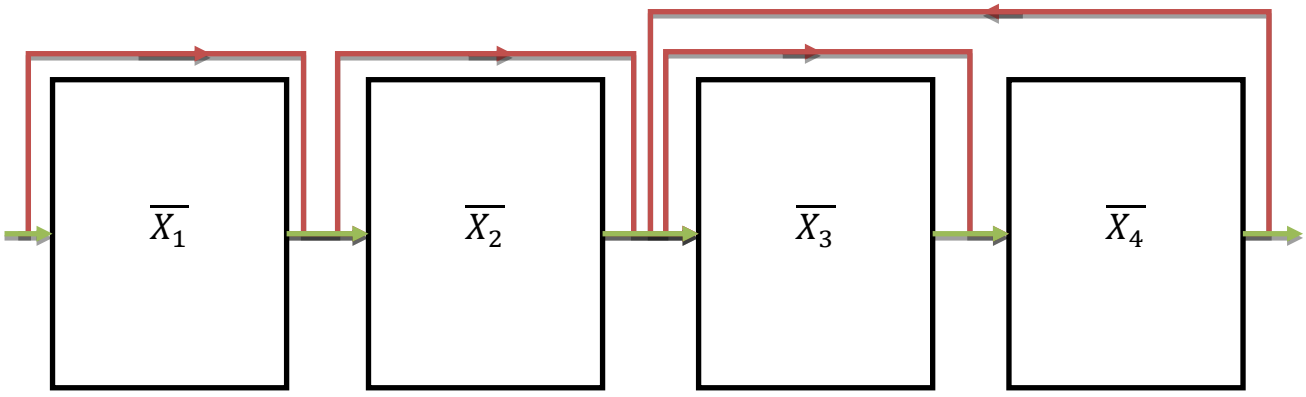


Рисунок 19 – Схема блочной нейронной сети на каскаде
из четырех микроконтроллеров с диагностикой и адаптацией.

В предлагаемом отказоустойчивом варианте метода устройство с номером N+1 является замещающим для устройства с номером N при условии, что замещению подвергается одно отказавшее устройство. Если требуется адаптировать каскад к потенциальному отказу двух устройств, идущих друг за другом, то устройство с номером N+2 станет замещающим для устройств N и N+1. Последнее устройство в каскаде, с номером D, не имеет устройства с номером больше своего, поэтому его диагностику и замещение берет на себя устройство с номером D-1, для этого у последнего устройства организуется обратная связь.

Таким образом, для того чтобы осуществить синтез каскада нейросетевых устройств с помощью модификации метода (Рисунки 20, 21), которая позволит

обеспечить определенный уровень отказоустойчивости, необходимо проделать следующие действия:

1. Осуществить декомпозицию монолитной нейронной сети с помощью оригинального метода, описанного в разделе 3.1. В результате будет получено D блоков БНС в файлах предлагаемого формата;

2. Выбрать глубину адаптации G, которая может принимать значения от 1 до D-1, этот параметр определит то, сколько соседних устройств сможет заместить текущее в случае их отказа или сбоя;

3. Перенести содержимое нужных файлов с блоками БНС в количестве G+1 в модифицированную программу для микроконтроллера;

4. Скомпилировать файл *.hex и загрузить его как прошивку для устройства;

5. Запустить в работу каскад контроллеров, соединенных между собой.

Для реализации метода на процессорных устройствах с операционной системой вторая половина шагов видоизменяется:

8. Перенести G+1 файл на устройство в соответствующую директорию файловой системы;

9. Запустить модифицированную программу выполнения блока БНС, написанную на языке программирования высокого уровня;

10. Запустить в работу каскад устройств, соединенных между собой.

На Рисунках 13 и 14 показаны схемы функционирования каскада вычислительных устройств, реализованные с помощью адаптируемой конфигурации метода синтеза, в штатном режиме и в случае отказа одного из устройств в каскаде. На Рисунке 14 устройство с номером $\overline{X_2}$ неисправно и устройство под номером $\overline{X_{D-1}}$ в процессе диагностики и реконфигурации начинает выполнять свои функции и дополнительно блок НС предыдущего устройства, актуальные направления главного потока передачи данных отмечены более жирными зелеными стрелками. Предложенная модификация позволяет заместить G устройств из каскада с общим числом устройств D. Глубина адаптации G для рассматриваемого примера равна 1, то есть каскад может адаптироваться к потере одного устройства подряд.

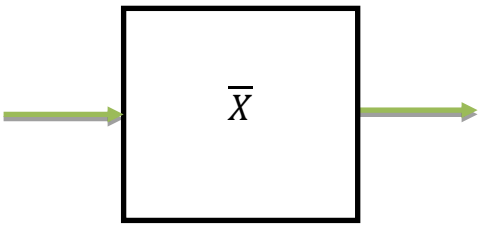


Рисунок 20 – Схема функционирования МНС на одном устройстве в штатном режиме

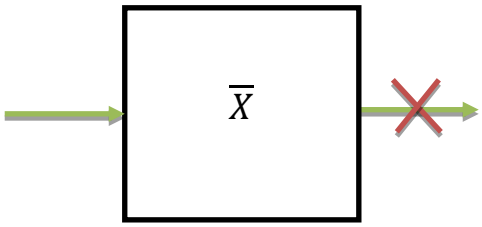


Рисунок 21 – Схема функционирования МНС на одном устройстве в случае отказа

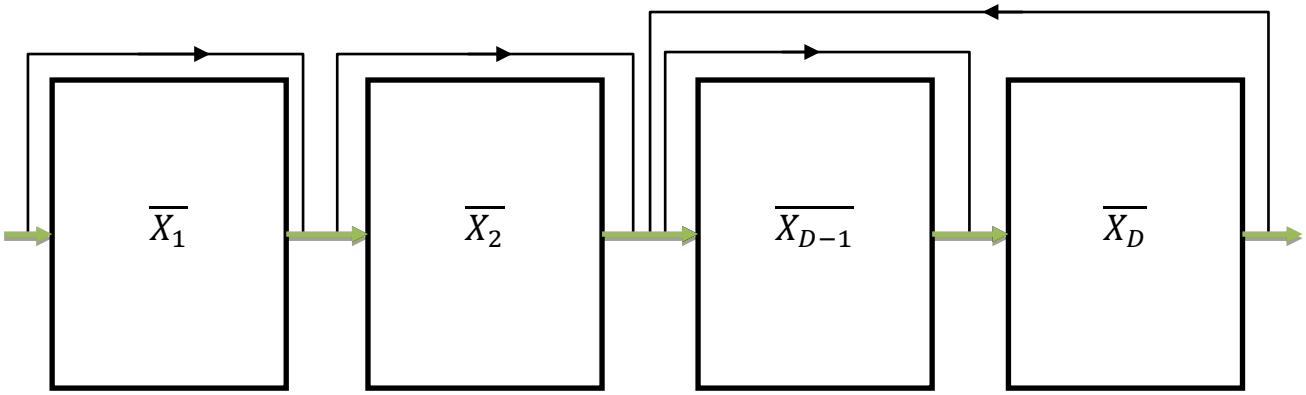


Рисунок 22 – Схема функционирования каскада
вычислительных устройств в штатном режиме

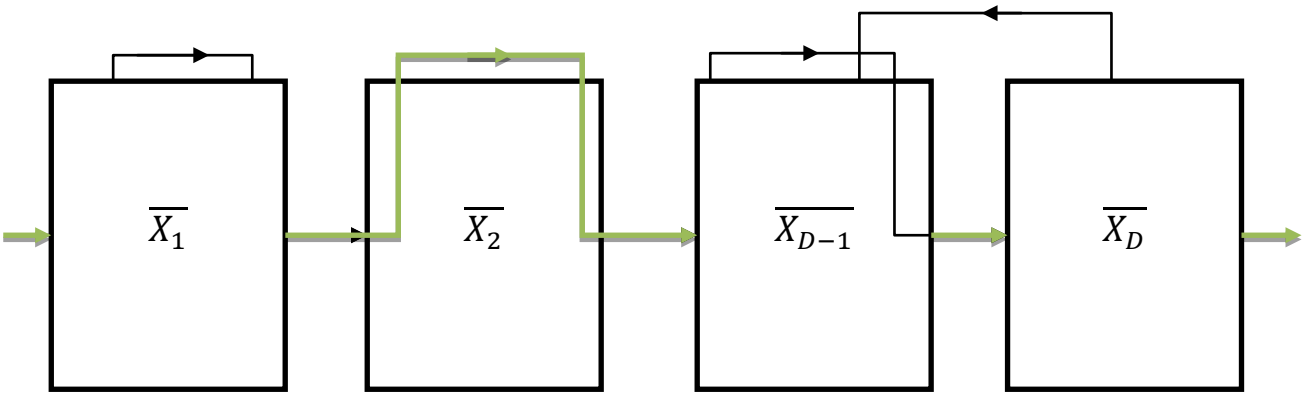


Рисунок 23 – Схема функционирования каскада вычислительных устройств
в случае отказа одного из устройств

Нейросетевое устройство, реализующее монолитную нейронную сеть (Рисунок 20) не может отказать без потери работоспособности всей нейронной сети, ведь все функции нейросетевых вычислений исполняются только на нем, в случае его отказа (Рисунок 21) работа нейронной сети прекращается. Однако, в случае каскада нейросетевых устройств, разработанного с применением отказоустойчивой реализации предлагаемого метода (Рисунок 22), в случае отказа одного из устройств, произойдет диагностика и реконфигурация вычислительной системы, что позволит сохранить работоспособность нейронной сети на всем каскаде в целом (Рисунок 23).

3.4. Выводы по главе

Предложенный усовершенствованный метод синтеза устройств нейросетевого распознавания позволяет осуществить декомпозицию монолитной нейронной сети на блоки блочной нейронной сети, которые в свою очередь распределяются по соответствующим нейросетевым устройствам. Для этого необходимо декомпозировать нейронную сеть с помощью алгоритма разделения, распределить блочные сети по вычислительным узлам и запустить распределенный алгоритм выполнения блочной нейронной сети. С помощью разработанного метода возможна декомпозиция многослойных обученных нейронных сетей, с различными архитектурами, но для каждого крупного семейства архитектур потребуется произвести модификацию математической модели в соответствии с особенностями рассматриваемой архитектуры. Метод подходит для декомпозиции нейронных сетей с любым функциональным назначением, то есть не привязан к смысловому наполнению операций, производимых нейронной сетью, а проводит манипуляции именно со структурной схемой декомпозируемой нейронной сети. Был разработан метод, в котором можно обеспечить отказоустойчивость, то есть, который позволяет осуществить диагностику и адаптацию каскада устройств, для замещения функций одного или нескольких вычислительных узлов каскада с помощью передачи вычислительных функций от отказавшего устройства к замещающему, для этого потребуется увеличение числа связей между физическими устройствами.

Благодаря разработанному усовершенствованному методу синтеза нейросетевых устройств, который позволяет настраивать процесс декомпозиции исходя из входных параметров, например, пропорционально или равномерно разделять вычислительную нагрузку между узлами, становится возможным реализовать в вычислительной системе дополнительный ввод-вывод информации и обработку данных, не меняя физической архитектуры ее реализации, то есть, не добавляя новых устройств в систему.

ГЛАВА 4. РАЗРАБОТКА АЛГОРИТМОВ И ПРОГРАММ ДЕКОМПОЗИЦИИ МОНОЛИТНОЙ НЕЙРОННОЙ СЕТИ И ФУНКЦИОНИРОВАНИЯ БЛОЧНОЙ НЕЙРОННОЙ СЕТИ

4.1. Определение формата файла, описывающего блочную нейронную сеть

Каждая библиотека/фреймворк для работы с нейронными сетями имеет свою собственную сериализацию, например, Caffe использует буферы протокола, Torch имеет встроенную схему сериализации, а объекты Theano могут быть сериализованы с помощью pickle [53]. В некоторых системах, таких как OverFeat или darknet, веса и смещения сохраняются на диске в двоичном формате в виде соответствующих float (или double) смежных массивов. Обратите внимание, что это не распространяется на архитектуру сети/модели, которая должна быть известна или представлена отдельно (например, явно объявлена во время загрузки). Библиотека, подобная libccv, хранит структуру и веса в базе данных SQLite.

Как следствие, нейронные сети хранятся совершенно различными способами и в различных форматах, поэтому, в рамках исследования был предложен новый формат хранения нейронных сетей. Для работы с нейронными сетями в унифицированном виде на устройствах с программируемой логикой потребовалось создать несложный формат хранения нейронной сети в файле, чтобы иметь возможность читать и записывать нейронные сети при разделении на блоки из постоянной памяти вычислительного устройства. Формат файлов получил расширение .ann – сокращенное от artificial neural network. Нейронная сеть в этом файле хранится в следующем формате:

- в первой строке единственное целое число – номер блока нейронной сети, представленного в файле, для нейронной сети не подвергавшейся декомпозиции (монолитной) в этой строке будет число 0;
- во второй строке единственное целое число – количество слоев в сети K;
- далее в каждой строке описывается отдельный слой нейронной сети. В описании слоя части разделяются литерой «,». Сначала идут два целых числа –

порядковый номер слоя и число нейронов на слое, затем через разделитель располагаются описания каждого из нейронов;

• в описании нейрона части разделяются литерой «;». Описание состоит из пяти частей. Два целых числа в начале обозначают номер нейрона на слое и номер функции активации данного нейрона в справочнике функций. Далее идут три массива: массив констант для вычислений, массив весов связей и массив номеров нейронов предыдущего слоя с которыми связан данный нейрон. Для разделения элементов массива используется символ пробела.

Описание небольшой нейронной сети в предлагаемом в статье формате представлено в Примере 1.

```
0
3
0,3,0;12;1;1;0,1;12;1;1;1,2;12;1;1;2
1,3,0;10;1;1 1;0 1,1;10;1;1 1;1 2,2;10;1;1 1;0 2
2,1,0;11;0;1 1 1;0 1 2
```

Пример 1. Нейронная сеть, использованная при классификации терминов в ранних версиях программного продукта TSBuilder

Это исходное состояние нейронной сети, которая была использована в наших предыдущих исследованиях [68]. Преимуществом такого способа хранения является его компактность, недостаток выходит из преимущества – данное представление, даже в случае небольшой нейронной сети, сложно читается человеком.

Перейдем непосредственно к рассмотрению алгоритма разделения монолитной нейронной сети на каскад блочных нейронных сетей для туманных вычислений на устройствах с программируемой логикой.

Массив констант хранится в справочнике, в нашем случае – в файле NeuronFunctions.java. Сейчас он представлен 13 функциями. Этот список функций может быть расширен. В данном файле инкапсулируется выбор функции для вычисления в конкретном нейроне.

```
public enum NeuronFunctions {
  None(999),
  Sum (0),
  Max (1),
  Sigmoida (2),
  Linear (3),
  Threshold (4),
  Or(5),
  And(6),
  Tanh (7),
  ReLu (8),
  MaxCounter(9),
  Ntwo(10),
  Nthree(11),
  Equals(12); }
```

Здесь представлены следующие функции: сумма входных параметров, максимум среди входных параметров, сигмоида, линейная функция активации нейрона, пороговая функция, булева функция «ИЛИ», булева функция «И», гиперболический тангенс, функция подсчета максимальной частоты входных параметров, эквивалентность и авторские функции активации, которые использовались нами на начальных этапах исследования в нейронной сети продукта TSBuilder [41, 43, 44, 45, 68, 69].

Недавние исследования демонстрируют [54, 80], что все больше систем и фреймворков используют универсальный формат хранения данных ONNX (Open Neural Network Exchange) и соответствующая ему библиотека работы с нейронными сетями, она позволяет конвертировать в свой формат нейронные сети других библиотек, таких как Caffe [101], Keras [71], PyTorch[52], TensorFlow [39, 96] и другие. В связи с этим, в продолжении исследований по представленной тематике целесообразно наладить работу предлагаемого метода с данным форматом хранения нейронных сетей.

4.2. Разработка алгоритма декомпозиции монолитной нейронной сети на блоки блочной НС

Предлагаемый алгоритм декомпозиции нейронной сети использует в своей основе оригинальную математическую модель, подробнее ознакомиться с которой можно в Главе 2, как следствие, ограничения его использования состоят в следующем: с помощью алгоритма можно декомпозировать те нейронные сети, которые возможно отобразить с помощью предлагаемой математической модели. Подробнее о рассматриваемом алгоритме можно прочитать в работе [3]. Декомпозиции с помощью разработанного алгоритма могут подвергаться нейронные сети для распознавания, управления, классификации, генерации и других задач, которые решаются нейросетевыми методами. Возможны различные сочетания входных параметров алгоритма разделения, мы рассмотрим вариант, когда монолитная нейронная сеть, имеющая в своем составе K слоев, разделяется на D различных блочных нейронных сетей [42]. В рассматриваемом варианте алгоритма мы ориентируемся на равномерное распределение нагрузки на все вычислительные узлы, если это возможно. То есть мы пытаемся распределить на каждое устройство одинаковое число слоев из исходной нейронной сети. Для того, чтобы алгоритм учитывал различные вычислительные мощности вычислительных устройств достаточно поменять методику расчёта количества слоев для каждого устройства с равномерной на пропорциональную.

Рассмотрим предлагаемый алгоритм разделения по шагам (Рисунок 24).

1. Создается объект класса NNetwork. Объекты этого класса описывают как монолитные, так и блочные нейронные сети. Такие классы называются унифицированными классами.

2. В созданный объект считывается исходная монолитная нейронная сеть из файла.

NNetwork monoNetwork = NNetwork.ReadNNFromFile("/TSNetwork.ann");

3. Создается объект класса Separator, который отвечает непосредственно за разделение нейронной сети. Именно этот объект получит на вход параметры, отвечающие за различные варианты декомпозиции: число устройств D, массив вы-

числительных мощностей $\bar{P}$, флаг потребности учитывать пропускную способность канала q.

Separator separator = new Separator(D, P, q);

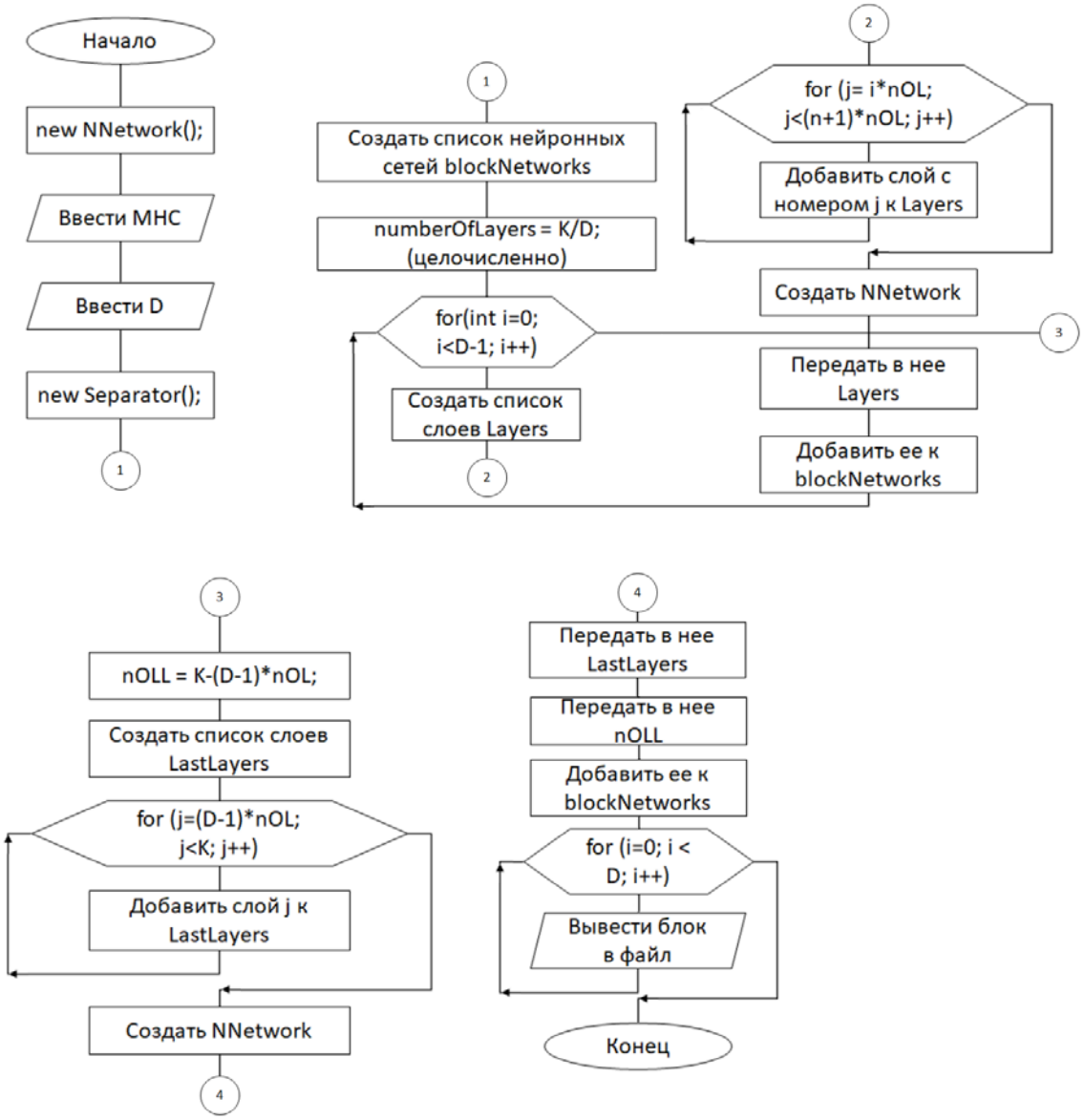


Рисунок 24 – Схема разработанного алгоритма декомпозиции MHC

4. Вызывается метод splitOnEqualNumberOfLayers объекта separator, реализующий разделение на блочные нейронные сети с равномерным распределением слоев исходной нейронной сети. На вход методу передается нейронная сеть monoNetwork и число устройств D, на которые требуется разделить нейронную сеть.

