МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

ЛАГУНОВ НИКИТА АЛЕКСЕЕВИЧ

НЕЙРОСЕТЕВОЕ МОДЕЛИРОВАНИЕ РАСПОЗНАВАНИЯ МНОГОПАРАМЕТРИЧЕСКИХ ОБЪЕКТОВ

Специальность 05.13.18 – Математическое моделирование, численные методы и комплексы программ
Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
кандидат физ.-мат. наук
Мезенцева О.С.

Ставрополь – 2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность диссертационного исследования обусловлена тем, что выделение и распознавание объектов на изображениях – очень важная задача как робототехники и компьютерного зрения, так и прикладных программ. Решение этой задачи может существенно улучшить возможности искусственных систем воспринимать окружающее пространство, разделять его на отдельные логические части и осуществлять интерактивное взаимодействие с объектами окружающего мира.

В настоящее время для решения этой задачи используется огромное количество различных методов: выделение краев, цветовое сегментирование изображения, применение искусственных нейронных сетей, использование особых точек, методы сравнения с эталоном, методы генерации признаков. Но все еще не достигнута достаточная адекватность выделения и распознавания объектов в реальном времени, что не позволяет достичь требуемых показателей в реальных задачах.

Среди всех математических моделей и методов распознавания образов, наиболее перспективными являются нейронные сети. Актуальность имеет задача совершенствования моделей и алгоритмов распознавания. Особо важное значение несут два параметра распознающих систем: скорость работы и качество распознавания. Исходя из этого, вытекает два направления исследований: улучшение, оптимизация и построение новых моделей, а также модификация существующих алгоритмов с помощью применения различных технологий и нестандартных архитектур процессоров с целью повышения скорости их работы.

Отдельно можно выделить задачу формирования методики создания визуальных обучающих выборок для нейронных сетей. На сегодняшний день не существует пошаговой, универсальной методики формирования обучающего множества. Недоисследованным является вопрос влияния различных параметров изображений выборки и предобработки визуальных

данных на качество обучения нейронной сети. Для большинства научных задач используются готовые выборки (NORB, Pascal dataset), в создание которых вложено огромное количество труда. Создание подобной выборки для решения частной задачи – довольно трудоемкий процесс, многие этапы которого являются недостаточно очевидными. Поэтому, важной является разработка простой, понятной, пошаговой методики полуавтоматического создания больших визуальных обучающих выборок, а также исследование влияния различных параметров выборки на качество обучения и распознавания образов нейронной сетью.

Метод распознавания объектов с использованием искусственных нейронных сетей имеет существенные преимущества перед иными методами распознавания. Методы распознавания объектов по форме контуров сегодня почти не используются по двум основным причинам: объект в реальных условиях может не иметь четкого контура, сливаться с окружающим фоном или пересекаться с прочими, нецелевыми объектами, что заметно усложняет не только распознавание объекта, но и точное построение контура его формы. Морфологические преобразования также в большей степени подходят для работы с символами, чем со сложными образами. В решении практических задач распознавания символов, классификаторы на основе нейронных сетей показывают лучшие максимальные показатели: 80.56% против 90.74% [72]. В другом исследовании сверточные нейронные сети показали существенно лучшие показатели в сравнении с машинами опорных векторов и методами, основанными на анализе формы объекта [110].

В направлении нейросетевого распознавания образов существуют определенные противоречия в практике и в науке. Для решения практических задач распознавания необходимо улучшить обобщающую способность СНС. Для того, чтобы практические задачи были решены, требуют доработки методы выделения и распознавания объектов на изображениях, в частности существующие нейросетевые модели.

Современные нейросетевые методы распознавания образов на наборе из нескольких классов физических объектов показывают результаты распознавания в 72% для одного из классов [99] и меньше [91], в то время как необходимая точность для решения практических задач может достигать 100%. Многообразие условий экспериментов не позволяет однозначно сравнивать результаты для различных методов, но очевидным является тот факт, что существующие модели все еще могут быть модифицированы с целью повышения показателей распознавания.

Для оценки качества работы нейронной сети общепринято использовать показатель, называемый обобщающей способностью [57], выражающийся через ошибку обобщения (generalization error). Обобщающую способность можно оценить через отношение числа корректных распознаваний к общему числу примеров валидационного множества. Иными словами, как пишет Хайкин в своей книге «Нейронные сети: полный курс», через «частоту сделанных машиной ошибок при ее тестировании на не встречающихся ранее примерах. При этом предполагается, что тестовые данные принадлежат тому же семейству, что и данные обучения».

Объект исследования – программно-аппаратные комплексы выделения и распознавания объектов на изображениях.

Предмет исследования – методы и алгоритмы сегментирования и распознавания цифровых изображений.

Цель диссертационной работы – повышение обобщающей способности СНС без потерь скорости ее работы.

Научная задача – разработка метода выделения и распознавания объектов, базирующегося на использовании нейронов второго порядка, обеспечивающих повышение обобщающей способности СНС без потерь скорости ее работы.

Реализация поставленной цели может быть разбита на следующие **частные задачи**:

– разработка численного метода отсеивания гипотез по низкочастотной структуре;

– разработка метода выделения объектов на основе модели R-CNN;

– разработка параллельного алгоритма обработки данных в СНС первого и второго порядков, ориентированного на процессоры векторно-матричной архитектуры;

– разработка методики полуавтоматического формирования эффективных визуальных обучающих выборок;

– разработка программного комплекса, реализующего разработанные алгоритмы и позволяющего выделять и распознавать объекты на изображениях, с применением разработанной методики создания обучающих выборок.

Для решения поставленных в работе научных задач использованы методы математического моделирования, искусственных нейронных сетей, выделения объектов, обработки и сегментирования изображений.

Научная новизна диссертационной работы обоснована следующим:

1. Разработан метод выделения и распознавания объектов на основе модели R-CNN, отличающийся от известных использованием СНС высокого порядка и численного метода отсеивания гипотез.

2. Разработан численный метод отсеивания гипотез расположения объекта, отличающийся использованием дополнительного нормализованного градиента для анализа низкочастотной структуры изображения.

3. Разработан параллельный алгоритм обработки данных в СНС второго порядков, ориентированный на процессоры векторно-матричной архитектуры.

4. Разработана методика создания визуальных обучающих выборок, базирующаяся на принципе полуавтоматического создания обучающих примеров.

5. Разработан программный комплекс, реализующий разработанный метод выделения и распознавания объектов.

Практическая значимость работы заключается в том, что:

1. Использование нейронов второго порядка в разработанном методе выделения и распознавания объектов повышает обобщающую способность СНС в среднем на 4%.

2. Использование численного метода отсеивания гипотез позволяет снизить среднее число обрабатываемых нейронной сетью гипотез, а, следовательно, и время, требуемое на их обработку, в среднем на 16%.

3. Применение методики создания визуальных обучающих выборок повышает обобщающую способность СНС в среднем на 6%.

4. Разработанный параллельный алгоритм повышает скорость обработки данных в СНС второго порядка в среднем на 10,5%.

5. Разработанный программный комплекс можно использовать для распознавания объектов с камеры мобильного робота и других искусственных системах компьютерного зрения.

Перечень **положений, выносимых на защиту**, включает:

1. Метод выделения и распознавания объектов на основе модели R-CNN, отличающийся от известных использованием СНС высокого порядка и модификацией этапа генерации гипотез.

2. Численный метод отсеивания гипотез расположения объекта, отличающийся использованием дополнительного нормализованного градиента для анализа низкочастотной структуры изображения.

3. Параллельный алгоритм обработки данных в СНС второго порядка, ориентированный на процессоры векторно-матричной архитектуры.

4. Методика создания визуальных обучающих выборок, базирующаяся на принципе полуавтоматического создания обучающих примеров.

5. Программный комплекс выделения и распознавания объектов, использующий разработанный метод выделения и распознавания объектов, параллельный алгоритм и методику создания обучающих выборок.

Достоверность и обоснованность полученных результатов обеспечивается применением современной технологии математического моделирования, корректностью математических постановок задач, и результатами натурных экспериментов.

Авторский вклад в разработку. Основные результаты и выводы диссертационной работы получены лично автором. Авторским вкладом является разработка модели выделения и распознавания объектов на основе модели R-CNN, проведение экспериментов по созданию и тестированию обучающих выборок, разработка метода выделения объектов на основе селективного поиска с уменьшением количества генерируемых гипотез, разработка параллельного алгоритма обработки сигнала в сверточных нейронных сетях первого и второго порядков.

В коллективных работах автора экспериментально исследованы влияние освещенности на качество распознавания объектов нейронной сетью и использование моделей сверточных нейронных сетей высоких порядков, разработан программный комплекс для выделения и распознавания объектов.

Апробация работы. Основные результаты диссертационной работы докладывались и обсуждались на следующих научных конференциях и семинарах: Всероссийская научная конференция «Современные проблемы математического моделирования, супервычислений и информационных технологий» (г. Таганрог, 2012), II международная научно-практическая конференция «Актуальные проблемы современной науки» (г. Ставрополь, 2013), «Academic science – problems and achievements IV» (North Charleston, 2014), I Всероссийская научно-техническая конференция «Вычислительные и информационные технологии в науке, технике и образовании» (г. Ставрополь, 2014), I Всероссийская научно-техническая конференция «Фундаментальные и прикладные аспекты компьютерных технологий и информационной безопасности» (г. Таганрог, 2015).

Внедрение. В диссертационной работе изложены результаты исследований, выполненных в 2013 – 2015 годах. Работа выполнялась в

соответствии с планами НИР СКФУ. Основные результаты исследований были внедрены в ГК «Стилсофт» в ходе выполнения работ по системе биометрической идентификации личности по изображению лица человека АИС «Синергет Розыск» (акт о внедрении от 3 сентября 2015г.), в ПАО НПО «Андроидная техника» в ходе выполнения работ по составной части НИР «Разработка программного комплекса распознавания изображений с камеры мобильного робота» (акт о внедрении от 25 августа 2015г.), и в учебный процесс СКФУ (акт о внедрении от 25 июня 2015г.).

Публикации. По теме диссертации опубликовано всего 18 работ в журналах и трудах конференций, из них 5 работ – в изданиях, рекомендованных ВАК РФ для опубликования научных положений диссертационных работ, получено 6 свидетельств о государственной регистрации программ для ЭВМ.

Структура и объем диссертации. Диссертация состоит из введения, пяти глав, заключения, списка литературы из 153 наименования и 3 приложений. Общий объем 179 страниц.

Во введении обоснована актуальность темы диссертации, определена научная новизна и практическая значимость работы, сформулированы цель и задачи исследования, приведена краткая характеристика полученных результатов и представлены положения, выносимые на защиту.

В первой главе представлен анализ научно-технической литературы, посвященной существующим математическим моделям, методам и алгоритмам классификации образов и выделения объектов на изображениях. Сформулирована научная задача диссертационной работы и проведена ее декомпозиция на частные подзадачи.

Во второй главе представлен метод выделения и распознавания объектов на изображениях, описаны численный метод отсеивания гипотез по низкочастотной структуре и модель сверточной нейронной сети второго порядка.

В третьей главе представлено решение второй частной задачи – разработка параллельного алгоритма обработки данных в сверточных нейронных сетях второго порядка, ориентированного на процессоры с векторно-матричной архитектурой.

В четвертой главе представлено решение третьей частной задачи. Описана методика создания эффективных визуальных выборок для нейронных сетей. Рассматриваются несколько аспектов создания обучающей выборки: различные параметры изображений, тип фона, способ создания выборки, применение фильтрации и преобразование обучающих примеров.

В пятой главе разработан программный комплекс выделения и распознавания объектов на изображениях, с использованием разработанных метода, параллельного алгоритма и методики. Программный комплекс состоит из 3 модулей: модуль подготовки материала и генерации обучающей выборки, модуль выделения объектов, модуль распознавания образов с помощью сверточной нейронной сети второго порядка. Проведена проверка адекватности разработанной модели выделения и распознавания объектов на основе данных натурного эксперимента.

ГЛАВА 1 АНАЛИЗ НАУЧНО-ТЕХНИЧЕСКОЙ ЛИТЕРАТУРЫ ПО ТЕМЕ ДИССЕРТАЦИОННОГО ИССЛЕДОВАНИЯ

Данная глава содержит результаты анализа основных методов распознавания образов, методов выделения объектов на изображении, выбор конкретной модели выделения и распознавания объектов на изображении, а также формулирование задачи и частных подзадач диссертационного исследования.

1.1 Анализ алгоритмов и методов распознавания образов

1.1.1 Анализ программно-аппаратных комплексов распознавания объектов

Программно-аппаратный комплекс – это совокупность программных и технических средств, работающих совместно для выполнения одной или нескольких схожих задач [24]. Соответственно, любой программно-аппаратный комплекс состоит из двух основных частей: программной части и аппаратной части. Аппаратная часть представляет из себя устройства сбора и/или обработки информации, может включать, например, процессоры, биометрические детекторы, платы видеозахвата и т. д. Программная часть – специализированный набор команд для обработки и интерпретации данных, собранных аппаратной частью.

Для программно-аппаратных комплексов распознавания визуальных объектов в качестве входных данных могут быть использованы видео, набор изображений или статичное изображение, следовательно для сбора входных данных чаще всего используется камера и компьютер, к которому она подключена. Аппаратная часть влияет на скорость обработки данных, качество обрабатываемых изображений, разрешение и т.д.

Программно-аппаратные комплексы для распознавания могут быть разделены на несколько типов, в зависимости от конкретной задачи, которую они решают: распознавание автомобильных номеров, лиц, конкретного набора объектов и т.д. Ядром является программа, в основе алгоритма действия которой лежит один или несколько методов распознавания образов.

Программную часть определяют: используемый тип алгоритма распознавания (нейронные сети, признаки Хаара, машина опорных векторов и т.д.), конкретная модель, реализующая алгоритм, конкретная реализация модели на одном из языков программирования, а также общие принципы,

лежащие в основе работы системы (использование скользящего окно, сегментирование, сжатие изображений и т.д.).

Программная часть может базироваться на применении одного или нескольких методов распознавания: контурный анализ, морфологические преобразования, метод главных компонент, линейный дискриминантный анализ, скрытые маковские модели, метод опорных векторов, искусственные нейронные сети.

Далее будет приведен анализ основных существующих методов распознавания объектов на изображениях.

При использовании многих методов распознавания, необходим этап предварительного выделения объекта на сложном изображении, без которого система не сможет осуществить корректный анализ входных данных. В качестве основных методов для выделения объектов на изображении могут быть использованы: выделение по цвету, выделение границ объектов, поиск объекта по шаблону, использование признаков Хаара, комплексные модели, алгоритм которых подразумевает как выделение, так и распознавание объектов. Анализ каждого из этих методов также будет описан в данной главе.

Существует Image recognition API от iTraff Technology – технология, позволяющая создавать простые приложения для распознавания объектов без глубокого знания современных интеллектуальных систем и сложного программирования. Технология направлена в первую очередь на создание мобильных приложений для магазинов и производств. К достоинствам системы можно отнести простоту работы и легкость создания небольших приложений. Однако проект имеет очевидную коммерческую направленность, и не предоставляет пользователю гибкость в настройке распознающей системы, может быть применен только для создания простейших коммерческих приложений.

Программный комплекс для идентификации объектов, разработанный Mallenom Systems – система машинного зрения, позволяющая

идентифицировать любые объекты в процессе производства на производственной линии конвеера по различным признакам, или проверять соответствие объекта различным параметрам. В качестве признаков объекта могут выступать: размеры, форма, цвет, наличие отверстий, маркировка, штрих-коды, этикетка и т.д. Данный программный комплекс имеет узкую направленность на использование в производстве, настраивается специалистами фирмы и не может быть использован для решения более общих задач.

Программная среда BrainMaker, разработанная компанией California Scientific Software, позволяет создавать приложения на основе различных архитектур нейронных сетей и применять их для прогнозирования рядов и распознавания образов. Существенные минусы BrainMaker: недостаточная гибкость настроек под конкретную задачу и отсутствие поддержки СНС.

Российская разработка Deductor – платформа, реализующая множество алгоритмов из различных направлений интеллектуальных систем: нейронные сети, генетические алгоритмы, деревья решений, методы математической статистики, методы работы с базами данных и др. У системы есть существенный недостаток: реализованные в рамках платформы нейронные сети предназначены в первую очередь для анализа данных и прогнозирования и плохо приспособлены для распознавания объектов на изображениях.

Библиотека компьютерного зрения с открытым исходным кодом OpenCV, получившая широкое распространение для решения научных и прикладных задач. Библиотека написана на C++ и содержит алгоритмы сегментирования изображений, трекинга объектов в реальном времени и модули машинного обучения, в том числе реализации нейронных сетей. OpenCV содержит самые базовые архитектуры и возможности, что затрудняет использование этой библиотеки в качестве самодостаточного решения для задачи распознавания нескольких классов объектов на изображении. Однако, это хороший инструмент для базовой обработки

визуальных данных и создания интерфейсов между отдельными модулями распознающей системы.

Среди широкого разнообразия нейроэмуляторов и программных реализаций нейронных сетей можно выделить несколько наиболее перспективных вариантов для анализа.

Fann – одна из первых нейросетевых библиотек с исходным кодом, позволяет создавать типовые многослойные нейронные сети прямого распространения через использование нескольких команд, производить обучение и тестирование созданных архитектур. Минусами библиотеки являются полное отсутствие поддержки СНС, не самый удобный формат обучающих множеств, неоптимальное использование дискового пространства для хранения выборок, а также низкое быстродействие разрабатываемых программ.

Программное нейросетевое решение ***Veles*** [142], разработанное компанией Самсунг под лозунгом «Вы изменяете параметры, Велес позаботится об остальном». К плюсам библиотеки можно отнести возможность запускать на ноутбуке или производительном кластере, простоту запуска в несколько команд, возможность мониторинга процесса обучения и использования облачных ресурсов, наличие большого числа различных архитектур нейронных сетей, в том числе полносвязных, сверточных, рекуррентных модели, а также возможность быстро сконвертировать созданную модель в отдельное приложение.

Наряду с плюсами, Veles имеет ряд существенных недостатков: несмотря на наличие встроенных средств для создания обучающих множеств, создание выборки требует много сил и времени, разработка имеет недостаточную гибкость настроек под конкретные задачи распознавания, отсутствует поддержка нейронных сетей высоких порядков.

TensorFlow – библиотека машинного обучения с открытым кодом от Google [141]. Плюсы: визуализация архитектуры нейронной сети и процесса обучения, высокая скорость работы. Минусы: работа над библиотекой еще не

закончена, она постоянно расширяется и дописывается, доработка библиотеки и устранение ошибок во многом ложится на пользователей, отсутствует поддержка нейронов высоких порядков.

Neon – нейросетевой фреймворк с поддержкой СНС и глубокого обучения как на CPU, так и с поддержкой GPU [122]. Пользовательская часть написана на python в то время, как ядро системы использует собственную библиотеку разработчиков на языке ассемблера. Neon поддерживает несколько моделей обучения и является быстрейшей реализацией СНС на сегодняшний день, но крайне не гибок в плане подготовки обучающего материала и настройки обучающих параметров. Также как и большинство нейроэмуляторов, не предполагает использование нестандартных архитектур нейронных сетей.

Torch – библиотека и скриптовый язык с открытым исходным кодом, осованный на языке программирования Lua [143]. Предоставляет широкий набор алгоритмов грубокого машинного обучения на базе LuaJIT и C. В то же время, не является строго специализированным решением для эмуляции нейронных сетей, не содержит модули для реализации нейронов высоких порядков и не поддерживает платформу Windows.

Caffe – популярная библиотека СНС, совместимая с C++, Python, Matlab [79]. Позволяет создавать нейронные сети послойно, гибко изменяя архитектуру через добавление и удаление отдельных слоев. Является одним из лучших решений в области глубокого обучения и эмуляции нейронных сетей на сегодня, но, также как и остальные библиотеки, обходит стороной вопрос реализации нейронов с нестандартными сумматорами.

Neural Designer – эмулятор нейронных сетей, в основе которого лежит библиотека с открытым кодом OpenNN [123]. Является программным средством для анализа данных, в первую очередь распознавания образов. Несмотря на широкую известность, не содержит реализации наиболее распространенных современных нейросетевых моделей, в том числе СНС и НС с обратными связями.

В таблице 1.1 приведены сравнительные характеристики рассмотренных эмуляторов и библиотек нейронных сетей.

Таблица 1.1 – Сравнение программных реализаций нейронных сетей

Продукт	Открытый исходный код	Платформа	Язык	Поддержка CUDA	Рекуррентные сети	СНС	НС высоких порядков
FANN	Да	Win	C++	Нет	Нет	Нет	Нет
Veles	Да	Win, Linux	C++, Java, Python	Да	Нет	Да	Нет
Tensor Flow	Да	Linux, Mac	Python, C/C++	Да	Да	Да	Нет
Neon	Да	Mac	Python	Да	Нет	Да	Нет
Torch	Да	Linux, Android, iOS, Mac	C, Lua	Решение сторонних разработчиков	Да	Да	Нет
Caffe	Да	Ubuntu, Win, OS X	C++, Python	Да	Да	Да	Нет
Neural Designer	Нет	Win, OS X, Linux	C++	Нет	Нет	Нет	Нет

Как видно, большинство современных библиотек и нейроэмуляторов имеют схожие недостатки, затрудняющие их использование в качестве основы для программно-аппаратного комплекса распознавания многопараметрических объектов.

1.1.2 Анализ метода распознавания, основанного на работе с контурами объектов

Контурный анализ – это метод распознавания объектов, основанный на работе с границами, называемыми также силуэтными линиями объекта, которые разделяют разные области, имеющие равномерную яркость [56]. Также контур на изображении может быть порожден разрывами в отражательных свойствах поверхности, равно как и отсутствием непрерывности в ее ориентации (рис. 1.1).

Данный подход предполагает, что контур объекта содержит в себе всю необходимую для распознавания информацию, при этом внутренние точки объекта не принимаются во внимание. С одной стороны, область применимости контурного анализа имеет сильные ограничения, но в то же время такой подход позволяет перейти к работе непосредственно с пространством контуров от двумерного пространства исходного изображения, за счет чего существенно снизить алгоритмическую и вычислительную сложность задачи. Как бы то ни было, во многих случаях, методы контурного анализа успешно справляются с решением задачи распознавания, обеспечивая инвариантность к таким видам преобразования, как поворот, перенос и изменение масштаба изображения.

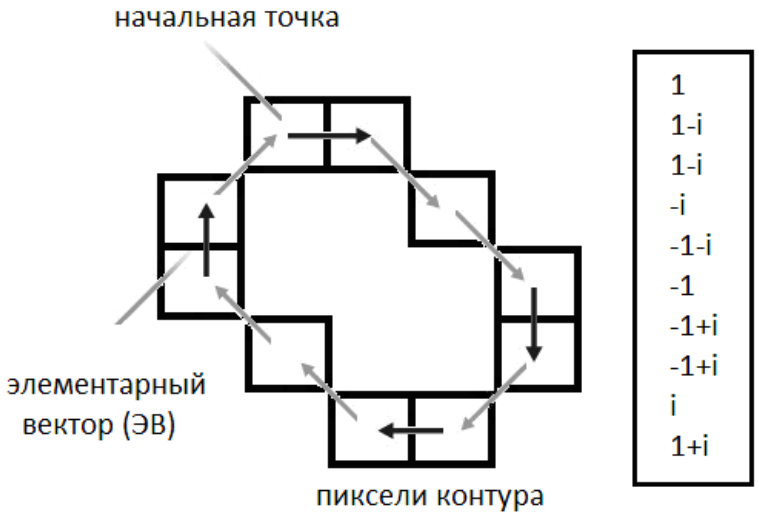


Рисунок 1.1 – Пример примитивного контура

Под контуром понимается некоторая совокупность точек на изображении, которая отделяет важную графическую информацию, относящуюся к объекту, от всего остального массива данных, принимаемого за фон [8]. Для того, чтобы определить эту совокупность пикселей, сначала на изображении выбирается одна начальная точка, затем каждая последующая точка описывается комплексным числом, характеризующим ее смещение относительно предыдущей. При этом действительная часть задает размер смещения по оси X, а мнимая – по оси Y. Такой подход позволяет

однозначно определить обход контура в заданном направлении, т.к. контур любого трехмерного объекта, по своей физической природе, всегда является замкнутым, и обход контура всегда приводит к начальной точке. Использование комплексных чисел вместо векторов, как в двумерном кодировании, имеет ряд преимуществ для задачи распознавания, т.к. любой контур, описанный данным способом, имеет следующие свойства:

1) Сумма элементарных векторов замкнутого контура равна нуль-вектору.

2) Независимость от параллельного переноса исходного объекта, инвариантность при сдвиге вектора, т.к. это приведет только к изменению начальной точки.

3) Поворот объекта на любой угол может быть описан, как поворот каждого элементарного вектора, входящего в контур, на тот же угол.

4) Любое изменение масштаба изображения равносильно умножению элементарных векторов на определенный коэффициент.

Общая схема контурного анализа изображения предполагает предварительное преобразование данных в формат бинарного изображения, а также удаление шумов и помех, повышение контрастности, небольшое сглаживание, затем выделение и фильтрация по площади, периметру, форме контуров для работы, преобразование их в вид, обеспечивающий единообразие по длине и сглаживанию, и, наконец, поиск максимально близкого к данному контуру шаблона путем перебора всей коллекции найденных контуров.

Если предположить, что изображение для анализа имеет размер n*n пикселей и покрыто сеткой с равномерным шагом S, то суммарная длина всех линий, входящих в сетку, может быть вычислена по формуле:

$$L = 2 \cdot n^{2/s}. \qquad (1.1)$$

Исходя из этого, можно сделать вывод, что, хотя переход из двумерного пространства изображения к работе с контурами не уменьшает размерность задачи, но существует ряд условий, благодаря которым

контурный анализ на современных вычислительных системах может применяться как средство распознавания реального времени [19]. Во-первых, если на снимке присутствует только один объект, системе необходимо будет работать только с одним контуром, а значит общее количество пикселей на изображении даст линейную зависимость от n. Во-вторых, методы контурного анализа инвариантны к изменению масштаба и поворотам, поэтому имеют очевидное преимущество перед базовыми алгоритмами двумерных методов, с необходимостью перебора углов поворота и масштаба (SURF, SIFT, Хаар). В-третьих, алгоритмы контурного анализа легко распараллеливаются за счет независимости отдельных контуров друг от друга. Кроме того, все контуры, как правило, отсортировываются и обрабатываются последовательно, причем результат вычислений первого контура, в большинстве случаев достаточный для принятия решения, может быть предоставлен в первую очередь, без ожидания завершения обработки остальных.

Как и любой подход, контурный анализ имеет свои *недостатки*, среди которых в первую очередь стоит отметить два фактора [55]. Во-первых, целевой объект может не иметь четкого контура и быть слабо выраженным на окружающем его фоне. Зашумление помехами, одинаковые яркость и цвет у объекта и фона, отсутствие четкой границы, размытость – все это может привести к тому, что контур либо вообще будет невозможно выделить, либо он выделится неправильно и не будет иметь сходства с реальной границей объекта. Второй фактор, сильно усложняющий практическое применение контурного анализа – это тот факт, что объект может иметь пересечения с другими объектами, а также находиться в неполной видимости относительно наблюдателя, в то время как все алгоритмы распознавания контура построены на том, что контур содержит всю необходимую информацию об объекте, а не только о некоторых его частях. Также, к основным недостаткам контурного анализа можно отнести тот факт, что при изменении угла обзора и других параметров относительно наблюдателя, неизбежно изменятся и

наблюдаемые границы объекта, поэтому необходимым является поиск метода выделения краев для каждого случая, либо специальная предобработка изображения. Также положительные моменты, касающиеся сложности задачи, описанные ранее, относятся только к обработке контуров, которая, однако, требует предварительного их выделения.

1.1.3 Анализ метода распознавания, основанного на морфологических преобразованиях

Изменения оптических характеристик объекта, условий освещения, угла обзора приводят к неизбежному, порой радикальному, различию между наблюдаемыми сценами. Однако одним из основных требований любой высокоуровневой интеллектуальной системы является независимость качества процесса распознавания объекта от условий его визуальной регистрации. Одним из подходов для решения этой проблемы стал морфологический анализ, который в большинстве случаев применяется к монохромным бинарным изображениям [43].

В основе данного метода лежат принципы нелинейной суперпозиции сигналов, теория множеств, а также классы нелинейных или морфологических систем. Теория основана на обработке и анализе различных геометрических структур, содержащихся в бинарном изображении, но также может быть применена в теории графов и других областях. Главной целью морфологического подхода является нахождение максимального инварианта, то есть некоторого класса, который определял бы те свойства, параметры, характеристики, которые относятся к исследуемому объекту, в то же время, не зависящие от условий, в которых формируется изображение. При этом, для моделирования изменения условий или средств регистрации изображения, используются математические методы, например, преобразование яркости.

Любая операция в бинарной морфологии является преобразованием определенного вида некоторого подмножества точек изображения, называемых область. При этом результатом операции является новое бинарное изображение, а для его получения используются исходное изображение и структурный элемент, отражающий некоторую геометрическую форму произвольной структуры и размера. В качестве

структурного элемента могут выступать прямоугольник или круг фиксированного размера, в которых определена начальная точка, обычно соответствующая центральному пикселю двоичного изображения.

Обобщенный алгоритм любой морфологической операции выглядит следующим образом [54]. Сначала строится результирующая поверхность заданного размера и заполняется нулями. Затем исходное изображение подвергается попиксельному сканированию, при этом для каждой точки рассчитывается наложение структурного элемента на соответствующую область изображения, и последующая проверка того, насколько лежащие под структурным элементом точки соответствуют его структуре. Если в результате расчетов фиксируется необходимая степень сходства, то в соответствующее место результирующей поверхности записывается единица.

Морфологический оператор – это фильтр определенного вида, который применяется для морфологического анализа бинарного изображения. Базовыми морфологическими операторами можно считать фильтры максимум и минимум, которые приводят к расширению, либо сужению исходного изображения. Если принять w за малый структурный двоичный элемент, алгебраическая разность, так называемый градиент эрозии, может улучшить видимость контуров бинарного изображения.

$$EG(f) = f - (foW) \tag{1.2}$$

Формула градиента наращения, аналогичного, но противоположного по смыслу градиенту эрозии, выглядит следующим образом:

$$DG(f) = (f \oplus W) - f\,. \tag{1.3}$$

С помощью комбинирования вышеописанных формул, возможно создание большого числа различных контурных операторов, например:

1) Морфологические операторы минимума и максимума, усиливающие контуры объекта $\max[EG(f), DG(f)]$ или $\min[EG(f), DG(f)]$.

2) Оператор Лапласа $DG(f) - EG(f)$.

3) Оператор Бойхера $EG(f) + DG(f)$.

Среди основных операций математической морфологии можно выделить операции переноса, наращивания, эрозии, замыкания и размыкания бинарного изображения.

Операция переноса происходит в том случае, когда некое множество точек изображения X сдвигается на вектор v, за счет чего каждый пиксель множества меняет свое месторасположения на заданное расстояние. Математически перенос записывается в виде $X_v = \{x + v \mid x \in X\}$, сам же вектор переноса, принимая тот факт, что рассматривается двумерное изображение, может быть задан в виде пары чисел $(\Delta a, \Delta b)$, соответствующих компонентам вектора в направлении строк и столбцов изображения соответственно.

К базовым операциям математической морфологии относят также операции наращивания и эрозии [18], которые по своему смыслу и видимому воздействию на изображение являются противоположными, хотя и имеют весьма схожий алгоритм действия. Наращивание изображения I некоторым структурным элементом E обозначается как $I \oplus E$, а эрозия – $I \Theta E$. Структурный элемент при выполнении базовой операции применяется ко всем пикселям изображения, причем каждый раз происходит логическое сложение структурного элемента с точками изображения в окрестностях данного пикселя и заполнение суммирующего бинарного изображения, изначально инициализированного нулевыми значениями. Действие оператора наращивания проявляется в утолщении отдельных линий, при котором контуры объектов становятся более явными. Эрозия, напротив, носит деструктивный характер, все части, меньшие структурного элемента, удаляются из изображения, тонкие линии стираются, объекты, соединенные этими линиями, становятся разъединенными и уменьшаются.

$$I \oplus E = \bigcup_{e \in E} I_e, \tag{1.4}$$

$$I \Theta E = \{z \in I \mid E \subseteq I\}. \tag{1.5}$$

Базовые операции можно комбинировать, получая на выходе синергический эффект и свойства, которых нельзя добиться использованием

операций наращивания и эрозии по отдельности. Если первым в последовательности операций стоит наращивание, а за ним – эрозия, то такая комбинация называется замыканием. При наращивании происходит увеличение размеров контура и удаление небольших промежуточных участков, а операция эрозии смягчает это действие и позволяет добиться тех же результатов, но без увеличения контуров. Если же сначала обработать бинарное изображение с помощью оператора эрозии, а затем применить наращивание, то можно получить эффект удаления небольших объектов и шума без уменьшения размеров целевых объектов. Таким образом, с помощью комбинирования базовых операций, можно управлять результатом обработки, оказывая влияние на те или иные свойства изображения.

$$I \bullet E = (I \oplus E)\Theta E, \tag{1.6}$$

$$I \circ E = (I\Theta E) \oplus E. \tag{1.7}$$

Применительно к полутоновым изображениям алгоритм морфологических преобразований для распознавания образов состоит в следующем: Инициализируется массив с размерами исходного изображения, окно структурирующего элемента проходит по изображению, и для каждого состояния выбирается максимальное или минимальное значение интенсивности пикселя.

Такой подход имеет существенные *недостатки* [49]:

1) проблема выбора подходящего порога бинаризации;

2) неопределенность выбора размеров структурирующего элемента.

3) плохо подходит для работы со сложными образами, поэтому применяется преимущественно для распознавания символов, автомобильных знаков и других объектов, имеющих довольно простую структуру.

1.1.4 Анализ методов, основанных на построении модели объекта

В отдельную группу можно отнести методы, направленные на выявление параметров и закономерностей изображения, с использованием методов машинного обучения и математической статистики [28]. Для вычисления и формирования вектора признаков, используемого для процесса распознавания и отнесения конкретного объекта к определенному классу, может использоваться само исходное изображение, что повышает объективность и количество полезной информации, потенциально содержащейся в векторе. С другой стороны, такой подход является крайней ресурсозатратным за счет высокой размерности формируемого пространства признаков.

К описанной группе методов распознавания можно отнести метод главных компонент, линейный дискриминантный анализ, скрытые Марковские модели и метод опорных векторов.

