Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»

*На правах рукописи*

КРИВОШЕЕВ
Аркадий Владимирович

# МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ СИСТЕМЫ МУЛЬТИАГЕНТНОГО АНСАМБЛИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ КОМПОНЕНТОВ РАСПОЗНАВАНИЯ ОБРАЗОВ

2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей

**ДИССЕРТАЦИЯ**
на соискание ученой степени
кандидата технических наук

Научный руководитель:
Иващенко Антон Владимирович,
доктор технических наук, профессор

САМАРА – 2024

# ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

**Актуальность темы исследования**

Распознавание образов является одним из активно развивающихся направлений научных исследований, связанным с автоматическим обнаружением закономерностей в данных с помощью компьютерных алгоритмов. Наиболее распространенная область применения математического и программного обеспечения распознавания образов – это системы компьютерного зрения. Теоретические исследования в этой области разделяют в зависимости от применяемых алгоритмов и наборов доступных данных по типу обучения: контролируемое и не контролируемое. С последним чаще связывают область применения технологий искусственного интеллекта, в частности искусственных нейронных сетей.

Несмотря на наличие успешных решений в этой области, в последнее время возникают новые задачи, требующие сочетания нескольких технологий искусственного интеллекта. Это связано с тем, что достаточно сложно создать и обучить единственную искусственную нейронную сеть, решающую широкий спектр задач в изменяющихся внешних условиях. Например, в интеллектуальных системах компьютерного зрения для контроля действий оператора, или отслеживания движений пациента медицинской реабилитации, наблюдается большое разнообразие типов движений, что в существующих решениях приводит к постоянному повторению циклов обучения и настройки алгоритмов.

Таким образом, актуальной является **научно-техническая задача** комплексирования автономных искусственных нейронных сетей в интеллектуальной системе распознавания образов, способной адаптироваться к меняющимся внешним условиям эксплуатации.

Большой вклад в теорию и практику разработки распределенного программного обеспечения с автономным поведением, в том числе

мультиагентных систем, внесли В.И. Городецкий, О.Н. Граничин, А.П. Еремеев, О.В. Карсаев, А.А. Кожухов, П.О. Скобелев, А.В. Соллогуб, В.Б. Тарасов, О.Я. Кравец, А.А. Макаренков, S. Liu, L. Xie, D. Ye, H. Zhang, M. Zhang. Известны работы по комбинированному применению нейронных сетей R. Asadi, L.C. Jain, P. Král, L. Lenc, C.P. Lim, N. Mustapha, A. Quteishat, M. Sulaiman, J. Tweedale, N. Ueda и др.

Однако архитектурное решение по мультиагентному ансамблированию нейронных сетей в распределенной системе искусственного интеллекта в настоящее время отсутствует.

**Целью диссертационной работы** является совершенствование программной архитектуры адаптивной системы распознавания образов путем реализации моделей и алгоритмов сочетания интеллектуальных компонентов с автономным поведением.

Для достижения поставленной цели были решены следующие **задачи**:

1. Разработка нового метода организации взаимодействия автономных интеллектуальных компонентов программного обеспечения для распознавания образов.
2. Разработка архитектуры мультиагентной программной системы, реализующей взаимодействия компонентов искусственного интеллекта.
3. Разработка алгоритмов распознавания образов в ансамбле агентов на основе искусственных нейронных сетей.
4. Реализация и исследование программного обеспечения мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов.
5. Апробация разработанного программного обеспечения на примерах распознавания образов в системах компьютерного зрения.

**Объектом исследования** диссертационной работы является интеллектуальное математическое и программное обеспечение распознавания образов.

**Предмет исследования** – способы сочетания и совместного функционирования автономных интеллектуальных компонентов в адаптивной системе искусственного интеллекта.

**Методы исследования.** Для решения поставленных задач использовались теории и технологии искусственных нейронных сетей, имитационного моделирования, поддержки принятия решений, мультиагентные технологии.

**Соответствие паспорту специальности.** Результаты исследования соответствуют пунктам специальности 3. Модели, методы, архитектуры, алгоритмы, языки и программные инструменты организации взаимодействия программ и программных систем; 4. Интеллектуальные системы машинного обучения, управления базами данных и знаний, инструментальные средства разработки цифровых продуктов; 7. Модели, методы, архитектуры, алгоритмы, форматы, протоколы и программные средства человеко-машинных интерфейсов, компьютерной графики, визуализации, обработки изображений и видеоданных, систем виртуальной реальности, многомодального взаимодействия в социокиберфизических системах.

**Научная новизна работы.** В диссертации получены следующие результаты, характеризующиеся научной новизной:

1. Метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов, отличающийся реализацией динамического комплексирования автономных искусственных нейронных сетей, позволяющий обеспечить адаптивность системы в условиях изменяющейся обстановки без переобучения интеллектуальных компонентов.

2. Архитектура мультиагентной системы распознавания образов, отличающаяся реализацией предиктивного оркестратора для согласования работы нескольких интеллектуальных агентов и позволяющая сократить время распознавания за счет более эффективного выбора агентов в отличие от классических моделей на основе ветвлений.

3. Мультиагентный алгоритм распределения задач в адаптивной системе распознавания образов, отличающийся возможностью динамического изменения критериев выбора интеллектуальных агентов при корректировке условий задачи распознавания образов и позволяющий повысить качество распознавания.

4. Структура распределенной системы компьютерного зрения на базе мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов, отличающаяся комбинированным применением искусственных нейронных сетей, предназначенных для решения частных задач и обеспечивающая адаптивность к меняющимся условиям эксплуатации без перенастройки алгоритмов распознавания.

**Теоретическая значимость работы** заключается в расширении области применения технологий распознавания образов на базе искусственного интеллекта путем реализации их совместного и комплексного использования в рамках мультиагентной архитектуры специализированного программного обеспечения.

**Практическая значимость работы** заключается в следующем. Предложенный метод и алгоритм организации взаимодействия автономных интеллектуальных компонентов программного обеспечения для распознавания образов и реализующая их архитектура программного обеспечения были использованы для реализации систем компьютерного зрения, текстопонимания и текстогенерации, что позволило расширить возможности их функционирования с учетом постоянных изменений условий эксплуатации.

**Достоверность результатов исследований** подтверждается корректностью использования теоретических методов, сравнением полученных результатов с результатами выполнения реальных проектов и апробацией предложенных разработок на практике.

**Основные положения, выносимые на защиту:**

1. Метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов обеспечивает адаптивность системы в условиях изменяющейся обстановки без переобучения интеллектуальных компонентов.

2. Архитектура мультиагентной системы распознавания образов на базе предиктивного оркестратора позволяет сократить время распознавания за счет более эффективного выбора агентов в отличие от классических моделей на основе ветвлений.

3. Мультиагентный алгоритм распределения задач в адаптивной системе распознавания образов позволяет повысить качество распознавания.

4. Структура распределенной системы компьютерного зрения на базе мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов обеспечивает адаптивность к меняющимся условиям эксплуатации без перенастройки алгоритмов распознавания.

**Апробация работы**

Результаты исследования внедрены в ООО «Открытый код» и использованы в системе фотофиксации приборов учета для энергосбытовых компаний, системе контроля ручных операций по видеоизображению с рабочего места оператора сборочного производства, библиотеке подпрограмм текстопонимания и текстогенерации в составе системы документооборота и управления цифровым контентом организации. Также результаты работы внедрены в учебный процессе ФГБОУ ВО «Самарский государственный технический университет» на программах бакалавриата и магистратуры по направлению «Программная инженерия».

Основные результаты работы докладывались и обсуждались на следующих конференциях: Annual science fiction prototyping conference (Брюгге, 2018, Гент, 2020); Международной конференции «Intelligent Systems Conference (IntelliSys)» (Лондон, 2018); European Simulation and Modeling

Conference (Пальма-де-Майорка, 2019, Рим, 2021); Moscow Workshop on Electronic and Networking Technologies (Москва-Пенза, 2020); Международной научной конференции «Математические методы в технике и технологиях ММТТ-33» (Казань, 2020); 27th Conference of Open Innovations Association (Тренто, 2020); Международной конференции «MIP: Engineering-2020» (Красноярск, 2020); Международной конференции «Creativity in Intelligent Technologies and Data Science» (Волгоград, 2021); Всероссийской научно-практической конференции «Мехатроника, автоматизация и управление на транспорте» (Самара, 2021); IX International Conference on Information Technology and Nanotechnology (Самара, 2023).

Диссертационная работа отмечена Дипломом призера Всероссийского инженерного конкурса для аспирантов в 2022/23 году.

**Публикации.** По результатам исследования опубликовано 28 работ, в том числе 6 статей в рецензируемых научных изданиях, рекомендованных ВАК РФ, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученой степени кандидата наук по специальности диссертации, и 16 статей в изданиях, индексируемых в международных информационных базах WoS и Scopus, получено 2 свидетельства о регистрации программ для ЭВМ и 1 патент.

**Структура и объем работы.** Диссертация состоит из введения, 4 глав, заключения, библиографического списка из 109 наименований и приложений. Общий объем работы составляет 155 страниц.

# ГЛАВА 1. МЕТОДЫ И ТЕХНОЛОГИИ ИНТЕЛЛЕКТУАЛЬНОГО РАСПОЗНАВАНИЯ ОБРАЗОВ

## 1.1 Проблемы практического применения технологий интеллектуального распознавания образов

Современные разработки в области искусственного интеллекта и дополненной реальности [1, 2] предоставляют широкие возможности в различных сферах цифровой экономики. Наиболее значимые результаты достигаются в области визуализации данных и поддержки принятия решений, направленных на развитие существующих приложений новыми возможностями сбора, обработки и управления данными. Реализуя эти технологии, традиционные компании превращаются в компании с цифровым мышлением, идущие по пути цифровой трансформации с учетом современных аспектов применения информационных технологий [3]. При этом большие надежды связываются с внедрением искусственного интеллекта в робототехнике, направленным на замену операторов автономными техническими устройствами. Несмотря на успех решений такого рода в промышленных приложениях, проблема взаимодействия людей и роботов в общем пространстве все еще остается открытой.

Совместная работа программно-аппаратных компонентов с автономным поведением и искусственным интеллектом и персонала предприятия идентифицируется как смешанный интеллект. Данные возможности широко рассматриваются в коллаборативной робототехнике [4]. Большинство разработчиков прилагают усилия, чтобы избежать такого взаимодействия или сократить его до минимально необходимых операций, связанных с запуском, настройкой и обслуживанием оборудования. Другие вводят ограничения по времени и пространству (например, размещают

роботизированное оборудование в специальных закрытых помещениях или боксах).

Однако дальнейшее развитие цифровой экономики приведет к необходимости нарушить строгое разделение сфер деятельности. Сотрудникам цифровых предприятий придется иметь дело с роботами, а производственная система, в которой ранее человек был главным игроком, при этом превращается в комбинированную среду, в которой человек работает в сочетании с автономными комплексами, устройствами и программами. Поэтому роль искусственного интеллекта в различных сферах жизни человека значительно возрастет.

Проблемы построения смешанного интеллекта тесно связаны с его определением. Когнитивные вычисления [5], основа искусственного интеллекта, подразумевают симуляцию активности человеческого мозга, которая позволяет машинам обрабатывать информацию, изучать мир и анализировать события, подобно людям, возможно, более продуктивно. Искусственный интеллект эффективно используется в таких областях, как производство, электроэнергетика, бизнес, медицина, образование, логистика, торговля, банковское дело и т.д.

Концепция «Индустрия 4.0» [6, 7] объявляет взаимодействие человека и искусственного интеллекта одной из основных целей. Взаимодействие означает способность машин, устройств, датчиков и людей соединяться и общаться друг с другом через Интернет вещей или Интернет людей. Эти процессы обладают свойством самоорганизации, в связи с чем, в распределенных сетях автономных сущностей необходимо применять методы децентрализованного управления. Это обеспечивает способность киберфизических систем принимать решения самостоятельно. Для реализации такого рода систем имитационного моделирования и управления широко применяются мультиагентные технологии [8, 9].

Практическое использование мультиагентных моделей основывается на следующих основных предположениях. Во-первых, программные агенты

должны соответствовать реальным активным субъектам по целям, ограничениям и логике взаимодействия. Это возможно для простых агентов, когда логику принятия решений можно задавать в виде правил в базе знаний. В случае введения более сложных агентов, таких как, например, Интернет боты, применение данных моделей затрудняется. Во-вторых, трудно обеспечить эффективное взаимодействие компьютерных агентов и действующих лиц в реальном времени из-за необходимости координировать это взаимодействие и обрабатывать непредсказуемые отклонения от бизнес-процессов, вызванные влиянием человеческого фактора.

Современные технологии дополненной реальности (AR) применяются не только в компьютерных играх, но и в промышленных приложениях для построения интерактивных пользовательских интерфейсов. Устройства дополненной реальности, такие как специальные очки, планшеты и смартфоны, предоставляют своим пользователям новые возможности по сравнению с классическими пользовательскими интерфейсами. Например, персоналу, производящему фотофиксацию счетчиков учета, на интерфейсе может быть выведена дополнительная информация о счетчике, предыдущих показаниях, дате следующей поверки и т.п. Однако, несмотря на высокую доступность и распространенность этих технологий, внедрение дополненной реальности на практике остается сложной задачей благодаря различиям в восприятии пользователями реальных и виртуальных объектов.

Технологии дополненной реальности широко используются в игровой индустрии, в образовании, интерактивных руководствах и системах поддержки принятия решений [10, 11]. При этом следует отличать особенности применения дополненной реальности в компьютерных играх и сфере развлечений и его использования в профессиональной деятельности. Использование AR-устройств в профессиональной среде сталкивается с трудностями, связанными с их второстепенной ролью. Из-за высокой неопределенности и низкой производительности такие устройства не могут

стать основой бизнес-процессов и часто дублируют существующие информационные технологии.

Основное отличие интерфейсов дополненной реальности заключается в их возможности размещать виртуальные и реальные объекты в одном пространстве и времени. Подсказки и панели управления могут быть представлены пользователю в любой точке трехмерной сцены. Они могут перекрывать и даже пересекаться друг с другом. В связи с этим, технология дополненной реальности должна создавать интерактивные и контекстно-зависимые пользовательские интерфейсы, которые обеспечивают возможности компьютерного зрения и распознавания объектов.

В результате пользователь попадает в цикл человеко-компьютерного взаимодействия [12], что приводит к зависимости логики построения пользовательского интерфейса от особенностей человеческого восприятия. Система должна связывать отображаемые наборы данных (например, изображения, текст, измеренные значения, сканы) и визуальные сцены. Предлагается вовлекать лиц, принимающих решения, в процесс обработки и визуализации данных посредством постоянного взаимодействия с системой, что помогает оптимизировать поведение, как людей, так и алгоритмов.

## 1.2 Распознавание образов с помощью искусственных нейронных сетей

Нейронная сеть – это математическая модель, основной частью которой является искусственный нейрон, осуществляющий нелинейное преобразование суммы произведения входных сигналов на весовые коэффициенты [13].

Сфера ИИ (искусственный интеллект) – это автоматизация задач, ранее решаемых лишь человеческим интеллектом. Этот огромный спектр разнородных задач включает в себя очень разнообразные методы:

- машинное обучение;

- глубокое обучение;
- генетические алгоритмы;
- множество подходов, не связанных на прямую с обучением.

Система машинного обучения, как следует из названия, обучается, но не программируется в явном виде. На вход система принимает многочисленные и часто разнородные примеры, имеющие отношение к решаемой задаче, а она ищет во входных примерах некую статистическую структуру. В свою очередь, данная структура позволяет алгоритму определить правила для автоматического решения поставленной задачи.

Модель машинного обучения преобразовывает данные на входе в значимые результаты, «обучаясь» на данных ей примерах входных данных и результатов. Поэтому главная задача машинного и глубокого обучения – значимое преобразование исходных данных или, говоря другими словами, обучение представлению исходных данных, максимально приближающему нас к ожидаемому результату.

Глубокое обучение – это особый раздел машинного обучения: другой подход к поиску представления данных. Он делает упор на анализ последовательных слоев (или уровней) все более значимых представлений. Под глубиной в глубоком обучении подразумевается не более глубокое понимание, которое достигается этим подходом, а многослойность в представлении модели. Количество слоев, из которых состоит модель данных, называют глубиной модели.

В глубоком обучении многослойные представления изучаются (чаще всего) с применением моделей, называемых нейронными сетями. Их структура представлена в виде слоев, наложенных друг на друга. Понятие нейронной сети заимствовано из нейробиологии. Хотя источником некоторых основополагающих идей глубокого обучения частично являются науки о мозге, модель глубокого обучения не является моделью мозга человека. Нет фактов, доказывающих, что мозг работает по принципам,

подобным механизмам, которые используются в современных моделях глубокого обучения.

Представления, полученные алгоритмом глубокого обучения, можно представить так, как показано на рис. 1.

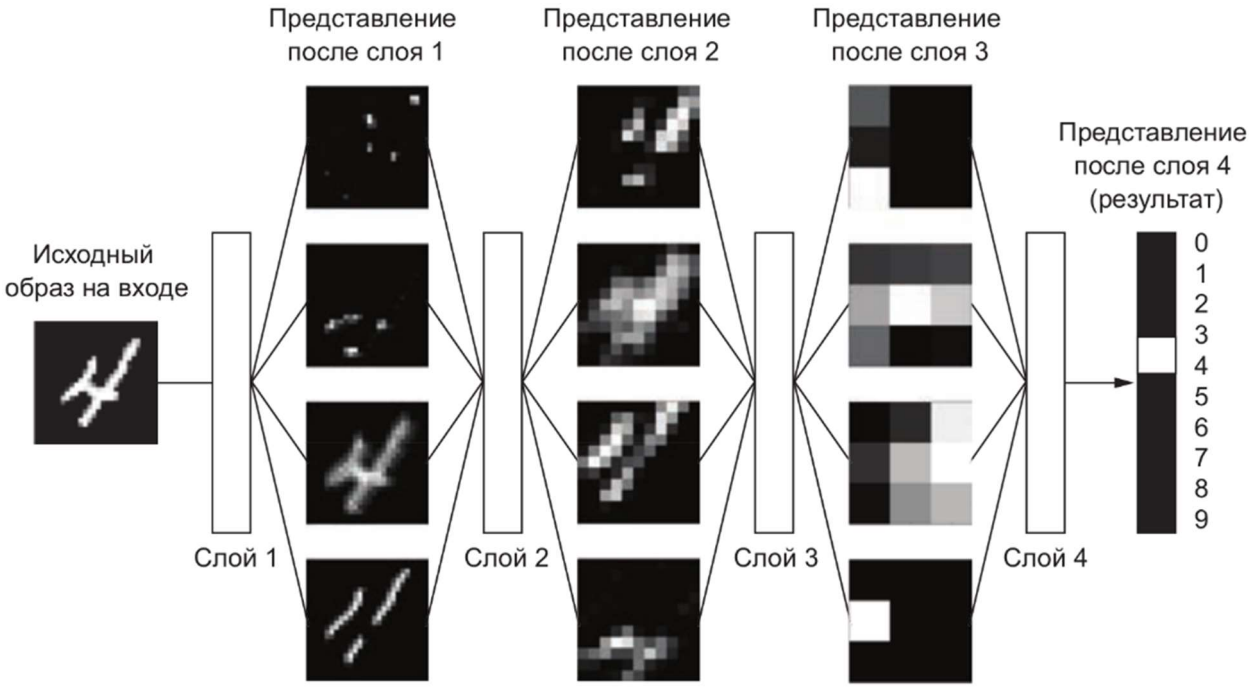


Рис. 1. Глубокая нейронная сеть для классификации цифр

С каждым новым слоем глубокая нейронная сеть уходит все дальше от начального изображения (представленного матрицей размером пикселей, состоящей из цифр от 0 до 255). Сеть абстрагируется все больше и больше, пока на последнем (выходном) слое не выдаст цифру, которую мы видим на изображении.

Глубокое обучение достигло множества прорывов в сложных для машинного обучения областях:

- классификация изображений на уровне человека;
- распознавание речи на уровне человека;
- распознавание рукописного текста на уровне человека;
- повышение качества машинного перевода с одного языка на другой;
- улучшение качества чтения текста вслух машиной;

- создание цифровых помощников: Yandex – Алиса, Google Now и Amazon – Alexa;
- управление автомобилем на уровне, сравнимом с человеком;
- повышение точности целевой рекламы Google, Baidu и Bing;
- повышение релевантности поиска в Интернете;
- машины могут отвечать на вопросы, заданные вслух;
- алгоритм обыграл человека в «Го».

Глубокие нейронные сети автоматизировали одну из сложных задач машинного обучения – выбор признаков, значимых для решения задачи, из множества доступных. Глубокие нейронные сети автоматически извлекают важные признаки в процессе обучения. Но за эту возможность приходится «расплачиваться» очень большими вычислительными ресурсами. Стоит отметить, что это касается лишь обучения нейронной сети. Обученная нейронная сеть работает быстро и требует очень мало ресурсов (способна работать даже на мобильном устройстве). К тому же нейронная сеть дает еще одно преимущество перед другими алгоритмами машинного обучения – она может дообучаться на отдельных примерах без полного переобучения на всех данных.

Методика глубокого обучения имеет две важные характеристики:

1. Поэтапно, послойно конструирует все более сложные представления.

2. Исследует промежуточные представления совместно, за счет чего каждый слой обновляется в соответствии с информацией, полученной от представлений других слоев.

Сверточные нейронные сети (convolutional neural networks, CNN) — широкий класс архитектур, основная идея которых состоит в том, чтобы переиспользовать одни и те же части нейронной сети для работы с разными небольшими, локальными участками входов. Сверточные нейронные сети применяются для решения разнообразных задач, но основным применением является обработка изображений [14].

Сверточные нейронные сети состоят преимущественно из сверточных слоев и слоев объединения.

Сверточный слой — это основной блок сверточной нейронной сети. Нейроны в первом сверточном слое не связаны с каждым одиночным пикселем во входном изображении, а только с пикселями в собственных рецепторных полях (рис. 2). В свою очередь каждый нейрон во втором сверточном слое связан только с нейронами, находящимися внутри небольшого прямоугольника в первом слое. Данная связь осуществляется с помощью операции под названием «свертка». В итоге такая архитектура позволяет сети сосредоточиться на низкоуровневых признаках в первом скрытом слое, затем скомпоновать их в признаки более высокого уровня в следующем скрытом слое и т.д. [15].

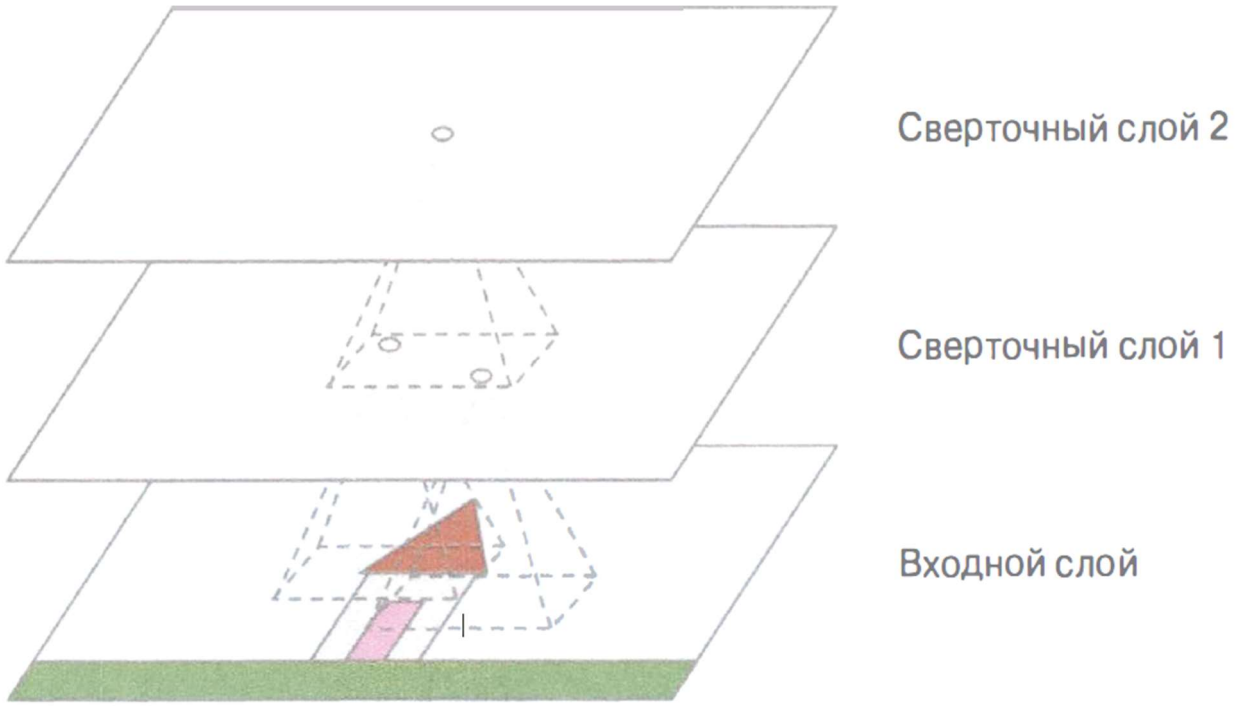


Рис. 2. Слои сети CNN с прямоугольными локальными рецепторными полями

Свертка — это математическая операция, которая плавно перемещает одну функцию по другой и измеряет интеграл их точечного умножения [15].

Если $x^l$ — карта признаков в слое под номером $l$, то результат двумерной свертки с ядром размера 2d + 1 и матрицей весов W размера (2d + 1) * (2d + 1) на следующем слое будет:

$$y_{i,j}^{l} = \sum_{-d \le a,b \le d} W_{a,b} x_{i+a,j+b}^{l},$$

Где $y_{i,j}^{l}$ — результат свертки на уровне $l$,

$x_{i,j}^{l}$— ее вход, и по совместительству выход всего предыдущего слоя.

Свертка также может иметь и прямоугольным ядро.

Пример свертки приведен на рис. 3.

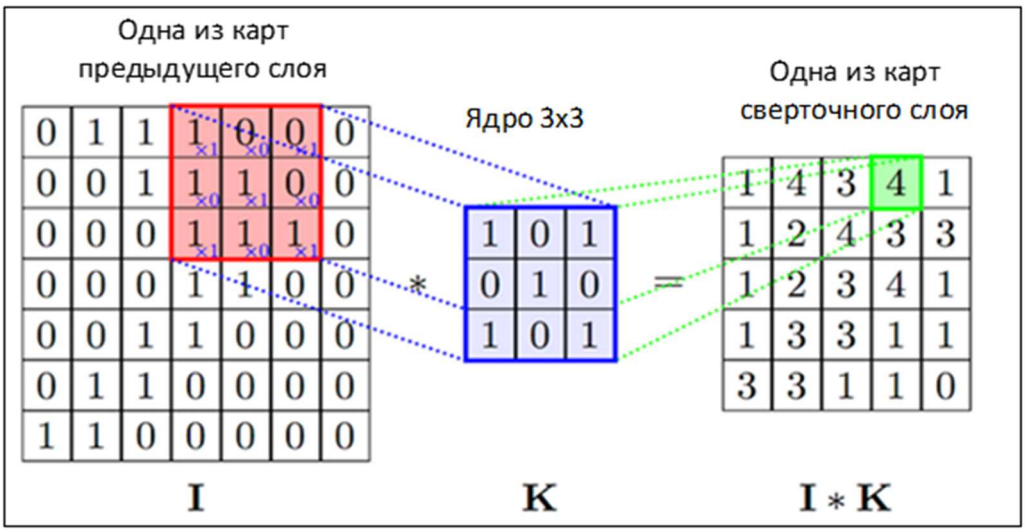


Рис. 3. Пример свертки

Операция свертки обладает следующими свойствами:

• сохраняет структуру входа (порядок в одномерном случае, взаимное расположение пикселов в двумерном и т. д.), так как применяется к каждому участку входных данных в отдельности;

• обладает свойством разреженности, так как значение каждого нейрона очередного слоя зависит только от небольшой доли входных нейронов;

• многократно переиспользует одни и те же веса, так как они повторно применяются к различным участкам входа.

Почти всегда после свертки в нейронной сети следует нелинейность, которая записывается следующим образом:

$$z_{i,j}^{l} = h(y_{i,j}^{l})$$

В качестве функции $h$ наиболее часто используются следующие функции активации [16]:

1) Тождественная:

$$h(x) = x;$$

2) Логистическая:

$$h(x) = \sigma(x) = \frac{1}{1 + e^{-x}};$$

3) Гиперболический тангенс (th, tanh):

$$h(x) = th(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})};$$

4) Линейный выпрямитель (Rectified linear unit, ReLU):

$$h(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases};$$

5) Линейный выпрямитель с «утечкой» (Leaky rectified linear unit, Leaky ReLU):

$$h(x) = \begin{cases} 0.01x, & x < 0 \\ x, & x \geq 0 \end{cases};$$

6) Экспоненциальная линейная функция (Exponential linear unit, ELU):

$$h(x) = \begin{cases} \alpha(e^{x} - 1), & x < 0 \\ x, & x \geq 0 \end{cases};$$

7) Масштабированная экспоненциальная линейная функция (Scaled exponential linear unit, SELU):

$$h(x) = 1.0507 \begin{cases} 1.67326(e^{x} - 1), & x < 0 \\ x, & x \geq 0 \end{cases}.$$

Наиболее часто используемой функцией является ReLU [15].

Изображения, подающиеся на вход нейронной сети, представляются в виде нескольких прямоугольных матриц, каждая из которых задает уровень одного из цветовых каналов в каждом пикселе изображения. Так, например, изображение размером 200 х 200 пикселов является набором из 120 000 чисел, три матрицы интенсивностей размером 200 х 200 каждая. Если

изображение черно-белое, то такая матрица будет одна. Для результатов масс-спектрометрии, где в каждом пикселе находится целый спектр, матриц может оказаться достаточно много. Но предполагается, что в каждом пикселе входного изображения стоит некоторый тензор (обычно одномерный, то есть вектор чисел), и его компоненты называются каналами. Такие же матрицы будут получаться и после прохождения через сверточный слой: в них по-прежнему будет пространственная структура, похожая на исходное изображение, однако каналов может стать больше. Значения каждого признака, полученное в результате свертки по области в исходном изображении, теперь будут представлять собой целую матрицу. Каждая такая матрица называется картой признаков. Каналы исходного изображения в литературе называются картами признаков, аналогично карты признаков очередного слоя иногда называют каналами [17].

Внутри одной карты признаков все нейроны разделяют те же самые параметры (веса и член смещения), но разные карты признаков могут иметь отличающиеся параметры. Рецепторное поле нейрона распространяется на все карты признаков предшествующих слоев. Т.е., сверточный слой одновременно применяет множество фильтров к своим входам, становясь способным обнаруживать множество признаков повсюду в своих входах (см. рис. 4).

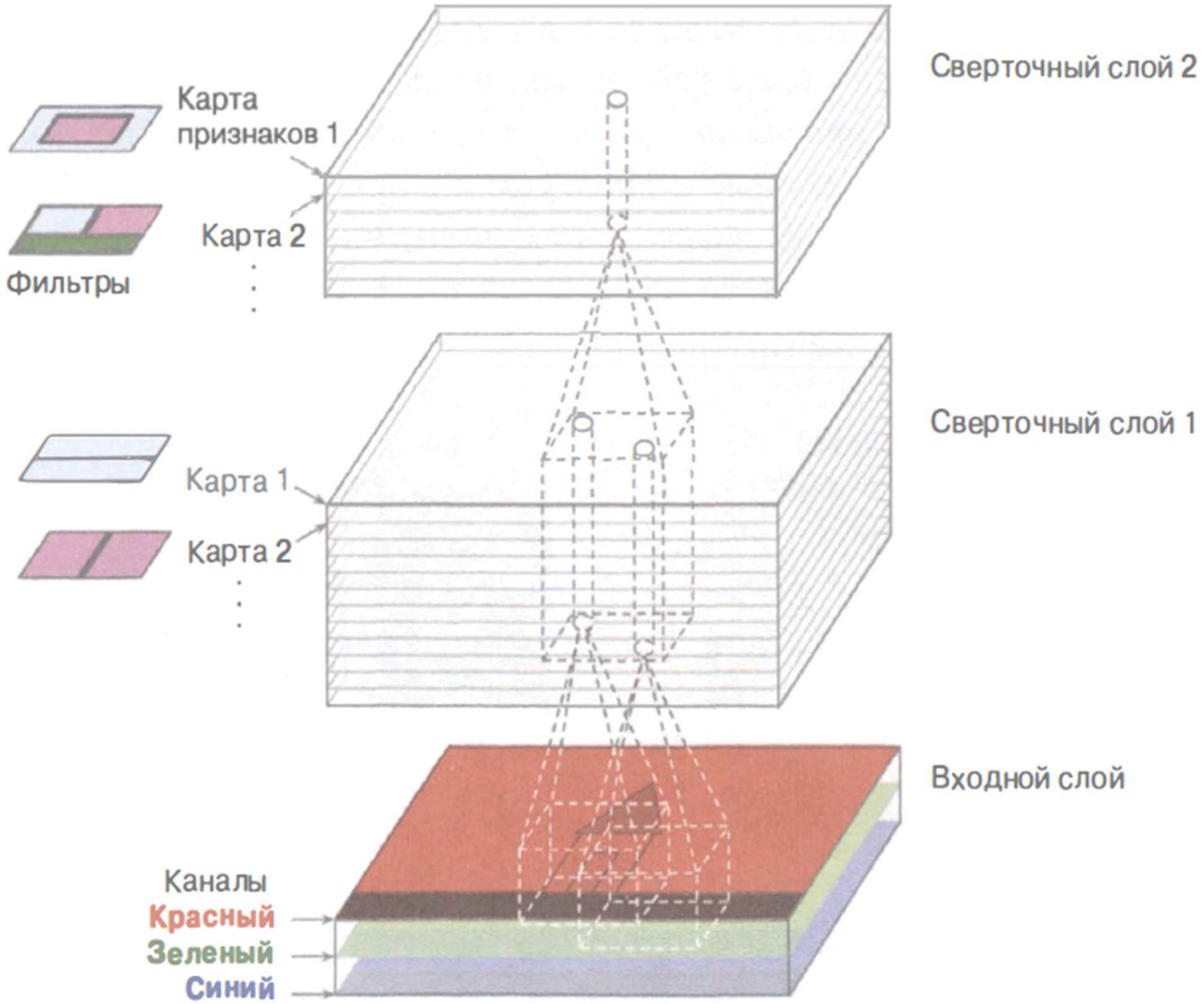


Рис. 4. Сверточные слои с множеством карт признаков

Таким образом, вычисление выход каждого нейрона в сверточном слое осуществляется с помощью следующего уравнения [15]:

$$z_{i,j,k} = h\left(b_k + \sum_{u=0}^{f_h-1}\sum_{v=0}^{f_w-1}\sum_{k'=0}^{f_{l-1}-1} x_{i',j',k'} \cdot w_{u,v,k',k}\right),$$

$$i' = i + u, \quad j' = j + v,$$

где $z_{i,\,j,\,k}$ – выход нейрона, расположенного в строке i и столбце j в карте признаков k сверточного слоя $l$;

$h$ – функция активации нейрона;

$f_h$ и $f_w$ – высота и ширина рецепторного поля;

$f_{l-1}$ – количество карт признаков в предыдущем слое $l$ - 1;

$x_{i',j',k'}$ – выход нейрона, расположенного в слое $l$ - 1, строка i', столбец j', карта признаков k';

$b_k$ – член смещения для карты признаков k (в слое $l$);

$w_{u,v,k',k}$ – вес связи между любым нейроном в карте признаков k слоя $l$ и его входом, расположенным в строке $u$, столбце $v$ (относительно рецепторного поля нейрона) и карте признаков k'.

Слой объединения (pooling layer, также его называют слоем субдискретизации или подвыборки) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей прямоугольного размера уплотняется до одного пикселя, проходя некоторое нелинейное преобразование.

В сверточных сетях обычно исходят из предположения, что наличие или отсутствие того или иного признака гораздо важнее, чем его точные координаты, поэтому данный слой служит для прореживания (т.е. сжатия) входного изображения для сокращения вычислительной нагрузки, расхода памяти и количества параметров (тем самым ограничивая риск переобучения).

Обычно в качестве операции объединения к каждой локальной группе нейронов применяется операция взятия максимума. Иногда встречаются и другие операции объединения, например взятие среднего, однако именно максимум встречается на практике чаще всего и для большинства практических задач дает хорошие результаты. Формально операция объединения определяется так [17]:

$$x_{i,j}^{l+1} = \max_{-d \le a,b \le d} z_{i+a,j+b}^{l},$$

где $x_{i,j}^{l+1}$ – значения нейрона в строке $i$ столбце $j$ слоя объединения $l$;

$z_{i+a,j+b}^{l}$ – значение нейрона в строке $i$+$a$ столбце $j$+$b$ сверточного слоя $l$;

$d$ – размер окна объединения, для d = 2 эта ситуация проиллюстрирована на рис. 5.

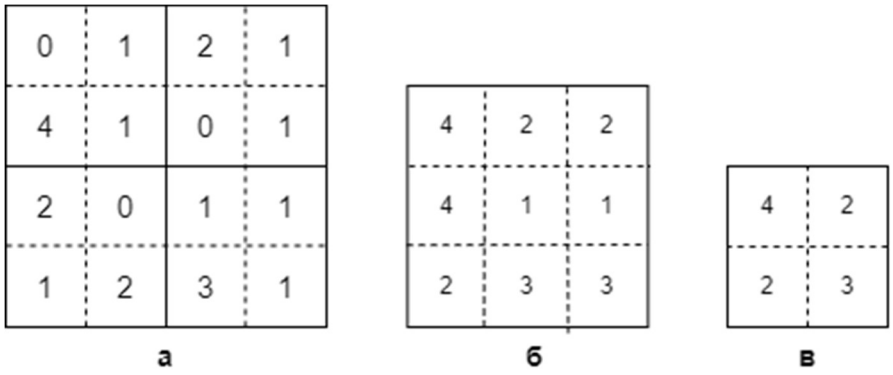


Рис. 5. Пример объединения с окном размера 2x2 : *а* — исходная матрица; *б* — матрица после объединения с шагом 1; *в* — матрица после объединения с шагом 2. Штриховка в исходной матрице *а* — соответствует окнам, по которым берется максимум с шагом 2; в части *в* — результат показан соответствующей штриховкой

Типовая архитектура сверточной сети состоит из последовательно составленных сверточных и объединяющих слоев. Выход очередного слоя используется как вход для следующего, а разные карты признаков служат каналами. Размер слоя за счет объединения будет постепенно сокращаться, и в конце концов последние слои сети смогут просматривать весь вход, а не только маленькое окошечко из него. На конце такой сети добавляются обычные полносвязные слои прямого распространения (рис. 6), они служат для объединения карт признаков.