ArrayList<NNetwork> blockNetworks = separator.splitOnEqualNumberOfLayers (monoNetwork, D);

5. Дальнейшие операции производятся внутри метода splitOnEqualNumberOfLayers. Инициализируется список для хранения создаваемых блочных нейронных сетей blockNetworks.

```
ArrayList<NNetwork> blockNetworks = new ArrayList();
```

6. Подсчитывается число слоев, которое будет передано на каждую блочную нейронную сеть, кроме последней. Для этого осуществим целочисленное деление числа слоев K на число устройств D.

```
int numberOfLayers = network.K / D;
```

7. В цикле с итератором i, от 1 до D-1: создается массив слоев, которые будут переданы в очередную блочную нейронную сеть.

```
for (int i = 1; i< D; i++) {
   ArrayList<Layer> layers = new ArrayList();
   for (int j = i*numberOfLayers; j < (i+1)*numberOfLayers; j++) {
      layers.add(network.LayersList.get(j));
   }
   blockNetworks.add(new NNetwork(numberOfLayers, layers));
}
```

8. В цикле с итератором j, добавляются слои, которые будут переданы в блочную сеть, номера слоев считаются начиная с i*numberOfLayers. Окончание цикла по итератору j.

```
   for (int j = i*numberOfLayers; j < (i+1)*numberOfLayers; j++) {
      layers.add(network.LayersList.get(j));
   }
```

9. Создается новая блочная нейронная сеть, ей передается номер в соответствии с итератором i, число слоев и непосредственно слои монолитной нейронной сети layers. Нейронная сеть добавляется в список blockNetworks.

```
blockNetworks.add(new NNetwork(i, numberOfLayers, layers));
```

10. Окончание цикла по итератору i. Подсчитывается число слоев в последней блочной нейронной сети. Оно может быть отличным от numberOfLayers и получается вычитанием числа слоев во всех предыдущих слоях из общего числа слоев K.

int numberOfLastLayers = network.K - (D-1)*numberOfLayers;

11. Копируются все оставшиеся в монолитной нейронной сети слои, создается последняя нейронная сеть. Ей передаются слои lastLayers и сеть добавляется в общий список blockNetworks.

```
ArrayList<Layer> lastLayers = new ArrayList();
for (int j = (D-1)*numberOfLayers; j < network.K; j++) {
   lastLayers.add(network.LayersList.get(j));
}
blockNetworks.add(new NNetwork(D, network.K - (D-1)*numberOfLayers, lastLayers));
```

12. Блочные нейронные сети полностью готовы, они сохраняются в файлы. На вход функции сохранения подается список нейронных сетей blockNetworks и предпочитаемое название файлов.

NNetwork.SaveNNToFile (blockNetworks, "TSNetworks").

Каждая нейронная сеть сохраняется на диск в отдельном файле. Название файла формируется из переданной в функцию строки, но́мера нейронной сети в каскаде и расширения .ann. Например, TSNetworks1.ann для первой блочной нейронной сети.

4.3. Разработка алгоритма выбора оптимального варианта декомпозиции нейронной сети для реализации каскада нейросетевых устройств

На основании постановки научной задачи исследования (Раздел 1.4) разработан алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах, который реализует многокритериальную оптимизацию, строя Парето-оптимальное множество. Алгоритм состоит из 5 этапов, каждый из которых выполняет определенную операцию. В первом этапе вводятся требуемые характеристики систем, которые предполагают формирование непустого множества, которое требуется минимизировать. Во втором этапе происходит формирование множества возможных решений. В третьем этапе происходит вычисление минимальных поднаборов и формирова-

ние множества Парето. В четвертом этапе происходит вычисление оптимального набора или наборов для заданных параметров систем. В пятом этапе происходит формирование результатов в виде таблиц и графика. Схема предлагаемого алгоритма представлена на Рисунке 25.

Рисунок 25 – Схема предлагаемого алгоритма выбора оптимального варианта декомпозиции нейронной сети

В рассмотренном в данном исследовании случае, оптимизация происходила по четырем критериям: стоимости вычислительной системы, потребляемой каскадом мощности, времени осуществления нейросетевых вычислений и итоговом времени отклика системы на запросы обработки данных, которые не связаны с нейросетевой обработкой.

4.4. Разработка алгоритма запуска и работы распределенной НС

Для вычисления результатов работы блочной нейронной сети был разработан алгоритм вычисления результатов работы нейронной сети в предложенном формате (Рисунок 26).

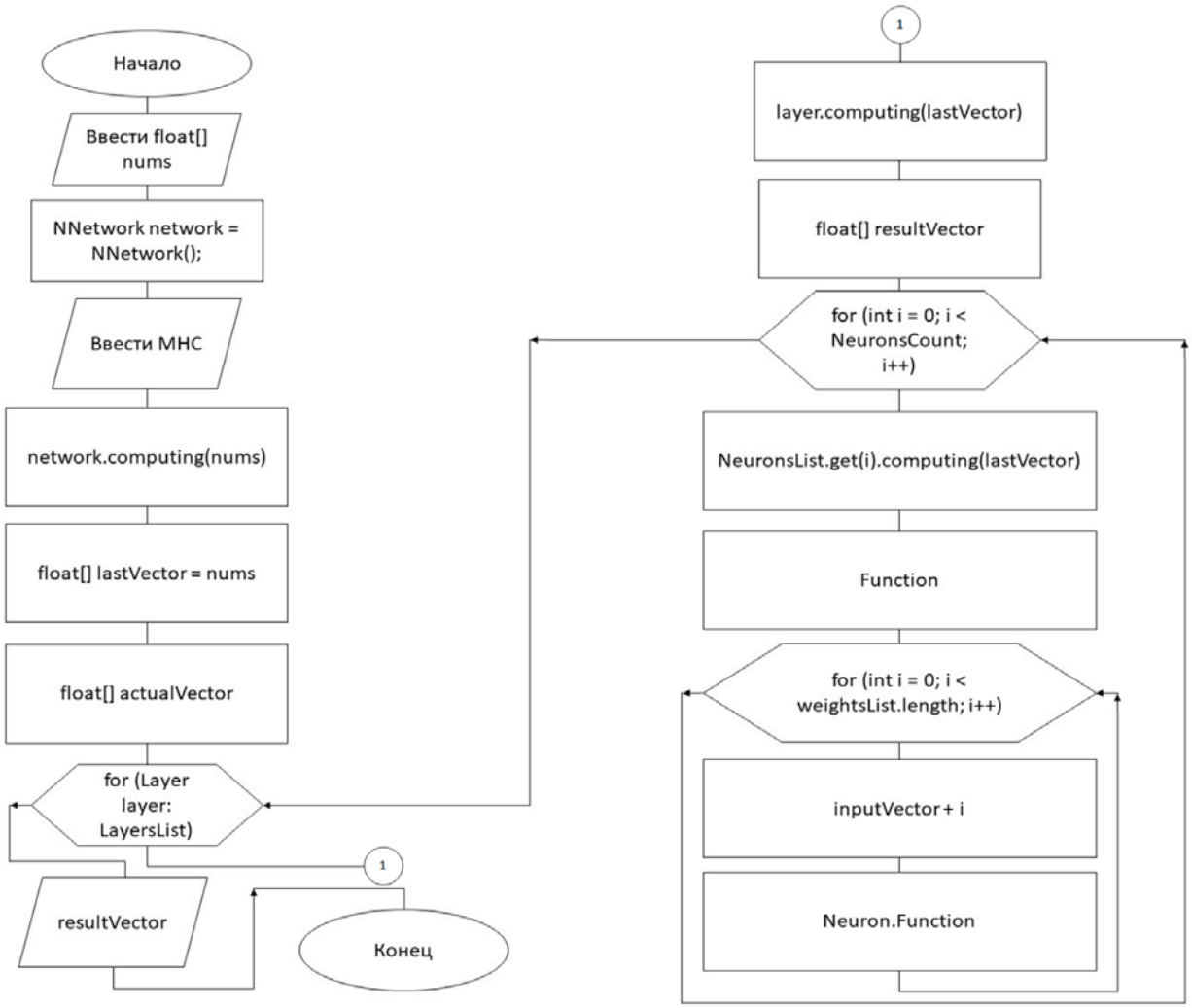


Рисунок 26 – Схема алгоритма вычисления результатов работы БНС

Рассмотрим алгоритм запуска и работы нейронной сети, которая получена из одного конкретного *.ann файла:

1. Получен входной вектор данных float[] nums.

2. Создается объект класса NNetwork, объекты этого класса описывают как монолитные, так и блочные нейронные сети. Такие классы называются унифицированными классами.

3. В этот объект считывается нейронная сеть из файла.

NNetwork network = NNetwork.ReadNNFromFile("/TSNetwork.ann");

4. Вызвать метод объекта NNetwork для вычисления результата работы нейронной сети computing.

float[] result = network.computing(nums);

5. Дальнейшие операции производятся внутри метода computing (NNetwork). Инициализируем векторы для хранения результатов работы текущего слоя actualVector и результатов работы предыдущего слоя lastVector, изначально инициализируем его входным вектором nums, полученным на вход нейронной сети.

```
float[] lastVector = nums;
float[] actualVector;
```

6. В цикле, который проходит итератором по всем элементам списка слоев нейронной сети LayersList, запускаем соответствующий метод вычисления результатов слоя computing для каждого слоя layer.

```
for (Layer layer: LayersList) {
    actualVector = layer.computing(lastVector);
    lastVector = actualVector;
}
```

7. Дальнейшие операции производятся внутри метода computing (Layer). Инициализируем результирующий вектор resultVector, каждое из значений в котором будет результатом вычисления конкретного нейрона, поэтому его размерность NeuronsCount (число нейронов на слое).

```
float[] resultVector = new float[NeuronsCount];
```

8. В цикле с итератором i от 0 до NeuronsCount-1, получаем из списка нейронов соответствующий нейрон get(i) и запускаем для нейрона метод computing.

```
for (int i = 0; i < NeuronsCount; i++) {
    resultVector[i] = NeuronsList.get(i).computing(lastVector);
}
```

9. Дальнейшие операции производятся внутри метода computing (Neuron). Вызывается метод вычисления результатов для класса Functions, отвечающего за работу всех функций активации нейронов. На вход подается номер функции активации f, константа для функции c, вектор весов синапсов weightsList, вектор номеров нейронов предыдущего слоя, с которыми связан данный нейрон, sinapses и вектор результатов работы предыдущего слоя lastVector.

```
return Functions.computing(f, c, weightsList, sinapses, lastVector);
```

10. Дальнейшие операции производятся внутри метода computing (Functions). Инициализируем переменную для результата result, и вектор, который пойдет на вход функции активации нейрона. В цикле с итератором i от 0 до weightsList.length-1, умножаем вес синапса weightsList[i] на входной сигнал, пришедший по соответствующему синапсу lastVector[sinapses[i]].

```
float result = 0;
float[] inputVector = new float[weightsList.length];
for (int i = 0; i < weightsList.length; i++) {
   inputVector[i] = weightsList[i]*lastVector[sinapses[i]];
}
```

11. Вызываем функцию calculate, передав ей номер функции, константу и входной вектор для функции активации inputVector.

```
result = calculate(f, c, inputVector);
```

12. В функции calculate выбирается по номеру функции соответствующая реализация функции активации. Вызывается функция программного кода, которая реализует данную функцию активации с нужными ей параметрами.

```
private static float calculate(NeuronFunctions f, float c, float[] inputVector) {
   if (f == NeuronFunctions.Equals) {
      return inputVector[0];
   }
   if (f == NeuronFunctions.Ntwo) {
      return (float) Functions.Ntwo(inputVector[0], inputVector[1]);
   }
   if (f == NeuronFunctions.Nthree) {
      return (float) Functions.Nthree(inputVector[0], inputVector[1], inputVector[2]);
   }
…
```

13. Возвращаем result из класса Functions, далее возвращаем этот же результат из класса Neuron.

14. Затем в классе Layer этот результат попадает в результирующий вектор с соответствующим номером resultVector[i], и этот вектор возвращается в класс NNetwork.

15. После прохождения итератора по всему списку LayersList последнее значение в векторе lastVector и является результатом работы нейронной сети. Возвращаем его на уровень выше и осуществляем дальнейшую обработку полученного результата.

Описанный алгоритм работает для распределенного случая на каждом из вычислителей с тем дополнением, что между вычислителями необходимо наладить каналы связи по unicast socket или иным доступным на устройстве каналам связи, по которым будет передаваться результат работы каждого из блоков блочной нейронной сети. Результирующий вектор первого блока станет вектором входных данных для второго блока и так далее, пока последний блок не выдаст результат работы всего каскада устройств. Обобщение требуемых шагов в единый алгоритм работы распределенной блочной нейронной сети будет выглядеть следующим образом:

1. Устройство реализующее блок с номером i получает вектор исходных данных либо от внешних акторов, для первого устройства в каскаде, либо от предыдущего устройства по каналам связи;
2. Обрабатывает полученный вектор с помощью блока с номером i (принимает значения от 1 до D соответственно);
3. Передает результаты обработки блоку i+1 по каналам связи, либо же выводит результат вовне, если это блок с номером D.

В случае отказоустойчивой реализации предлагаемого метода, результирующий вектор первого блока по дополнительно организованной связи передается на третий блок и станет входным вектором для третьего блока, в случае отказа или сбоя второго блока. То есть алгоритм функционирования распределенной блочной нейронной сети, в случае модифицированной отказоустойчивой реализации алгоритма с глубиной адаптации G будет выглядеть так:

1. Устройство реализующее блок с номером i получает G+1 векторов для всех устройств, кроме первого, поскольку у него возможен только один исходный вектор от внешних акторов, все остальные получают векторы либо от предыдуще-

го устройства по каналам связи, либо из внешней среды с помощью дополнительных каналов связи в количестве G;

2. Если вектор данных пришел по стандартному каналу с указанием номера устройства i-1, то:

3. Обрабатывает полученный вектор с помощью блока с номером i (принимает значения от 1 до D соответственно), выполняя только собственный блок вычислений;

4. Иначе, если за заданное время не пришло вектора данных по стандартному каналу, ожидается вектор от устройства i-2 (в случае его отсутствия i-3, вплоть до максимальной глубины адаптации, то есть до i-1-G);

5. Обрабатывает полученный вектор с помощью блоков с номерами i-1 и i (в случае получения данных от более младших устройств, аналогично до блока i-G), выполняя собственный блок вычислений и необходимые блоки отказавших устройств перед собой;

6. Для блоков с номером D-G и далее, проверить наличие обработки данных на блоках, идущих впереди с помощью дополнительно созданных обратных связей, в случае отсутствия ответа от следующего блока – принять на себя и его вычислительные функции, выполнив блок с номером i+1 (и до D при необходимости)

7. Передает результаты обработки блоку i+1 по каналам связи, либо же выводит результат вовне, если это блок с номером D, к вектору при передаче добавляется маркер порядкового номера устройства, которое ведет передачу выходного вектора.

Разработанные алгоритмы были реализованы в виде программных продуктов NNSplitter и NNImplementer на языке программирования JAVA [27, 28].

4.5. Оценки сложности разработанных алгоритмов

В процессе диссертационного исследования с целью оценки эффективности разработанного метода синтеза нейросетевых устройств в режиме туманных вычислений была проведена оценка временной сложности разработанных алгоритмов: алгоритма декомпозиции МНС на блоки БНС, алгоритма работы блочной нейронной сети и алгоритма выбора оптимальной декомпозиции НС по определенным параметрам. Все представленные алгоритмы имеют полиномиальную сложность.

Временная сложность алгоритма (в худшем случае) — это функция от размера входных данных, равная максимальному количеству элементарных операций, проделываемых алгоритмом для решения экземпляра задачи указанного размера. Запись вида $f(n) = O(g(n))$ означает, что функция $f(n)$ возрастает медленнее, чем функция $g(n)$ при c = c1 и n = N, где c1 и N могут быть сколь угодно большими числами. При записи и оценке сложности в Big O Notation [62] (в виде функции $O(n)$) не происходит точной оценки сложности, а осуществляется оценка сложности максимального порядка, то есть в результате получится не число операций, а функция, порядок роста которой не будет превзойден функцией, описывающей временную сложность алгоритма. В записи данным способом существуют следующие правила: константы отбрасываются, если в O есть сумма, нас интересует самое быстрорастущее слагаемое, перемножение сложностей функций происходит в том случаи, если они выполняются во вложенном цикле (одна относительно другой).

По итогам оценки сложности предлагаемых алгоритмов были получены следующие результаты асимптотической оценки сложности по времени (безразмерная величина, представленная в Big O Notation).

Алгоритм декомпозиции МНС:

$$O\left(K * \max\left(\left\{H_0,\ldots,H_{K-1}\right\}\right)\right)$$

Алгоритм вычисления результатов работы НС в формате *.ann:

$$O\left(K * \max\left(\left\{H_0,\ldots,H_{K-1}\right\} * \max\left(\left\{H_0,\ldots,H_{K-1}\right\}\right)\right)\right)$$

Порядок сложности вычислений результатов блока на устройстве будет рассчитываться аналогично, с поправкой на то, что в блок войдет лишь часть слоев нейронной сети.

Поэтому оценка сложность вычислений, проводимых одним вычислительным устройством:

$$O\left(L_i * \max\left(\left\{H_0,\ldots,H_{L_i-1}\right\} * \max\left(\left\{H_0,\ldots,H_{L_i-1}\right\}\right)\right)\right)$$

Самым эффективным как по времени, так и по нагрузке на сеть будет решение с использованием всех предлагаемых оптимизаций.

Модификация алгоритма вычисления результатов функционирования распределенной блочной нейронной сети, с глубиной адаптации G имеет максимальную оценку сложности вычислений, проводимых одним вычислительным устройством:

$$O\left(L_i * \max\left(\left\{H_0,\ldots,H_{L_i-1}\right\} * \max\left(\left\{H_0,\ldots,H_{L_i-1}\right\}\right)\right) * G\right)$$

И минимальную оценку сложности, для случая выполнения устройством только своих функций, которая совпадет с соответствующей оценкой для оригинального алгоритма.