Анализ метода главных компонент

В общем случае перевод черно-белого изображения размером m x n пикселей в вектор приводит к появлению структуры с размерностью m x n. При начальных размерах изображения 96x96 точек, получим вектор, включающий 9216 чисел. На практике работа с пространством большой размерности может быть весьма затруднительной. Для упрощения задачи путем снижения размерности пространства признаков без потерь информативности исходного образа может быть использован метод главных компонент. Вектор X преобразуется в вектор Y с меньшей размерностью с сохранением общей дисперсии.

Обычно вычисляются расстояние от тестового вектора до его проекции и расстояние от этой проекции до усредненного вектора тренировочного

набора. Знание этих двух величин позволяет решить относится ли рассматриваемый вектор к определенному классу. Векторное пространство разбивается на собственное пространство, в которое входят главные компоненты, и ортогональное дополнение этого пространства. Основная дисперсия набора данных в этом случае направлена вдоль нескольких осей, называемых главными, и информация, характеризующая большую часть изменчивости изображения, сосредоточена в области нескольких главных компонент. Оставшиеся компоненты можно отбросить, перейдя в пространство существенно меньшей размерности [25].

Недостатки метода:

1) Метод главных компонент очень требователен к качеству условий съемки объектов. Все изображения выборки должны иметь качественную предварительную обработку и схожие значения таких параметров, как поворот, масштаб, положение относительно центра, яркость и удаление фона.

2) В ситуации, когда элементы входного множества расположены на поверхности гиперсферы, невозможно понизить размерность с помощью какого то линейного преобразования.

3) При движении в направлении максимизации дисперсии, не всегда можно достигнуть максимально возможной информативности.

Анализ метода распознавания, основанного на линейном дискриминантном анализе

Основной идеей этого метода является выбор такой проекции пространства объектов на пространство признаков, при которой внутриклассовое различие входящих в выборку объектов будет минимальным, а межклассовое – максимизировано, чтобы наибольшим образом упростить задачу классификации [6]. Тесно связан с линейным дискриминантным анализом линейный дискриминант Фишера.

Цель подхода заключается в том, что проекции векторов признаков, принадлежащих разным классам, должны быть максимально удалены друг от друга при переходе от пространства R^n в подпространство R^m, где m существенно меньше n.

В общем случае для каждого объекта a имеется некоторый набор наблюдений x, называемых признаками, формирующий обучающую выборку. Задача заключается в том, чтобы построить модель, способную прогнозировать a на основе наблюдений независимо от того, входит объект в обучающую выборку или нет.

Недостатки подхода:

1) Способен давать адекватные результаты только при нормальном распределении исходных данных.

2) В случае, если ковариационные матрицы классов являются вырождены, линейный дискриминантный анализ становится непригодным для использования.

3) Аналогичный результат получается, если ковариационные матрицы разных классов имеют существенные различия между собой.

Анализ метода распознавания, основанного на Скрытых Марковских Моделях

Скрытые Марковские Модели (СММ) дают возможность учитывать при анализе сигнала его пространственно-временные параметры, получить его математическую модель [97]. Данный метод широко используется для распознавания речи и визуальной графической информации.

Скрытые Марковские Модели относятся к классу стохастических моделей и делают упор на статистические данные о процессе или объекте. Перед использованием, их необходимо настроить, максимизировав вероятность генерации сигналов-векторов из обучающего множества. Отнесение предъявленного системе объекта к одному из классов также

происходит путем вычисления вероятности генерации вектора признаков, соответствующего данному классу.

Недостатком такого подхода является то, что Скрытые Марковские Модели стремятся лишь к максимально возможному отклику отдельной модели на конкретный класс и не учитывают различий между классами, а также вероятность принадлежности объекта к неизвестному модели классу.

Анализ метода опорных векторов

Используя данный вид классификации с обучением, исследователь стремится найти такую функцию f(x) для пространства векторов R^n, которая принимала бы отрицательные значения, если вектор принадлежит одному классу, и положительные, в случае принадлежности входных данных к другому классу [74].

Для построения разделяющей функции используется набор тренировочных данных $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$. Задача заключается в нахождении функции F $X \to Y$. Для случая двух классов, задача сводится к поиску разделяющей прямой. Если же классификация осуществляется по нескольким классам, речь следует вести о разделяющей гиперплоскости. Оптимальной разделяющей прямой называется та, которая наилучшим образом разделяет два класса, т.е. прямая, расстояние от которой до каждого класса максимально.

Линейный классификатор имеет крайне ограниченные возможности, существует большое количество классов, линейно неразделимых в пространстве R^n. Из этой ситуации можно выйти путем перехода в пространство более высокой размерности, но вместе с повышением размерности пространства увеличивается также сложность обработки данных.

Недостатки метода опорных векторов: для классификации объектов используется не все множество обучающих данных, а лишь те из примеров,

которые лежат на границах, сложность в интерпретации параметров построенной модели и невозможность точно предсказать вероятность попадания примера в один из классов.

1.1.5 Анализ метода распознавания, основанного на применении искусственных нейронных сетей

Большим преимуществом искусственных нейронных сетей является то, что эта модель создана на основе современных представлений о способе обработки информации человеческим мозгом, который обладает способностью изменять организацию своих структурных компонентов – нейронов – таким образом, чтобы решать конкретные, сложные, плохо формализованные задачи, к числу которых относится также и распознавание образов. Причем на распознавание сложного объекта в комплексных изменяющихся условиях, мозг тратит около 100-200 миллисекунд.

С. Хайкин в своей книге «Нейронные сети. Полный курс» дает такое определение данного понятия: *«Нейронная сеть – это громадный распределенный параллельный процессор, состоящий из элементарных единиц обработки информации, накапливающих экспериментальные знания и предоставляющих их для последующей обработки*» [57].

В первую очередь нейронные сети характеризуют такие их свойства как массивное распараллеливание процесса обработки информации и способность к обобщению. Несмотря на это, на сегодняшний день, нейронные сети еще не способны давать сложные готовые решения, из-за чего их приходится интегрировать в многосоставные системы для решения конкретных, пусть и не тривиальных задач.

Искусственные нейронные сети имеют ряд полезных свойств:

1. *Нелинейность* всей сети и отдельных нейронов, из которых она состоит.

2. *Адаптивность*. Нейронные сети могут изменять свою структуру, приспосабливаясь к изменениям в окружающей среде или предметной области.

3. *Устойчивость к отказам*. При повреждении отдельных нейронов или связей в аппаратной реализации нейронной сети, работоспособность всей системы сохраняется.

4. *Единый подход к анализу и проектированию*, универсальность в решении задач, относящихся к различным предметным областям.

Нейронная сеть состоит из большого числа элементарных процессоров – нейронов, выполняющих простые вычисления, но связанных друг с другом и, таким образом, в совокупности способных решать сложные, плохо формализованные задачи. Нейроны группируются в слои таким образом, что на вход каждого нейрона в следующем слое поступают выходные значения всех нейронов предыдущего слоя (такие сети называют полносвязными). При этом каждой связи, соединяющей два нейрона, соответствует определенный вес, определяющий силу взаимодействия между ними. Связи между нейронами могут быть возбуждающими или тормозящими.

При всех своих объективных достоинствах нейронные сети остаются чрезвычайно сложным механизмом обработки информации. Очень важно заранее серьезно подойти к выбору модели сети, чтобы хотя бы примерно определить, какие результаты мы можем ожидать от системы при решении конкретной поставленной задачи.

«Стандартные» сети прямого распространения способны выполнять очень сложные задачи по распознаванию объектов. Однако этот вид архитектуры нейронных сетей имеет некоторые ограничения. В ситуациях, когда исследуемый объект появляется вместе с другими объектами, сети прямого распространения могут быть перегружены и не способны корректно осуществить распознавание [152].

Сверточные нейронные сети обрабатывают исходное изображение не полностью, а отдельными «порциями», последовательно уменьшая его

размер или выделяя характерные наиболее важные признаки, уходя на новый уровень абстракции. В этих сетях формируются так называемые карты признаков, которые стороннему наблюдателю кажутся размытыми, искаженными копиями исходного изображения, но для нейронной сети имеют совершенно иной смысл, содержат инварианты и характерные признаки. Основная идея сверточной нейронной сети заключается в чередовании субдискретизирующих (S-layers) слоев, сверточных (C-layers) слоев, а также полносвязных (F-layers) выходных слоев [73]. Таким образом, объединяются воедино три архитектурные идеи, которые помогают достичь инвариантности к искажениям и сдвигам исходного изображения: идеи о локальных полях восприятия, о разделяемых весах и о пространственной субдискретизации (рисунок 1.2) [34].

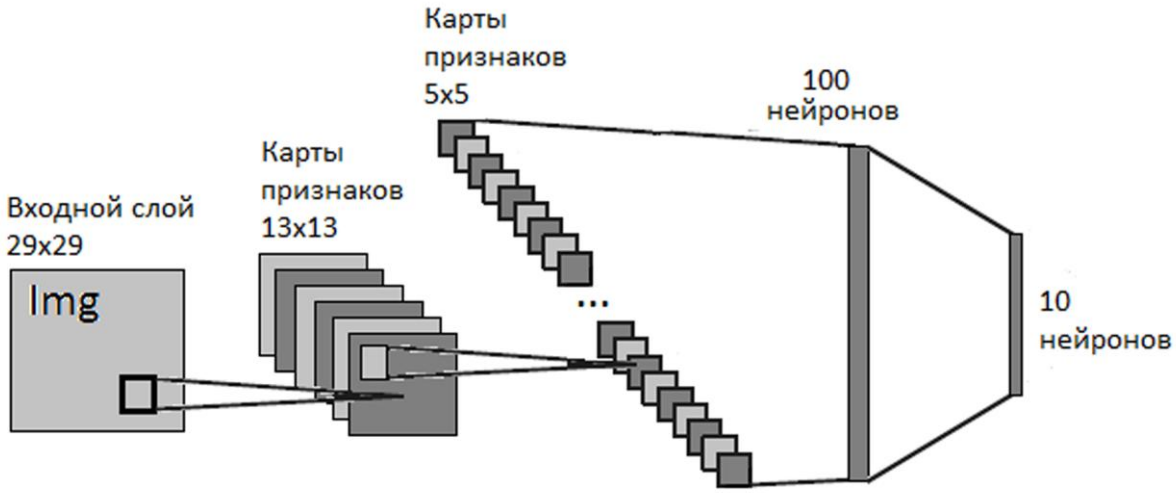


Рисунок 1.2 – Структура сверточной нейронной сети с тремя скрытыми слоями

Под локальным подразумевается такое восприятие, когда на вход одного нейрона подается не всё изображение, а некоторые его области. При таком подходе топология изображения сохраняется от слоя к слою.

Вторая идея – концепция разделяемых весов – заключается в том, что для большого количества связей используется очень небольшое количество весов. Иными словами, если, к примеру, на вход подается изображение

размерами 29x29 пикселей, то на вход каждого из нейронов следующего слоя поступит небольшой участок этого изображения. Размеры такого участка могут быть различными, к примеру, 5x5. Причем для обработки каждого из фрагментов будет использован один и тот же набор весов (ядер).

Суть третьей идеи о пространственной субдискретизации с помощью S-слоев в сверточных нейронных сетях заключается в уменьшении пространственной размерности изображения.

Чередование слоев позволяет составлять карты признаков из карт предыдущего слоя, что способствует на практике распознаванию сложных иерархий признаков.

В сверточных слоях каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов, результат подвергается операции суммирования. Полученная сумма определяет один пиксель выходного изображения, называемого картой признаков. На каждом слое происходит формирование нескольких таких карт. Изначально исходное изображение разделяется на области, из которых строятся карты признаков. При этом используется один и тот же набор весовых коэффициентов, а на выходе получается единственное число, соответствующее одному элементу выходной карты. Иными словами, алгоритм выполняется несколько раз для каждого элемента, затем загружаются другие коэффициенты и формируется следующая карта признаков.

На субдискретизирующих слоях выходами являются не одно, а несколько значений, примерно в два раза меньше числа входов, что позволяет понижать размерность изображения и выявлять в дальнейшем более общие, инвариантные к масштабированию признаки.

Наконец, полносвязные слои представляют собой классические сети прямого распространения, хорошо зарекомендовавшие себя в распознавании простых образов.

Для распознавания образов, нейронную сеть необходимо обучить, используя некоторый набор примеров – обучающую выборку [1]. В теории

нейронных сетей и машинного обучения под обучающей выборкой понимается конечный набор прецедентов-примеров, выделенных особым образом из генеральной совокупности, то есть всего множества возможных примеров. В качестве прецедентов могут выступать образцы, случаи, объекты, события, испытания и т.д. Выбор прецедентов может быть либо случайным, если исследователь не может по какой-то причине управлять процессом отбора примеров, либо управляемым. В последнем случае возникает проблема оптимального формирования обучающей выборки, т.е. выбора наиболее подходящих для наилучшего обучения и распознавания прецедентов, а также детальный контроль их общего вида и различных параметров.

Для каждого примера измеряются (собираются) определенные данные, которые образуют описание прецедента. В общем виде, каждый прецедент – это совокупность двух векторов: входного значения (закодированного двумерного изображения) и выходного (требуемый вид выходного состояния последнего слоя нейронной сети, либо простое соответствие определенному классу объектов). Таким образом, входной сигнал является непрерывной совокупностью данных, а выходной – одним из возможных дискретных значений.

Различают обучающую и тестовую выборки [7]. Обучающая выборка – выборка, с помощью которой производится настройка архитектуры нейронной сети. Однако нельзя полагаться на излишне оптимистичные показатели распознавания, если проверять оценку качества функционирования нейронной сети по той же выборке X^m, с помощью которой и был произведен процесс обучения. Ситуацию, когда нейронная сеть просто «запоминает» прецеденты, предоставленные ей для обучения, но не способна обобщить полученный опыт на новые данные, называется переобучением. Ключевое значение имеет способность сети к обобщению на наборе данных, выходящих за рамки обучающей выборки, поэтому существует понятие тестовой (или контрольной) выборки, по которой

оценивается качество построенной модели. Для того, чтобы сравнить несколько моделей, обученных на обучающей выборке, по критерию качества распознавания, необходимо создать еще одну, так называемую, проверочную или валидационную выборку, позволяющую избежать смещения при сравнении и получения нереалистично оптимистичных результатов.

Таким образом, среди всех подходов к распознаванию образов нейронные сети выгодно отличает их свойство адаптивности и универсальность [3]. Это мощный инструмент, обеспечивающий высокое качество распознавания образов. Кроме того, структура искусственных нейронных сетей хорошо подходит для обработки визуальной информации, при этом обеспечивая распределенную, параллельную обработку данных. Нелинейность, адаптивность, устойчивость к отказам, универсальность в решении задач делают искусственные нейронные сети наиболее подходящим аппаратом для решения задачи диссертационного исследования, по сравнению с остальными подходами [57].

1.2 Анализ алгоритмов и методов выделения целевых объектов на изображениях

Большинство моделей распознавания образов не могут работать со сложными сценами, содержащими большое количество объектов, и требуют предварительного выделения участков, содержащих целевой объект. Особенно актуально это для систем видеонаблюдения, при этом в данном случае возникает необходимость в обработке большого объема последовательных кадров, вместо одного изображения.

Таким образом, в разработанном методе необходимо наличие подмодуля выделения объектов и логического разбиения входного изображения на набор участков по определенному признаку. Далее будет

приведен обзор наиболее распространенных методов и алгоритмов выделения объектов на изображениях.

1.2.1 Анализ метода выделения объектов по цвету

Одним из наиболее простых способов отделить объект от фона является выделение по цвету [64]. При этом в качестве признака объекта используются только количественные характеристики цвета в одном из цветовых пространств (RGB, HSV, LAB). К примеру, в RGB цвет кодируется тремя числами (Red, Green, Blue). В таком случае, компоненты искомого цвета вычитаются поочередно из компонентов каждого пикселя изображения, формируя три маски, которые обрабатываются пороговой функцией и складываются в результирующую маску. В этой маске будут выделены те области, цвет которых наиболее близок к искомому цвету. Также можно вычислять евклидово расстояние в трехмерном пространстве для цвета каждого пикселя, но такой подход является более ресурсозатратным. В большинстве случаев, при выделении объектов по цвету, изображение переводится в пространство HSV, потому что это обеспечивает большую точность и однозначность поиска цвета.

Недостатки: цвета на изображении могут иметь нечеткие границы, переходить один в другой, также объект может состоять из нескольких цветов.

1.2.2 Анализ метода выделения объектов через нахождение контуров

Термин выделение контуров или выделение границ используется для обозначения ряда математических методов, направленных на поиск точек цифрового изображения, в которых яркость резко изменяется [153]. Найденные точки обычно объединяются в массив и образуют сглаженные

линии, называемые границами. Выделение контуров является одним из основных средств компьютерного зрения и цифровой обработки изображений, часто применяется в областях, связанных с выделением локальных свойств изображения и распознаванием.

Одним из наиболее распространенных методов выделения контуров является детектор границ Кэнни. Алгоритм обработки изображения с помощью данного фильтра включает в себя следующие этапы:

1) Сглаживание с целью удаления шума, с использованием фильтра вида:

$$M = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * V \tag{1.8}$$

2) Поиск градиентов по четырем направлениям с использованием четырех различных фильтров под углами 0, 45, 90 и 135 градусов.

3) Подавление немаксимумов с целью избавиться от лишних границ.

4) Пороговая фильтрация и трассировка области неоднозначности. Выделение потенциальных границ с помощью порогов, объединение связанных границ, а также удаление всех краев, не относящихся к сильным границам.

Недостатки: не всегда контуры являются четкими, часто границы размыты, невозможно точно объединить или разбить замкнутые контуры в один объект.

1.2.3 Анализ метода выделения объектов, основанного на поиске по шаблону

Данный метод позволяет находить на изображении участки, наиболее похожие на объект-ориентир [77]. Шаблон накладывается в режиме плавающего окна на разные части изображения, и вычисляется корреляция между двумя областями, при этом может также происходить перебор

масштабов. Те участки, на которых различие между двумя изображениями минимальны помечаются как искомые. Этого можно достичь, обработав изображение функцией расчета корреляции и выделив локальные минимумы или максимумы функции.

Во время выполнения алгоритма, могут быть использованы различные функции расчета корреляции:

$$R(x,y)=\sum_{x',y'}\left(T(x',y')-I(x+x',y+y')\right)^2, \tag{1.9}$$

$$R(x,y)=\frac{\sum_{x',y'}\left(T(x',y')-I(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'}T(x',y')^2\cdot\sum_{x',y'}I(x+x',y+y')^2}}, \tag{1.10}$$

$$R(x,y)=\sum_{x',y'}\left(T'(x',y')\cdot I'(x+x',y+y')\right). \tag{1.11}$$

Недостатки: метод требует больших вычислительных затрат и много времени на обработку. Любой сдвиг, существенное изменение масштаба или изменение освещения приводят к «потере» объекта.

1.2.4 Анализ метода выделения объектов, основанного на использовании признаков Хаара

Признаки Хаара задаются набором смежных прямоугольных областей, для каждой такой области вычисляется вес, который равен сумме яркостей всех попадающих на нее пикселей после позиционирования на изображении [145]. В качестве областей используются только прямоугольные области, т.к. вычислительная сложность для произвольных фигур заметно возрастает (рис. 1.3).

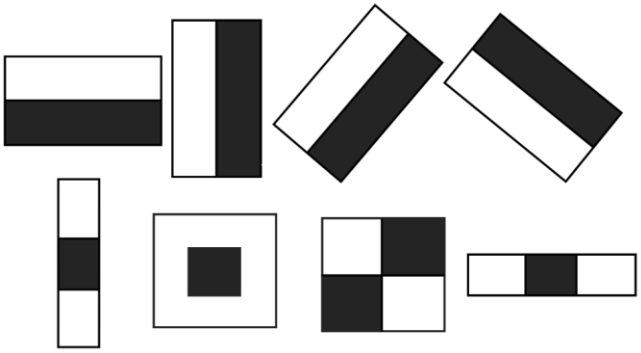


Рисунок 1.3 – Признаки Хаара

Для существенного упрощения вычислений используют интегральное представление изображения, в котором значение точки с координатами (x, y) равно сумме значений яркостей всех точек, лежащих выше и левее данной. Интегральное изображение рассчитывается один раз и может быть использовано для простого и быстрого вычисления суммарной яркости любой прямоугольной области.

$$I(x,y) = \sum_{i=0,j=0}^{i\le x,\, j\le y} I(i,j) \tag{1.12}$$

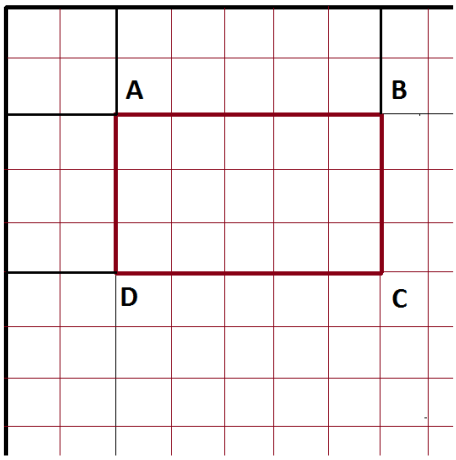


Рисунок 1.4 – Интегральное изображение

К примеру, на рисунке 1.4 суммарную яркость прямоугольной области можно вычислить, используя следующую формулу:

$$I_S = I_A + I_C - I_B - I_D . \tag{1.13}$$

Независимо от размера области, операция потребует выполнения трех математических операций и четырех обращений к памяти. Для быстрого вычисления значения интенсивности в конкретной точке, можно воспользоваться формулой:

$$I_{x,y} = I_{x,y-1} + I_{x-1,y} - I_{x-1,y-1}. \quad (1.14)$$

Совместно с признаками Хаара часто используют алгоритмы обучения AdaBoost и машины опорных векторов [88]. Распознавание производится через сравнение каскада сверток объекта и тестового изображения.

Недостатки: долгое обучение и необходимость в создании большой обучающей выборки, содержащей как позитивные примеры для каждого класса объектов. Так и негативные примеры «пустого» пространства, а также чувствительность к поворотам и резкой смене освещения.

1.3 Формулирование задачи исследования и декомпозиция на частные подзадачи

1.3.1 Анализ нейросетевых моделей для выделения и распознавания объектов и выбор конкретной модели

В предыдущих параграфах было обосновано использование нейронных сетей в качестве средства классификации образов и необходимость применения алгоритмов выделения объектов на изображении. Следующим закономерным этапом является выбор одной или двух конкретных моделей, наиболее подходящих для решения поставленной задачи.

За последние несколько лет в направлении выделения и распознавания объектов на изображении произошел ряд важных изменений [89]. Главные из них – переход от использования плавающего окна к сегментированию изображения и использование глубоких признаков.

Наиболее успешные подходы последних лет, победители главного мирового конкурса по распознаванию ImageNet:

1) Метод селективного поиска (Segmentation as Selective Search) [109].

2) Использование региональных признаков (Regionlets for Object Detection) [146].

3) Использование больших иерархий признаков (Rich Feature Hierarchies for Accurate Object Detection) [108].

Все эти методы распознавания используют сегментирование в качестве предобработки изображения с целью генерации некоторого количества (от тысячи до десяти тысяч) гипотез или объектов-кандидатов для дальнейшей обработки различными алгоритмами классификации. Такой подход дает гораздо меньшее количество гипотез для проверки, чем использование плавающего окна, за счет чего значительно повышается скорость и точность распознавания.

Основные тенденции, определившие изменения в подходе выделения объектов:

1. Замена подхода использования скользящего окна на сегментирование и генерацию ограниченного числа гипотез.

2. Использование вместо алгоритмов, основанных на выделении признаков, машин опорных векторов и других классификаторов.

3. Вместо обычных выделяемых признаков все чаще отдается предпочтение глубоким признакам.

Сверточные нейронные сети и глубокое обучение являются наиболее современными и перспективными подходами. На их основе разработано множество эффективных моделей и алгоритмов распознавания образов.

1. *Multibox*. Модель, разработанная компанией Google, использует специально обученную нейронную сеть для генерации гипотез о местоположении объекта на изображении, а также значений «уверенности» в том, что в этом окне действительно содержится объект. Далее, в случае детектирования одного класса, результат выводится непосредственно из выходных данных сети. Если же производится детектирование нескольких

классов объектов, то используется вторая нейронная сеть, осуществляющая классификацию [91].

2. *OverFeat*. Использование сверточного подхода вместо плавающего окна, во время которого осуществляется последовательная свертка слоев со всем изображением. На выходе, при таком подходе, получается карта энергий, показывающая в каких местах изображения наиболее вероятно нахождение объекта [133].

3. *R-CNN*. Данная модель состоит из двух частей: блока генерации гипотез местоположения объекта на изображении и классификатора в виде сверточной нейронной сети. На первом этапе генерируется около 2000 гипотез, которые масштабируются, образуя выходной дескриптор [99].

Модель Multibox является закрытой разработкой компании Google, не имеет открытого исходного кода и опубликованных точных алгоритмов работы, поэтому, хоть и перспективна, но не может быть использована в научных исследованиях по объективным причинам. Модель OverFeat имеет открытый исходный код, но вместе с этим и ряд серьезных недостатков, вытекающих непосредственно из положительных характеристик. Так, объединение в рамках одной нейронной сети функций выделения и распознавания объектов лишает модель гибкости, серьезно усложняет алгоритмическую сложность и требует крайне точного и комплексного обучения. В основе модели R-CNN, как будет показано далее, лежит простой, но эффективный принцип модульного разделения задач между разными алгоритмами. При этом сама идея позволяет довольно легко модернизировать и улучшать модель, оптимизировать к конкретной задаче и повышать характеристики распознавания.

1.3.2 Анализ модели R-CNN для выделения и распознавания объектов

Модель R-CNN состоит из трех модулей. Первый модуль генерирует гипотезы о местоположении объекта на изображении независимо от класса объекта. Второй модуль представляет из себя большую сверточную нейронную сеть, которая извлекает признаки из каждой гипотезы, преобразуя их в вектор фиксированной длины. Третий модуль – классификатор на основе машины опорных векторов, осуществляющий классификацию векторов признаков на наборе конкретных классов объектов.

Сама модель R-CNN допускает использование любого алгоритма генерации гипотез местоположения объекта, обладающих достаточным качеством работы, но в статье, в которой эта модель была впервые представлена [99], используется алгоритм селективного поиска, который является наиболее оптимальным и эффективным вариантом на сегодняшний день.

В классическом варианте модели все гипотезы преобразуются независимо от первоначального размера, чтобы соответствовать входам сети 227x227 пикселей. Из входного вектора формируется 4096-размерный вектор признаков для каждой из гипотез, созданных на первом этапе. Преобразование происходит путем распространения сигнала через пять слоев сверточной нейронной сети и два полносвязных слоя на выходе.

После обработки входного изображения с помощью метода селективного поиска формируется около двух тысяч гипотез местоположения объекта. Каждая из гипотез деформируется и прогоняется через сверточную нейронную сеть для выделения признаков. Каждый из выходных векторов отправляется для распознавания в классификатор (рис. 1.5).

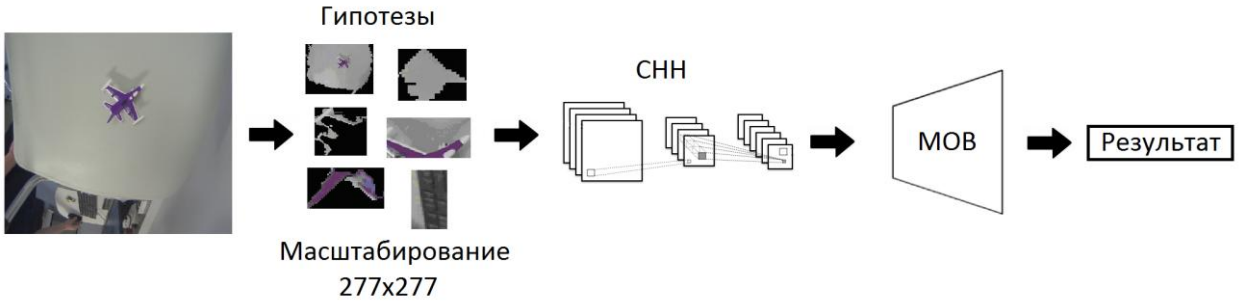


Рисунок 1.5 – Схема модели R-CNN

В качестве методов генерации гипотез местоположения объекта могут быть также использованы:

1) Objectness [67].

2) Category-independent object proposals (Гипотезы местоположения объекта независимо от класса) [106].

3) Multi-scale combinatorial grouping (Комбинаторная группировка с использованием разных масштабов) [69].

4) Constrained parametric min-cuts (Генерация гипотез на основе разреза графов) [81].

5) Другие методы сегментирования изображения и выделения заметных областей.

Математически модель R-CNN описывается следующим образом:

$$S_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k), \tag{1.15}$$

$$S_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k), \tag{1.16}$$

$$S_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}, \tag{1.17}$$

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}, \tag{1.18}$$

$$s(r_i, r_j) = \alpha_1 s_{colour}(r_i, r_j) + \alpha_2 s_{texture}(r_i, r_j) + \alpha_3 s_{size}(r_i, r_j) + \alpha_4 s_{fill}(r_i, r_j), \tag{1.19}$$

$$r_t = r_i \cup r_j, \text{где s} = \max, \tag{1.20}$$

$$R(r_1, r_2, ..., r_m), \text{где m} < \text{n}, \tag{1.21}$$

$$X(x_1, x_2, ..., x_g), \tag{1.22}$$

$$s_{c1} = \sum_i \sum_j (x_i \cdot w_{ij}^{c1}), i = \overline{1,n},\ j = \overline{1,m};\ h_{ij}^{c1} = \frac{1}{1-e^{-s_{c1}}}, \tag{1.23}$$

$$s_{s1} = \sum_i \sum_j (h_i^{c1} \cdot k_{ij}^{s1}), i = \overline{1,n},\ j = \overline{1,m};\ h_{ij}^{s1} = \frac{1}{1-e^{-s_{s1}}}, \tag{1.24}$$

$$s_{c2} = \sum_i \sum_j (h_i^{s1} \cdot w_{ij}^{c2}), i = \overline{1,n},\ j = \overline{1,m};\ h_{ij}^{c2} = \frac{1}{1-e^{-s_{c2}}}, \tag{1.25}$$

$$s_{s2} = \sum_i \sum_j (h_i^{c2} \cdot k_{ij}^{s2}), i = \overline{1,n},\ j = \overline{1,m};\ h_{ij}^{s2} = \frac{1}{1-e^{-s_{s2}}}, \tag{1.26}$$

$$s_{c3} = \sum_i \sum_j (h_i^{s2} \cdot w_{ij}^{c3}), i = \overline{1,n},\ j = \overline{1,m};\ h_{ij}^{c3} = \frac{1}{1-e^{-s_{c3}}}, \tag{1.27}$$

$$s = \sum_i \sum_j (h_i^{c3} \cdot w_{ij}), i = \overline{1,n},\ j = \overline{1,m};\ y_{ij} = \frac{1}{1-e^{-s}}, \tag{1.28}$$

где $S_{colour}(r_i, r_j)$ – мера схожести по цвету, $S_{texture}(r_i, r_j)$ – мера схожести текстуры, $S_{size}(r_i, r_j)$ – оценка размера, $S_{fill}(r_i, r_j)$ – мера соответствия двух областей, $s(r_i, r_j)$ – комбинация четырех мер схожести, r_t – регион с наибольшей схожестью, R – финальный набор регионов, X – входной вектор НС, s_{c1} – взвешенная сумма h_{ij}^{c1} – выход первого С-слоя, y_{ij} – выход сети.

Селективный поиск в контексте детектирования объектов на изображении и распознавания образов представляет множество разнообразных стратегий для рассмотрения изображения с как можно более обширного числа точек зрения. Метод селективного поиска включает в себя следующие аспекты.

1) Учет всех масштабов. Объекты могут иметь на изображении различные размеры и пропорции. Некоторые объекты также имеют более размытые границы, чем другие. Таким образом, во внимание должны быть приняты все возможные масштабы, наиболее естественный способ для осуществления этой идеи – использование иерархического алгоритма.

2) Разнообразие. Не существует единственной оптимальной стратегии для того, чтобы сгруппировать отдельные части изображения. В отдельных случаях части могут формировать объект по цвету, текстуре или общим

границам. Также, тени и другие продукты неравномерной освещенности могут влиять на то, как части формируют объект. Следовательно, вместо использования одной стратегии, которая хорошо работает в большинстве случаев, более выгодно иметь набор разнообразных стратегий, которые покрывали бы все возможные ситуации.

3) Скорость вычислений. Целью селективного поиска является формирование набора возможных положений объекта на изображении. Вычисление такого набора не должно требовать больших вычислительных и временных затрат.

В сентябре 2004 года P. Felzenszwalb, D. Huttenlocher предложили подход к сегментированию изображений, который стал одним из основных стандартов для систем компьютерного зрения на многие годы [96]. Данный метод сегментирования получил название «Эффективная сегментация на графах» и имеет два важных свойства:

1. Восприятие в первую очередь важных областей изображения, отражающих глобальные участки.

2. Высокая скорость, время, затрачиваемое на выполнение алгоритма, линейно зависит от количества пикселей изображения.

Таких результатов метод достигает за счет того, что каждый пиксель изображения принимается за вершину графа, где вес ребра между двумя вершинами вычисляется по формуле:

$$w(v_i, v_j) = \left| I(p_i) - I(p_j) \right|, \tag{1.29}$$

где $I(p_i)$ – интенсивность точки изображения.

В ходе выполнения сегментации, вершины-точки будут объединяться с соседними пикселями, ребра которых имеют наименьшую длину, таким образом образуя большую по размерам вершину. В конечном итоге изображение будет поделено на блоки, соответствующие отдельным объектам.

Метод селективного поиска использует несколько цветовых пространств, наиболее подходящих для инвариантного выделения областей.

В качестве таких цветовых пространств могут выступать: RGB, интенсивность I (черно-белое изображение), Lab, r и g составляющие нормализованного кадра в пространстве RGB, HSV, отдельно H составляющая пространства HSV.

В качестве измеримой меры схожести двух областей используются четыре простых и быстрых для вычисления параметра. Каждый параметр измеряется в диапазоне от 0 до 1, что позволяет комбинировать и использовать их совместно.

1. $S_{colour}(r_i, r_j)$ – схожесть по цвету. Для каждой из областей изображения вычисляется одномерная гистограмма из 25-ти блоков на каждый цветовой канал. Таким образом, n=75, если используется три цветовых канала, как в цветовых пространствах RGB, HSV.

$$S_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k) \tag{1.30}$$

Цветовая гистограмма может быть легко распространена по иерархии, с применением следующей формулы:

$$C_t = \frac{size(r_i) \cdot C_i + size(r_j) \cdot C_j}{size(r_i) + size(r_j)}. \tag{1.31}$$

2. $S_{texture}(r_i, r_j)$ – мера схожести текстуры.