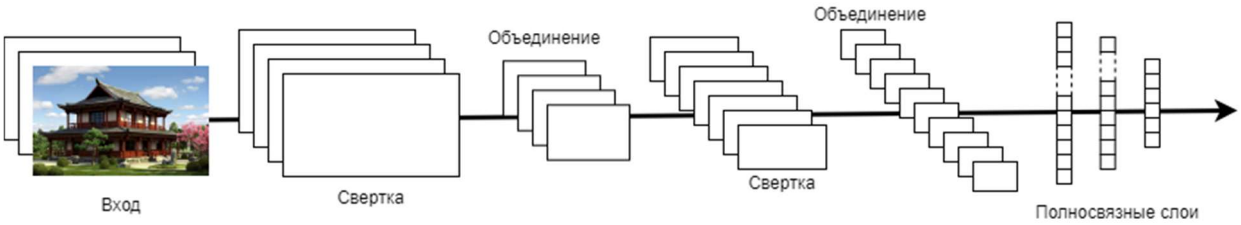


Рис. 6. Типовая архитектура сети CNN

LeNet была одной из первых сверточных нейронных сетей и способствовала развитию глубокого обучения. С 1988 года, после многих лет

исследований и множества успешных итераций, новаторская работа получила название LeNet-5.

В 1989 г. Yann LeCun и др. в Bell Labs впервые применили алгоритм обратного распространения к практическим приложениям и полагали, что способность изучать обобщение сети может быть значительно улучшена за счет предоставления ограничений из предметной области задачи. Он объединил сверточную нейронную сеть, обученную алгоритмами обратного распространения, для чтения рукописных чисел и успешно применил её для идентификации рукописных номеров почтовых индексов, предоставленных Почтовой службой США. Это был прототип того, что позже стало называться LeNet [18]. В том же году LeCun описал небольшую задачу распознавания рукописных цифр в другой статье и показал, что, хотя проблема линейно разделима, однослойные сети демонстрируют плохие возможности обобщения. При использовании инвариантных к сдвигу детекторов признаков в многослойной сети с ограничениями модель может работать очень хорошо. Он считал, что эти результаты доказывают, что минимизация количества свободных параметров в нейронной сети может повысить способность нейронной сети к обобщению [19].

В 1990 году в их статье снова описывалось применение сетей обратного распространения для распознавания рукописных цифр. Они выполнили только минимальную предварительную обработку данных. Входные данные состояли из изображений, каждое из которых содержало число, а результаты тестирования цифровых данных почтового индекса, предоставленные Почтовой службой США, показали, что модель имеет коэффициент ошибок всего 1% и процент отказов около 9% [20].

Их исследования продолжались в течение следующих восьми лет, и в 1998 году Yann LeCun, Leon Bottou, Yoshua Bengio, и Patrick Haffner рассмотрели различные методы распознавания рукописных символов на бумаге и использовали стандартные рукописные цифры для определения эталонных задач. Эти модели сравнивались, и результаты показали, что сеть

превзошла все остальные модели. Они также предоставили примеры практического применения нейронных сетей, такие как две системы для распознавания рукописных символов в Интернете и модели, которые могли считывать миллионы чеков в день [21].

LeNet обладает основными элементами сверточной нейронной сети, такими как сверточный слой, слой объединения и полносвязный слой, см. рис. 7.

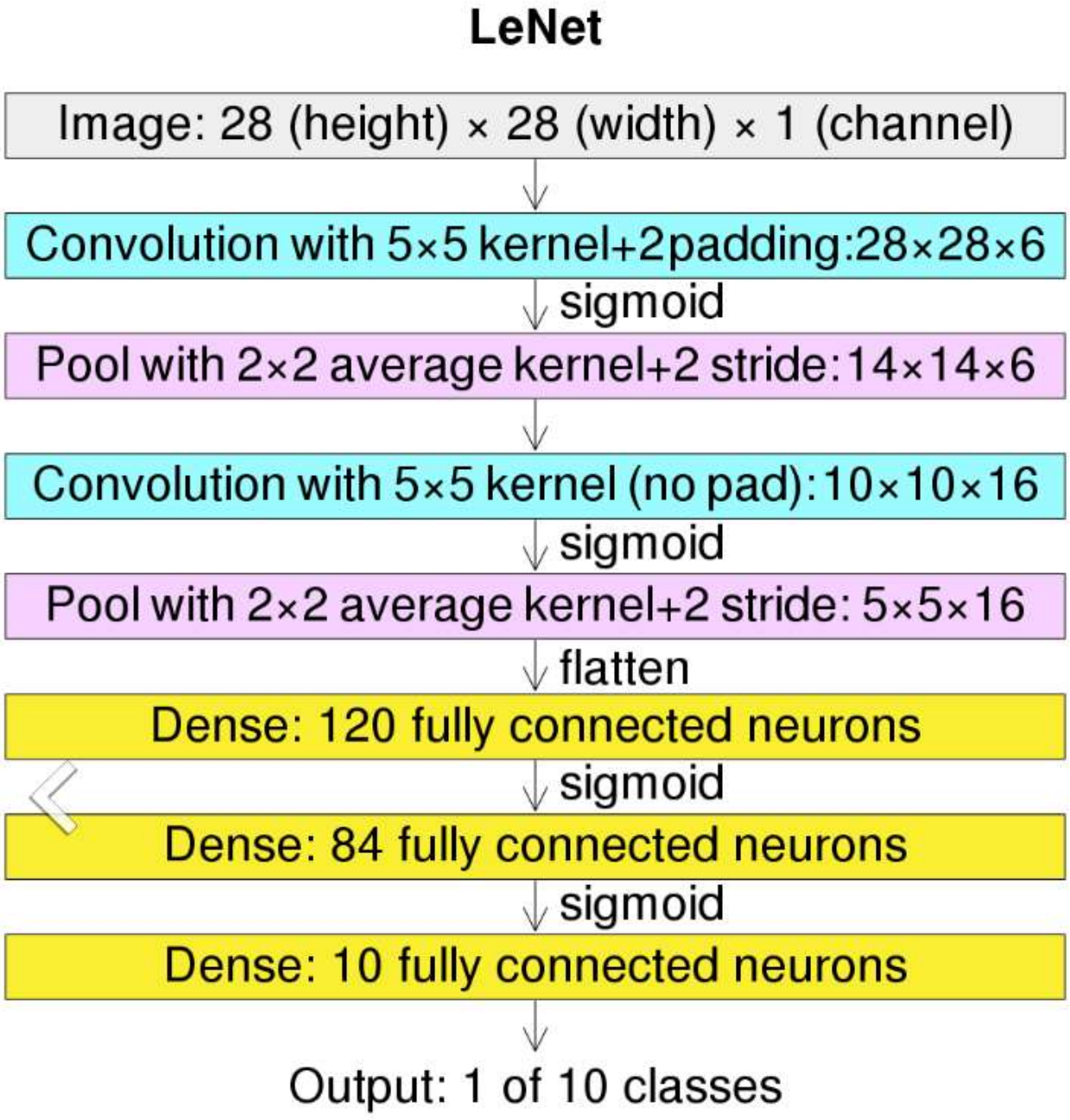


Рис. 7. Архитектура LeNet-5

Как показано на рисунке (входные данные изображения размером 32*32 пикселя): LeNet-5 состоит из семи слоев. Помимо ввода, любой другой слой может обучать параметры. Слой свертки Convolutional в дальнейшем обозначаются буквой C, слои объединения Pool – буквой S, а полносвязные слои - буквой F.

Слой C1 представляет собой слой свертки с шестью ядрами свертки 5x5 и размером отображения признаков 28x28, он помогает предотвратить выпадение информации входного изображения за границу ядра свертки. Слой S2 — это слой подвыборки/объединения, который создает 6 изображений признаков размером 14x14. Каждая ячейка в каждой карте объектов связана с районами 2x2 на соответствующей карте объектов в C1. Слой C3 представляет собой слой свертки с 16 ядрами свертки 5x5. Входными данными первых шести карт признаков C3 является каждое непрерывное подмножество трех карт признаков в S2, входными данными следующих шести карт признаков является вход четырех непрерывных подмножеств, а входными данными следующих трех карт признаков является четыре прерывистых подмножества. Наконец, входные данные для последнего графа признаков поступают из всех графов признаков S2. Слой S4 подобен S2, с размером 2x2 и выводом 16 графиков признаков 5x5. Слой C5 — это слой свертки со 120 ядрами размером 5x5. Каждая ячейка связана с соседством 5*5 на всех 16 графах признаков S4. Здесь, поскольку размер графа признаков S4 также равен 5x5, выходной размер C5 равен 1*1. Таким образом, S4 и C5 полностью связаны. C5 помечен как сверточный слой вместо полносвязного слоя, потому что, если вход LeNet-5 станет больше, а его структура останется неизменной, его выходной размер будет больше 1x1, т.е. он по факту перестанет являться полносвязным слоем. Слой F6 полностью подключен к C5, и выводятся 84 графа признаков. Слой F7 является выходным и содержит 10 выходов, по одному на каждую цифру.

В настоящее время алгоритмы искусственного интеллекта и машинного обучения широко используются для анализа изображений и распознавания образов [22 - 26]. Проблемы разработчиков этих алгоритмов связаны с выбором наиболее подходящей модели нейронной сети, сбором наиболее эффективного обучающего набора данных и тем самым повышением качества анализа. Реализация интеллектуального распознавания на практике часто ограничивается масштабом и спецификой проблемной области.

Следующие модели обнаружения объектов традиционно используются для определения положения объектов на изображении: Regions With CNNs (RCNN), Single Shot MultiBox Detector (SSD) и You Only Look Once (YOLO) [27]. Эти модели позволяют определять положение объекта на изображении и классифицировать его. Для решения этой проблемы они применяют один из двух разных подходов.

Первый разбивает изображение на области и классифицирует каждую область на разные классы объектов (RCNN и его производные). Второй рассматривает обнаружение объектов как проблему регрессии или классификации (YOLO, SSD и т. Д.). По результатам сравнения моделей обнаружения объектов в [28, 29], модели на основе архитектуры YOLOv2 показывают хорошие результаты при наивысшей скорости обработки изображений. Для мобильных устройств скорость обработки является такой же важной характеристикой модели, как и точность. Поэтому модель YOLOv2 - одна из лучших для использования в мобильных устройствах.

Рекуррентные нейронные сети (РНС, англ. Recurrent neural network; RNN) — вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей произвольной длины. Поэтому сети RNN

применимы в таких задачах, где нечто целостное разбито на части, например: распознавание рукописного текста или распознавание речи [30].

В более общем плане сети RNN могут работать с последовательностями произвольной длины, а не только с входными данными фиксированного размера, как многие другие нейронные сети. Например, они в состоянии принимать на входе предложения, документы или аудио-видео-образцы, приведенные в цифровой вид, что делает их чрезвычайно полезными для систем обработки естественного языка, смыслового анализа или анализа видео-потока [17].

Классические нейронные сети (такие как многослойные перцептроны) имеют фиксированное число входов и воспринимают каждый из них как независимый. В рекуррентных же сетях связи между нейронами могут идти не только от нижнего слоя к верхнему, но и от нейрона к «самому себе», точнее, к предыдущему значению самого этого нейрона или других нейронов того же слоя [17]. Именно это позволяет отразить зависимость переменной от своих значений в разные моменты времени: нейрон обучается использовать не только вход на текущий временной шаг, но и собственный выход с предыдущего временного шага. Можно представить нейронную сеть из одного нейрона, развернутую по оси времени, см. рис. 8 (справа). Процесс называется развертыванием сети во времени [15].

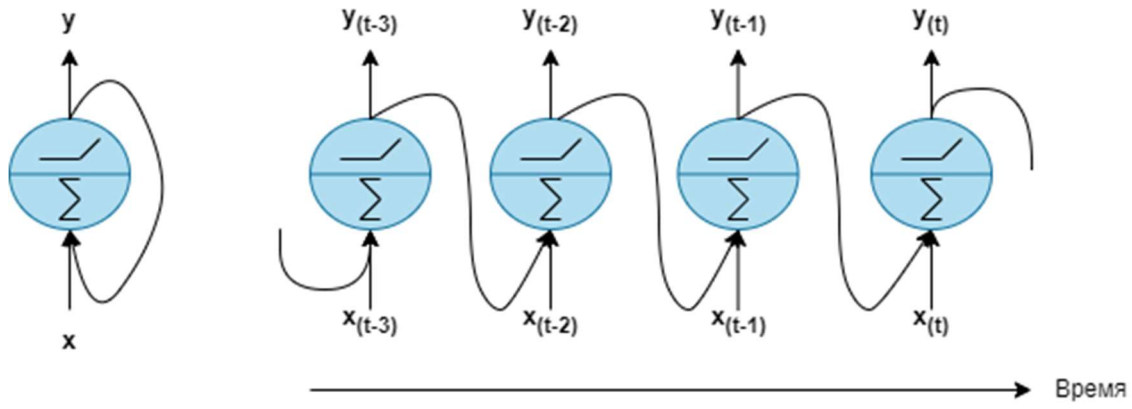


Рис. 8. Рекуррентный нейрон (слева), развернутый во времени (справа)

Из нейронов можно создавать слои. На каждом временном шаге t каждый нейрон получает входной вектор $x_{(t)}$ и выходной вектор из предыдущего временного шага $y_{(t-1)}$ что продемонстрировано на рис. 9. Если для одного нейрона выход был скаляром, то для слоя и вход, и выход являются векторами [15].

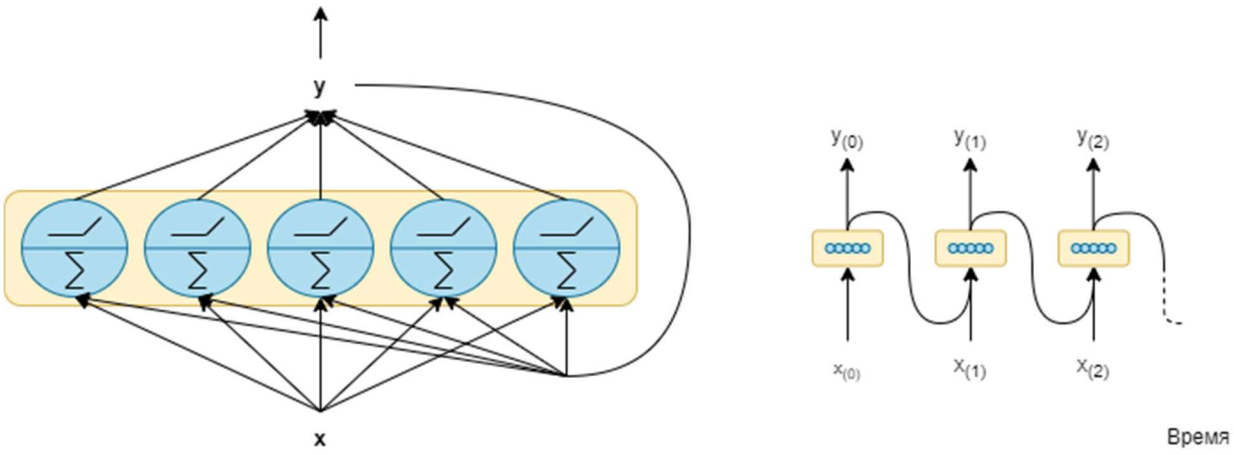


Рис. 9. Слой рекуррентных нейронов (слева), развернутый во времени (справа)

При использовании стохастического градиентного спуска выход рекуррентного слоя определяется как [17]:

$$y_{(t)} = h\left(W_x^T \cdot x_{(t)} + W_y^T \cdot y_{(t-1)} + b\right),$$

где h – функция активации;

$W_x$ – матрица $n_{входов}$ x $n_{нейронов}$, содержащая веса связей для входов текущего временного шага;

$n_{входов}$ – количество входных признаков;

$n_{нейронов}$ – количество нейронов;

$x_{(t)}$ – входные признаки на временном шаге t;

$W_y$ – матрица $n_{нейронов}$ x $n_{нейронов}$, содержащая веса связей для выходов предыдущего временного шага;

$y_{(t-1)}$ – выход слоя на временном шаге t-1;

b – вектор размера $n_{нейронов}$, содержащий член смещения каждого нейрона.

В случае выбора мини-пакетного градиентного спуска, когда несколько образцов подаются на вход нейронной сети, выход рекуррентного слоя преобразуется в:

$$Y_{(t)} = h\left(W_x^T \cdot X_{(t)} + W_y^T \cdot Y_{(t-1)} + b\right),$$

где $Y_{(t)}$ – матрица $m$ x $n_{нейронов}$, содержащая выходы слоя на временном шаге t для каждого образца в мини-пакете;

m - количество образцов в мини-пакете;

$X_{(t)}$ – матрица $m$ x $n_{нейронов}$, содержащая выходы слоя на временном шаге t для каждого образца в мини-пакете [15].

Поскольку выход рекуррентного нейрона на временном шаге t — это функция от всех входов из предшествующих временных шагов, можно считать, что он обладает некоторой формой памяти. Часть нейронной сети, которая сохраняет состояние через временные шаги, называется ячейкой памяти или просто ячейкой. Одиночный рекуррентный нейрон или слой рекуррентных нейронов представляет собой самую базовую ячейку.

По характеру входов и выходов рекуррентные нейронные сети делят на пять вариантов [17]:

а) один вход, один выход (one-to-one, рис. 10, а); в данном случае рекуррентная сеть используется как обычный перцептрон: независимо обрабатывается каждый элемент входов и на его основе получается соответствующий выход, скрытые состояния не передаются и не используются;

б) один вход, последовательность выходов (one-to-many, рис. 10, б) в данном случае единичный вход разворачивается в последовательность выходов; например, аннотирование изображений представляет собой такую задачу: на входе изображение, на выходе текст (последовательность);

в) последовательность входов, один выход (many-to-one, рис. 10, в); используется для решения задачи классификации последовательностей;

например, анализ видеоряда: по последовательности изображений можно классифицировать действия объекта;

г) последовательность входов, затем последовательность выходов (many-to-many, рис. 10, г); В данном случае входную последовательность кодируется неким скрытым состоянием, а затем раскодируется обратно в уже совершенно другую последовательность; например, по этой общей схеме работают системы машинного перевода (вход — предложение на одном языке, выход — на другом) и диалоговые системы (вход — реплика собеседника, выход — своя собственная реплика);

д) синхронизированные последовательности входов и выходов (synchronized many-to-many, рис. 10, д); частный случай «many-to-many», используется, когда необходимо переносить на следующий временной шаг скрытое состояние; например, когда нужно разметить видеопоток, в котором каждый последующий кадр, представляет собой самостоятельную картинку, но с большой вероятностью похожую на предыдущую и следующую [17].

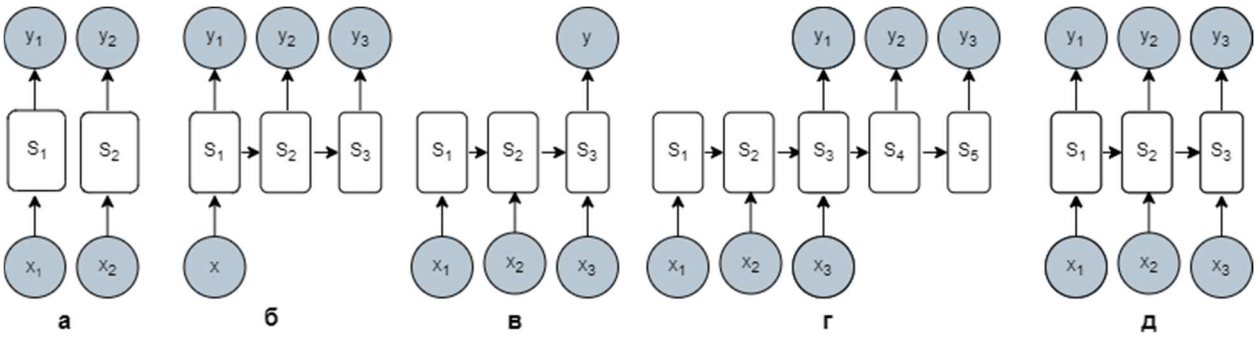


Рис. 10. Варианты использования рекуррентных сетей: а — один вход, один выход; б — один вход, последовательность выходов; в — последовательность входов, один выход; г — последовательность входов, затем последовательность выходов; д — синхронизированные последовательности входов и выходов.

Для обучения сеть RNN на длинных последовательностях, необходимо прогонять ее через множество временных шагов, делая развернутую сеть RNN очень глубокой. Подобно любой глубокой нейронной сети сеть RNN подвергается проблемам исчезновения/взрывного роста

градиентов что приводит к бесконечно долгому обучению. Второй проблемой сетей RNN является постепенное исчезновение памяти первых входов. Из-за трансформаций, которым подвергаются данные при проходе через сеть RNN, после каждого временного шага определенная информация утрачивается. Через некоторое время состояние сети RNN практически не содержит следов первых входов. При наличии в начале последовательности важной информации, результаты работы рекуррентной сети будут неудовлетворительными. Для решения проблемы были выведены ячейки с долговременной памятью, лишенные данного недостатка. Одной из самых используемых ячеек долговременной памяти является ячейка LSTM.

Ячейка долгой краткосрочной памяти LSTM (Long Short-Term Memory) модификация базовой ячейки, предложенная в 1997 году [31] Сеппом Хохрайтером и Юргеном Шмидхубером, и с годами усовершенствованная такими учеными, как Алекс Грейвз, Хасим Сак [32], Войцех Заремба [33] и многими другими.

Архитектура ячейки представлена на рис. 11. Ячейка LSTM выглядит в точности как обыкновенная ячейка, но состояния ячейки разделяется на два вектора: h(t) – краткосрочное состояние и c(t) – долгосрочное состояние.

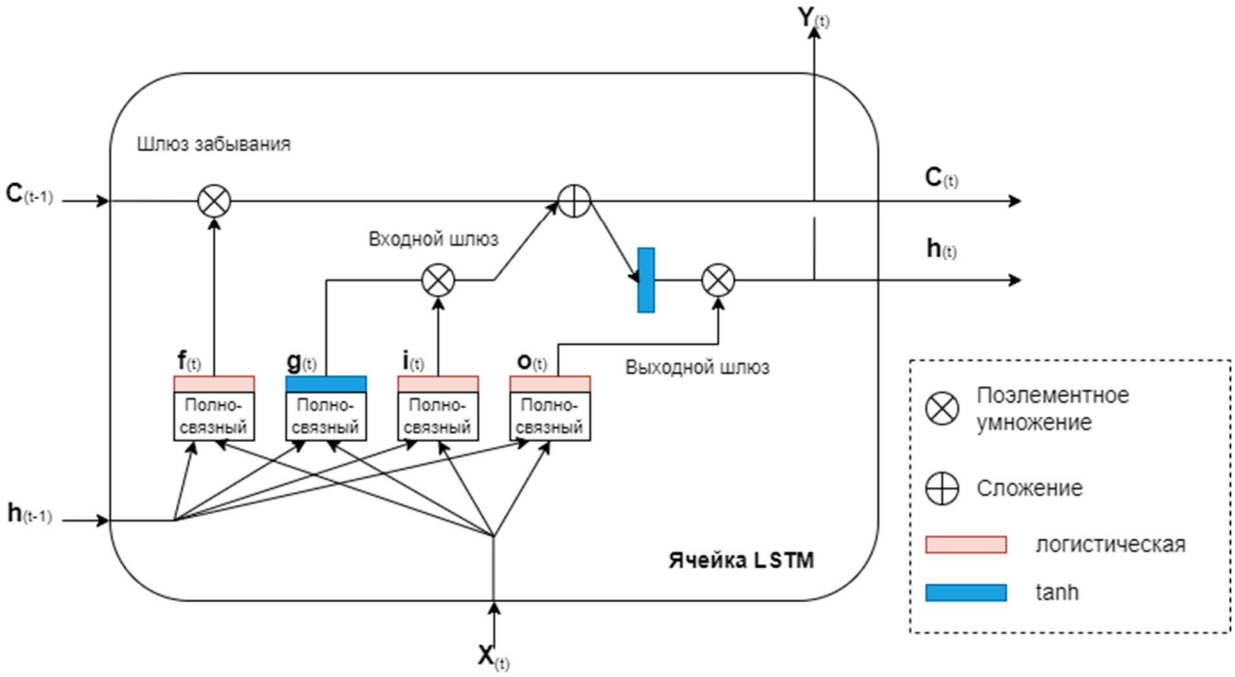


Рис. 11. Ячейка LSTM

Основная идея такой архитектуры состоит в том, что нейронная сеть может узнать, что ей следует хранить в долгосрочном состоянии, а что со временем отбрасывать. При работе нейронной сети долгосрочное состояние $c_{(t-l)}$ изначально проходит через шлюз забывания, в котором некоторые воспоминания могут быть отброшены, и затем посредством операции сложения к нему добавляется ряд новых воспоминаний, выбранных входным шлюзом. Результат $c_{(t)}$ подается прямо на выход без каких-либо дальнейших трансформаций. Таким образом, на каждом временном шаге одни воспоминания отбрасываются, а другие добавляются. Кроме того, после применения операции сложения долгосрочное состояние копируется и пропускается через функцию гиперболического тангенса (tanh), а результат фильтруется выходным шлюзом. В итоге получается краткосрочное состояние $h_{(t)}$ (равное выходу ячейки для временного шага $y_{(t)}$).

Внутри ячейки находится четыре полносвязных слоя:

1) Главный слой, он анализирует текущий вход $x_{(t)}$ и предыдущее (краткосрочное) состояние $h_{(t-1)}$. Аналогичен единственному слою в базовой ячейке, но в ячейке LSTM выход данного слоя $g_{(t)}$ поступает не прямо в $y_{(t)}$ и $h_{(t)}$, а частично сохраняется в долгосрочном состоянии.

2) Оставшиеся три слоя являются контроллерами шлюзов. Поскольку они используют логистическую функцию активации, их выходы находятся в диапазоне от 0 до 1. Их выходы передаются операциям поэлементного умножения, поэтому если они выдают нули, то закрывают шлюз, а если единицы, то открывают его. Они делятся на:

2.1) Шлюз забывания ($f_{(t)}$), который управляет тем, какие элементы долгосрочного состояния должны быть удалены;

2.2) Входной шлюз (i(t)), который управляет тем, какие части g(t) должны быть добавлены к долгосрочному состоянию;

2.3) Выходной шлюз (o(t)), который управляет тем, какие части долгосрочного состояния должны быть прочитаны и выданы на данном временном шаге в h(t) и y(t).

Благодаря такому устройству, ячейка LSTM способна обучаться распознавать вход с важной информацией (входной шлюз), записывать его в долгосрочное состояние, сохранять его столько, сколько необходимо (шлюз забывания), и извлекать его по мере необходимости.

Вычисление долгосрочного состояния ячейки и ее выхода на каждом временном шаге при использовании стохастического градиентного спуска происходит следующим образом (уравнения для мини-пакета градиентного спуска практически неотличимы):

$$i_{(t)} = \sigma\left(W_{xi}^T \cdot x_{(t)} + W_{hi}^T \cdot h_{(t-1)} + b_i\right),$$

$$f_{(t)} = \sigma\left(W_{xf}^T \cdot x_{(t)} + W_{hf}^T \cdot h_{(t-1)} + b_f\right),$$

$$o_{(t)} = \sigma\left(W_{xo}^T \cdot x_{(t)} + W_{ho}^T \cdot h_{(t-1)} + b_o\right),$$

$$g_{(t)} = tanh\left(W_{xg}^T \cdot x_{(t)} + W_{hg}^T \cdot h_{(t-1)} + b_g\right),$$

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)},$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes \tanh\left(c_{(t)}\right),$$

где $W_{xi}$, $W_{xf}$, $W_{xo}$, и $W_{xg}$ – матрицы весов каждого из четырех слоев для их связи с входным вектором x(t);

$W_{hi}$, $W_{hf}$, $W_{ho}$ и $W_{hg}$ – матрицы весов каждого из четырех слоев для их связи с предыдущим краткосрочным состоянием $h_{(t-1)}$;

• $b_i$, $b_f$, $b_o$, и $b_g$ – члены смещения для каждого из четырех слоев.

### 1.3 Распознавание образов с помощью баз знаний

База знаний – это база данных, содержащая правила вывода и информацию о человеческом опыте и знаниях в предметной области [34].

Основное отличие баз знаний (БЗ) от баз данных (БД) заключается прежде всего в хранении и использовании сохраненного контента. Как следует из названия, базы знания хранят в себе знания, а базы данных – данные. И если термин «данные» интерпретируется пользователями и исследователями в целом идентично, то термин «знания» имеет плавающие

границы, зависящие от исследуемой области. В предметной области информатики термин «знание» можно описать как: «вид информации, отражающей опыт специалиста (эксперта) в определенной предметной области, его понимание множества текущих ситуаций и способы перехода от одного описания объекта к другому» [35].

С точки зрения информатики для знания характерны следующие признаки [36]:

1) Внутренняя интерпретируемость. Отдельные элементы некоторой модели могут быть объяснены с помощью самой модели. Для этого в модели обычно выделяет несколько уровней (уровни описания структуры данных, экземпляров данных и т.п.), причем элементы, находящиеся на всех уровнях, должны быть равноправны (часть же важна, как и целое).
2) Структурированность. Структура знания должна определяться смыслом данных или их семантикой, в ее основе лежит иерархия обобщения (от общего к частному) или агрегации (от целого к части).
3) Активность. Знания могут генерировать новые знания, используя модель.
4) Связность. Аналог целостности БД. Представляет собой причинно-следственные связи, закономерности, и иные количественные отношения знаний.

Знания могут быть классифицированы по следующим категориям:

- Поверхностные — знания о видимых взаимосвязях между отдельными событиями и фактами в предметной области.
- Глубинные — абстракции, аналогии, схемы, отображающие структуру и природу процессов, протекающих в предметной области. Эти знания объясняют явления и могут использоваться для прогнозирования поведения объектов.

Современные экспертные системы работают в основном с поверхностными знаниями. Это связано с тем, что на данный момент нет универсальных методик, позволяющих выявлять глубинные структуры знаний и работать с ними.

Система, основанная на знаниях (СОЗ) – это компьютерная программа, которая умеют рассуждать и использовать базу знаний для решения сложных проблем. Такие системы собирают знания из различных источников, затем символизируют их и сохраняют для решения сложных проблем, с которыми предстоит столкнутся в будущем. Общим элементом, который объединяет все СОЗ, является попытка представить знания явно с помощью инструментов, а не неявно с помощью кода, как это делают обычные программы [37]. Инструментами могут быть онтологии и правила, применяемые к фактам. Основная польза от использования таких систем заключается в том, что с их помощью можно подменять/дополнять специалистов в медицинской диагностике, производстве, обучении персонала, мониторинге процессов и т. д.

СОЗ были впервые разработаны исследователями искусственного интеллекта (ИИ). Чаще всего их называют экспертными системами (ЭС). Однако можно отделить СОЗ от ЭС [38]. ЭС используются для решения проблемы совместно с экспертами, тогда как СОЗ относится к архитектуре, которую он представляет в явном виде, т.е. в визуальном представлении, а не в виде машинописного кода. Можно сказать, что все ЭС — это СОЗ, но обратное неверно. Обе системы используют предоставленный или сохраненный набор правил для решения проблем. ЭС зависит от наличия достаточного количества информации для выполнения различных задач, тогда как СОЗ собирает некоторый небольшой объем человеческих знаний, а затем анализирует проблему, чтобы найти решение.

Основными компонентами систем СОЗ являются:

1) База знаний

СОЗ использует свою базу знаний для хранения всех полученных данных о знаниях. Собранные знания от людей или информационных систем, сначала символизируются, а затем сохраняются. Базу знаний нельзя считать типичной базой данных [39]. Она не работает с таблицами, в которых хранятся строки, числа и иногда другие объекты; скорее, она направляет указатели на объекты, которые используют дополнительные указатели. База знаний может быть представлена в виде объектной модели с классами, имеющими подклассы и экземпляры. Ее также можно назвать онтологией. Форма данных, хранящихся в базе знаний, является структурной.

2) Механизм логического вывода

Механизм логического вывода (МЛВ) является важным компонентом СОЗ. Он применяет логические правила к фактам, хранящимся в базе знаний. МЛВ может как создавать новые знания из сохраненных, так и определять, какие знания необходимо сформулировать для решения проблемы. МЛВ работает по типичной структуре правила «IF-THEN». Иными словами, можно сказать, что МЛВ использует принцип дедукции, при котором некоторые утверждения используются для определения результирующего вывода, называемого заключением.

3) Подсистема рассуждений

СОЗ опирается на знания, полученные от экспертов. Иногда сами эксперты узнают что-то новое или вспоминают забытое при работе с СОЗ. Типичная СОЗ должна обеспечивать возможность рассуждений [40]. Должен быть механизм для объяснения внутренней информации и процесса рассуждений. Такое средство помогает пользователям и экспертам управлять знаниями.

Общий алгоритм рассуждений состоит из (а) создания конфликтного набора правил посредством анализа и применения условий и (б) выбора наиболее подходящего правила из конфликтного набора; затем используется стратегия разрешения конфликтов для (в) выполнения этого правила с целью его доказательства или же создания нового правила.

Согласно [41], в рассуждениях можно выделить пять технологий: рассуждения, основанные на правилах; рассуждения, основанные на изложении фактов; рассуждения, основанные на онтологии; рассуждения, основанные на прецедентах; генетические алгоритмы.

4) Пользовательский интерфейс

Пользовательский интерфейс обеспечивает двунаправленную связь между пользователем и системой. Наиболее часто встречаются графические интерфейсы, обеспечивающие удобную работу с системой посредством диалоговых окон, текстовых полей, списков, элементов с активным управлением и т.п. [42]. Также пользовательский интерфейс позволяет вводить в систему информацию о знаниях, задавать системе вопросы и просматривать результаты ответов.

5) Самообучение

Самообучаемой является система, способная самостоятельно выявлять закономерности в информации и собственной организации (поведения) и формировать на их основе собственную уникальную систему знаний, которая может использоваться в дальнейшем функционировании системы. Не всегда разработчикам становится не под силу заложить все возможности внутри системы; в таком случае разрабатывается механизм, позволяющий системе самостоятельно обновлять знания [43].

Основное преимущество самообучения заключается в том, что система зачастую может не запрашивать у пользователя ввод данных и старается находить первичные правила в своей области интереса. Однако, пользователь и администратор такой системы должны быть очень осторожны, результат работы самостоятельного обучения должен быть проверен экспертами [44].

## 1.4 Мультиагентные технологии распознавания образов

Многоагентная система (МАС) – это система, образованная несколькими взаимодействующими интеллектуальными агентами.

Понятие «интеллектуального агента» используется для представления и описания поведения активных объектов в изменяющихся внешних средах (адаптивные, самоорганизующиеся, сетецентрические системы). Такие агенты способны оценивать ситуацию, взаимодействовать с другими агентами, принимать самостоятельные решения в группе интеллектуальных агентов [45]. Именно эта гибкость делает МАС подходящим для решения задач в различных дисциплинах, включая информатику, электротехнику и многие другие [46]. Для разработки МАС требуется решение широкого круга сложных задач, таких как координация между агентами [47], обучение [48] и эффективного распространения информации [49].

В литературе существует множество определений агентов, обусловленных различными особенностями агентов, зависящими от приложения. Авторы [50] определили агента как «гибкую автономную сущность, способную воспринимать окружающую среду через подключенные к ней датчики». Это определение было подтверждено другими исследователями в данной дисциплине, например, [51]. Другая перспектива была представлена в [52], где авторы определили агент как «инкапсулированную вычислительную систему, которая расположена в некоторой среде и способна гибко, автономно действовать в этой среде для достижения цели своего проектирования». Авторы [53] в свою очередь дали довольно общее понятие агентов, которое может широко применяться во многих дисциплинах. Это обобщенное определение, учитывающее фундаментальные способности и особенности агентов, выглядит следующим образом: Агент - объект, который помещается в среду и воспринимает различные параметры, которые используются для принятия решения на

основе цели объекта. Сущность выполняет необходимые действия в среде на основании этого решения.

Затем, авторы [53] расшифровывают ключевые слова, включенные в определение:

1) Сущность: сущность относится к типу агента. Агент может быть программным обеспечением, например агенты безопасности сервера, аппаратный компонент, например термостат или их комбинация, например робот.

2) Среда: среда относится к месту, где находится агент. Среда может быть сетью в случае агентов мониторинга трафика, программным обеспечением, когда агент отслеживает действия программных компонентов и т. д. Агент использует информацию, полученную из среды, для принятия решений.

3) Параметры: различные типы данных, которые агент может воспринимать из среды, называются параметрами. Например, параметрами агента футбольного робота являются положение и скорость членов команды и противников, а также положение мяча.

4) Действие: каждый агент может выполнять действие, которое приводит к некоторым изменениям в среде. Например, когда футбольный робот бьет по мячу, положение мяча меняется. Агент может выполнять набор дискретных или непрерывных действий. В непрерывном наборе действий агент может выполнять неограниченное количество действий, например футбольный матч. Дискретный набор действий, напротив, имеет конечный набор действий, например агент, управляющий термостатом в комнате.

Следующие функции позволяют агентам иметь широкую применимость и решать сложные задачи [54, 55]:

• Коммуникабельность: агенты могут делиться своими знаниями, а также запрашивать информацию у других агентов, чтобы повысить производительность в достижении своих целей.

• Автономность: каждый агент может независимо выполнять процесс принятия решений и предпринимать соответствующие действия.

• Проактивность: каждый агент использует свою историю, измеренные параметры и информацию других агентов, чтобы предсказать возможные будущие действия. Эти прогнозы позволяют агентам предпринимать эффективные действия, которые соответствуют их целям. Эта способность подразумевает, что один и тот же агент может выполнять разные действия при размещении в разных средах.

Хотя агент, работающий сам по себе, может действовать (на основе автономии), реальная выгода агентов может быть использована только тогда, когда они работают совместно с другими агентами, т.е. в составе многоагентной системы (MAC).

Согласно работе [56] среда имеет несколько функций, которые влияют на сложность агентно-ориентированной системы:

• Доступность: доступность означает точность, с которой агенты могут воспринимать данные из окружающей среды. В доступной среде агенты могут получать точные и актуальные данные из среды. Например, агент сетевого брандмауэра может захватывать весь поток трафика сети. Напротив, в недоступной среде агент обнаруживает зашумленные и/или неполные данные. Физический мир, то есть любое событие на Земле, является примером недоступной среды, поскольку датчики, которые собирают данные, вносят собственный шум, который нарушает наблюдаемый (и, следовательно, измеренный) сигнал.

• Детерминизм: это относится к предсказуемости результатов действия. В детерминированной среде результаты предсказуемы, например в среде, которую можно смоделировать с помощью конечного автомата, агент точно знает следующее состояние для каждого действия. В недетерминированных средах результат действия не совсем предсказуем, поскольку на него могут влиять другие факторы, например в игре исход зависит от действий всех участников.