4.6. Выводы по главе

Первый из предлагаемых алгоритмов представляет пошаговую трассировку декомпозиции монолитной нейронной сети на блоки нейронной сети, ориентированной на туманные вычисления. Второй алгоритм – позволяет взять любой из полученных в результате работы первого алгоритма блок нейронной сети и заставить его функционировать в рамках нейросетевого устройства. Разработанные алгоритмы могут быть реализованы на любом языке программирования, который поддерживает объектно-ориентированную парадигму, а также на языках, работающих в других парадигмах, но потребует небольших корректировок для работы напрямую с памятью, минуя объекты. Экспериментальные реализации производились на языке программирования высокого уровня JAVA, проведенные оценки сложности показали, что предлагаемые алгоритмы способны за необходимое время решать задачи декомпозиции и функционирования распределенной нейронной сети. Оценки сложности соответствуют оценкам сложности функционирования монолитной нейронной сети, то есть не произошло понижения скорости вычислений относительно монолитной реализации. Само собой, за исключением накладных расходов, которые потребуются для передачи данных по каналам связи, и операций ввода-вывода в буферы исходных данных и результатов работы каждого из узлов, но эти временные затраты не относятся к алгоритмам, которые были рассмотрены в данной главе.

ГЛАВА 5. СХЕМОТЕХНИЧЕСКОЕ МОДЕЛИРОВАНИЕ, ПРОТОТИПИРОВАНИЕ И ОЦЕНКА ЭФФЕКТИВНОСТИ МЕТОДА СИНТЕЗА НЕЙРОСЕТЕВЫХ УСТРОЙСТВ НА СХЕМОТЕХНИЧЕСКИХ МОДЕЛЯХ И ФИЗИЧЕСКИХ УСТРОЙСТВАХ

5.1. Схемотехническое моделирование и прототипирование нейросетевых устройств, разработанных предлагаемым методом, в режиме туманных вычислений

Для схемотехнического моделирования и прототипирования были разработаны программы нейросетевых устройств реализации искусственных нейронных сетей на программируемой логике [37, 55, 67], ориентированных на туманные вычисления [65, 100]. Для этого был проанализирован путь от исходной монолитной нейронной сети до нейросетевого устройства, реализующего конкретный вычислительный блок распределенной блочной нейронной сети.

Для работы с нейронными сетями в унифицированном виде на устройствах с программируемой логикой потребовалось создать легковесный формат хранения нейронной сети в файле. Формат файлов получил расширение .ann – сокращенное от artificial neural network. МНС в данном формате подается на вход алгоритму декомпозиции нейронной сети, также ему на вход подается число вычислительных устройств D, на которые требуется декомпозировать имеющуюся сеть. В нашем эксперименте оставшимися параметрами были условие разделение по слоям и разделение на примерно равные части, то есть случай вычислительных устройств одинаковой производительности. В результате декомпозиции получаем D файлов в формате *.ann, в каждом из которых находится один из блоков распределенной нейронной сети. В файле для разделения номера блока, числа слоев и описания каждого из слоев используется символ перевода каретки на новую строку, но в дальнейшем полученное представление нейронной сети будет использовано в программе для микроконтроллера, поэтому разделитель верхнего уровня был заменен на символ «:».

Для разработки и компиляции прошивки для микроконтроллера ATMega32 [46] было использовано программное обеспечение IDE CodeVisionAVR [61]. CodeVisionAVR — интегрированная среда разработки программного обеспечения для микроконтроллеров семейства AVR фирмы Atmel. Для реализации программы для вычислительного узла использовался язык программирования C. При разработке программы для контроллера создается проект, при создании выбирается модель конкретного микроконтроллера, для которого создается прошивка, в нашем случае ATMega32. Также при создании указываются стартовые параметры, то есть значения регистров конфигураций, отвечающие за настройку режима работы контроллера, при необходимости существует возможность поменять эти значения уже непосредственно в программном коде разрабатываемого проекта. Нейронная сеть устанавливается в прошивку копированием содержимого файла *.ann в исходный код программы в виде строки char nnetworkString[]. Пример интеграции описания нейронной сети в код программы представлен на Рисунке 27.

```
69  unsigned long delay = 0;
70  unsigned int inputVector = 0;
71  unsigned int outputVector = 0;
72  int time = 0;
73  int fo=0;
74  char nnetworkString[] = "30:0,8,0;12;1;1;0,1;12;1;1;1,2;12;1;1;2,4;12;1;1;0,5;
```

Рисунок 27 – Пример переноса нейронной сети в код программы для контроллера

Существует три типа программ, различных по способу получения и обработки данных:

- для первого контроллера, с обработкой входных данных;
- для промежуточных вычислителей с получением, обработкой и отправкой данных;
- для последнего контроллера с получением, обработкой и выводом результатов.

Рассмотрим подробнее этот вариант прошивки, его можно увидеть на Рисунке 28.

```
515   while (1)
516         {
517           if (fo==1)
518           {
519            fo=0;
520            time = TCNT2;
521            outputVector = computingNN(inputVector);
522            delay = countDelay(TCNT2, time);
523            printf("RV %u T%i %u%c%c", outputVector, j,delay,10,13);
524            PORTA=outputVector;
525            j++;
```

Рисунок 28 – Основной код программы финального контроллера

Флаг *f0* выставляется в 1 в рамках прерывания получения данных *interrupt [USART_RXC] void usart_rx_isr(void)*, которое вызывается при получении определенного количества бит, здесь же полученные данные помещаются во входной вектор *inputVector = data*. В основной же части в переменную *time* помещается значение счетного регистра TCNT2, содержащего число тиков таймера на текущий момент, по сути – локальное время выполнения. Далее осуществляется вызов непосредственно функции работы нейронной сети, встроенной в данную программу. В переменную *delay* помещается вычисленная разница между временем после окончания вычисления результатов работы сети и переменной *time*. Полученный вектор и задержка в тиках выводятся на экран виртуального терминала. Полученный вектор отправляется на PORTA для демонстрации на сигналах светодиодов.

Схемотехническое моделирование различных конфигураций каскадов нейросетевых устройств было осуществлено в САПР Proteus [48, 91, 97]. Произведено моделирование работы различных вариантов нейронных сетей: монолитной нейронной сети на одном микроконтроллере, чтобы иметь эталонный вариант работающей модели, и представлены модели, в которых последовательно соединялись друг с другом три, пять и семь микроконтроллеров, соответственно. Для каждого из случаев применения нескольких вычислителей в каскаде осуществлялась декомпозиция исходной монолитной нейронной сети в соответствии с количеством контроллеров.

Устройством, которое будет реализовывать нейросетевые вычисления был выбрал микроконтроллер ATmega32 фирмы ATMEL семейства AVR. представляет собой 8-разрядный микроконтроллер общего назначения с RISC-ядром [51, 79]. Кри-

сталлы AVR имеют в своем составе Flash-память программ, память данных EEPROM и SRAM, порты ввода-вывода и типовые для 8-битных микроконтроллеров периферийные устройства [81]. Что немаловажно для нашей задачи, микроконтроллер имеет программируемый USART, что и позволит нам передавать от одного устройства к другому результаты работы одного блока нейронной сети, которые станут исходными данными для следующего блока. Объем Flash-памяти 32 КБайта, что и будет предельным размером загружаемой для исполнения на микроконтроллере программы. Реализация программы для вычислительного устройства будет происходить на языке C, а именно его имплементации MicroC, созданной специально для реализации программ, которые будут исполняться на микроконтроллерах [60].

Настройка передачи с использованием UART начинается с непосредственно соединения пина PD1/TXD (трансмиттера UART) первого контроллера с пином PD0/RXD (ресивера UART) принимающего контроллера, но этого еще недостаточно для полноценной работы. Все настройки приемопередатчика хранятся в регистрах конфигурации UCSRA, UCSRB и UCSRC. Самое главное осуществить конфигурацию UCSRB в битах 3 - TXEN и 4 – RXEN, их необходимо выставить в единицу для разрешения передачи и приема соответственно. Далее требуется установить в единицу флаги 6 – TXCIE и 7 – RXCIE, разрешающие прерывания передачи и приема, нас интересуют в первую очередь прерывания приема, которые активно использовалось для получения данных из канала UART. У AVR есть регистр UDR это UART Data Register. На самом деле это два разных регистра, но имеют один адрес. Просто на запись попадает в один (регистр передатчика), а на чтение берет из другого (регистр приемника). Настройки микроконтроллера, с которыми осуществлялось его использование представлены на Рисунке 29.

Микроконтроллер в наших экспериментах работал на частоте 8 МГц. В случае с одним микроконтроллером используется генератор слов PATTERN GENERATOR, для генерации входных сигналов, который передавал сгенерированное слово на настроенные как входы пины PB1-8. Также к PC0 и PC1 были подключены выходы генератора слов CLKOUT и CASCADE, которые отвечали за генерацию сигнала при отправке нового слова и отправке последовательности с самого

начала соответственно. Пины PA1-8 были настроены как выходы и передавали на LED-RED светодиоды информацию о результате вычислений, произведенных монолитной нейронной сетью. Также результат передавался на Virtual Terminal который подключался к PD1/TXD, то есть трансмиттеру UART и получал результаты вычислений на табло для каждого из подаваемых примеров данных. Описанная схема подробно представлена на Рисунке 30.

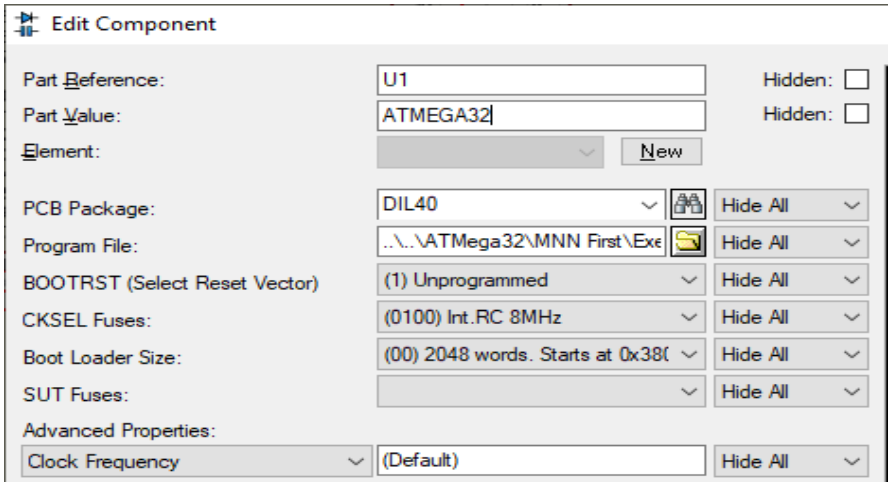


Рисунок 29 – Параметры запуска контроллера ATMega32

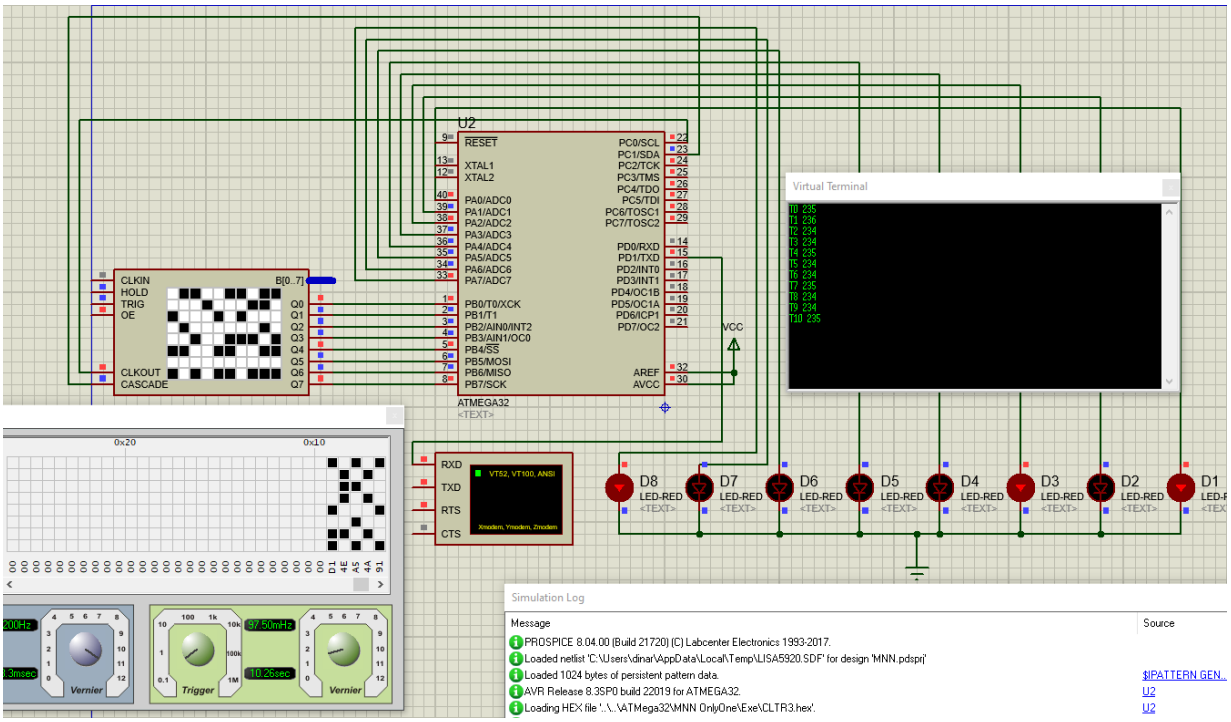


Рисунок 30 – Схема электрическая функциональная работы МНС на одном микроконтроллере ATMega32 в системе схемотехнического моделирования

В первом эксперименте МНС была разделена на три блока, каждый из которых исполнялся на отдельном микроконтроллере [4]. Для этого нейросетевая обработка данных была частично разделена между контроллерами. Так первый контроллер остался подключенным к генератору слов, получал от него вектор исходных данных и передавал контроллеру сигналы синхронизации. Третий, то есть последний контроллер получил функции вывода результирующих данных, как на массив светодиодов LED-RED1-8, так и на экран виртуального терминала, используя трансмиттер UART, то есть пин PD1/TXD.

Соединение же первого и второго микроконтроллеров осуществлялось с помощью UART, а именно трансмиттер первого микроконтроллера TXD соединялся с ресивером второго контроллера RXD. Подробно схема представлена на Рисунке 31.

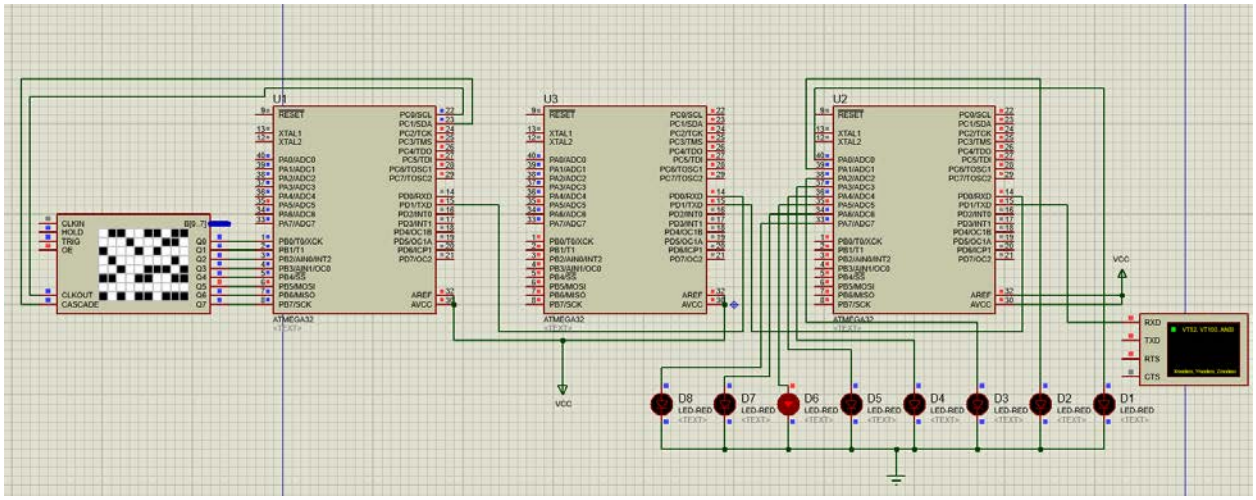


Рисунок 31 – Схема электрическая функциональная работы БНС на трех микроконтроллерах ATMega32 в системе схемотехнического моделирования

Похожим образом собирается схема на пять микроконтроллеров, первый микроконтроллер получает входные данные, осуществляет вычисления и передает результат по UART второму. Следующий принимает данные, обрабатывает их и передают контроллеру номер три. Аналогично отрабатывают третий и четвертый вычислительные узлы, необходимо четко следить, чтобы на промежуточных контроллерах биты управления передачей и приемом, а также биты, разрешающие прерывания по отправке и получения выставлялись в единицы. Последнее вычис-

лительное устройство завершает обработку и выводит результаты на массив светодиодов и в терминал. С результатами схемотехнического моделирования можно ознакомиться на Рисунке 32. Схожим образом, но с добавлением еще двух промежуточных вычислительных блоков осуществляется моделирование каскада из семи нейросетевых устройств.

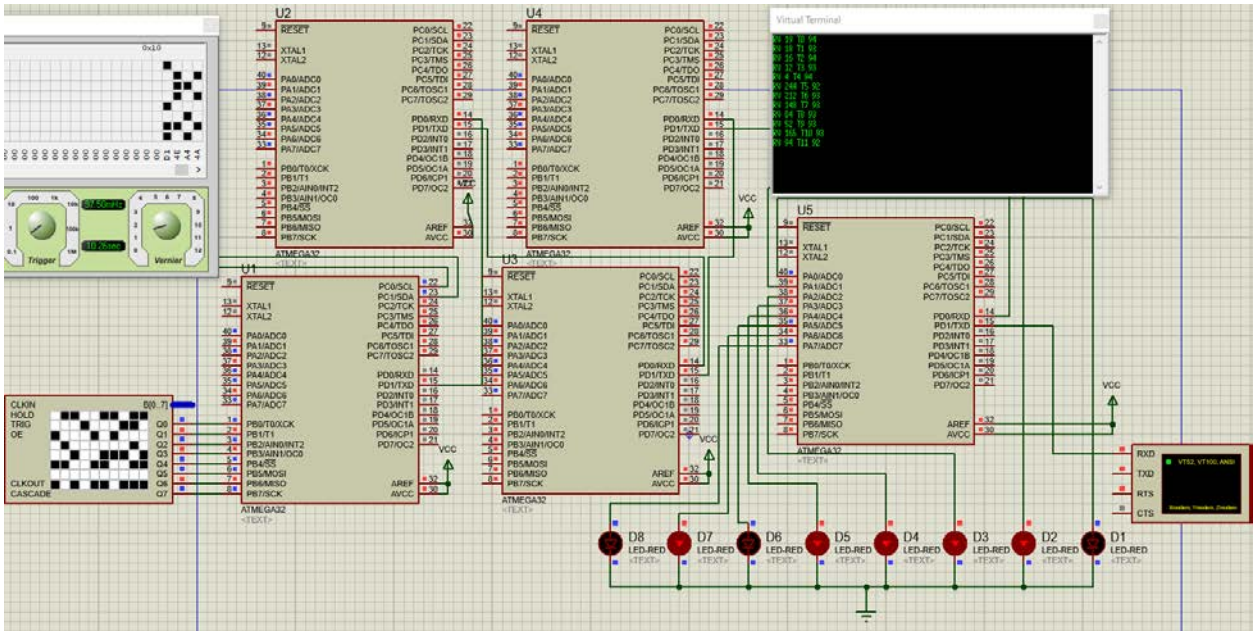


Рисунок 32 – Схема электрическая функциональная работы БНС на пяти микроконтроллерах ATMega32 в системе схемотехнического моделирования

В процессе исследования было выполнено прототипирование по разработанной схеме (Рисунок 31) также был собран испытательный стенд на базе двух микроконтроллеров ATMega32A, которые были объединены в каскад нейросетевых устройств, с их помощью метод синтеза был апробирован на физических вычислительных устройствах, собранный стенд представлен на Рисунке 33.