Для каждого из восьми направлений строится гистограмма из десяти блоков. Таким образом, получается текстурная гистограмма размерностью n=240 для трех измерений цветового пространства.

$$S_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k) \tag{1.32}$$

3. $S_{size}(r_i, r_j)$ – оценка размера, позволяющая в первую очередь объединять малые по размеру области.

$$S_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}, \tag{1.33}$$

где $size(im)$ – размер всего изображения в пикселях

4. $S_{fill}(r_i, r_j)$ – мера того, насколько две области соответствуют друг другу. Если область r_1 находится внутри области r_2, логично объединить их, чтобы не допускать образования «дыр» в изображении. Чтобы осуществить этот процесс быстро, используются только размеры областей и площадь окружающих рамок.

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)} \quad (1.34)$$

Комбинация четырех мер схожести вычисляется следующим образом:

$$s(r_i, r_j) = \alpha_1 s_{colour}(r_i, r_j) + \alpha_2 s_{texture}(r_i, r_j) + \alpha_3 s_{size}(r_i, r_j) + \alpha_4 s_{fill}(r_i, r_j), \quad (1.35)$$

где $\alpha_i \in \{0,1\}$, означает используется ли данная мера схожести или нет.

1.3.3 Формальная постановка задачи распознавания образов

Распознавание образов – это раздел науки, целью которого является классификация объектов, называемых образами, по каким-либо признакам, отнесение объекта к определенному классу или категории. При этом в основе классификации лежат прецеденты – принимаемые в качестве образца при решении задачи классификации, объекты, уже классифицированные ранее. Другими словами, прецедент – это образ, правильная классификация которого заранее известна [11].

Для того, чтобы задача классификации была возможна, принято условно считать, что все образы, объекты и явления разбиты на некоторое конечное множество классов, при этом каждый класс включает в себя конечное множество уже изученных объектов-прецедентов. При таких исходных данных, задача распознавания заключается в том, чтобы отнести новый распознаваемый объект с определенной долей вероятности к какому-либо классу из имеющегося множества [48].

Задача распознавания образов является краеугольным камнем и лежит в основе большинства интеллектуальных информационных систем, а сама

идея принятия решения на основе имеющихся прецедентов является основой всего естественнонаучного мировоззрения.

Область применения теории распознавания образов невероятно широка, она включает огромное множество видов проблем и задач, таких как: машинное зрение, распознавание рукописных символов, медицинская диагностика, геологические исследования, распознавание речи, дактилоскопия и т.д [12].

Данные, получаемые при измерении объекта с целью использования в процессе классификации, называются признаками. Набор признаков, относящихся к одному объекту, формируют так называемый вектор признаков, расположенного и принимающего значение в пространстве признаков. При этом принято считать, что каждому конкретному вектору признаков соответствует только один образ, также как каждому образу ставится в соответствие единственное значение вектора признаков.

Важное место в теории распознавания образов занимает классификатор – некое решающее правило, относящее образ к одному из известных классов на основании предъявленного вектора признаков.

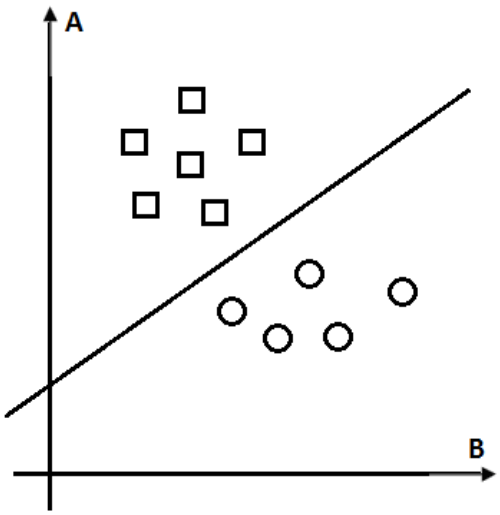


Рисунок 1.6 – График прямой, разделяющей прецеденты разных классов объектов

Анализ материалов по теме диссертационного исследования показал, что для того, чтобы создать систему классификации, необходимо определить [15]: каким образом выбирать векторы признаков, какие признаки являются наиболее существенными для определения границ классов и разделения объектов по ним, каким образом возможно построить классификатор, как следует оценивать качество созданной системы.

Перечисленные вопросы формируют задачи проектирования, известные как задача генерации признаков, задача селекции признаков, задача построения классификатора и задача количественной оценки системы с точки зрения степени ошибочности классификации.

В зависимости от того, имеется ли у нас в наличии информация в виде прецедентов или нет, различают распознавание с обучением и без обучения [21]. В случае, когда классификация происходит на основе имеющегося множества прецедентов, сама задача распознавания называется классификацией с обучением.

Если же имеется некоторое множество векторов признаков, экспериментально полученных для некоторого набора объектов, но нет точной информации о том, к каким конкретно классам принадлежит каждый вектор, то такое распознавание называется распознаванием без обучения. При этом предварительно нужно решить проблему разделения множества образов по сходству на основе близости имеющихся векторов признаков, также называемую кластеризацией.

Формальная постановка задачи распознавания выглядит следующим образом: дано некоторое множество объектов Ω, разделенное на подмножества, называемые классами $\omega : \omega \in \Omega$, для каждого класса задана некоторая информация, описывающая его [53]. Кроме этого, в исходные данные задачи входит совокупность всех доступных наблюдений, называемая пространством признаков P. Руководствуясь такими определениями, можно записать три выражения, характеризующие задачу:

1) $g(\omega) = \Omega \to M, M = \{1,2,...,m\}$ – неизвестная наблюдателю функция, называемая индикаторной, разбивает пространство образов на некоторое множество непересекающихся классов.

2) $x(\Omega) = \Omega \to P -$ функция, характеризующая образ объекта, доступный наблюдателю, определяющая для каждого объекта соответствующую точку в пространстве признаков, принадлежащую одному из возможных непересекающихся множеств точек.

3) $g'(x) = P \to M -$ функция, связывающая определенным образом множество классов с пространством признаков, называемая решающим правилом. Задача распознавания в общем виде заключается в построении решающего правила с минимальным количеством ошибок в определении класса.

Для решения задачи распознавания на практике можно использовать один из основных существующих методов: оптическое (морфологическое) распознавание, изучение контура объекта, метод главных компонент, линейный дискриминантный анализ, Скрытые Марковские Модели, метод опорных векторов или искусственные нейронные сети. Оптическое распознавание образов подразумевает перебор возможных видов объекта под различными углами, с использованием трансформаций смещения, изменения масштаба и т.д [62]. Например, если перед исследователем лежит задача распознавания символов, следует перебирать вид и различные свойства шрифта. Второй подход заключается в нахождении контуров объектов и последующем исследовании наличия углов, связности и прочих его свойств [14]. Наконец, для того, чтобы использовать математический аппарат искусственных нейронных сетей, необходимо иметь в наличии некоторый набор примеров правильного решения задачи распознавания для обучения нейронной сети и настройки ее структуры с учетом специфики данной конкретной задачи [30].

1.3.4 Формулирование задачи исследования

Анализ научно-методического аппарата позволил выявить определенные противоречия в теории и практике.

Для решения практических задач необходимо повышение качества распознавания объектов на изображениях. В то же время, для того, чтобы практические задачи были решены, требуют модификации существующие методы и нейросетевые модели выделения и распознавания объектов на изображениях.

Таким образом, целью диссертационного исследования является повышение обобщающей способности СНС без потерь скорости ее работы.

Обобщающую способность можно оценить через отношение числа корректно распознанных примеров к общему количеству примеров в валидационном множестве [57]. На обобщающую способность нейронной сети могут оказывать влияние такие параметры, как: размер валидационного множества, количество ядер свертки или карт признаков, определяемое количеством нейронов, а также архитектура нейронной сети, определяемая комбинацией и количеством слоев, включающих нейроны высокого порядка.

Научная задача диссертационного исследования – разработка метода выделения и распознавания объектов, базирующегося на использовании нейронов второго порядка, обеспечивающих повышение обобщающей способности СНС без потерь в скорости ее работы.

Реализация поставленной задачи может быть декомпозирована на следующие частные задачи:

– разработка численного метода отсеивания гипотез по низкочастотной структуре;

– разработка метода выделения объектов на основе модели R-CNN;

– разработка параллельного алгоритма обработки данных в СНС первого и второго порядков, ориентированного на процессоры векторно-матричной архитектуры;

– разработка методики полуавтоматического формирования эффективных визуальных обучающих выборок;

– разработка программного комплекса, реализующего разработанные алгоритмы и позволяющего выделять и распознавать объекты на изображениях, с применением разработанной методики создания обучающих выборок.

Можно сделать вывод, что наряду с достоинствами, у нейронных сетей есть свои недостатки, как у любого подхода. В первую очередь это касается необходимости создания обучающей выборки, без которой сеть невозможно будет использовать. На сегодняшний день недоисследованным является вопрос создания эффективных обучающих выборок. Для большинства научных исследований используются готовые большие выборки (NORB, Caltech 101, Pascal) [112]. Но если речь идет о решении конкретной прикладной задачи, присутствует неясность в том, как влияют определенные параметры обучающей выборки на качество обучения и распознавание сети, какие способы автоматизации использовать для создания выборки, какие алгоритмы и методы при этом использовать.

Для нейросетевых алгоритмов, как и для любых других, критична скорость выполнения, но, к сожалению, большинство современных моделей не рассчитаны на быструю обработку. Поэтому, критично важным является вопрос ускорения работы сверточных нейронных сетей.

Также, большинство моделей, основанных на нейронных сетях не способно самостоятельно выделять отдельные объекты на изображении и требует предварительного сегментирования и предварительной обработки входного изображения на входе сети.

Наконец, недоисследованным является вопрос использования нестандартных моделей сверточных нейронных сетей, с включением нейронов высоких порядков, в которых вместо стандартной операции взвешенного суммирования используются сумматоры второго, третьего

порядка, тригонометрические функции и т.д. с целью повышения качества распознавания образов [40].

Можно выделить следующие направления для исследования:

1. Разработка метода выделения и распознавания объектов на изображении, в основе которого будет лежать модель R-CNN. В методе должна быть использована нейронная сеть с архитектурой, включающей нейроны высоких порядков, а также оптимизация селективного поиска объектов за счет уменьшения количества гипотез.

2. Разработка параллельного алгоритма обработки данных в сверточных нейронных сетях второго порядка с использованием процессоров векторно-матричной архитектуры.

3. Исследование того, как определенные параметры обучающей выборки влияют в конечном итоге на качество обучения, разработка методики полуавтоматического создания эффективных обучающих выборок для нейронных сетей.

4. Разработка программного комплекса для выделения и распознавания объектов на изображениях, основанного на созданных модели, параллельном алгоритме и методике.

Выводы

1. Проведен обзор научно-технический литературы, посвященной существующим математическим моделям, методам и алгоритмам классификации образов. Проанализированы основные характеристики и особенности программно-аппаратных комплексов распознавания объектов на изображениях, проведено сравнение различных реализаций нейронных сетей и нейроэмуляторов. Обоснована актуальность нейронных сетей в качестве наиболее подходящего класса моделей для диссертационного исследования.

2. Проведен обзор существующих методов выделения объектов на изображении, выделены недостатки и достоинства каждого метода, обоснован выбор модели R-CNN для исследования и улучшения в данной диссертационной работе.

3. Обоснована важность обучающей выборки и ее влияние на качество распознавания образов нейронной сетью. Показана необходимость правильно подбирать параметры, способы создания и предобработки изображений выборки.

4. Показано, что в использовании нейронов высоких порядков в сверточных нейронных сетях скрыт большой потенциал улучшения работы системы без экстенсивного расширения ее архитектуры за счет внедрения новых нейронов.

5. Показано, что архитектура векторно-матричных процессоров во многом соответствует принципам, лежащим в основе СНС, и может быть использована для ускорения рассматриваемой модели.

6. Сформулирована научная задача диссертационной работы и проведена ее декомпозиция на частные подзадачи, включающие разработку метода выделения и распознавания объектов на изображениях, численного метода отсеивания гипотез, параллельного алгоритма обработки данных в СНС второго порядка и методики создания визуальных обучающих выборок для нейронных сетей.

ГЛАВА 2 РАЗРАБОТКА МЕТОДА ВЫДЕЛЕНИЯ И РАСПОЗНАВАНИЯ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ

Данная глава содержит описание разработанного метода выделения и распознавания объектов на изображениях и разработанного численного метода отсеивания гипотез по низкочастотной структуре, описание экспериментов по выбору архитектуры СНС второго порядка и вывод формул обратного распространения для нейронов второго порядка на выходном и внутренних слоях СНС.

2.1 Разработка структуры метода выделения и распознавания объектов на изображениях

Предлагаемый метод выделения и распознавания объектов на изображениях базируется на двух основных принципах. Во-первых, на внедрении нейронов второго порядка на первом и втором сверточных слоях СНС. Во-вторых, на использовании численного метода отсеивания гипотез по низкочастотной структуре и цветовому содержанию в алгоритме селективного поиска. Внедрение нейронов второго порядка повышает обобщающую способность сети [57] (параметр, оцениваемый через отношение числа корректных распознаваний примеров, не входящих в обучающую выборку, к общему размеру валидационного множества), но в то же время и алгоритмическую сложность, а, следовательно, снижает скорость выделения и распознавания объектов. Для того, чтобы компенсировать эту задержку и в целом ускорить процесс обработки данных, используется численный метод отсеивания гипотез.

На рисунке 2.1 приведена обобщенная схема разработанного метода, состоящая из двух основных блоков: селективного поиска объектов на изображении с отсеиванием гипотез и СНС второго порядка.

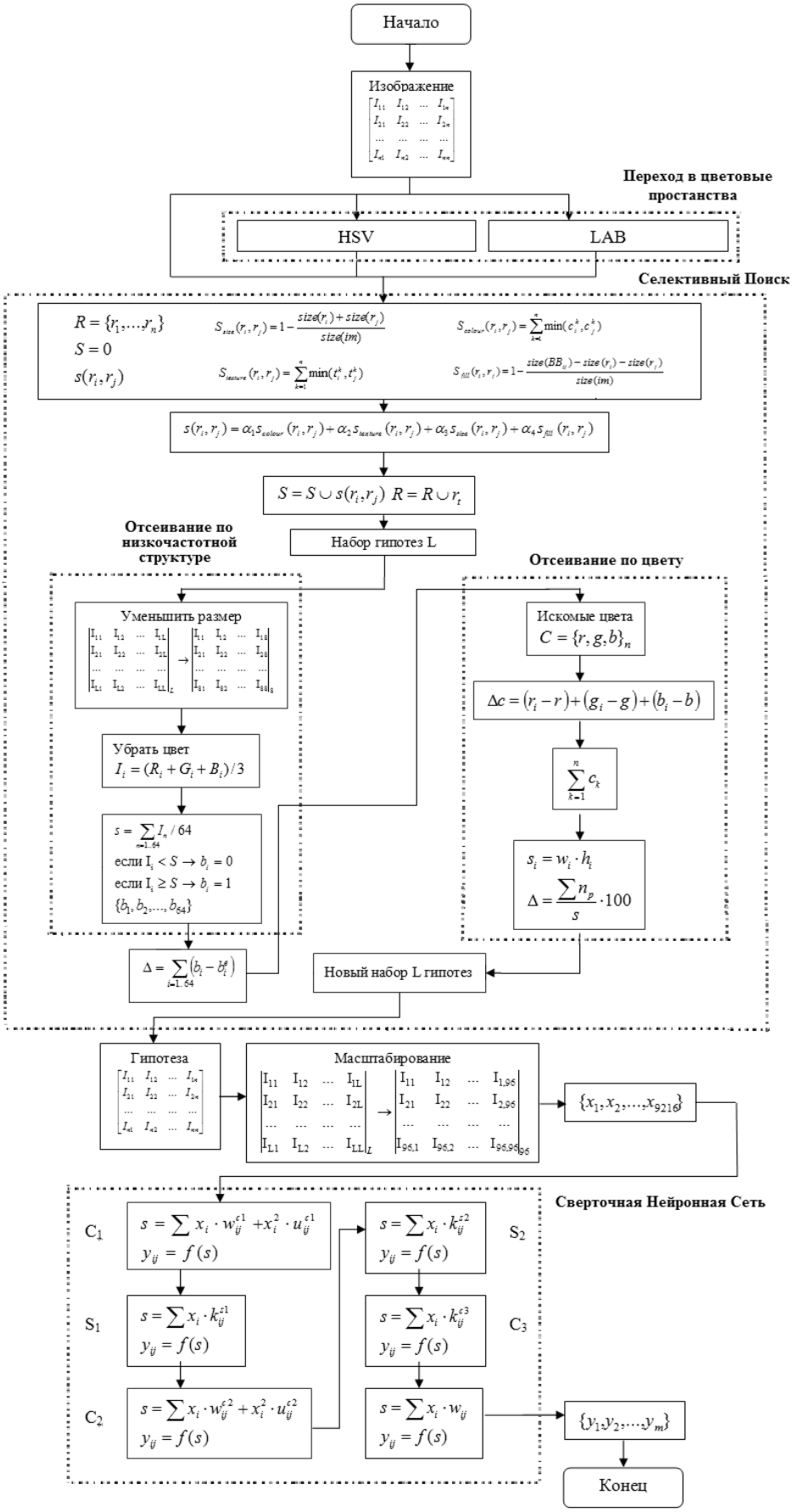


Рисунок 2.1 – Обобщенная схема алгоритма, реализующего разработанный метод выделения и распознавания объектов

Для исходного изображения путем сегментирования в нескольких цветовых пространствах (RGB, HSV, LAB) с помощью селективного поиска генерируется набор гипотез наиболее вероятного расположения целевых объектов. В базовом алгоритме существенная часть гипотез отсеивается при распознавании в «пустой» класс и исключается из набора. Оставшиеся гипотезы, для которых подтвердилась принадлежность к одному из целевых классов объектов, позволяют сформировать выходные данные алгоритма, включающие номер класса, размеры и расположение объекта на изображении. В разработанном методе до распознавания используется численный метод для отсеивания гипотез, таким образом, не все генерируемые алгоритмом гипотезы поступают на вход СНС – часть из них отсеивается с помощью более простых и быстрых методов. Блок отсеивания гипотез включает два этапа: отсеивание по низкочастотной структуре и отсеивание по цвету.

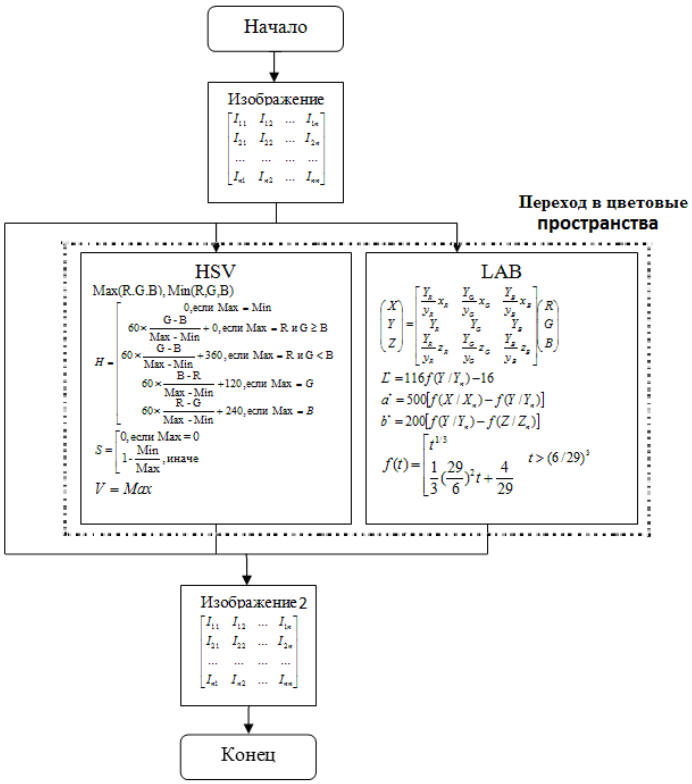


Рисунок 2.2 – Обобщенная схема перевода изображений в разные цветовые пространства

На рисунке 2.2 изображена расширенная обобщенная схема перевода изображений в разные цветовые пространства. При работе в цветовом пространстве RGB, исходное изображение остается неизменным, для перевода в пространства HSV и LAB используются стандартные формулы, позволяющие перекодировать отдельные составляющие, задающие цвет пикселя и вид, используемый в той или иной цветовой системе.

На рисунке 2.3 изображена расширенная схема селективного поиска объектов на изображении. Изображение разбивается на отдельные области с помощью сегментирования и для каждой пары областей вычисляется комплексная мера схожести на основе пяти частных мер схожести. Области с наибольшей схожестью объединяются. Этот процесс происходит рекурсивно, формируя в результате набор гипотез о местоположении искомого объекта на изображении.

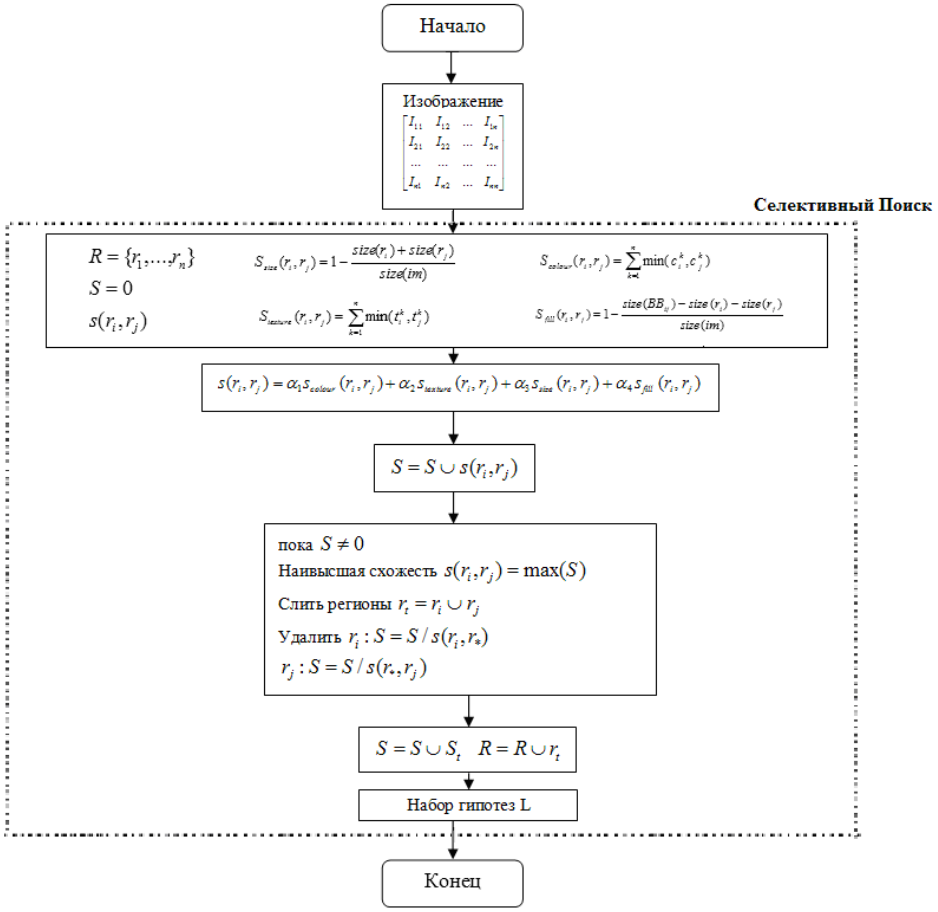


Рисунок 2.3 – Обобщенная схема алгоритма генерации гипотез на основе селективного поиска объектов на изображении

На рисунке 2.4 изображена расширенная схема отсеивания гипотез по цвету. Исходными данными являются набор из N гипотез о местоположении объекта, набор искомых цветов и два пороговых значения: для сравнения цветов и для процента пикселей, содержащих искомый цвет от общей площади гипотезы. Во время выполнения отсеивания для каждой гипотезы из исходного набора вычисляется количество пикселей, достаточно близких к искомому цвету на основании порогового значения, и процент, который составляют эти пиксели от общего числа точек, составляющих изображение гипотезы. На основании этого выносится решение войдет ли гипотеза в новый набор, получаемый на выходе алгоритма. Подробное описание алгоритмов отсеивания гипотез по низкочастотной структуре и по цвету будет дано далее в этой главе.

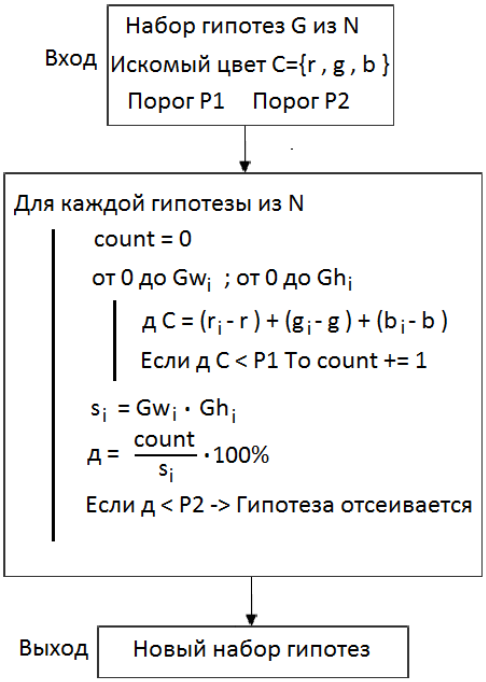


Рисунок 2.4 – Обобщенная схема отсеивания гипотез по цвету

На рисунке 2.5 изображена расширенная схема СНС, показано количество и размер карт на каждом из слоев сети, а также размер и количество ядер, используемых для операции свертки, а также размер входов и выходов и формулы, соответствующие отдельным сверточным и субдискретизирующим слоям. Схема иллюстрирует то, каким образом

формируется финальный вектор признаков, на основе которого производится распознавание.

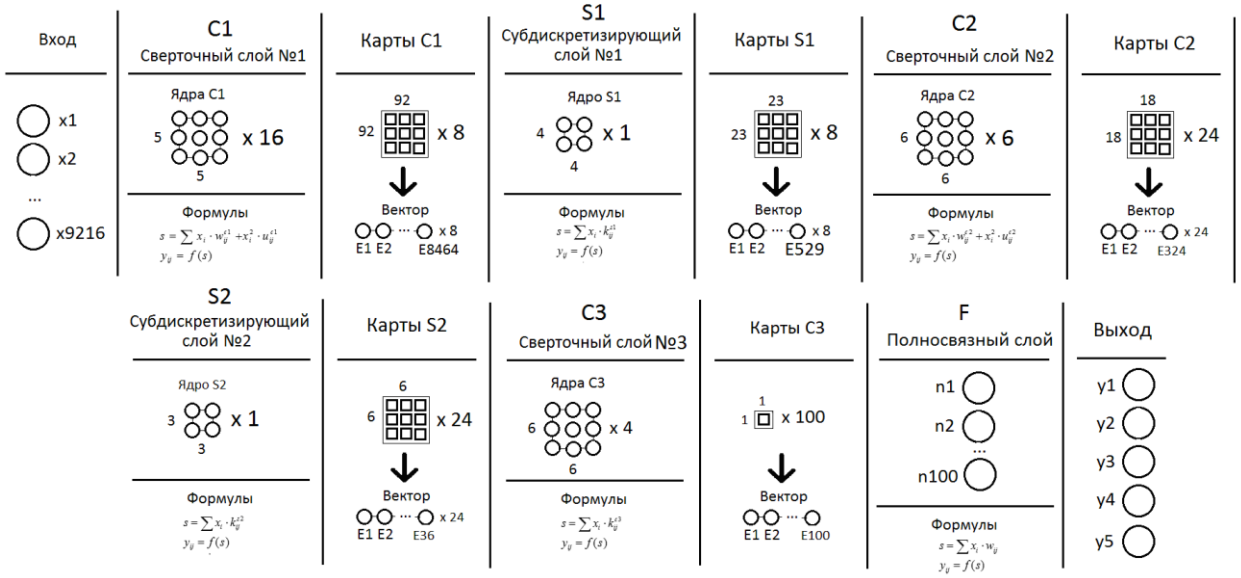


Рисунок 2.5 – Расширенная схема СНС второго порядка, используемой в разработанном методе

Для детализации вышеописанного метода необходимо решить две подзадачи: выбрать слои для внедрения нейронов высокого порядка в архитектуре СНС и разработать численный метод отсеивания гипотез по низкоуровневой структуре.

2.2 Внедрение нейронов второго порядка в архитектуру СНС

2.2.1 Вывод формул обратного распространения для сверточной нейронной сети второго порядка

Для того, чтобы использовать нейроны второго порядка в СНС, осуществлять обучение и корректировку весов, необходимо вывести формулы обратного распространения для использования с полиномиальными сумматорами.

Сигнал ошибки выходного j-го нейрона для n-го обучающего примера определяется соотношением:

$$e_j(n) = t_j(n) - y_j(n), \tag{2.1}$$

где $t_j(n)$ – учитель.

Значение энергии ошибки j-го нейрона определяется как $\frac{1}{2}e_j^2(n)$, следовательно, общая энергия ошибки по всем нейронам выходного слоя выражается формулой:

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n), \tag{2.2}$$

где C – множество всех нейронов выходного слоя.

Если принять общее число образов в обучающей выборке за N, то энергия среднеквадратичной ошибки (MSE) будет вычисляться по формуле:

$$E_{av}(n) = \frac{1}{N}\sum_{n=1}^{N} E(n). \tag{2.3}$$

Взвешенная сумма для j-го нейрона [26]:

$$v_j(n) = \sum_{j=0}^{m} w_{ji}(n) y_i(n), \tag{2.4}$$

где w_{j0} – порог, $y_0 = +1$.

Для того, чтобы рассчитать выход нейрона, необходимо применить функцию нелинейного преобразования:

$$y_j(n) = f_j(v_{ji}(n)). \tag{2.5}$$

Учитывая, что функция, моделируемая сетью – сложная, для выражения компонента градиента применим цепное правило:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}. \tag{2.6}$$

Вычислим множители приведенного выше выражения, учитывая, что в случае выходного слоя $e_j(n)$ является известной величиной.

$$\frac{\partial E(n)}{\partial e_j(n)} = \left(\sum \tfrac{1}{2} e_j^2(n)\right)' \mid e_j(n) = \sum \tfrac{1}{2} \cdot 2 e_j(n) \cdot 1 = \sum e_j(n). \tag{2.7}$$

Полученный результат можно отнести ко всему выходному слою, для отдельного нейрона формула будет выглядеть следующим образом:

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n), \tag{2.8}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = \left(t_j(n) - y_j(n)\right)' \mid y_j(n) = -\left(y_j(n)\right)' \mid y_j(n) = -1, \tag{2.9}$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = f_j{}'\left(v_j(n)\right) \mid v_j(n), \tag{2.10}$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n)\right)' \mid w_{ji}(n) = \sum_{i=0}^{m} y_i(n), \tag{2.11}$$

$w_{ji}(n)$ – это целый набор связей. Для отдельной связи формула будет выглядеть следующим образом:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n). \tag{2.12}$$

Следовательно, изначальная формула вычисления компонента градиента для связи выходного нейрона примет вид:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n) \cdot f_j{}'\left(v_j(n)\right) \cdot y_i(n). \tag{2.13}$$

Степень изменения отдельно веса вычисляется по формуле:

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}. \tag{2.14}$$

Знак «минус» указывает на то, что осуществляется градиентный спуск, потому что градиент указывает на максимальное возрастание функции. Подставив выражение (2.13) в (2.14) получим:

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot f_j{}'\left(v_j(n)\right) \cdot y_i(n) = \eta \cdot \delta_j(n) \cdot y_i(n). \tag{2.15}$$

Если рассматривать не выходной, а скрытый слой, то $f_j{}'\left(v_j(n)\right)$ не будет вызывать проблем, т.к. зависит от вида активационной функции. Несмотря на то, что на скрытом слое нет учителя, и нейроны скрытого слоя не доступны непосредственно, они все еще несут ответственность за ошибку, получаемую на выходе сети, т.к. участвовали в формировании выходного сигнала. Следовательно, на нейроны скрытого слоя можно распространить ошибку выходного слоя.

Запишем цепное правило для $\delta_j(n)$ нейрона скрытого слоя:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial v_j(n)} = -\left(\frac{\partial E(n)}{\partial e_j(n)} \cdot \frac{\partial e_j(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} \right). \quad (2.16)$$

Но для нейронов скрытого слоя отсутствует $\frac{\partial E(n)}{\partial e_j(n)}$, следовательно, цепочка без ошибки примет вид:

$$\partial_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \cdot \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \cdot f'(v_j(n)). \quad (2.17)$$

Остается только выразить $\frac{\partial E(n)}{\partial y_j(n)}$ через выходной слой, учитывая, что на нейрон j влияет множество k нейронов выходного слоя.

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum \frac{\partial E(n)}{\partial e_k(n)} \cdot \frac{\partial e_k(n)}{\partial y_j(n)} = \sum e_k(n) \cdot \frac{\partial e_k(n)}{\partial y_j(n)}. \quad (2.18)$$

Перейдем к частной производной и применим цепное правило и получим:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum e_k(n) \cdot \frac{\partial e_k(n)}{\partial v_k(n)} \cdot \frac{\partial v_k(n)}{\partial y_j(n)}. \quad (2.19)$$

Для скрытого слоя $e_k(n) = t_k(n) - y_k(n) = t_k(n) - f_k(v_k(n))$, следовательно:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = (t_k - f_k(v_k(n)))' \mid v_k(n) = -f_k'(v_k(n)). \quad (2.20)$$

В то же время, $v_k(n) = \sum_{j=0}^{m} w_{kj}(n) \cdot y_j(n)$, следовательно:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = \left(\sum_{j=0}^{m} w_{kj} \cdot y_j(n) \right)' \mid y_j(n) = w_{kj}(n), \quad (2.21)$$

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum e_k(n) \cdot f_k'(v_k(n)) \cdot w_{kj}(n) = -\sum \partial_k(n) \cdot w_{kj}(n). \quad (2.22)$$

Окончательная формула обратного распространения для локального градиента $\partial_j(n)$ скрытого нейрона j:

$$\partial_j(n) = f_j'(v_j(n)) \cdot \sum_k \partial_k(n) \cdot w_{kj}(n). \quad (2.23)$$

Если нейрон выходной, то локальный градиент $\partial_j(n)$ равен произведению производной активационной функции $f_j'(v_j(n))$ на сигнал ошибки $e_j(n)$ для нейрона j. Если же нейрон находится в скрытом слое, то градиент рассчитывается как сумма градиентов нейронов следующего скрытого или выходного слоя, непосредственно связанных с данным нейронов.