• Динамизм: это относится к изменениям, происходящим в среде, которые не зависят от действий, предпринимаемых агентами. Среда, в которой изменения могут происходить только в результате действия агентов, считается статичной. Во всех остальных случаях среда считается динамической. Напомним, что процесс принятия решений агентами основан на воспринимаемой информации из окружающей среды. Когда окружающая среда меняется, ранее полученная информация может больше не быть точной. Таким образом, в динамической среде агенты должны обнаруживать изменение состояния окружающей среды и обновлять полученную информацию, что влечет за собой больше накладных расходов по сравнению со статической средой, где исходная полученная информация может использоваться для принятия решений в течение срока службы агента.

• Непрерывность: это относится к непрерывности или дискретности среды агента. В соответствии с этой особенностью среда МАС подразделяется на две категории: непрерывная и дискретная. Непрерывная среда влияет на состояние агента через непрерывную функцию, например, агент, который перемещается в физической среде. Дискретная среда вынуждает агента входить в набор заранее определенных состояний, например мобильный агент, который измеряет свое положение с отметкой времени при перемещении в среде.

Авторы [57] считают, что среды в МАС характеризуются:

1) Адаптивностью, т.е. возможностью динамически менять состав элементов, включенных в среду. В парадигме МАС это свойство заключается в том, что количество и тип агентов может изменяться в соответствии с событиями во внешней среде. При этом работа агентов осуществляется путем открытой передачи информации в среду о своих возможностях, а также считыванием информации о возможностях остальных агентов с последующим выбором нужных для достижения собственных целей. В случае изменения состава

среды данные о возможностях агентов актуализируется, что позволяет обновлять схемы решения задач у агентов.

2) Самоорганизацией, под которой понимается обеспечивание максимизации эффективности взаимодействия между отдельными объектами в среде. В МАС взаимодействие агентов в большинстве случаев основывается на правилах, имеющих проблемно-ориентированное происхождение.
3) Концепцией сетецентрической системы, заключающейся в наличии в среде элементов сбора информации (сенсоры), объектов, осуществляющих взаимодействия со средой и между собой (акторы) и информационно-управляющие элементы, осуществляющие анализ ситуации с последующим управлением сенсорами и акторами для решения поставленных задач. В МАС агенты могут являться как представителями сенсоров, акторов или информационно-управляющих элементов, так и их комбинацией. Агентам приписываются соответствующие сценарии и правила принятия решений элементов сетецентрической системы.

Основные характеристики МАС, такие как: эффективность, низкую стоимость, гибкость и надежность, делают его эффективным решением для решения сложных задач. Их эффективность проистекает из разделения труда, присущего МАС, когда сложная задача делится на несколько более мелких задач, каждая из которых назначается отдельному агенту [58]. Естественно, связанные с этим накладные расходы, например, обработка и потребление энергии, амортизируются между несколькими агентами, что часто приводит к более дешевому решению по сравнению с подходом, при котором вся сложная проблема должна решаться одним мощным объектом. Распределенный характер решения задач, принятый в МАС, также обеспечивает высокую надежность. В случае сбоя агента задачу можно легко переназначить другим агентам.

Для изучения МАС агенты и их отношения моделируются с помощью графов. Графы широко используются в информатике для моделирования сложных систем, например распространению болезнетворных микроорганизмов среди живых существ [59] и их математический анализ. Когда МАС моделируются как граф, каждая вершина представляет агента, а ребро между двумя вершинами указывает, что два агента взаимодействуют друг с другом. Действия, предпринятые агентом, могут потенциально изменить отношения между агентами и, таким образом, изменить структуру графа. Окончательное решение, принятое агентом, применяется к соответствующему графу, который может изменить ребра или структуру графа [60].

В МАС агенты обычно имеют только частичную информацию, поскольку агент в основном общается со своими прямыми соседями. С одной стороны, это снижает накладные расходы на связь, что, в свою очередь, обеспечивает масштабируемость, поскольку накладные расходы остаются относительно низкими по мере увеличения количества агентов. С другой стороны, увеличивается время и накладные расходы на связь, связанные с поиском агента, который предоставляет конкретную услугу. Чтобы уменьшить накладные расходы на поиск агента, особенно для крупномасштабных МАС, вводится понятие промежуточных агентов. Агенты среднего звена ведут список услуг, предлагаемых всеми агентами. Любой агент, который ищет конкретную услугу, сначала связывается со средним агентом, который направляет его соответствующему агенту, предлагающему эту услугу [61]. В зависимости от реализации промежуточные агенты можно разделить на:

• Координатор (Facilitator): как показано на рис. 12, координатор действует как посредник между агентом, отправляющим запрос (запрашивающим), и агентом, предоставляющим услугу (запрашиваемым). Координатор направляет запрос соответствующему агенту. Ответ отправляется обратно координатору, который передает его запрашивающей

стороне. Очевидно, что координатор становится узким местом и потенциальной единственной точкой отказа [62]. Чтобы амортизировать эффект центрального координатора, для ответа на запросы используются несколько совместных координаторов [63]. Этот метод требует, чтобы координаторы общались, чтобы оставаться синхронизированными и балансировать нагрузку, то есть запросы, между собой.

• Посредник (Mediator): эта реализация отличается от описанной выше тем, что запрашивающая и запрашивающая агенты могут напрямую взаимодействовать друг с другом, как показано на рис. 13. Это снижает нагрузку на посредника по сравнению с реализацией координатора.

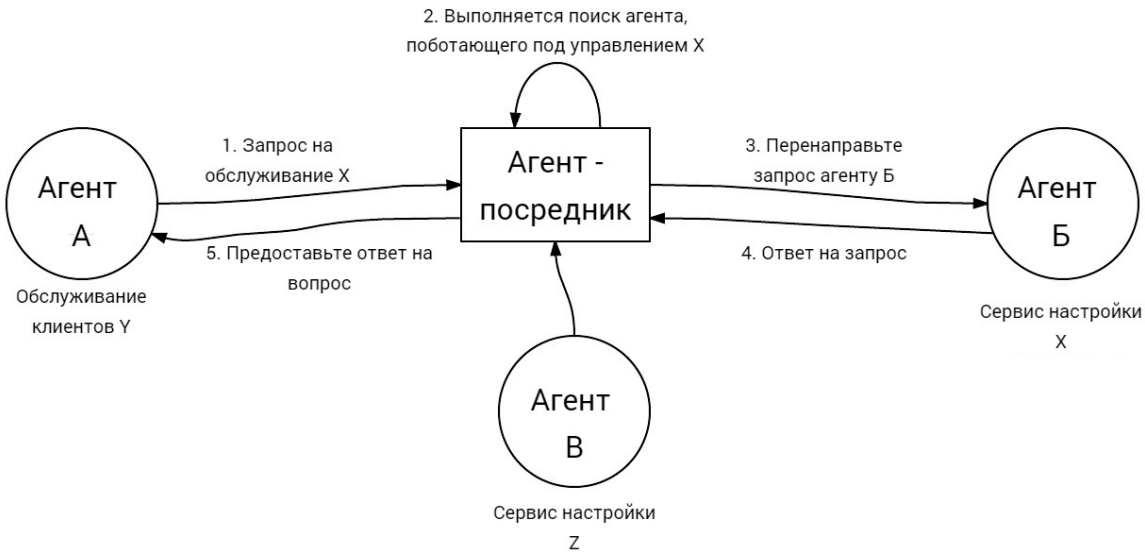


Рис. 12. Вариант исполнения Координатора

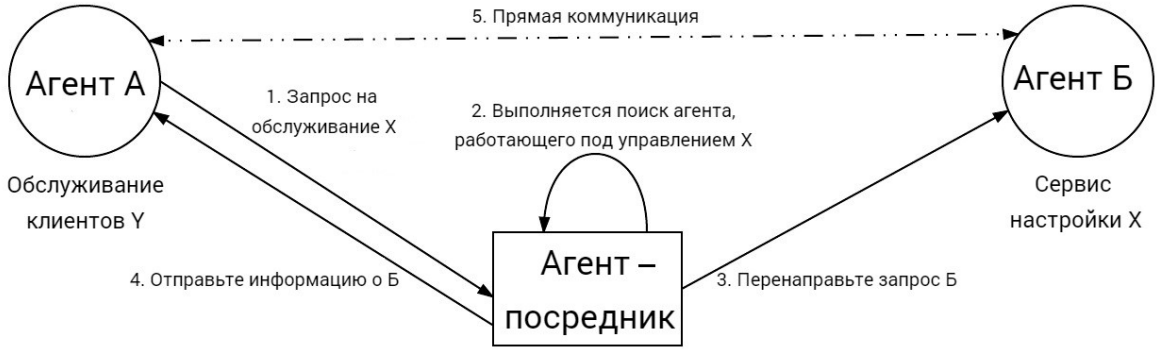


Рис. 13. Вариант исполнения Посредника

Основными особенностями МАС, описанными в работе [64], являются:

1) Руководство

Подразумевает наличие лидера, то есть агента, который определяет цели и задачи для других агентов на основе одной глобальной цели. Присутствие или отсутствие такого лидера можно использовать для классификации МАС как без лидера или как ведомого лидера. В МАС без лидера каждый агент самостоятельно принимает решение о своих действиях, исходя из своих собственных целей. На решение каждого агента влияет решение других агентов, если агенты сотрудничают для достижения консенсуса по определенной функции. Напротив, в режиме «следование за лидером» агент-лидер определяет действия для остальных агентов. Лидер либо заранее определен, либо совместно выбирается агентами [65]. МАС может иметь мобильного лидера или группу агентов, действующих в качестве лидеров. Когда есть несколько лидеров, они совместно направляют последователей, общаясь друг с другом для принятия решений.

2) Функция принятия решения

МАС можно классифицировать на основе пропорциональности изменений выходных данных решающей функции изменениям входных данных. В соответствии с этим МАС делятся на линейные и нелинейные. В линейной МАС решение агента пропорционально воспринимаемым параметрам из окружающей среды, например агент термостата выключает нагреватель, когда температура достигает порогового значения. Эта функция упрощает математический анализ линейных агентов. В нелинейной МАС решение агента не пропорционально измеренным показателям из-за нелинейности входных данных для процесса принятия решения, например, космический корабль должен доставить топливо на космическую станцию, но должен учитывать, что это топливо тратится на собственные нужды во время доставки [66].

3) Гетерогенность

По неоднородности агентов MAC можно разделить на две категории, а именно: однородные и неоднородные. Однородные MAC включают агентов, которые имеют одинаковые характеристики и функциональные возможности, в то время как разнородные MAC включать агентов с разнообразными функциями.

4) Параметры договора

В некоторых приложениях MAC агентам необходимо согласовать определенные параметры, известные как метрики. По количеству показателей MAC классифицируются как MAC первого, второго или более высокого порядка. В MAC первого порядка агенты сотрудничают, чтобы согласовать одну метрику. Например, несколько космических аппаратов двигаются по некоторым траекториям, чтобы исследовать заданный район [67]. В MAC второго порядка агенты должны согласовать две метрики, например, учитывать положение и свободность таксиста для подачи машины клиенту [68]. В MAC высокого порядка [69] соглашение между агентами достигается, когда показатели (один или два) и их производные высокого порядка сходятся к общему значению. Например, в производственной среде под каждый заказ нужно не только находить свободных рабочих и оборудование, но и оптимизировать такой выбор.

5) Рассмотрение задержек

Агенты могут столкнуться с несколькими источниками задержки при выполнении задач. Например, задержка в средствах связи, например беспроводной или проводной, используемый агентами для обмена данными или задержки в планировании ресурсов для каждого агента. В зависимости от того, являются ли задержки значительными и актуальными, MAC можно разделить на две группы: с задержкой или без задержки. Первый учитывает источники задержки, а второй предполагает отсутствие источников задержки. Последний случай довольно упрощен, так как нет задержки связи

и обработки. Однако в большинстве реальных приложений всегда возникают нетривиальные задержки.

6) Топология

Топология относится к расположению и отношениям агентов. Топология MAC может быть статической или динамической. В статической топологии положение и отношения агента остаются неизменными в течение всего времени существования агента. В динамической топологии MAC положение и отношения агента изменяются по мере того, как агент перемещается, покидает или присоединяется к MAC или устанавливает новые коммуникации, то есть отношения, с другими агентами.

7) Частота передачи данных

Агенты воспринимают окружающую среду и делятся полученными данными с другими агентами либо по времени, либо по событию. В первом случае агент непрерывно отслеживает окружающую среду, собирает данные и через заранее определенные интервалы времени отправляет все вновь обнаруженные данные другим агентам. В MAC, инициируемом событием, агент определяет среду только при наступлении определенного события. Затем агент отправляет собранные данные другим агентам.

8) Мобильность

По динамичности агентов можно разделить на статические или мобильные. Статический агент всегда находится в одной и той же позиции в среде, в то время как мобильные агенты могут перемещаться в среде. Мобильный агент может размещаться у других агентов, что означает, что он использует ресурсы других агентов, контролирует их или определяет среду с позиции других агентов для выполнения действий. Например, агент системы обнаружения вторжений перемещается между несколькими серверами (агентами) в сети для анализа серверных процессов и связи и, таким образом, обнаружения атак.

Проблемы MAC обычно связаны с конкретным применением. Тем не менее в подавляющем большинстве приложений возникают следующие ключевые проблемы:

1) Контроль координации

Действие, выполняемое каждым агентом, влияет на среду и, следовательно, на решение, принимаемое другими агентами. Контроль координации относится к управлению агентами для совместного достижения своих целей [70]. В результате координации возникает множество проблем, таких как поиск консенсуса, управляемость, синхронизацию, взаимосвязь и формирование:

1.1) Поиск консенсуса

В MAC консенсус относится к достижению глобального соглашения по конкретной интересующей характеристике. Консенсус широко изучен в литературе [71, 72]. Множество особенностей MAC, перечисленных в разделе 1.4.3, влияют на проблему поиска консенсуса, поскольку они связаны со взаимодействием и сотрудничеством между агентами.

Авторы в [73] установили новый алгоритм поиска консенсуса как для фиксированной, так и для динамической топологии MAC. Авторы математически доказывают, что с помощью своих алгоритмов агенты могут прийти к соглашению по конкретной интересующей особенности. Результаты моделирования показывают, что с течением времени разногласия между агентами по интересующей характеристике уменьшаются до тех пор, пока они, наконец, не достигнут согласия. Авторы также определили новый тип консенсуса, известный как средний консенсус, при котором агенты достигают консенсуса по среднему значению начального состояния конкретной интересующей особенности. В тех случаях, когда поиск уникального параметра для достижения консенсуса агентами либо требует значительных накладных расходов, либо невозможен из-за разнообразия агентов и их характеристик, можно использовать усредненный консенсус для обеспечения согласованности агентов.

Один из хорошо известных примеров консенсуса — это группирование (flocking), которое относится к коллективному поведению ряда агентов для достижения групповой цели [74]. Например, в связи с возрастанием интенсивности трафика воздушного движения и ограниченной пропускной способностью зон воздушного пространства, все больше функций диспетчерского управления передается бортовым средствам самолетов [75]. В одной из новаторских работ по группированию [76] авторы предложили три основных правила, которые приводят к модели группирования:

• Центрирование группы: агенты стараются держаться ближе друг к другу в группе;

• Избегание препятствий: агенты избегают столкновений с ближайшими товарищами по группе;

• Сопоставление скорости: агенты пытаются сопоставить скорость ближайших товарищей по группе.

Эти три правила известны как сплоченность, разделение и согласованность соответственно.

Достижение консенсуса имеет две основные подзадачи, основанные на особенностях лежащей в основе MAC, а именно:

• Отслеживание: в MAC с одним лидером агенты должны достичь консенсуса по позиции лидера. Это гарантирует, что агенты поддерживают связь с лидером. Эта проблема известна как отслеживание [77].

• Сдерживание: Сдерживание аналогично отслеживанию, за исключением того, что у MAC более одного лидера [78]. Лидеры могут ограничивать положение агентов, чтобы контролировать границы своей группы. Кроме того, лидеры могут подключаться друг к другу для обмена данными управления или данными, производимыми агентами, например считываемые данные из окружающей среды.

1.2) Управляемость

Управляемость относится к ситуации, когда МАС можно перевести из начального состояния в определенное состояние с помощью определенных правил [79, 80]. Следовательно, есть уверенность в достижении определенного состояния путем выполнения определенных шагов. Двумя ключевыми метриками, которые влияют на управляемость МАС, являются степень динамизма топологии и степень детерминизма в среде (см. раздел 1.4.2). В динамической МАС топология периодически меняется, влияя на связи между агентами и, следовательно, на их сотрудничество. В недетерминированных средах результаты действий непредсказуемы, поэтому агенты должны принимать решение о новых действиях после наблюдения за результатом своих предыдущих действий, что вызывает задержку и ограничивает активность агентов. В литературе управляемость в основном достигается централизованным образом, когда назначенные лидеры инструктируют последователей для достижения определенной цели. Управляемость имеет множество приложений, включая построение программы полета, грузопотока и расчета ресурсов Международной космической станции [81].

1.3) Синхронизация

Синхронизация означает, что действия, выполняемые каждым агентом, согласованы по времени с другими агентами [49]. Синхронизация имеет тесную связь с проблемой консенсуса, которая хорошо освещена в [82] следующим образом: «Консенсус означает, что агенты достигают соглашения относительно некоторой определенной величины, которая зависит от состояния всех агентов, в то время как синхронизация определяет взаимосвязанное поведение в течении времени между разными агентами». Повышенная неоднородность агентов усложняет синхронизацию. Причина в том, что однородные агенты могут быть синхронизированы по общему признаку, в то время как для гетерогенных агентов синхронизация должна достигаться по нескольким отдельным признакам [83].

1.4) Связность

В некоторых случаях агенты требуют постоянной связи друг с другом, например агенты в MAC с агентом-лидером всегда должны быть связаны с ним. Это требование создает проблему связности, которая выражается в следующем ряде проблем: I) мобильность: мобильность агентов приводит к частым отключениям между агентами и последующему восстановлению новых подключений, II) шумная среда: любое вмешательство в среду может прервать соединение между двумя агентов и, следовательно, требуют повторного установления этих соединений, и III) ограниченное представление топологии MAC: поскольку агенты имеют ограниченный обзор, позиционирование агента и налаживание связей для достижения максимальной эффективности является сложной задачей [84].

В зависимости от типа связности MAC можно разделить на гарантированно связанные и без гарантии подключения. В гарантированно связанной MAC постоянное соединение между агентами всегда присутствует, в то время как в модели без гарантии установления подключения связь между агентами может отсутствовать. Связность широко применяется в распределенных и транспортных системах [85]. В этих случаях положение агентов периодически меняется, и агенты всегда должны поддерживать связь друг с другом.

1.5) Порядок

В некоторых случаях может быть необходимо, чтобы группа агентов в MAC была организована в определенную структуру, и чтобы эта структура поддерживалась в течение определенного периода времени (который может быть даже на протяжении всего срока службы MAC). Например, группу безымянных летательных аппаратов (БПЛА) можно попросить сформировать определенный строй, чтобы найти конкретный элемент в окружающей среде или измерить несколько параметров окружающей среды. Эта проблема известна как проблема управления формированием в литературе [86, 87]. Три основных шага в формировании структуры: 1) поиск наиболее эффективной структуры для применения ко всем агентам, 2) организация агентов на

основе определенной структуры и 3) поддержание определенной структуры в течение определенного времени. Неоднородность агентов, ограниченный взгляд на окружающую среду и динамичность МАС или окружающей среды делают описанные шаги очень сложными [88].

В работе [89] предложен метод формирования с учетом позиции, при котором центральный агент, который знает положение всех агентов, контролирует структуру. Формирование структуры имеет широкий спектр применения в вооруженных силах, при решении задач дистанционного зондирования земли, управлении легковыми и грузовыми перевозками [61, 67, 68, 90].

2) Обучение

В МАС каждый агент самостоятельно принимает решение о соответствующем действии для достижения своей цели на основе нескольких показателей. Агенты могут использовать алгоритмы машинного обучения [91] для обнаружения и прогнозирования изменений в окружающей среде и адаптации к новым ситуациям [92]. Следующие проблемы усложняют внедрение обучающих систем для МАС [48, 93, 94]: I) накладные расходы на обработку и связь методов обучения, которые потребляют ресурсы агентов, II) среда МАС может быть динамической. Таким образом, агенты должны часто воспринимать обновленную информацию, которая будет использоваться методом машинного обучения, который, в свою очередь, потребляет значительный объем ресурсов агента; III) топология МАС может измениться, что потребует повторного соединения с соседними агентами; IV) защита агентов от злонамеренных агентов. которые вводят ложную информацию, и V) масштабируемость метода обучения для крупномасштабных МАС.

В системах многоагентного обучения используются два основных метода машинного обучения: обучение с подкреплением и генетическое программирование. Обучение с подкреплением — это метод проб и ошибок, при котором каждый агент меняет свое состояние и наблюдает за

вознаграждением или штрафом, которые он получает от окружающей среды или других агентов [48, 93, 95]. Каждый агент многократно использует действия с положительными эффектами, избегая при этом повторения действий с отрицательными эффектами [94]. При обучении с подкреплением агенты не имеют заранее определенных знаний или политик в отношении среды. В литературе существуют различные методы обучения с подкреплением для агентов, такие как Q-обучение, Minimax-Q, Q-обучение друг или враг и Q-обучение Нэша [96]. Второй широко используемый метод обучения в системах многоагентного обучения — это генетическое программирование, которое является разновидностью эволюционного алгоритма в машинном обучении. При генетическом программировании несколько компьютерных программ кодируются как набор генов, которые развиваются с помощью эволюционного алгоритма [97]. По прошествии времени только самые эффективные гены выживают и конкурируют с другими генами, чтобы приблизиться к требуемому решению.

3) Обнаружение неисправностей

Обнаружение и изоляция неисправных агентов - фундаментальная задача, поскольку неисправный агент может поразить других агентов, с которыми он сотрудничает [98]. Современные методы обнаружения и изоляции сбоев (ОИС) в основном централизованы, когда центральный агент стремится обнаруживать, а затем изолировать неисправных агентов [99]. Централизованные методы являются неоптимальными для крупномасштабных и распределенных систем, таких как МАС. Это связано с типичными проблемами, которые мешают централизованным подходам, включая единую точку отказа и возможность перегрузки одного агента (если многие агенты отправляют запросы этому агенту). Следовательно, обнаружение и изоляция сбоев требует распределенных решений, в которых все агенты обмениваются данными для обнаружения и изоляции неисправных узлов [100].

4) Распределение задач

Проблема заключается в распределении задач между агентами с учетом связанных затрат, времени и накладных расходов на связь и обработку информации. Распределение задач может быть централизованным или децентрализованным [101]. Авторы [102] предлагают гибридный подход, объединяя агентскую систему в несколько кластеров. В каждом кластере один узел (известный как глава кластера) распределяет задачи между членами кластера.

Выделяют две основные метрики для распределения задач между агентами:

• Ресурсы агента: данная метрика относится к общему количеству ресурсов каждого агента. Агентам назначаются задачи пропорционально их ресурсам. Однако очень важно учитывать текущую нагрузку на агента, прежде чем назначать ему новую задачу. Если задачи, назначенные агенту, превышают ресурсы агента (т. е. агент перегружен), то задержка получения ответа от агента значительно увеличится. Чтобы предотвратить перегрузку, используется балансировка нагрузки для равномерного распределения нагрузки между агентами.

• Позиция агента: позиция агента влияет на задержку связи, а также на накладные расходы (например, количество пакетов, которые необходимо передать для связи с другими агентами) [101, 103]. Таким образом, при назначении задач следует учитывать позицию агента, чтобы уменьшить накладные расходы. Например, если задача X требует ресурсов, принадлежащих агентам B и C, то предпочтительно выделить X агенту, который находится рядом с B и C [103].

Распределение задач имеет разнообразные приложения, включая распределение задач по распределению машин таксопарка в режиме реального времени [85].

5) Позиционирование

У каждого агента есть ограниченное представление (только его соседи) топологии MAC. При таком ограниченном представлении найти

конкретного агента, то локализовать его, может быть непросто. Агент может быть найден на основе: I) наличия определенных ресурсов, II) выполнения определенных служб или III) владения определенной идентичностью. Большинство существующих в литературе методов позиционирования централизованы. Однако централизованные методы не обязательно будут масштабироваться для крупномасштабных MAC, что увеличивает потребность в распределенных решениях [104]. В распределенном решении агенты обмениваются информацией о своем соседстве, например, функциями, услугами или идентификаторами, с другими соседями. Следовательно, они могут коллективно построить глобальную топологию для позиционирования любого агента. Однако это увеличит накладные расходы на связь между агентами. Кроме того, произойдет значительная задержка в обновлении информации об агентах в MAC. Хотя гибридная топология может быть необходима для баланса накладных расходов, централизации и распределения, определение оптимального уровня распределения для позиционирования остается открытым вопросом.

Динамичность агентов явно влияет на задачу позиционирования. Локализация динамических агентов требует больше коммуникационных и вычислительных ресурсов по сравнению со статическими агентами. Агенты могут позиционировать динамического агента, оценивая его будущее положение (или состояние), используя его текущую скорость или направление движения. Это известно как оценка состояния [105]. Используя оценку состояния, агенты могут непрерывно следить за мобильным агентом, что потенциально может преодолеть проблему отслеживания.

6) Связь между агентами

Основные подходы к коммуникации включают:

• Передача сообщений: в этом методе агенты напрямую обмениваются сообщениями друг с другом. Агенты используют связь "точка-точка" или широковещательную связь для общения с другими агентами. В первом случае агент А может напрямую поговорить с агентом Б, если он знает адрес

Б. В модели широковещательной связи агент А отправляет сообщение всем своим соседям. Чтобы обеспечить интерпретируемость сообщения, агенты в сообщении должны использовать согласованную структуру.

• Черная доска (Blackboard): при этом способе связи агенты могут совместно обмениваться данными друг с другом, используя центральное хранилище, называемое Blackboard. Каждый агент хранит свои данные на доске, доступной для чтения другими агентами. Для управления доступом агентов к хранящимся на ней знаниям доска использует управление знаниями.

Семантика сообщений довольно важна для обеспечения того, чтобы агенты, взаимодействующие друг с другом, имели одинаковую интерпретацию передаваемых данных. Это особенно сложно с гетерогенными агентами. Простой пример - когда агент А посылает агенту В температуру 12°C, тогда температура не должна интерпретироваться агентом В как 12°F. Язык общения агентов (ACL) направлен на решение вышеупомянутой проблемы. ACL предоставляет уникальный формат сообщений и онтологию для всех агентов для обмена данными и интерпретации полученных сообщений. Авторы [106] разделили ACL на две основные категории: процедурные и декларативные. В процедурных ACL связь между агентами моделируется как обмен процедурными директивами. В декларативных ACL декларативные операторы используются для указания определений, предположений и утверждений.

### 1.5 Выводы по Первой главе

Современные разработки в области искусственного интеллекта предоставляют широкие возможности в различных сферах человеческой деятельности. Наиболее значимые результаты достигаются в области технологий компьютерного зрения и распознавания образов на основе искусственных нейронных сетей, направленных на развитие существующих

приложений новыми возможностями сбора, обработки и управления данными.

Требование одновременного учета большого количества признаков в данных приводит к необходимости перехода от простого поиска и статистического анализа данных к более сложному интеллектуальному анализу. На сегодняшний день в качестве наиболее популярного подхода к такому анализу используется подход на основе мультиагентных технологий. Каждый агент вносит свой вклад в общее дело, решая некоторую часть большой задачи. Если задача сложная, то агентов может потребоваться большое количество, либо же придется использовать более сложных, комплексных агентов, например, использующих нейронные сети. Такую задачу не получается решить на одном высокопроизводительном устройстве, и необходимо строить системы распределенных вычислений. Координацией агентов в распределенной сетевой среде обычно занимается специальный агент – агент-координатор (Faciliator).

Основными моделями поведения агента-координатора для распределения задач и выбора агентов в многоагентной среде являются: аукцион (выбор агентов осуществляется после опроса/торга), набор правил (агенты выбираются автоматически на основании правил) и использование всех доступных агентов (выбираются все доступные агенты, из их результатов отбираются лучшие). Перспективной является реализация новых моделей поведения на основе нейросетевых технологий. Такие модели могут быть основаны на многослойных перцептронах. Но более перспективным является применение рекуррентных нейронных сетей (Recurrent neural network, RNN) с использованием управляемые рекуррентных блоков (Gated Recurrent Units, GRU), так как блоки GRU позволяют более эффективно обрабатывать длинные последовательности с информацией, чем классическая реализация RNN. Применение рекуррентных нейронных сетей позволяет агенту-координатору находить скрытые связи в поступающих данных и использовать их для решения своих задач.

Однако, несмотря на прогресс в области построения распределенной обработки данных с помощью комбинированной реализации различных нейронных сетей, проблема динамического ансамблирования нейронных сетей в распределенной системе искусственного интеллекта в настоящее время не решена.

# ГЛАВА 2. МЕТОД МУЛЬТИАГЕНТНОГО АНСАМБЛИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ КОМПОНЕНТОВ АДАПТИВНОЙ СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ

## 2.1 Обобщенная постановка задачи мультиагентного распознавания образов

На сегодняшний день в качестве наиболее популярного подхода к распределенной обработки данным используется подход на основе многоагентных систем. Каждый агент вносит свой вклад в общее дело, решая некоторую часть большой задачи. Если задача сложная, то агентов может потребоваться большое количество, либо же придется использовать более сложных, комплексных агентов, например, использующих нейронные сети. Такую задачу, вероятно, не получится решить на одном даже высокопроизводительном устройстве и необходимо будет распределять агентов по некоторой сетевой среде вычислительных устройств, что приведет к удорожанию решения.

Координацией агентов в распределенной сетевой среде занимается специальный агент – агент-координатор (Faciliator). Для решения задачи координации предлагается метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов, отличающийся от аналогов реализацией динамического комплексирования автономных искусственных нейронных сетей, позволяющий обеспечить адаптивность системы в условиях изменяющейся обстановки без переобучения интеллектуальных компонентов.

Специфика мультиагентного ансамблирования интеллектуальных компонентов распределенной системы распознавания образов состоит в обеспечении автономного активного поведения отдельных программных

агентов, нацеленных на идентификацию и классификацию встречающихся объектов. В качестве типовых примеров таких объектов можно привести образы предметов или событий в системах компьютерного зрения или семантические конструкции в системах текстопонимания. Такой объект может существовать в реальности, быть представлен (изображен) определенным образом в виртуальной реальности или описан с помощью некоторого шаблона последовательности действий или наблюдений.

Под объектом $c_n, n = \overline{1,N}$, будем понимать любой объект, как реальный, существующий в действительности, так и виртуальный, например, показание датчика в виде числа или логическое суждение. Каждый объект $c_n$ характеризуется рядом признаков $w_{nm}, m = \overline{1,M}$, при этом сами признаки могут быть как числовыми, так и категориальными. Предполагаем, что информацию $D$, по которой возможно обнаружить объекты, можно получить посредством датчиков или же принять по сети из внешней системы. Для нахождения объектов по полученной информации введем агенты $a_l, l = \overline{1,L}$, где каждый агент $a_l$ на основе полученной информации $D$ будет находить объекты $b_i, i = \overline{1,I}$, с признаками $w_{ij}, j = \overline{1,J}$, с вероятностью $p_i \in [0,1]$ .

Введем функцию идентификации объектов Id:

$$Id(b_1, c_2) = \begin{cases} 1, \text{если } p(b_1, c_2) \geq p^* \\ 0, \text{если } p(b_1, c_2) < p^* \end{cases} \quad (2.1)$$

где $p^*$ - пограничное значение вероятности,

$p(b_1, c_2)$ – вероятность, полученная от агента, что опознанный объект b1 является реальным объектов c2.

Для признаков функция идентификации выглядит следующим образом:

$$Id(w_{1j}, w_{2j}) = Id(b_1, c_2) \quad (2.2)$$

Тогда для агентов должны выполняться условия:

$$\sum_l \sum_{in} Id(b_i, c_n) * (1 - \delta(b_i, c_n)) \to min \quad (2.3)$$

$$\sum_l \sum_{in} \delta(b_i, c_n) * (1 - Id(b_i, c_n)) \rightarrow min \quad (2.4)$$

$$\sum_l T(a_l) \rightarrow min \quad (2.5)$$

где δ – дельта функция:

$$\delta(x, y) = \begin{cases} 1, \text{если } x = y \\ 0, \text{если } x \neq y' \end{cases}$$

$T(a_l)$ – время работы агента $a_l$, если агент $a_l$ не задействован, $T(a_l) = 0$.

Под условием (2.3) понимается требование по минимизации ошибки первого рода, когда агенты находят неверные объекты. Под условием (2.4) понимается требование по минимизации ошибки второго рода, когда агенты не находят верные объекты. Под условием (2.5) понимается требование по минимизации времени работы агентов.

## 2.2 Описание вариантов мультиагентной инфраструктуры

При организации агентов существует ряд способов связи и соединений между агентами. В определенных подходах, которые описаны ниже, агенты могут быть сгруппированы на основе определенных характеристик, таких как цели, ресурсы и услуги [107]. Распространенные подходы к организации МАС [55, 65, 108]:

• Плоская: это самая основная организационная структура, в которой все агенты считаются равными. Нет назначенного лидера, и каждый агент общается со своими соседями. Данный вариант организации иллюстрируется на рис. 14.

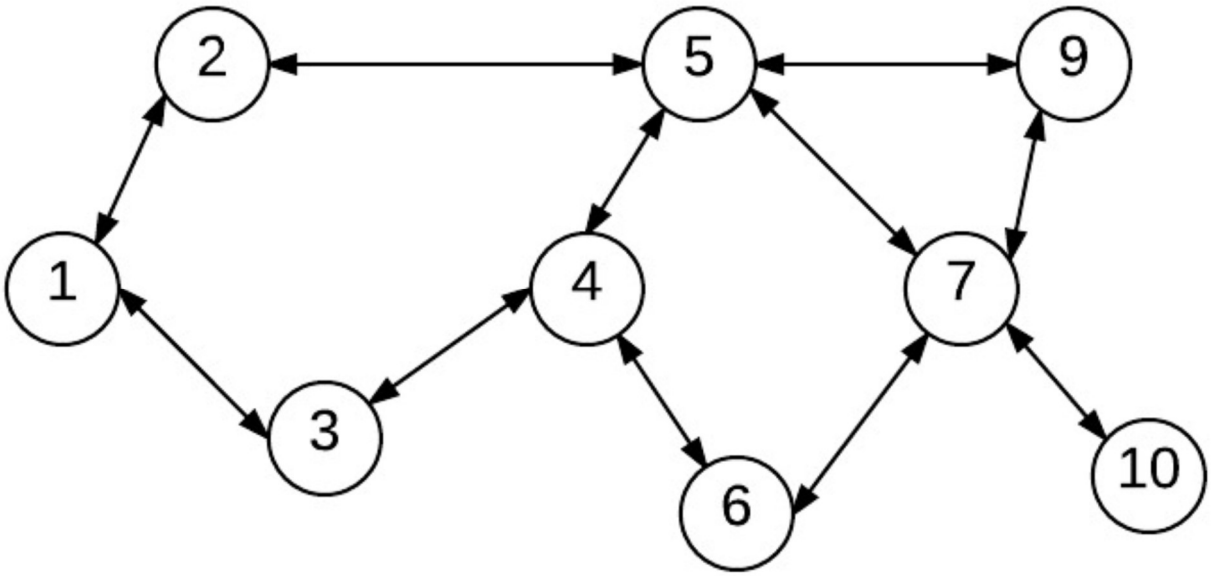


Рис. 14. Плоская организация MAC

При организации агентов в виде плоской структуры для работы модели достаточно выполнения условий (3 – 5).

• Иерархическая: В иерархической организации агенты имеют древовидные отношения, как показано на рис. 15. Родители контролируют своих дочерних агентов, то есть детей, и могут иметь своих собственных родителей. На самом высоком уровне есть один агент, известный как корневой агент (агент A). Иерархическая организация может привести к задержке или созданию узких мест, особенно у корневого агента (или родителей), поскольку он отвечает за обработку сообщений всех дочерних агентов. В зависимости от количества родительских агентов, то есть тех, кто контролирует других агентов, иерархическая организация может быть разделена на два типа: простая и единообразная. В простом подходе корневой агент имеет исключительные полномочия и контролирует все коммуникации. При единообразном подходе в иерархии имеется более одного родительского агента, что означает, что в дополнение к корню все или отдельные родители могут также иметь своих детей.

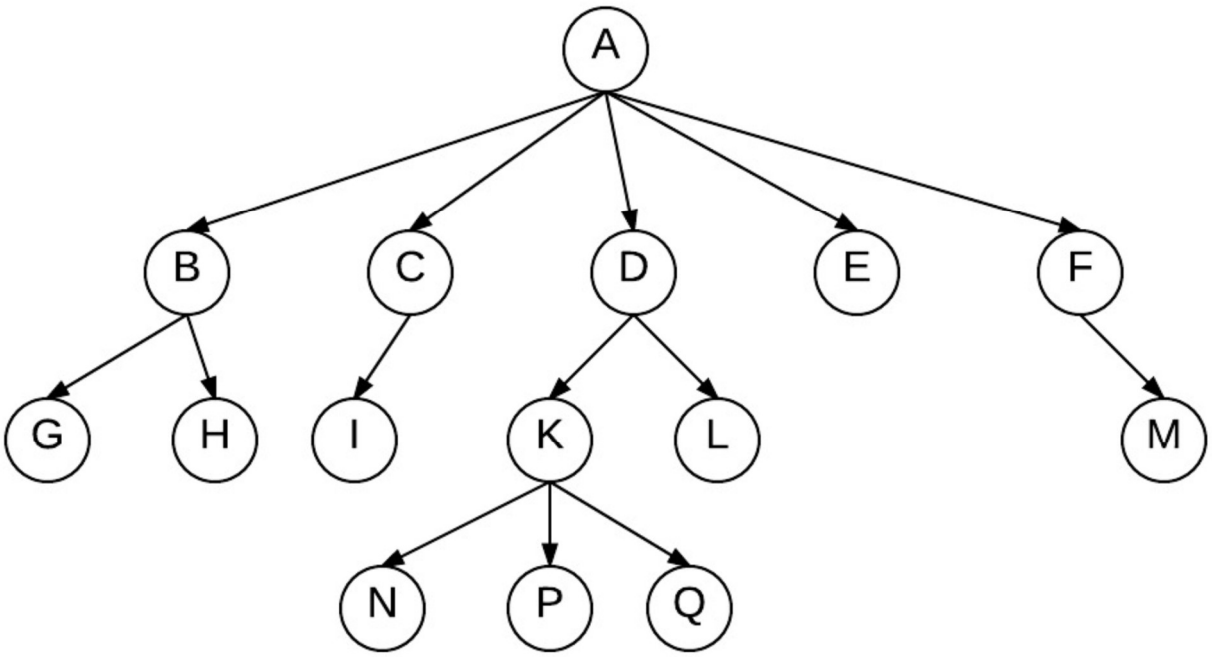


Рисунок 15. Иерархическая организация МАС

При иерархической организации агентов выделяется 3 стратегии выбора агентов: аукцион, автоматическая диспетчеризация и конкуренция:

1) Аукцион — это опрос агентов с целью выяснить, кто из них способен обработать данные. На основе результатов опроса родительский агент выбирает дочерних агентов для обработки данных;

Для организации аукциона дочерние агенты должны иметь возможность на основе входящих данных выдать вероятность их успешной обработки. Реализация такой функции у агентов будет зависеть от входящих данных и задач, стоящих перед агентом. В обобщенном виде метод по анализу входящих данных выглядит следующим образом:

$$P_{a_l} = f_{a_l}(D),\ P_{a_l} \in [0, 1],$$

где $f_{a_l}$ – функция подсчета вероятность положительного распознавания агентом объектов в данных $D$.