Для тестирования работоспособности предлагаемого метода синтеза нейросетевых устройств на процессорных системах был собран испытательный стенд на основе двух одноплатных компьютеров Raspberry Pi 4 Model B [98], которые были объединены в каскад средствами Ethernet через маршрутизатор, к которому также был подключен ноутбук, позволявший подключаться к ним и следить за ходом экспериментов, собранный стенд представлен на Рисунке 34.

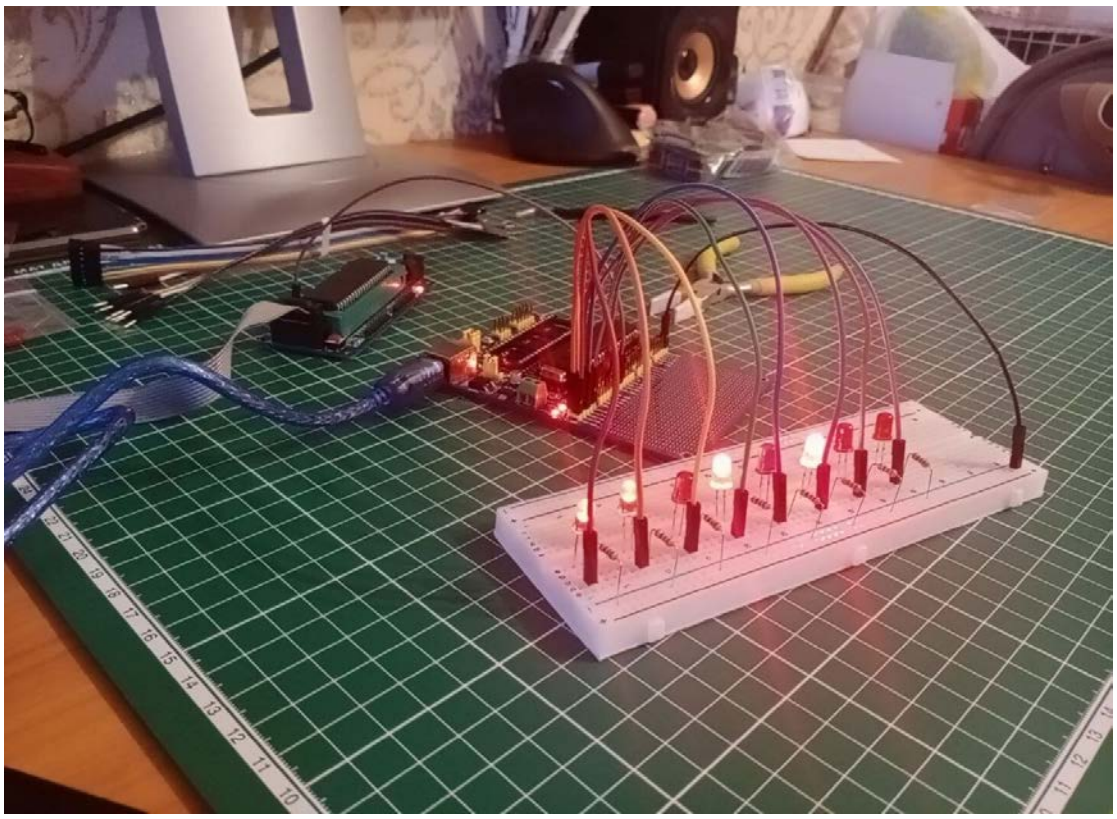


Рисунок 33 – Каскад из двух микроконтроллеров ATMega32A, реализующий работу блочной нейронной сети

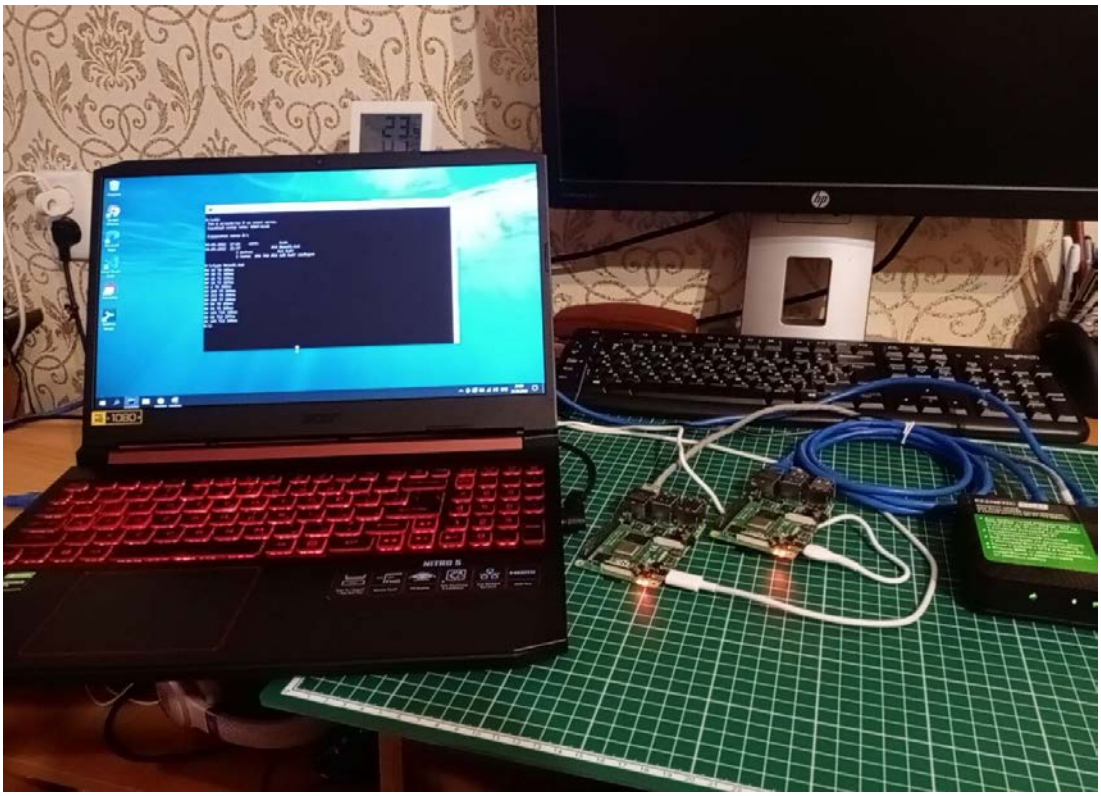


Рисунок 34 – Каскад из двух одноплатных компьютеров Raspberry Pi 4 Model B, реализующий работу блочной нейронной сети

Представленные стенды подтвердили работоспособность предлагаемого усовершенствованного метода синтеза, поскольку результаты работы блочной нейронной сети на обоих стендах оказались идентичны тем результатам, которые были получены в процессе моделирования на схемотехнической модели в системе моделирования Proteus.

Модификация метода синтеза, которая позволяет осуществлять диагностику и реконфигурирование, потребовала в том числе изменений в электрических функциональных схемах каскадов БНС, а именно – в создании дополнительных связей между устройствами. Для примера будет рассмотрена схема из трех микроконтроллеров с глубиной адаптации G = 1, то есть с потенциальной возможностью контроллеров взять на себя функции соседнего устройства, обновленная электрическая функциональная схема представлена на Рисунке 35.

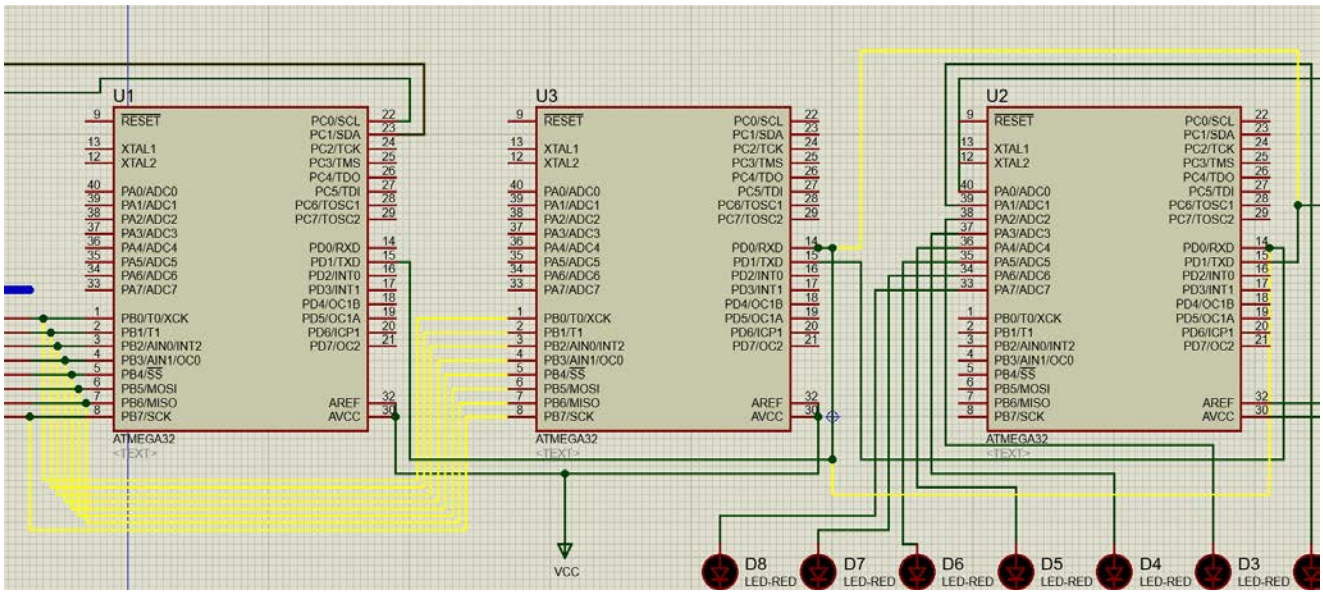


Рисунок 35 – Схема электрическая функциональная БНС на трех микроконтроллерах ATMega32 с дополнительными связями для обеспечения отказоустойчивости

На представленном рисунке добавлены связи между внешним генератором и вторым контроллером в каскаде, связь между вторым и третьим устройствами, и обратная связь от последнего устройства к предпоследнему для обеспечения диагностики третьего устройства в каскаде.

5.2. Оценка эффективности метода синтеза нейросетевых устройств для реализации туманных вычислений

Было проведено несколько запусков различных схемотехнических моделей. Первым было осуществлено моделирование исполнения МНС на одном микроконтроллере. Далее осуществлена декомпозиция нейронной сети на блоки для исполнения на трех, пяти и семи контроллерах. На вход каждой модели были последовательно представлены одинаковые исходные данные, созданные в генераторе слов. Результаты всех описанных итераций моделирования представлены в Таблице 3.

Таблица 3 – Результаты работы каждой из схем на одинаковых входных параметрах

1 ATMega32(МНС)	3 ATMega32(БНС)	5 ATMega32(БНС)	7 ATMega32(БНС)
00010011	00010011	00010011	00010011
00010010	00010010	00010010	00010010
00010000	00010000	00010000	00010000
00001100	00001100	00001100	00001100
00000100	00000100	00000100	00000100
11110100	11110100	11110100	11110100
11010100	11010100	11010100	11010100
10010100	10010100	10010100	10010100
01010100	01010100	01010100	01010100
00110100	00110100	00110100	00110100
00100100	00100100	00100100	00100100

Полученные результаты подтверждают, что предложенный метод синтеза нейросетевых устройств для реализации режима туманных вычислений работает верно, ведь независимо от того, выполняется нейронная сеть на одном устройстве, или вычисления происходят в каскаде с различным числом вычислительных устройств, результаты во всех перечисленных случаях получаются одинаковыми. Метод синтеза позволяет получить каскад блоков блочной НС, который дает те же самые результаты, что и исходная монолитная НС. Для тестирования предложенного усовершенствованного метода синтеза нейросетевых устройств были использованы различные нейронные сети с подходящими архитектурами: нейронная сеть для оценки погодный условий, которая уже упоминалась ранее в разделе с доказательством адекватности математической модели, а также нейронные сети из предыдущих проведенных исследований. Эти исследования были посвящены вопросам компьютерной лингвистики, а именно работе в направлении NLP (Natural Language

Processing), а именно поиску и классификации новых научных терминах в текстах на естественных языках, которые относились к определенным категориям научного знания [41, 68]. Для тестирования использовалось несколько различных нейронных сетей, которые были использованы в программном продукте TSBuilder [43, 44, 45] в качестве нейросетевого классификатора новых составных терминов (терминов из нескольких слов), которые были найдены в тексте. Результаты работы нейросетевой классификации в дальнейшем использовались для построения онтологий в определенных сферах научного знания [69, 70]. Далее были осуществлены замеры времени выполнения вычислений на одном микроконтроллере при каждой декомпозиции, измерялось время выполнения на последнем в каскаде вычислительном узле. Для корректной работы таймера в контроллерах ATMega32 необходимо выставить значение предделителя, то есть числа пропускаемых сигналов внутреннего тактового генератора, который был настроен на частоту 8 МГц. Предделитель устанавливается в регистре управления TCCR2, в битах CS20, CS21 и CS22, в наших экспериментах они были выставлены в единицы, что соответствует предделителю в 1024. Таким образом при данном предделителе и установленной частоте один тик микроконтроллера будет составлять примерно 92 мкс. Средние результаты числа тиков таймера первого эксперимента представлены на Рисунка 36, 37.

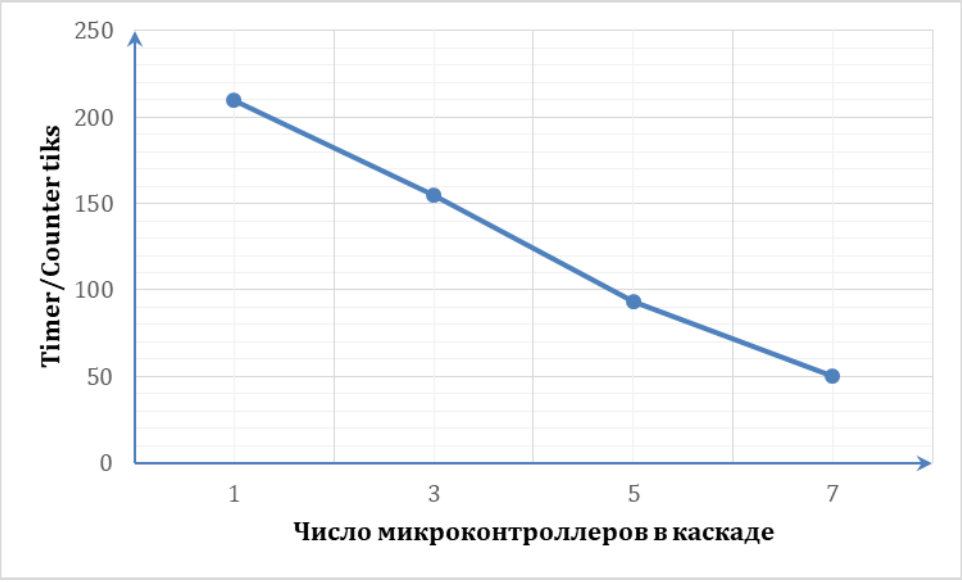
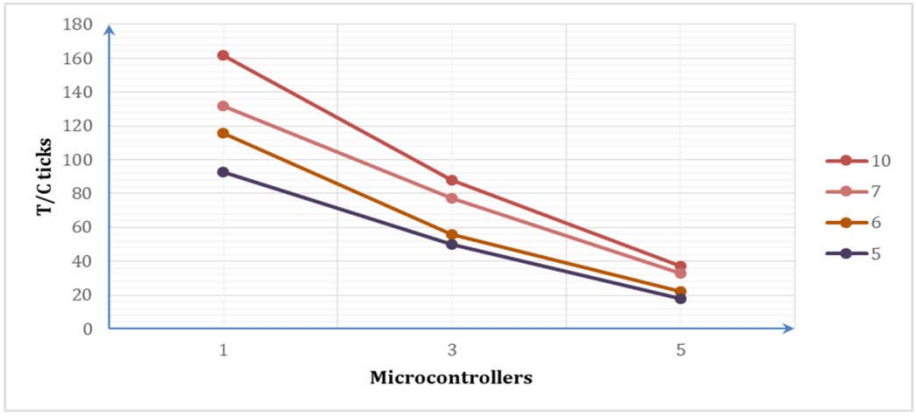
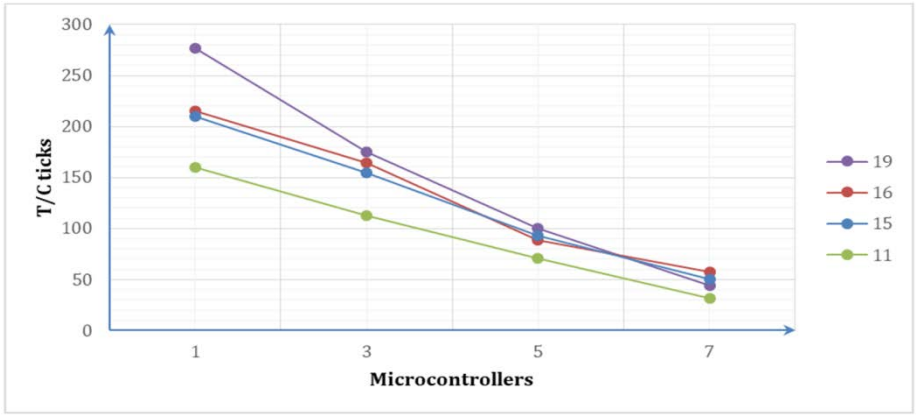


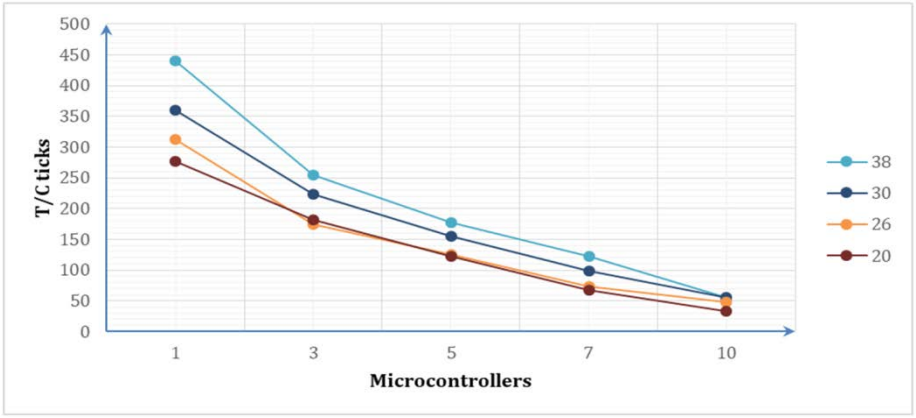
Рисунок 36 – Зависимость затраченного одним контроллером числа тиков внутреннего таймера от числа микроконтроллеров в первом схемотехническом моделировании



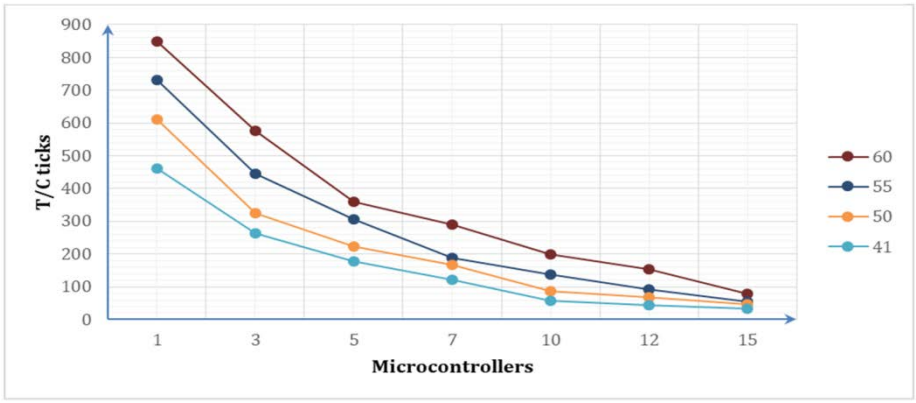
а



б



в



г

Рисунок 37 – Зависимость числа тиков внутреннего таймера
от числа микроконтроллеров для нейронных сетей различной сложности:
а) 5-10 слоев; б) 10-20 слоев; в) 20-40 слоев; г) 40-60 слоев

Полученные результаты говорят о том, что нагрузка на конкретный вычислительный узел сокращается, пропорционально проценту оставшихся слоев НС для вычисления, с поправкой на накладные вычислительные расходы. Полученные результаты подтверждают эффективность предложенного метода.