Для сверточных нейронных сетей главное отличие заключается в том, что каждая связь нейрона объединяет не два фиксированных узла, а представляет из себя суммарное связывающее состояние для нескольких нейронов предыдущего слоя [137], в зависимости от положения плавающего окна (рис. 2.6).

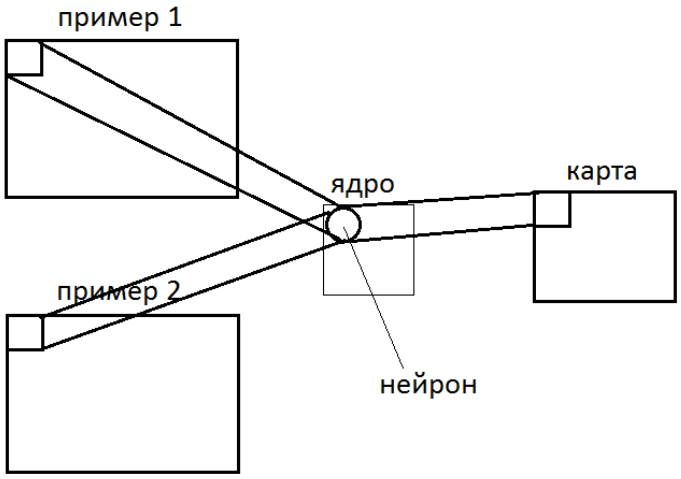


Рисунок 2.6 – Схема соединения карт

Поэтому при вычислении $\partial_j(n)$ необходимо накопить и посчитать суммарный градиент для всех состояний плавающего окна.

$$\nabla E^p = \nabla\left(\frac{\partial E}{\partial w_1};\frac{\partial E}{\partial w_2};\ldots;\frac{\partial E}{\partial w_n}\right), \tag{2.24}$$

где P – общее количество настраиваемых параметров

$$\partial_s(n) = \eta \cdot \sum y_i \cdot \partial_j = \eta \cdot \left(y_1\partial_1 + y_2\partial_2 + \ldots + y_m\partial_m\right). \tag{2.25}$$

Сверточные нейронные сети высоких порядков (Higher-order Neural Networks, HONNs) – расширение над сетями прямого распространения (рис.

2.7), в которых используются полиномиальные сумматоры нейронов, которые позволяют сети более качественно извлекать информацию из входного сигнала [70]. При этом вместо обычного взвешенного суммирования используются функции высоких порядков – квадратные, кубические, тригонометрические и т.д. Главным недостатком такого подхода является необходимость вводить дополнительные массивы весовых коэффициентов.

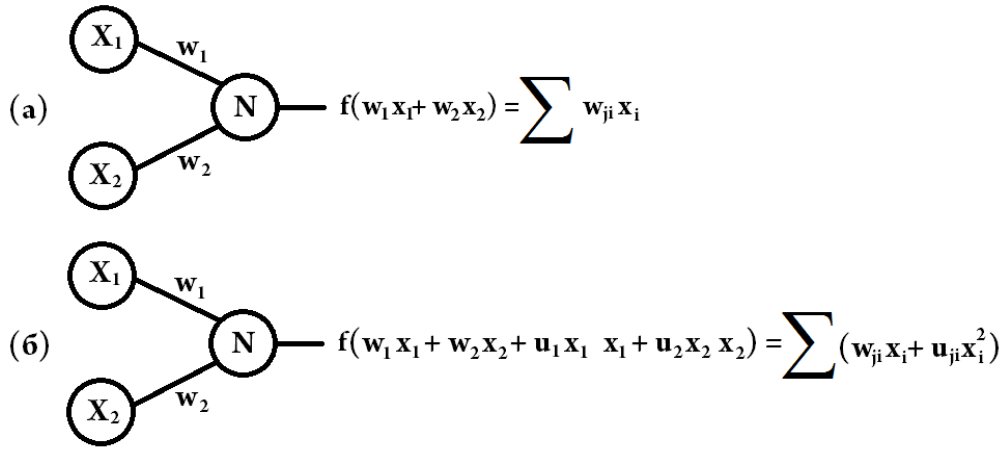


Рисунок 2.7 – (а) стандартный нейрон (б) нейрон с полиномиальным сумматоров второго порядка

Такой подход не затрагивает активационные функции нейронов и принципы вычисления ошибки, поэтому единственным отличием от стандартного алгоритма обратного распространения будет вычисление взвешенной суммы. Взвешенная сумма j-го нейрона второго порядка вычисляется следующим образом:

$$v_j(n) = \sum_{j=0}^{m} \left(w_{ji}(n) y_i(n) + u_{ji}(n) y_i^2(n) \right). \qquad (2.26)$$

Для третьего порядка:

$$v_j(n) = \sum_{j=0}^{m} \left(w_{ji}(n) y_i(n) + u_{ji}(n) y_i^2(n) + \mu_{ji}(n) y_i^3(n) \right). \qquad (2.27)$$

Тогда, для вычисления того, насколько нужно изменить весовые коэффициенты w и u, вычислим две частных производных:

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n) + u_{ji}(n) y_i^2(n) \right)' \mid w_{ji}(n) = \sum_{i=0}^{m} y_i(n), \qquad (2.28)$$

$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n) + u_{ji}(n) y_i^2(n) \right)' \mid u_{ji}(n) = \sum_{i=0}^{m} y_i^2(n). \qquad (2.29)$$

Таким образом, для коррекции весов нейронной сети второго порядка, если полиномиальные сумматоры находятся на выходном слое:

$$\Delta w_{ji}(n) = \eta \cdot e_j(n) \cdot f_j'\left(v_j(n)\right) \cdot y_i(n) = \eta \cdot \delta_j(n) \cdot y_i(n), \qquad (2.30)$$

$$\Delta u_{ji}(n) = \eta \cdot e_j(n) \cdot f_j'\left(v_j(n)\right) \cdot y_i^2(n) = \eta \cdot \delta_j(n) \cdot y_i^2(n). \qquad (2.31)$$

Если же сумматоры высоких порядков находятся на внутренних слоях сверточной нейронной сети:

$$\Delta w_{ji}(n) = f_j'\left(v_j(n)\right) \cdot \sum_k \partial_k(n) \cdot w_{kj}(n) \cdot y_i(n), \qquad (2.32)$$

$$\Delta u_{ji}(n) = f_j'\left(v_j(n)\right) \cdot \sum_k \partial_k(n) \cdot w_{kj}(n) \cdot y_i^2(n). \qquad (2.33)$$

2.2.2 Разработка архитектуры сверточной нейронной сети второго порядка

Следующим этапом после вывода формул обратного распространения для нейронов второго порядка является экспериментальное исследование того, на каких слоях СНС эти нейроны дают наибольшее увеличение обобщающей способности сети [57]. Нейроны высокого порядка могут быть использованы на любом слое нейронной сети. В зависимости от расположения, их использование будет различным образом влиять на результат работы сети. Для того, чтобы найти оптимальную комбинацию слоев, было создано несколько вариантов нейронных сетей на основе стандартной сверточной нейронной сети первого порядка. Каждый вариант сети был подвержен тестированию на качество обучения и распознавания на одной и той же выборке.

Для проведения экспериментов была взята выборка, содержащая 5 классов объектов и 7000 обучающих примеров, и разбита на обучающее и тестовое множества в соотношении 6/7 (6000 примеров) к 1/7 (1000 примеров). Предварительно выборка была перемешана.

Итоговый формат файлов выборки – два файла типов «dat» и «cat» (рис. 2.8) [84].

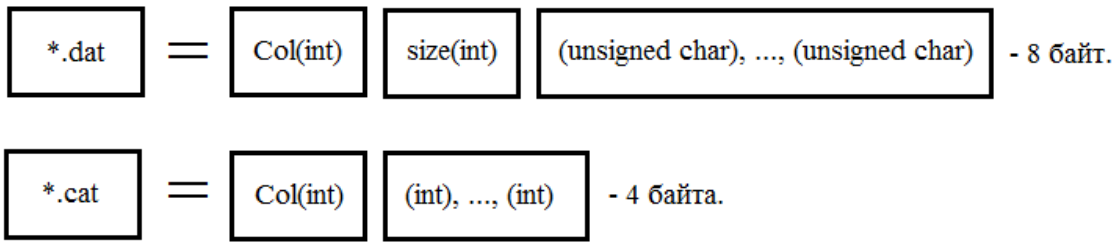


Рисунок 2.8 – Формат файлов выборки

Изначальная конфигурация выбранной для эксперимента сверточной нейронной сети включала три сверточных слоя, два субдискретизирующих слоя и полносвязный слой на выходе. Другими словами, сеть включала в себя следующие слои: входной слой, первый сверточный слой, первый субдискретизирующий слой, второй сверточный слой, второй субдискретизирующий слой, третий сверточный слой, полносвязный слой, выходной слой. В эксперименте не использовалось попарное произведение сигналов из-за высоких вычислительных затрат, которые несет такой подход [116].

Важно понимать, что результат, который дает использование полиномиальных сумматоров, может сильно отличаться в зависимости от многих условий, и зависит от конкретной задачи распознавания, конкретной архитектуры сети и того, на каком слое выбранной архитектуры эти сумматоры располагаются [61]. Из этого следует логичный вывод, что следует выбрать слои-кандидаты для использования нейронов высоких порядков, написать несколько вариантов архитектуры сети и протестировать подготовленную обучающую выборку на каждом из них.

Такая незначительная с первого взгляда вещь, как использование нейронов высоких порядков на некоторых слоях сети, затрагивает весь алгоритм ее функционирования: начиная от инициализации массивов переменных, заканчивая обновлением весов и тестированием качества обучения.

Таблица 2.1 – Результаты экспериментов по выбору оптимальной артитектуры СНС

№	I	C1	S1	C2	S2	C3	F	O	TM	BM
1									98	64
2							2		98	63
3		2							99	70
4				2					98	69
5						2			98	65
6		2		2					98	74
7				2		2			99	65
8		2				2			97	65
9		2		2		2			98	74

Результаты экспериментов показали (таб. 2.1), что данные о работе сети на тестовом множестве не могут быть использованы для адекватного сравнения, т.к. для всех вариантов архитектур показатель корректных распознаваний был близок к 100%. Объясняется это тем, что тестовая выборка была получена из того же набора данных, что и обучающее множество, и включала в себя объекты с тем же набором ракурсов, с теми же освещением и пропорциями, что и в примерах, на которых сеть обучалась. Для современных сверточных нейронных сетей такая задача распознавания является тривиальной и решается довольно просто [130]. Поэтому необходимо было усложнить задачу, добавив разнообразие в тестовое множество за счет реальных снимков объектов и различных преобразований обучающих примеров. Таким образом, акцент ставился в первую очередь на способность сети к обобщению обучающей информации.

Как было сказано ранее, результаты могут сильно отличаться для различных задач распознавания, различных архитектур и даже конкретных

обучающих множеств. В том случае, если конкретный слой со стандартными линейными сумматорами мог проводить нужные разделяющие поверхности и выделять необходимые признаки, то введение полиномиальных сумматоров не будет оказывать положительного действия на качество распознавания. Если же потенциал слоя со стандартной архитектурой был раскрыт недостаточно, и разделяющие поверхности не были построены с достаточной точностью, то нейроны высших порядков могут существенно улучшить ситуацию. Поэтому в первую очередь был протестирован вариант добавления нейронов высоких порядков на полносвязный выходной слой. Такая модификация не показала значительного увеличения точности распознавания, хотя для многослойного персептрона добавление полиномиальных нейронов приводит к существенному улучшению работы сети [60].

Проанализировав логику работы сверточных нейронных сетей, можно сделать вывод, что установка нейронов высоких порядков на S-слои не имеет смысла и может привести только к излишнему усложнению вычислений. Причиной является то, что S-слои выполняют очень простую функцию понижения размерности карт признаков, результат действия которой не может быть кардинально улучшен [75].

Таким образом, возможными кандидатами для введения нейронов высоких порядков, остаются только сверточные слои. При условии использования полиномиальных сумматоров второго порядка и наличия трех сверточных слоев в архитектуре сети получаем $2^3 = 8$ возможных комбинаций.

Таблица 2.1 показывает, что наибольшее влияние на качество распознавания образов произвело внедрение нейронов высших порядков на первый сверточный слой, и чем дальше находится слой с нейронами второго порядка от входа сети, тем это влияние становится меньше.

Внедрение двух сумматоров увеличивает позитивное действие нейронов высших порядков, но положительный эффект также зависит от

положения сумматоров в архитектуре сети. Так, использование сумматоров на первом и втором слоях дает существенно большие значения выходных показателей обучения, чем позиционирование на втором и третьем сверточных слоях сети. Добавление третьего полиномиального слоя не приводит к каким-либо позитивным изменениям и является излишним усложнением архитектуры сети и вычислений. Следовательно, комбинацией сверточных слоев с нейронами второго порядка для использования в разрабатываемом методе является шестой вариант архитектуры СНС из таблицы 2.1.

2.3 Разработка численного метода отсеивания гипотез по низкочастотной структуре

Численный метод отсеивания гипотез заключается в том, чтобы сравнить каждую гипотезу из сгенерированного набора с эталонным изображением, на котором присутствует искомый объект, и по некоторым параметрам сделать вывод, что гипотеза не относится ни к одному из искомых классов.

В направлении компьютерного зрения существует два подхода к распознаванию образов: классификация, т.е. соотнесение предъявленного образца к одному из известных классов, и верификация, т.е. ответ на вопрос, принадлежит ли образец конкретному классу [41]. Разработанный метод решает задачу, противоположную верификации, то есть выделяет те гипотезы, которые на основании каких-то признаков не относятся ни к одному из известных системе классов.

В качестве параметров для отсеивания гипотез в разработанном численном методе используются бинаризованный нормализованный градиент и процентное соотношение целевых цветов на изображении.

В качестве первого признака для отсеивания окон-гипотез используется 64-х битный вектор, аналогичный бинаризованному

нормализованному градиенту, описанному в [82]. Он необходим для того, чтобы убрать из изображения высокочастотную информацию, оставив только низкочастотную структуру. Алгоритм получения вектора состоит из следующих шагов [83]:

1) Уменьшение размера до 8x8 точек независимо от первоначального размера или соотношения сторон изображения.

2) Перевод изображения в оттенки серого.

3) Вычисление среднего значения для всех 64-х компонентов вектора.

4) Преобразование всех значения в биты, т.е. осуществление операции бинаризации. Если значение компонента вектора больше среднего значения, он устанавливается в 1, в обратном случае 0.

5) Перевод отдельных битов в одно 64-х битное значение.

6) Сравнение степени различия двух векторов, используя расстояние Хэмминга [5].

На рисунке 2.9 приведен пример того, как выглядит бинаризованный нормализованный градиент, а также несколько примеров гипотез с вычисленными параметрами схожести. В данном случае, если принять порог схожести равным тридцати пяти, отсеются два первых изображения, а три останутся в качестве кандидатов для распознавания.

Рисунок 2.9 – Образец, уменьшенное изображение, бинаризованный градиент и пять примеров гипотез с вычисленной степенью различия

Важным отличием разработанного метода от существующих является использование двух бинаризованных нормализованных градиентов. Первый строится исходя из среднего значения интенсивностей пикселей изображения, для построения второго каждое значение интерполированной гипотезы сравнивается с предыдущим: если предыдущее значение больше либо равно, то соответствующий бит приравнивается к нулю, если в конкретном значении интенсивности происходит переход от меньшего к большему значению, то в соответствующий бит бинаризованного градиента записывается единица. Таким образом, при использовании двух градиентов за счет синергии достигается большая инвариантность в отсеивании гипотез.

Алгоритм построения второго вектора признаков выглядит следующим образом:

1) Уменьшение размера до 8x8 точек независимо от первоначального размера или соотношения сторон изображения.

2) Перевод изображения в оттенки серого.

3) Преобразование всех значения в биты, т.е. осуществление операции бинаризации. Если значение компонента вектора больше предыдущего значения, соответствующий бит устанавливается в 1, в обратном случае 0.

4) Перевод отдельных битов в одно 64-х битное значение.

5) Сравнение степени различия двух векторов, используя расстояние Хэмминга.

Вторым признаком для отсеивания изображений-гипотез является отбор по цветовому профилю.

Классические гистограммы и метод когерентных цветовых векторов не подходят для решения поставленной задачи, потому что направлены в первую очередь на верификацию идентичных изображений в визуальных базах данных, но в условиях масштабирования, поворотов и изменения фона работают плохо.

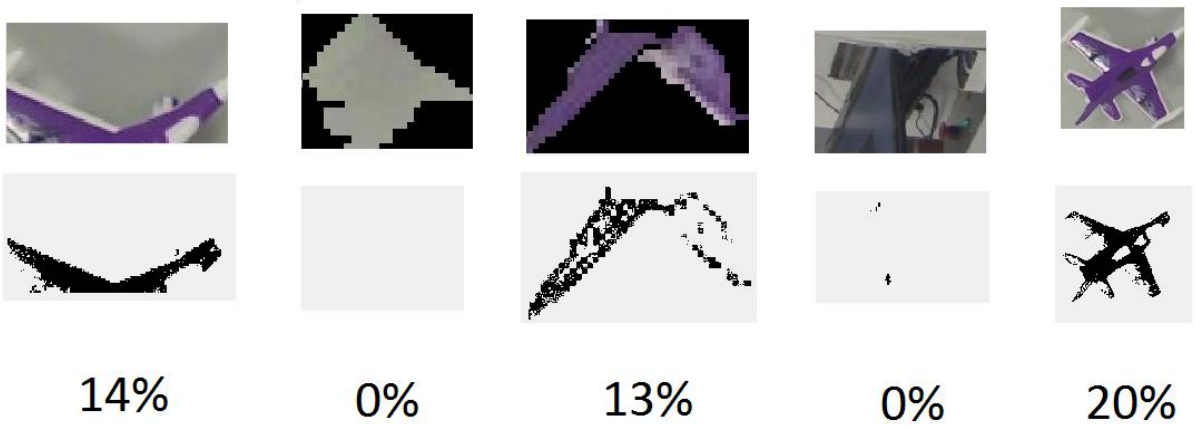


Рисунок 2.10 – Пример отсеивания по цвету с указанием процента пикселей, удовлетворяющих условию, от общего размера изображения

Поэтому, можно осуществлять отсеивание по одному из доминирующих цветов объекта [135]. При таком подходе, сначала выбирается цвет для проверки, который задается тремя компонентами в зависимости от используемого цветового пространства. Далее для каждой точки гипотезы вычисляется насколько далеко она лежит в цветовом пространстве от искомого цвета, на основании чего для каждого изображения строится маска.

$$d_i = (r_i - r) + (g_i - g) + (b_i - b) \tag{2.34}$$

Формула используется в классическом подходе к выделению объектов по цвету на изображении. Решение об отсеивании принимается в зависимости от процента, который составляют пиксели, близкие к искомому цвету, от общего количества пикселей изображения (рис. 2.10).

Разработанный метод отсеивания гипотез [37]:

1. Формирование эталонных градиентов:

перевод эталонного изображения в оттенки серого $E_k = (R_k + G_k + B_k)/3$, где E_k – точка изображения эталонного градиента, R,G,B – отдельные компоненты цвета.

изменение размера эталонного изображения (интерполяция) $E(e_1,e_2,...,e_n) \to E'(e_1,e_2,...,e_{64})$, где E – исходное изображение, E' – сжатое изображение с новым размером.

вычисление среднего значения интенсивности $s_e = \sum_{n=1..64} E_n / 64$,

$B_e^1\{b'_1,b'_2,...,b'_{64}\}$, если $\mathrm{E_i} < s_e \to b'_i = 0$, если $\mathrm{E_i} \geq s_e \to b'_i = 1$,

$B_e^2\{b'_1,b'_2,...,b'_{64}\}$, если $\mathrm{E_i} < E_{i+1} \to b'_i = 0$, если $\mathrm{E_i} \geq E_{i+1} \to b'_i = 1$, где B_e^1 – первый эталонный градиент, B_e^2 – второй эталонный градиент.

2. Выбор гипотезы из набора $G(G_1,G_2,...,G_N)$.

3. Формирование градиентов гипотезы:

перевод изображения в оттенки серого $I_k = (R_k + G_k + B_k)/3$, где I_k – точка изображения,

изменение размера изображения, интерполяция $I(i_1,i_2,...,i_n) \to I'(i_1,i_2,...,i_{64})$, где I – исходное изображение, I' – сжатое изображение с новым размером.

вычисление среднего значения интенсивности $s = \sum_{n=1..64} I_n / 64$,

$B_i^1\{b_1,b_2,...,b_{64}\}$, если $\mathrm{I_i} < s \to b_i = 0$, если $\mathrm{I_i} \geq s \to b_i = 1$,

$B_i^2\{b_1,b_2,...,b_{64}\}$, если $\mathrm{I_i} < I_{i+1} \to b_i = 0$, если $\mathrm{I_i} \geq I_{i+1} \to b_i = 1$, где B_i^1 – первый градиент гипотезы, B_i^2 – второй градиент гипотезы.

4. Вычисление разности $\Delta = (\Delta_1 \cdot \varphi_1 + \Delta_2 \cdot \varphi_2)/2$, где в общем случае $\varphi_1 = 1$, $\varphi_2 = 1$, Δ_1 и Δ_2 – параметры, характеризующие различие между градиентами гипотезы и эталонного объекта первого и второго типа.

$\Delta_1 = \sum_{k=1}^{64} r_k$ из $R_1\{r_1,r_2,...,r_{64}\}$, где если $\mathrm{b_k} = b'_k \to r_k = 1$, если $\mathrm{b_k} \neq b'_k \to r_k = 0$, $k = \overline{1,64}$,

$\Delta_2 = \sum_{k=1}^{64} r_k$ из $R_2\{r_1,r_2,...,r_{64}\}$, если $\mathrm{b_k} = b'_k \to r_k = 1$, если $\mathrm{b_k} \neq b'_k \to r_k = 0$, $k = \overline{1,64}$.

5. Вычисление $d_k = (r_i - r) + (g_i - g) + (b_i - b)$ – расстояния до искомого цвета $C = \{r,g,b\}$.

Вычисление цветовой разности $\Delta_c = \frac{n_p}{s_i} \cdot 100$, где n_p – количество пикселей гипотезы, расстояние до искомого цвета которых меньше некоторой константы ($d_k < d_{const}$), $s_i = w_i \cdot h_i$ – площадь, занимаемая гипотезой.

6. Сравнение Δ и Δ_c с пороговыми значениями P_1 и P_2. Если верно хотя бы одно из условий: $\Delta < P_1$ и $\Delta_c > P_2$, то гипотеза отсеивается. Если $\Delta \geq P_1$ и $\Delta_c \leq P_2$, то гипотеза остается в наборе.

7. Выбор новой гипотезы из набора, возвращение в пункт 2. Если достигнут предельный номер гипотез, алгоритм останавливается.

Таким образом, каждая из сгенерированных во время выполнения селективного поиска гипотез, обрабатывается алгоритмом, в результате чего принимается решение о достаточном соответствии гипотезы эталонному объекту, и часть гипотез отсеивается до распознавания на основании низкоуровневой структуры и цветового содержания.

2.4 Разработка метода выделения и распознавания объектов

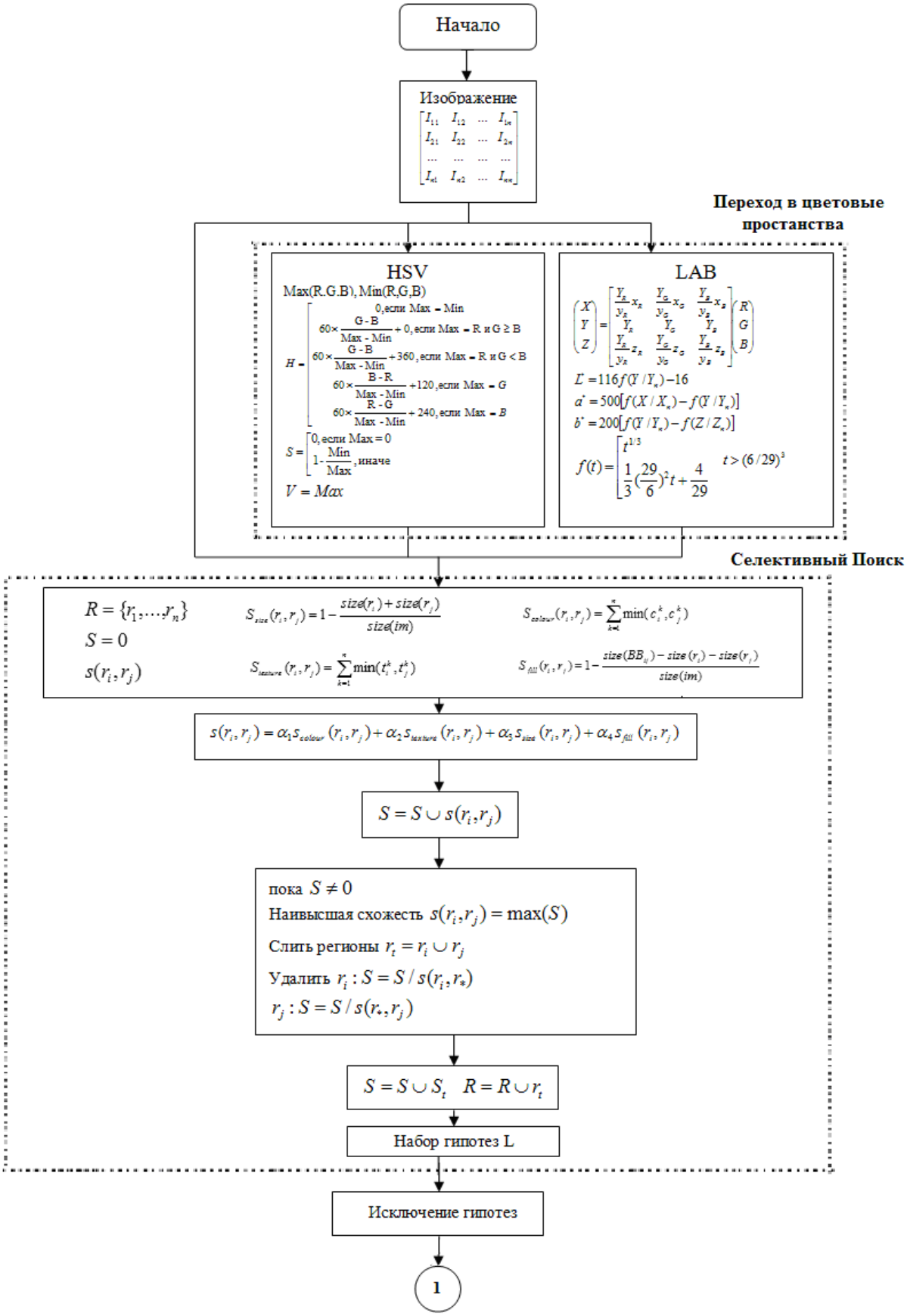
На рисунке 2.11 представлен алгоритм разработанного метода выделения и распознавания объектов на изображении, включающего подмодуль селективного поиска объектов, численный метод отсеивания гипотез и сверточную нейронную сеть второго порядка выбранной архитектуры.

Согласно методу, представленное в виде матрицы предобработанное изображение подвергается операции уменьшения размерности через выделение фона на основании предыдущей информации о рассматриваемой сцене, содержащей объект. Затем изображение переводится из цветового пространства RGB в пространства HSV и LAB с применением стандартных формул [93], и для каждого региона результирующих изображений вычисляются четыре базовых меры схожести и одна результирующая мера схожести. После этого выбираются и объединяются регионы с наибольшей

схожестью. Этапы повторяются итеративно, формируя набор гипотез о местоположении объекта.

Каждая гипотеза уменьшается в размере до матрицы 8x8 значений, переводится в оттенки серого, на основании результата вычисляется бинаризованный нормализованный градиент и производится отсеивание гипотез. Параллельно вычисляется цветовое содержание регионов гипотез и близость его к основным цветам искомых объектов, на основании которого производится второй этап отсеивания.

Все оставшиеся гипотезы подвергаются масштабированию до 96x96 пикселей и преобразуются в вектор из 9216 элементов. Вектор подается на вход сверточной нейронной сети и проходит последовательно два сверточных слоя второго порядка, два субдискретизирующих слоя, один стандартный сверточный слой и полносвязный слой на выходе сети. В результате гипотеза соотносится с одним из классов объектов, либо не соответствует ни одному из классов.


Начало
Изображение
Переход в цветовые простанства
HSV
Max(R.G.B), Min(R,G,B)
LAB
Селективный Поиск
Набор гипотез L
Исключение гипотез
1

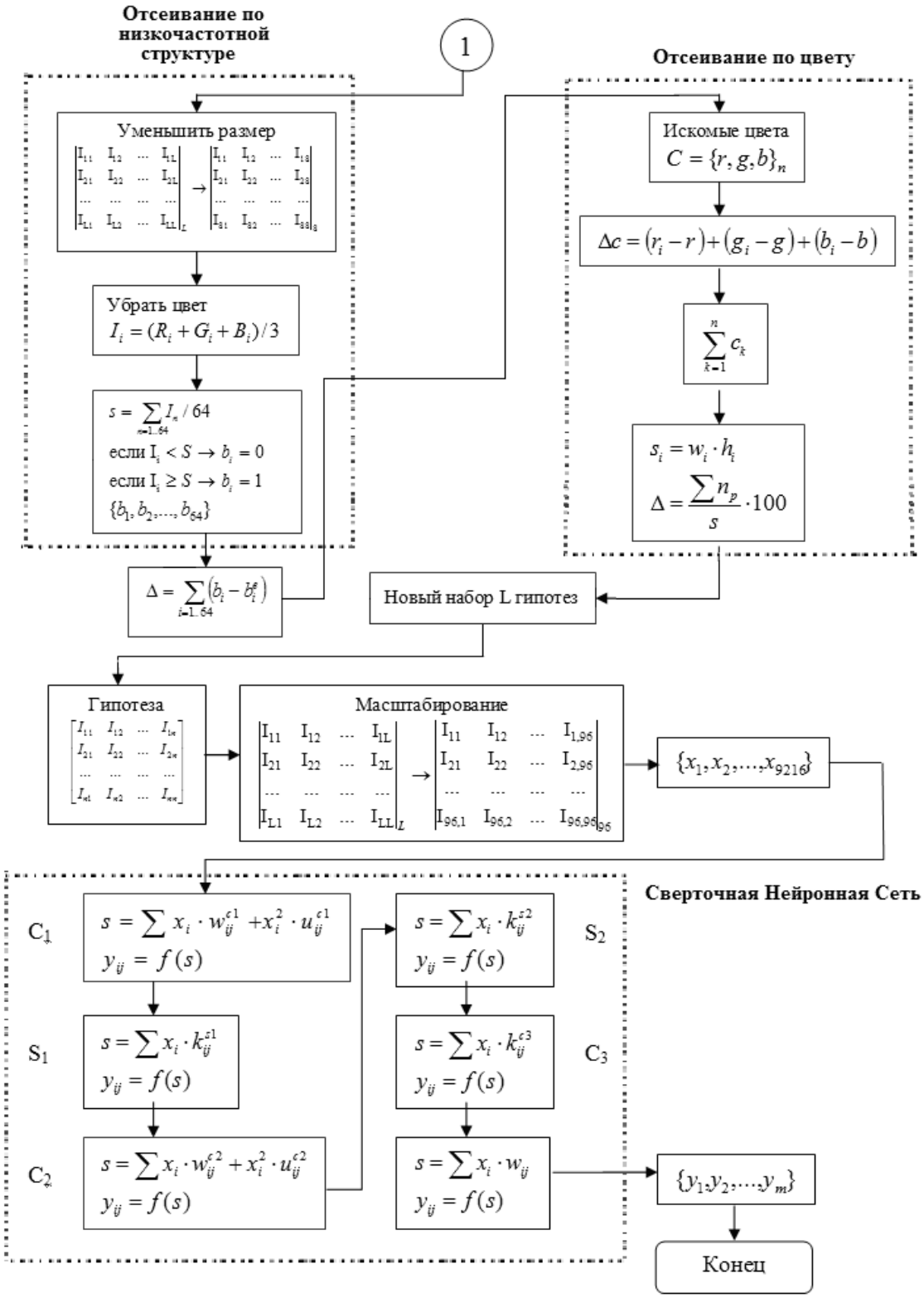


Рисунок 2.11 – Алгоритм, реализующий модель выделения и распознавания объектов на изображениях

Описанная в первой главе модель, при добавлении нейронов второго порядка и численного метода отсеивания гипотез, принимает вид:

$$S_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k), \tag{2.35}$$

$$S_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k), \tag{2.36}$$

$$S_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}, \tag{2.37}$$

$$fill(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}, \tag{2.38}$$

$$s(r_i, r_j) = \alpha_1 s_{colour}(r_i, r_j) + \alpha_2 s_{texture}(r_i, r_j) + \alpha_3 s_{size}(r_i, r_j) + \alpha_4 s_{fill}(r_i, r_j), \tag{2.39}$$

$$r_t = r_i \cup r_j, \text{где s} = \max, \tag{2.40}$$

$$R(r_1, r_2, ..., r_m), \text{где m} < \text{n}, \tag{2.41}$$

$$r_k(i_1, i_2, ..., i_n) \to r_k'(i_1, i_2, ..., i_{64}), \text{где } k = \overline{0, m}, \tag{2.42}$$

$$i_s = \sum_{n=1..64} i_n / 64, \tag{2.43}$$

$$B_e^1\{b'_1, b'_2, ..., b'_{64}\}, \text{ если i}_\text{i} < i_s \to b'_i = 0, \text{ если i}_\text{i} \geq i_s \to b'_i = 1, \tag{2.44}$$

$$B_e^2\{b'_1, b'_2, ..., b'_{64}\}, \text{ если i}_\text{i} < i_{i+1} \to b'_i = 0, \text{ если i}_\text{i} \geq i_{i+1} \to b'_i = 1, \tag{2.45}$$

$$\Delta_k = \frac{1}{2} \cdot \left(\sum_{p=1}^{64} \left| b_p - b'_p \right| + \sum_{p=1}^{64} \left| b_p - b'_p \right| \right), \tag{2.46}$$

$$d_k = (r_k - r) + (g_k - g) + (b_k - b), \; k = \overline{1, n}, \tag{2.47}$$

$$\Delta_c = \frac{n_p}{w_i \cdot h_i} \cdot 100, \tag{2.48}$$

$$R'(r_1, r_2, ..., r_{m'}), \text{где m}' \leq \text{m}, \tag{2.49}$$

$$X(x_1, x_2, ..., x_g), \tag{2.50}$$

$$s_{c1} = \sum_i \sum_j (x_i \cdot w_{ij}^{c1} + x_i^2 \cdot u_{ij}^{c1}), i = \overline{1, n}, \; j = \overline{1, m}; \; h_{ij}^{c1} = \frac{1}{1 - e^{-s_{c1}}}, \tag{2.51}$$

$$s_{s1} = \sum_i \sum_j (h_i^{c1} \cdot k_{ij}^{s1}), i = \overline{1, n}, \; j = \overline{1, m}; \; h_{ij}^{s1} = \frac{1}{1 - e^{-s_{s1}}}, \tag{2.52}$$

$$s_{c2} = \sum_i \sum_j (h_i^{s1} \cdot w_{ij}^{c2} + h_i^{s1} \cdot h_i^{s1} \cdot u_{ij}^{c2}), i = \overline{1, n}, \; j = \overline{1, m}; \; h_{ij}^{c2} = \frac{1}{1 - e^{-s_{c2}}}, \tag{2.53}$$

$$s_{s2} = \sum_i \sum_j (h_i^{c2} \cdot k_{ij}^{s2}), i = \overline{1,n}, j = \overline{1,m};\ h_{ij}^{s2} = \frac{1}{1 - e^{-s_{s2}}}, \quad (2.54)$$

$$s_{c3} = \sum_i \sum_j (h_i^{s2} \cdot w_{ij}^{c3}), i = \overline{1,n}, j = \overline{1,m};\ h_{ij}^{c3} = \frac{1}{1 - e^{-s_{c3}}}, \quad (2.55)$$

$$s = \sum_i \sum_j (h_i^{c3} \cdot w_{ij}), i = \overline{1,n}, j = \overline{1,m};\ y_{ij} = \frac{1}{1 - e^{-s}}, \quad (2.56)$$

где $S_{colour}(r_i, r_j)$ – мера схожести по цвету, $S_{texture}(r_i, r_j)$ – мера схожести текстуры, $S_{size}(r_i, r_j)$ – оценка размера, $S_{fill}(r_i, r_j)$ – мера соответствия двух областей, $s(r_i, r_j)$ – комбинация четырех мер схожести, r_t – регион с наибольшей схожестью, i_s – среднее значение интенсивности, B_e^1 и B_e^2 – нормализованные градиенты, Δ_k – мера схожести по низкочастотной структуре, d_k – расстояние между цветами в цветовом пространстве, Δ_c – мера схожести по цвету, n_p – количество пикселей, для которых $d_k < d_{const}$, R' – финальный набор регионов, X – входной вектор НС, s_{c1} – взвешенная сумма, h_{ij} – выходы отдельного слоя, y_{ij} – выходы сети.