По итогам аукциона для обработки данных выбираются агенты с максимальным значением метрики $P_{a_l}$.

2) Если применяется конкурентная стратегия, то в обработку данных вовлекаются все доступные дочерние агенты, а по результатам обработки отбираются лучшие. Оценка агентов производится согласно метрике $P_{a_l}$:

$$P_{a_l} = \sum_{i=1}^{I}(p_i - p^*).$$

По итогам работ выбирается результат работы одного или нескольких агентов с максимальным значением метрики. Агенты со значением $P_{a_l} \leq 0$ не участвуют в выборе результатов.

3) При автоматической диспетчеризации родительский агент самостоятельно выбирает, каких дочерних агентов предпочтительнее использовать для обработки данных;

Выбор основывается на следующих данных:

- история применения агентов в процессе обработки данных при использовании конкурентной стратегии из п.2 в виде таблицы 1. использования агентов:

Таблица 1

| Итерация (или время) | Данные | Примененные агенты |
|---|---|---|
| 0 | $D_0$ | - |
| 1 | $D_1$ | $a_1$, $a_2$, … $a_k$ |
| … | … | … |
| K | $D_K$ | $a_1$, $a_3$ |
| … | … | … |
| $N_{текущ}$ | $D_{Nтекущ}$ | $a_2$, $a_3$, … $a_{k'}$ |

- соответствие паттернов в данных к результирующем выбору агента:

$$F'(D) \rightarrow \{a_n, a_m, \dots a_k\},$$

где F'(D) – функция выбора агента на основе паттернов в данных D.

Формирование соответствия паттернов выбранным агентам, т.е. функции F'(D), может назначаться как полуавтоматически с использованием частотного анализа для последующего создания конструкций вида «if-else», так и автоматически, с использованием нейронных сетей или моделей машинного обучения.

Для частотного анализа из данных D выделяются паттерны PD. Затем на основе значений категориальных паттернов или на основе интервалов значений числовых паттернов строятся гистограммы, в которых по оси y указывается наиболее часто выбираемый агент $a_{\text{выбр}}$ и процент выбора $p^{a_{\text{выбр}}}$ данного агента среди остальных других, см. рис. 16. Для случая, когда выбор агента не производился, вводится агент $a_{null}$. У каждого агента al подсчитывается частота p_al его выбора на основе паттернов PD:

$$p_{a_l} = \sum_{PD} \delta\left(a_l, a_{\text{выбр}}\right) \cdot p^{a_{\text{выбр}}}.$$

В итоге выбирается агент/агенты с наибольшим значением $p_{a_l}$. Если выбран агент $a_{null}$, то выбор агента не производится. Данное поведение лежит в основе функции F'(D).

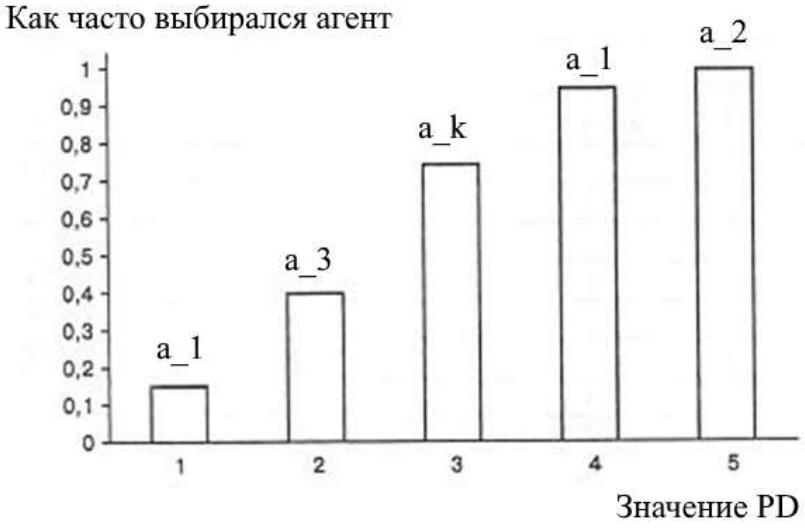


Рис. 16. Пример гистограммы выбора агента в зависимости от значения PD

• Организация в виде холонов:

В такой организации агенты организованы в несколько групп, которые известны как холоны, на основе определенных характеристик, например, неоднородности или сенсорной способности агентов в холоне. Затем холоны накладываются на несколько слоев, как показано на рис. 17. Агенты могут общаться с другими агентами в том же холоне или в других холонах того же уровня. Таким образом, в холонической организации агент может быть членом более чем одного холона в одном слое. Для связи верхнего уровня используется главный агент, который выбирается среди агентов с наибольшим количеством доступных ресурсов в холоне. Эта организация подходит для MAC, когда каждый верхний агент (член холона на верхнем уровне) требует, чтобы субагенты (члены холона нижнего уровня) совместно решали конкретную задачу. У каждого субагента могут быть аналогичные требования к нижним слоям.

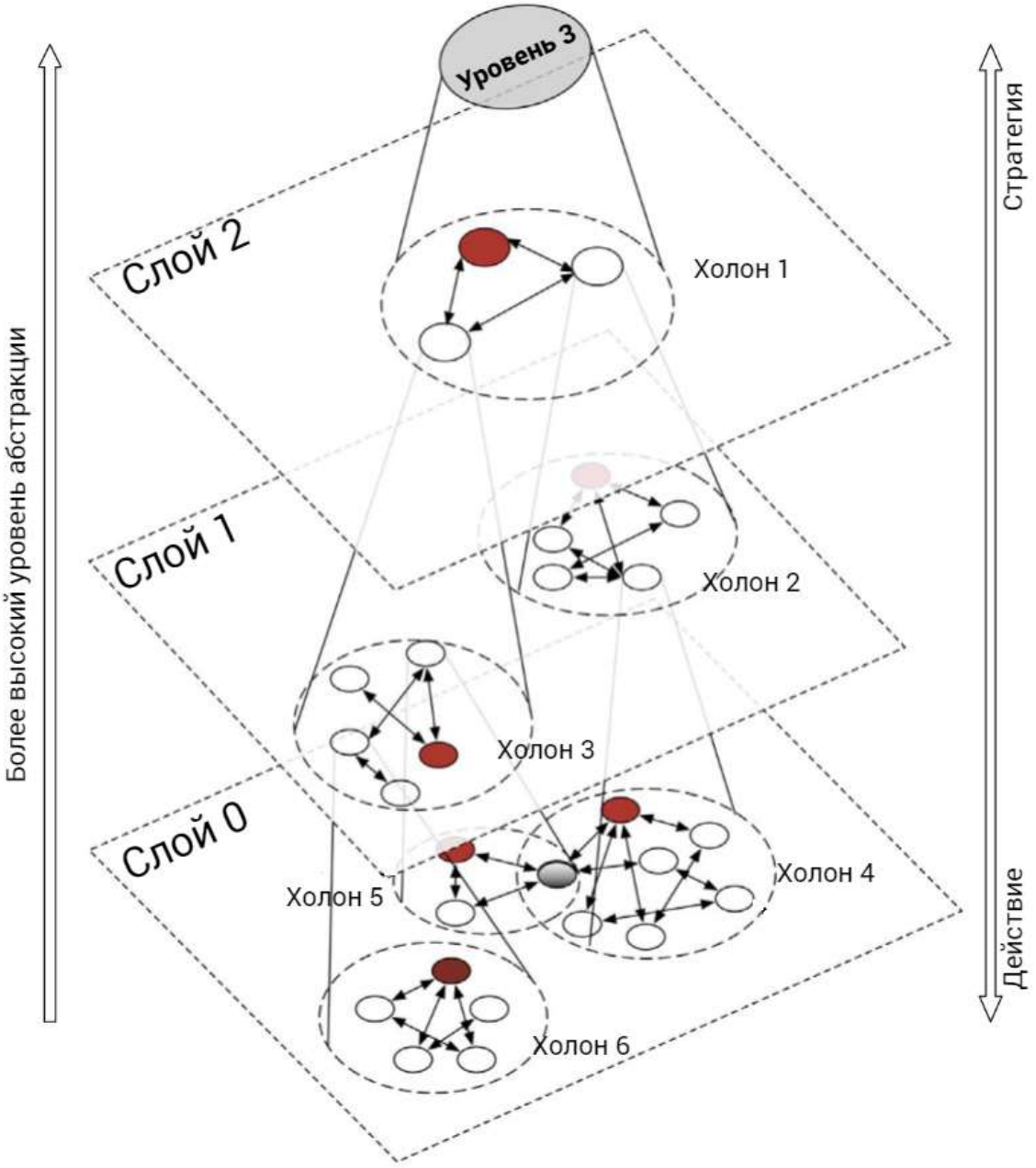


Рис. 17. Организация МАС в виде холонов

Условия (2.3 – 2.5) действуют как на верхнеуровневую структуру агентов, так и на структуру каждого субагента с нижнего уровня. При развитой организации агентов в виде холонов решение условий (2.3 – 2.5) может занимать продолжительное время. В такому случае допускается замена условий (2.3) и (2.4). Для этого введем новые требования по минимизации ошибок 1-го и 2го рода у субагентов $a_{l'}$:

$$\sum_{l'}\left(\sum_{in} I(b_i, c_n) * (1 - \delta(b_i, c_n))\right)^2 \rightarrow min \qquad (2.6)$$

$$\sum_{l'}(\sum_{in}\delta(b_i, c_n) * (1 - I(b_i, c_n)))^2 \to min \quad (2.7)$$

Как можно увидеть, основное отличие условий (2.6) и (2.7) от (2.3) и (2.4) заключается в наличии возведения в квадрат значений потерь каждого субагента. Благодаря этому корректировке подвергаются в первую очередь наименее точные дочерние агенты, что позволяет быстро улучшить точность и полноту родительского агента и заранее остановить обучение дочерних агентов при достижении требуемых показателей. Тем не менее такой подход не лишен недостатков: наиболее точные агенты могут не успеть достигнуть наивысших показателей точности.

• Комбинационная:

В комбинационной организации агенты временно группируются в зависимости от их целей. Следовательно, агенты могут достичь своей цели с меньшими накладными расходами (задержка обработки и связь) по сравнению с организацией, в которой нет такой группировки. На рис. 18 показана коалиционная организация. Каждый агент может быть частью более чем одной коалиции, например, агент №7. Достигнув своей цели, агенты разрушают созданную коалицию. Внутренняя организация коалиции обычно плоская; однако другие организации, например иерархическая организация, могут использоваться для дальнейшего уменьшения накладных расходов.

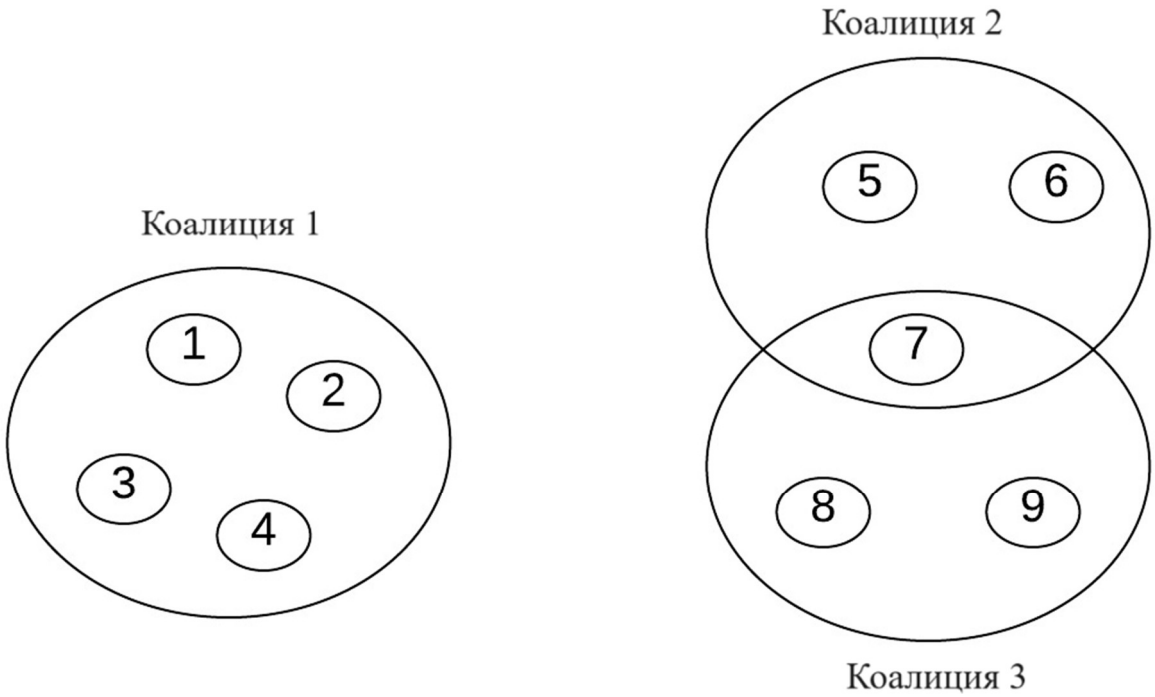


Рис. 18. Комбинационная организация МАС

Таким образом, существует компромисс между уменьшением накладных расходов в результате комбинирования и понесенными накладными расходами на поиск агентов с той же целью и формирование их в коалицию. Эта организация подходит, когда в МАС существует группа агентов с похожими целями, которые объединяют их сотрудничество в достижении их цели. Например, группа машин (агентов) скорой помощи может собраться вместе, чтобы спасти людей во время землетрясения. Формирование коалиции помогает машинам скорой помощи эффективно достичь своей цели, поскольку она распределяет их справедливо, чтобы покрыть более широкую территорию и, таким образом, спасти больше раненых.

Помимо условий (2.3 – 2.5) в зависимости от задачи могут быть применены дополнительные условия, вида:

$$\forall\, l_1 \neq l_2, \forall b_i \in a_{l1}, \forall b_j \in a_{l2}: b_i \neq b_j \qquad (2.8)$$

Условие (2.8) гласит, что каждый агент в коалиции идентифицирует только свой класс объектов, т.е. подмножества объектов, определяемые агентами, не должны пересекаются. Возвращаясь к примеру со скорой помощью, условие (2.8) можно трактовать как: «каждая машина должна обслуживать свою территорию и не пересекаться с другими машинами».

• Командная:

В командной организации агенты создают группу (команду) и определяют групповую цель, которая отличается от их собственной цели. В зависимости от времени, необходимого для достижения цели команды, команда может быть краткосрочной или долгосрочной. Агенты в команде сотрудничают для достижения цели команды. На рис. 19 изображена командная организация. Цель команды может быть обновлена, что приведет к изменению обязанностей, ролей и полномочий агентов в команде. Каждая команда может запрашивать информацию у агентов в других командах, чтобы улучшить свой собственный процесс принятия решений. Команда может иметь внутреннюю организацию (например, иерархическую) для повышения производительности и эффективности в достижении командных целей. Например, в группе агентов, ответственных за анализ городского трафика, агенты создают иерархическую структуру, так что данные трафика могут быть интегрированы родителями для уменьшения накладных расходов на связь.

Количество агентов в команде (известное как размер команды) - одна из ключевых проблем в организации команды. Большая команда может воспринимать больше данных из окружающей среды; однако объединение данных и знаний нескольких агентов требует высокой обработки. Окончательное решение команды менее сложно в небольших группах, однако данные, используемые небольшими командами, ограничены. При выборе размера команды следует учитывать компромисс между накладными расходами на принятие решений и точностью данных.

В отличие от комбинирования, где агенты группируются для достижения своей цели, в команде агенты пытаются достичь цели команды. Таким образом, эта организация подходит, когда несколько агентов пытаются достичь одной и той же цели.

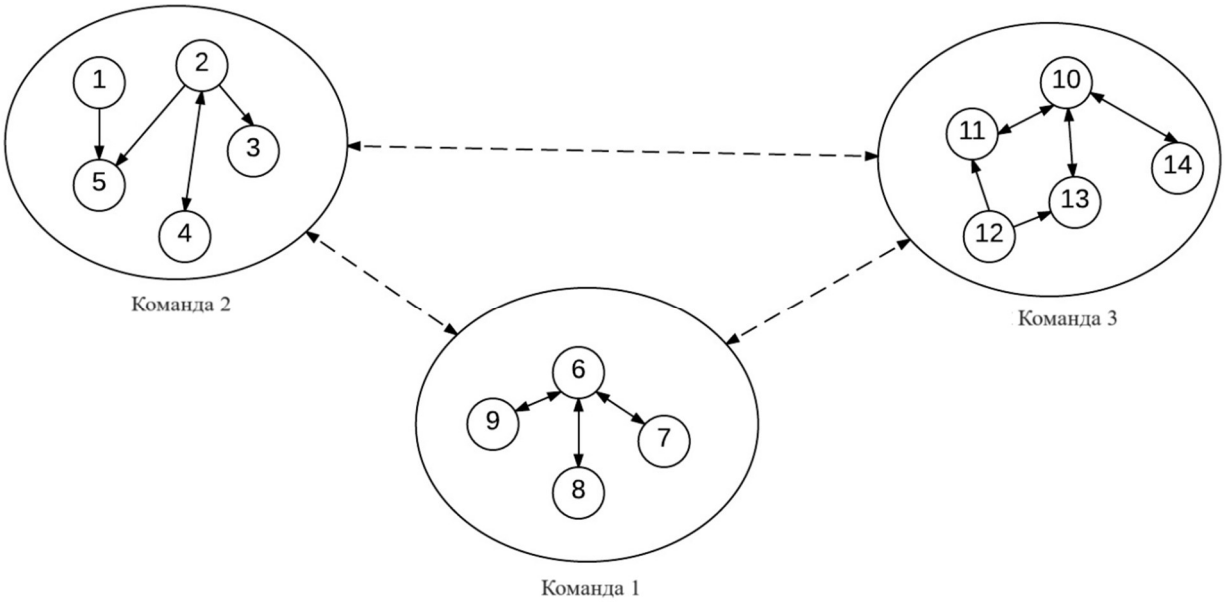


Рис. 19. Командная организация MAC

Внутри команды $A_L$ у агентов $a_l$, $l \in L$, цели, т.е. определяемые объекты $b$, могут быть одинаковые, но среди команд цели должны отличаться, для этого к командам, по аналогии с агентами, применяется условие (2.6). Объекты у агентов в группе должны или совпадать с объектами команды, или же совпадать с признаками объектов команды, или помогать другим членам команды находить их объекты.

Размер команды – это компромисс между точностью и тем, сколько агентов останется для формирования других команд, для его нахождения вводится условие:

$$\sum_{l \in L} \sum_{in} \left( \delta(b_i, c_n) \cdot \left(1 - I(b_i, c_n)\right) + I(b_i, c_n) \cdot \left(1 - \delta(b_i, c_n)\right) \right) \cdot len(L) \rightarrow min,$$

где *len(L)* – количество используемых агентов в команде.

• Матричная:

В матричной организации каждый агент администрируется, то есть управляется, по крайней мере, двумя главными агентами (известными как менеджеры), как показано на рис. 20. Примером, где можно использовать эту организацию, является компания, где каждый сотрудник отчитывается перед менеджером по продукту (который контролирует продукт) и функциональным менеджером (который контролирует функции задачи). Эта организация эффективна, когда агенты контролируются более чем одним лидером (или менеджером).

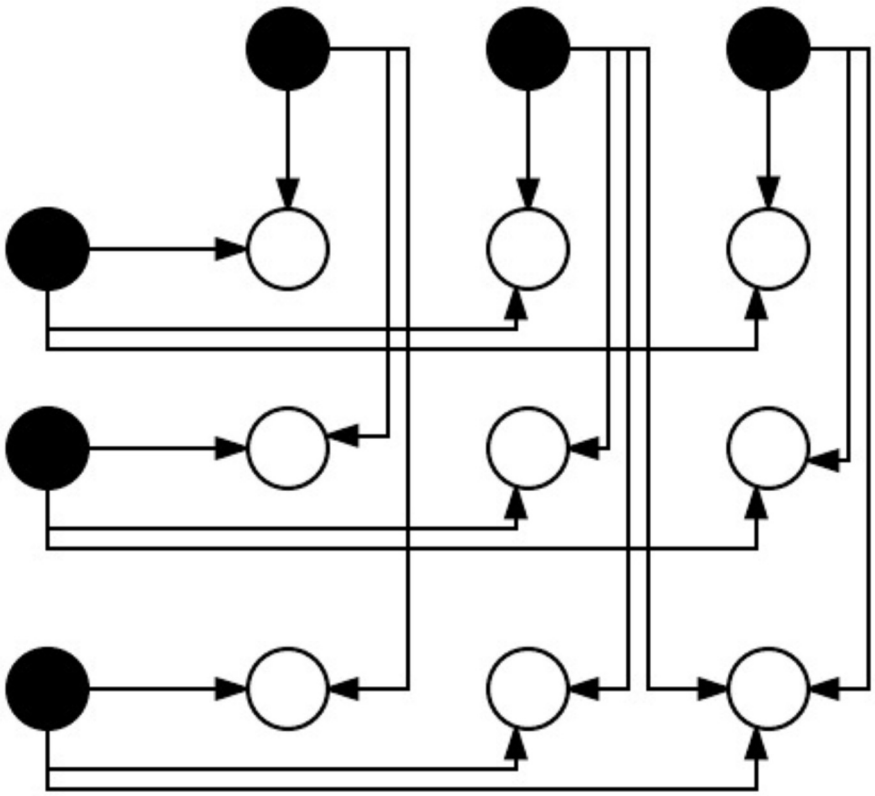

Рис. 20. Матричная организация МАС

Тот из менеджеров, что контролирует задачи, выбирает исполнителей по той же логике, что и в иерархической организации агентов выделяется: аукцион, автоматическая диспетчеризация и конкуренция. Второй менеджер контролирует результат выданного задания. В обязанности второго менеджера входит проверка списков полученных объектов $\{b_i, \{w_{ij}\}, p_i\}$ от агентов на взаимное соответствие и удаление нерелевантных объектов:

Объект $b_i$ удаляется как нерелевантный, если

$$\exists\, j\colon \begin{cases} w_{ij} \notin \left[w_{ij}^{min}, w_{ij}^{max}\right], \text{если } w_{ij} - \text{числовой признак} \\ w_{ij} \notin \{w_{ij}^{norm}\}, \text{если } w_{ij} - \text{категориальный признак} \end{cases},$$

$w_{ij}^{min}$ – минимальное возможное значение для признака $w_{ij}$ объекта $b_i$,

$w_{ij}^{min}$ – максимально возможное значение для признака $w_{ij}$ объекта $b_i$,

$\{w_{ij}^{norm}\}$ – список нормальных значений для признака $w_{ij}$ объекта $b_i$.

Объект $b_i$ удаляется как взаимоисключающий, если в списке $\{b_i, w_{ij}, p_i\}$ $\exists\, k \neq i\colon p_i < p_k$ и $\forall j\colon w_{ij} \simeq w_{kj}$ .

• Конгломерационная:

В конгломерации агенты в определенном месте образуют собрание, чтобы выполнить свои требования, которые они не могут выполнить в одиночку. Фермерский рынок можно рассматривать как собрание людей, где люди собираются, чтобы продать/купить то, что им нужно. Каждый агент может покинуть собрание или присоединиться к нему, но в каждый момент времени он должен быть частью только одного собрания. Удовлетворенность агента в собрании, то есть степень, в которой агент выполняет свои требования, зависит от других агентов в собрании. В собрании всегда должен быть хотя бы один член. На рис. 21 показана организация собрания. Эта организация эффективна, когда каждому агенту требуются ресурсы других агентов для достижения своей цели или выполнения своих задач.

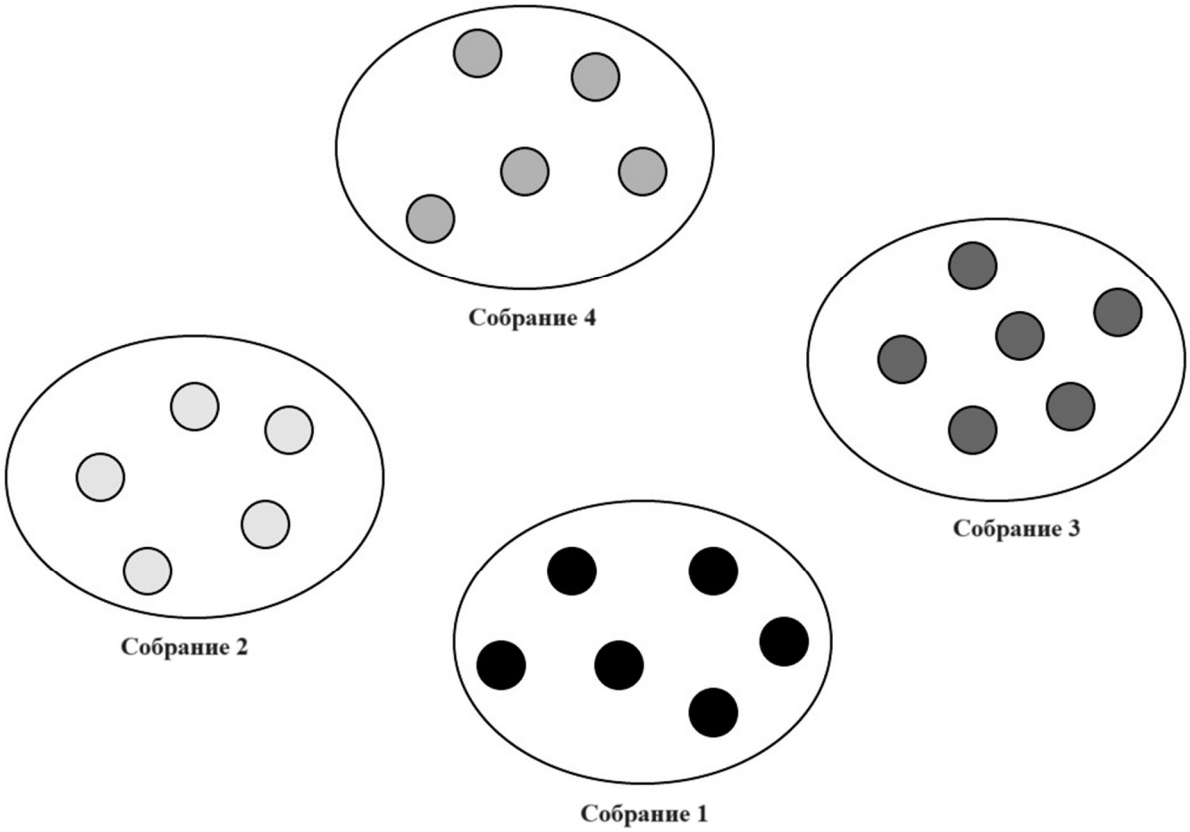


Рис. 21. Конгломерационная организация МАС

Данная организация особенно эффективна, когда агентам требуется найти один и тот же объект, либо же когда объекты одних агентов в собрании используются в качестве признаков объектов других агентов в том же собрании. Чтобы агенту $a_l$, нацеленному на поиск объектов $b_i$, понять, следует ли им вступать в собрание $S_q$, вводится мера удовлетворенности:

$$m_{a_l}{}^{Sq} = \begin{cases} +1 \text{ за каждого агента с аналогичным объектом } b_i \\ +1 \text{ за каждого агента, использующего } b_i \text{ в качестве признаков для объектов } b_j \\ +1 \text{ за каждого агента, чьи объекты } b_j \text{ являются признаками для объектов } b_i \\ -1 \text{ за каждого агента, который не удовлетворяет любому из вышеперечисленных признаков} \end{cases}$$

Агент $a_l$ выбирает собрание с наибольшей положительной метрикой $m_{a_l}{}^{Sq}$. Если все метрики отрицательны, то агент в собрания не вступает. При нахождении агентом $a_l$ объекта $b_i$, или же по достижению максимального

времени без нахождения объекта, метрики должны быть заново пересмотрены.

## 2.3 Проблема влияния изменения точности/полноты на производительность модели

Разные агенты могут потребовать для решения задачи разные инструменты (алгоритмы). Так, если агент представляет собой нейронную сеть, основными алгоритмами для решения условий (2.3) и (2.4) (увеличение точности и полноты агента) будут: подбор гиперпараметров нейронной сети методами решетчатого поиска или с использованием генетического алгоритма, подбор архитектуры нейронной сети и т.п. Для разрешения условия (2.5) (увеличение производительности агента) для подбора гиперпараметров подойдут: подбор гиперпараметров нейронной сети методами решетчатого поиска или с использованием генетического алгоритма, подбор архитектуры нейронной сети и т.п.

Как можно заметить, для всех описанных условий подходят одни и те же инструменты. Это приводит к тому, что улучшение одного условия может негативно сказаться на остальных условиях. Для нахождения баланса между условиями необходимо ввести меры по их взаимному влиянию на результат.

В качестве такой меры для условий (2.3) и (2.4) традиционно служит F-мера [109], являющаяся гармоническим средним между точностью и полнотой:

$$\mathrm{F} = \frac{2}{\frac{1}{R} + \frac{1}{P}} = 2 \cdot \frac{P \cdot R}{P + R},$$

где $P$ – точность (precision),

$R$ – полнота (recall).

В случае, когда необходимо предоставить преимущество точности или полноте, применяется $F_\beta$ мера:

$$\mathrm{F}_{\beta} = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \ ,$$

где $\beta$ – параметр, отвечающий за то, насколько полнота должна быть важна относительно точности. При $\beta = 0$ $F_\beta$ соответствует точности $P$, при $\beta \to \infty$ $F_\beta$ соответствует полноте $R$. Типовыми же значениями для параметра помимо 1 являются 2 при котором полнота оценивается выше, чем точность, и 0.5, при которой полнота оценивается ниже, чем точность.

Отношение между условиями (2.3) и (2.4) с условием (2.5) – это компромисс между точностью и производительностью, для его нахождения вводится условие:

$$\left(\sum_{l}\sum_{in}\left(\frac{\delta(b_i, c_n) \cdot \left(1 - I(b_i, c_n)\right) \cdot I(b_i, c_n) \cdot \left(1 - \delta(b_i, c_n)\right)}{\beta^2 \cdot \delta(b_i, c_n) \cdot \left(1 - I(b_i, c_n)\right) + I(b_i, c_n) \cdot \left(1 - \delta(b_i, c_n)\right)}\right)\right)^{\alpha} \cdot$$

$$\cdot \left(\sum_{l} T(a_l)\right)^{1-\alpha} \to \min$$

где $\alpha \in [0, 1]$ – коэффициент баланса между точностью и производительностью. Если α = 0, то приоритет целиком отдается производительности, при α = 1 на первый план выходит точность, значение α = 0,5 устанавливает равнозначный приоритет точности и быстродействия.

В наиболее простой организационной структуре все агенты считаются равными, нет назначенного лидера, и каждый агент общается со своими соседями. При иерархической организации агенты имеют древовидные отношения: родители контролируют своих дочерних агентов и могут иметь своих собственных родителей. На самом высоком уровне есть один агент, известный как корневой. Иерархическая организация может привести к задержке или появлению узких мест, особенно у корневого агента, поскольку он отвечает за обработку сообщений всех дочерних агентов. В зависимости от количества родительских агентов, иерархическая организация может быть разделена на два типа: простая и единообразная. В простом подходе корневой агент имеет исключительные полномочия и контролирует все

коммуникации. При единообразном подходе в иерархии имеется более одного родительского агента, что означает, что в дополнение к корню все или отдельные родители могут также иметь своих детей.

Для иерархической организации агентов выделены две стратегии выбора агентов: автоматическая диспетчеризация и конкуренция. Если применяется конкурентная стратегия, то в обработку данных вовлекаются все доступные дочерние агенты, а по результатам обработки отбираются лучшие. По итогам работ выбирается результат работы одного или нескольких агентов с максимальным значением метрики.

При автоматической диспетчеризации родительский агент самостоятельно выбирает, каких дочерних агентов предпочтительнее использовать для обработки данных. Выбор основывается на анализе истории применения агентов в процессе обработки данных при использовании конкурентной стратегии в виде таблицы использования агентов и сопоставлении паттернов в данных результирующем выбору агента.

Формирование соответствия паттернов выбранным агентам может назначаться как полуавтоматически с использованием частотного анализа для последующего создания конструкций вида «if-else», так и автоматически, с использованием нейронных сетей или моделей машинного обучения.

Для частотного анализа из данных выделяются паттерны, затем на основе значений категориальных паттернов или на основе интервалов значений числовых паттернов строятся гистограммы, в которых по оси ординат указывается наиболее часто выбираемый агент и процент выбора данного агента среди остальных других. В итоге выбирается агент/агенты с наибольшим весом.

## 2.4 Основные алгоритмы, обеспечивающие сходимость метода

Алгоритм обратного распространения ошибки (Back propagation algorithm) – один из наиболее известных алгоритмов машинного обучения и обучения нейронных сетей. Использует выходную ошибку нейронной сети для вычисления величин коррекции весов нейронов в ее скрытых слоях.

Анализ с помощью характеристической кривой (ROC-analisys) – графический метод оценки качества работы бинарного классификатора и выбора дискриминационного порога для разделения классов. Отражает связь между вероятностью ложной тревоги и вероятностью правильного обнаружения.

Генетический алгоритм (Genetic algorithm) – метод решения задач оптимизации, основанный на принципах процессов естественного отбора (мутация, скрещивание, отбор). Является частью более обширного направления искусственного интеллекта — эволюционных вычислений.

Подбор гиперпараметров (Selection of hyperparameters) – метод подбора параметров алгоритмов, значения которых устанавливаются перед запуском процесса обучения (тогда как обычные параметры вычисляются в процессе обучения).

Дерево решений (Decision Trees) – один из наиболее популярных инструментов классификации в интеллектуальном анализе данных и бизнес-аналитике. Строится на основе решающих правил вида «если, то», упорядоченных в древовидную иерархическую структуру.

Дисперсионный анализ (Analysis of variance) – статистический метод для определения влияния различных факторов на исследуемую переменную. Применяется для выбора наиболее важных факторов и оценки их влияния.

Классификация с учетом издержек (Cost-sensitive classification) – случай бинарной классификации, когда издержки ошибок классификации не одинаковы. Имеет большое значение в машинном обучении для

квалификационных моделей (регрессия, нейросеть) в условиях несбалансированной выборки.

Метод муравьиной колонии (Ant colony optimization) – алгоритм для нахождения приближенных решений задач оптимизации на графах, таких как задача коммивояжера, транспортная задача и аналогичных. Используется во многих приложениях анализа.

Метод Ньютона (Newton method) – алгоритм для экспериментального поиска экстремума функции. Используется в анализе данных для решения задач оптимизации, где требуется определить нуль первой производной либо градиента в случае многомерного пространства.

Нечеткая логика (Fuzzy logic) – форма многозначной логики, в которой истинные значения переменных могут быть любыми действительными числами от 0 до 1 включительно. Применяется во многих областях, от теории управления до искусственного интеллекта.

Решающее правило (Decision Rule) – в машинном обучении и анализе данных — правила вида «если, то», определяющие принадлежность объекта к заданному классу. Применяются в деревьях решений и алгоритмах последовательного покрытия.

## 2.5 Выводы по Второй главе

Представлена обобщенная постановка задачи интеллектуального взаимодействия между агентами. Деятельность агентов абстрагируется до нахождения ими некоторых объектов. Это позволяет математически описать требования к результатам работы как самого агента, так и к объединениям агентов в различные модели. Для таких объединений приведены различные варианты инфраструктур модели, к каждому варианту даны комментарии и приведены дополнительные требования к организации агентов внутри модели.

После постановки задачи выявлена проблема отрицательного влияния точности и полноты агента и предложенных моделей на основе многоагентного подхода на производительность. Для решения данной проблемы предлагается установить приоритет между производительностью и точностью с полнотой.

В конце главы приведены варианты алгоритмов для сходимости моделей. Представленные в данной главе варианты моделей предоставляются пользователю на выбор, при потребности в решении соответствующих задач.

Таким образом, метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов состоит в инкапсуляции искусственных нейронных сетей посредством программных агентов, объединенных с помощью связей и соединений в архитектуру виртуального мира, наиболее рационально обеспечивающую их взаимодействие между собой для реализации стратегий автоматической диспетчеризации и конкуренции. Представленный метод обеспечивает адаптивность системы в условиях изменяющейся обстановки без переобучения интеллектуальных компонентов.

# ГЛАВА 3. АРХИТЕКТУРА И АЛГОРИТМЫ МУЛЬТИАГЕНТНОЙ СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ НА БАЗЕ ПРЕДИКТИВНОГО ОРКЕСТРАТОРА

## 3.1 Архитектура мультиагентной системы распознавания образов на базе предиктивного оркестратора

Основная идея при создании многоагентной среды состоит в том, чтобы разделить систему на распределенные части с автономным поведением. Эти части активно взаимодействуют в поисках наилучшего сочетания вариантов обработки данных для решения исходной проблемы. С точки зрения реализации нейронных сетей на практике такой подход дает возможность комбинировать несколько решений вместо обучения одной нейронной сети, что может потребовать значительных затрат и времени. Учитывая характер предлагаемого подхода, он может быть реализован с использованием многоагентной технологии в качестве парадигмы разработки программного обеспечения. Архитектура среды с агентом-координатором (Matcher) представлена на рис. 22.

Существует три типа стратегий агента Matcher: аукцион, автоматическая диспетчеризация и конкуренция:

• Аукцион—это опрос агентов с целью выяснить, кто из них способен обработать данные. На основе результатов опроса агент Matcher выбирает агентов для обработки текущего набора данных;

• При автоматической диспетчеризации агент Matcher самостоятельно выбирает, какой агент или агенты предпочтительнее использовать для обработки данных. Данный тип стратегии по умолчанию используется в модели IMatcher;

• Если применяется конкурентная стратегия, то в обработку данных вовлекаются все доступные агенты, а по результатам обработки отбираются лучшие.