При частоте 8 МГц один «тик» таймера микроконтроллера (Tick clock – понятие, связанное с обработкой прерываний) будет составлять порядка 92 мкс.

Подтверждена теорема об эквивалентности монолитной и блочной нейронных сетей. Полученные результаты моделирования показывают, что представленная теорема, доказанная выше, подтверждается также эмпирическими данными. Результирующие вектора совпадают между собой.

В процессе работы удалось также эмпирически подтвердить теорему о существовании блочной нейронной сети, создав из исходной монолитной нейронной сети блочные нейронные сети различных размерностей.

В рамках моделирования и проведения экспериментов по запуску блочных нейронных сетей осуществлялась декомпозиция двадцати различных монолитных нейронных сетей. Каждая из них была декомпозирована на каскады нескольких вычислительных устройств. На вход блочной нейронной сети в каждом из моделирований направлялись пятьдесят одинаковых векторов входных параметров.

Количество проведенных моделирований и испытаний блочных нейронных сетей, позволяет считать метод эффективно решающим поставленные задачи. Среди 20 моделирований блочных нейронных сетей 20 дали результат, совпадающий с исходной монолитной нейронной сетью, что позволяет говорить об эффективности в 100%. Кроме того, результаты моделирования показывают, что каскад из блочных нейронных сетей сохраняет основные параметры монолитной нейронной сети, из которой он был декомпозирован. Например, Точность (Ac, Accuracy) — это способность модели давать правильный результат относительно общего количества исследований. Это происходит потому, что блочная нейронная сеть сохраняет все слои нейронов и связи между ними, которые были в исходной сети.

Для тестирования модификации метода синтеза было проведено несколько запусков различных схемотехнических моделей с дополнительными связями ме-

жду устройствами и искусственно созданными отказами определенных устройств. Осуществлена декомпозиция нейронной сети на блоки для исполнения на трех и пяти микроконтроллерах. На вход каждой модели были последовательно представлены одинаковые исходные данные, созданные в генераторе слов. Результаты всех описанных итераций моделирования представлены в Таблице 4.

Таблица 4 – Результаты работы каждой из схем на одинаковых входных параметрах

3 ATMega32(МНС)	3 (отказ узла 2) ATMega32(БНС)	5 ATMega32(БНС)	5 (отказ узла 1) ATMega32(БНС)
11001001	11001001	11001001	11001001
00010010	00010010	00010010	00010010
00010000	00010000	00010000	00010000
00000100	00000100	00000100	00000100
11110100	11110100	11110100	11110100
11010100	11010100	11010100	11010100
01010100	01010100	01010100	01010100
00100100	00100100	00100100	00100100

Полученные результаты подтверждают, что предложенная модификация метода синтеза нейросетевых устройств для реализации режима туманных вычислений позволяет осуществить адаптацию с заранее заданной глубиной G, ведь независимо от того, выполняется нейронная сеть на полностью исправном каскаде устройств, или вычисления происходят в каскаде с отказом одного из вычислительных узлов, результаты во обоих случаях получаются одинаковыми.

5.3. Оценка эффективности по результаты внедрения метода синтеза нейросетевых устройств для реализации туманных вычислений

Оценка проводилась по декомпозиции конкретной нейронной сети пятью способами. Для четырех различных конфигураций оборудования: одного без изменений, и трех с установкой дополнительного вычислительного устройства. Рассматриваемая в примере поиска оптимального решения нейронная сеть для биометрической идентификации состояла из 50 слоев и более чем 1000 математических нейронов (FingerNet, модификация модели ResNet50). Точность (Accuracy) рассматриваемой монолитной нейронной сети составляла 95.7% [88], точность

всех полученных в результате различных вариантов декомпозиции блочных НС совпала с точностью исходной нейросети.

В текущей конфигурации вычислительная система состоит из следующих элементов: микроконтроллер Atmel AT91SAM7X256-AU [103], коммутатор D-Link DGS-1100-05PDV2, одноплатный микрокомпьютер Raspberry PI Zero, 3-портовый управляемый коммутатор 10/100 Ethernet KSZ8993M, исходная схема каскада вычислительных устройств представлена на Рисунке 38.

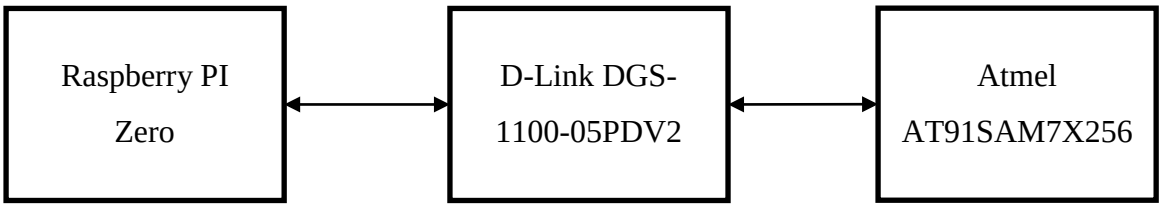


Рисунок 38 – Исходная схема вычислительного каскада, использованного во время внедрения

В случае исходной архитектуры будет происходить каскадирование в режиме остаточного ресурса, то есть в рамках реализации нейросетевых вычислений будут использоваться оставшиеся относительно свободными вычислительные мощности на уже имеющихся в исходной схеме устройствах.

В других рассмотренных случаях предлагалось добавить в каскад дополнительное вычислительное устройство: в первом случае – микроконтроллер ATMega32, во втором – еще один одноплатный компьютер Raspberry PI Zero, в третьем – Raspberry PI Model 4 B. Схема альтернативной архитектуры каскада представлена на Рисунке 39.

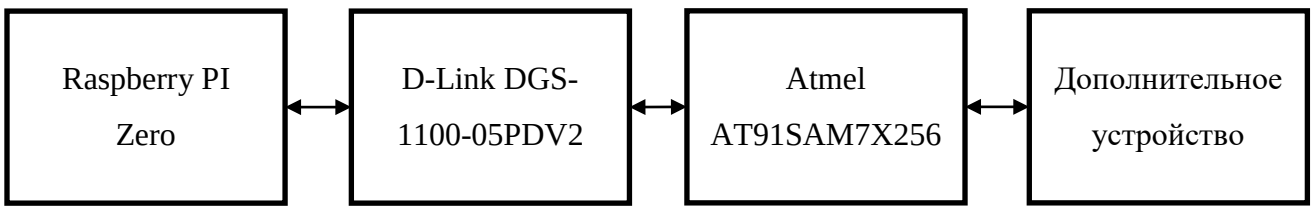


Рисунок 39 – Расширенная схема вычислительного каскада, использованного во время внедрения

В этом случае будут рассмотрены как варианты реализации монолитной нейронной сети одиночным добавленным вычислителем, так и варианты добавления нового вычислителя в каскад с целью распределения на него части нейросетевой нагрузки параллельно с тем, чтобы использовать ресурсы уже имеющихся устройств.

Каждый из четырех предложенных вариантов реализации, по результатам анализа исходных данных и прототипирования вычислительных каскадов, был оценен по основным параметрам оптимизации для рассматриваемой задачи: стоимость $\boldsymbol{C}$, энергопотребление $\boldsymbol{W}$, время выполнения нейросетевых вычислений на конкретном каскаде вычислителей $\boldsymbol{T^1}$ и время отклика используемого пульта голосования на поступившую извне команду $\boldsymbol{T^2}$. Результаты полученных оценок по предлагаемым параметрам оптимизации представлены ниже в Таблицах 5-8.

Ниже представлены исходные значения параметра стоимость $\boldsymbol{C}$ в Таблице 5.

Таблица 5 – Стоимость каскада вычислительных устройств $\boldsymbol{C}$

Стоимость	C (Руб.)	ΔC (Руб.)	ΔC (%)
Схема 1 (Исходная)	10601	0	0
Схема 2 (Добавлено у-во 1)	12211	1610	15,18725
Схема 3 (Добавлено у-во 2)	15232	4631	43,68456
Схема 4 (Добавлено у-во 3)	21501	10900	102,8205

И исходные значения параметра энергопотребление $\boldsymbol{W}$ в Таблице 6.

Таблица 6 – Энергопотребление каскада вычислительных устройств W

Энергопотребление	W (Вт.)	ΔW (Вт.)	ΔW (%)
Схема 1 (Исходная)	6,676	0	0
Схема 2 (Добавлено у-во 1)	6,811	0,135	2,022169
Схема 3 (Добавлено у-во 2)	7,526	0,85	12,73217
Схема 4 (Добавлено у-во 3)	10,176	3,5	52,4266

И полученные значения параметра T^1 представлены в Таблице 7.

Таблица 7 – Время выполнения нейросетевых вычислений на конкретном каскаде вычислителей T^1

Параметр	T^1исх (мс)	$T^1$1 (мс)	$T^1$2 (мс)	$T^1$3 (мс)
Монолитная сеть на устройстве	-	139	57	22
Способ 1 (Одинаково слоев)	104	125	95	82
Способ 2 (Пропорционально слоев)	86	107	85	53
Способ 3 (Одинаково нейронов)	103	123	94	82
Способ 4 (Пропорционально нейронов)	84	104	85	52
Способ 5 (Минимум отправки данных)	89	102	91	80

Значения параметра T^2 представлены в Таблице 8.

Таблица 8 – Время отклика используемого пульта голосования на поступившую извне команду T^2

Параметр	T^2исх (мс)	$T^2$1 (мс)	$T^2$2 (мс)	$T^2$3 (мс)
Монолитная сеть на устройстве	65	-	-	-
Способ 1 (Одинаково слоев)	134	111	111	111
Способ 2 (Пропорционально слоев)	105	96	89	80
Способ 3 (Одинаково нейронов)	134	109	109	109
Способ 4 (Пропорционально нейронов)	104	95	89	80
Способ 5 (Минимум отправки данных)	110	98	97	98

Было проведено несколько запусков различных схемотехнических моделей и физических стендов. Осуществлена декомпозиция нейронной сети на блоки для исполнения на каскаде устройств в рамках каждой из предложенных схем различными способами [8]. В случаях равномерного распределения слоев и нейронов по всем вычислительным устройствам, каждое получило примерно по 33% и 25% вычислительной нагрузки для каскадов из трех и четырех вычислителей соответственно. В других случаях, когда нагрузка на устройства распределялась пропор-

ционально их мощности, распределение нагрузки получилось более сложным, оно представлено в Таблице 9.

Таблица 9 – Распределение нагрузки по узлам каскада устройств в случае распределения пропорционально вычислительной мощности

Pi Zero		D-Link		Atmel		Новое у-во		Параметр
%	Слоев	%	Слоев	%	Слоев	%	Слоев	Имя у-ва
55	27	22,5	11	22,5	11	-	-	-
44,8	22	18,4	9	18,4	9	18,4	10	ATMega32
35,5	17	14,5	7	14,5	7	35,5	19	Pi Zero
22,7	11	9,3	4	9,3	4	58,7	31	Pi 4 Model B

На вход каждой модели были последовательно представлены одинаковые исходные данные, созданные в генераторе слов. Были произведены замеры показателей времени выполнения нейросетевых вычислений и времени отклика пультов голосования. Исходя из представленных данных можно приступить к Парето-оптимизации и выбрать наиболее оптимальный вариант для его последующего использования. В случае, если в сектор оптимальности по Парето попадет несколько различных решений, то все из них считаются Парето-оптимальными и допустимо использовать любой из них в качестве конечного результата на усмотрение того, кто будет реализовывать итоговый вариант вычислительной системы. На Рисунке 40 представлена точечная диаграмма, каждый из узлов – один из экспериментальных случаев по времени работы, представленных в таблицах 7 и 8. Цветовая легенда показывает, к какой из четырех архитектур каскада относится каждый случай.

На Рисунке 41 представлена та же точечная диаграмма, только цветовая легенда показывает, к какому из пяти способов декомпозиции относится каждый случай.

Таким образом, самым оптимальным вариантом в двухкритериальной оптимизации будет вариант на архитектурной схеме номер 3, с декомпозицией по 4 способу. Но у нас есть еще два параметра оптимизации, один из которых является ключевым в рассматриваемой задача, по параметру стоимости лидирует исходная архитектура вычислительной системы, поскольку для нее нет необходимо-

сти закупать дополнительное оборудование. Поэтому в случае с четырьмя критериями оптимизации, выигрывает уже вариант с исходной архитектурой, в котором НС также подверглась декомпозиции по 4 способу. Хотя на Рисунке 42 видно, что в зону улучшения по двум временным параметрам попадают еще 3 решения.

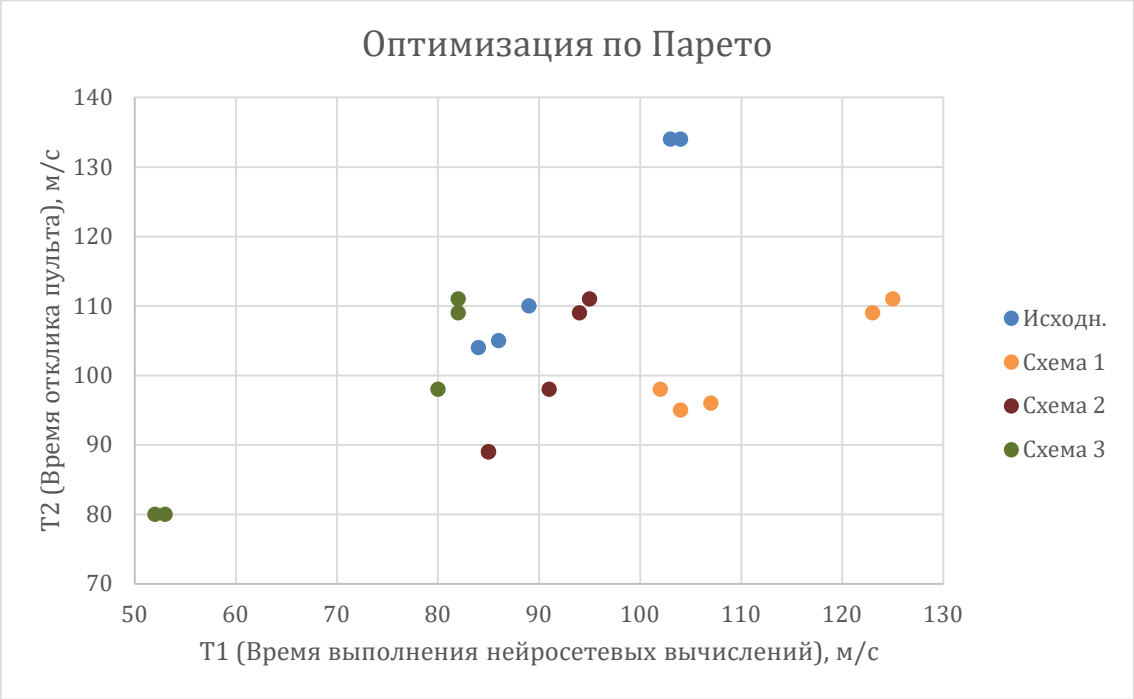


Рисунок 40 – Точечная диаграмма экспериментов (по схеме сборки каскада)

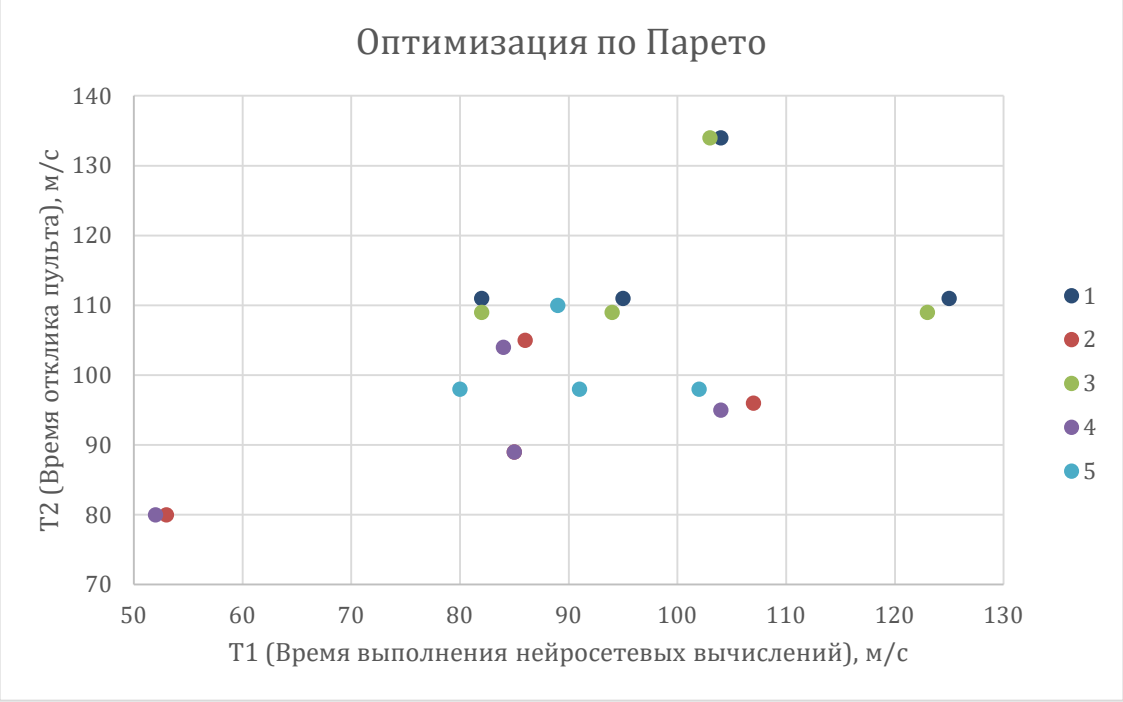


Рисунок 41 – Точечная диаграмма экспериментов (по способу декомпозиции НС)

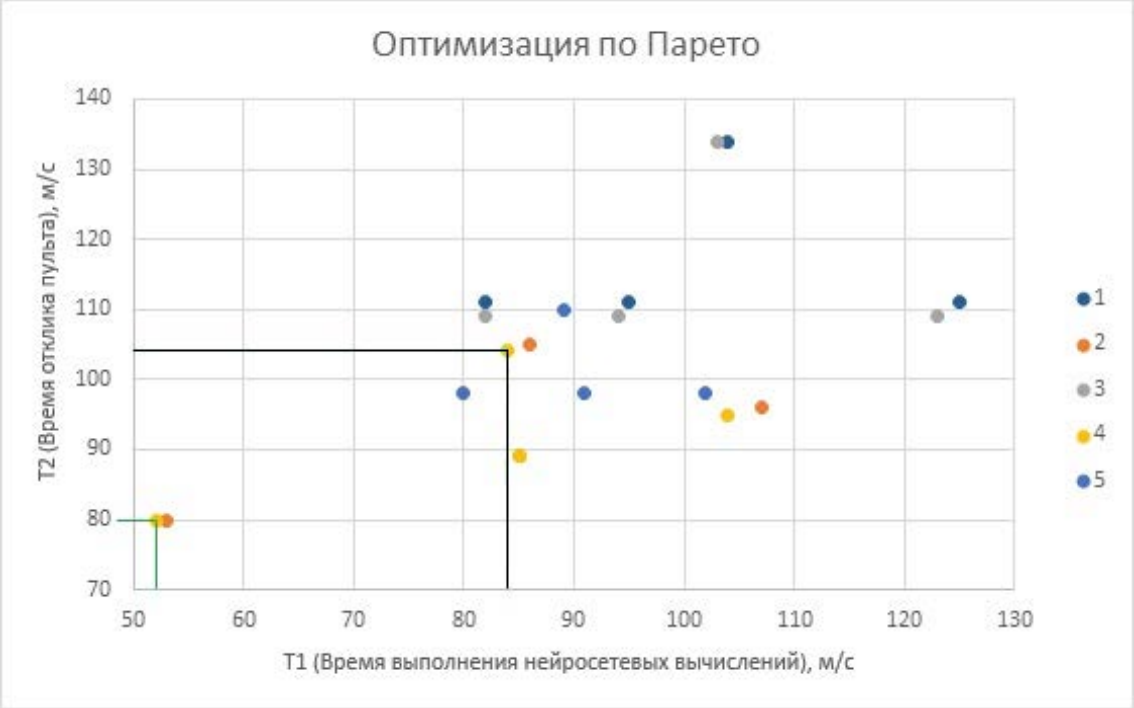


Рисунок 42 – Парето-оптимальное решение в двухкритериальной задаче (выделено зеленым цветом) и решение, Парето-оптимальное в четырехкритериальной задаче (выделено черным)

Нейронная сеть была декомпозирована по 4 способу, то есть с пропорциональным разделением нагрузки между вычислителями. Для решения четырехкритериальной задачи [77], была построена таблица с рассматриваемыми вариантами декомпозиции и построения архитектуры вычислительного каскада (Таблица 10).