По сравнению с математической моделью из первой главы, данная модель содержит следующие изменения: добавлен этап отсеивания гипотез после селективного поиска (2.42 – 2.49), за счет чего массив входных векторов для обработки СНС имеет меньшую размерность, а также в архитектуру СНС добавлены слои с нейронами второго порядка (2.51, 2.53).

Разработанный метод выделения и распознавания объектов на изображениях базируется на модели R-CNN и использовании численного метода отсеивания гипотез по низкочастотной структуре и нейронов второго порядка в архитектуре СНС. Использование двух этапов в алгоритме выделения и распознавания объектов предположительно обеспечивает высокое быстродействие и качество распознавания при этом в основе лежит простая идея, не требующая полного переосмысления исходной модели и кардинального изменения ее структуры. Полученные результаты будут подробно показаны в пятой главе данного диссертационного исследования.

2.5 Экспериментальное исследование методов уменьшения размерности входных данных и выбор метода вычитания фона

В условиях, когда технические условия позволяют получить и обработать несколько последовательных кадров, разработанный метод выделения и распознавания объектов может быть расширен за счет предварительного выделения фона от переднего плана.

Уменьшение количества генерируемых в процессе работы алгоритма гипотез может быть достигнуто не только за счет отсеивания, но и за счет понижения размерности обрабатываемых данных через фильтрацию исходного изображения на основе его изменений во времени, а также исключение гипотез, для которых можно сделать вывод, что они не относятся ни к одному из целевых объектов, на основании размера.

Вычитание фона – широко распространенный подход к выделению движущихся объектов в последовательности видеокадров, снятых со статической камеры путем нахождения разности между текущим кадром и так называемым опорным кадром, или моделью фона. К полученной разности двух изображений применяется пороговая фильтрация, с помощью которой удаляется не имеющий особой важности шум, и локализуется искомый объект [120]. Модель фона должна хорошо отражать ту сцену, в которой происходит поиск и выделение, поэтому, следует придерживаться следующих рекомендаций:

1) Исключить по возможности появление движущихся объектов в пределах видимости камеры.

2) Постоянно обновлять модель фона, чтобы адаптировать ее к изменениям в геометрии и освещении сцены [151].

На сегодняшний день существует большое количество методов вычитания фона: от простейших подходов, целью которых является максимальная скорость при минимальных требованиях к вычислительной мощности и ресурсам, до более сложных методов, направленных на

достижение максимальной точности выделения. Все методы разработаны таким образом, чтобы работать в реальном времени [90].

Временной медианный фильтр

В данном подходе в качестве фона используется среднее значение последних n кадров. При этом результирующее среднее значение вычисляется на наборе последних n кадров и среднего значения вычисленного на последней итерации. Таким образом, достигается большая стабильность модели фона.

$$\Delta I = \sum_{n} I_k - I_t \tag{2.57}$$

Главным недостатком такого подхода является необходимость хранить большие пиксельные массивы предыдущих кадров. Также метод вычисления средних значений не использует строгие статистические данные, что не дает возможности адаптировать пороговый фильтр для выделения объектов.

Гауссово среднее

Подход предполагает вычисление фона независимо для каждого пикселя с координатами (i,j). Для каждого нового кадра, появившегося в момент времени t, Гауссово среднее вычисляется по формуле:

$$\varphi_t = \alpha I_t + (1-\alpha)\varphi_{t-1}. \tag{2.58}$$

где I_t – интенсивность текущего кадра в точке в момент времени t, φ_{t-1} – предыдущее значение вычисляемого параметра, α – коэффициент, подбирающийся под конкретную задачу.

Такой подход требует гораздо меньшее количество используемой памяти в сравнении с хранением массива последних n кадров, положительным моментом является также ускорение процесса выделения фона. Классификация происходит с использованием двух параметров (φ_t, σ_t),

где σ_t – стандартное отклонение. Если принять p_{ij}^t за значение пикселя (i, j) в момент времени t, то p_{ij}^t будет классифицирован как пиксель фона, если $\left|p_{ij}^t - \gamma_{t-1}\right| > k\sigma_t$.

Использование ядер свертки

При таком подходе, для каждого пикселя на решение о принадлежности к фону влияет не только значение интенсивности самого пикселя, но и соседние точки в некотором диапазоне. Другими словами для каждой точки изображения рассчитывается в пределах фиксированного окна свертка с ядром, а результат этой операции непосредственно влияет на разделение.

Существует разновидность алгоритма, когда заранее создается массив наиболее часто встречающихся гистограмм фона для конкретной ситуации, и сравнение свертки осуществляется поочередно с каждым значением из этого массива.

$$P(x_t) = \frac{1}{n}\sum_{i=1}^{n}\eta(x_t - x_i, \sum_t), \qquad (2.59)$$

где η – функция оценки, а $\sum_t$ – отображает размер ядра функции оценки.

Деление изображения на квадратные блоки

В основе подхода лежит предположение, что соседние блоки пикселей, принадлежащих фону, должны иметь схожие изменения с течением времени. Недостатком метода является то, что он не учитывает блоки пикселей на границах отдельных объектов фона.

Каждый кадр разделяется на отдельные блоки NxN пикселей, каждый блок рассматривается как вектор из N^2 элементов. Вычисляется среднее для каждого сэмпла, а также разница между сэмплом и средним, которая

характеризует изменение изображения. Затем, формируется ковариационная матрица $N^2 x N^2$. Собственные вектора трансформируются с целью уменьшить размерность вектором изменения изображений.

Для каждого блока рассчитывается текущий собственный вектор соседнего блока, затем изменение изображений L соседних векторов в пространстве собственных векторов подвергается линейной интерполяции, и полученное значение применяется к исходному блоку.

Собственные вектора фона

Данный подход основан на декомпозиции собственных векторов для всего изображения в целом [13]. В фазе обучения получают набор изображений, для которых вычисляется среднее, вычитаемое из каждого примера. Собственные вектора сохраняются в виде матрицы. Вычисляется проекция текущего кадра на пространство собственных векторов, которая проецируется назад на пространство изображения. В результате получается кадр фона, который не содержит небольших по размеру движущихся объектов. В завершении вычисляется разница между полученным и исходным кадрами.

На выбор алгоритма выделения фона влияют два параметра: время, затрачиваемое на выполнение алгоритма, и точность выделения. Время является абсолютным показателем, который легко измерить для конкретного типа системы. Точность выделения фона можно рассчитать по формуле:

$$Q = \frac{\sum N_p - \sum N_n}{N}, \qquad (2.60)$$

где $\sum N_p$ – количество правильно выделенных пикселей, $\sum N_n$ – количество ложно определенных точек, N – общее их количество.

В таблице 2.2 представлены результаты эксперимента для различных алгоритмов выделения фона.

Таблица 2.2 – Результаты точности работы для различных алгоритмов выделения фона

Метод	Цветовое пространство	Точность	Время
Медианный фильтр	RGB	0,71	0,03
	HSV	0,72	0,04
Гауссово среднее	RGB	0,71	0,02
	HSV	0,72	0,03
Ядра свертки	RGB	0,74	0,17
	HSV	0,74	0,18
Сегментирование	RGB	0,67	0,23
	HSV	0,67	0,24
Собственные вектора	RGB	0,41	0,25
	HSV	0,51	0,26

Следовательно, наибольшую точность выделения фона дал метод использования ядер и операции свертки, но важную роль играет также время, необходимое на осуществление вычислений. Поэтому в качестве метода выделения новых объектов от фона, целесообразно использовать первый или второй вариант. Также стоит отметить, что использование цветового пространства HSV во всех случаях дает прибавку по времени выполнения за счет необходимости конвертировать кадр из стандартного пространства RGB.

Еще одним расширением разработанного метода может служить отсеивание небольшого процента генерируемых гипотез по размеру и форме. В ситуациях, когда специфика рассматриваемой узкой задачи позволяет сделать определенные допущения, можно исключить из рассмотрения гипотезы слишком большие (больше 0.5 линейных размеров изображения), слишком малые (меньше 0.05 линейных размеров изображения), объекты с длинной тонкой структурой, которых алгоритм селективного поиска генерирует довольно много.

Этот пункт не является обязательным. Если условия задачи допускают сделать вышеописанные предположения – используется дополнительное отсеивание гипотез, если нет – никаких изменений не производится.

Селективный поиск на сегодняшний день является одним из самых совершенных и точных методов нахождения алгоритмов на изображении. Поэтому кардинально менять что-то в нем – трудная и неоправданная задача. Однако, все еще возможны небольшие изменения, которые существенным образом могут ускорить процесс нахождения объектов, учитывая специфику конкретной задачи и возможность пожертвовать точностью выделения в пользу скорости работы.

Возможность ускорения метода заключается в первую очередь в следующих аспектах (рис. 2.12):

1) Отбрасывание слишком больших и слишком малых гипотез, которые в любом случае не будут покрывать зоны, в которых может появиться объект.

2) Исключение объектов, имеющих тонкую, длинную структуру. Если мыслить обобщенно, то это могут быть провода или другие похожие элементы, но применительно к конкретной рассматриваемой задаче, в 100 % случаев такие гипотезы будут затрагивать лишь границы объектов и цифровой мусор, не представляющий интереса.

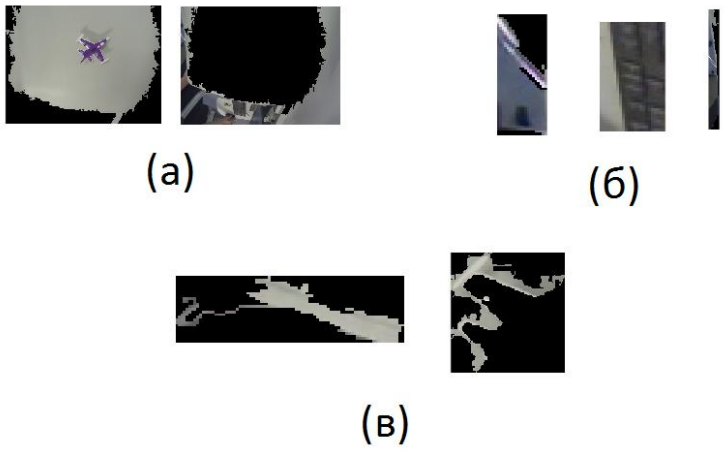


Рисунок 2.12 – Примеры исключаемых гипотез (а) слишком больших (б) слишком малых (в) длинных тонких структур

Описанные в данном параграфе дополнения имеют значение в перспективе для использования в разработанном методе выделения и

распознавания объектов, но в задачах с более обширным набором исходных данных (например, выделение объектов на видео), и могут существенно упростить адаптацию разработанного метода и ускорить вычисления.

Выводы

1. Разработан метод выделения и распознавания объектов на изображениях, базирующийся на модели R-CNN, использовании нейронов второго порядка в СНС и отсеивании гипотез, сгенерированных алгоритмом селективного поиска, по низкочастотной структуре. Показаны изменения в базовой модели R-CNN. Выполнена первая подзадача диссертационного исследования.

2. Разработан численный метод отсеивания гипотез по низкочастотной структуре, обеспечивающий значительное уменьшение числа генерируемых гипотез по сравнению со стандартным алгоритмом. В основе алгоритма лежат вычисление двух 64-битных векторов, описывающих низкочастотную структуру изображения, и анализ наличия искомого цвета целевого объекта.

3. Проведен анализ и выбрана оптимальная структура сверточной нейронной сети второго порядка для использования в разработанном методе. Наилучшие показатели обобщающей способности сети показала модификация архитектуры «Le-net5» с нейронами второго порядка на первом и втором сверточных слоях. Выведены формулы обратного распространения для сверточных нейронных сетей второго порядка.

4. Доказано, что внедрение нейронов высоких порядков, в частности второго порядка, положительно влияет на качество обучения, распознавания и работу сверточной нейронной сети.

5. Визуализированы и показаны схематично метод выделения и распознавания объектов на изображениях, отдельные блоки метода, принцип обработки данных в СНС второго порядка и алгоритм численного метода отсеивания гипотез по низкочастотной структуре. Осуществлен анализ различных методов выделения фона, дано обоснование выбора конкретных типов алгоритмов.

6. Описание экспериментальной апробации разработанного численного метода приведено в пятой главе.

ГЛАВА 3 РАЗРАБОТКА ПАРАЛЛЕЛЬНОГО АЛГОРИТМА ОБРАБОТКИ ДАННЫХ В СНС ВТОРОГО ПОРЯДКА, ОРИЕНТИРОВАННОГО НА ПРОЦЕССОРЫ С ВЕКТОРНО-МАТРИЧНОЙ АРХИТЕКТУРОЙ

Данная глава содержит описание разработанных параллельных алгоритмов обработки данных в СНС второго порядка, ориентированных на процессоры с векторно-матричной архитектурой, и анализ процессоров векторно-матричной архитектуры с точки зрения реализации нейронных сетей.

3.1 Анализ применимости процессоров векторно-матричной архитектуры для реализации нейронных сетей

В предыдущей главе был описан численный метод отсеивания гипотез по низкочастотной структуре, использование которого при селективном поиске объектов на изображении повышает скорость выделения и распознавания объектов за счет снижения количества векторов, которые необходимо обработать с помощью СНС. Для того, чтобы получить еще большее ускорение, целесообразно ускорить непосредственно саму обработку данных в СНС за счет параллельной обработки входного вектора. Для решения этой задачи могут быть использованы процессоры с векторно-матричной архитектурой за счет схожести с принципами нейросетевой обработки данных и возможности распараллеливания вычисления взвешенных сумм для нескольких нейронов с помощью матричного вычислительного устройства.

В процессорах векторной архитектуры в качестве операндов команд могут выступать массивы данных, называемые векторами. Векторные процессоры могут как дополнять традиционные скалярные вычислительные

системы, так и выступать в роли самостоятельной процессорной единицы [44].

Процессоры векторно-матричной архитектуры объединяют отдельные аспекты векторных и матричных систем. Одними из наиболее ярких представителей этого типа является линейка микропроцессоров NeuroMatrix, разработанных компанией НТЦ «Модуль».

Высокопроизводительные микропроцессоры NeuroMatrix основаны на элементах архитектур VLIW/SIMD и включают в свой состав узел для поддержки операций над векторами с элементами переменной разрядности, устройства вычисления адреса, управления и обработки скаляров. При обращении к внешней памяти процессор использует 32-разрядный вычисляемый адрес. Несмотря на то, что шина данных имеет разрядность 64 бита, обмен может происходить как по 32 разряда, так и по 64, в зависимости от типа команд и типа используемых в них констант.

Структура процессоров включает следующие блоки (рис. 3.1) [2]:

1) Векторный процессор.
2) Скалярный процессор.
3) Регистры управления доступом к внешней памяти.
4) Два таймера.
5) Два DMA-сопроцессора для управления работой коммуникационных портов.

Основной особенностью и причиной выбора процессоров архитектуры NeuroMatrix для решения задачи ускорения работы сверточных нейронных сетей является входящий в их состав векторный процессор, способный выполнять одновременно до 2048 параллельных операций за один такт [29]. Точность и производительность – два настраиваемых параметра, которые могут быть оптимизированы в соответствии с требуемыми для конкретной задачи условиями с помощью разбиения векторов на элементы от одного бита до 64-х.

За одно обращения к внешней памяти процессор способен прочитать или записать по каждой шине одно 64-х разрядное число, при этом осуществляя множественные преобразования данных в пределах одной инструкции (SIMD-обработка, single instruction – multiple data).

Скалярный процессор может использоваться как самостоятельный вычислительный блок и как RISC ядро, предназначенное для подготовки данных для векторного процессора. В состав скалярного процессора входят 8 регистров общего назначения (gr0 – gr7) и 8 специальных адресных регистров (ar0 – ar7) [4].

Список операций скалярного процессора включает в себя:

1) Управление конфигурацией векторного процессора.
2) Управление доступом к внешней памяти с помощью настройки специальных регистров.
3) Управление таймерами.
4) Пошаговое умножение.
5) Условные, безусловные и отложенные переходы.
6) Вызов функций с записью адреса возврата в стек.
7) Сдвиг на произвольное количество битов.
8) Разнообразные арифметические и логические операции.
9) Чтение и запись 32-х и 64-разрядных слов в память.
10) Различные виды адресации с изменением адресных регистров.

Центральная часть векторного процессора NeuroMatrix – операционное устройство – включает в себя рабочую и теневую матрицы и векторное АЛУ, и служит для выполнения арифметических, логических операций, перестановки битов в векторах данных, операции взвешенного суммирования через умножение с накоплением. Дополнительные возможности векторного процессора включают в себя операцию маскирования и аппаратно реализованную активационную функцию нейронов.

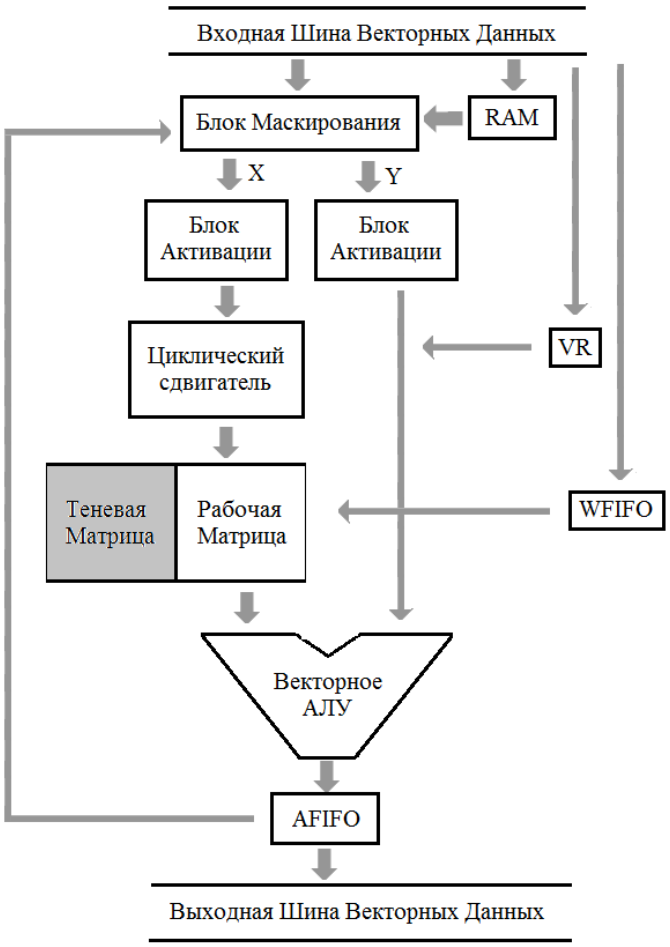


Рисунок 3.1 – Схема процессоров NeuroMatrix

В качестве кандидата для использования основных принципов разработанного параллельного алгоритма обработки сигнала в сверточной нейронной сети второго порядка может рассматриваться практически любая модель процессора с архитектурой SIMD.

К основным моделям можно отнести:

1. Линейка микропроцессоров NeuroMatrix.

1.1 Nm6403(1879BM1), изготовленный в 1998 на фабрике Samsung.

1.2 Nm6404(1879BM2), полностью совместим по системе команд с предыдущей моделью, создан в 2006 году.

1.3 Nm6405(1879BM4), использует ядро последнего поколения nmc3, разработан в 2010 году.

1.4 Система на кристалле СБИС ДЦТС (декодер цифрового телевизионного сигнала) также использует ядра nmc3, разработана в 2011 году [27].

1.5 Nm6406(1879ВМ5Я), разработанный в 2012 году, активно поддерживается компанией до сих пор. На репозитории GitHub регулярно появляются обновления программного обеспечения для данного процессора.

2. Процессоры обработки цифровых сигналов компании Texas Instruments, в частности DSP C6000 [126].

3. Семейство графических процессоров TeraScale, разработанное ATI Technologies/AMD и представленная в августе 2011 года [105].

Таблица 3.1 – Технологическая карта ядра NMC

Характеристики	NMC1	NMC2	NMC3	NMC4	
Технология КМОП, нм	500	250	90	65	40
Интеграция, экв. вентилей	90 000	250 000	250 000	4 000 000	
Напряжение питания, В	3,3	2,5	1,2	1,0	
Тактовая частота, МГц	40	150	320	500	750
Пиковая производительность, MMAC/s/(GFLOPS)	8 960	33 600	71 680	112 000	168 000
Пиковая производительность (8-битовые данные), MOPS/s	960	3 600	7 680	12 000	18 000
Пропускная способность интерфейса, MB/s	640	6 000	12 800	20 000	30 000
Потребляемая мощность, мВт	700	800	450	750	500

В настоящее время компания НТЦ «Модуль» активно развивает линейку векторно-матричных процессоров NeuroMatrix. Новый процессор 1879ВМ6Я (NM6407), представление которого состоится на одной из

весенних выставок 2016 года. Также ведутся активные работы по созданию еще как минимум двух процессоров этой тематики в 2016-2017 годах. Область применения: обработка широкополосных сигналов, в том числе различные виды цифровой фильтрации, преобразование Фурье, Адамара и прочие, обработка изображений, навигация, эмулирование различных архитектур нейронных сетей, высокопроизводительная коммутация сигналов, CDMA и TDMA базовые станции сотовой связи. Основные характеристики: тактовая частота – 500 МГц, Технология КМОП 65 нм, память на кристалле – 16 Мбит, потребляемая модность – 2,6 Вт.

В таблице 3.1 приведены основные характеристики различных ядер семейства NMC: NMC1 (NM6403), NMC2 (NM6405), NMC3 (NM6406), NMC4 (NM6407).

3.2 Общие принципы параллельной обработки данных в сверточных нейронных сетях

СНС второго порядка отличается от классической архитектуры СНС за счет использования полиномиальных сумматоров при обработке данных в отдельном нейроне, общие принципы обработки данных в нейронной сети при прямом и обратном проходе одинаковы [115]. Поэтому целесообразно сначала описать принципы параллельной обработки для классической СНС, а затем перейти к СНС второго порядка.

Можно составить следующую иерархию нейросетевых моделей по степени увеличения сложности алгоритмов параллельной обработки:

1. Классическая сеть прямого распространения или многослойный персептрон – представляет интерес как наиболее простой и в то же время базовый тип сетей. Любая другая архитектура в той или иной степени основана на классических идеях и принципах [98].

2. Сверточные нейронные сети. На сегодняшний день данный тип архитектуры нейронных сетей считается наиболее прогрессивным и лежит в

основе глубокого обучения, носящего высокий уровень абстракции и основывающегося на многочисленных нелинейных преобразованиях исходных данных [147].

3. Сверточные нейронные сети высоких порядков. Такой тип нейронных сетей является надстройкой над стандартными сетями прямого распространения, включающей нелинейные сумматоры в состав нейронов определенных слоев [71].

Известно, что слои в сверточной нейронной сети представляют собой группы нейронов, каждая из которых своими выходными значениями формирует определенную карту признаков, поэтому при инициализации сети, изначально выделяется блок памяти для выходных значений каждого из слоев и аналогичный блок, в котором будут накапливаться ошибки отдельных нейронов при обратном проходе. Во время прямого прохода последовательно заполняются все выходные векторы, а во время обратного прохода – векторы ошибок для каждого слоя нейронной сети.

Процессоры NeuroMatrix позволяют аппаратно эмулировать два типа активационных функций: пороговую и функцию насыщения (рис. 3.4) [65]. Сложная архитектура сети предполагает использование больших по модулю значений аргументов, поэтому использование пороговой функции активации не имеет смысла и превращает график изменения выходных значений в постоянную прямую. Поэтому логичным решением является использование функции насыщения с ограничением диапазона и таблицей возможных значений функции. Функция насыщения реализуется аппаратно:

rep 32 with activate afifo + 0;

Предполагается, что таблица содержит преобразованные к целому значения сигмоидальной функции для диапазона значений аргумента от -32768 до 32767, определяемых верхним и нижним порогом. Всего используется 65536 возможных значений аргумента.

$$f = 32767 \cdot \frac{1 - e^t}{1 + e^t}, t = -\frac{x}{2048} \qquad (3.1)$$

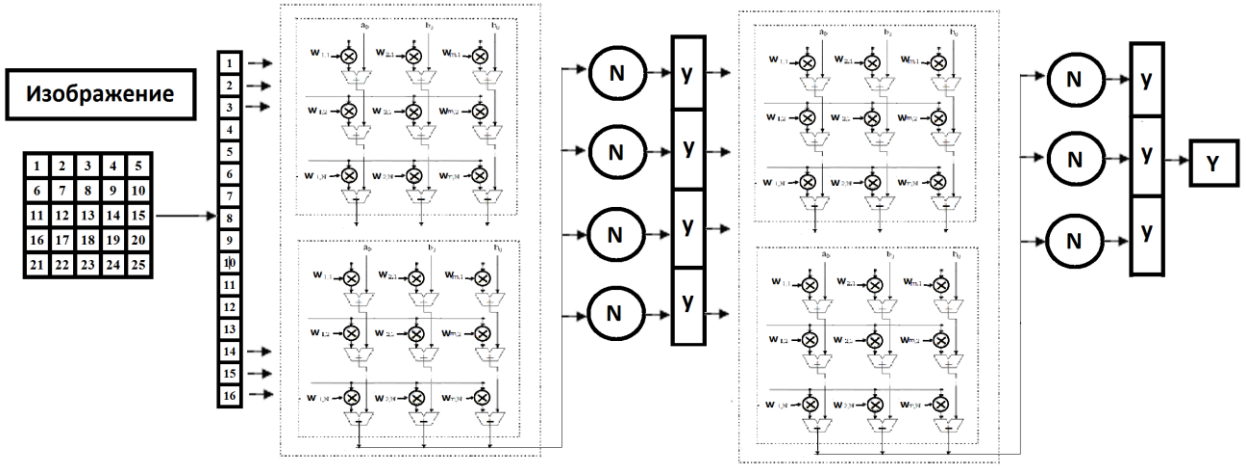


Рисунок 3.2 – Обобщенная схема реализации многослойной сети

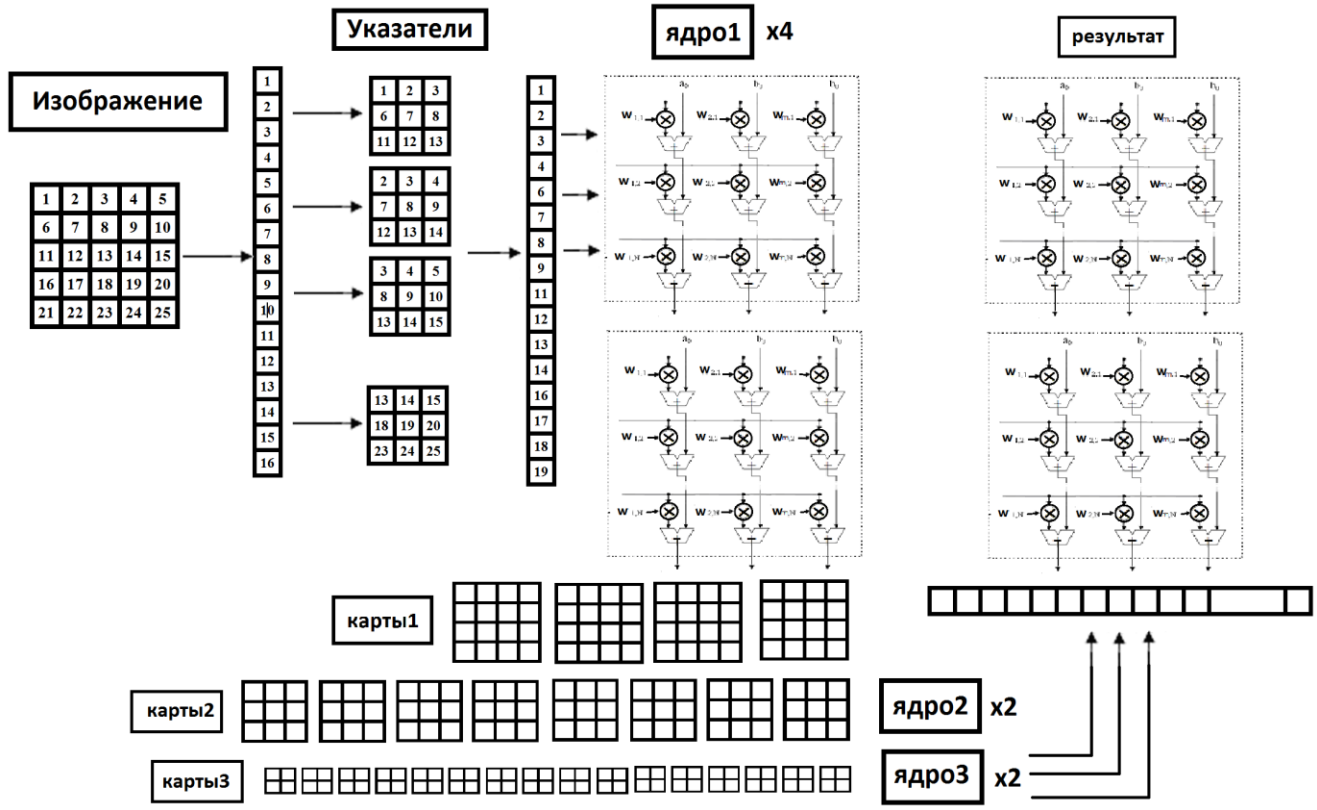
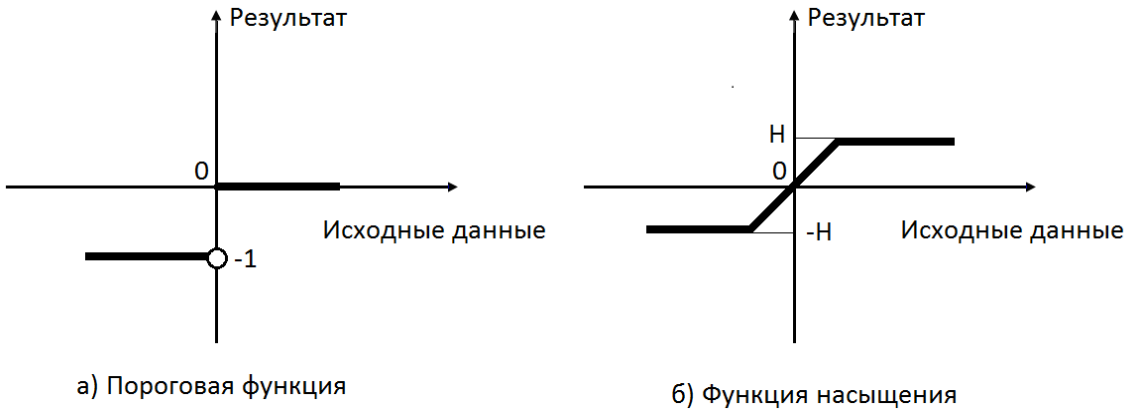


Рисунок 3.3 – Обобщенная схема реализации сверточной нейронной сети



а) Пороговая функция

б) Функция насыщения

Рисунок 3.4 – вид функции активации (а) пороговая функция (б) функция насыщения

Обобщенная схема реализации классической полносвязной нейронной сети представлена на рисунке 3.2 и является базой для перехода к более сложным моделям. В основе алгоритма лежат операции взвешенного суммирования, а также функция нелинейного преобразования, соответствующие основным элементам архитектуры искусственного нейрона [51]. Обе функции реализованы аппаратно, при этом, благодаря матричной архитектура, могут выполняться в несколько потоков.

В сверточных нейронных сетях нейроны одного слоя используют одни и те же «разделяемые» веса и осуществляют операцию свертки [121]. Объединяя результаты вычислений нейронов с одинаковым набором весовых коэффициентов, получают карты признаков, которые содержат информацию о тех или иных характерных чертах изображения на большом уровне абстракции. Помимо описанного слоя, сверточные нейронные сети содержат слои субдискретизации, уменьшающие размерность пространства карт признаков, а также полносвязные слои на выходе сети, завершающие процесс распознавания. Основная функция данного типа сетей – работа с изображениями и распознавание образов [102].