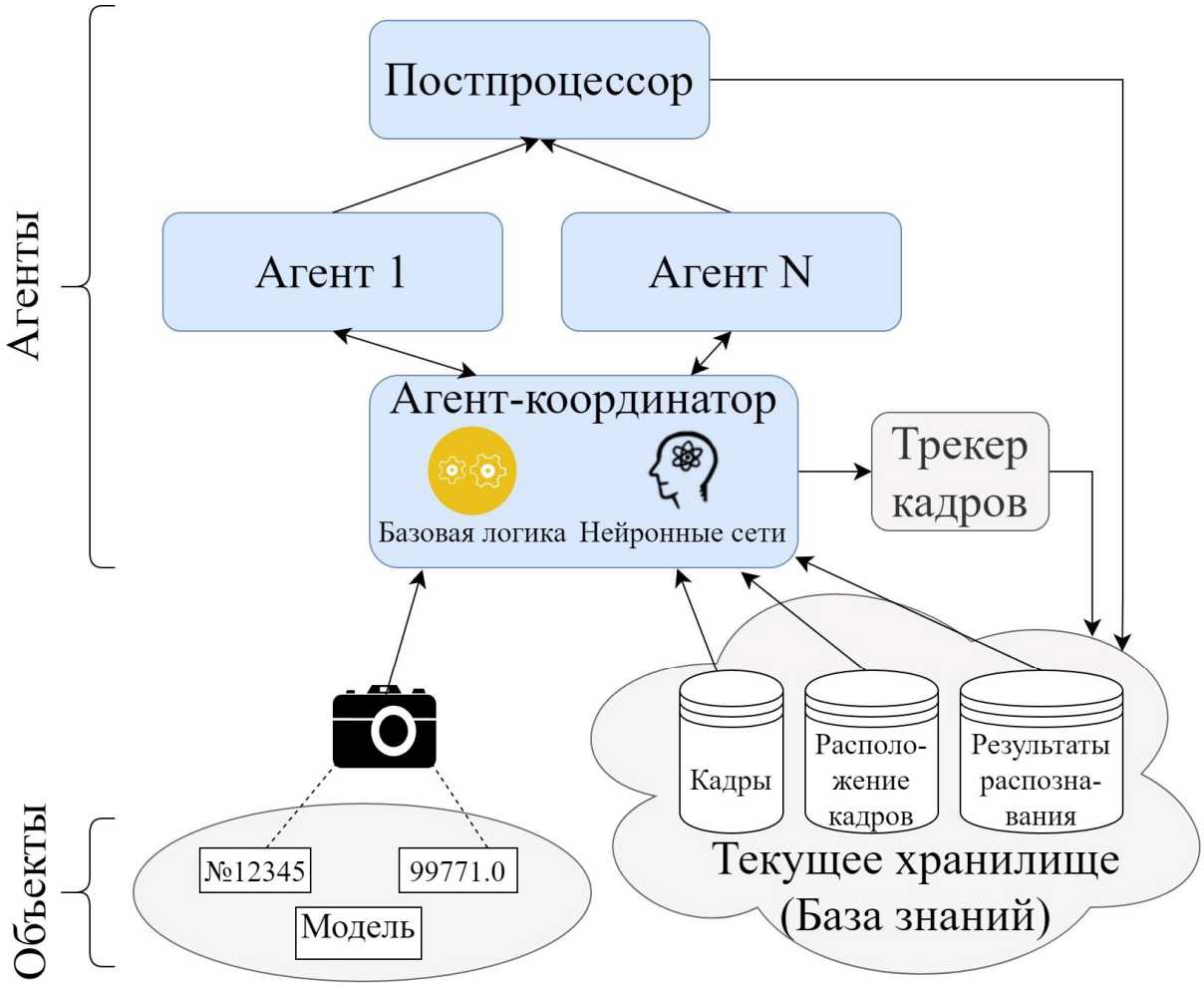


Рис. 22. Архитектура интеллектуального диспетчера

Агенты используют в своей работе базовую логику и нейронные сети. Распознаватели цифр используются для поиска чисел на изображении. Постпроцессор пытается идентифицировать цифры, относящиеся к счетчику, среди всех цифровых символов в поле зрения. Полученные данные передаются в текущее хранилище. Когда результирующие данные найдены, подключается трекер кадров, его назначение - рассчитывать перемещения камеры между кадрами.

Эти данные позволяют агенту Matcher сравнивать показания, снятые в разные моменты времени. Кроме того, в результате перемещения камеры на

значительное расстояние от счетчика Текущее хранилище может быть сброшено. Со временем данные счетчика, снятые под разными углами, накапливаются в текущем хранилище. Matcher выбирает лучшие результаты счетчика из всех успешных кадров и генерирует окончательный результат распознавания.

Стратегия аукциона позволяет распределить логику принятия решений между компонентами многоагентной архитектуры. При встрече с неизвестным объектом система может организовать опрос, отправив запросы всем задействованным распознающим агентам. Они могут либо ответить о возможности обработать изображение на основании предварительного понимания типа объекта, либо попытаться выполнить распознавание изображения и ответить в случае хорошего качества идентификации. Получив ответы, Matcher может выбрать лучший и начать переговоры с соответствующим агентом о дальнейшей идентификации.

Еще одной особенностью Matcher является возможность управления настройками датчиков, например, камеры, благодаря чему она может автоматически фокусироваться, включать фонарик и регулировать баланс белого (цветовой баланс).

Реализация многоагентного подхода позволяет обеспечить высокую автономность распознавателей, ввести новые алгоритмы распознавания с небольшими изменениями архитектуры и смешивать их в случае высокого уровня неопределенности. В отличие от других многоагентных реализаций распределенных интеллектуальных приложений, это решение не требует дифференциации областей применения нейронных сетей. Несколько разных распознавателей можно обучить с использованием пересекающихся множеств. Поэтому такая архитектура остается открытой и дает возможность для постоянного развития за счет добавления новых распознавателей без замены предыдущих.

При реализации предложенной архитектуры на практике было рассмотрено два варианта организации взаимодействия между агентами.

Внедрение централизованного распознавания изначально кажется очевидным решением с учетом требований системы и условий функционирования нейронной сети. Логика самого модуля (см. рис. 25) основана на довольно простой линейной архитектуре. Изображения показаний счетчиков, передаются через Интернет на центральный вычислительный сервер, где обрабатываются в однопоточном режиме (в потоке Main Thread) с помощью метода PreprocessImagePipeline.

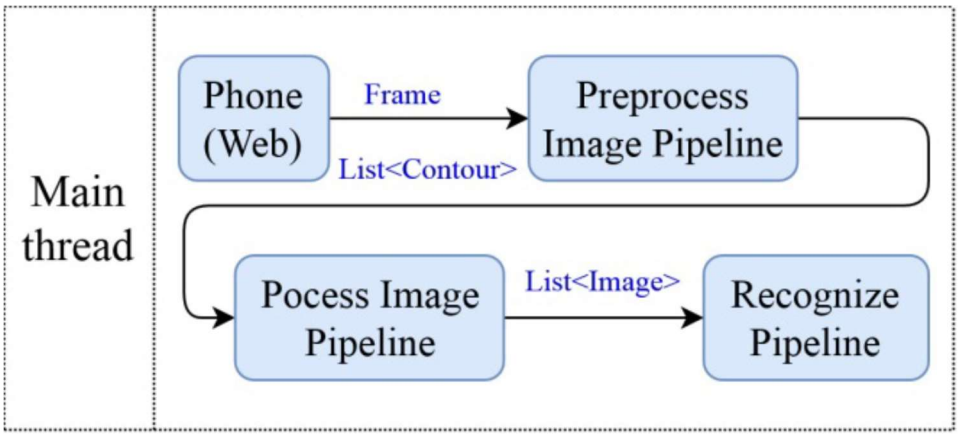


Рис. 23. Архитектура централизованного решения.

Этот метод обрабатывает входное изображение более чем 1000 различными способами, чтобы найти контуры показаний на изображении. Те обработанные изображения, на которых были найдены потенциально подходящие контуры, передаются в метод ProcessImagePipeline.

Метод ProcessImagePipeline более внимательно рассматривает получившиеся контуры и удаляет лишние. Используя оставшиеся контуры, он вырезает числа из обработанных изображений и передает их для распознавания методу RecognizePipeline. Внутри метода RecognizePipeline цифровые изображения распознаются с помощью нейронной сети LeNet. Для каждого варианта обработки изображения получается результат распознавания. Среди всего набора результатов выбирается наилучший.

Архитектура модуля распределенного распознавания представлена на рис. 26.

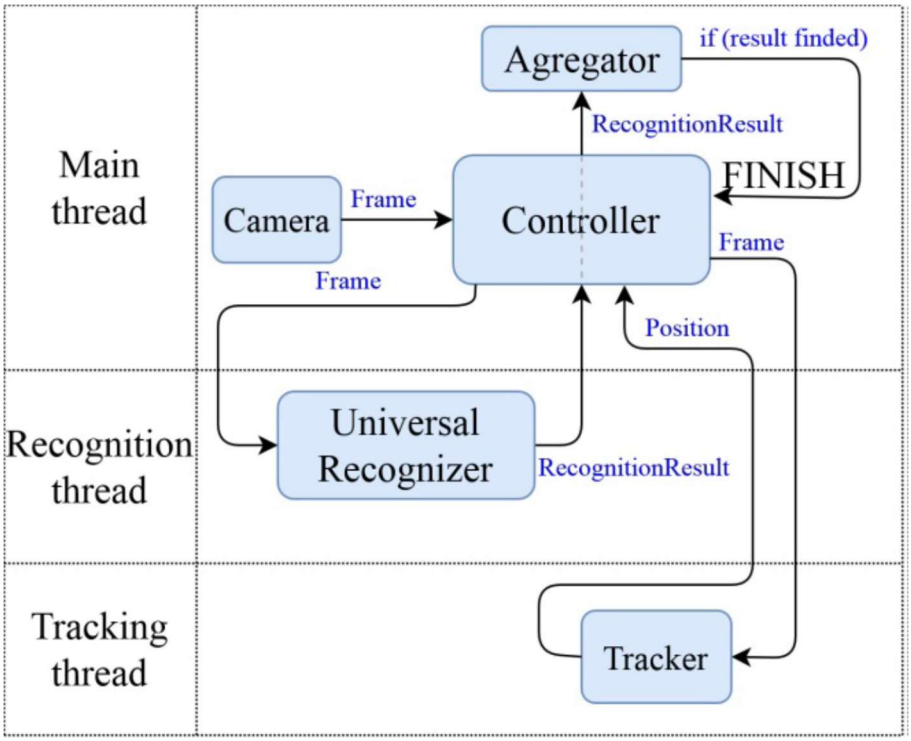


Рис. 24. Архитектура распределенного распознавания

В основном потоке метод Controller получает изображения с камеры, отправляет их в методы UniversalRecognzier и Tracker, а также передает результаты в метод Aggregator. Метод Aggregator сохраняет и анализирует результаты, по результатам анализа возвращает отчет методу Controller, в котором указывается, готов ли окончательный результат или необходимо продолжить сбор результатов.

В отдельном потоке Recognition работает метод UniversalRecognizer, который представляет собой централизованный модуль распознавания, описанный выше, в котором количество обработок входного изображения сокращается с 1000 до 10, а тип обработки выбирается случайным образом, а нейронная сеть LeNet заменена на YOLOv2.

В еще одном потоке для отслеживания перемещения камеры запускает метод Tracker. Он берет текущее изображение и возвращает разницу с предыдущим изображением, то есть направление, в котором камера двигалась. Эта информация впоследствии передается через

контроллер вместе с результатами распознавания в метод агрегатора. Там он помогает сравнивать между собой результаты, полученные с разной периодичностью, а также сбрасывать старые результаты при больших изменениях положения камеры.