Таблица 10 – Варианты решения четырехкритериальной задачи оптимизации

Схема	C (Руб.)	W (Вт.)	T^1 (мс)	T^2 (мс)
Схема 1 (Исходная)	10601	6,676	86	105
Схема 2 (Добавлено у-во 1)	12211	6,811	107	96
Схема 3 (Добавлено у-во 2)	15232	7,526	85	89
Схема 4 (Добавлено у-во 3)	21501	10,176	53	80

Парето-оптимальным решением было признано решение, сохраняющее исходную архитектуру вычислительного каскада (Схема 1), поскольку оно позволяет избежать значительных материальных затрат на закупку оборудования (параметр стоимости C) и увеличения суммарной мощности вычислительного каскада

(параметр W), сохраняя при этом допустимые скорости отклика устройств и время выполнения нейросетевых вычислений (параметры T^1 и T^2). Представим рассмотренные данные для всех решений из Таблицы 5 в виде лепестковой диаграммы (Рисунок 43).

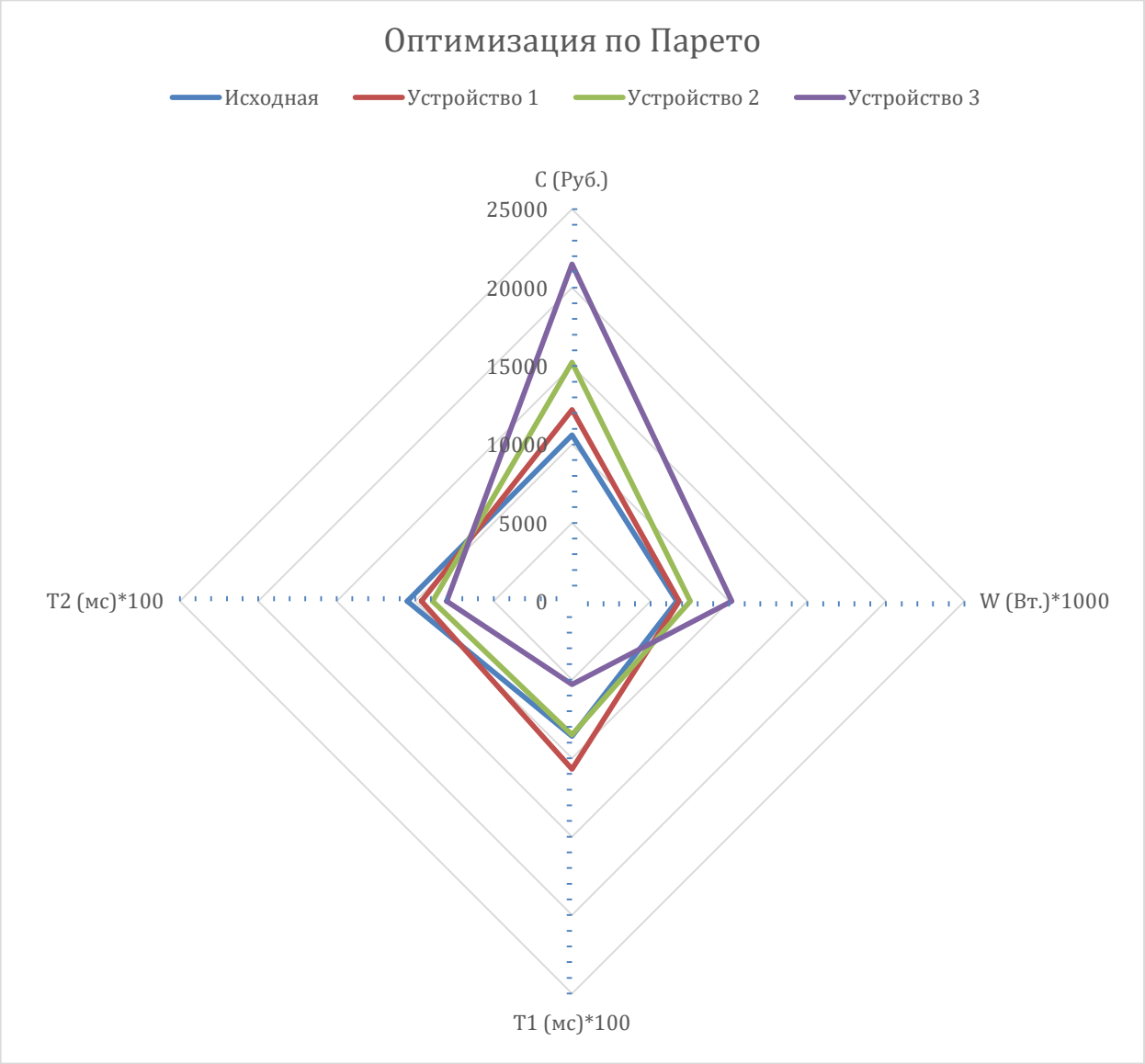


Рисунок 43 – Парето-оптимальное решение в четырехкритериальной задаче (выделено синим цветом)

На рисунке представлены декомпозиции монолитной нейронной сети способом номер 4 для всех схем каскадов физических устройств. Наиболее оптимальным вариантом в предлагаемой модели оптимизации по требуемым критериям вариант с сохранением исходной архитектуры и декомпозицией нейронной се-

ти на блоки способом номер 4, то есть с разделением на блоки пропорционально мощности с точностью до нейронов на слоях (Рисунок 43). Именно это решение является оптимальным, потому что эта точка принадлежит к исходной схеме физической архитектуры, которая является самой оптимальной по критерию стоимости (Таблица 10), а также позволяет получить наименьшую потребляемую мощность с допустимыми параметрами быстродействия нейросетевых вычислений и скорости отклика устройств пультов голосования на команды. Полученные результаты говорят о том, что в рамках предложенного метода и различных способов его реализации возможно провести оптимизацию и выбрать наиболее удачный сценарий декомпозиции нейронной сети. Полученные результаты подтверждают эффективность предложенного метода.

Подтверждено утверждение о том, что разделение монолитной нейронной сети на блочную, с последующим вычислением результатов работы блочной нейронной сети на каскаде устройств, позволяет снизить нагрузку на единичное вычислительное устройство в данном каскаде. Данное утверждение подтвердилось в результате проведения схемотехнического и физического моделирования. Моделирование подтвердило достоверность результатов исследования.

Внедрение полученного метода синтеза нейросетевых устройств происходило на базе ЗАО «Проминформ», разрабатывающего системы голосования для высших органов государственной власти: Государственная дума, Совет Федерации. Метод был использован для перераспределения вычислительных мощностей каскада устройств для выполнения вычислений нейронной сети, осуществляющей обработку дактилоскопических данных, полученных от специализированного сенсора с целью биометрической идентификации депутата на рабочем месте.

5.4. Выводы по главе

В рамках тестирования усовершенствованного метода были осуществлены несколько схемотехнических моделирований в Proteus и несколько физических моделирований запусков на устройствах ATMega32A и Raspberry Pi 4 Model B. Первым шагом осуществлялась декомпозиция монолитной нейронной сети, в со-

ответствии с входными параметрами, которые задавались для конкретной итерации испытаний. Далее осуществлялся синтез нейросетевого устройства в соответствии с тем, на какой аппаратной базе оно синтезировалось. После чего они объединялись в каскад и происходил запуск каскада, которому на вход подавались различные входные параметры, аналогичные тем, которые подавались и монолитным нейронным сетям. Проведенные запуски различных схемотехнических моделей и физических каскадов продемонстрировали работоспособность и эффективность предложенного метода синтеза нейросетевых устройств для реализации режима Fog Computing. Метод позволил также осуществить оптимизацию распределения вычислительной нагрузки по фиксированному заранее числу устройств, которые требовалось сделать каскадом реализации блочной нейронной сети. Это стало возможным благодаря возможности настраивать процесс декомпозиции в зависимости от пропорциональности вычислительным возможностям устройств в каскаде и возможности сопоставлять число операций не только по числу слоев нейронной сети, но и по числу нейронов на каждом вычислителе, которые можно было приблизительно пропорционально распределить (приблизительно, поскольку в рамках представленного метода разрывать слои не кажется предпочтительным).

В рамках исследования была поставлена задача поиска оптимального решения исследовательской задачи для ЗАО «Проминформ», проведены декомпозиции монолитной нейронной сети с разными исходными параметрами. Проведены тестовые запуски различных вариантов, полученных блочных нейронных сетей, замеры получившихся параметров и анализ полученных результатов. Осуществлена Парето оптимизация полученных вариантов и определен наиболее оптимальный вариант декомпозиции нейронной сети в рамках решения четырехкритериальной задачи оптимизации. Рассматриваемый метод позволил найти оптимальное по стоимости, энергопотреблению, времени выполнения нейросетевых вычислений и времени выполнения основных вычислений, решение для декомпозиции нейронной сети на каскад распределенных вычислителей при заданных ограничениях. Полученное решение позволило сэкономить средства компании и не допустить

увеличения нагрузки на электросеть зала заседаний, поскольку схема из каскадов вычислительный узлов предположительно будет применяться на рабочих местах депутатов, число которых может достигать нескольких сотен.

В приложениях представлены: листинг алгоритмов декомпозиции монолитной нейронной сети и запуска блочной нейронной сети, акты внедрения научных результатов в ЗАО Проминформ и ФГБОУ ВО ПНИПУ, свидетельства о государственной регистрации программ для ЭВМ.

ЗАКЛЮЧЕНИЕ

Представленная диссертационная работа посвящена решению важной научно-технической проблемы – улучшению эксплуатационно-технических показателей вычислительных систем и их элементов на основе декомпозиции искусственной нейронной сети и реализации полученных блоков в каскаде нейросетевых устройств. В диссертационной работе поставлены и решены следующие задачи исследования:

1. Разработана *математическая модель искусственной нейронной сети для синтеза нейросетевых устройств, ориентированных на туманные вычисления*. Это *позволило* учесть требуемую загрузку вычислительных узлов при распределении блоков нейронной сети между различными устройствами.

2. Разработан *метод синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления и его модификация, обеспечивающая отказоустойчивость*. Это *позволило* выбрать оптимальный вариант декомпозиции по заранее определенным параметрам, и организовать вычислительный каскад таким образом, чтобы продолжать работу даже в случае отказа части устройств в каскаде.

3. Разработан и апробирован *алгоритм декомпозиции монолитной нейронной сети на каскад блоков блочной нейронной сети, адаптированной для туманных вычислений*. Это *позволит* проводить многократную декомпозицию в глубину, например, если потребуется декомпозировать отдельный блок еще на несколько блоков.

4. Разработан и апробирован *алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах*. Это *позволило* найти оптимальную декомпозицию монолитной нейронной сети сразу по нескольким важным для вычислительной системы параметрам, за счет чего стало возможным расширение обработки данных вычислительной системой без увеличения ее стоимости.

Дальнейшие исследования целесообразны в области декомпозиции нейронных сетей с другими актуальными нейросетевыми архитектурами [23].

Практическая значимость диссертационного исследования заключается в экономии вычислительных ресурсов определенных узлов, за счет распределения нагрузки на менее загруженные части системы, а также повышения отказоустойчивости системы, за счет осуществления диагностики и реконфигурации каскада вычислительных узлов, и уменьшении затрат материальных ресурсов для реализации распределенных каскадов нейросетевых устройств, что подтверждается актом внедрения. Разработанные программные продукты прошли тестирование и получили свидетельства о государственной регистрации.

СПИСОК ЛИТЕРАТУРЫ

1. Алексеенко, Ю.В. Применение распределенных вычислений для реализации алгоритмов Deep Learning / Ю.В. Алексеенко // Современные тенденции развития науки и технологий. – 2015. – № 6–1. – С. 95–100.

2. Бахтин, В.В. Алгоритм построения графа совместной работы каскадов устройств нейросетевого распознавания, реализующих блочные нейронные сети / В.В. Бахтин, И.А. Подлесных // Сборник материалов IX Международной научной конференции, посвященной 85-летию профессора В.И. Потапова. – Омск, 2021. – С. 277–278.

3. Бахтин, В.В. Алгоритм разделения монолитной нейронной сети для реализации туманных вычислений в устройствах на программируемой логике / В.В. Бахтин // Вестник Пермского национального исследовательского политехнического университета. Электротехника. Информационные технологии, системы управления. – 2022. – № 41. – С. 123–145.

4. Бахтин, В.В. Исследование декомпозиции нейронной сети в системе схемотехнического моделирования Proteus / В.В. Бахтин, И.А. Подлесных, С.Ф. Тюрин // Вестник Пермского университета. Математика. Механика. Информатика. – 2022. – № 2 (57). – С. 73–80.

5. Бахтин, В.В. Математическая модель искусственной нейронной сети для устройств на плис и микроконтроллерах, ориентированных на туманные вычисления / В.В. Бахтин // Вестник Пермского национального исследовательского политехнического университета. Электротехника. Информационные технологии, системы управления. – 2021. – № 40. – С. 109–129.

6. Бахтин, В.В. Метод синтеза устройств нейросетевого распознавания на программируемой логике для реализации режима fog computing / В.В. Бахтин, С.Ф. Тюрин, И.А. Подлесных // Вестник Пермского национального исследовательского политехнического университета. Электротехника. Информационные технологии, системы управления. – 2022. – № 41. – С. 168–188.

7. Бахтин, В.В. Модификация алгоритма идентификации и категоризации научных терминов с использованием нейронной сети / В.В. Бахтин // Нейрокомпьютеры: разработка, применение. – 2019. – Т. 21, № 3. – С. 14–19.

8. Бахтин, В.В. Решение задачи многокритериальной оптимизации вариантов декомпозиции нейронной сети и компоновки каскада вычислительных устройств методом Парето / В.В. Бахтин, И.А. Подлесных, С.Ф. Тюрин // Вестник Пермского национального исследовательского политехнического университета. Электротехника. Информационные технологии, системы управления. – 2022. – № 43. – С. 136–156.

9. Бондарь, О.Г. Применение нейронных сетей в задаче количественного анализа состава воздушной среды / О.Г. Бондарь, Е.О. Брежнева, Р.Е. Чернышов // Известия Юго-Западного государственного университета. – 2020. – Т. 24, № 1. – С. 159–174.

10. Васильев, Н.П. Аналитическая оценка вероятности успешной адаптации к отказам модульных вычислительных систем с многоуровневой активной защитой [Текст] / Н.П. Васильев, И.Б. Шубинский // Известия высших учебных заведений. Приборостроение. – 1994. – Т. 37, № 3–4. – С. 47.

11. Виноградов. Г.П. Беспроводные сенсорные сети в защищаемых зонах / Г.П. Виноградов, А.С. Емцев, И.С. Федотов // Известия ЮФУ. Технические науки. – 2021. – № 1 (218). – С. 19–30.

12. Галимянов, Ф.А. Модель роста нейронной сети / Ф.А. Галимянов, Ф.М. Гафаров, Н.Р. Хуснутдинов // Матем. моделирование. – 2011. – Т. 23, № 3. – С. 101–108.

13. Гафаров, Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – С. 121.

14. ГОСТ 20911-89. Техническая диагностика. Термины и определения. [Текст]. – М.: Стандартинформ, 2009. – 11 с.

15. ГОСТ 27.002–2015. Надежность в технике Основные понятия. Термины и определения [Текст]. – Введ. 2017–03–01. – М.: Старнартинформ, 2016. – 23 с.

16. Гржибовский, А.М. Анализ номинальных данных (независимые наблюдения) [Электронный ресурс] / А.М. Гржибовский // Экология человека. –

2008. – № 6. – URL: https://cyberleninka.ru/article/n/analiz-nominalnyh-dannyh-nezavisimye-nablyudeniya (дата обращения: 21.01.2022).

17. Гудфеллоу, Я. Глубокое обучение [Электронный ресурс] / Я. Гудфеллоу, Б. Иошуа, А. Курвилль // Litres. – 2018. – URL: https://www.litres.ru/?GOLD=&utm_source=yandex&utm_medium=cpc&utm_campaign=web (дата обращения: 10.02.2023).

18. Ионов, С.Д. Распределенный запуск нейронных сетей на множестве вычислительных узлов [Электронный ресурс] / С.Д. Ионов // Вестник УГАТУ = Vestnik UGATU. – 2013. – № 2 (55). – URL: https://cyberleninka.ru/article/n/raspredelennyy-zapusk-neyronnyh-setey-na-mnozhestve-vychislitelnyh-uzlov (дата обращения: 07.02.2023).

19. Каменских, А.Н. Методика комбинированного резервирования асинхронных нейронных сетей / А.Н. Каменских, С.Ф. Тюрин // Нейрокомпьютеры: разработка, применение. – 2016. – № 8. – С. 36–40.

20. Мосин, С.А. Нейросетевой алгоритм для распознавания судов на аэроснимках морской поверхности и прибрежных зон / С.А. Мосин, С.Ю. Мирошниченко // Телекоммуникации. – 2021. – № 1. – С. 10–17.

21. Николенко, С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангельская. – СПб.: Издательский дом Питер, 2017.

22. Подлесных, И.А. Методы декомпозиции искусственных нейронных сетей с учетом возможности распараллеливания вычислений / И.А. Подлесных, В.В. Бахтин // Автоматизированные системы управления и информационные технологии: сборник материалов Всероссийской научно-технической конференции. – Пермь, 2022. – Т. 1. – С. 215–220.

23. Подлесных, И.А. Усовершенствование метода проектирования нейросетевых устройств для туманных вычислений [Электронный ресурс] / И.А. Подлесных, В.В. Бахтин // Инновационные технологии: теория, инструменты, практика (InnoTech-2022): тезисы XIV Международной интернет-конференции молодых ученых, аспирантов и студентов. – URL: https://innotech.pstu.ru/articles/ (дата обращения: 10.02.2023).

24. Подход к планированию загрузки процессоров мультипроцессорных систем критического назначения / Ю.В. Соколова, Е.В. Леун, П.В. Примаков, С.Ю. Самойлов // Труды МАИ. – 2022. – № 126.

25. Потапов, В.И. Фундаментальные проблемы разработки основ прикладной теории надежности искусственных нейронных сетей и нейрокомпьютерных систем / В.И. Потапов // Омский научный вестник. – 2006. – № 4(38). – С. 112–115.

26. Применение нейронных сетей в системах обеспечения информационной безопасности / А.В. Плугатарев, А.Л. Марухленко, М.А. Бугорский, А.С. Булгаков, М.А. Марченко // Безопасность информационных технологий. – 2021. – Т. 28, № 3. – С. 73–80.

27. Программа для ЭВМ № 2022611627 Российская Федерация. Программный продукт «NNSplitter» для декомпозиции монолитных НС на каскад блочных НС для синтеза нейросетевых устройств на программируемой логике: № 2022611627: заявл. 19.01.2022: опубл. 28.01.2022 / Бахтин В.В. – 1 с. – Текст: непосредственный.