Обобщенная схема обработки данных в сверточной нейронной сети показана на рисунке 3.3. Исходное изображение переводится в оттенки

серого таким образом, что каждому пикселю соответствует одно число в диапазоне от 0 до 255. Из полученных чисел формируется вектор и передается вместе с массивами значений ядер первого, второго и третьего уровней, а также весовыми коэффициентами последнего слоя в функцию вычисления прямого прохода сверточной сети. В соответствии с размерами окна формируется массив указателей так, что каждый указатель хранит информацию о номере первого пикселя соответствующего окна. Если принять ширину и высоту исходного изображения за W и H, а размеры окна за x и y, то количество указателей можно рассчитать по формуле:

$$N = (W - x + 1) * (H - y + 1). \quad (3.2)$$

Далее в цикле для каждого окна осуществляется операция взвешенного суммирования с ядром, результат помещается в соответствующую позицию массива карт текущего слоя [144]. Принимая во внимание специфическую архитектуру процессора, взвешенное суммирование проходит в три этапа: вычисление первоначального вектора, затем цикл обработки основной части окна (весовые коэффициенты ядра кодируются в формат теневой и рабочей матриц в фоновом режиме) и финальное сложение двух частей регистра AFIFO.

Аналогично формируются карты признаков второго и третьего уровней, отличия заключаются в уменьшении размеров самих карт, окон и ядер, а также увеличении общего их количества и разграничений между областями выходных векторов слоя.

Карты признаков третьего слоя, имеющие самый малый размер и аккумулирующие наиболее общую информацию о признаках исходного изображения, объединяются в вектор, который подается на последний слой, представляющий из себя классический персептрон из ста нейронов.

Заполнение массива указателей происходит с помощью двух меток и условных переходов, регулируемых значениями двух универсальных регистров. Для каждого уровня карт признаков используются отдельные массивы указателей и окон, но для экономии памяти можно использовать

одни и те же массивы, изменяя только границы адресации, т.к. по мере выполнения алгоритма требуемые размеры массивов уменьшаются.

В общем процесс обучения требует объявление следующих структур: входного вектора 96x96 элементов, массивы ядер для каждого слоя сети, массивы выходов нейронов, формирующие карты признаков, выходной вектор, массивы ошибок, накопленных на каждом из слоев сети, веса выходного слоя.

Таким образом, в разработанном параллельном алгоритме, по сравнению с функциями, написанными на языках высокого уровня, достигается определенное увеличение скорости за счет следующих факторов:

1. Использование языка ассемблера всегда дает выигрыш в скорости и позволяет производить вычисления на более низком уровне, обращаться непосредственно к памяти, а также регистрам и другим элементам архитектуры процессора [132].

2. В использованном алгоритме окна не формируются заранее, образуя большой исходный вектор, а вычисляются непосредственно во время загрузки весовых коэффициентов и взвешенного суммирования.

3. Загрузка и кодирование коэффициентов в теневую и рабочую матрицы происходит в фоновом режиме.

4. Матричная архитектура позволяет параллельно рассчитывать результат для нескольких нейронов слоя, а также ускорять операции с одиночными векторами.

3.3 Разработка параллельного алгоритма обработки данных в сверточной нейронной сети второго порядка

Разработанный параллельный алгоритм обработки данных в СНС второго порядка базируется на распределенном вычислении взвешенных сумм отдельных нейронов, а также применении операции маскирования входного вектора для формирование промежуточного набора векторов для

загрузки в рабочую матрицу и вычисления операции взвешенного суммирования для отдельных частей формул прямого и обратного распространения. Разбиение входного вектора на окна происходит непосредственно внутри алгоритма через использование массива указателей, таким образом не требуя сложной предобработки данных. Заполнение матрицы весовыми коэффициентами производится в теневом режиме, активно используются векторные регистры процессора для оптимизации количества обращений к памяти.

При использовании сумматоров второго порядка алгоритм вычисления карт признаков будет модифицирован с учетом выполнения следующих шагов для каждого нейрона:

1. Рабочая и Теневая матрицы разбиваются на ячейки 4x4. Возможно разбиение на большее число ячеек, но такой подход увеличивает риск переполнения единиц памяти при обработке больших массивов данных.

2. Загружаются весовые коэффициенты W и входы нейрона X, выполняется операция взвешенного суммирования, полностью аналогичная вычислениям в стандартной модели нейронов.

3. После этого вектор входных значений модифицируется с помощью операции маскирования, превращаясь в четыре новых вектора, содержащих по одному входному значению в нужных позициях.

4. Сформированный вектор квадратов входных значений загружается в рабочую матрицу, повторяется операция взвешенного суммирования с весовыми коэффициентами второго уровня.

5. Результат обрабатывается активационной функцией.

В остальном прямой проход сети будет аналогичен описанному ранее алгоритму для стандартной сверточной сети.

В регистр ar0 сохраняется адрес операнда X, весовые коэффициенты загружаются в теневую и рабочую матрицы, сохраняются адреса маски и переменной temp соответственно в регистры ar0 и ar1, далее четыре раза повторяется операция взвешенного суммирования с подгрузкой весовых

коэффициентов и сохранение результата операции из регистра afifo в память по адресу, хранящемуся в регистре ar1.

Далее в регистр ar0 записывается адрес ядра, весовые коэффициенты загружаются в теневую и рабочую матрицы и, с помощью команды vsum производится операция взвешенного суммирования значений, полученных на предыдущих шагах с ядром. Результат вычисляется с помощью еще одного использования векторного процессора.

Обновление весов происходит либо после каждого прохода сети, либо после обработки всех примеров обучающей выборки. Второй способ обеспечивает более качественную подстройку весовых коэффициентов, но в обоих случаях задача решается с помощью векторного АЛУ.

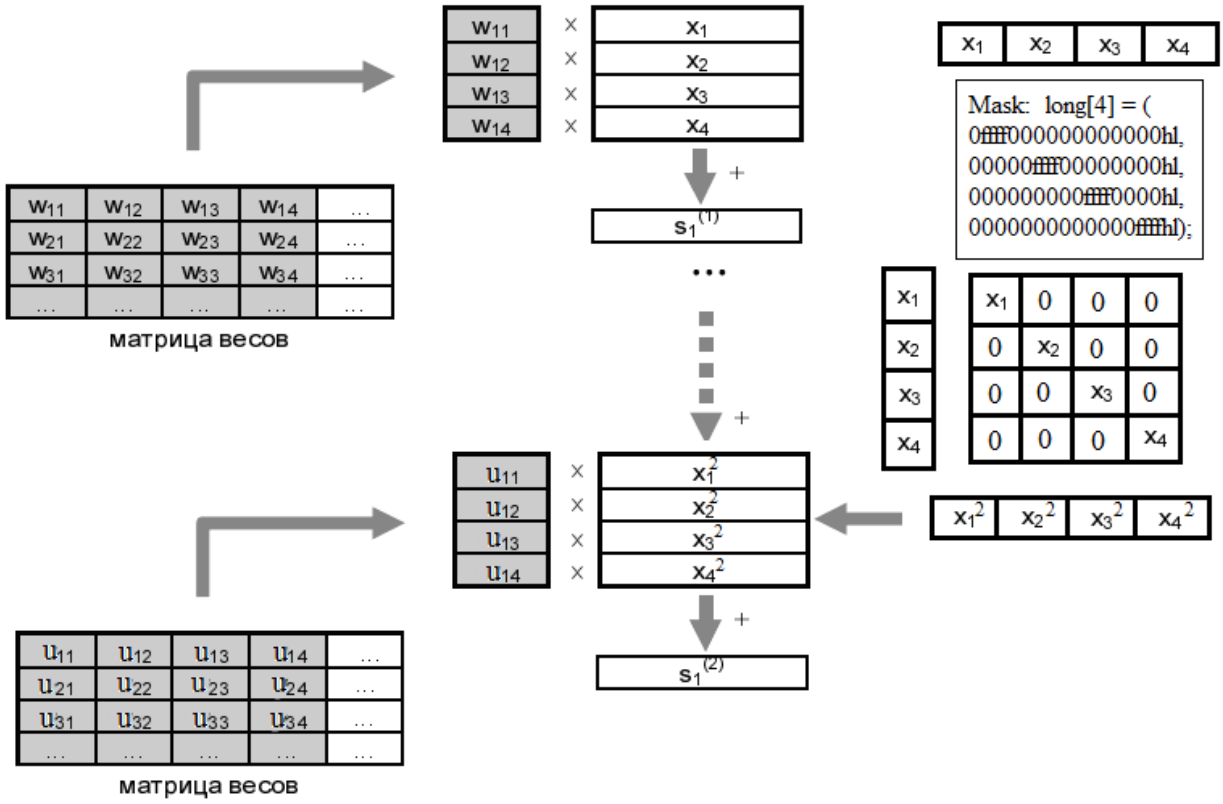


Рисунок 3.5 – Обобщенная схема вычисления взвешенных сумм нейрона второго порядка

На языке низкоуровневого программирования это выглядит следующим образом: 32 раза в регистр ram загружаются значения необходимого изменения настраиваемых параметров, устанавливается

счетчик в регистре gr2, далее следует цикл с условным переходом, в котором значение регистра gr2 постепенно изменяется при переходе и производятся запись по адресу, хранящемуся в регистре ar1 суммы регистрво afifo и ram и пустая команда.

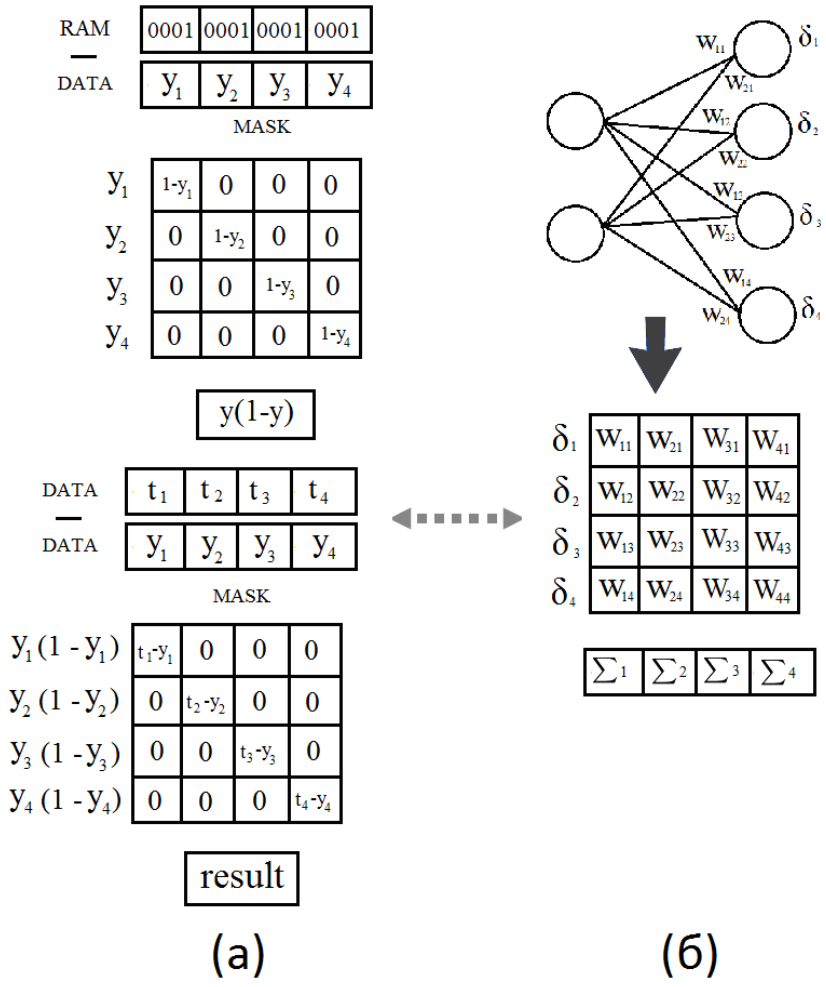


Рисунок 3.6 – Обобщенная схема алгоритма обратного распространения (а) для выходного слоя (б) для скрытого слоя

Обратный проход включает в себя вычисление градиента ошибки для каждого нейрона каждой карты каждого слоя нейронной сети. Для выходного слоя:

$$\delta_i = (t_i - y_i) \cdot f'(y_i). \tag{3.3}$$

Для скрытых слоев:

$$\delta_i = f'(y_i) \cdot \sum \delta_j \cdot W_{ij}. \tag{3.4}$$

При использовании сигмоидальной функции активации:

$$f(x) = \frac{1}{1+e^{-x}}, \tag{3.5}$$

производная будет выглядеть следующим образом:

$$f'(x) = f(x) \cdot (1 - f(x)). \tag{3.6}$$

Поэтому формулы преобразуются для выходного слоя:

$$\delta_i = -y_i \cdot (1 - y_i) \cdot (t_i - y_i). \tag{3.7}$$

Для скрытых слоев:

$$\delta_i = y_i \cdot (1 - y_i) \cdot \sum \delta_j \cdot W_{ij}. \tag{3.8}$$

Если рассматривать формулу 3.7 с точки зрения вычисления на векторном процессоре, то можно разбить ее на несколько простых операций на векторном АЛУ с использованием регистра ram: 0 – ram, 1 – ram, data – ram. После чего осуществляется взвешенное суммирование в два прохода, повторенное с помощью команды rep.

На рисунке 3.7 изображен обобщенный параллельный алгоритм обработки сигнала в сверточной нейронной сети второго порядка для слоя С1 с использованием процессоров векторно-матричной архитектуры.

При обработке данных на первом сверточном слое изображение фиксированного размера 96x96 пикселей подается на вход нейронной сети, преобразуется в вектор таким образом, чтобы каждое последующее состояние плавающего окна получалось посредством загрузки очередного блока данных. В память загружаются указатели на ядра и выходы слоев, после чего производится операция маскирования с помощью взвешенного суммирования на векторном процессоре, в результате которой получается вектор квадратов входных значений. Далее в режиме параллельной обработки вычисляются первая и вторая степени полинома квадратичного сумматора, которые складываются между собой и обрабатываются аппаратно реализованной операцией нелинейного преобразования, образуя элемент выходной карты признаков. Одновременно с этим загружаются новые

весовые коэффициенты в теневую матрицу процессора. Алгоритм повторяется для каждой из карт текущего слоя.

На рисунке 3.8 приведен обобщенный алгоритм обработки сигнала при обратном распространении для отдельного слоя сети.

Алгоритм обратного распространения начинается с загрузки в память выходного и эталонного для данного примера значений, а также инициализации регистра ram единичным при разбиении на четыре части вектором. В режиме параллельной обработки на векторном АЛУ для каждого компонента выходного вектора вычисляется значение $(1-y_i)$, затем производится операция маскирования и взвешенное суммирование с тем же вектором, после которого получается значение $y_i \cdot (1-y_i)$. Далее формулы вычисления будут отличаться для внешнего и внутренних слоев. В первом случае из памяти загружается значение эталонного вектора и на векторном АЛУ вычисляются значения $(t_i - y_i)$ для каждого компонента выхода сети. В завершении с помощью аппаратной операции маскирования и взвешенного суммирования на векторном процессоре получаются необходимые значения изменения настраиваемых параметров нейронной сети.

В случае вычисления значений для внутреннего слоя сначала для каждого нейрона вычисляется сумма произведений весовых коэффициентов и невязок для каждой связи с нейронами последующего слоя. После чего производится операция маскирования и взвешенное суммирование на векторном процессоре для вычисления результата.

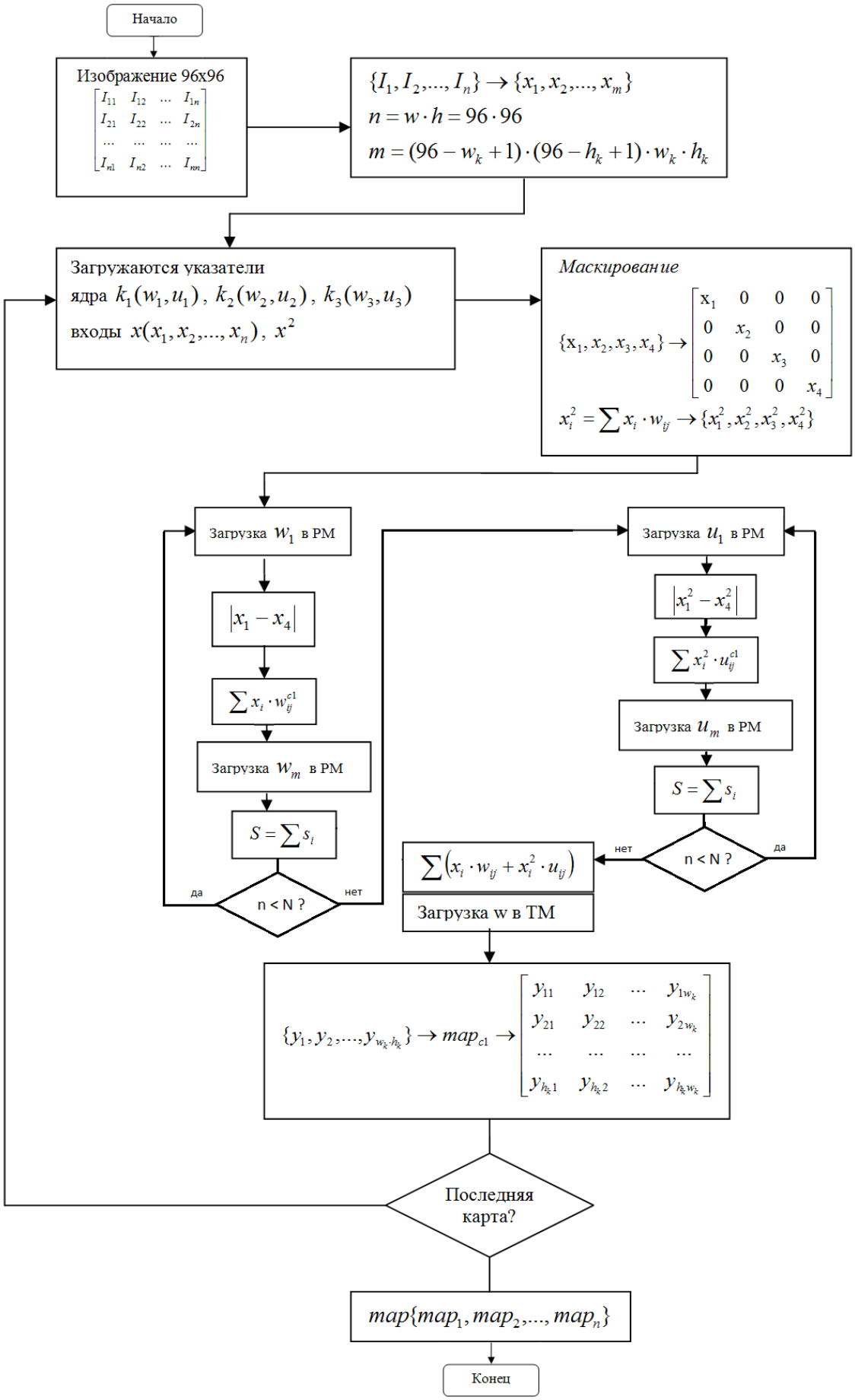


Рисунок 3.7 – Обобщенный алгоритм обработки сигнала в СНС второго порядка при переходе с входного слоя на слой С1.

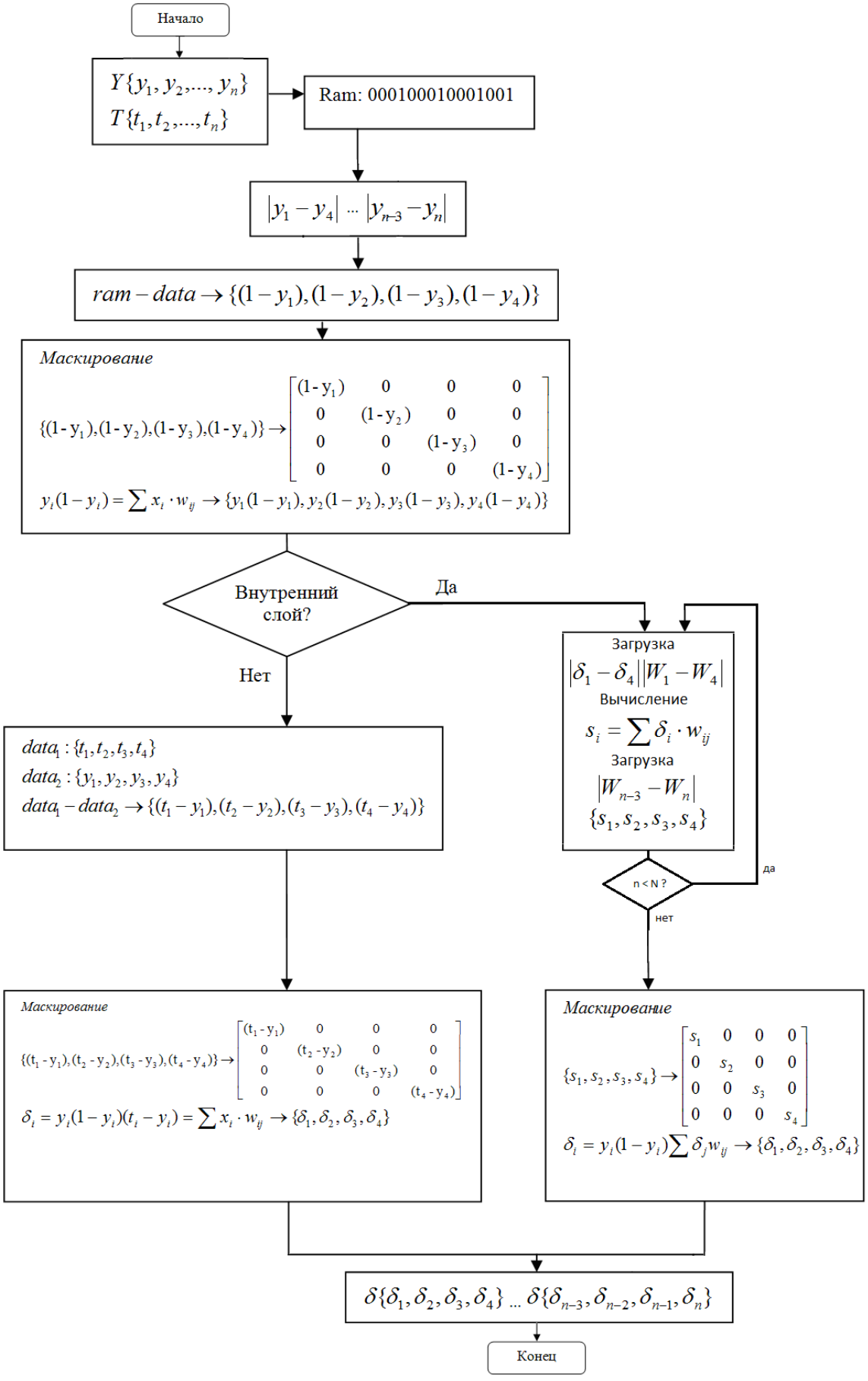


Рисунок 3.8 – Обобщенный алгоритм обработки сигнала в СНС второго порядка при обратном проходе на выходном и внутренних С-слоях

Выводы

1. Разработан параллельный алгоритм обработки данных в сверточных нейронных сетях второго порядка, ориентированный на процессоры векторно-матричной архитектуры. Выполнена вторая подзадача диссертационного исследования.

2. Доказаны актуальность и применимость процессоров векторно-матричной архитектуры для реализации различных моделей нейронных сетей: полносвязной многослойной сети прямого распространения, СНС, СНС второго порядка. Описаны и проанализированы структура и основные компоненты векторно-матричных процессоров NeuroMatrix.

3. Сформулированы основные принципы реализации сверточных нейронных сетей на процессорах с векторно-матричной архитектурой, в том числе: оптимальное использование теневой и рабочей матриц, векторных регистров процессора, оптимальная аппаратная реализация активационных функций, применение операций взвешенного суммирования и маскирования данных. Выявлены основные принципы обработки данных в нейронах с сумматорами второго порядка.

4. Проведен анализ существующих основных линеек процессоров, ориентированных на векторно-матричную обработку данных. Доказана актуальность и широкая распространенность данного вида процессоров.

5. Приведены схемы параллельного алгоритма обработки данных в сверточной нейронной сети второго порядка при прямом и обратном проходах сигнала по сети, схема вычисления взвешенной суммы для нейрона второго порядка на векторном процессоре. Обобщенно визуализирована обработка данных и формирование карт признаков на отдельных слоях нейронной сети.

6. Описание экспериментальной апробации параллельного алгоритма приведено в пятой главе.

ГЛАВА 4 РАЗРАБОТКА МЕТОДИКИ ПОЛУАВТОМАТИЧЕСКОГО СОЗДАНИЯ ВИЗУАЛЬНЫХ ОБУЧАЮЩИХ ВЫБОРОК ДЛЯ НЕЙРОННЫХ СЕТЕЙ

Данная глава содержит описание разработанной методики полуавтоматического создания визуальных обучающих множеств для нейронных сетей, а также описание экспериментов относительно того, как влияют различные параметры и способ генерации изображений обучающей выборки, применение фильтрации и деформации изображений на качество обучения и обобщающую способность нейронной сети, а также влияние освещенности и расширения обучающей выборки за счет изменения интенсивностей пикселей входящих в нее изображений.

4.1 Анализ методов предобработки изображений и этапов формирования обучающей выборки

На качество распознавания образов и обобщающую способность нейронной сети в равной степени влияют два фактора: нейросетевая архитектура и обучающая выборка, на которой проводилось обучение [45]. Даже при использовании современных моделей СНС, если выборка будет составлена неправильно, то и результат будет неудовлетворительным. Поэтому крайне важно исследовать условия и параметры, влияющие на качество обучения. Обучающая выборка содержит в себе всю информацию, которая будет доступна нейронной сети для решения задачи распознавания, никаких других источников данных сеть не имеет, поэтому качество обучающей выборки, как проекции отдельной области реального мира, является критично важным фактором успешного обучения. Даже если использовать современные мощные математические модели, если не удастся обеспечить достаточно высокое качество обучающей выборки, правильное распознавание будет в принципе невозможно. Поэтому, очень важно понять,

какие факторы оказывают ощутимое воздействие на качество обучающей выборки для решения конкретной задачи, какими правилами стоит руководствоваться при создании визуальной обучающей выборки. Для того, чтобы это понять, необходимо, во-первых, определить параметры и свойства, которые теоретически могут иметь значение в данном контексте, во-вторых, провести экспериментальное исследование по созданию различных выборок, в которых будет варьироваться степень влияния выбранных параметров, а также последующее обучение и оценку обобщающей способности нейронной сети.

Для решения наиболее общих задач распознавания, а также научных исследований существуют готовые решения, большие обучающие выборки, например NORB [87], содержащие большое количество изображений определенных типов. Но для решения конкретных прикладных задач, в ситуациях, когда требуется ограничить, или наоборот расширить выборку конкретным набором объектов, может потребоваться составление нового обучающего множества.

При составлении обучающей выборки для нейронной сети, очень важно осуществить нормализацию входящих в нее примеров, таким образом, обеспечив их единообразие [22]. В случае работы с визуальными образами, в качестве такой предобработки используются: перевод изображений в оттенки серого для того, чтобы уменьшить цветовое разнообразие паттернов и существенно снизить сложность задачи распознавания образов; сжатие изображений с помощью бикубической интерполяции с целью приведения входных паттернов к фиксированному размеру; локальная нормализация; обработка изображений с помощью фильтров Габора; применение пороговой функции, для того, чтобы исключить присутствие среди выходов фильтра отрицательных и чрезмерно больших значений.

Были выделены следующие аспекты создания обучающего множества, которые могут напрямую повлиять на качество последующего обучения: способ создания выборки, тип фона для изображений выборки, параметры

изображений обучающей выборки, фильтрация изображений, аффинные преобразования и эластичные искажения, влияние освещения на качество распознавания, важность наличия «пустого» класса в выборке. Поэтому процесс формирования и настройки обучающей выборки для задачи распознавания визуальных объектов нейронными сетями включает следующие этапы:

1) Выбор источника данных для обучения и способа создания обучающих изображений.

2) Интерполяция, сжатие изображений до требуемого размера.

3) Перевод изображения в оттенки серого.

4) Нормализация.

5) Настройка параметров изображений.

6) Применение фильтров.

7) Наложение шума.

Рисунок 4.1 – Предобработка данных обучающей выборки

В разработанной методике используются три основных этапа предобработки данных: интерполяция, перевод изображения в оттенки серого, нормализация.

Билинейная интерполяция – функция вычислительной математики, расширяющая линейную интерполяцию и нашедшая широкое применение в задаче изменения размеров изображения [47]. Ее действие заключается в интерполяции значения изображения в определенной точке на основании данных о точках, которые ее окружают. Этот шаг является крайне важным ввиду того, что размерность входов нейронной сети крайне ограничена вычислительными ресурсами – даже небольшое изображение 100x100 пикселей формирует вектор из 10 000 элементов. Поэтому, прежде чем

использовать созданный пример для обучения, его необходимо сжать до приемлемого размера, при этом сведя к минимуму потерю важных для обучения данных о свойствах и признаках объекта, и, напротив, максимально выделив участки изображения, аккумулирующие важные черты объекта, напрямую влияющие на качество обучения нейронной сети.

Алгоритм билинейной интерполяции выглядит следующим образом: сначала выбирается первичное направление, то есть ось абсцисс или ось ординат. В выбранном направлении осуществляется интерполяция между известными точками и получения двух вспомогательных значений R1 и R2, между которыми повторно проводится линейная интерполяция. При этом порядок следования осей в алгоритме не оказывает влияния на результат вычислений.

Если принять значения функции вокруг точки некоторой D за $F_{11} = (x_1,y_1)$, $F_{12} = (x_1,y_2)$, $F_{21} = (x_2,y_1)$ и $F_{22} = (x_2,y_2)$, то вспомогательные значения могут быть получены по формулам:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(F_{11}) + \frac{x - x_1}{x_2 - x_1} f(F_{21}), \quad (4.1)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(F_{12}) + \frac{x - x_1}{x_2 - x_1} f(F_{22}). \quad (4.2)$$

Тогда среднее значение функции в точке D вычисляется следующим образом:

$$f(D) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2). \quad (4.3)$$

Перевод изображения в оттенки серого – вычисление для каждой точки усредненного значения ее RGB-составляющих. Основная цель этого процесса – уменьшение размерности входного вектора данных [9]. Большинство современных систем распознавания работает с монохромными изображениями, в таком виде гораздо проще выделять особенности объекта, при этом существенно взрастают скорость и точность обработки данных.

Существуют различные подходы к *нормализация изображения*: масштабирование, локальная контрастная нормализация, стандартизация особенностей [63].

Простое масштабирование применяется в том случае, если используемая модель распознавания предполагает, что элементы вектора входящего сигнала находятся в границах [0,1] или [-1,1]. Изображения выборки чаще всего кодируются числами в диапазоне от 0 до 255, поэтому простое масштабирование в этом случае может быть достигнуто путем деления каждого числа, составляющего вектор, на 255.

Если данные являются стационарными (имеющими одинаковое распределение по всем измерениям), то возможно вычитание среднего значения для каждого примера. В случае работы с изображениями, вычисляется так называемое «среднее изображение», которое вычитается из каждого примера обучающей выборки для того, чтобы исключить общие для всех изображений элементы, оставив только уникальную, полезную для обучения информацию.

Возможен также усложненный вариант – *стандартизация особенностей* (features standardization), при котором каждое измерение данных должно иметь нулевое математическое ожидание и единичную дисперсию, что достигается за счет вычисления и последующего вычитания среднего для каждого измерения в рамках обучающей выборки, и деления результата на стандартное отклонение [75]. Такой метод нормализации широко используется при работе с машинами опорных векторов.

В теории глубокого обучения широкое практическое применение нашла *локальная контрастная нормализация*, основанная на исследованиях восприятия визуальной информации человеческим мозгом [101]. Данный метод направлен на преобразование изображения с целью обеспечения однородности математического ожидания и дисперсии в определенной области. Локальная нормализация может принести большую пользу в условиях непостоянного освещения и теневых артефактов.

4.2 Экспериментальное исследование влияния способа формирования изображений обучающей выборки на обобщающую способность нейронной сети

При создании обучающей выборки наиболее первостепенным фактором, влияющим на результат, является способ формирования обучающих примеров.

Распознавание визуальных образов является задачей, имеющей невероятную вычислительную сложность, поскольку каждый объект, находящийся в реальном мире, может отражаться на сетчатке глаза и формировать бесконечное число двумерных изображений в зависимости от своего расположения в пространстве, позиции относительно наблюдателя, освещения, заднего фона и т.д. Чтобы научить искусственную систему с большой вероятностью соотносить представленное изображение с ограниченным набором объектов, необходимо составить большую обучающую выборку, включающую в себя тысячи различных изображений, обеспечивающих необходимое разнообразие состояний каждого из объектов. При решении данной задачи существует два основных подхода [150].

Первый из возможных подходов – это создание большого набора «естественных» изображений, непосредственно снимков реального мира. При правильном выполнении такой подход способен дать мощную обучающую выборку, с помощью которой возможно добиться хорошего уровня распознавания нейронной сетью, но создание выборки потребует огромных трудозатрат и большого количества времени [10].

Второй подход заключается в том, чтобы синтезировать изображения, используя как данные, непосредственно связанные с целевым объектом, так и не имеющие к нему прямого отношения. Парадоксальность ситуации заключается в том, что, опыт применения таких обучающих выборок как NORB (NYU Object Recognition Benchmark) и Caltech, показал, что такие выборки могут более полно отражать данные об окружающем мире и

служить лучшим ресурсом для обучения нейронных сетей, чем наборы «естественных» фотографий, потому как лучше охватывают диапазон возможных вариаций изображения, наблюдаемых в реальном мире.

Существуют различные типы представления обучающей выборки. При проведении исследований были использованы два распространенных типа. Первый, характерный для базы данных NORB [119] – это бинарный файл, в котором первые два числа кодируют общее количество примеров и размер единичного паттерна. За ними следует набор из N матриц, кодирующих изображения, представленные в оттенках серого, где каждому пикселю соответствует одно число в диапазоне от 0 до 255. Для определения типа объекта, закодированного в очередной матрице, к файлу выборки добавляются еще два файла, один из которых содержит вектор чисел с последовательностью номеров объектов, а другой – информацию о выборке в целом, а также соответствия номеров объектов их реальным наименованиям. Второй тип представления данных обучающей выборке, характерный для библиотеки FANN (Fast Artificial Neural Network Library) [94]: выборка представляет собой текстовый файл, в первой строке которого записаны три числа – размер выборки, количество входов и количество выходов – а далее строки поочередно кодируют очередной входной пример и требуемое выходное значение.