Предложенная архитектура может быть доработана путем замены агента-координатора (модель IMatcher) на рекуррентную нейронную сеть на основе входящих данных – модель RMatcher. Использование нейронной сети позволяет выявлять скрытые паттерны (закономерности) во входящих данных и производить выбор агентов более эффективно, в отличие от классических моделей на основе условий (ветвлений). В качестве нейронной сети была выбрана рекуррентная нейронная сеть, состоящая из двух слоев ячеек LSTM, поверх которых также были расположены два полносвязных слоя. Архитектура нейронной сети представлена на рис. 25.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, None, 60)          21360
_________________________________________________________________
batch_normalization_1 (Batch (None, None, 60)          240
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                11904
_________________________________________________________________
batch_normalization_2 (Batch (None, 32)                128
_________________________________________________________________
dense_1 (Dense)              (None, 28)                924
_________________________________________________________________
batch_normalization_3 (Batch (None, 28)                112
_________________________________________________________________
dense_2 (Dense)              (None, 8)                 232
=================================================================
Total params: 34,900
Trainable params: 34,660
Non-trainable params: 240
```

Рис. 25. Архитектура нейронной сети в модели RMatcher

В качестве функции потерь была выбрана категориальная перекрестная энтропия (CategoricalCrossentropy), роль оптимизатора была отдана методу адаптивной оценки моментов (ADAM), у рекуррентных слоев LSTM в качестве функции активации был выбран гиперболический тангенс (tanh), а у полносвязных слоев – линейная активация. Количество нейронов в каждом слое указано на рис. 26, остальные гиперпараметры нейронной сети брались по умолчанию согласно библиотеке глубокого обучения Keras v2.3.1.

Рекуррентная нейронная сеть в модели RMatcher для своей работы использует все те же данные, что и модель IMatcher, см. рис. 24. Под входными данными следует понимать:

1) Данные с камеры (общая яркость кадра, а также нижний и верхний квартили яркости);
2) Данные с каждого агента, после обработки postprocessor-ом (процент уверенности в распознавании цифр, кол-во распознанных цифр);
3) Данные с postprocessor (вероятность нахождения показаний счетчика, наиболее вероятный тип счетчика)
4) Данные с Frame Tracker (вектор смещения относительно предыдущего кадра, необходимость сброса текущего хранилища)
5) Состояние Knowledge Base (количество кадров с момента запуска распознавания, суммарное количество найденных показаний и динамика изменения вероятности нахождения распознанных показаний за последние 10 кадров)

Выходные данные включают в себя:

1) Данные о выборе каждого агента в виде цифры 0 или 1.
2) Необходимость сброса накопленных данных и начала распознавания заново в виде цифры 0 или 1.
3) Активация автоматической фокусировки, включение фонарика и регулировка баланс белого, каждый параметр в виде цифры 0 или 1.

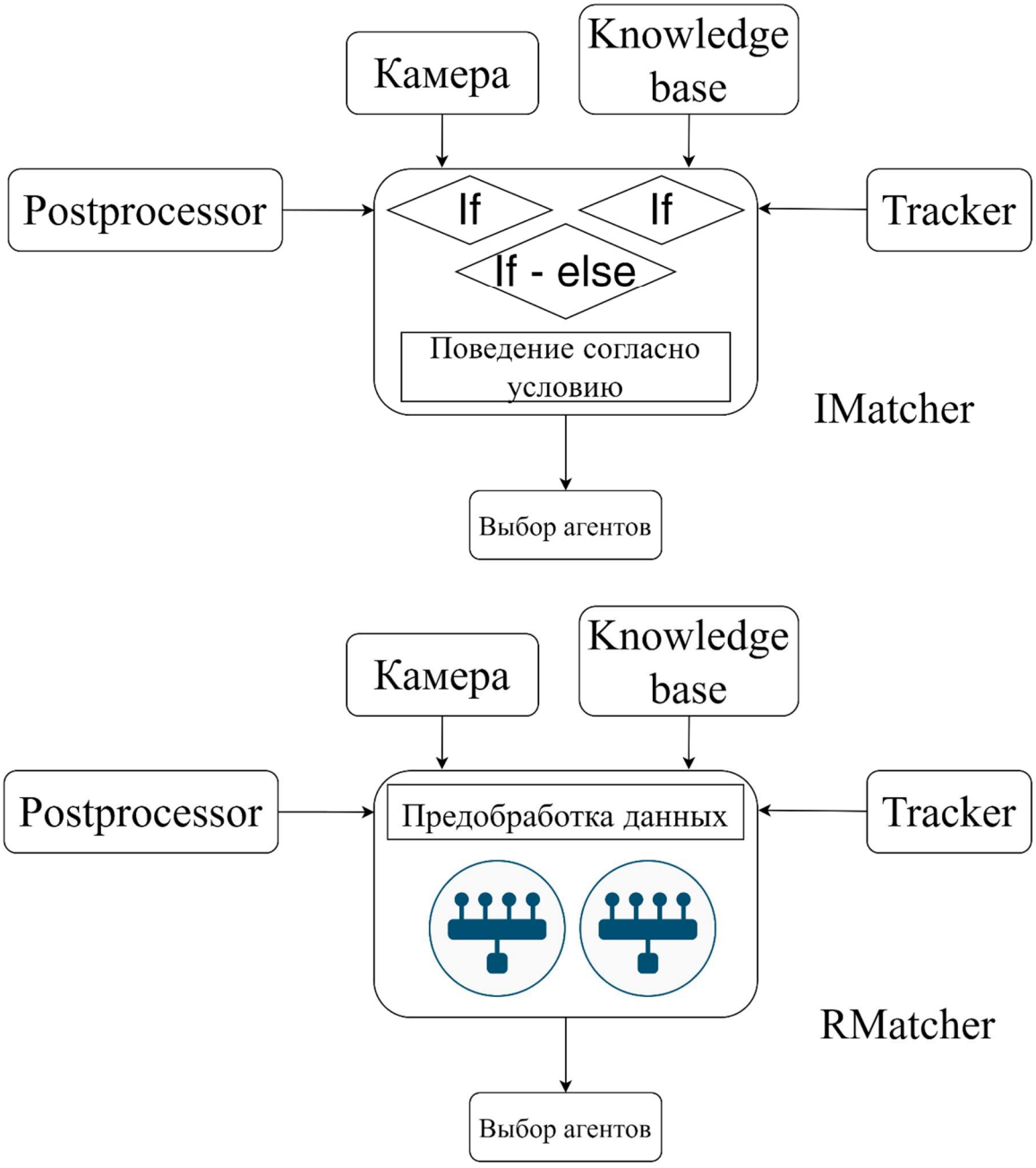


Рис. 26. Схематический вид приема и выдачи данных агентами IMatcher (сверху) и RMatcher (снизу)

### 3.2 Алгоритм интеллектуальной обработки данных

Для реализации взаимодействия интеллектуальных компонентов был разработан следующий алгоритм распределения задач в адаптивной системе распознавания образов.

Пусть данные с условных датчиков $D$ поступают на входы выбранных агентов $a_l, l = \overline{1, L_{выбр}}$. Агенты $a_l$ на выходе возвращают список объектов $b_i$ с характеристиками $w_{ij}$, для каждого объекта возвращается вероятность его нахождения $p_i$: $a_l(D) \rightarrow \{b_i, \{w_{ij}\}, p_i\}$ - агент обрабатывает свойственным ему образом входные данные и возвращает список найденных объектов.

Шаг 1. Постпроцессор *pp* проверяет списки полученных объектов $\{b_i, \{w_{ij}\}, p_i\}$ на взаимное соответствие, убирает нерелевантные, а также, при необходимости добавляет к объектам дополнительные характеристики. Получившиеся объекты заносятся в базу знаний:

$$pp(\{b_i, w_{ij}, p_i\}) \rightarrow \{b_{i*}, w_{i*j}, p_{i*}\}, \quad (3.1)$$

Шаг 2. Объект $b_i$ удаляется как нерелевантный, если

$$\exists\, j: \begin{cases} w_{ij} \notin \left[w_{ij}^{min}, w_{ij}^{max}\right], \text{если } w_{ij} - \text{числовой признак} \\ w_{ij} \notin \{w_{ij}^{norm}\}, \text{если } w_{ij} - \text{категориальный признак} \end{cases},$$

$w_{ij}^{min}$ – минимальное возможное значение для признака $w_{ij}$ объекта $b_i$,

$w_{ij}^{min}$ – максимально возможное значение для признака $w_{ij}$ объекта $b_i$,

$\{w_{ij}^{norm}\}$ – список нормальных значений для признака $w_{ij}$ объекта $b_i$.

Шаг 3. Объект $b_i$ удаляется как взаимоисключающий, если в списке $\{b_i, w_{ij}, p_i\}$ $\exists\, k \neq i: p_i < p_k$ и $\forall j: w_{ij} \simeq w_{kj}$

Шаг 4. Для ряда объектов, при необходимости, постпроцессором могут быть достроены дополнительные характеристики на основе определенных агентом характеристик:

$$\exists\, i_k \in \{i_1, i_2, \ldots i_N\}: w_{i_k}^* = F(\{w_{i_k,j}\}), \quad (3.2)$$

где $N$ – количество объектов, у которых можно/нужно достраивать признаки, $w_{i_k}^*$ - новый признак объекта, $F(\{w_{i_k,j}\})$ – функция для построения нового признака на основе существующих признаков $\{w_{i_k,j}\}$.

Существует ряд признаков $w_{ij}$, положение/значение которых зависит непосредственно от исходных данных $D_k$.

Шаг 5. Трекер *tr* следит за изменением характеристик/положения/ у объектов, возвращает дельту изменения признака на основе изменения исходных данных:

$$\forall\, i\ \exists\, h \in \{h_1, h_2, \dots h_L\}, \exists m \in \{m_1, m_2, \dots m_{\mathrm{L}}\}: tr\left(D_h^{t^{pred}}, D_h^t\right) \rightarrow \Delta w_{im}^{t^{cur}} \qquad (3.3)$$

где $t^{cur}$ – текущее значение времени, $t^{pred}$ – предыдущее значение времени,
$L$ – количество доступных к отслеживанию исходных данных/признаков.

Шаг 6. Если $\exists\, m: \Delta w_{im}^{t^{cur}} > \Delta w_{im}^{crit}$, где $\Delta w_{im}^{crit}$ – критическое значение для признака $w_{im}$, то трекер передает сигнал на координатора для сброса сбора сведений.

Шаг 7. База знаний *bz* принимает список объектов $\{b_i,\ \{w_{ij}\},\ p_i\}_t$ из постпроцессора и данные по изменению характеристик $\Delta w_{im}^{t^{cur}}$ из трекера. Ее цель – по списку объектов найти окончательный результат.

Шаг 8. Для выполнения поставленной цели *bz* выполняет следующие действия:

Действие 8.1. Обновление характеристик $w_{ij}$ у объектов за предыдущие периоды времени:

$$\exists m \in \{m_1, m_2, \dots m_L\}: \forall\, i \text{ и } \forall\, t < t^{cur} \text{ если } w_{im}^t \in \{w_{ij}\}_t,$$

$$\text{то } \begin{cases} w_{im}^t = w_{im}^t + \Delta w_{im}^{t^{cur}}, \text{если } w_{ij} - \text{числовой признак} \\ w_{im}^t = f(w_{im}^t, \Delta w_{im}^{t^{cur}}), \text{если } w_{ij} - \text{категориальный признак} \end{cases} \qquad (3.4)$$

где $t^{cur}$ – текущий момент времени,
$f$ – заранее заданная функция по изменению категориальных признаков.

Действие 8.2. Повышение вероятности $p$ у объектов, которые на протяжении времени $t^*$ не исчезали и не меняли своих характеристик:

На основе (2.1) рассчитаем функцию идентификации объектов *Id*:

$$Id(b_1, b_2) = \begin{cases} 1, \text{если } p(b_1, b_2) \geq p^* \\ 0, \text{если } p(b_1, b_2) < p^* \end{cases},$$

где $p^*$ - пограничное значение вероятности.

Тогда если $\exists\, b_i^*: \int_{\mathrm{t^{cur}}-\mathrm{t^*}}^{\mathrm{t^{cur}}} \frac{Id(b_i^t, b_i^*)}{t^*}\,\mathrm{d}t \geq p^*$, то установим $p_i = 1$ и $b_i = b_i^*$ в каждом *i*-том элементе списка $\{b_i,\ \{w_{ij}\},\ p_i\}_t$, $b_i^*$ – объект, максимально похожий на объекты $b_i^t$ для $t \in [t^{cur} - t^*, t^{cur}]$, $t^*$ – период времени, за который рассматриваются объекты.

Действие 8.3. Применения ряда критериев *K* на обновленных исторических данных:

Имеем список $\{b_i,\ \{w_{ij}\},\ p_i\}_t$ и ряд критериев по поиску результата $K_x \in \{K_1, K_2, \dots K_{\mathrm{X}}\}$

$$\forall\, x\ \in [1, 2, \dots \mathrm{X}]: K_x\left(\left\{b_i, \left\{w_{ij}\right\}, p_i\right\}_t\right) \to k_{xt}, \tag{3.5}$$

$k_{xt} \in [0, 1]$– коэффициент нахождения результата критерием $K_x$

Если $k_{xt} > k^*$, считаем, что результат найден. По умолчанию $k^* = 2/3$.

Если результат не найден, то продолжаем собирать данные.

Интеграл изменения по времени результатов (распознавания) должен быть больше $k_x^{**}$, где значение $k_x^{**}$ зависит от критерия $K_x$.

$$\forall\, x\ \in [1, 2, \dots \mathrm{X}]: \int_{t^{cur}-t^*}^{t^{cur}} |K_x'|\,\mathrm{d}t = \int_{t^{cur}-t^*}^{t^{cur}} \mathrm{sign}(K_x') * \frac{\mathrm{d}K_x}{\mathrm{d}t}\,\mathrm{d}t =$$

$$\int_{K_{x(t=t^{cur}-t^*)}}^{K_{x(t=t^{cur})}} \mathrm{sign}(K_x')\,\mathrm{d}K_x \geq k_x^{**}$$

Смысл этого выражения состоит в том, что критерии должны постоянно меняться, причем не обязательно улучшаться, иначе процесс или встает на паузу или начинается заново.

### 3.3 Выводы по Третьей главе

Представлена архитектура интеллектуальной среды с использованием многоагентного подхода. Она позволяет комбинировать произвольное количество агентов, в том числе, на основе нейронных сетей, для решения различных задач. Введен агент-координатор, отвечающий за выбор решающего агента, расписаны стратегии выбора агентов. Описан вариант замены агента-координатора на агента на основе рекуррентной нейронной сети.

Таким образом, представленная архитектура мультиагентной системы распознавания образов, основанная на реализации предиктивного оркестратора IMatcher или RMatcher позволяет реализовать сочетание интеллектуальных компонентов с автономным поведением. В результате решена проблема комплексирования автономных искусственных нейронных сетей в интеллектуальной системе распознавания образов, способной адаптироваться к меняющимся внешним условиям эксплуатации. Использование нейронной сети в основе предиктивного оркестратора RMatcher позволяет выявлять скрытые паттерны (закономерности) во входящих данных и производить выбор агентов более эффективно, в отличие от классической модели IMatcher на основе условий (ветвлений). Представленный мультиагентный алгоритм распределения задач в адаптивной системе распознавания образов предоставляет возможность динамического изменения критериев выбора интеллектуальных агентов при корректировке условий задачи распознавания образов и позволяет повысить качество распознавания.

# ГЛАВА 4. РЕЗУЛЬТАТЫ РЕАЛИЗАЦИИ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ НА БАЗЕ МУЛЬТИАГЕНТНОГО АНСАМБЛИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ КОМПОНЕНТОВ АДАПТИВНОЙ СИСТЕМЫ РАСПОЗНАВАНИЯ ОБРАЗОВ

## 4.1 Формирование обучающего набора данных для примера системы компьютерного зрения

Предложенные в диссертации метод и средства были реализованы в виде структуры в специализированном мобильном приложении для автоматизированной фиксации показателей электросчетчиков путем фотографирования показаний, распознавания и передачи их в центр обработки данных, распознавания и оперативного анализа сотрудниками региональной энергораспределительной компании.

При изначальном решении задачи распознавания показаний электрических счетчиков была создана и введена в эксплуатацию централизованная архитектура, но по результатам работы выяснилось, что мобильные устройства не всегда обеспечивают стабильную связь с центральным сервером. Иногда возникали проблемы с производительностью, когда, помимо профессиональных обходчиков, руководство компании отправляло других сотрудников для сбора данных. Такой рост запросов отрицательно сказался на времени распознавания. Для решения этих проблем было решено передать процесс распознавания с центрального сервера на специализированные автономные распознаватели. Эти модули могут быть развернуты либо на смартфонах инспекторов, либо на выделенных серверах в облаке.

В результате была разработана архитектура распределенного распознавания показаний. По результатам работы модуля централизованного

распознавания полученные результаты были обобщены в концепцию интеллектуального диспетчера.

В качестве основы набора данных для обучения агентов на основе нейронной сети с архитектурой LeNet использовался датасет MNIST (сокращение от «Modified National Institute of Standards and Technology»), представляющий собой базу данных образцов рукописного написания цифр. Данные в датасете состоят из заранее подготовленных примеров изображений, на основе которых проводится обучение и тестирование систем. Образцы в наборе MNIST являются серыми полутоновыми изображениями размером 28x28 пикселей. База данных MNIST содержит 60000 изображений для обучения и 10000 изображений для тестирования, примеры изображений приведены на рис. 27.

Во время тестовой эксплуатации программного комплекса были дополнительно получены 2700 изображений цифр с приборов учета, см. рис. 28. По итогу эти изображения вошли в набор для обучения.

Рис. 27. Пример изображений в датасете MNIST

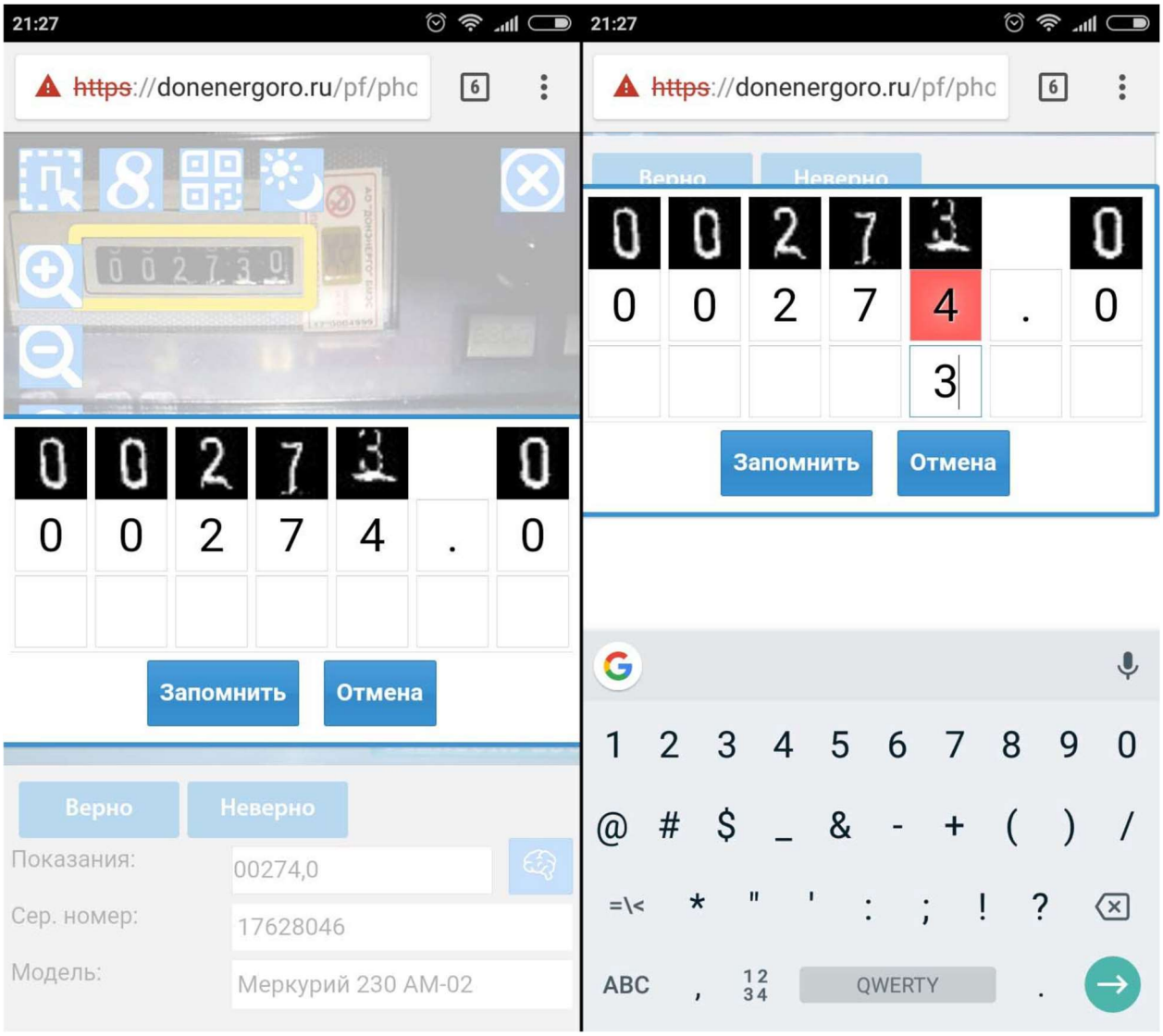


Рис. 28. Пример создания дополнительных данных для дообучения нейронной сети

Вскоре после начала работ выяснилось, что с помощью набора MNIST добиться хорошего результата не удастся. Для создания нового набора данных было собрано около 1000 шрифтов. На основе этих шрифтов были сформированы изображения цифр в количестве 10 000 экземпляров.

После дополнения этого набора поворотами и сдвигами был создан набор данных, состоящий из 196 000 изображений цифр, разделенных на обучающий набор (80%) и тестовый набор (20%), см. рис. 29. Оказалось, что 30% всех цифр «1» не отличаются друг от друга, они были заменены дополнительными преобразованиями оставшейся исходной «1».

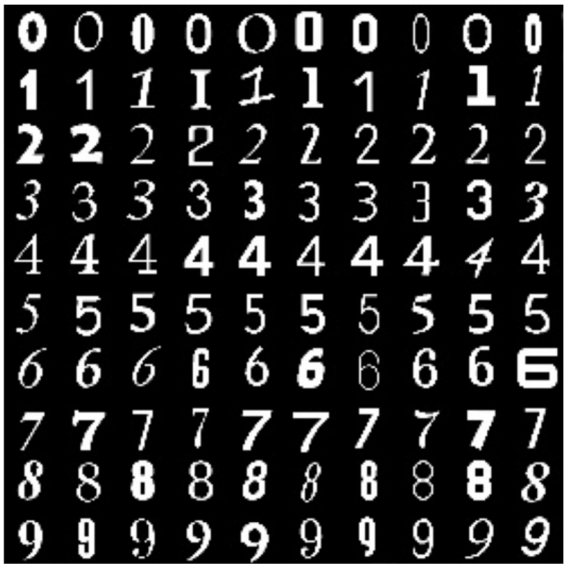

Рис. 29. Пример изображений цифр

Для обучения нейронной сети на основе архитектуры YOLOv2 было решено использовать смоделированный набор данных. Причина заключалась в том, что для обучения нейронной сети требовалось большое количество размеченных фотографий счетчиков, а собрать столько изображений и разметить вручную не представлялось возможным. Был создан генератор изображений, которая производит случайные преобразования изображений чисел путем масштабирования, затемнения, размытия, поворота и добавления фона и т.п.

На входе Генератора содержится набор из 3000 изображений цифр, полученных из системных шрифтов, и фоновых изображений. Набор фоновых изображений собран вручную из открытых источников и составляет 5 тысяч экземпляров. Каждое фоновое изображение изначально случайным образом нарезается на квадраты заданного размера (изображение размером 416x416 пикселей, согласно размеру матрицы, подаваемой в нейронную сеть с архитектурой YOLOv2).

Затем Генератор берет случайное изображение цифры и выполняет с ним серию преобразований: масштабирование, поворот на небольшой угол, сдвиг, затемнение и т.п. Аугментированное изображение цифры вставляется в случайное место на фоновом изображении, причем Генератор выполняет проверку яркости: цифры вставляются только там, где яркость фона ниже

яркости числа. Область изображения, в которую вставлена цифра, запоминается, чтобы в дальнейшем избежать наложения цифр друг на друга. Таким образом, формируется набор данных для обучения нейронной сети.

Для обучения нейронной сети был сгенерирован набор синтетических данных, содержащий 4000 изображений со случайно расположенными цифрами. С небольшой вероятностью полученное изображение могло быть размыто. Пример полученных изображений для обучения нейронной сети показаны на рис. 30.

Рис. 30. Пример полученных изображений

Набор обучающих данных характеризуется следующими параметрами:

• Цвет цифр изменяется на случайный оттенок серого в диапазоне от 200 до 255;

• Цифры наклонены под произвольным углом до 40 градусов;

• После вставки цифр на фон окончательное изображение затемняется на случайное значение от 0% до 50%;

• Окончательное изображение размывается на случайную величину от 0 до 50%;

• Цифры вставляются в фоновое изображение и сохраняются в папках с обучающими и тестовыми изображениями в соотношении 80:20.
Набор данных был разделен на обучающий набор (train set) из 3200 изображений и тестовый набор (test set) из 800 изображений.

По результатам экспериментов над набором №1 были сделаны следующие выводы о том, как улучшить обучающий набор данных:

1) Необходимо разделить обучающий набор данных на три равные части:

• без трансформации;

• с небольшим затемнением и размытием;

• с сильным затемнением и размытием;

2) Уровни размытия и затемнения должны быть ограничены. Большое количество затемненных и размытых изображений в наборе данных приводит к плохому распознаванию показаний в нормальных условиях.

3) Увеличение количества изображений в наборе данных приводит к повышению производительности нейронной сети (увеличению точности и полноты).

В результате изменений Генератора, согласно сделанным выводам, были получен набор данных, состоящий из 100000 изображений.

По результатам внедрения в промышленную эксплуатацию модели IMatcher был получен набор данных для 9000 счетчиков. Каждый элемент данного набора представляет собой последовательность входных данных для модели IMatcher на каждый кадр съемки и итоговые результаты работы IMatcher, т.е. выбор агентов для распознавания, работа с камерой и сброс хранилища. Также в эту последовательность входят данные о типе счетчика и действительное значение показаний.

Во входных данных числовые параметры, например, количество кадров с момента запуска распознавания, вероятность нахождения показаний счетчика и т.п., были отмасштабированы в диапазон от 0 до 1 относительное своих минимальных и максимальных возможных значений. Если же

минимальные и максимальные значения у параметров достоверно не известны, то они задавались вручную на основе анализа накопившейся информации. Так, например, для параметра «количество кадров с момента запуска распознавания» максимальное значение составило 1200, что при частоте 30 кадров/с соответствует 1200/30 = 40с съемки. Категориальные параметры, такие как тип счетчика и маркер необходимости сброса данных, с помощью метода быстрого кодирования (One-Hot Encoding) были переведены в вектора из нулей длиною, соответствующей количеству возможных категорий, с единицей напротив актуальной категории. Числовые параметры и категориальные параметры объединялись в суммарный вектор признаков длиной 60.

В каждом элементе набора данных для каждого кадра результаты работы IMatcher были проинтерпретированы следующим образом: если по результатам выбора IMatcher происходило любое изменение данных в Knowledge base, то данные оставались неизменными, в противном же случае, когда изменений не было, результаты выбора обнулялись, т.е. искусственно создавалась ситуация, будто бы IMatcher не выбирал агентов на данном кадре.

После предобработки каждый элемент в наборе данных представлял из себя:

1) Последовательность векторов входных данных для каждого кадра, подвергшегося распознаванию. Каждый вектор состоит из 60 значений;
2) Последовательность векторов выходных данных для каждого кадра, подвергшегося распознаванию. Каждый вектор состоит из 8 значений;
3) Значение показаний счетчика, в т.ч. показания после запятой;

Предобработанный набор данных из 9000 элементов был разделен на тренировочный и тестовый набор в составе 7200 и 1800 элементов соответственно.

Для проверки качества работы модулей распознавания был собран тестовый набор данных, включающий 138 изображений цифровых счетчиков

(всего 777 знаков) и 95 изображений аналоговых счетчиков (всего 534 знака). Поскольку модель распределенного распознавания, IMatcher и RMatcher принимает на вход видеопоток, для них были смоделированы последовательность изображений посредством ряда преобразований тестируемой фотографии, см. рис. 31. Благодаря этому удалось добиться преемственности при расчете точности у разных моделей.


Рис. 31. Пример преобразования исходного изображения счетчиков в видео последовательность

Некоторые изображения в наборе данных для тестирования в силу различных обстоятельств не поддаются анализу и считыванию показаний даже человеку, см. рис. 32. Самыми частыми причинами плохого качества изображений являются: замыливание фотографии, избыточная/недостаточная яркость фотографии и посторонние предметы на табло счетчика. Для более репрезентативного сравнения модулей распределенного распознавания между собой из полного набора изображений был выбран поднабор, состоящий из фотографии без явных признаков, мешающих считыванию показаний. Размер такого набора данных, состоящий из изображений с хорошо читаемыми показаниями приборов учета, составил 70 изображений цифровых счетчиков (всего 378 знаков) и 42 изображения аналоговых счетчиков (всего 212 знаков). Данный набор использовался для оценки работы все моделей распознавания счетчиков, что позволило сравнивать качество их работы между собой.


Рис. 32. Типовые примеры счетчиков с нечитабельными показаниями. Слева направо: посторонние предметы перед табло; недостаточное освещение, снят издалека; блики.

## 4.2 Обучение интеллектуальных программных агентов

Распознавание показаний жидкокристаллических счетчиков производились путем сравнения контуров показаний с эталонными контурами, элементы искусственного интеллекта в данном случае не использовались. Изображения аналоговых счетчиков предварительно обрабатывались с целью нахождения контуров показаний, в дальнейшем по контурам вырезались изображения цифр показаний и эти изображения распознавались нейронной сетью. Нейронная сеть была обучена на полученном наборе данных. Обучение проводилось в среде Python на базе библиотек Keras и Tensorflow на ПК под управлением Windows 7 со следующими характеристиками: Intel Core i5-3450/16Gb RAM/1000Gb HDD. Анализ показал, что нейронная сеть переобучается, см. рис. 33 и 34. После ручного выбора гиперпараметров для регуляризации ситуация с обучением нейронной сети улучшилась, см. рис. 35 и 36, но качество распознавания показаний счетчиков осталось на том же уровне.

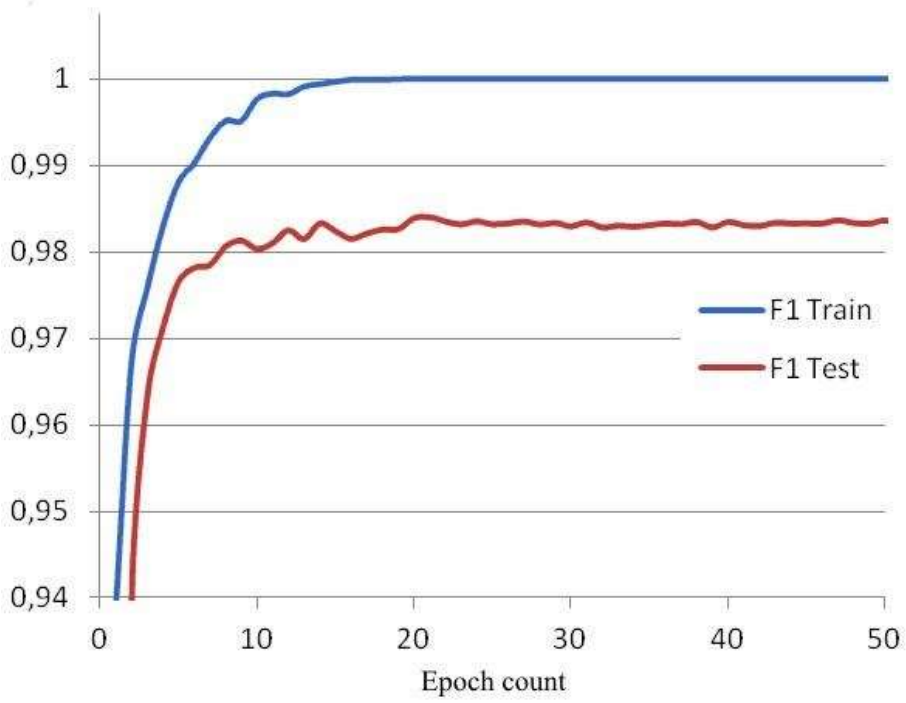


Рис 33. Результаты обучения нейронной сети (F1)

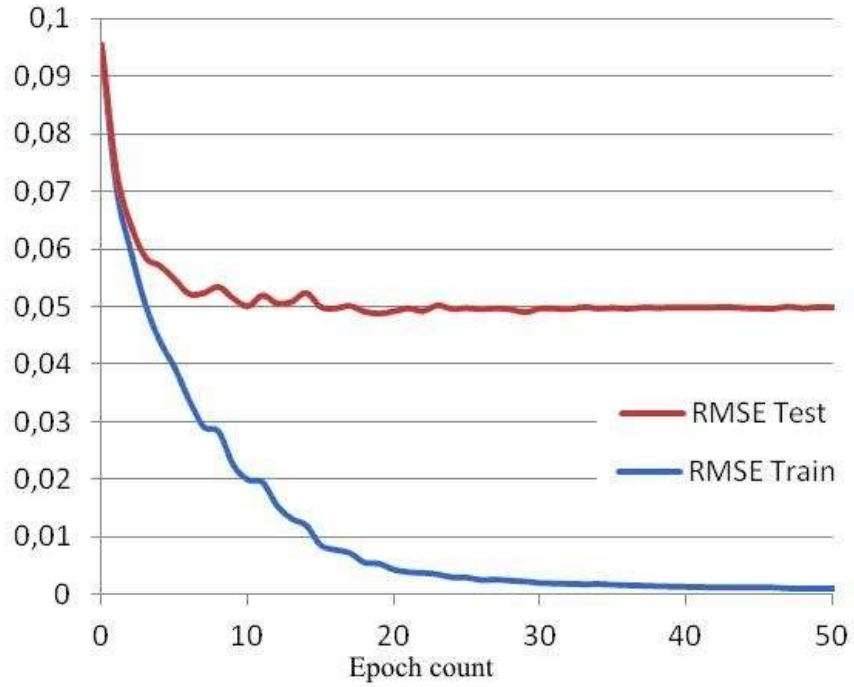


Рис. 34. Результаты обучения нейронной сети (стандартная ошибка)

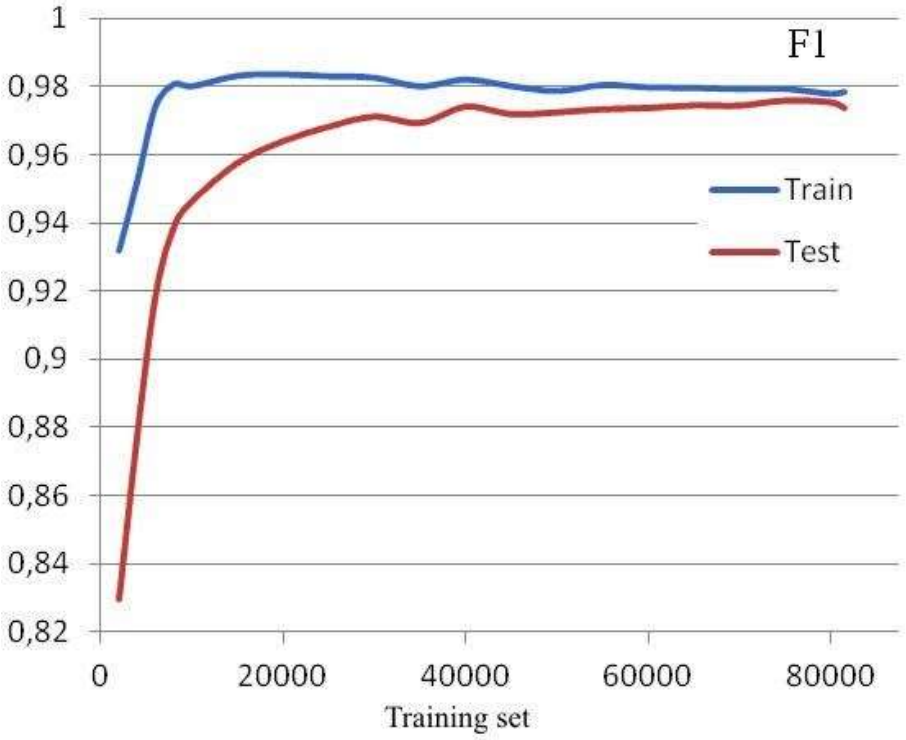


Рис. 35. Результаты обучения улучшенной нейронной сети (F1)

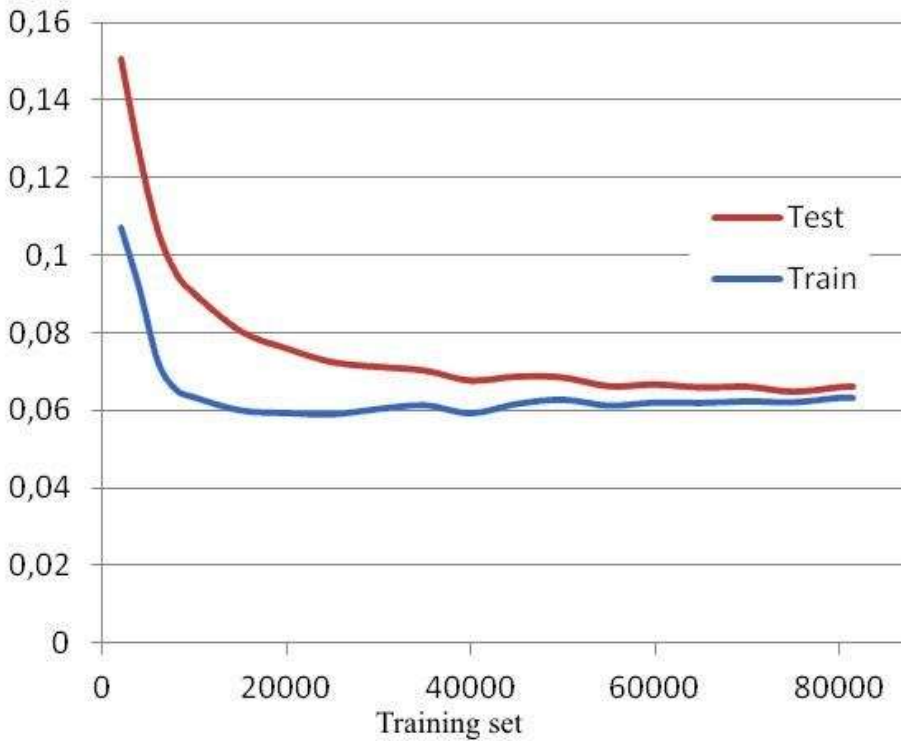


Рис. 36. Результаты обучения улучшенной нейронной сети (стандартная ошибка)

В результате была получена нейросеть с качеством:

- мера F1 на тренировочном наборе = 0.9943;
- мера F1 на тестовом наборе = 0.9903;
- стандартная ошибка на тренировочном наборе = 0.0282;
- стандартная ошибка на тестовом наборе = 0.0368.

Для численной оценки точности распознавания были написаны тесты. На каждом изображении из набора для проверки (см. п. 3.5) была размечена область с показаниями, затем эта область сдвигалась в различных направлениях 9 раз, на выходе получалось 10 различных вариантов области. С помощью различных положений областей показаний моделировалось съемка счетчика с разных ракурсов. Итоговый результат распознавания представлен в таблицах 2 и 3.

Следующие показатели были приняты за оценку точности (precision) и полноты (recall):

$$Precision = \frac{N_{all\ found} - N_{wrong}}{N_{all\ found}}, \quad Recall = \frac{N_{all\ found} - N_{wrong}}{N_{all} - N_{wrong}},$$

где $N_{all\ found}$ – все найденные числа в показаниях счетчиков; $N_{wrong}$ – неверно распознанные числа в показаниях счетчиков; $N_{all}$ – количество всех чисел в показаниях счетчиков.

Таблица 2 - Результаты оценки точности и полноты распознавания показаний аналоговых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 212 | 198 |
| $N_{all\_found}$ | 168 | 159 |
| $N_{wrong}$ | 33 | 28 |
| *Precision* | 0,804 | 0,824 |
| *Recall* | 0,754 | 0,771 |

Таблица 3 - Результаты оценки точности и полноты распознавания показаний цифровых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 378 | 290 |
| $N_{all\_found}$ | 319 | 265 |
| $N_{wrong}$ | 43 | 22 |
| *Precision* | 0,865 | 0,917 |
| *Recall* | 0,824 | 0,907 |

Хотя алгоритм способен обрабатывать достаточно сложные изображения, см. рис. 37, тем не менее точность и полнота такого подхода получились не самыми впечатляющими. Так усредненная точность составила (80.4% + 86.5%) / 2 = 83.5%, а усредненная полнота: (75.4% + 82.4%) / 2 = 78.9%.


Рис. 37. Пример успешного распознавания

Время выполнения алгоритма на одной фотографии и одном варианте положения области показаний на высокопроизводительном сервере без нагрузки в среднем колеблется в районе 0.5 секунд. Тем не менее, при увеличении числа запросов, время ответа от сервера начинает возрастать. Как можно наблюдать на рис. 38 при единовременном выполнении на одном даже очень производительном сервере распознавания большой группой людей время распознавания увеличивается в разы и может достигать значения более минуты! Время выполнения не выходит за пределы 10 минут (600 секунд), т.к. после 10 минут бездействия связь с сервером обрывается.

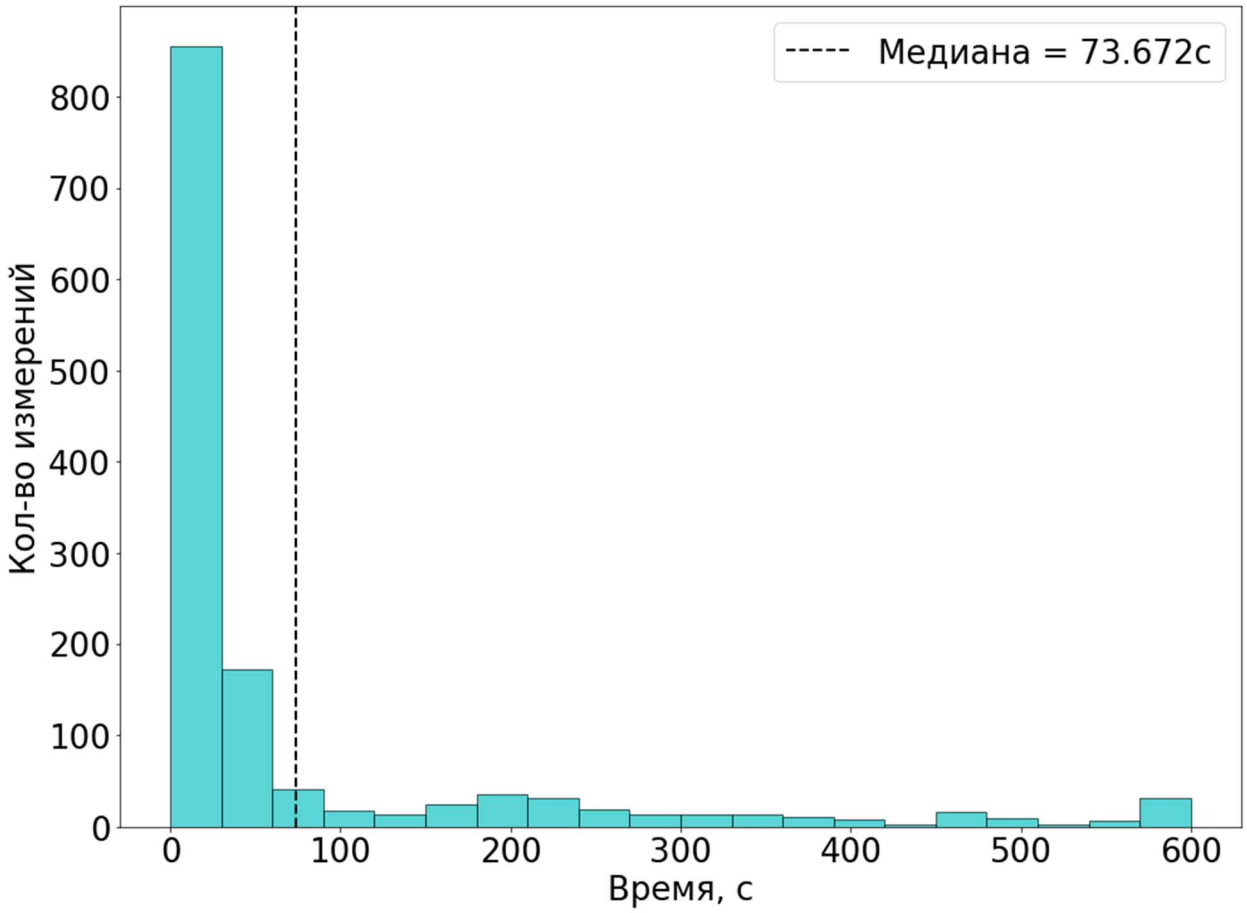


Рис. 38. Распределение времени распознавания показаний счетчика у централизованной архитектуры

Распознавание показаний жидкокристаллических производились аналогично централизованной модели из предыдущего пункта. т.е. путем сравнения контуров показаний с эталонными контурами. При распознавании цифр показаний процесс обработки изображения также похож на аналогичный процесс у централизованной архитектуры из предыдущего пункта. Разница заключается лишь в обучающем наборе для нейронной сети на основе архитектуры LeNet. Обучение нейронной сети проводилось в среде Python на базе библиотек Keras и Tensorflow на ПК под управлением Windows 7 со следующими характеристиками: Intel Core i5-3450/16Gb RAM/1000Gb HDD. Результаты обучения представлены на рис. 39 и 40.

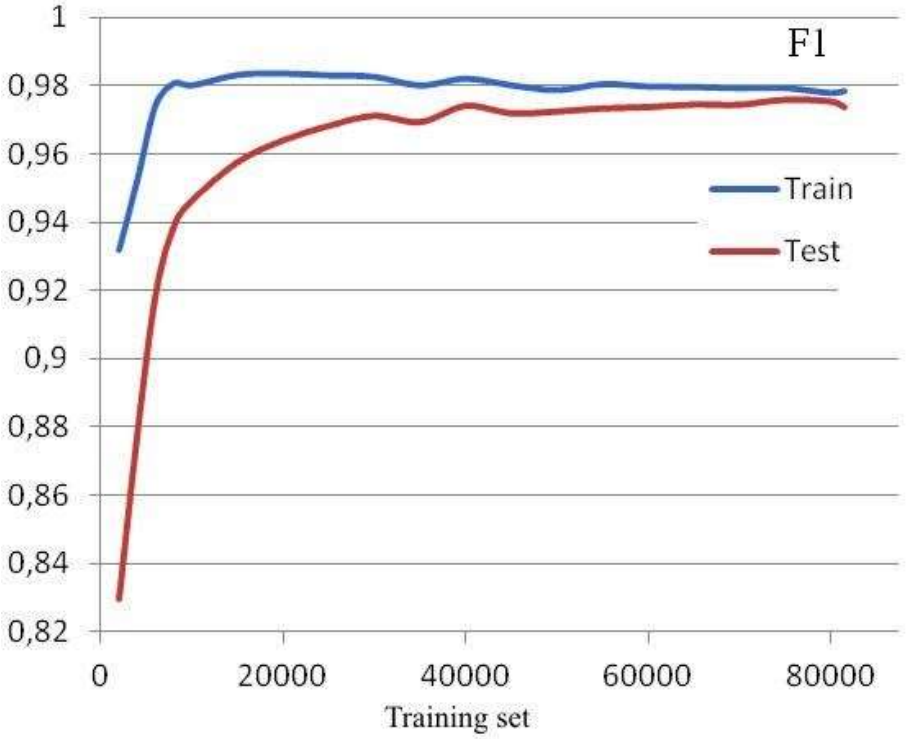


Рис. 39. Результаты обучения улучшенной нейронной сети (F1)

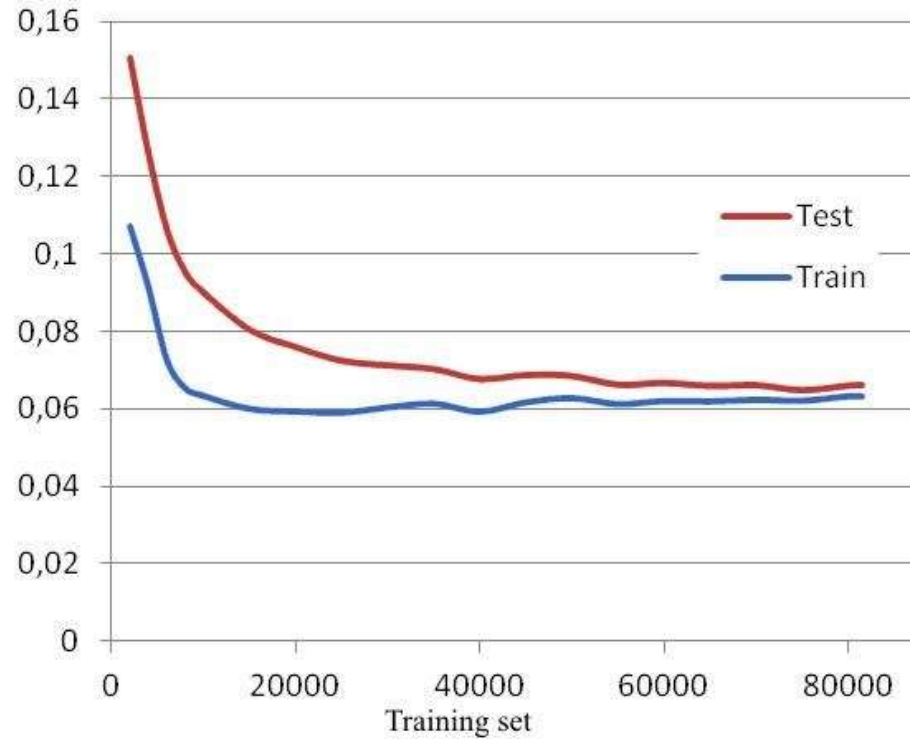


Рис. 40. Результаты обучения улучшенной нейронной сети (стандартная ошибка)

По итогам обучения была получена нейросеть с качеством:

- мера F1 на тренировочном наборе = 0.9943;
- мера F1 на тестовом наборе = 0.9903;
- стандартная ошибка на тренировочном наборе = 0.0282;
- стандартная ошибка на тестовом наборе = 0.0368.

Результаты оценки точности и полноты представлены в таблицах 4 и 5.

Таблица 4. Результаты оценки точности и полноты распознавания показаний аналоговых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 212 | 198 |
| $N_{all\_found}$ | 178 | 172 |
| $N_{wrong}$ | 25 | 22 |
| *Precision* | 0,860 | 0,872 |
| *Recall* | 0,818 | 0,852 |

Таблица 5. Результаты оценки точности и полноты распознавания показаний цифровых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 378 | 290 |
| $N_{all\_found}$ | 331 | 268 |
| $N_{wrong}$ | 36 | 16 |
| *Precision* | 0,891 | 0,940 |
| *Recall* | 0,863 | 0,920 |

Время выполнения алгоритма на различных мобильных устройствах колеблется в довольно широком диапазоне от 0,1с до 35с, см. рис. 41. В среднем время распознавания занимает порядка 4с, что является огромным преимуществом по сравнению с централизованной архитектурой.

Таким образом, решение с распределенной архитектурой устраняет недостатки, свойственные распознаванию единичной фотографии, позволяя пользователю перемещать мобильное устройство, чтобы найти положение камеры, когда показания счетчика видны достаточно хорошо для распознавания.

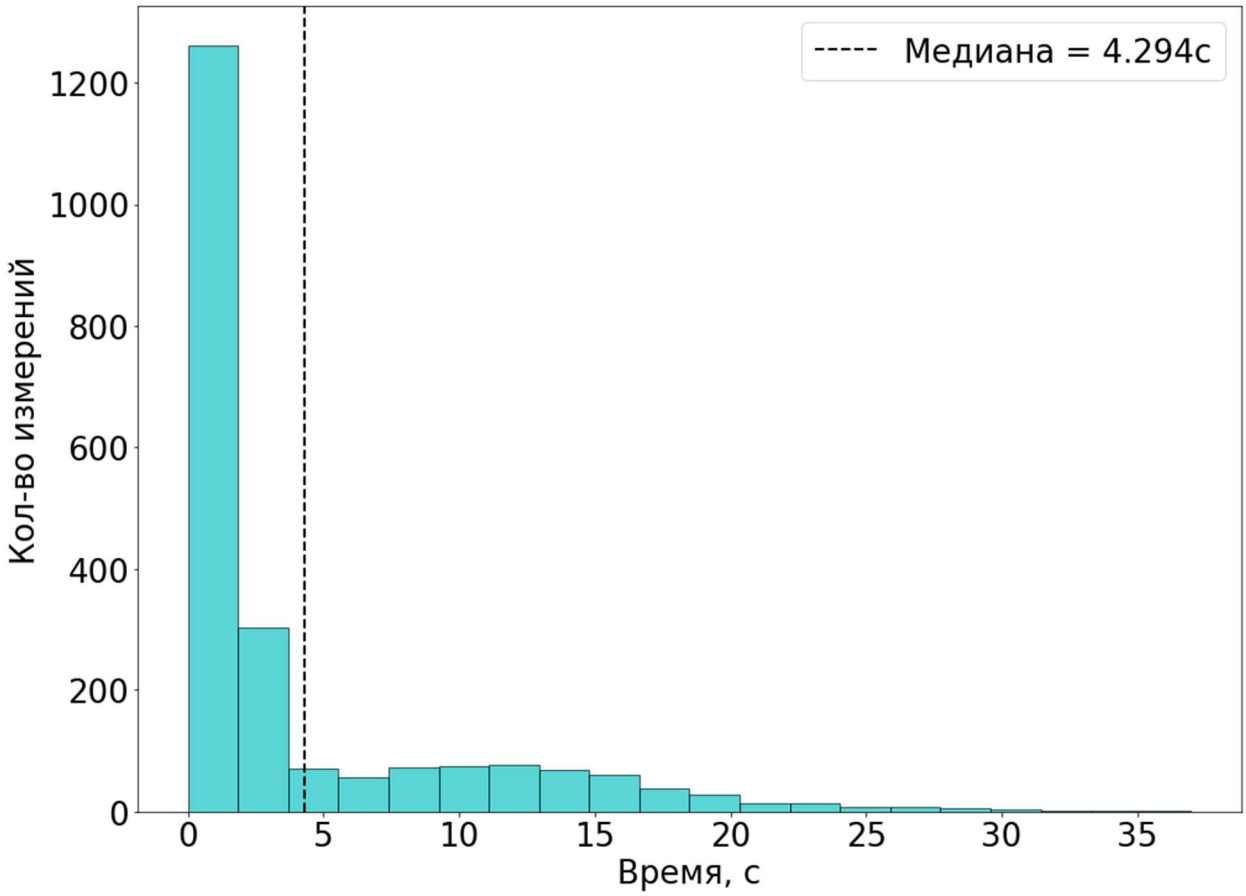


Рис. 41. Распределение времени распознавания показаний счетчика у распределенной архитектуры

### 4.3 Обучение интеллектуального диспетчера IMatcher

Обучение проводилось в среде Python на базе библиотек Keras и Tensorflow на ПК под управлением Windows 7 со следующими характеристиками: Intel Core i5-3450/16Gb RAM/1000Gb HDD. Все основные параметры модели нейронной сети были сохранены, как и в исходной модели YOLOv2.