28. Программа для ЭВМ № 2022615562 Российская Федерация. Программный продукт «NNImplementer» для реализации запуска и работы блочной НС на каскаде нейросетевых устройств на программируемой логике: № 2022615562: заявл. 27.03.2022: опубл. 31.03.2022 / Бахтин В.В. – 1 с. – Текст: непосредственный.

29. Тюрин, С.Ф. БМК-реализация самосинхронного генератора логических функций для нейронных сетей / С.Ф. Тюрин, А.Н. Каменских // Нейрокомпьютеры: разработка, применение. – 2018. – № 7б. – С. 26–32.

30. Тюрин, С.Ф. Обеспечение надежности технических средств путем их троирования и расчетверения / С.Ф. Тюрин // Надежность. – 2019. – Т. 19 (1). – С. 4–9. DOI: 10.21683/1729-2646-2019-19-1-4-9

31. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – М.: Издательский дом Вильямс, 2008. – 2-е изд.

32. Шубинский, И.Б. Надежные отказоустойчивые информационные системы. Методы синтеза [Текст]: методы синтеза / И.Б. Шубинский. – М: Журнал «Надежность», 2016. – 544 с.

33. Шуленин, В.П. Изучение свойств ранговых аналогов F-критерия Фишера при отклонениях от гауссовской модели дисперсионного анализа [Электронный ресурс] / В.П. Шуленин, В.В. Табольжин // Вестн. Том. гос. ун-та. Управление, вычислительная техника и информатика. – 2008. – № 1 (2). – URL: https://cyberleninka.ru/article/n/izuchenie-svoystv-rangovyh-analogov-f-kriteriya-fishera-pri-otkloneniyah-ot-gaussovskoy-modeli-dispersionnogo-analiza (дата обращения: 01.02.2022).

34. Южаков, А.А. Моделирование нейросети с использованием возможностей объектно-ориентированных языков программирования / А.А. Южаков, В.Е. Щавлев // Вестник Пермского национального исследовательского политехнического университета. Электротехника, информационные технологии, системы управления. – 2012. – № 6. – С. 248–256.

35. Ясницкий, Л.Н. Интеллектуальные системы / Л.Н. Ясницкий. – М.: Лаборатория знаний, 2016.

36. A Beginner’s Guide To Understanding Convolutional Neural Networks – Adit Deshpande – Engineering at Forward | UCLA CS ’19 [Электронный ресурс]. – URL: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/ (дата обращения: 21.10.2021).

37. A convolutional neural network fully implemented on fpga for embedded platforms / M. Bettoni [et al.] // 2017 New Generation of CAS (NGCAS). – 2017. – P. 49–52. DOI: 10.1109/NGCAS.2017.16

38. Aazam, M. Fog Computing Architecture, Evaluation, and Future Research Directions / M. Aazam, S. Zeadally, K. Harras // IEEE Communications Magazine. – 2018. – No. 56. – P. 46–52. DOI: 10.1109/MCOM.2018.1700707

39. Abadi, M. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems / M. Abadi, A. Agarwal. – 2015.

40. An Efficient Binary Convolutional Neural Network With Numerous Skip Connections for Fog Computing / L. Wu, X. Lin, Z. Chen, J. Huang, H. Liu, Y. Yang // IEEE Internet of Things Journal. – 2021. – Vol. 8, no. 14. – P. 11357–11367. DOI: 10.1109/JIOT.2021.3052105

41. Bakhtin, V. Developing an Algorithm for Identification and Categorization of Scientific Terms in Natural Language Text through the Elements of Artificial Intelligence / V. Bakhtin, E. Isaeva // 14th International Scientific-Technical Conference on Actual Problems of Electronic Instrument Engineering (APEIE) – 44894. Proceedings. – Novosibirsk, 2018. – P. 384–390.

42. Bakhtin, V.V. Algorithm for Decomposition of a Monolithic Neural Network into a Cascade of Block Neural Networks for the Fog Computing / V.V. Bakhtin // 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – 2022. – P. 238–241. DOI: 10.1109/ElConRus54750.2022.9755533

43. Bakhtin, V.V. New TSBuilder: Shifting towards Cognition / V.V. Bakhtin, E.V. Isaeva // 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2019. – P. 179–181. DOI: 10.1109/EIConRus.2019.8656917

44. Bakhtin, V.V. TSBuilder 2.0: Improving the Identification Accuracy Due to Synonymy / V.V. Bakhtin, E.V. Isaeva, A.V. Tararkov // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2020. – P. 225–228. DOI: 10.1109/EIConRus49466.2020.9039207

45. Bakhtin, V.V. TSMiner: from TSBuilder to Ecosystem / V.V. Bakhtin, E.V. Isaeva, A.V. Tararkov // 2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – 2021. – P. 221–224. DOI: 10.1109/ElConRus51938.2021.9396569

46. Bhuiya, A. Development of Wi-Fi communication module for ATmega microcontroller based mobile robot for cooperative autonomous navigation / A. Bhuiya, A. Mukherjee, R.K. Barai // 2017 IEEE Calcutta Conference (CALCON). – Kolkata, India, 2017. – P. 168–172. DOI: 10.1109/CALCON.2017.8280718

47. Cao, W. Fast Deep Neural Networks With Knowledge Guided Training and Predicted Regions of Interests for Real-Time Video Object Detection / W. Cao, J. Yuan, Z. He, Z. Zhang, Z. He // IEEE Access. – 2018. – Vol. 6. – P. 8990–8999. DOI: 10.1109/ACCESS.2018.2795798

48. Chen, J. Application of Proteus software in MCU teaching / J. Chen, S. Yang // 2011 Second International Conference on Mechanic Automation and Control Engineering. – 2011. – C. 6359–6362. DOI: 10.1109/MACE.2011.5988496

49. Deep learning in the fog / A. Sobecki, J. Szymański, D. Gil, H. Mora // International Journal of Distributed Sensor Networks. – 2019. – Vol. 15(8). DOI: 10.1177/1550147719867072

50. DeRonne, K.W. Pareto Optimal Pairwise Sequence Alignment / K.W. DeRonne, G. Karypis // IEEE/ACM Transactions on Computational Biology and Bioinformatics. – 2013. – Vol. 10, no. 2. – P. 481–493. DOI: 10.1109/TCBB.2013.2

51. Design and realization of a single-phase inverter with numerical control based on an Atmega32 / M. El Ouariachi [et al.] // 2017 14th International Multi-Conference on Systems, Signals & Devices (SSD). – 2017. – P. 239–244. DOI: 10.1109/SSD.2017.8166982

52. Design of NNEF-PyTorch Neural Network Model Converter / K.H. Lee, J. Park, S.-T. Kim, J.Y. Kwak, C.S. Cho // 2021 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, Republic of. – 2021. – P. 1710–1712. DOI: 10.1109/ICTC52510.2021.9621003

53. DL4J, Torch7, Theano and Caffe [Электронный ресурс]. – URL: https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch (дата обращения: 23.01.2021).

54. Extending the ONNX Runtime Framework for the Processing-in-Memory Execution / S.Y. Kim, J. Lee, C.H. Kim, W.J. Lee, S.W. Kim // 2022 International Conference on Electronics, Information, and Communication (ICEIC). – Jeju, Korea, Republic of, 2022. – P. 1–4. DOI: 10.1109/ICEIC54506.2022.9748444

55. Faggin, F. Neural network hardware / F. Faggin // [Proceedings 1992] IJCNN International Joint Conference on Neural Networks. – 1992. – Vol. 1. – P. 153. DOI: 10.1109/IJCNN.1992.287238

56. Federated Learning: Strategies for Improving Communication Efficiency / J. Konečný [et al.] // ArXiv161005492 Cs. – 2017.

57. Federated Machine Learning: Concept and Applications / Q. Yang [et al.] // ACM Trans. Intell. Syst. Technol. – 2019. – Vol. 10, № 2. – P. 12:1–12:19.

58. Fog Computing and Convolutional Neural Network Enabled Prognosis for Machining Process Optimization / Y.C. Liang, W.D. Li, X. Lu, S. Wang // Data Driven Smart Manufacturing Technologies and Applications. – 2021.

59. Fog Computing: A Comprehensive Architectural Survey / P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, A. Leon-Garcia // IEEE Access. – 2020. – Vol. 8. – P. 69105–69133. DOI: 10.1109/ACCESS.2020.2983253

60. Fryza, T. Basic C code implementations for AVR microcontrollers / T. Fryza // 2007 14th International Workshop on Systems, Signals and Image Processing and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. – 2007. – P. 434–437. DOI: 10.1109/IWSSIP.2007.4381134

61. Fryza T. Instruction-level programming approach for very long instruction word digital signal processors / T. Fryza, R. Mego // 2017 24th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Batumi, Georgia – 2017 – P. 518-521. DOI: 10.1109/ICECS.2017.8292060

62. Gayathri Devi, S. An abstract to calculate big o factors of time and space complexity of machine code / S. Gayathri Devi, K. Selvam, S.P. Rajagopalan // International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011). – 2011. – P. 844–847. DOI: 10.1049/cp.2011.0483

63. Hayman, S. The McCulloch-Pitts model / S. Hayman // International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339). – 1999. – Vol. 6. – P. 4438–4439. DOI: 10.1109/IJCNN.1999.830886

64. HOME – WEMOS documentation [Электронный ресурс]. – URL: https://www.wemos.cc/ (дата обращения: 23.09.2022).

65. Hong, H.J. From cloud computing to fog computing: unleash the power of edge and end devices / H.J. Hong // 2017 IEEE international conference on cloud computing technology and science (CloudCom). – 2017. – P. 331–334. DOI: 10.1109/CloudCom.2017.53

66. Hubert, A.B. Random activation weight neural net (RAWN) for fast non-iterative training / A.B. Hubert, Te Braake, Gerrit Van Straten // Engineering Applications of Artificial Intelligence. – 1995. – Vol. 8, iss. 1. – P. 71–80. DOI: 10.1016/0952-1976(94)00056-S

67. Huynh, T.V. Deep neural network accelerator based on FPGA / T.V. Huynh // 2017 4th NAFOSTED Conference on Information and Computer Science. – 2017. – P. 254–257. DOI: 10.1109/NAFOSTED.2017.8108073

68. Isaeva, E. Collecting the Database for the Neural Network Deep Learning Implementation / E. Isaeva, V. Bakhtin, A. Tararkov // Digital Science. DSIC18 2018. Advances in Intelligent Systems and Computing. – 2019. – Vol. 850. – P. 12–18. DOI: 10.1007/978-3-030-02351-5_2

69. Isaeva, E. Formal Cross-Domain Ontologization of Human Knowledge / E. Isaeva, V. Bakhtin, A. Tararkov // Information Technology and Systems. ICITS 2020. Advances in Intelligent Systems and Computing. – 2020. – Vol. 1137. – P. 94–103. DOI: 10.1007/978-3-030-40690-5_10

70. Isaeva, E. Ontologization and Term System Modelling by means of AI Methods/ E. Isaeva, A. Tararkov, **V. Bakhtin** // Specialized Knowledge Mediation. – 2022. – P. 139–149. – Springer, Cham, 2022. DOI: 10.1007/978-3-030-95104-7_7

71. Keras [Электронный ресурс]. – URL: https://github.com/keras-team/keras (дата обращения: 28.11.2021).

72. Kohonen, T. The self-organizing map / T. Kohonen // Proceedings of the IEEE. – 1990. – Vol. 78, no. 9. – P. 1464–1480. DOI: 10.1109/5.58325

73. LeCun, Y. Deep learning: 7553 / Y. LeCun, Y. Bengio, G. Hinton // Nature. Nature Publishing Group. – 2015. – Vol. 521, № 7553. – P. 436–444.

74. Li, L. Deep learning for smart industry: Efficient manufacture inspection system with fog computing / L. Li, K. Ota, M. Dong // IEEE Transactions on Industrial Informatics. – 2018. – Т. 14, № 10. – С. 4665–4673.

75. Liang, Y.C. Adaptive Diagnostics on Machining Processes Enabled by Transfer Learning / Y.C. Liang, W.D. Li, S. Wang, X. Lu // Data Driven Smart Manufacturing Technologies and Applications. Springer Series in Advanced Manufacturing. – Springer, Cham, 2021. DOI:10.1007/978-3-030-66849-5_4

76. Madhuri, Y. Vision-based sign language translation device / Y. Madhuri, G. Anitha, M. Anburajan // 2013 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India. – 2013. – P. 565–568. DOI: 10.1109/ICICES.2013.6508395

77. Miettinen, K. Nonlinear multiobjective optimization / K. Miettinen // Springer Science & Business Media, 2012. – Vol. 12.

78. Mobilenets: Efficient convolutional neural networks for mobile vision applications / A.G. Howard [et al.] // ArXiv Preprint ArXiv170404861. – 2017.

79. Mostafa, G. Design of a single chip Digital Weighing Machine using ATmega32 microcontroller architecture / G. Mostafa // 2015 International Conference on Advances in Electrical Engineering (ICAEE). – 2015. – C. 93–96. DOI: 10.1109/ICAEE.2015.7506804

80. ONNC: A Compilation Framework Connecting ONNX to Proprietary Deep Learning Accelerators / W.-F. Lin [et al.] // 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS). – Hsinchu, Taiwan, 2019. – P. 214–218. DOI: 10.1109/AICAS.2019.8771510

81. PID-Neural controller based on AVR Atmega128 / X.K. Pham [et al.] // 2008 10th International Conference on Control, Automation, Robotics and Vision. – 2008. – P. 1573–1576. DOI: 10.1109/ICARCV.2008.4795759

82. Podlesnykh, I.A. Advanced Fog Neural Network Design Method / I.A. Podlesnykh, V.V. Bakhtin // 2023 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – 2023. – P. 244–247. DOI: 10.1109/ElConRus54750. 2022.9755677

83. Podlesnykh, I.A. Mathematical Model of a Recurrent Neural Network for Programmable Devices Focused on Fog Computing / I.A. Podlesnykh, V.V. Bakhtin // 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). – 2022. – P. 395–397. DOI: 10.1109/ElConRus54750.2022.9755677

84. Priyabhashana, H.M.B. Data Analytics with Deep Neural Networks in Fog Computing Using TensorFlow and Google Cloud Platform / H.M.B. Priyabhashana, K.P.N. Jayasena // 2019 14th Conference on Industrial and Information Systems (ICIIS). – 2019. – P. 34–39. DOI: 10.1109/ICIIS47346.2019.9063284

85. Priyadarshini, R. Deepfog: fog computing-based deep neural architecture for prediction of stress types, diabetes and hypertension attacks / R. Priyadarshini, R.K. Barik, H. Dubey // Computation. – 2018. – Vol. 6, № 4. – P. 62.

86. Redmon, J. YOLOv3: An Incremental Improvement / J. Redmon, A. Farhadi // ArXiv, abs/1804.02767. – 2018.

87. 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. DOI: 10.1037/h0042519

88. Shervin, Minaee. FingerNet: Pushing The Limits of Fingerprint Recognition Using Convolutional Neural Network / Shervin Minaee, Elham Azimi, Amirali Abdolrashidi // ArXiv abs/1907.12956. – 2019.

89. Sleesongsom, S. Multiobjective Optimization with Even Pareto Filter / S. Sleesongsom // 2008 Fourth International Conference on Natural Computation. – Jinan, China, 2008. – P. 92–96. DOI: 10.1109/ICNC.2008.766

90. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size / F.N. Iandola [et al.] // ArXiv Preprint ArXiv160207360. – 2016.

91. Su, B. Application of Proteus virtual system modelling (VSM) in teaching of microcontroller / B. Su, L. Wang // 2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT). – 2010. – Vol. 2. – P. 375–378. DOI: 10.1109/EDT.2010.5496343

92. Surbiryala, C. Rong Cloud Computing: History and Overview / C. Surbiryala // 2019 IEEE Cloud Summit. – 2019. – P. 1–7. DOI: 10.1109/CloudSummit 47114.2019.00007

93. Tang, Z. Recurrent neural network training with dark knowledge transfer / Z. Tang, D. Wang, Z. Zhang // 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2016. – P. 5900–5904. DOI: 10.1109/ICASSP.2016.7472809

94. Teach, learn, and make with the Raspberry Pi Foundation [Электронный ресурс]. – URL: https://www.raspberrypi.org/ (дата обращения: 23.09.2022).

95. Teerapittayanon, S. Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices / S. Teerapittayanon, B. McDanel, H.T. Kung // 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). – Atlanta, GA, USA, 2017. – P. 328–339. DOI: 10.1109/ICDCS.2017.226

96. TensorFlow Lite | ML для мобильных и пограничных устройств [Электронный ресурс] // TensorFlow. – URL: https://www.tensorflow.org/lite?hl=ru (дата обращения: 28.11.2021).

97. The construction of single-chip microcomputer virtual experiment platform based on proteus / W. Xinhuan [et al.] // 2010 5th International Conference on Computer Science & Education. – 2010. – C. 609–611. DOI: 10.1109/ICCSE.2010.5593538

98. Upton, E. Raspberry Pi user guide / E. Upton, G. Halfacree // John Wiley & Sons. – 2014.

99. Voice controlled home automation system using Natural Language Processing (NLP) and Internet of Things (IoT) / P.J. Rani, J. Bakthakumar, B.P. Kumaar, U.P. Kumaar, S. Kumar // 2017 Third International Conference on Science Technology Engineering & Management (ICONSTEM). – Chennai, India, 2017. – P. 368–373. DOI: 10.1109/ICONSTEM.2017.8261311

100. Wadhwa, H. Fog computing with the integration of internet of things: Architecture, applications and future directions / H. Wadhwa, R. Aron // 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). – 2018. – P. 987–994. DOI: 10.1109/BDCloud.2018.00144

101. Wang, L. An Automatic Conversion Tool for Caffe Neural Network Configuration oriented to OpenCL-based FPGA Platforms / L. Wang, Y. Zhao, X. Li // 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). – Chengdu, China, 2019. – P. 195–198. DOI: 10.1109/ITNEC.2019.8729070

102. Yasnitsky, L.N. Technique of design of integrated economic and mathematical model of mass appraisal of real estate property by the example of Yekaterinburg housing market / L.N. Yasnitsky, V.L. Yasnitsky // Journal of Applied Economic Sciences. – 2016. – Vol. 11, no 8. – P. 1519–1530.

103. Yong, Fan. The design for data collection terminal based on 32-bit Atmel / Fan Yong // 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot. – 2011. – P. 7809–7811. DOI: 10.1109/MACE.2011.5988862

104. Zupan, J. Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them / J. Zupan // Acta Chimica Slovenica. – 1994. – Vol. 41, no. 3. – P. 327–352.