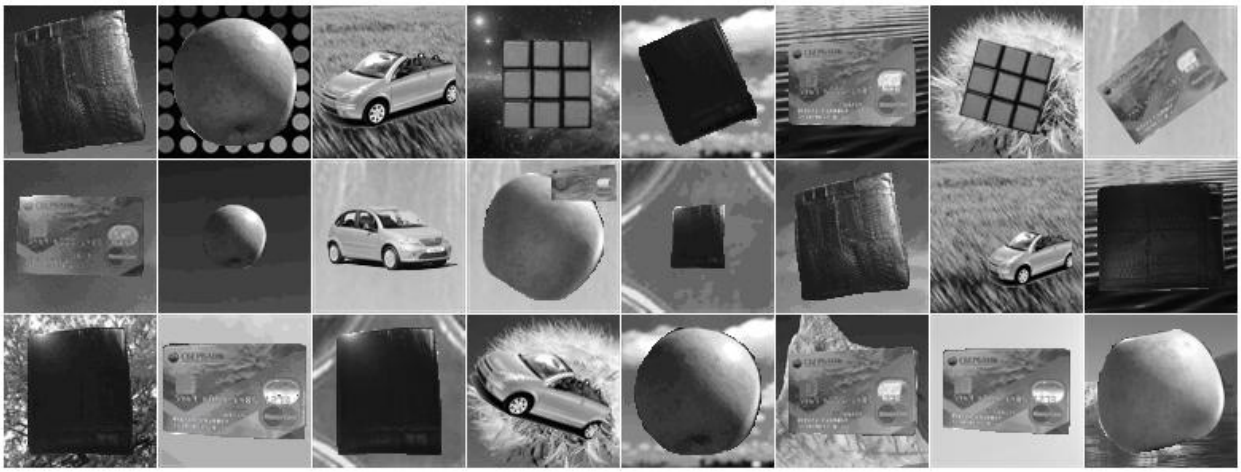


Рисунок 4.2 – Примеры изображений обучающей выборки

Для разработки методики создания визуальных обучающих выборок большой интерес и первостепенное значение имеют *способы генерации примеров*, то есть то, как формируется массив изображений, и какие методы автоматизации при этом используются. В случае с визуальными выборками наиболее показательным моментом является источник информации об объекте: в разных случаях это могут быть объекты реального мира, синтезированные данные, полученные искусственным путем, а также комбинирование этих двух подходов [31]. На основании этого было выделено три основных способа формирования обучающей выборки: «ручное» создание выборки или сбор большого количества примеров, использование синтеза на основе 3d-моделей и генерация примеров обучающего множества из фотографий объектов и подготовленных фоновых изображений.

Первый способ. Паттерны обучающей выборки являются непосредственным отражением реального мира. Другими словами, выборка формируется путем ручного создания многочисленных снимков объекта, что является довольно трудоемким процессом, который, однако, в определенной степени поддается автоматизации [149].

Ручное формирование выборки является наиболее старым способом, который применялся на заре создания нейронных сетей. Он представляет собой довольно монотонную работу по ручному созданию множества изображений определенной тематики, отсканированных, сфотографированных, либо цифровых рисунков. Чаще всего данный способ применялся для создания достаточно простых выборок, например рукописных цифр и букв. Целью в данном случае является создание большой коллекции возможных «ситуаций» (фотографий объекта при различных внешних условиях, а также с разных точек зрения), которые в совокупности содержали бы в себе достаточно информации для того, чтобы позволить искусственной системе провести процесс обучения и с необходимой

вероятностью распознать объект. Кроме большой сложности метод ручного создания выборок содержит и другие характерные особенности, делающие реальное его применение достаточно затруднительным. Так, неочевидным является необходимое число фотографий для оптимального обучения нейронной сети, а также то, каким образом следует обеспечивать единообразие и сочетаемость паттернов выборки для того, чтобы они имели синергическое действие.

Второй способ. Все данные, составляющие изображения, синтезируются с помощью вычислительных средств, для чего используются 3D-модели, а также коллекция синтезированных, либо подготовленных заранее фонов.

Подобный способ получил практическое применение в виде обучающей выборки Caltech 101 [95] и хорошо подходит для применения в задачах с использованием наиболее общих категорий объектов. В качестве исходного источника данных используются 3D-модели, имеющие достаточно малый уровень детализации. К примеру, это могут быть модели самолета, автомобиля, птиц, животных и т.д. С помощью вычислительного алгоритма трехмерное представление объектов преобразуется в определенный набор пикселей двухмерной сцены, в зависимости от точки наблюдения, которая равномерно изменяется в соответствии с заданным шагом алгоритма. Затем из полученных изображений исключается виртуальный, не несущий никакой дополнительной информации, фон, а освободившееся пространство заполняется заранее подготовленными, либо синтезированными графическими данными. Таким образом, получается автоматизированное создания большого числа изображений, отражающих то, как объект выглядит для системы-наблюдателя с разных ракурсов, а также необходимое разнообразие области, которая объектом не является, что также важно для качественного обучения нейронной сети.

Третий способ. Усредненный вариант, когда паттерны формируются с помощью наложения рисунков объектов на фоны, но сами объекты

выделяются из фотографий реального мира. То есть в данном случае используются оба источника данных, но при этом многообразие объектов сводится к фиксированному набору ракурсов, что компенсируется за счет большого количества фонов, а также применения алгоритмов трансформации признаков (моделирование освещения и т.д.).

Такой способ создания обучающих выборок комбинирует первые два подхода таким образом, что паттерны обучающей выборки формируются методом простого синтеза, с использованием заранее подготовленных фоновых изображений, а также реальных данных о конкретном объекте физического мира, выделенных из нескольких снимков, снятых с разных ракурсов. Похожий подход был использован при создании базы данных NORB.

В качестве исходного материала для экспериментов по выбору типа создания изображений были отобраны следующие классы объектов: «яблоко», «чашка», «лимон», «бокал», «книга». На рисунке 4.3 приведены изображения исходных предметов.

Рисунок 4.3 – Предметы и 3D-модели, послужившие материалом для создания обучающей выборки

Для того чтобы проверить как различные типы формирования обучающей выборки влияют на качество обучения, распознавания и

обобщающую способность нейронной сети, был создан различными способами набор обучающих выборок, а также тестовое множество изображений. Для обеспечения объективности исследования в тестовую выборку не входило ни одно из изображений какой-либо обучающей выборки. В каждом случае тестовый материал был одним и тем же и содержал фотографии объектов, сделанные в реальных условиях. Количество паттернов в каждой обучающей выборке для каждого класса предметов, а также общее количество примеров были одинаковыми. Для проведения эксперимента потребовалось десять физических объектов, по два на каждый класс предметов, для создания тестовой и обучающих выборок, а также четыре 3D-модели.

Для создания *первого типа выборки*, содержащей снимки реального мира, была написана программа, осуществляющая многократное создание снимков объекта при повороте и перевод в соответствующий формат. При этом для эмулирования естественного фона для объектов был использован монитор с запущенным видеорядом, за счет которого фон постоянно менялся, внося разнообразие в содержащуюся в выборке информацию. Кроме этого, в функционал программы входила возможность выделить квадратный целевой участок видео-потока, информация которого непосредственно использовалась для создания паттернов выборки.

Второе обучающее множество было создано с применением трехмерных моделей и специального программного обеспечения, которое позволяло производить построение и вывод модели на экран, постепенно изменяя значение угла обзора. Полученные таким образом изображения объектов накладывались на подготовленную коллекцию фонов, тем самым создавая большое количество примеров для обучения.

Третья выборка была создана похожим на вторую образом, но с использованием реальных фотографий объектов, из которых вырезалась часть, хранящая данные о предмете, и накладывалась последовательно на

каждый из имеющихся фонов. Для второго и третьего типов использовался одинаковый набор фонов.

Результаты эксперимента показали, что наименьшее качество обучения, выражающееся через обобщающую способность сети, достигается при использовании первого типа выборок, использование синтезирующего подхода приносит ощутимую прибавку в количестве правильно распознанных примеров по сравнению с первым подходом, но наилучшую степень распознавания нейронная сеть демонстрирует при применении в качестве обучающего множества выборки третьего типа.

Для того, чтобы проверить сохраняется ли похожая картина при увеличении размера обучающей выборки, были созданы новые множества примеров способами, аналогичными вышеописанным, но содержащие в два раза больше, а также в два раза меньше паттернов для обучения. Результаты экспериментов представлены в таблице 4.1. I – ручное создание выборки, II – метод использования 3d-моделей объектов, III – метод наложения фотографий на фоны. Обобщающая способность нейронной сети – параметр, позволяющий оценить качество распознавания через отношение числа корректно распознанных примеров к общему размеру валидационного множества. Таким образом, обобщающая способность напрямую связана с понятием ошибки обобщения.

Таблица 4.1 – Обобщающая способность нейронной сети после обучения на выборках различного размера, созданных с применением одного из трех способов генерации изображений

Размер обучающей выборки	Тип формирования изображений		
	I	II	III
150	45	72	78
300	65	79	85
600	48	81	83

На рисунке 4.4 показана динамика корректного распознавания выборок, созданных различными способами, для двух и пяти категорий объектов. Процент правильно распознанных образов для задачи классификации по двум классам превосходит аналогичный параметр распознавания пяти категорий, потому что эта задача является более простой и базовой, но общее соотношение для различных подходов не изменяется.

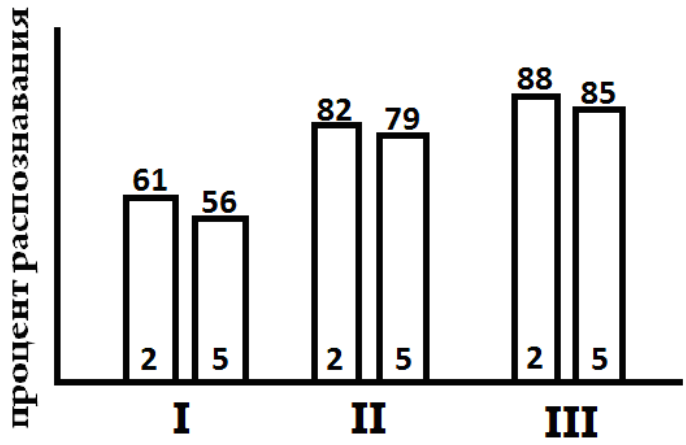


Рисунок 4.4 – Сравнение результатов распознавания для случаев двух и пяти классов объектов

Можно сделать вывод, что оптимальным является третий тип формирования изображений выборки. Получение обучающих выборок с помощью синтеза 3d-моделей и двумерных фонов дает меньшие по сравнению с третьим подходом, но достаточно высокие результаты. Первый тип формирования выборок дал самые низкие результаты обобщающей способности нейронной сети.

4.3 Экспериментальное исследование влияния типа фона изображений обучающей выборки на обобщающую способность нейронной сети

Важной составляющей формирования обучающей выборки изображений для нейронных сетей является влияние характера фона, на который будут накладываться изображения предметов, на качество обучения.

Для того, чтобы проверить какой тип фона является наиболее предпочтительным, были выделены следующие основные типы фонов: простой однотонный белый фон, изображения с фазовыми преобразованиями, белый шум и цены реального мира [128].

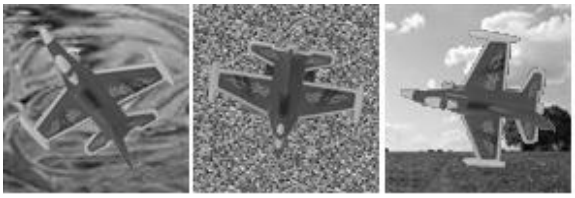


Рисунок 4.5 – Примеры изображений с разными типами фонов

Для тестирования применимости разных типов фонов была выбрана задача классификации по двум категориям, с автомобилем и самолетом в качестве классов объектов для распознавания. Наличие всего двух категорий объектов делает задачу распознавания довольно простой, но в то же время наглядной и хорошо подходящей для тестирования специфических параметров. Экспериментальные исследования на основе классификации двух классов объектов являются широко распространенной практикой.

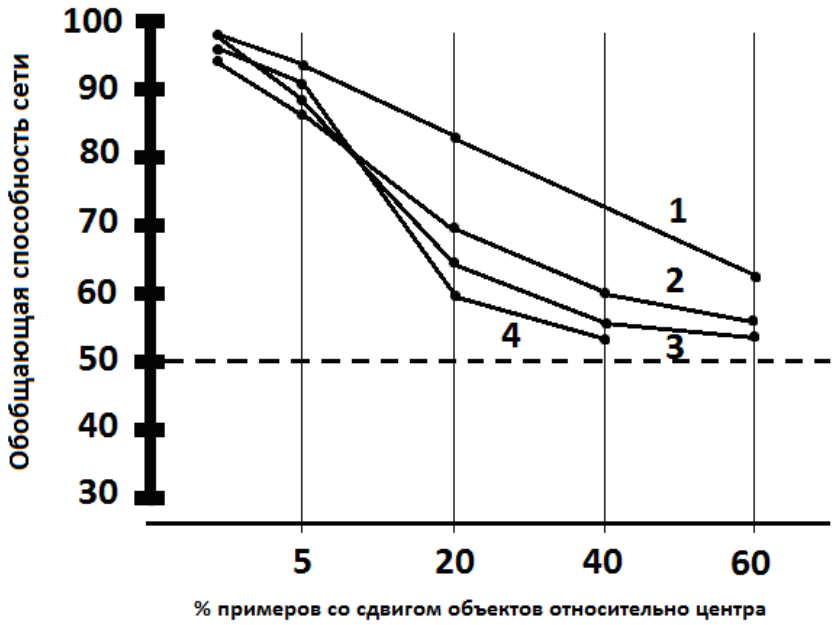


Рисунок 4.6 – График изменения обобщающей способности для разных типов фонов (1) сцены реального мира (2) белый шум (3) нулевой фон (4) фон с фазовыми преобразованиями

В случае, если изображения выборки имеют фиксированные размер и положение объекта на изображении, использование различных типов фонов не приводит к изменению обобщающей способности нейронной сети, из-за простоты задачи – для всех случаев результат одинаково высокий. Поэтому в эксперименте была использована вариативность различных параметров изображения. Результаты приведены на рисунке 4.6.

Таким образом наилучшую степень распознавания сеть продемонстрировала на выборке с реальными сценами. Белый фон хотя и имеет хорошие показатели на задачах с малой вариативностью, но при увеличении сложности задачи приводит к ухудшению качества обучения и распознавания. Изображения с фазовыми преобразованиями дали крайне нестабильные и неудовлетворительные результаты. Использование белого шума является хорошей альтернативой сложным фонам на основе реальных сцен.

4.4 Экспериментальное исследование влияния базовых параметров изображений обучающих выборок на обобщающую способность нейронной сети

Следующим шагом после выбора способа создания и типа фона для изображений обучающей выборки, является экспериментальное исследование того, как изменение основных параметров изображений (размер объектов, сложность фонов, сдвиг и поворот объектов и т.д.) влияет на качество обучения, и выбор оптимальных значений для этих параметров [36].

Для объективного и эффективного анализа необходимо выбрать критерии обучающей выборки, те признаки, варьируя которые можно будет оценивать их влияние на качество распознавания и обобщающую способность сети [59]. Все выбранные критерии являются базовыми и могут послужить хорошим фундаментом для создания больших обучающих

выборок. Во всех случаях для тестирования НС использовался один универсальный набор примеров, включающий пять различных объектов, снятых с пяти ракурсов, а также несколько заранее подготовленных наборов фонов, сгруппированных по признаку сложности. Таким образом, обеспечивались единообразие условий проверки и объективность тестирования.

Размер объектов. Размер входного вектора для нейронной сети фиксирован (96 на 96 пикселей), но размер объектов, накладываемых на фоны, может изменяться (были взяты границы для минимального значения – 40, для максимального – 90 пикселей). Эксперимент показал, что процент распознанных изображений в тестовой выборке при увеличении размера объекта плавно возрастает до определенного рубежа, примерно равного 80-85 % от размера фона, затем резко снижается (рис. 4.7, табл. 4.2).

Сложность фонов. Все собранные фоновые изображения были условно разбиты по степени сложности – от практически однотонных изображений, до комплексных, включающих множество деталей и насыщенных участков. По результатам экспериментов можно сделать вывод, что НС показывает наилучшую способность к распознаванию при обучении на несложных фоновых изображениях, даже при наличии фонов с большой степенью насыщенности и разнообразия элементов в тестовой выборке. В этом случае гораздо большее влияние на настройку весовых коэффициентов оказывает область, в которой сконцентрировано наибольшее количество информации об объекте. Несмотря на это, включение небольшого процента (5-15%) сложных фонов в обучающую выборку, положительно влияет на распознавание объектов в естественных сценах, а также трудных, зашумленных условиях (рис. 4.7, табл. 4.2).

Сдвиг объектов относительно центра. На данном этапе исследования в обучающую выборку последовательно добавлялось некоторое количество паттернов со смещенным положением целевого объекта. В итоге концентрация таких изображений в выборке менялась от 0

до 80 процентов. Несмотря на ожидание, что такие действия могут привести к улучшению обобщающей способности сети, результат оказался прямо противоположным – число правильно распознанных примеров неумолимо сокращалось. В среднем, увеличение части изображений со смещением на 10% в обучающей выборке, уменьшает долю правильно распознанных примеров на 1 – 2 %.

Зашумление. При исследовании данного параметра, на небольшой процент паттернов обучающей выборки, помещался посторонний объект, находящийся близко к границе изображения и не перекрывающий целевой объект. В случае небольшого числа зашумления, такой подход положительно влияет на обобщающую способность сравнительно с «чистой» обучающей выборкой, особенно при наличии сложных фонов в тестовой выборке, но с увеличением процента зашумленных изображений, дает обратный эффект.

Количество ракурсов объекта – увеличение точек, с которых сделаны снимки объектов для обучающей выборки, при неизменном общем количестве кадров. При единственном ракурсе качество распознавания было существенно ниже, чем при трех или пяти ракурсах. В то же время, чрезмерное увеличение вариативности точки наблюдения за объектом приводит к ухудшению обучения. При выборе точного количества ракурсов стоит учитывать также размер самой обучающей выборки.

Поворот объектов вокруг своей оси – может восприниматься как частный случай увеличения количества ракурсов объекта, поэтому данный критерий очень схож с предыдущим случаем. Однако за счет того, что вариативность ракурсов и количество паттернов, приходящееся на каждый ракурс, получается существенно меньше, то и вид кривой будет более пологим (рис. 4.8, табл. 4.2).

Изменение яркости\насыщенности изображений – в данном случае часть выборки обрабатывалась следующим образом: изображение переводилось из формата кодирования RGB в HSB (Hue, Saturation,

Brightness), после чего изменялся параметр насыщенности или яркости, а обновленное изображение приводилось к первоначальному виду.

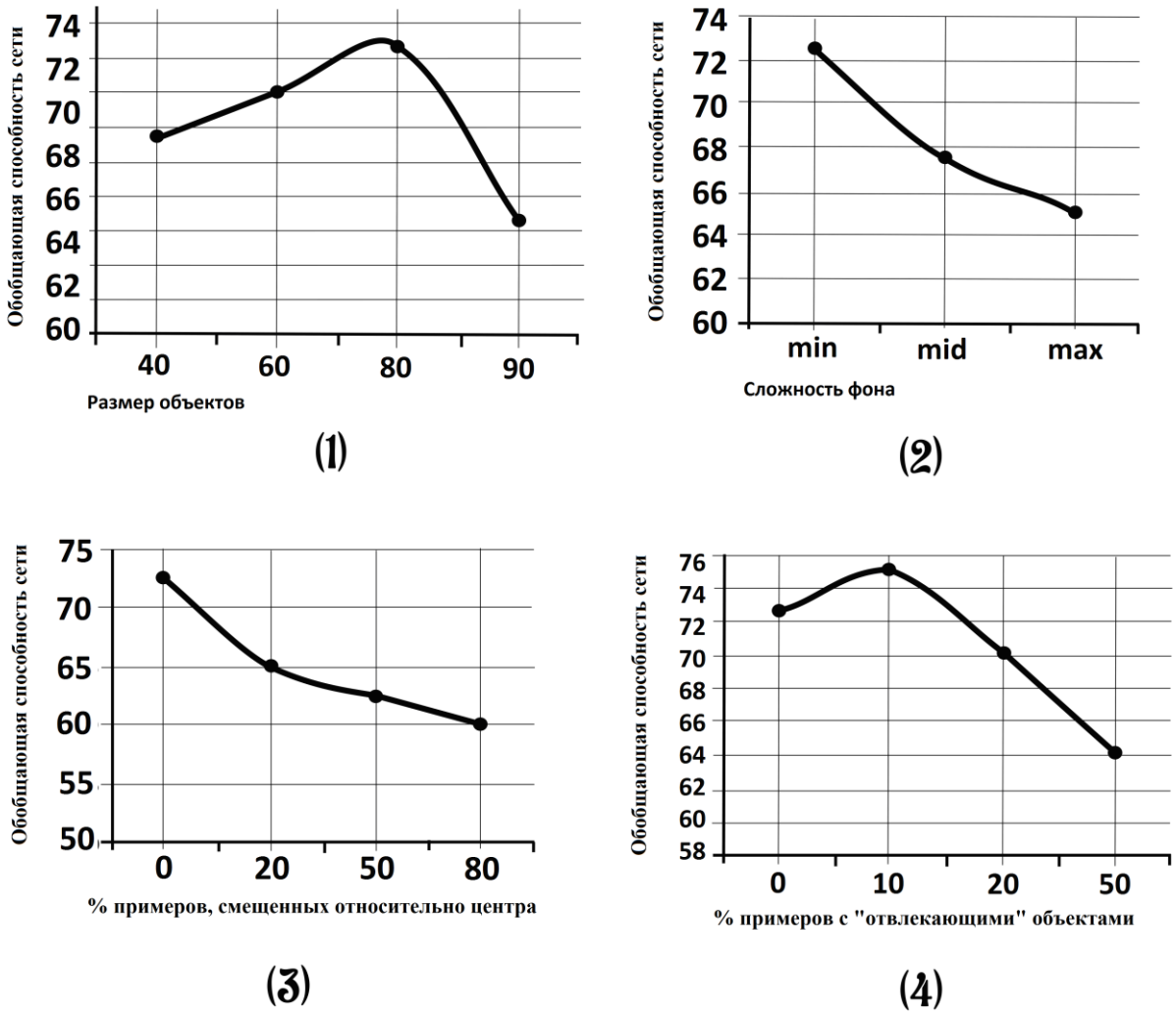


Рисунок 4.7 – Зависимость обобщающей способности сети от (1) размера объектов (2) сложности фона (3) процента изображений, смещенных относительно центра (4) наличия шума

В реальном мире освещение, расположение источников света является непостоянным и изменяется со временем, нельзя рассчитывать, что программа распознавания будет использоваться исключительно в идеальных условиях, поэтому обеспечение некоторой вариативности паттернов для обучения может обеспечить улучшение обобщающей способности сети. Однако, эксперимент не подтвердил каких-либо существенных улучшений или ухудшений качества распознавания при изменении процента

обработанных изображений в пределах 0 – 20 % и изменении параметра яркости в пределах 20 единиц. Как будет показано далее, изменение интенсивности изображений может существенно улучшить качество распознавания объектов нейронной сетью, но при условии низкой или неравномерной освещенности.

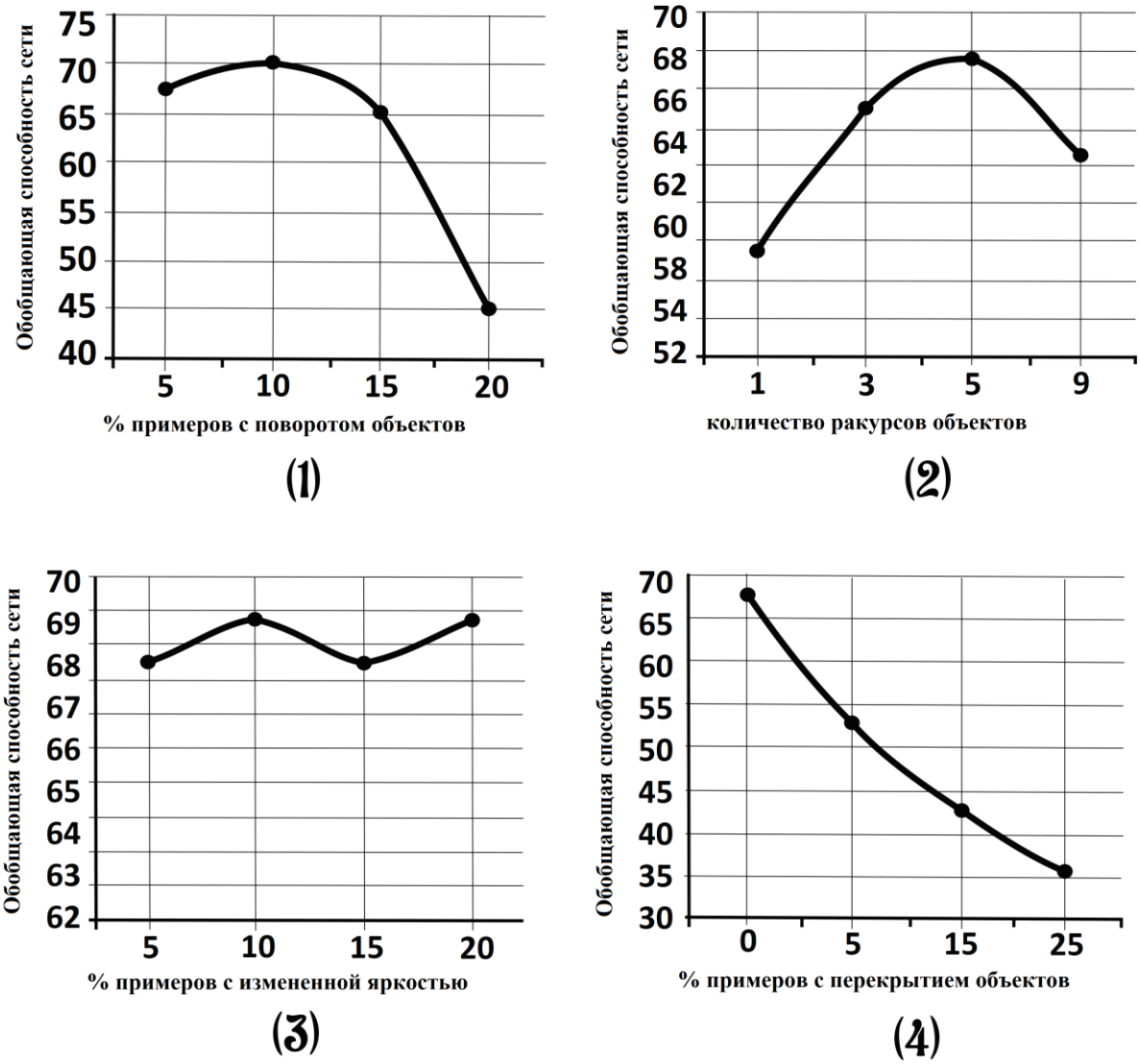


Рисунок 4.8 – Зависимость обобщающей способности сети от (1) поворота объекта вокруг своей оси (2) количество ракурсов исходного объекта (3) изменения яркости изображений (4) наличия частичного перекрытия объектов

Перекрытие объекта – еще один случай зашумление, при котором дополнительный объект не просто появляется на изображении, но

перекрывает образ целевого объекта. Целью такого изменения является попытка обучить сеть справляться с ситуациями, когда часть предмета перекрыта держащей его рукой или другой помехой. Обычно, это приводит к полной потере способности адекватно распознавать объект. Исследования показали, что «исчезновение» части объекта на изображении сильно мешает правильному обучению сети и не рекомендуется для использования при создании обучающего множества (рис. 4.8, табл. 4.2).

Было создано шесть обучающих множеств для каждого исследуемого параметра изображений выборок, с различными значениями параметров, которые детально приведены в таблице 4.2. В таблице 4.3 приведены данные об обобщающей способности нейронной сети для обучающих выборок с различными значениями исследуемых параметров. В таблице 4.2 приведены конкретные значения параметров, которые можно соотнести с данными из последующей таблицы. Параметр «насыщенность фона» не имеет ярко выраженного числового показателя, выборки в данном случае разделены на шесть условных классов в зависимости от: характера фона (простой однотонный фон или сложная сцена), сложности текстуры, количества объектов, составляющих фон и т.д.

Таблица 4.2 – Значения параметров обучающих выборок

№	Параметр изображений	Номер выборки					
		1	2	3	4	5	6
1	Размер объекта в пикселях	40	50	60	70	80	90
2	Насыщенность фона	От минимального значения до максимального					
3	Процент паттернов со сдвигом объекта	0	20	35	50	65	80
4	Процент паттернов с шумом	0	10	20	30	40	50
5	Количество ракурсов	1	2	3	5	7	9
6	Процент паттернов с поворотом объектов	5	10	12	15	18	20
7	Яркость	0	5	10	15	20	25
8	Перекрытие объекта	0	5	10	15	20	25

Таблица 4.3 – Динамика изменения обобщающей способности нейронной сети для различных параметров изображений обучающей выборки

№	Параметр изображений	Номер выборки					
		1	2	3	4	5	6
1	Размер объекта в пикселях	67,5	68	70	70,5	72,5	62,5
2	Насыщенность фона	72,5	71	67,5	67,5	66,5	65
3	Процент паттернов со сдвигом объекта	72,5	65	64	62,5	62	60
4	Процент паттернов с шумом	72,5	74	70	67	65	64
5	Количество ракурсов	57,5	58	65	67,5	64	62,5
6	Поворот вокруг оси	67,5	72,5	68	70	65	45
7	Яркость	72,5	67,5	68,7	67,5	67,5	68,7
8	Перекрытие объекта	72,5	67,5	52,5	42,5	35	35

Для финального теста была использована выборка, содержащая необработанные изображения тех же объектов в естественных условиях, без сгенерированных искусственно фонов (рис. 4.9). За счет повышения сложности задачи, процент правильно распознанных изображений сократился, но выявленные соотношения для каждого случая существенным образом не изменились, что подтверждает результаты экспериментов.

Рисунок 4.9 – Усложнение задачи использованием необработанных фотографий объектов

Таким образом, смещение объектов относительно центра обучающих паттернов не дает существенного улучшения качества распознавания, даже при условии наличия изображений со сдвигом объектов в тестовой выборке, в то время как размер изображения объектов относительно размера паттернов

напрямую влияет на качество распознавания и обобщающую способность сети. Оптимальная величина объекта находится в районе 80-85% от величины фона.

Усложнение и повышение насыщенности фона в общем случае снижает эффективность обучения сети, однако, наличие небольшого процента изображений с комплексным фоном, рекомендуется для решения сложных задач распознавания, а зашумление путем добавления «отвлекающих» объектов на изображение дает положительный эффект при небольшом проценте таких примеров (0-10 % всей обучающей выборки).

Количество используемых ракурсов объекта напрямую влияет на качество обучающей выборки и должно быть достаточным, но не чрезмерно большим, чтобы с одной стороны, обеспечивать необходимый уровень вариативности, но с другой, не приводить сеть к переобучению. Изменение яркости изображений, входящих в состав обучающей выборки, не приводит к серьезным изменениям качества распознавания нейронной сети при постоянном и достаточном освещении. Полное «удаление» части объекта путем перекрытия не приводит к положительным изменениям в распознавательной способности нейронной сети, по крайней мере, на небольших объемах обучающих выборок.

4.5 Экспериментальное исследование влияния фильтрации изображений обучающей выборки на обобщающую способность нейронной сети

Использование различных фильтров на изображениях обучающей выборки является распространенной практикой в области нейросетевого распознавания [58]. Применяется в первую очередь для того, чтобы осуществить нормализацию обучающих паттернов, а также выделить характерные черты, повысив «полезность» примеров для обучения. В рамках данного этапа исследований были проанализированы: фильтр Габора, фильтр

выделения краев (canny edge detection), фильтр Гауссова размытия, медианный фильтр и фильтр увеличения четкости.

Существует два способа фильтрации: использование фильтров напрямую и параллельная фильтрация [17], направленная на то, чтобы подчеркнуть и сделать более явными важные свойства изображения без сильных искажений. Процесс параллельной фильтрации включает в себя следующие этапы:

1. Создание копии исходного изображения.

2. Обработка определенным видом фильтра и получение сильно искаженного с точки зрения человеческого восприятия изображения.

3. При необходимости инвертирование и нормализация обработанных примеров.

4. Сложение исходного изображения с полученным в результате применения фильтра, выделение таким образом.

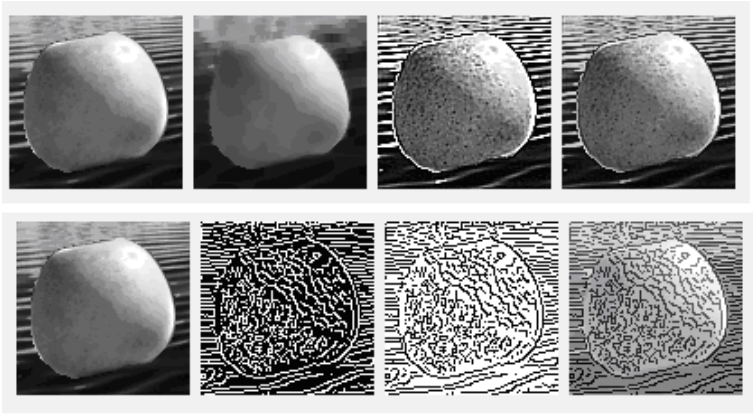


Рисунок 4.10 – Примеры параллельной фильтрации

Фильтр размытия по Гауссу. Вид импульсных фильтров с нормальным (Гауссовым) распределением [125], использующийся для размытия изображений. Фильтр основан на построение матрицы свертки, которая применяется к исходному изображению, при этом пиксели принимают меньшие веса в зависимости от их удаленности от центральной точки в пределах концентрической окружности.

Двумерный фильтр Гаусса описывается формулой:

$$G(u,v) = \frac{1}{2\pi\sigma^2} e^{-(u^2+v^2)/(2\sigma^2)}, \quad (4.4)$$

где u и v – переменные, отражающие радиус размытия для двух измерений $(u^2 + v^2 = r^2)$, а σ – коэффициент стандартного отклонения распределения по Гауссу.

Фильтр увеличения четкости. Относится к типу матричных фильтров, использующих матрицу свертки для обработки изображения. При этом матрица свертки имеет вид, показанный на рисунке 4.11 [139].

-1	-1	-1
-1	9	-1
-1	-1	-1

Рисунок 4.11 – Матрица свертки фильтра повышения четкости

Использование такого набора коэффициентов позволяет увеличить разницу значений на границах.

Медианный фильтр. Позволяет получить эффект размытия и сглаживания изображения [104]. Алгоритм действия фильтра заключается в том, что для каждого пикселя в пределах заданного окна находится среднее значение, которое и записывается в результирующее изображение. Чем больше размер окна, тем большую степень размытия даст применение фильтра. Под средним значением понимается не среднее арифметическое всех чисел, а значение срединного элемента в упорядоченном массиве, отражающем значения конкретного окна.