В качестве функции оптимизации был выбран стохастический градиентный спуск со значениями параметров learning rate = 5e-3, decay = 0.0005, momentum = 0.9. Функция потерь (Loss function) была взята из оригинальной работы по модели YOLO. Исходные веса были импортированы из нейросети, натренированной на датасете COCO. Обучение проводилось в течение 100 эпох. В случае, если в течение 10 эпох не наблюдалось уменьшения значения функции потерь на тестовой части датасета (Test set), то обучение прерывалось. Пример графика изменения функции потерь в процессе обучения для 88 эпох показан на рис. 42.

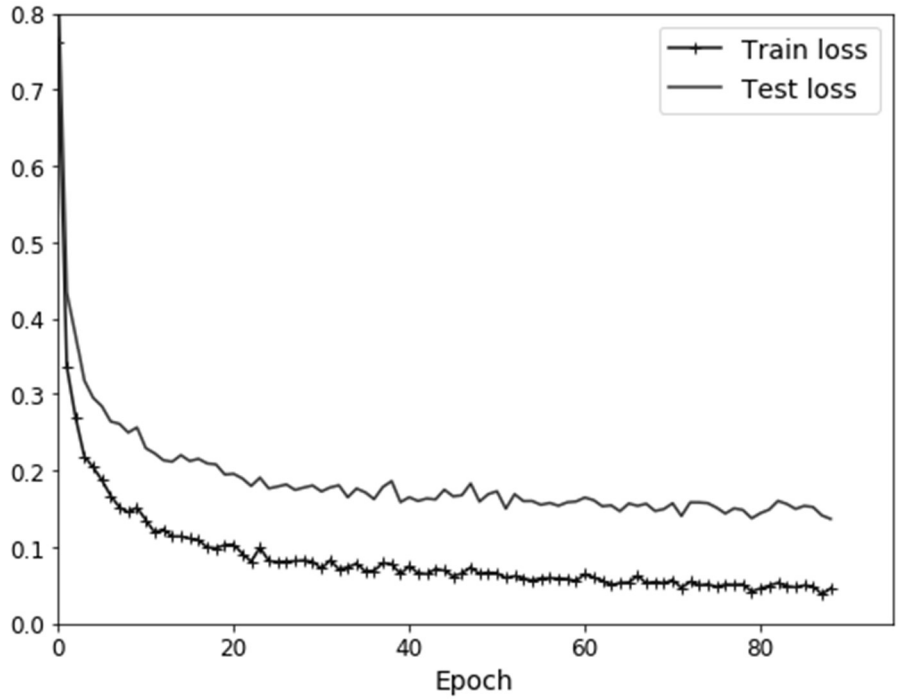


Рис. 42. Функции потерь для обучающего и тестового набора данных

Проверка работоспособности нейронной сети происходила на 87 реальных изображениях счетчиков (Valid set), данные изображения были распознаны обученной нейронной сетью. В результате распознавания на изображения были нанесены все цифры с вероятностью распознавания более 50%. Полученные примеры распознавания счетчиков приведены на рис. 43.


Рис. 43. Примеры распознавания показаний счетчиков

В результате обучения нейронной сети точность на тестовом датасете достигала значения 96%, а полнота - 90%.

Новое решение имеет следующие особенности:

1) Показания все еще плохо распознаются в условиях низкой освещенности и / или при некачественной съемке;

2) Из-за разных шрифтов, используемых в показаниях счетчиков, нейросеть иногда может путать похожие цифры (например, 3 и 5, 1 и 7);

3) Реализация системы на мобильных устройствах средней мощности под управлением ОС Android показывает, что скорость обработки одного кадра со счетчиком может составлять 1 - 3 секунды, что неприемлемо для реальных приложений;

4) Хотя достигнутые на данной нейронной сети точность и полнота удовлетворительны, но ее размер слишком велик для использования на мобильном устройстве (400 МБ).

Для решения задач со скоростью и размером в сверточных слоях нейронной сети количество ядер свертки было уменьшено в 4 раза, что привело к снижению параметров примерно в 16 раз: с 50 миллионов до 3,2 миллиона. Уменьшение параметров сделало невозможным использование трансферного обучения. Инициализатор по умолчанию в Tensorflow 1.13.1 (glorot_uniform_initializer) использовался для инициализации весов. Остальные параметры нейронной сети остались прежними.

Результаты обучения и распознавания представлены в Таблице 6 и на рис. 44.

Таблица 6. Результаты распознавания облегченной нейронной сети на расширенной выборке

| | **100 000 Images** |
|---|---|
| $N_{all}$ | 434 |
| $N_{all\_found}$ | 357 |
| $N_{wrong}$ | 72 |
| *Precision* | 0.80 |
| *Recall* | 0.79 |

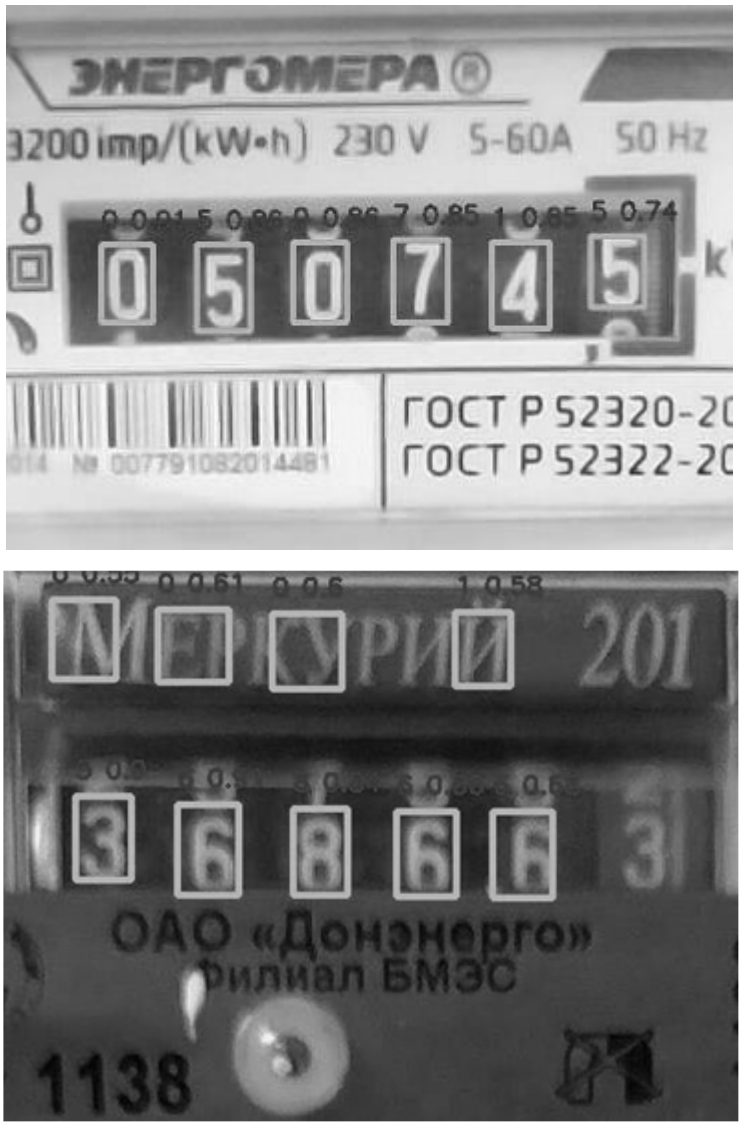


Рис. 44. Примеры распознавания облегченной нейронной сети на расширенной выборке

Несмотря на меньшую точность и полноту по сравнению с исходной нейронной сетью, это решение позволяет значительно увеличить скорость распознавания и уменьшить размер нейронной сети с 400 до 24,5 МБ, что позволяет использовать ее на мобильном устройстве.

Основные проблемы распознавания возникают с фотографиями очень низкого качества (см. рис. 45). Причин плохого качества изображения показаний счетчика может быть много, например, недостаточная фокусировка при съемке, случайные движения камеры во время съемки, блики от солнца и т. д.

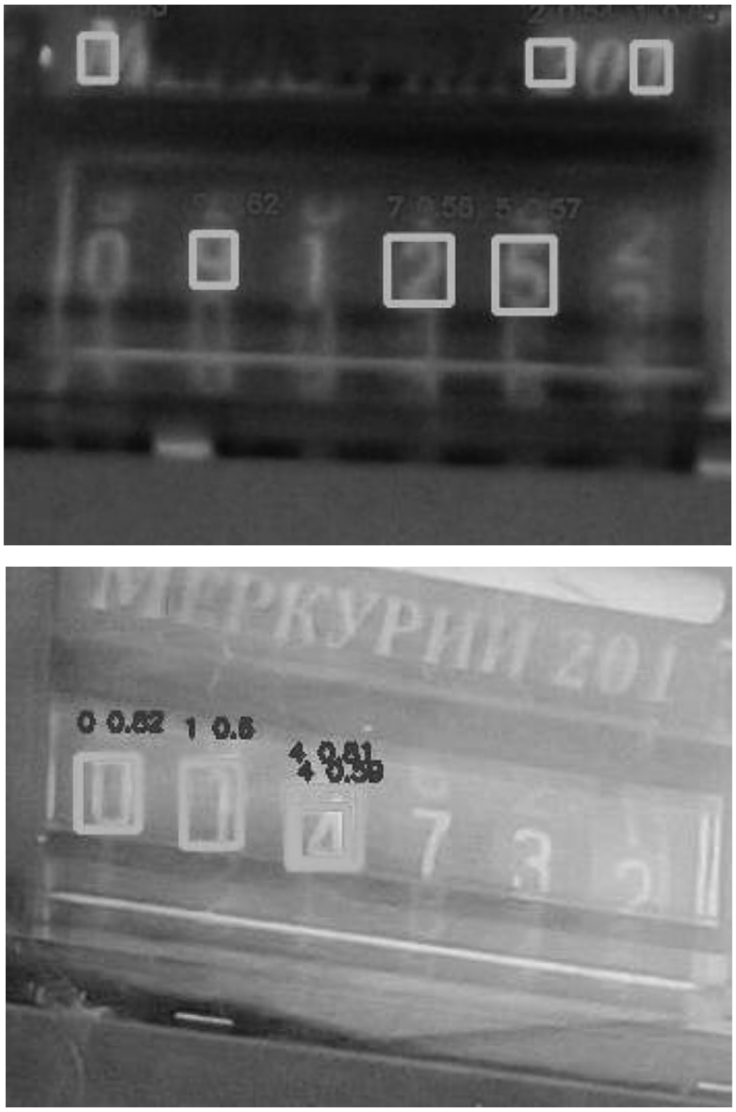


Рис. 45. Пример распознавания изображения низкого качества

Результаты оценки точности и полноты представлены в таблицах 7 и 8.

Таблица 7. Результаты оценки точности и полноты распознавания показаний аналоговых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 212 | 198 |
| $N_{all\_found}$ | 205 | 194 |
| $N_{wrong}$ | 12 | 8 |
| *Precision* | 0,941 | 0,959 |
| *Recall* | 0,965 | 0,979 |

Таблица 8. Результаты оценки точности и полноты распознавания показаний цифровых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 378 | 290 |
| $N_{all\_found}$ | 367 | 285 |
| $N_{wrong}$ | 8 | 4 |
| *Precision* | 0,978 | 0,986 |
| *Recall* | 0,970 | 0,983 |

Время выполнения алгоритма на различных мобильных устройствах колеблется в довольно широком диапазоне от 0,3с до 40с, см. рис. 46. В среднем время распознавания занимает порядка 5с.

Архитектура IMatcher обеспечивает дополнительные преимущества от объединения нескольких нейронных сетей в надежное интеллектуальное решение для распределенной фотосъемки. В отличие от других решений, оно позволяет интегрировать несколько распознавателей, обученных с использованием различных (интерсекционных) наборов, что делает его открытым для постоянной доработки путем добавления новых распознавателей без замены предыдущих или же с модификацией самого агента IMatcher-а.

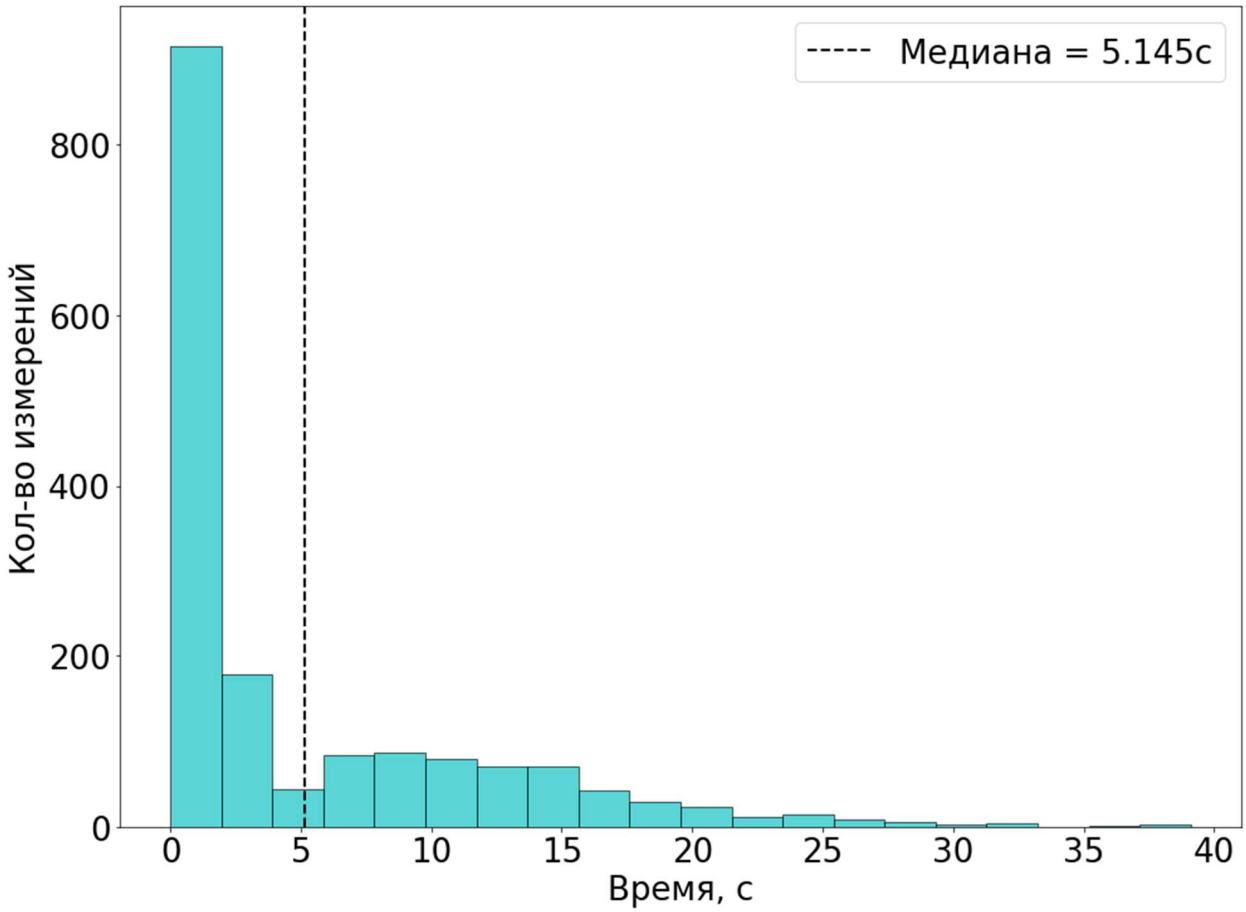


Рис. 46. Распределение времени распознавания показаний счетчика у архитектуры IMatcher

### 4.4 Обучение интеллектуального диспетчера RMatcher

В работе модели RMatcher у агентов применялись предобученные нейронные сети из модели IMatcher. Тренировке была подвергнута центральная составляющая модели RMatcher – рекуррентная нейронная сеть, описанная в п. 3.1. Обучение проходило в среде Python 3.8 на базе библиотек Keras и Tensorflow на ПК под управлением Windows 10 со следующими характеристиками: Intel Core i9-10900/32Gb RAM/512Gb SDD/ NVIDIA GeForce RTX 2070 SUPER 8Gb. Набор данных подробно описан в п. 3.4.2. После обучения в течение 87 эпох значение функции ошибки на тренировочном и тестовом наборе составили 0.165 и 0.04 соответственно, см. рис. 47.

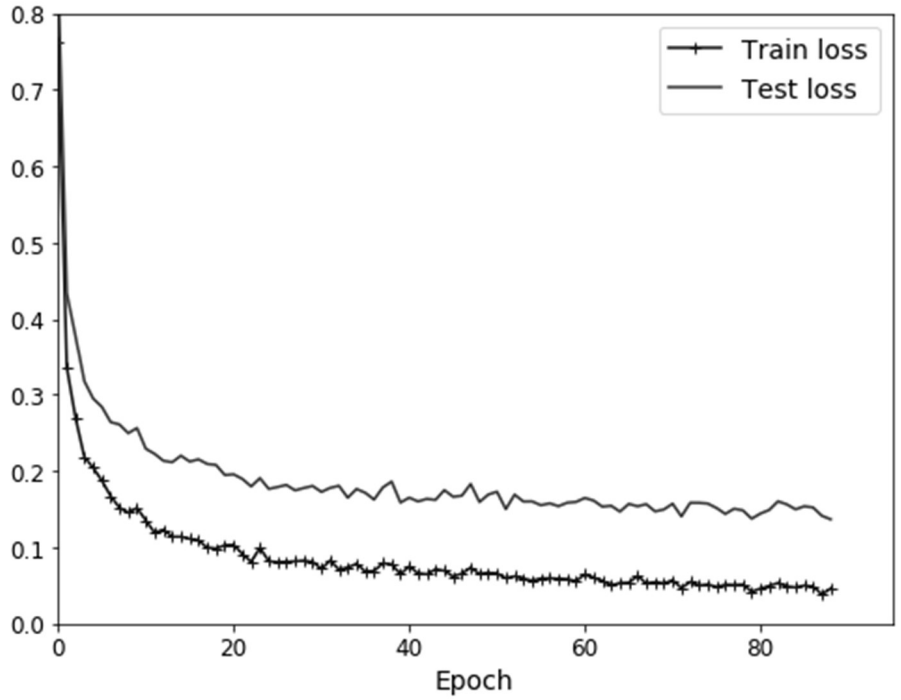


Рис. 47. График обучения рекуррентной нейронной сети для модели RMatcher

После обучения рекуррентной нейронной сети, модель RMatcher заменила модель IMatcher в архитектуре. Результаты распознавания на проверочном наборе представлены в Таблицах 9 и 10.

Таблица 9. Результаты оценки точности и полноты распознавания показаний аналоговых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 212 | 198 |
| $N_{all\_found}$ | 204 | 193 |
| $N_{wrong}$ | 14 | 10 |
| *Precision* | 0,931 | 0,948 |
| *Recall* | 0,960 | 0,973 |

Таблица 10. Результаты оценки точности и полноты распознавания показаний цифровых счетчиков на проверочной выборке

| | **Все показания (с учетом дробной части)** | **Только целочисленные показания** |
|---|---|---|
| $N_{all}$ | 378 | 290 |
| $N_{all\_found}$ | 365 | 284 |
| $N_{wrong}$ | 10 | 5 |
| *Precision* | 0,973 | 0,982 |
| *Recall* | 0,965 | 0,979 |

Время выполнения алгоритма на различных мобильных устройствах колеблется в довольно широком диапазоне от 0,2с до 30с, см. рис. 48. В среднем время распознавания занимает порядка 4с.

Архитектура RMatcher позволяет без уменьшения точности получать значительный выигрыш в производительности. В отличие от других решений, оно позволяет без снижения производительности либо добавлять дополнительные распознаватели, либо запускать программный комплекс на низкопроизводительных устройствах.

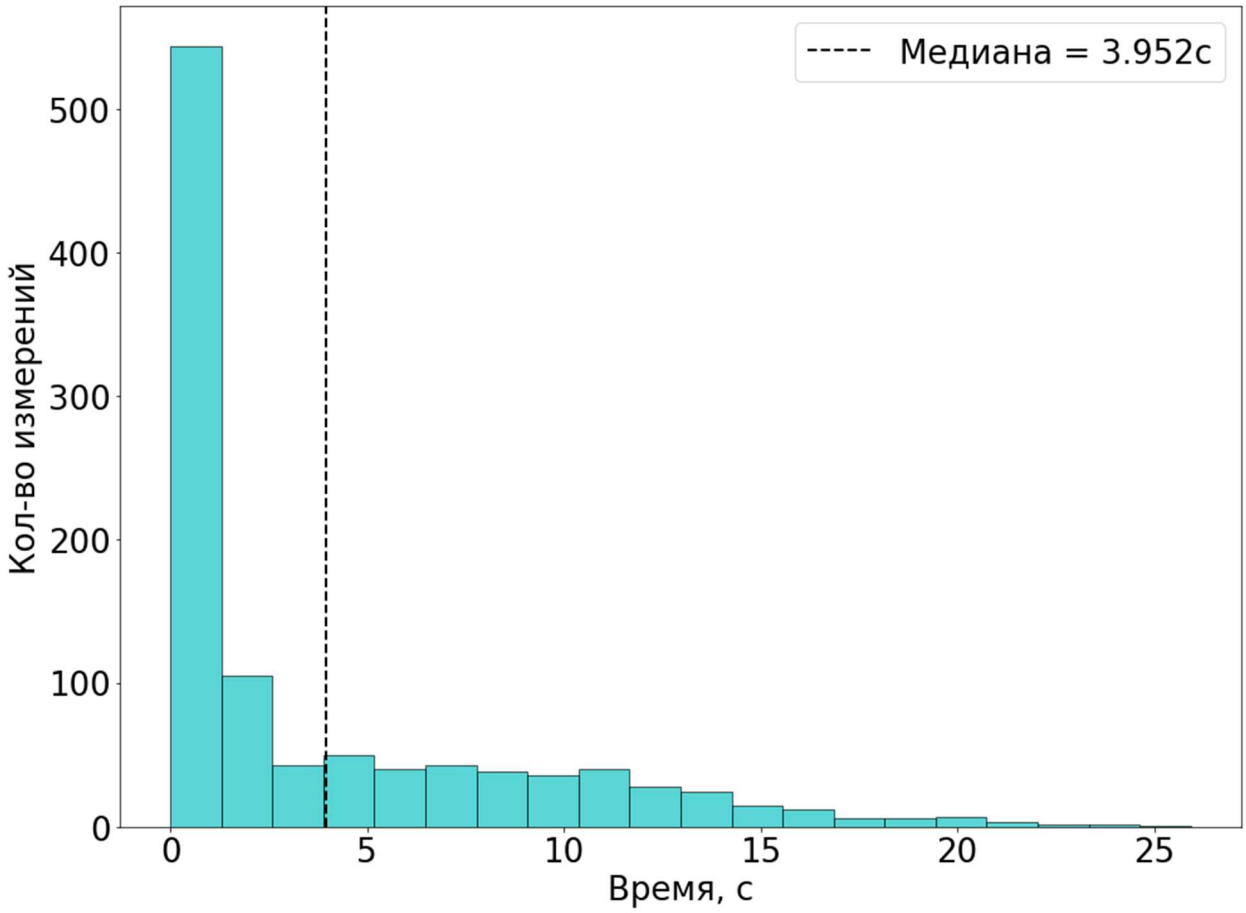


Рис. 48. Распределение времени распознавания показаний счетчика у архитектуры RMatcher

Сравнение качества распознавания различных моделей происходило на проверочном наборе, описанном в п. 3.5. Итоговые результаты сравнения точности и полноты распознавания показаний счетчиков из таблиц 2 - 10 приведены в таблице 11.

Таблица 11. Сравнение результатов работы модулей на проверочном наборе

| | Централизованная архитектура | Распределенная архитектура | Архитектура IMatcher | Архитектура RMatcher |
|---|---|---|---|---|
| | Точность | | | |
| Аналоговые счетчики | 80,4% | 86,0% | 94,1% | 93,1% |
| Цифровые счетчики | 86,5% | 89,1% | 97,8% | 97,3% |
| | Полнота | | | |
| Аналоговые счетчики | 75,4% | 81,8% | 96,5% | 96,0% |
| Цифровые счетчики | 82,4% | 86,3% | 97,0% | 96,5% |

Явно заметен рос качества и полноты от централизованной модели до модели IMatcher, являющейся на текущий момент state-of-the-art, т.е. передовым, вариантом реализации архитектуры программного комплекса для распознавания показаний. Модель RMatcher же незначительно проигрывает модели IMatcher. Тем не менее у модели RMatcher есть одно явное преимущество – она более производительна, чем модель IMatcher, т.е. может считывать показания быстрее. В среднем RMatcher распознает показания почти на 25% быстрее IMatcher – 3.9с против 5.1 с, см. рис. 49. С учетом незначительной разницы в точности, применение модели RMatcher более предпочтительно.

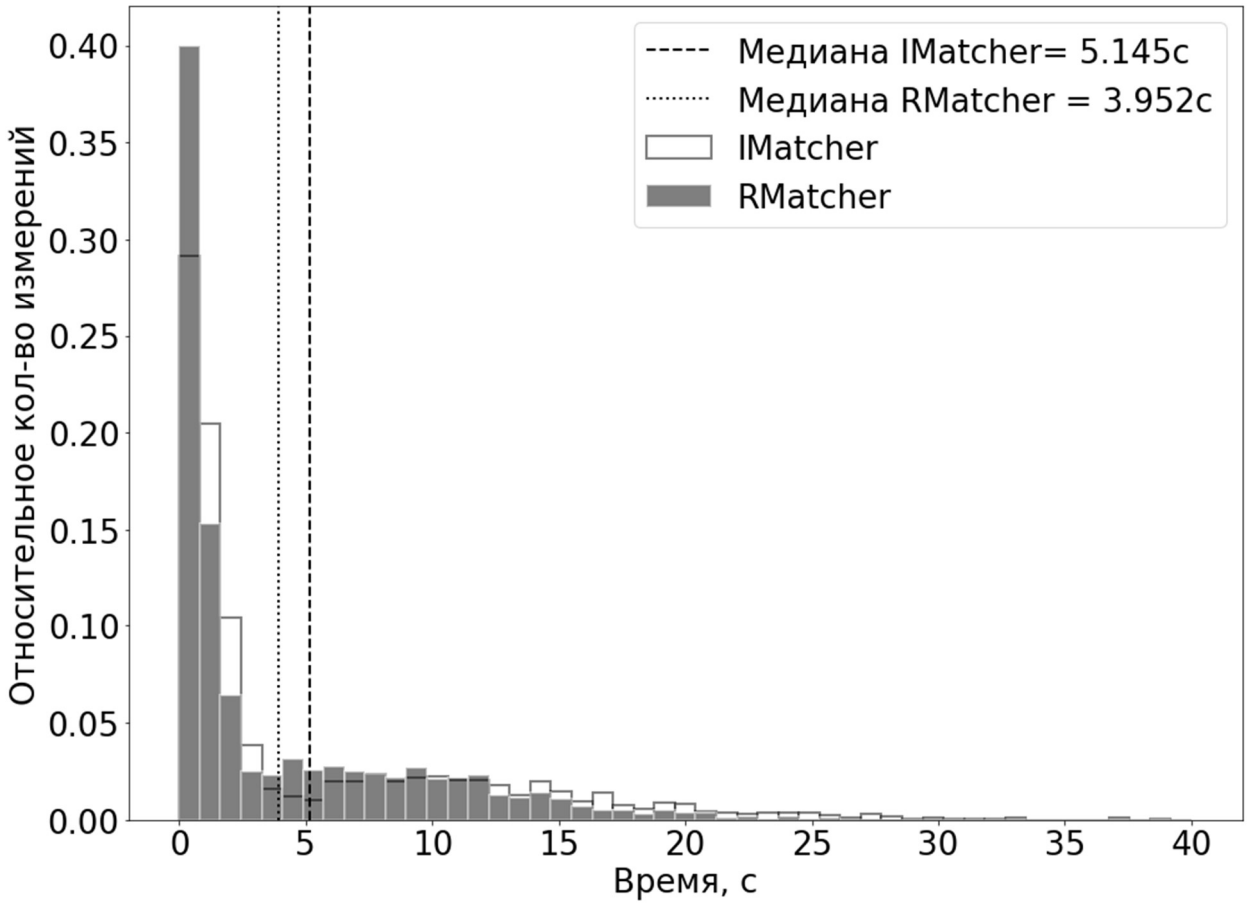


Рис. 49. Гистограмма распределения времени распознавания для архитектур IMatcher и RMatcher

Повышения производительности у модели RMatcher удалось добиться благодаря замене ряда жестких условий выбора агентов на одну рекуррентную нейронную сеть, запрограммированную выбирать агентов только тогда, когда это необходимо, в противном же случае просто пропускать кадр и ожидать следующий. Тем не менее у такого подхода есть и большой минус: нейронная сеть лишь учится не обрабатывать кадры, не приносящие дополнительных данных для распознавания показаний. А находить потенциально хорошие кадры для обработки агентами, которые по той или иной причине забраковала модель IMatcher, модель RMatcher не может, т.к. таких данных не было в обучающей выборке. Поэтому модель RMatcher и получается такой производительной, т.к. она обрабатывает

только «хорошие» кадры, а также незначительно менее точной, в результате редких ошибочных решений не обрабатывать «хорошие» кадры.

### 4.5 Адаптивная система распознавания образов при автоматизированной фиксации показателей электросчетчиков

Предложенное решение было использовано в специализированном мобильном приложении для фотографирования показаний электросчетчиков, передачи их в центр обработки данных, распознавания и оперативного анализа сотрудниками региональной энергораспределительной компании ДонЭнерго. В данной компании остро стояла проблема по регистрации показаний пользовательских приборов учета расхода электроэнергии. Например, ряд сотрудников компании, ответственных за сбор показаний, не всегда ответственно выполняли возложенные на них обязанности. Вместо личного осмотра каждого прибора они проставляли на бумаге значения показаний наобум без посещения адресов, что в дальнейшем приводило к нестыковке данных о реально затраченных объемов энергии с суммарными показаниями всех приборов учета. Помимо данной проблемы были ряд случаев, когда просрочки по платежам за электроэнергию доходили до суда, в процессе которого компания не могла предоставить фотографии прибора учета, что осложняло ход судебного процесса.

Данные проблемы смог решить программный комплекс, включавший в себя модуль централизованного распознавания (см. п. 3.3.1), после внедрения в эксплуатацию. Благодаря этому комплексу стало возможно отслеживать действия персонала и сохранять не только сами показания, но и изображения приборов учета с данными показаниями. Основной задачей модуля централизованного распознавания была помощь сотрудникам компании в автоматическом заполнении поля с показаниями. Сам процесс распознавания происходил на центральном сервере на основе изображений, снятых пользователями системы, см. рис. 50.


Рис. 50. Пользовательский интерфейс программного комплекса

После прохождения тестовой эксплуатации стало понятно, что централизованная архитектура не подходит для задачи распознавания показаний. Проблема оказалась в цикличности работы сотрудников, занимающихся обходами и съемом показаний приборов учета. В компании выделялось несколько дней в месяц для сбора показаний, когда многие сотрудники становились обходчиками и ходили по адресам. Помимо этого, сбор и распознавание показаний осуществлялось преимущественно в дневное время. Это привело к тому, что сервер для распознавания показаний работал в полную силу только несколько дней в месяц, помимо этого из-за большого количества запросов время распознавания одного прибора учета вырастало вплоть до бесконечности, т.е. пользователи системы не могли дождаться ответа от сервера. Часть ответов даже были отклонены по достижению таймаута в 10 минут, см. рис. 51.

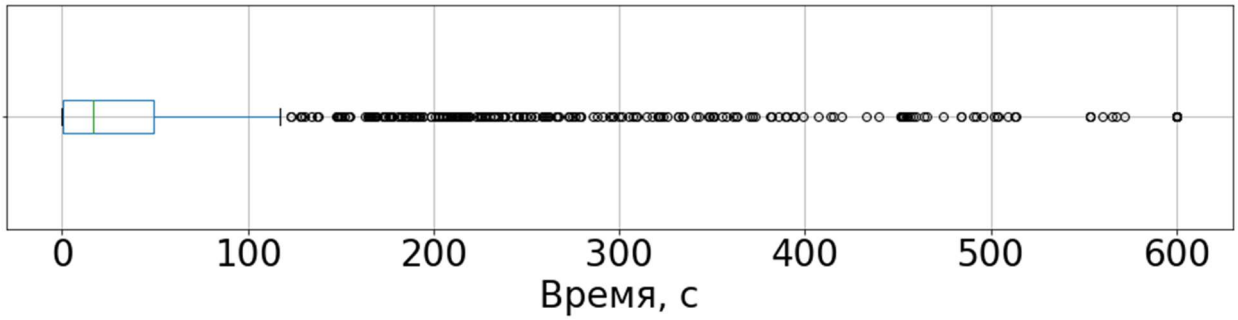


Рис. 51. Распределение времени распознавания централизованным модулем.

Увеличение количества высокопроизводительных серверов для снижения среднего времени распознавания не вписывалось в бизнес-стратегию компании ДонЭнерго. В связи с этим для решения проблемы низкой скорости распознавания сам процесс распознавания было решено перенести на мобильные устройства сотрудников компании. Для этого архитектура централизованного распознавания была облегчена и доработана для работы с видео потоком. Так в программном комплексе появился распределенный модуль распознавания (см. п. 3.3.2), работающий на мобильных устройствах обходчиков. Так среднее время распознавания уменьшилось более чем в 10 раз: с 74 секунд до 4,3 секунд, см. рис. 52. Еще одним плюсом данного модуля стала его масштабируемость: для подключения к программному комплексу филиалов предприятия не понадобилось закупать дополнительных серверов.

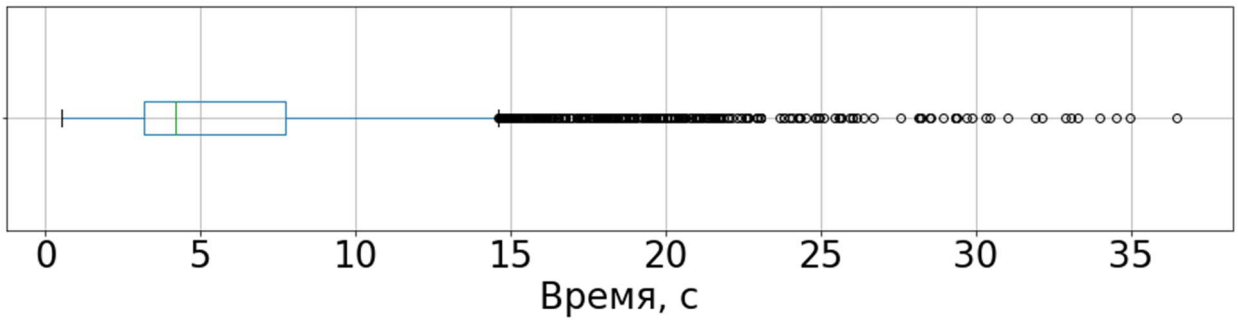


Рис. 52. Распределение времени распознавания распределенным модулем.

По прошествии времени конкуренты компании начали внедрять аналогичные программные комплексы, обладающие более высокими характеристиками. Для удержания лидирующих позиций на рынке компания ДонЭнерго решила обновить свой программный комплекс, в т.ч. улучшить

скорость и качество работы модуля распределенного распознавания показаний. Анализ актуальной литературы показал, что самым передовым решением для распределенного распознавания приборов учета является архитектура с использованием многоагентного подхода с агентами на основе современных нейронных сетей. Так в программном комплексе появился модуль распознавания показаний на основе архитектуры с интеллектуальным агентом-координатором (IMatcher), см. рис. 53 и п. 3.1. Данный модуль показал высокие показатели точности и производительности, по сравнению с предыдущим модулем. Хотя среднее время распознавания незначительно подросло с 4.3с до 5.1с, см. рис. 54, но точность и полнота распознавания выросли в среднем на 7-8%.

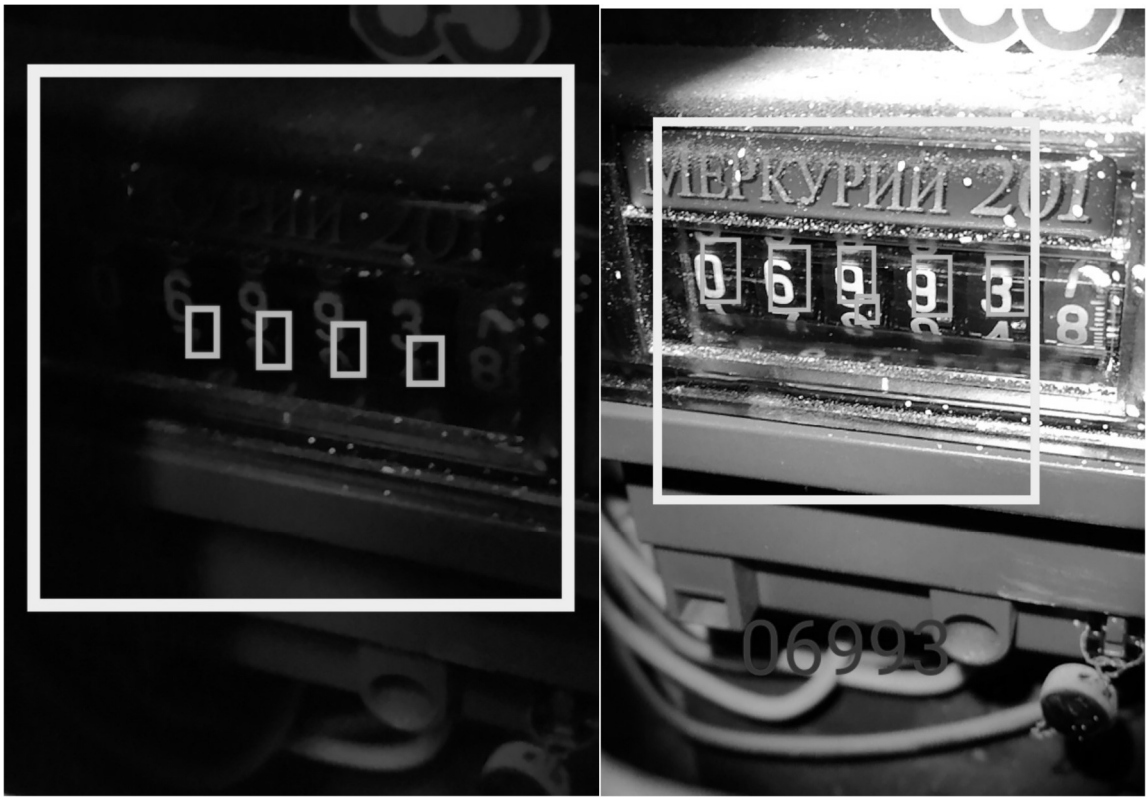


Рис. 53. Распознавание счетчика началось (слева) и закончено (справа) через 2 секунды после начала

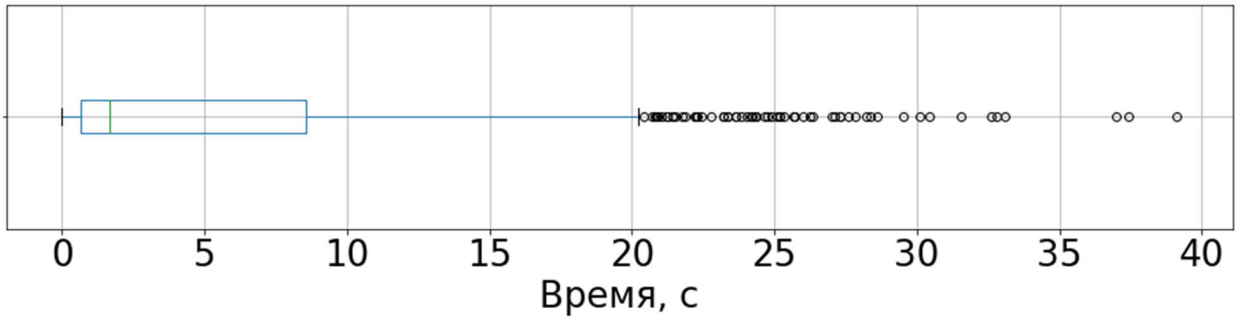


Рис. 54. Распределение времени распознавания модулем IMatcher.

Для закрепления положения компании на рынке работа над программным комплексом была продолжена. В контексте модуля распознавания показаний улучшению подверглась производительность процесса распознавания, для чего были проанализированы результаты работы модуля на основе архитектуры IMatcher, а затем, на основе этих результатов была обучена модель RMatcher. По итогам эксплуатации было подтверждено повышение производительности модуля, см. рис. 55, что позволило комфортно работать с программным комплексом даже на малопроизводительных мобильных устройствах.

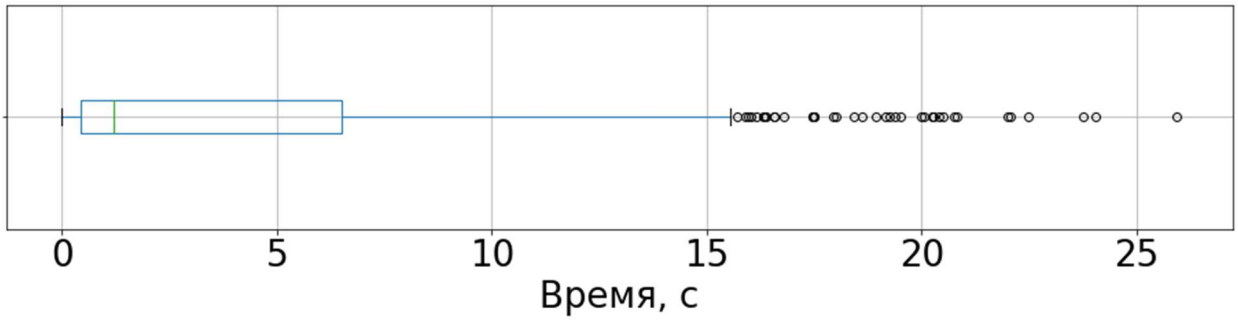


Рис. 55. Распределение времени распознавания модулем RMatcher.

## 4.6 Ансамблирование нейронных сетей распознавания образов в системе текстопонимания и тектогенерации

Апробация предложенного метода и архитектуры также была проведена при реализации интеллектуальной системы текстопонимания и текстогенерации для совместного использования нескольких искусственных нейронных сетей с разными обучающими наборами данных для комплексного решения различных задач семантического разбора текстов на русском языке.

Библиотека подпрограмм для текстопонимания и текстогенерации на основе машинного обучения предназначена для обработки и распознавания текстов и изображений, анализа и классификации различных типов корпоративных документов с использованием искусственных нейронных сетей. В основе решения лежат программные компоненты, разработанные на основе широкого опыта реализации систем семантического поиска и анализа больших объемов слабоструктурированной информации, больших данных, а также реализации систем электронного оборота в различных сферах применения.

Блок анализа документации состоит из следующих модулей:

- модуль семантического поиска по документу;
- модуль семантического поиска по ситуации;
- модуль извлечения атрибутов;
- модуль генерации документации;
- модуль классификации документации;
- модуль оптического распознавания документа;
- конфигуратор извлекаемых данных;
- библиотека правил для извлечения данных.

Ключевые возможности решения включают:

1. Распознавание документов, семантический разбор и выделение отдельных атрибутов;

2. Семантический поиск по запросам, сформулированным на естественном языке;
3. Классификация документов, нахождение одинаковых признаков и подбор документации в соответствии с текущей ситуацией;
4. Автоматическое заполнение карточки документов и автоматическая маршрутизация корреспонденции;
5. Автоматическое создание документов на основе имеющихся образцов в соответствии с новыми требованиями;
6. Сравнение документов и поиск семантически похожих фрагментов текста.

Внедрение блока анализа документации на базе библиотеки текстопонимания и тектогенерации позволяет обеспечить ряд уникальных преимуществ:

1. Совершенствование и роботизация процессов документооборота за счет применения передовых интеллектуальных технологий текстопонимания и текстогенерации;
2. Повышение удобства и адекватности поиска по слабоформализованным запросам;
3. Гармонизация нормативной документации, выявление и устранение расхождений и противоречий;
4. Устранение логических и семантических ошибок при разработке и вводе в действие новых нормативных документов.
5. Приведение нормативной документации в соответствие с законодательством.
6. Формирование различной аналитической отчетности по применению нормативной документации.
7. Постоянное повышение удобства работы с большими объемами документации за счет адаптации системы к действиям пользователей.

8. Сокращение времени работы с нормативной документацией и устранение негативного влияния человеческого фактора.

Примеры успешного внедрения подсистемы текстопонимания и текстогенерации на практике включают:

1. Разработка технологии извлечения данных из неструктурированных документов для автоматизации задач ввода данных в корпоративных учетных системах;
2. Интеллектуальная система анализа, экспертизы и генерации проектной документации;
3. Реализация системы по работе с нормативно-техническими документами Научно-технического центра;
4. Интеллектуальная система управления цифровым контентом;
5. Цифровая трансформация процессов оказания государственных услуг на базе системы ECM интеллект;
6. Мониторинг депрессивного и суицидального поведения детей и подростков с использованием современных технологий искусственного интеллекта.

Работа системы проиллюстрирована на Рис. 56. Основная проблема состояла в необходимости обработки различных текстовых документов с разной структурой и содержимым. Вместо создания единой интеллектуальной системы распознавания, обученной на объединенном наборе данных большого объема, был выбран вариант решения, основанный на комбинировании нескольких нейронных сетей, обученных для решения отдельных задач распознавания образов и реализации динамического алгоритма их совместного применения в зависимости от специфики обрабатываемого текста.

В качестве программного решения этой задачи был предложен шаблон программного агента, позволяющий находить новые решения по текстопониманию в процессе динамического комбинирования нейросетевых алгоритмов распознавания образов. В рамках программной реализации была

построена мультиагентная архитектура, в которой каждый новый вариант алгоритма распознавания был представлен отдельным программным агентом. Для организации мультиагентного взаимодействия была выбрана иерархическая архитектура программного решения.

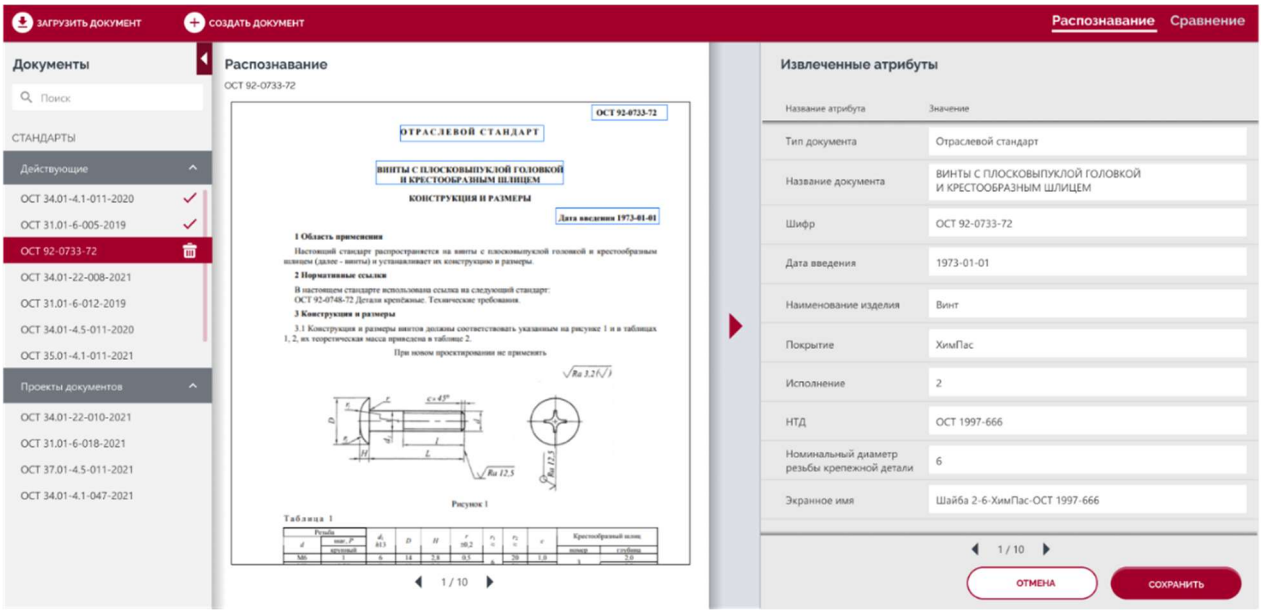


Рис. 56. Система текстопонимания и тектогенерации на основе ансамблирования нейронных сетей распознавания образов

Основное отличие представленного решения состоит в механизме ансамблирования агентов, основанном на иерархической организации с автоматической диспетчеризацией при отсутствии приоритета выбора на основе скорости или точности модели. Результат тестирования предложенного подхода на наборе из 200 документов представлен на Рис. 57. При загрузке всех доступных агентов для каждой задачи был получен наиболее точный результат с самой низкой производительностью. Применение методики мультиагентного ансамблирования позволяет существенно повысить призводительность при незначительном снижении точности распознавания. Подбор варианта ансамблирования позволяет еще приблизить точность к наилучшему значению.

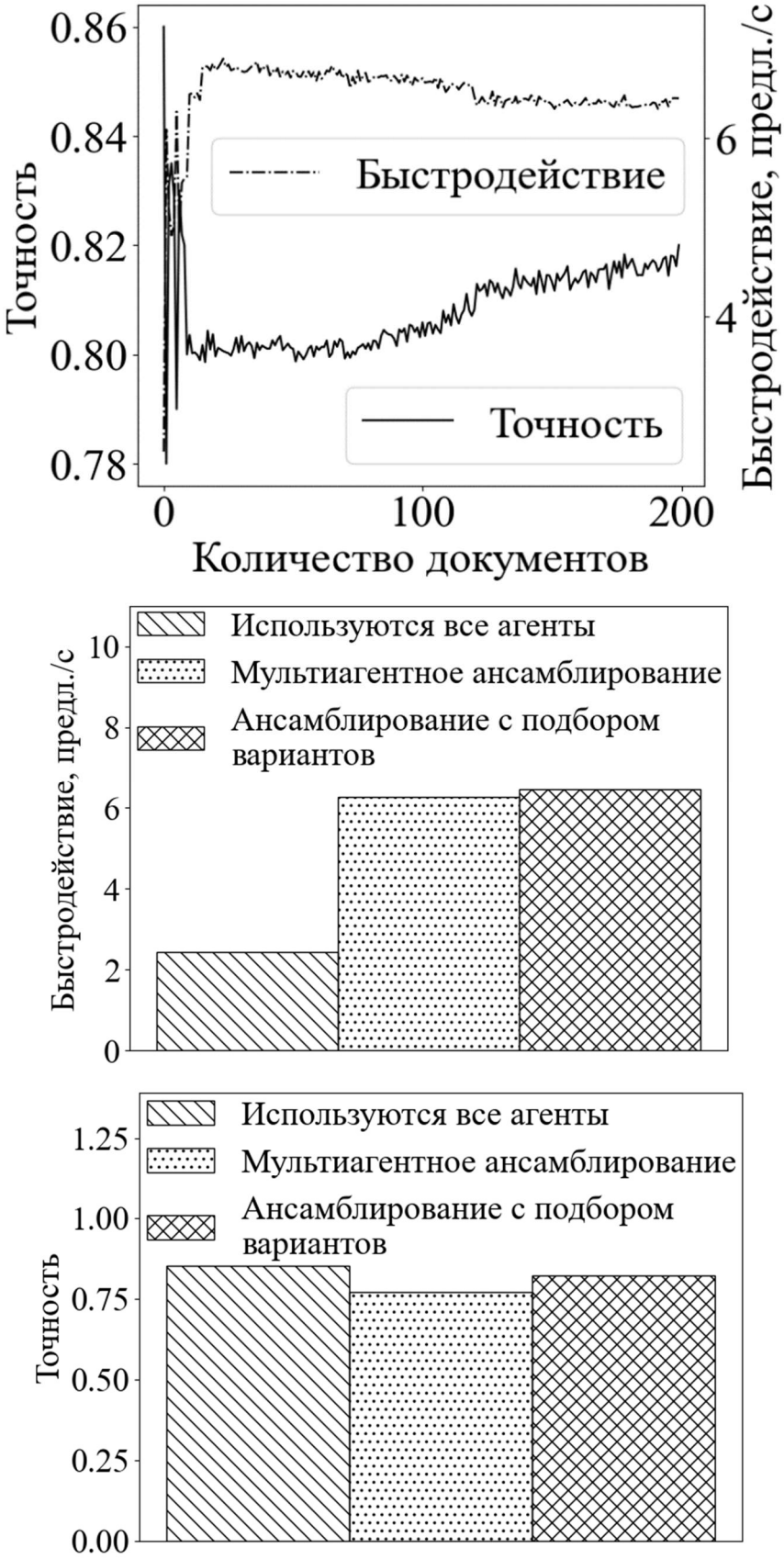


Рис. 57. Результаты тестирования модели и сравнительного анализа вариантов алгоритма ансамблирования

## 4.7 Выводы по Четвертой главе

В главе описаны как готовые датасеты изображений цифр, так и созданные вручную. Для создания искусственного набора данных используется специально созданный генератор. Создан датасет, содержащий исторические данные о работе агента-координатора на основе стратегии автоматической диспетчеризации служащий для последующей замены данного агента на рекуррентную нейронную сеть.

Приведены результаты обучения моделей нейронных сетей для агентов в различных архитектурах. Произведено их сравнение между собой. В результате, модель поведения агента-координатора на основе автоматической диспетчеризации по сравнению с остальными моделями поведения показывает лучшее быстродействие при практически неизменной точности. Реализация распределенной системы компьютерного зрения на базе мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов позволяет сократить на 25% время распознавания при автоматизированной фиксации показателей электросчетчиков. Применение ансамблирования интеллектуальных компонентов в рамках динамической диспетчеризации позволяет повысить производительность программного обеспечения библиотеки автоматизированного текстопонимания и текстогенерации в 2,5 раза по сравнению с классическими методами комбинирования нейронных сетей.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения диссертационного исследования получены следующие основные результаты:

1. В диссертации предложен метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов, отличающийся от аналогов реализацией динамического комплексирования автономных искусственных нейронных сетей, позволяющий обеспечить адаптивность системы в условиях изменяющейся обстановки без переобучения интеллектуальных компонентов.

2. Разработана архитектура мультиагентной системы распознавания образов, отличающаяся реализацией предиктивного оркестратора для согласования работы нескольких интеллектуальных агентов и позволяющая сократить время распознавания за счет более эффективного выбора агентов в отличие от классических моделей на основе ветвлений.

3. Представленная архитектура мультиагентной системы распознавания образов, основанная на реализации предиктивного оркестратора IMatcher или RMatcher, позволяет реализовать сочетание интеллектуальных компонентов с автономным поведением. В результате решена проблема комплексирования автономных искусственных нейронных сетей в интеллектуальной системе распознавания образов, способной адаптироваться к меняющимся внешним условиям эксплуатации. Использование нейронной сети в основе предиктивного оркестратора RMatcher позволяет выявлять скрытые паттерны (закономерности) во входящих данных и производить выбор агентов более эффективно, в отличие от классической модели IMatcher на основе условий (ветвлений).

4. Разработан мультиагентный алгоритм распределения задач в адаптивной системе распознавания образов, отличающийся от аналогов возможностью динамического изменения критериев выбора

интеллектуальных агентов при корректировке условий задачи распознавания образов и позволяющий повысить качество распознавания.

5. Реализована распределенная система компьютерного зрения на базе мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов, отличающаяся комбинированным применением искусственных нейронных сетей, предназначенных для решения частных задач и обеспечивающая адаптивность к меняющимся условиям эксплуатации без перенастройки алгоритмов распознавания.

6. Реализация распределенной системы компьютерного зрения на базе мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов позволяет сократить на 25% время распознавания при автоматизированной фиксации показателей электросчетчиков. Применение ансамблирования интеллектуальных компонентов в рамках динамической диспетчеризации позволяет повысить производительность программного обеспечения библиотеки автоматизированного текстопонимания и текстогенерации в 2,5 раза по сравнению с классическими методами комбинирования нейронных сетей.

Рекомендации и перспективы: предложенные в диссертации метод и архитектура рекомендуются к использованию в программном обеспечении с элементами искусственного интеллекта в области компьютерного зрения.

# СПИСОК ЛИТЕРАТУРЫ

1. Patel K., McCarthy M.P. Digital transformation: the essentials of e-business leadership, KPMG/McGraw-Hill. - 2000. - 134 p.

2. Singh M.P. Augmented reality interfaces. Natural web interfaces // IEEE Internet Computting, № 17. – 2013. - pp. 66 - 70

3. Dimov E.M., Maslov O.N. Information technologies of the digital economy: educational and research aspects // Infokommunikacionnye tehnologii, Vol. 17. № 1. – 2019. - pp. 100 - 115

4. Yuschenko A.S. Collaborative Robotics - State of Art and New Problems // Mekhatronika, Avtomatizatsiya, Upravlenie, № 18(12). – 2017. - pp. 812 – 819

5. Kelly III J. Computing, cognition and the future of knowing. IBM Research: Cognitive Computing. IBM Corporation. 2015. - 7 p.

6. Kagermann H., Wahlster W., Helbig J. Recommendations for implementing the strategic initiative Industrie 4.0. Final report of the Industrie 4.0 Working Group. 2013. - 82 p.

7. Lasi H., Kemper H.-G., Fettke P., Feld T., Hoffmann M. Industry 4.0 // Business & Information Systems Engineering. № 4(6). - 2014.- pp. 239 – 242

8. Wooldridge M. An introduction to multi-agent systems. John Wiley and Sons, Chichester. 2002. - 340 p.

9. Gorodetskii V.I. Self-organization and multiagent systems: I. Models of multiagent self-organization // Journal of Computer and Systems Sciences International, vol. 51. issue 2. – 2012. - pp. 256 – 281

10. Krevelen R. Augmented Reality: technologies, applications, and limitations. Vrije Universiteit Amsterdam, Department of Computer Science. 2007

11. Singh M., Singh M.P. Augmented Reality interfaces. Natural Web Interfaces IEEE Internet Computing. - 2013. - pp. 66 – 70

12. Holzinger A. Interactive machine learning for health informatics: when do we need the human-in-the-loop? // Brain Informatics, vol. 3. Issue 2. – 2016. - pp. 119 – 131

13. Ахтеров А.В., Кирильченко А.А. Основы теоретической робототехники. Искусственные нейронные сети. Обзор // Препринты ИПМ им. М.В. Келдыша. 2008. № 2. 20 с. URL: http://library.keldysh.ru/preprint.asp?id=2008-2

14. Сверточная нейронная сеть https://ru.wikipedia.org/wiki/Сверточная_нейронная_сеть (дата обращения 24.11.21)

15. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО "Альфа-книга": 2018. – 688 с.

16. Nwankpa C., Ijomah W., Gachagan A., Marshall S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning, 2020

17. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»)

18. Lecun Y. Generalization and network design strategies. Technical Report CRG-TR-89-4. Department of Computer Science, University of Toronto. 1989

19. LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D. Backpropagation Applied to Handwritten Zip Code Recognition // Neural Computation. 1 (4). – 1989 – pp. 541 – 551

20. LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jacker L.D. Handwritten digit recognition with a back-propagation network // Advances in Neural Information Processing Systems. 2. – 1990. – pp. 396 – 404

21. Lecun Y., Bottou L., Bengio Y., Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 86 (11). – 1998. - pp. 2278–2324