ПРИЛОЖЕНИЕ А

Листинг кода программного продукта «NNSplitter»

```
package network;
import static java.lang.Math.exp;

/**
 *
 * @author bakhtin
 */
public class Functions {

   public static double Ntwo(double weight1, double weight2) //метод выбирает, какой из типов 2 терминов присвоить составному термину
   {
      double resultType = 0;

      if ((1 / (exp(-(weight1 + weight2)) + 1) > ((double) 1 / 2)) && (weight1 >= weight2)) {
         resultType = weight1;
      } else {
         resultType = weight2;
      }
      return resultType;
   }

   public static double Nthree(double weight1, double weight2, double weight3) //метод выбирает, какой из типов 3 терминов присвоить составному термину
   {
```

```
        double resultType = 0;

        if ((weight3 >= weight2) && (weight3 >= weight2)) {
            resultType = weight3;
        } else {
            if ((weight2 >= weight1) && (weight2 >= weight3)) {
                resultType = weight2;
            } else {
                resultType = weight1;
            }
        }
        return resultType;
    }

    static float computing(NeuronFunctions f, float c, float[] weightsList, int[] sinapses,
float[] lastVector) {
        float result = 0;

        float[] inputVector = new float[weightsList.length];
        for (int i = 0; i < weightsList.length; i++) {
            inputVector[i] = weightsList[i]*lastVector[sinapses[i]];
            System.out.print(inputVector[i] + " ");
        }
        System.out.println();

        result = calculate(f, c, inputVector);

        return result;
    }
```

```
    private static float calculate(NeuronFunctions f, float c, float[] inputVector) {
        if (f == NeuronFunctions.Equals) {
            return inputVector[0];
        }
        if (f == NeuronFunctions.Ntwo) {
            return (float) Functions.Ntwo(inputVector[0], inputVector[1]);
        }
        if (f == NeuronFunctions.Nthree) {
            return (float) Functions.Nthree(inputVector[0], inputVector[1], inputVector[2]);
        }
        return 0;
    }

}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package network;

import java.util.ArrayList;

/**
 *
 * @author bakhtin
 */
public class Layer {
    public int Number; //номер слоя
    public int NeuronsCount; //число нейронов на слое
```

```
public ArrayList<Neuron> NeuronsList; //нейроны на слое

public Layer(int Number, int NeuronsCount, ArrayList<Neuron> NeuronsList) {
    this.Number = Number;
    this.NeuronsCount = NeuronsCount;
    this.NeuronsList = NeuronsList;
}

public Layer() {
    this.Number = 0;
    this.NeuronsCount = 0;
    this.NeuronsList = new ArrayList();
}

public void setNumber(int Number) {
    this.Number = Number;
}

public void setNeuronsCount(int NeuronsCount) {
    this.NeuronsCount = NeuronsCount;
}

public void addNeuron(Neuron neuron) {
    this.NeuronsList.add(neuron);
}

public void print() {
    System.out.print("[");
    System.out.print(Number + "," + NeuronsCount + ",");
    for (Neuron neuron: NeuronsList) {
```

```
            neuron.print();
        }
        System.out.println("]");
    }

    public float[] computing(float[] lastVector) {
        float[] resultVector = new float[NeuronsCount];
        for (int i = 0; i < NeuronsCount; i++) {
            resultVector[i] = NeuronsList.get(i).computing(lastVector);
        }
        return resultVector;
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package network;

import java.util.ArrayList;

/**
 *
 * @author bakhtin
 */
public class NNetwork {
    public int K; //число слоев нейронов
    public ArrayList<Layer> LayersList; //список слоев
```

```
public NNetwork() {
    this.K = 0;
    this.LayersList = new ArrayList();
}

public NNetwork(int K, ArrayList<Layer> LayersList) {
    this.K = K;
    this.LayersList = LayersList;
}

public void setK(int K) {
    this.K = K;
}

public void addLayer(Layer layer) {
    this.LayersList.add(layer);
}

public void print() {
    System.out.println(K);
    for (Layer layer: LayersList) {
        layer.print();
    }
}

public float[] computing(float[] input) {
    float[] resultVector = new float[0];
    float[] lastVector = input;
    float[] actualVector;
    for (Layer layer: LayersList) {
```

```
            actualVector = layer.computing(lastVector);
            lastVector = actualVector;
        }
        resultVector = lastVector;
        return resultVector;
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package network;

/**
 *
 * @author bakhtin
 */
public class Neuron {
    public float result; // выход аксона нейрона

    public int number; //номер нейрона на слое
    public NeuronFunctions f; //функция активации нейрона
    public float c; //константа для функции активации
    public float[] weightsList; // веса синапсов
    public int[] sinapses; //номера нейронов предыдущего слоя, с которыми связан данный нейрон

    public Neuron(NeuronFunctions f, float[] weightsList, int[] sinapses) {
        this.f = f;
```

```
    this.weightsList = weightsList;
    this.sinapses = sinapses;
}

public Neuron() {
    this.number = 0;
    this.f = NeuronFunctions.None;
    this.c = 0;
    this.weightsList = new float[0];
    this.sinapses = new int[0];
}

public void setNumber(int Number) {
    this.number = Number;
}

public void setNeuronFunction(int NeuronsCount) {
    this.f = NeuronFunctions.getById(NeuronsCount);
}

public void setConst(float c) {
    this.c = c;
}

public void setWeightsList(float[] weightsList) {
    this.weightsList = weightsList;
}

public void setSinapses(int[] sinapses) {
    this.sinapses = sinapses;
```

```
    }

    public void print() {
        System.out.print("{");
        System.out.print(number + ";" + f + ";" + c + ";");
        for (float weight: weightsList) {
            System.out.print(weight + " ");
        }
        System.out.print(";");
        for (int sinapse: sinapses) {
            System.out.print(sinapse + " ");
        }
        System.out.print("}");
    }

    public float computing(float[] lastVector) {
        return Functions.computing(f, c, weightsList, sinapses, lastVector);
    }
}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package network;

import java.util.ArrayList;

/**
 *
```

```
 * @author bakhtin
 */
public class Separator {

    public Separator() {
    }

    public ArrayList<NNetwork> splitOnEqualNumberOfLayers(NNetwork network, int
D) {
        ArrayList<NNetwork> blockNetworks = new ArrayList();
        int numberOfLayers = network.K / D;

        for(int i = 0; i< D-1; i++) {
            ArrayList<Layer> layers = new ArrayList();
            for (int j = i*numberOfLayers; j < (i+1)*numberOfLayers; j++) {
                layers.add(network.LayersList.get(j));
            }
            blockNetworks.add(new NNetwork(numberOfLayers, layers));
        }

        ArrayList<Layer> layers = new ArrayList();
        for (int j = (D-1)*numberOfLayers; j < network.K; j++) {
            layers.add(network.LayersList.get(j));
        }
        blockNetworks.add(new NNetwork(network.K - (D-1)*numberOfLayers, layers));

        return blockNetworks;
    }
}
package network;
```

```
/**
 *
 * @author bakhtin
 */
public enum NeuronFunctions {
   None(999),
   Sum (0),
   Max (1),
   Sigmoida (2),
   Linear (3),
   Threshold (4),
   Or(5),
   And(6),
   Tanh (7),
   ReLu (8),
   MaxCounter(9),
   Ntwo(10),
   Nthree(11),
   Equals(12);

   private int number;

   NeuronFunctions(int Number) {
      this.number = Number;
   }

   public int getNumber() {
      return number;
   }
```

```
    public static NeuronFunctions getById(int Number) {
        for (NeuronFunctions nf: NeuronFunctions.values()) {
            if (nf.getNumber() == Number) {
                return nf;
            }
        }
        return NeuronFunctions.None;
    }

    @Override
    public String toString() {
        return "NeuronFunctions{" +
                "title='" + number + '\'' +
                '}';
    }
}
package com.pstu.nnetworkseparator;

import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.net.URL;
import java.util.ArrayList;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import network.NNetwork;
import network.Separator;
```

```
public class FXMLController implements Initializable {

    @FXML
    private Label label;

    @FXML
    private void handleButtonAction(ActionEvent event) {
        //System.out.println("Start");
        label.setText("NN separated");
        NNetwork network = MainApp.ReadNNFromFile();

        MainApp.SeparateNN(network);

        //float[] nums = {0.75f, 0.8f, -0.77f};
        //float[] result = network.computing(nums);
        //System.out.print("Result = ");
        //for (float rez: result) {
        //    System.out.print(rez + " ");
        //}
        //System.out.println();
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
    }

    private void SaveNNToFile(NNetwork rez, int i) {
        try {
          FileOutputStream fileOut = new FileOutputStream(i+".ann");
```

```
        fileOut.write(rez.K);
          System.out.println("The Object "+ i +"  was succesfully written to a file");

          } catch (Exception ex) {
           ex.printStackTrace();
          }
    }
}
package com.pstu.nnetworkseparator;

import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
import network.Layer;
import network.NNetwork;
import network.Neuron;
import network.Separator;
import sun.misc.IOUtils;
```

```
import static javafx.application.Application.launch;
import network.Functions;


public class MainApp extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("/fxml/Scene.fxml"));

        Scene scene = new Scene(root);
        scene.getStylesheets().add("/styles/Styles.css");

        stage.setTitle("JavaFX and Maven");
        stage.setScene(scene);
        stage.show();
    }

    /**
     * The main() method is ignored in correctly deployed JavaFX application.
     * main() serves only as fallback in case the application can not be
     * launched through deployment artifacts, e.g., in IDEs with limited FX
     * support. NetBeans ignores main().
     *
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println(Functions.Ntwo(0.75, 0.8));
        System.out.println(Functions.Ntwo(0.8, -0.77));
        System.out.println(Functions.Ntwo(0.75, -0.77));
```

```
        System.out.println(Functions.Nthree(Functions.Ntwo(0.75, 0.8), Functions.Ntwo(0.8, -0.77), Functions.Ntwo(0.75, -0.77)));
        launch(args);
    }

    public static NNetwork ReadNNFromFile() {
        NNetwork network = new NNetwork();
        try {
            InputStream is = MainApp.class.getResourceAsStream("/TSNetwork.ann");
            BufferedReader reader = new BufferedReader(new InputStreamReader(is));

            int K = 0;
            if (reader.ready()) {
                String countLayers = reader.readLine();
                K = Integer.parseInt(countLayers);
            }

            network.setK(K);

            for (int i = 0; i< K; i++) {
              if (reader.ready()) {
                String LayerString = reader.readLine();
                System.out.println(LayerString);

                Layer layer = new Layer();
                String[] parts = LayerString.split(",");

                layer.setNumber(Integer.parseInt(parts[0]));
                int neuronsCount = Integer.parseInt(parts[1]);
                layer.setNeuronsCount(neuronsCount);
```

```
    network.addLayer(layer);

    for (int j = 2; j < 2+neuronsCount; j++) {
        Neuron neuron = new Neuron();
        String[] neuronParts = parts[j].split(";");
        neuron.setNumber(Integer.parseInt(neuronParts[0]));
        neuron.setNeuronFunction(Integer.parseInt(neuronParts[1]));
        neuron.setConst(Float.parseFloat(neuronParts[2]));

        String[] weightsParts = neuronParts[3].split(" ");
        float[] weights = new float[weightsParts.length];
        for (int k = 0; k < weightsParts.length; k++) {
            weights[k] = Float.parseFloat(weightsParts[k]);
        }
        neuron.setWeightsList(weights);

        String[] sinapsesParts = neuronParts[4].split(" ");
        int[] sinapses = new int[sinapsesParts.length];
        for (int k = 0; k < sinapsesParts.length; k++) {
            sinapses[k] = Integer.parseInt(sinapsesParts[k]);
        }
        neuron.setSinapses(sinapses);

        layer.addNeuron(neuron);
    }
  }
}

network.print();
```

```
        } catch (IOException ex) {
            Logger.getLogger(MainApp.class.getName()).log(Level.SEVERE, null, ex);
        }
        return network;
    }

    public static void SeparateNN() {
        int K = 10;
        int D = 3;
        System.out.println("K = "+K + " D = " + D);

        ArrayList<Layer> layers = new ArrayList();
        for (int i=0; i < K; i++) {
            layers.add(new Layer(i, 5, null));
        }

        NNetwork network = new NNetwork(K, layers);

        Separator separator = new Separator();
        ArrayList<NNetwork> blockNetworks = separa-
tor.splitOnEqualNumberOfLayers(network, D);

        for (NNetwork bnetwork : blockNetworks)
        {
            System.out.println(bnetwork.K + " " + bnetwork.LayersList.size());
        }
    }

    public static void SeparateNN(NNetwork network) {
```

```
        int D = 2;
        System.out.println("K = "+network.K + " D = " + D);

        Separator separator = new Separator();
        ArrayList<NNetwork> blockNetworks = separator.splitOnEqualNumberOfLayers(network, D);

        int i = 0;
        for (NNetwork bnetwork : blockNetworks)
        {
            System.out.println(bnetwork.K + " " + bnetwork.LayersList.size());
            SaveNNToFile(bnetwork, i);
            i++;
        }
    }

    private static void SaveNNToFile(NNetwork rez, int i) {
        try {
          PrintWriter out = new PrintWriter(i+".ann");
        out.println(rez.K);
        for (Layer layer : rez.LayersList) {
            out.print(layer.Number+","+layer.NeuronsCount+",");
            int j = 0;
            for (Neuron neuron : layer.NeuronsList) {
                out.print(neuron.number+";"+neuron.f.getNumber()+";"+neuron.c+";");

                int k = 0;
                for (float weight : neuron.weightsList) {
                    out.print(weight);
                    if (k < neuron.weightsList.length - 1)
```

```
                out.print(" ");
            k++;
            }

            out.print(";");

            int f = 0;
            for (int sinapse : neuron.sinapses) {
               out.print(sinapse);
               if (f < neuron.sinapses.length - 1)
                  out.print(" ");
               f++;
            }

            if (j < layer.NeuronsList.size() - 1)
               out.print(",");
            j++;
          }
          out.println();
       }
         System.out.println("The Object "+ i +"  was succesfully written to a file");
       out.close();
         } catch (Exception ex) {
          ex.printStackTrace();
         }
    }

}
```

ПРИЛОЖЕНИЕ Б

Свидетельство о государственной регистрации программы для ЭВМ «Программный продукт «NNSplitter»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022611627

Программный продукт «NNSplitter» для декомпозиции монолитных НС на каскад блочных НС для синтеза нейросетевых устройств на программируемой логике

Правообладатель: ***Бахтин Вадим Вячеславович (RU)***

Автор(ы): ***Бахтин Вадим Вячеславович (RU)***

Заявка № **2022610667**
Дата поступления **19 января 2022 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***28 января 2022 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 0x75699C0D08AEB78543BD83DF819A6CD1
Владелец **Ивлиев Григорий Петрович**
Действителен с 24.12.2021 по 24.12.2022

Г.П. Ивлиев

ПРИЛОЖЕНИЕ В

Свидетельство о государственной регистрации программы для ЭВМ «Программный продукт «NNImplementer»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2022615562

Программный продукт «NNImplementer» для реализации запуска и работы блочной НС на каскаде нейросетевых устройств на программируемой логике

Правообладатель: ***Бахтин Вадим Вячеславович (RU)***

Автор(ы): ***Бахтин Вадим Вячеславович (RU)***

Заявка № **2022615009**
Дата поступления **27 марта 2022 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***31 марта 2022 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 68b80077e14e40f0a94edbd24145d5c7
Владелец Зубов Юрий Сергеевич
Действителен с 2.0.2022 по 26.05.2023

Ю.С. Зубов

ПРИЛОЖЕНИЕ Г

Акт внедрения результатов в АПК зала заседаний ЗАО Проминформ

Закрытое акционерное общество
ПРОМИНФОРМ

ИНН: 5905000214, КПП: 590201001
ОГРН: 1025901212369
Россия, 614000, Пермский край, г. Пермь,
ул. Газеты Звезда, 24А
тел/факс: (342) 212-35-94, 212-35-08
e-mail: box@prominform.com
www.prominform.com

Исх. № 201 от 19.05.2022

АКТ

о внедрении результатов кандидатской диссертационной работы
«Метод синтеза нейросетевых устройств для реализации режима fog computing»

Комиссия в составе:
Председатель комиссии: Попов Вадим Вительевич, руководитель отдела разработки ЗАО «Проминформ».
Член комиссии: Гайнутдинов Эльдар Фаридович, инженер-программист отдела разработки ЗАО «Проминформ».

Составила акт о том, что результаты диссертационной работы «Метод синтеза нейросетевых устройств для реализации режима fog computing» Бахтина Вадима Вячеславовича использованы при проектировании прототипа системы биометрического распознавания АПК ЗАЛА ЗАСЕДАНИЙ ИЖВН.425XXX.XXX.

В ходе выполнения работ разработан процесс распознавания биометрических данных (отпечатков пальцев) с использованием каскада вычислительных устройств, реализующих блочную нейронную сеть, входящих в состав АПК зала заседаний.

При создании модификации АППАРАТНО-ПРОГРАММНОГО КОМПЛЕКСА ЗАЛА ЗАСЕДАНИЙ ИЖВН.425XXX.XXX были использованы и внедрены следующие результаты диссертационной работы:

– математическая модель искусственной нейронной сети для синтеза нейросетевых устройств, ориентированных на туманные вычисления, позволившая сбалансировать размеры декомпозированных блоков нейронной сети в зависимости от характеристик физических устройств, входящих в вычислительный каскад;

– алгоритм декомпозиции монолитной нейронной сети на каскад блоков блочной нейронной сети, адаптированной для туманных вычислений, позволивший осуществить декомпозицию нейронной сети для анализа биометрических данных;

– метод синтеза устройств реализации искусственных нейронных сетей, ориентированных на туманные вычисления, позволивший реализовать каскад нейросетевых устройств для работы с биометрическими данными, при этом снизив объем обработки информации на тех вычислительных узлах, которые требовалось разгрузить, и обеспечив функциональное резервирование;

– алгоритм выбора оптимального варианта декомпозиции нейронной сети для реализации на распределенных вычислительных устройствах, позволивший найти оптимальную декомпозицию монолитной нейронной сети сразу по нескольким важным для вычислительной системы параметрам.

Применение результатов диссертационной работы обеспечивает:

– сокращение затрат на создание прототипа системы биометрической идентификации на 27%;

– сокращение энергопотребления прототипа системы биометрической идентификации на 12,7%

– уменьшение объема нейросетевой обработки информации на контроллере пульта на 45% по сравнению с вычислением всей нейронной сети на одном физическом устройстве, с сохранением допустимых времени выполнения нейросетевых вычислений и среднего времени отклика устройтсв на входные сигналы.

Предложенные Бахтиным В.В. в его диссертационной работе решения будут использованы при проектировании новой модели АПК ЗАЛА ЗАСЕДАНИЙ ИЖВН.425XXX.XXX.

Генеральный директор _____ Д.А. Бурди

М.П.

Председатель комиссии _____ В.В. Попов

Член комиссии _____ Э.Ф. Гайнутдинов

ПРИЛОЖЕНИЕ Д

Акт внедрения в учебный процесс кафедры АТ ПНИПУ

УТВЕРЖДАЮ
Проректор по образовательной деятельности
Пермского национального исследовательского политехнического университета
доктор технических наук, доцент
_____ / А.Б. Петроченков /
«__» апреля 2023 г.

АКТ
о внедрении научных результатов в учебный процесс,
полученных Бахтиным Вадимом Вячеславовичем
при выполнении диссертационной работы
на соискание ученой степени кандидата технических наук
«Метод синтеза нейросетевых устройств для реализации режима fog computing»

Комиссия в составе:

Председатель: Фрейман Владимир Исаакович, доктор технических наук, доцент, профессор, заместитель заведующего кафедрой «Автоматика и телемеханика» по учебной и методической работе

Члены комиссии: Хижняков Юрий Николаевич, доктор технических наук, доцент, профессор кафедры «Автоматика и телемеханика».
Гончаровский Олег Владленович, кандидат технических наук, доцент, доцент кафедры «Автоматика и телемеханика».

составила настоящий акт о том, что результаты диссертационной работы «Метод синтеза нейросетевых устройств для реализации режима fog computing» соискателя Бахтина В.В. используются для проведения лекционных и практических занятий, лабораторных работ в рамках программы бакалавриата по направлениям подготовки 11.03.02 «Инфокоммуникационные технологии и системы связи» и 27.03.04 «Управление в технических системах».

Предложенные модель, метод и алгоритмы построения и функционирования распределенных нейросетевых систем для режима fog computing нашли применение в дисциплинах «Вычислительная техника и информационные технологии», «Проектирование радиоэлектронных устройств и встроенных микропроцессорных систем». Результаты диссертационного исследования применены в рамках практических занятий в виде методик расчётов оптимальных параметров каскада вычислительных устройств для реализации нейронной сети и заданий по схемотехническому моделированию и

реализации алгоритмов на языках Java и MicroC.

Эффект от внедрения результатов диссертационной работы заключается в повышении уровня освоения профессиональных компетенций и их компонентов (знаний, умений и владений) в области проектирования и реализации распределенных вычислительных систем. Это соответствует требованиям Федеральных государственных образовательных стандартов высшего образования нового поколения, построенных с учетом требований профессиональных стандартов.

Председатель комиссии:
доктор технических наук, доцент _____ / Фрейман В.И. /

Члены комиссии:
доктор технических наук, доцент _____ / Хижняков Ю.Н. /

кандидат технических наук, доцент _____ / Гончаровский О.В. /