22. Еремеев В. А., Макаренков А. А. Использование сверточных нейронных сетей для идентификации структурно-однородных областей на космических снимках Земли. Цифровая обработка сигналов № 3. – 2022. – с. 45-48.

23. Pt B., Subashini P. Optimization of image processing techniques using neural networks – A review, WSEAS Transactions on Information Science and Applications, 8 (8). – 2011. - pp. 300-328

24. Jena M., Mishra S., Review of neural network techniques in the verge of image processing, Advances in Intelligent Systems and Computing, vol 628. Springer Singapore. – 2018. - pp. 345-361

25. Goodfellow I., Bengio Y., Courville A., Bengio Y. Deep learning, MIT press Cambridge, Vol. 19. – 2018. - pp. 305-307

26. Orlov S.P., Girin R.V. Intelligent technologies in the diagnostics using object's visual images. Studies in Systems, Decision and Control, vol. 259. Springer Nature Switzerland. – 2020. - pp. 301-312

27. Zhao Z., Zheng P., Xu S., Wu X., Object detection with deep learning: a review, IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11. – 2019. - pp. 3212-3232

28. Arcos-Garcia A., Alvarez-Garcia J., Soria Morillo L. Evaluation of deep neural networks for traffic sign detection systems, Neurocomputing, vol. 316. – 2018. - pp. 332-344

29. Pi Y., Nath N., Behzadan A. Convolutional neural networks for object detection in aerial imagery for disaster response and recovery. Advanced Engineering Informatics 43. 101009. – 2020. – 14 p.

30. Рекуррентная нейронная сеть https://ru.wikipedia.org/wiki/Рекуррентная_нейронная_сеть (дата обращения 24.11.21)

31. Hochreiter S., Schmidhuber J. Long Short-Term Memory. Neural Computation 9 (8). – 1997. - pp. 1735–1780

32. Sak H., Senior A. W., Beaufays F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proc. Interspeech 2014. – 2014. - pp. 338–342

33. Zaremba W., Sutskever I., Vinyals O. Recurrent neural network regularization. arXiv preprint. https://arxiv.org/pdf/1409.2329v5 (дата обращения 24.12.21)

34. ISO/IEC/IEEE 24765-2010(E), Systems and software engineering – Vocabulary. IT-Regie – The business of IT for the business. https://www.cse.msu.edu/~cse435/Handouts/Standards/IEEE24765.pdf (дата обращения 24.12.21)

35. Мудрик А. В., Кандыбович Л. А. Менеджмент знаний. Терминологический словарь-справочник. Минск. – 2010. – 752 с.

36. Маевский С.С. Термин «база знаний» в контексте разработки базы знаний адаптивного ЭОР по русскому языку. Мир науки, культуры, образования, №5(78). – 2019. – с. 411-414

37. Marvis U., Shade K., Atisi F., Olayiwola B. Overview of knowledge-based system, International Journal of Computer & Communication Engineering Research vol. 2, no.3. – 2024. - pp. 125-129

38. Assawamakin A., Chalortham N., Ruangrajitpakorn T., Limwongse C., Supnithi T., Tongsima S. A development of knowledge representation for thalassemia prevention and control program, 7th International Conference on Natural Language Processing and Knowledge Engineering. – 2011. – pp. 190-193

39. Wennerberg P. O. Ontology based knowledge discovery in social networks JRC Joint Research Center, European Commission, Institute for the Protection and Security of the Citizen (IPSC), Tech. Rep. – 2006. – pp. 489-498

40. Bench-Capon T. J. M. Knowledge-based systems and legal applications. vol. 36. Academic Press. 2014. – 264 p.

41. Liu S., Zaraté P. Knowledge based decision support systems: A survey on technologies and application domains. Lecture Notes in Business Information Processing. – 2014. - pp. 62–72

42. Ligęza A., Nalepa G. J. A study of methodological issues in design and development of rule-based systems: proposal of a new approach. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 1, no. 2. – 2011. - pp. 117-137

43. A. Mehmood, A. Ghafoor, H. F. Ahmed, Z. Iqbal, Adaptive transport protocols in multi agent system. Fifth International Conference on Information Technology: New Generations. – 2008. - pp. 720-725

44. Mehmood A., Khan S., Shams B., Lloret J. Energy-efficient multi-level and distance-aware clustering mechanism for WSNs. International Journal of Communication Systems, vol. 28, no. 5. – 2013. - pp. 972–989

45. Легович Ю. С., Максимов Д. Ю. Выбор исполнителя в группе интеллектуальных агентов, УБС, 56. – 2015. – с. 78–94

46. Городецкий В. И., Бухвалов О. Л., Скобелев П. О. Современное состояние и перспективы индустриальных применений многоагентных систем. Управление большими системами: сборник трудов № 66. – 2017. – с. 94-157.

47. Городецкий В. И. Теория, модели, инфраструктуры и языки спецификации командного поведения автономных агентов. Обзор (Часть 1). Искусственный интеллект и принятие решений № 2. – 2011. – с. 19-30.

48. Скобелев П. О. Интеллектуальные системы управления ресурсами в реальном времени: принципы разработки, опыт промышленных внедрений и перспективы развития. Информационные технологии № S1. – 2013. – с. 1-32.

49. Горшков А. В., Кравец О. Я. Исследование механизма распространения информации в мультиагентной системе во временном окне. Моделирование, оптимизация и информационные технологии № 1, Т. 11. – 2023. – с. 15-16.

50. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. Artificial Intelligence, Volume 175, Issues 5–6. – 2011. - pp. 935-937

51. Jain L. C., Srinivasan D. Innovations in Multi-Agent Systems and Application. Springer. – 2010. – 312 p.

52. Ye D., Zhang M., Vasilakos A. V. A survey of self-organization mechanisms in multiagent systems. IEEE Trans. Syst., Man, Cybern., Syst., vol. 47, no. 3. – 2017. - pp. 441–461

53. Dorri A., Kanhere S., Jurdak R. Multi-Agent Systems: A survey. IEEE Access. Vol 6. – 2018. - pp. 28573-28593

54. Городецкий В. И., Карсаев О. В., Самойлов В. В., Серебряков С. В. Инструментальные средства для открытых сетей агентов. Известия Российской академии наук. Теория и системы управления № 3. – 2008. – с. 106-124.

55. Городецкий В.И., Грушинский М.С., Хабалов А.В. Многоагентные системы (обзор). Новости искусственного интеллекта № 2. – 1998. - с. 64 - 116.

56. Wooldridge M. An Introduction to Multiagent Systems. John Wiley and Sons. – 2009. – 461 p.

57. Иващенко А. В., Карсаев О. В., Скобелев П. О., Царев А. В., Юсупов Р. М. Мультиагентные технологии для разработки сетецентрических систем управления. Известия ЮФУ. Технические науки, 116(3). – 2011. – с. 11-23.

58. Городецкий В. И., Бухвалов О. Л. Модель и архитектура инфраструктуры системы группового управления роботами. Робототехника и техническая кибернетика № 1(14). – 2017. – с. 33-44.

59. Koshy-Chenthittayil S., Archambault L., Senthilkumar D., Laubenbacher R., Mendes P., Dongari-Bagtzoglou A. Agent Based Models of Polymicrobial Biofilms and the Microbiome-A Review. Microorganisms 9(2). – 2021 – p. 417

60. Черненький В. М., Терехов В. И., Гапанюк Ю. Е. Структура гибридной интеллектуальной информационной системы на основе метаграфов. Нейрокомпьютеры: разработка, применение № 9. – 2016. – с. 3-13.

61. Иващенко А. В., Карсаев О. В., Скобелев П. О., Царев А.В., Юсупов Р. М. Мультиагентные технологии для разработки сетецентрических систем управления. Известия ЮФУ. Технические науки № 3(116). – 2011. – с. 11-23.

62. Liu Q., Gao L., Lou P. Resource management based on multi-agent technology for cloud manufacturing. Electronics, Communications and Control (ICECC), 2011 International Conference on. IEEE. – 2011. - pp. 2821–2824.

63. Al-Shrouf F. M. Facilitator agent design pattern of procurement business systems. Computer Software and Applications. COMPSAC’08. 32nd Annual IEEE International. IEEE. – 2008. - pp. 505–510.

64. Городецкий В. И. Теория, модели, инфраструктуры и языки спецификации командного поведения автономных агентов. Обзор (Часть 1). Искусственный интеллект и принятие решений № 2. – 2011. – с. 19-30.

65. Городецкий В. И. Самоорганизация и многоагентные системы. I. модели многоагентной самоорганизации. Известия Российской академии наук. Теория и системы управления № 2. – 2012. — с. 92.

66. Ivaschenko A., Skobelev P., Khamits I., Sychova M. Multi-agent system for scheduling of flight program, cargo flow and resources of international space station. Lecture Notes in Computer Science. Vol. 6867. – 2011. – pp. 165-174.

67. Скобелев П. О., Соллогуб А. В., Иващенко А. В. Симонова Е. В., Степанов М. Е. Царев А. В. Решение задач дистанционного зондирования земли с применением мультиагентных технологийю Вестник Самарского государственного технического университета. Серия: Технические науки. № 3(28). – 2010. – с. 47-54.

68. Glaschenko A., Ivaschenko A., Skobelev P., Rzevski G. Multi-agent real time scheduling system for taxi companies. Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems. – 2009. – pp. 1320-1326.

69. Скобелев П. О., Иващенко А. В., Андреев М. В., Бабанин И. О. Мультиагентные технологии для управления распределением производственных ресурсов в реальном времени. Вычислительные технологии в естественных науках. Перспективные компьютерные системы: устройства, методы и концепции. Институт космических исследований Российской академии наук. – 2011. – с. 110-122.

70. Каляев, И. А., Гайдук А. Р., Капустян С. Г. Самоорганизация в мультиагентных системах. Известия ЮФУ. Технические наукиⅣ 3(104). – 2010. – с. 14-20.

71. Амелина Н. О. Мультиагентные технологии, адаптация, самоорганизация, достижение консенсуса. Стохастическая оптимизация в информатике Т. 7. – 2011. – с. 149-185.

72. Чеботарев П. Ю., Агаев Р. П. Об асимптотике в моделях консенсуса. Управление большими системами: сборник трудов – № 43. – 2013. – с. 55-77.

73. Olfati-Saber R., Murray R. M. ‘Consensus problems in networks ofagents with switching topology and time-delays. IEEE Trans. Autom. Control, vol. 49, no. 9. – 2004. - pp. 1520–1533

74. Olfati-Saber R. Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Trans. Autom. Control, vol. 51, no. 3. – 2006. - pp. 401–420

75. Городецкий В. И., Карсаев О. В., Самойлов В. В., Серебряков С. В. Прикладные многоагентные системы группового управления. Искусственный интеллект и принятие решений № 2. – 2009. – с. 3-24.

76. Reynolds C. W. Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH Comput. Graph., vol. 21, no. 4. – 1987. - pp. 25–34

77. Fu J., Wang J. Adaptive coordinated tracking of multi-agent systems with quantized information. Syst. Control Lett., vol. 74. – 2014. - pp. 115–125

78. Liu S., Xie L., Zhang H. Containment control of multi-agent systems by exploiting the control inputs of neighbors. Int. J. Robust Nonlinear Control, vol. 24, no. 17. – 2014. - pp. 2803–2818

79. Liu B., Su H., Li R., Sun D., Hu W. Switching controllability of discrete-time multi-agent systems with multiple leaders and time-delays. Appl. Math. Comput., vol. 228. – 2014. - pp. 571–588.

80. Liu B., Chu T., Wang L., Zuo Z., Chen G., Su H. Controllability of switching networks of multi-agent systems. Int. J. Robust Nonlinear Control, vol. 22, no. 6. – 2012. - pp. 630–644

81. Иващенко, А. В., Сюсин И. А. Сетевой принцип взаимодействия в мультиагентной системе построения программы полета, грузопотока и расчета ресурсов МКС. Теория активных систем, Труды международной научно-практической конференции том 3. – 2011. – с. 225-229.

82. Su S., Lin Z., Garcia A. Distributed synchronization control of multiagent systems with unknown nonlinearities. IEEE Trans. Cybern., vol. 46, no. 1. – 2016. - pp. 325–338

83. Проскурников А. В. Консенсус в нелинейных стационарных сетях с идентичными агентами. Автоматика и телемеханика № 9. – 2015. – с. 44-63.

84. Фишов А. Г., Карджаубаев Н. А. Децентрализованное мультиагентное регулирование напряжения в электрических сетях. Вестник Иркутского государственного технического университета Т. 22. № 6(137). – 2018. – с. 183-195

85. Ivaschenko A., Tsarev A., Vaysblat A., Skobelev P. Smart solutions multi-agent platform for dynamic transportation scheduling. ICAART 2011 - Proceedings of the 3rd International Conference on Agents and Artificial Intelligence. – 2011. – pp. 372-375.

86. Ерофеева В. А., Иванский Ю. В., Кияев В. И. Управление роем динамических объектов на базе мультиагентного подхода. Компьютерные инструменты в образовании № 6. – 2015. – с. 34-42.

87. Соллогуб А. В., Скобелев П. О., Симонова Е. В., Царев А.В., Степанов М.Е. Модели для решения сетецентрических задач планирования и управления групповыми операциями кластера малоразмерных космических аппаратов. Информационно-управляющие системы № 1(56). – 2012. – с. 33-38.

88. Chen Y. Q., Wang Z. Formation control: A review and a new consideration. Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS) – 2005, - pp. 3181–3186.

89. Xia Y., Na X., Sun Z., Chen J. Formation control and collision avoidance for multi-agent systems based on position estimation. ISA Trans., vol. 61. – 2016. - pp. 287–296

90. Иващенко А. В., Пейсахович Д. Г. Управление интерактивной диспетчеризацией ресурсов посреднического транспортного оператора. Системы управления и информационные технологии № 1-1(55). – 2014. – с. 151-155.

91. Ivaschenko A., Krivosheev A., Sitnikov P. Pre-launch AI matcher for distributed intelligent photo surveying. Procedia Computer Science. – 2021. – pp. 538-545

92. Ivaschenko A., Krivosheev A., Stolbova A., Golovnin O. Hybridization of intelligent solutions architecture for text understanding and text generation. Applied Sciences (Switzerland) Vol. 11. No 11. – 2021. – p. 5179

93. Еремеев А. П., Кожухов А. А., Голенков В. В., Гулякина Н. А. О реализации средств машинного обучения в интеллектуальных системах реального времени. Программные продукты и системы № 2. – 2018. – с. 239-245.

94. Hwang K.-S., Jiang W.-C., Chen Y.-J. Model learning and knowledge sharing for a multiagent system with Dyna-Q learning. IEEE Trans. Cybern., vol. 45, no. 5. – 2015. - pp. 978–990

95. Еремеев А. П., Кожухов А. А. Реализация методов обучения с подкреплением на основе темпоральных различий и мультиагентного подхода для интеллектуальных систем реального времени. Программные продукты и системы № 1. – 2017. – с. 28-33.

96. Bianchi R. A. C., Martins M. F., Ribeiro C. H. C., Costa A. H. R., Heuristically-accelerated multiagent reinforcement learning. IEEE Trans. Cybern., vol. 44, no. 2. – 2014. - pp. 252–265

97. Курейчик В. М., Родзин С. И. Эволюционные вычисления: генетическое и эволюционное программирование. Новости искусственного интеллекта № 5. – 2003. – с. 13-19.

98. Davoodi M. R., Khorasani K., Talebi H. A., Momeni H. R. Distributed fault detection and isolation filter design for a network of heterogeneous multiagent systems. IEEE Trans. Control Syst. Technol., vol. 22, no. 3. – 2014. - pp. 1061–1069

99. Liu X., Gao X., Han J. Robust unknown input observer based fault detection for high-order multi-agent systems with disturbances. ISA Trans., vol. 61. – 2016. - pp. 15–28

100. Davoodi M. R., Meskin N., Khorasani K. Simultaneous fault detection and consensus control design for a network of multi-agent systems. Proc. Eur. Control Conf. (ECC). – 2014. - pp. 575–581

101. Krothapalli N. K., Deshmukh A. V. Distributed task allocation in multi-agent systems. Proc. IIE Annu. Conf. – 2002. - p. 1

102. Ding S. X. Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms and Tools. Springer. – 2008. – 501 p.

103. Ризванов Д. А. Алгоритмы управления ресурсами в сложных системах с применением многоагентных технологий. Вестник Уфимского

государственного авиационного технического университета Т. 17 № 5(58). – 2013. – С. 63–69

104. Lin C., Lin Z., Zheng R., Yan G., Mao G. Distributed source localization of multi-agent systems with bearing angle measurements. IEEE Trans. Autom. Control, vol. 61, no. 4. – 2016. - pp. 1105–1110

105. Fallah M. A., Malhame R. P., Martinelli F. Distributed estimation and control for large population stochastic multi-agent systems with coupling in the measurements. Proc. Eur. Control Conf. (ECC). – 2013. - pp. 4353–4358

106. Balaji P., Srinivasan D. An introduction to multi-agent systems. Innovations in Multi-Agent Systems and Applications. Berlin, Germany: Springer. – 2010. - pp. 1–27

107. Horling B., Lesser V. A survey of multi-agent organizational paradigms. Knowl. Eng. Rev., vol. 19, no. 4. – 2004. - pp. 281–316

108. Brooks C. H., Durfee E. H. Congregation formation in multiagent systems. Auto. Agents Multi-Agent Syst., vol. 7, nos. 1–2. – 2003. - pp. 145–170.

109. Chinchor N. MUC-4 Evaluation Metrics. Proc. of the Fourth Message Understanding Conference. – 1992. - pp. 22–29

# ПРИЛОЖЕНИЕ 1. АКТЫ ВНЕДРЕНИЯ

САМАРСКИЙ ПОЛИТЕХ
Опорный университет

**МИНОБРНАУКИ РОССИИ**
**федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Самарский государственный технический университет»**
**(ФГБОУ ВО «СамГТУ»)**

УТВЕРЖДАЮ
Проректор по учебной работе
Самарского государственного
технического университета,
д.п.н., профессор
_____ О.В. Юсупова
«___» _____ 20 ___ года

**АКТ**
внедрения материалов диссертационной работы
Кривошеева Аркадий Владимировича
в учебный процесс

Настоящим актом подтверждается, что результаты диссертационной работы «Математическое и программное обеспечение системы мультиагентного ансамблирования интеллектуальных компонентов распознавания образов» Кривошеева Аркадия Владимировича используются в учебном процесс на кафедре «Вычислительная техника» института автоматики и информационных технологий ФГБОУ ВО «Самарский государственный технический университет» (СамГТУ) в рамках подготовки бакалавров по направлению 09.03.01 Информатика и вычислительная техника, профиль «Цифровые комплексы, системы и сети». Полученные теоретические и экспериментальные данные и методики, разработанные в диссертации, используются при проведении лекционных и лабораторных занятий по дисциплинам:

– Технологии обработки больших данных;

– Методы и средства анализа больших данных;

– Технологии искусственного интеллекта.

Результаты экспериментальных исследований были использованы при подготовке курсовых проектов и выпускных квалификационных работ бакалавров.

И.о. директора института автоматики
и информационных технологий,
д.т.н., доцент _____ К.В. Савельев

И.о. заведующего кафедрой
«Вычислительная техника»,
к.х.н., доцент _____ А.В. Чуваков

open{ code: }
открытый код

Общество с ограниченной ответственностью "Открытый код"
Россия, 443001, г. Самара, ул. Ульяновская, 52/55, этаж 15, ком.14
Тел./факс: (846) 331-11-11, 331-21-01(02,03,04)
E-mail: info@o-code.ru | www.o-code.ru
ОГРН 1036300222100 | ИНН 6313007301 | КПП 631501001

Утверждаю
Управляющий ООО «Открытый код»
_____ О.Л. Сурнин
«28» 02 2022

АКТ
о внедрении результатов научно-исследовательской работы
специалиста Кривошеева Аркадия Владимировича
на тему «Многоагентное решение для интеллектуального выбора решающего агента»

Настоящим актом подтверждаем, что результаты научно-исследовательской работы Кривошеева А.В. на тему «многоагентное решение для интеллектуального выбора решающего агента», а именно технологии распределения задач между агентами в сетевой среде, были использованы при разработке НИР "Библиотеки подпрограмм (SDK) для текстопонимания и текстогенерации на основе машинного обучения для использования в системах управления цифровым контентом нового поколения" по конкурсу РФРИТ в 2021 году.

Директор по управлению проектами - Исполнительный директор, к.т.н. П.В. Ситников

Заместитель руководителя Департамента разработки ПС Д.Н. Крупин

open{code:}
открытый код

Общество с ограниченной ответственностью "Открытый код"
Россия, 443001, г. Самара, ул. Ульяновская, 52/55, этаж 15, ком.14
Тел./факс: (846) 331-11-11, 331-21-01(02,03,04)
E-mail: info@o-code.ru | www.o-code.ru
ОГРН 1036300222100 | ИНН 6313007301 | КПП 631501001

Утверждаю
Управляющий ООО «Открытый код»

_____ О.Л. Сурнин
«28» 02 2022

АКТ
о внедрении результатов научно-исследовательской работы
специалиста Кривошеева Аркадия Владимировича
на тему «Определение девиантного поведения подростков по профилю в социальных сетях»

Настоящим актом подтверждаем, что результаты научно-исследовательской работы Кривошеева А.В. на тему «определение девиантного поведения подростков по профилю в социальных сетях», а именно нейросетевые технологии определения депрессивных и суицидальных наклонностей, были использованы при разработке проекта "Мониторинг депрессивного и суицидального поведения у детей и подростков" используемого в ГКУ СО "Областной центр социальной помощи семье и детям" в 2021 году.

Директор по управлению проектами - Исполнительный директор, к.т.н. П.В. Ситников

Ведущий аналитик И.Н. Дубинина

open{ code }

открытый код

**Общество с ограниченной ответственностью "Открытый код"**

Россия, 443001, г. Самара, ул. Ульяновская, 52/55, этаж 15, ком.14
Тел./факс: (846) 331-11-11, 331-21-01(02,03,04)
E-mail: info@o-code.ru | www.o-code.ru
ОГРН 1036300222100 | ИНН 6313007301 | КПП 631501001

Утверждаю
Управляющий ООО «Открытый код»
_____ О.Л. Сурнин
«17» 06 2024

АКТ

о внедрении результатов диссертационной работы
специалиста Кривошеева Аркадия Владимировича
на тему «Математическое и программное обеспечение системы мультиагентного ансамблирования интеллектуальных компонентов распознавания образов»

Настоящим актом подтверждаем, что результаты диссертационной работы Кривошеева А.В. на тему «Математическое и программное обеспечение системы мультиагентного ансамблирования интеллектуальных компонентов распознавания образов», а именно:

1) Архитектура мультиагентной системы распознавания образов на базе предиктивного оркестратора;
2) Метод мультиагентного ансамблирования интеллектуальных компонентов адаптивной системы распознавания образов
3) Мультиагентный алгоритм распределения задач в адаптивной системе распознавания образов,

были использованы при разработке "Комплексной интеллектуальной системы документооборота и управления цифровым контентом Центральной избирательной комиссии Российской Федерации".

Директор по управлению проектами - Исполнительный директор, д.т.н. П.В. Ситников

Заместитель руководителя Департамента разработки ПС Д.Н. Крупин

# ПРИЛОЖЕНИЕ 2. СВИДЕТЕЛЬСТВА О РЕГИСТРАЦИИ ПРОГРАММ ДЛЯ ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

**RU2022611781**

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

**ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ**

Номер регистрации (свидетельства):
**2022611781**
Дата регистрации: **01.02.2022**
Номер и дата поступления заявки:
**2022611009 28.01.2022**
Дата публикации и номер бюллетеня:
**01.02.2022 Бюл. № 2**
Контактные реквизиты:
**info@o-code.ru, 443001, г. Самара, ул. Ульяновская, 52/55, этаж 15, комната 14**

Автор(ы):
**Земцов Владимир Владимирович (RU),**
**Иващенко Антон Владимирович (RU),**
**Кривошеев Аркадий Владимирович (RU),**
**Крупин Даниил Николаевич (RU),**
**Ситников Павел Владимирович (RU),**
**Столбова Анастасия Александровна (RU),**
**Сурнин Олег Леонидович (RU)**
Правообладатель(и):
**Общество с ограниченной ответственностью "Открытый код" (RU)**

Название программы для ЭВМ:
**Модуль обработки корпоративной документации**

**Реферат**:
Модуль обработки корпоративной документации предназначен для обеспечения возможности структурированного извлечения единиц информации из корпоративной документации на основе онтологии и решает следующие задачи: анализ текста документа; извлечение единиц информации из текста документа в соответствии с заданными правилами; извлечение единиц информации из текста документа с помощью нейросети; прием/передача данных; визуализация результатов обработки корпоративной документации. Тип ЭВМ: IBM PC-совмест. ПК; ОС: Linux.

**Язык программирования**: Python

**Объем программы для ЭВМ**: 2,7 МБ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

**RU2022668401**

# ГОСУДАРСТВЕННАЯ РЕГИСТРАЦИЯ ПРОГРАММЫ ДЛЯ ЭВМ

Номер регистрации (свидетельства):
**2022668401**
Дата регистрации: **06.10.2022**
Номер и дата поступления заявки:
**2022668053 06.10.2022**
Дата публикации и номер бюллетеня:
**06.10.2022 Бюл. № 10**
Контактные реквизиты:
**info@sirius-smr.ru**

Автор(ы):
**Головнин Олег Константинович (RU),**
**Додонова Евгения Александровна (RU),**
**Дубинина Ирина Николаевна (RU),**
**Иващенко Антон Владимирович (RU),**
**Кривошеев Аркадий Владимирович (RU),**
**Крупин Даниил Николаевич (RU),**
**Ситников Павел Владимирович (RU),**
**Сурнин Олег Леонидович (RU)**
Правообладатель(и):
**Общество с ограниченной ответственностью "Сириус-Самара" (RU)**

Название программы для ЭВМ:
**Цифровая платформа интегрального мониторинга**

**Реферат**:
Программа для ЭВМ предназначена для сбора, мониторинга и анализа показателей деятельности организаций, предприятий и учреждений. Программа для ЭВМ включает в себя программные инструменты BI и анализа больших данных, средства поддержки принятия решений на основе искусственного интеллекта. Программа для ЭВМ реализует следующие функции: - Интеграция с внешними информационными системами и базами данных; - Расчет целевых показателей развития; - Прогнозирование динамики изменения значений показателей на основе статистических, исторических и ситуационных данных; - Подключение к открытым источникам данных для поиска сообщений и обращений по заданным словам-маркерам (тегам) с последующей интеллектуальной классификацией; - Реализация информационной поддержки принятия решений. Программа для ЭВМ предназначена для использования в коммерческих предприятиях и в органах исполнительной власти.

**Язык программирования**: JavaScript, Python

**Объем программы для ЭВМ**: 743 Мб

# ПРИЛОЖЕНИЕ 3. ПАТЕНТЫ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 821 835**(13) **C1**

(51) МПК
*G06N 3/02* (2006.01)
*G06F 40/00* (2020.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

## (12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(52) СПК
*G06N 3/02 (2024.01); G06F 40/00 (2024.01)*

(21)(22) Заявка: **2023118951, 18.07.2023**

(24) Дата начала отсчета срока действия патента:
**18.07.2023**

Дата регистрации:
**26.06.2024**

Приоритет(ы):
(22) Дата подачи заявки: **18.07.2023**

(45) Опубликовано: **26.06.2024** Бюл. № 18

Адрес для переписки:
**443099, Самарская обл., г. Самара, ул. Алексея Толстого, 6, оф.6, Филиппенкова Наталья Владимировна**

(72) Автор(ы):
**Александрова Маргарита Владимировна (RU),**
**Головнин Олег Константинович (RU),**
**Головнина Анастасия Александровна (RU),**
**Иващенко Антон Владимирович (RU),**
**Кривошеев Аркадий Владимирович (RU),**
**Крупин Даниил Николаевич (RU),**
**Ситников Павел Владимирович (RU),**
**Сурнин Олег Леонидович (RU)**

(73) Патентообладатель(и):
**ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ "ОТКРЫТЫЙ КОД" (RU)**

(56) Список документов, цитированных в отчете о поиске: **ANTON IVASCHENKO et al., "Hybridization of Intelligent Solutions Architecture for Text Understanding and Text Generation", 02.06.2021, URL: https://www.mdpi.com/2076-3417/11/11/5179. RU 2691214 C1, 11.06.2019. RU 2652461 C1, 26.04.2018. RU 2737720 C1, 02.12.2020. US 11615422 B2, 28.03.2023. JP 2019204415 A, 28.11.2019. JP 2020187729 A,** (см. прод.)

## (54) СПОСОБ ТЕКСТОГЕНЕРАЦИИ НА ОСНОВЕ МАШИННОГО ОБУЧЕНИЯ

(57) Реферат:

Изобретение относится к способу текстогенерации на основе машинного обучения. Технический результат заключается в повышении скорости генерации целевого текстового документа за счет реализации многопоточной среды параллельных вычислений. Способ включает в себя создание экспаундеров, предназначенных для распознавания семантического содержания отдельных фрагментов текста анализируемого документа, входящего в цифровой контент, каждый из которых может инициировать вовлечение других экспаундеров для обеспечения наилучшей точности, использование нескольких отдельных моделей нейронных сетей, каждая из которых обучается на своем индивидуальном наборе данных, для распознавания отдельного текстового элемента, комбинирование необходимого набора нейронных моделей в конвейер, содержащийся в экспаундере, для решения задачи распознавания, формирование экспаундером нескольких возможных вариантов распознанных отдельных текстовых элементов, причем если экспаундер не содержит в себе нейронной сети для распознавания какого-либо текстового элемента, то создает запрос на



RU 2821835 C1

RU 2821835 C1