Фильтр Габора. Фильтр Габора – это мощный тип линейных фильтров, который часто используют для нахождения границ объектов и сегментирования изображений [50]. Математически двумерный фильтр Габора можно описать следующей формулой.

$$G(x,y) = K * \exp(-\tfrac{1}{2}(\frac{x^2}{a^2} + \frac{y^2}{b^2})). \quad (4.5)$$

Если принять интенсивность изображения в конкретной точке за I(x,y), а функцию Габора обозначить за G(i,j), то интенсивность нового изображения можно рассчитать по формуле:

$$I'(x,y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} I(x-\frac{n}{2}+i, y-\frac{n}{2}+j) * G(i,j). \qquad (4.6)$$

Фильтр Габора часто используют в различных системах компьютерного зрения и интеллектуальных системах. В подавляющем большинстве случаев целью использования данного типа фильтров является выделение наиболее значимых свойств изображения с одновременным редуцированием маловажных, не несущих полезной информации для распознавания, деталей.

Фильтр нахождения краев (Canny Edge Detection). Является сложным многоступенчатым фильтром, открытым Джоном Кэнни в 1986 году [80], и широко используется в области компьютерного зрения по сей день. Алгоритм обработки изображения с помощью данного фильтра включает в себя следующие этапы: сглаживание с целью удаления шума, поиск градиентов по четырем направлениям, подавление немаксимумов, пороговая фильтрация и трассировка области неоднозначности. Данный вид цифровых фильтров был подробно описан во второй главе.

Целью данного эксперимента является проверка того, может ли предобработка обучающей выборки, без использования экстремальных значений, привести к улучшению процесса обучения и качества распознавания нейронной сети на данных, которые не были подвергнуты фильтрации. Другими словами, можно ли с помощью фильтрации улучшить качество и сделать более явной хранимую в паттернах обучающей выборки информацию для обучения искусственной нейронной сети.

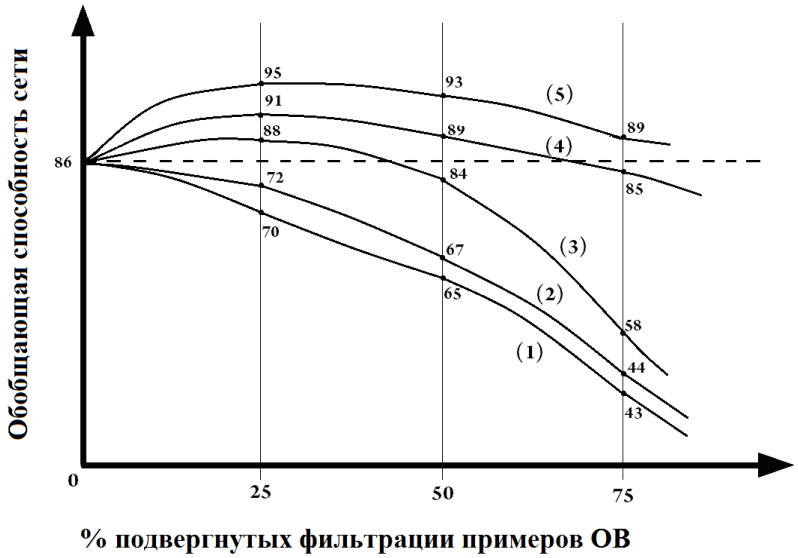


Рисунок 4.12 – Результат применения фильтров снизу-вверх (1) размытия (2) Габора (3) Медианный (4) Нахождение краев (5) Увеличения четкости

Исходным материалом являлась обучающая выборка с определенной известной степенью распознавания для выбранной модели нейронной сети. В ходе эксперимента к этой выборке поочередно применялся тот или иной тип фильтра, осуществлялось обучение сети и последующее сравнение качества распознавания с исходным значением. Для каждого фильтра рассчитывались три случая с различной степенью влияния на изображения обучающей выборки с помощью добавления дополнительных коэффициентов.

Как видно из графика, фильтры размытия по Гауссу и Габора приводят лишь к снижению процента правильно распознанных образов, вследствие того, что они оказывают довольно специфическое воздействие на изображения, не подчеркивают полезные для обучения свойства, а вносят разрушающие искажения. Гораздо лучшие результаты показал медианный фильтр, с помощью применения которого удалось добиться небольшого улучшения обобщающей способности сети при минимальном воздействии процесса фильтрации на исходные изображения. Однако, если продолжать усиливать влияние обработки, процент правильно распознанных примеров сначала незначительно, затем более стремительно падает, происходит

излишне сильное размытие исходных данных, которое мешает сети извлечь необходимое количество полезной информации. Наконец, наилучший положительный эффект на процессы обучения и распознавания оказали фильтры выделения краев и повышения четкости изображений. Кроме усиления краевых областей, как очень важной характеристики объектов с точки зрения нейронной сети, действие фильтров приводит также к уменьшению разброса значений и нормализации примеров визуальной обучающей выборки.

4.6 Экспериментальное исследование влияния расширения обучающей выборки за счет деформации изображений на обобщающую способность нейронной сети

Одним из способов улучшения обучения нейронной сети и качества классификации визуальных образов является увеличение обучающей выборки и общего количества примеров для обучения [114]. Этого можно добиться простым экстенсивным увеличением количества фонов или примеров того или иного объекта. Но также разнообразие и полноту обучающей выборки можно расширить с помощью искусственной деформации уже содержащихся в ней изображений.

Эластичные деформации – метод обработки изображений, который облегчает генерацию новых данных для обучения классификатора [86]. Идея данного подхода заключается в том, чтобы сдвинуть каждую точку изображения таким образом, чтобы новые координаты не были абсолютно одинаковы, и в то же время не были случайными значениями, разрушающими исходное изображение. Другими словами, соседние точки должны смещаться в похожем направлении, для чего используется непрерывное гауссово ядро, на основании которого создаются два массива со значениями смещений соответственно по оси X и по оси Y.

Аффинные преобразования – вид отображения пространства на себя, при котором сначала выбирается новый базис пространства, а затем для каждой точки рассчитывается положение в новой системе координат [127].

Существует устоявшаяся практика использования эластичных и аффинных преобразования при работе с базами данных рукописных символов (например, MNIST [118]). На подобных выборках метод способен показать хорошие результаты, как описано, например, в [85].

Цель данного эксперимента заключалась в том, можно ли применить аффинные или эластичные преобразования при работе с задачей классификации реальных объектов.

В случае работы с рукописными символами, эластичные деформации изображений обучающей выборки приводят к меньшей средней квадратичной ошибке, чем применение аффинных преобразований [117]. Эксперимент с визуальными объектами показал обратный результат – аффинные преобразования оказывают лучшее влияние на качество обучающей выборки, чем эластичные деформации. То, что хорошо подходит для простых линий, когда несколько иной изгиб линии может быть рассмотрен как новое написание символа, хуже работает на сложных объектах, имеющих замкнутый контур, объем и сложное соотношение областей с различной интенсивностью цвета.

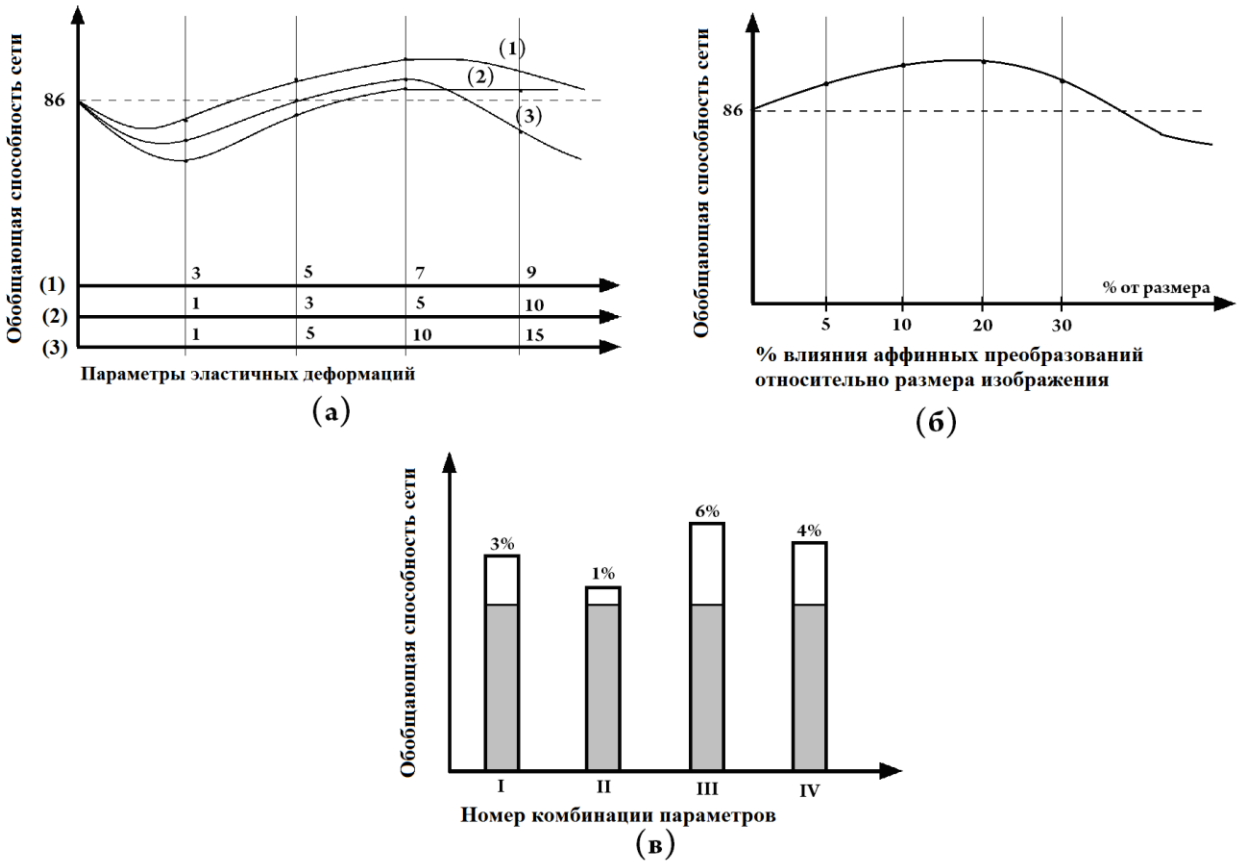


Рисунок 4.13 – Влияние на обобщающую способность нейронной сети (а) различных параметров эластичных деформаций (б) применения аффинных преобразований (в) сравнение улучшения качества обучения различных методов деформации изображений объектов и их комбинаций

На характер эластичных деформаций напрямую влияют размер ядра, величина отклонения и интенсивность. На первом этапе были найдены оптимальные значения данных параметров, оказывающие максимально полезное влияние на процесс обучения. На рисунке 4.13 (а) приведены результаты экспериментов для (1) размера ядра, (2) значения рандомизации (3) интенсивности воздействия преобразований. Эксперимент показал, что наилучший эффект достигается при значениях размера ядра 7x7 точек, отклонения, равного 5, и значения интенсивности от 10 и выше.

В выбранной реализации аффинных преобразований деформация задавалась с помощью двух матриц, соответствующих угловым точкам исходного и обработанного изображений. Для генерации примеров

обучающей выборки вторая матрица заполнялась с помощью функции генерации случайных значений в определенном диапазоне, единственным параметром которой являлся допустимый предел, на который новые координаты могут отличаться от исходных значений. На рисунке 4.13 (б) показан график изменения качества распознавания в зависимости от величины смещения в процентах от размеров изображения. Максимальной отметки график достигает при значении 15-20 %.

Помимо вышеописанных видов генерации новых изображений, проверке также подверглись комбинации этих подходов. Рисунок 4.13 (в) иллюстрирует улучшение обобщающей способности сети после обучения с применением I аффинных преобразований II эластичных деформаций III совместного использования подходов для разных примеров IV последовательной обработки одних и тех же изображений сначала одним, потом другим методом. Наилучшего синергического воздействия удалось достигнуть при использовании третьего варианта генерации новых примеров.

4.7 Исследование влияния освещения на качество распознавания

Важным фактором, влияющим на результат работы распознающей системы является освещение сцены, содержащей объект, особенно изменение этого параметра во времени [134]. Крайне критичным становится влияние освещения для систем, работающих с полутоновыми изображениями, в которых матрица входных данных кодируется с использованием только одного параметра интенсивности.

При таком подходе, различие между цветами заключается в изменении интенсивности точек изображения также как и различие между освещенностью одного цвета. Один и тот же объект может быть воспринят системой абсолютно по-разному, в зависимости от того меньшее или большее количество солнечного света падает на него в текущий момент.

На рисунке 4.14 приведен пример одного и того же полутонового изображения с измененной яркостью. Для человека не возникает трудности связать все примеры одним понятием, но для распознающей системы каждый из них кодируется совершенно разными числовыми значениями.

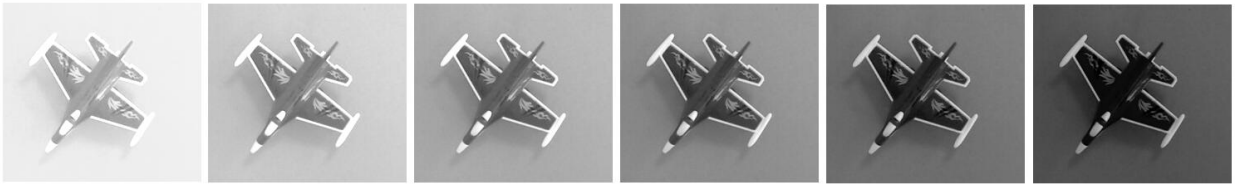


Рисунок 4.14 – Пример изменения параметра освещенности для конкретного изображения

В таблице 4.4 приведены результаты того, как изменение степени освещенности помещения, в котором проводится эксперимент, влияет на качество распознавания образов нейронной сетью. 10-20 тысяч люкс соответствуют светлому солнечному дню, с открытыми окнами, без прямых солнечных лучей, 1-6 люкс – вечернее время, искусственное освещение выключено, на окнах шторы, 0.1-0.7 люкс – поздний вечер, искусственное освещение выключено, окна завешены шторами. Таблица содержит данные для стандартной выборки и выборки, расширенной за счет изменения параметра интенсивности входящих в нее изображений.

В первом случае, высокое качество распознавания нейронная сеть показывает только при высокой степени освещенности, при 6-10 люкс процент правильно распознанных образов падает в 1.5-2 раза, при 1-6 люкс снижается до 10%, при освещенности менее 1 люкса корректного распознавания почти не происходит.

Во втором случае, при нормальных условиях проблем с распознаванием не возникает, при существенном снижении яркости освещения процент правильно распознанных объектов падает до 25-40%, как при добавлении цифрового шума, распознаются только крупные объекты, при освещенности менее 1 люкс лишь иногда происходит распознавание крупных объектов.

Таблица 4.4 – Влияние освещенности на качество распознавания нейронной сети

Уровень освещённости		10-20 тысяч люкс	6-10 тысяч люкс	6 люкс, 2 люкс, 1 люкс	0.7 люкс, 0.2 люкс, 0.1 люкс
Процент среднего распознавания для всех классов	Стандартная выборка	95%	45-60%	4-10%	0%
	Расширенная выборка	95%-99%	60-75%	25%–40%	3-15%

Таким образом, расширение обучающей выборки за счет включения примеров с измененными параметрами интенсивности положительно влияет на качество распознавания нейронной сети при различных внешних условиях освещения и повышает стабильность работы распознающей системы.

На основе всех описанных в данной главе параметров, способов создания обучающих выборок, методов применения фильтрации, аффинных преобразований и эластичных искажений, а также влияния параметра яркости на качество обучения была составлена методика создания эффективных обучающих выборок для нейронных сетей (см. приложение А).

Выводы

1. Разработана методика создания визуальных обучающих выборок для нейронных сетей. Разработанная методика включает шаги: выбор объектов для распознавания, создание коллекции фонов для обучающей выборки, создание фотографий выбранных объектов, полуавтоматическое выделение областей объектов из полученных фотографий, интерполяция изображений до необходимых размеров и перевод в оттенки серого, подготовка изображений для пустого класса, наложение всех имеющихся объектов на все фоны, запись в бинарный файл, интерполяция полученных изображений до размеров 96x96 пикселей, внесение дополнительного разнообразия в выборку за счет изменения освещенности всех полученных примеров, опциональное использование фильтров повышения четкости или выделения границ, опциональное расширение выборки за счет использования аффинных преобразований и эластичных искажений, запись файлов меток и информационного файла. Выполнена третья подзадача диссертационного исследования.

2. Экспериментальным путем определен оптимальный способ создания изображений визуальных обучающих выборок для нейронных сетей – автоматизированное формирование паттернов за счет наложения предобработанных визуальных данных, описывающих объекты, на набор фоновых изображений.

3. Показано, что оптимальным фоном для примеров обучающей выборки являются сцены реального мира, включая фотографии окружения, в котором предполагается проводить распознавание.

4. Доказано, что оптимальная величина объекта находится в районе 80-85% от величины фона, количество используемых ракурсов напрямую влияет на качество обучающей выборки. Предпочтительно избегать сдвига объектов относительно центра изображения и удаления или перекрытия части объекта.

5. Выбраны оптимальные параметры фильтрации, эластичных деформаций и аффинных преобразований для использования в разработанной методике создания обучающих выборок, составлены рекомендации по использованию фильтров и искажений при формировании обучающего множества.

6. Доказана важность влияния освещенности на обобщающую способность нейронной сети. Обоснована возможность расширения обучающей выборки за счет изменения интенсивности входящих в нее примеров.

ГЛАВА 5 ТЕСТИРОВАНИЕ И ЭКСПЕРИМЕНТАЛЬНАЯ ОЦЕНКА РАЗРАБОТАННЫХ АЛГОРИТМОВ РАСПОЗНАВАНИЯ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ

Данная глава содержит описание разработанного программно-аппаратного комплекса для выделения и распознавания объектов на изображениях, а также результатов экспериментов по тестированию разработанного метода выделения и распознавания объектов на изображениях, разработанного численного метода отсеивания гипотез по низкочастотной структуре, разработанного параллельного алгоритма обработки данных в СНС второго порядка и разработанной методики создания визуальных обучающих выборок.

5.1 Разработка программного комплекса для выделения и распознавания объектов

Для проведения диссертационного исследования был разработан программный комплекс для выделения и распознавания объектов, который включает в себя модуль создания визуальных обучающих выборок, модуль генерации гипотез на основе селективного поиска с отсеиванием гипотез по низкочастотной структуре и цвету, модуль распознавания образов, базирующийся на СНС второго порядка с возможностью параллельной обработки данных на векторно-матричном процессоре.

Модуль генерации гипотез и СНС второго порядка описаны во второй главе данного диссертационного исследования, параллельный алгоритм обработки данных в СНС второго порядка с применением процессоров векторно-матричной архитектуры описан в третьей главе, методика полуавтоматического создания визуальных обучающих выборок, лежащая в основе соответствующего модуля, описана в четвертой главе. Обобщенная схема разработанного программного комплекса представлена на рисунке 5.1.

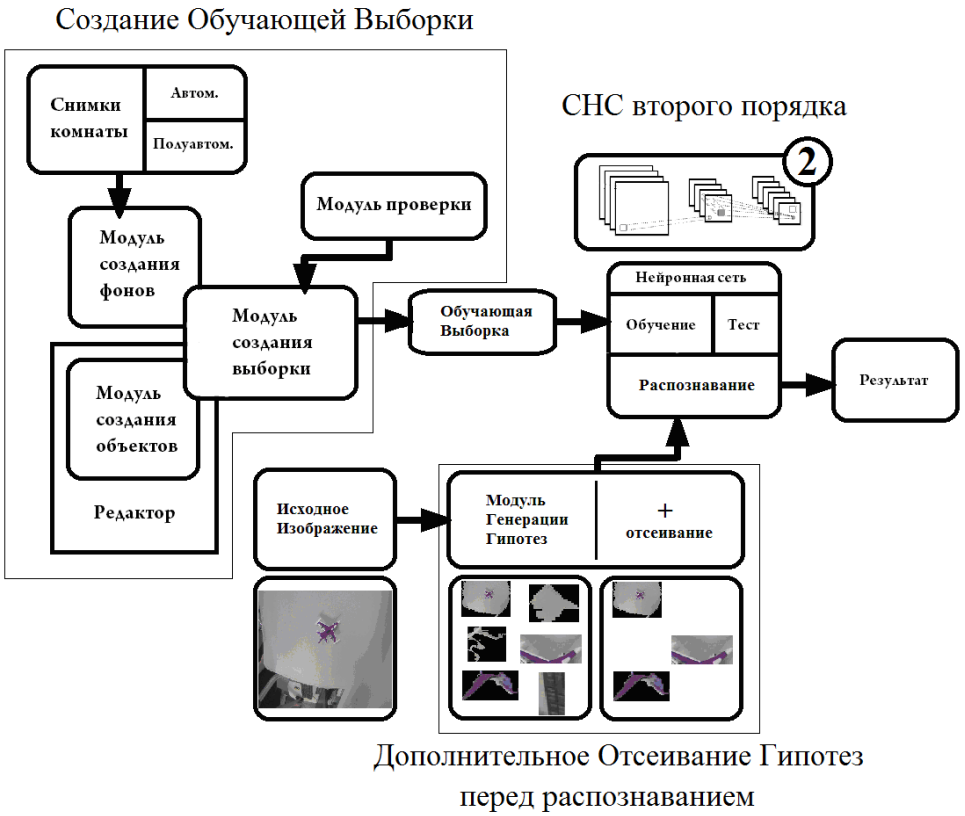


Рисунок 5.1 – Обобщенная схема программного комплекса

Важной частью разработанного программного комплекса является модуль создания визуальных обучающих выборок, позволяющий выделять и сохранять объекты из кадров видео потока данных, автоматизировать процесс наложения с применением различных параметров этих объектов на заранее подготовленные фоны, а также кодировать полученные изображения в формат обучающей выборки (рис. 5.2) [35].

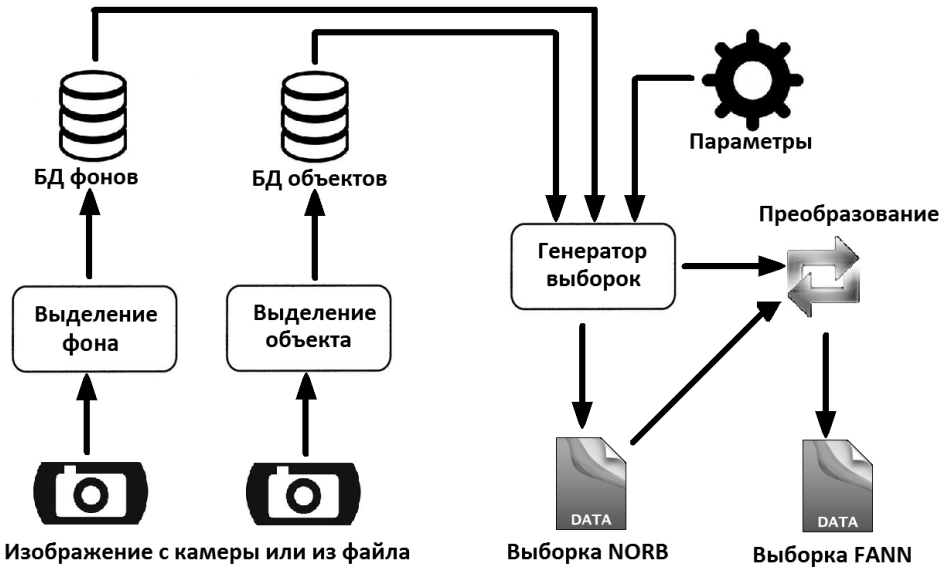


Рисунок 5.2 – Логическая структура программы-генератора обучающих выборок для нейронных сетей

Одной из составных частей программного комплекса является модуль создания визуальных обучающих выборок для нейронных сетей. Задача модуля – предоставить возможность подготовки исходного материала и полуавтоматического создания обучающих выборок больших размеров, с использованием оптимального алгоритма генерации массива изображений для наилучшего соответствия размеров, разнообразия и полноты выборки для конкретной задачи распознавания образов.

Обучающая выборка содержит в себе всю информацию, которая будет доступна нейронной сети для решения задачи распознавания, никаких других источников данных сеть не имеет, поэтому качество обучающей выборки, как проекции отдельной области реального мира, является критично важным фактором успешного обучения [113]. Количество информации, содержащейся во множестве обучающих примеров, должно быть необходимым и достаточным для того, чтобы отразить в себе все важные данные об объектах, без которых сам процесс распознавания был бы невозможен, с другой – не содержать излишней, не относящейся непосредственно к решаемой задаче, информации. Любые данные, которые не имеют смысла в контексте конкретной задачи, считаются шумом. Таким

образом, даже если использовать современные мощные математические модели, если не удастся обеспечить достаточно высокое качество формирования обучающей выборки, правильное распознавание будет в принципе невозможно.

Основная идея заключается в том, чтобы выделить определенное количество объектов из снимков в виде числовых массивов интенсивности в каждом пикселе, а также маски, задающей форму объекта, масштабировать полученные ракурсы и наложить их поочередно на каждый из подготовленных заранее фонов. Таким образом, получается выборка больших размеров, содержащая множество полезной разнообразной информации, на создание которой уходит значительно меньшее время, чем при ручном формировании множества примеров.

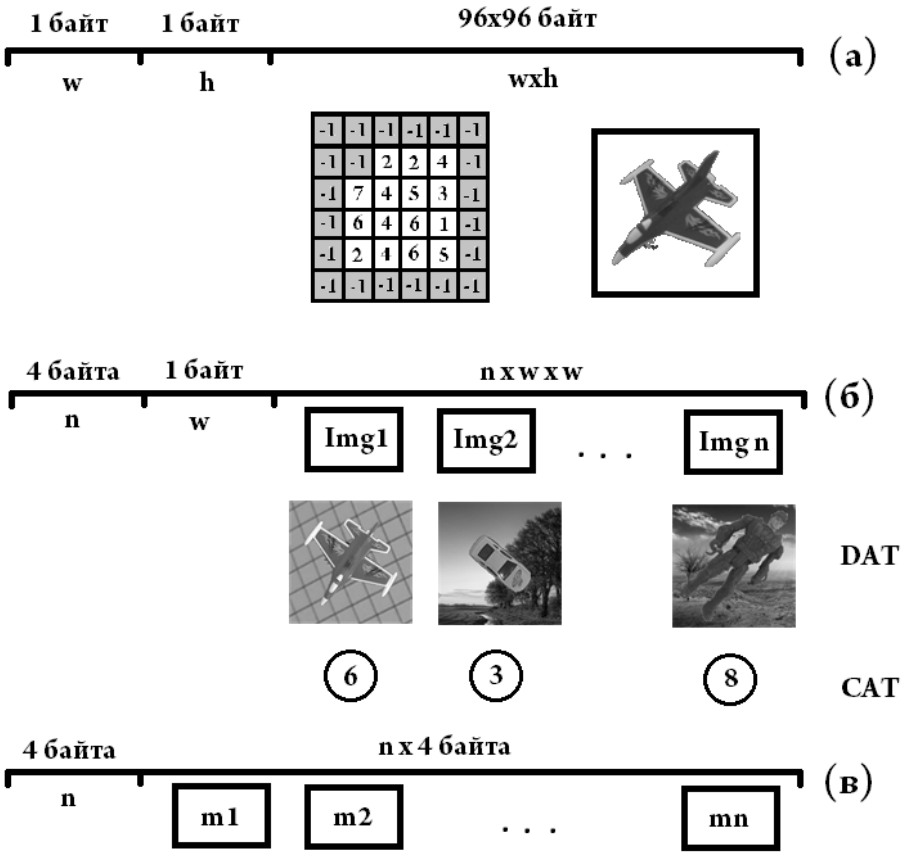


Рисунок 5.3 – (а) формат файлов объектов (б) формат файлов обучающей выборки (в) формат файлов меток

Модуль разделен на несколько частей: модуль выделения и сохранения фонов, модуль выделения и сохранения объектов, модуль генерации обучающих выборок (рис. 5.2). Каждая часть отвечает за определенную подзадачу, которую необходимо решить в процессе формирования обучающего множества.

Для создания обучающей выборки необходимо проделать следующие шаги:

1) Сделать необходимое количество снимков объекта каждого класса с разных ракурсов, выделить и сохранить объекты, изображенные на снимках, задав требуемое количество и степень масштабирования.

2) Создать коллекцию фонов из снимков реального мира, паков текстур и любых других изображений.

3) Выбрать объекты и фоны и создать на их основе обучающую выборку.

Обобщенный алгоритм создания обучающей выборки выглядит следующим образом:

1. Сформировать массивы исходных данных.

1.1. На основании имен выбранных классов сформировать полный список используемых объектов и массив изображений.

1.2. На основании имен выбранных фонов сформировать массив фоновых изображений.

1.3. Сформировать массив изображений пустого класса.

2. Определить переменные количества.

2.1. NN1 := количество объектов.

2.2. NN2 := количество фонов.

2.3. Ne := количество изображений пустого класса.

2.4. Размер выборки NN1 * NN2 + NN2 + Ne + 1.

3. Сформировать результирующие изображения (заполнить результирующее изображение фоном, наложить сверху объект в соответствие с маской, добавить пустой класс).

4. Сформировать файлы выборки.

Объекты и фоны хранятся в виде текстовых файлов, где первые два байта содержат параметры длины и высоты кадра соответственно, а последующее место занимает матрица размерностью w x h (рис. 5.3а) [124]. Маска хранится вместе с самим объектом, и нужна для того, чтобы задавать его форму. Формируемая выборка соответствует формату NORB и хранится в двух файлах с расширениями «.cat» и «.dat». Первый файл содержит набор чисел, характеризующих класс объекта, находящегося на определенной позиции в выборке (рис. 5.3в). Во втором файле хранится набор одинаковых по размеру матриц – примеров для обучения (рис. 5.3б).

Среди основных функций, не затрагивающих работу с интерфейсом модуля, можно выделить следующие:

– Функции записи и чтения изображения в бинарный файл,

– Функции изменения размера объекта и фона,

– Функция автоматического выделения объекта,

– Функция сохранения объекта,

– Функция сохранения фона,

– Функция создания выборки.

Созданный программный комплекс обладает следующими особенностями:

1. Основной формат выборок соответствует формату NORB.

2. Модуль позволяет создавать в полуавтоматическом режиме большие визуальные обучающие выборки, существенно сокращая затрачиваемое на их создание время.

3. Модуль разделен на три отдельных логических подмодуля: выделение и сохранение объектов, выделение и сохранение фонов и создание обучающей выборки.

4. Разработанный модуль является важной частью комплекса распознавания изображений, т.к. качественное обучение нейронной сети

является ключевым этапом ее использования и определяет качество распознавания образов [16].

5.2 Тестирование разработанного численного метода отсеивания гипотез по низкочастотной структуре

Разработанный численный метод отсеивания гипотез необходим для того, чтобы снизить время выделения и распознавания объектов за счет уменьшения количества гипотез до распознавания с помощью простых и быстрых алгоритмов. Это связано с тем, что внедрение нейронов высокого порядка в СНС увеличивает алгоритмическую сложность и время обработки одного вектора. При распознавании объектов может быть сгенерировано от 300 до нескольких тысяч векторов, которые в сумме требуют довольно много времени на обработку.

Для того, чтобы проверить на сколько изменяется время обработки одного изображения после внедрения нейронов второго порядка в СНС и использования численного метода отсеивания гипотез, был проведен эксперимент. В таблице 5.2 показаны более точные относительные показатели среднего ускорения и среднего количества генерируемых гипотез по сравнению с показателями стандартного алгоритма для изображений различных размеров. Исходя из этих данных, можно сделать вывод, что использование численного метода отсеивания гипотез позволяет для изображения размером 640x480 пикселей снизить число обрабатываемых нейронной сетью гипотез на 16-21%, а время, требуемое на их обработку, снизить на 18-22%. При увеличении размера изображения среднее снижение количества генерируемых гипотез увеличивается до 18-26%. Среднее снижение количества генерируемых гипотез составляет 16%.

Таблица 5.2 – Среднее ускорение и среднее снижение количества гипотез для разных размеров изображений

Размер изображения	Среднее ускорение	Среднее снижение количества генерируемых гипотез
320x240	5 – 8 %	8 – 14 %
640x480	18 – 20 %	16 – 21 %
1024x720	16 – 22 %	18 – 26 %

В таблице 5.3 приведены данные, позволяющие сравнить насколько изменяется обобщающая способность СНС при использовании нейронов второго порядка, а также средняя скорость выделения и распознавания объектов при использовании численного метода отсеивания гипотез. Рассматривается три случая: классический вариант СНС и селективного поиска, СНС с нейронами второго порядка без отсеивания гипотез, а также с использование численного метода [32].

Таблица 5.3 – Среднее время, затрачиваемое на выделение и распознавание объектов, и обобщающая способность нейронной сети для разных случаев

Алгоритм	СНС	СНС 2-го порядка	СНС 2-го порядка + отсеивание
Среднее время выполнения (с)	15	16,5	13,5
Средняя обобщающая способность сети (%)	84,5	89	89

Из таблицы видно, что после внедрения нейронов второго порядка в целом на полторы секунды увеличилось время обработки, но обобщающая способность сети возросла до 89% за счет усложнения ее структуры. В то же время, использование численного метода отсеивания гипотез, хотя и требует дополнительного времени на выполнение, в целом сокращает общее время выделения и распознавания объектов по сравнению со стандартным подходом. При этом обобщающая способность нейронной сети не падает и остается такой же, как во втором варианте.

Объясняется это тем, что среднее время одной обработки данных на 1000 проходов в СНС равно 31 мс, а в СНС 2-го порядка – 33 мс. При этом, при использовании численного метода отсеивания гипотез, векторов для обработки генерируется в среднем на 20 % меньше. При этом, на дополнительное отсеивание векторов в среднем затрачивается не более 0,5 с, поэтому в результате экономия времени в среднем равна 1,5 секунды, а обобщающая способность увеличивается на 4,5 % за счет использования нейронов второго порядка.

Таким образом, разработанный численный метод дает существенную прибавку в скорости выделения и распознавания объектов, без потерь в точности. Отсеивание от 5 до 20 % генерируемых гипотез о местоположении объекта никак не снижает точность выделения и обобщающую способность нейронной сети.

5.3 Тестирование производительности разработанного параллельного алгоритма

Следующий этап эксперимента – исследование того, насколько повышается скорость обработки данных в СНС второго порядка при использовании разработанного параллельного алгоритма.

Для тестирования производительности разработанного параллельного алгоритма обработки данных в СНС второго порядка были разработаны модули на языках программирования C++ и Ассемблера для процессоров NeuroMatrix. Так как целью эксперимента был исключительно тест производительности, необходимыми и достаточными в данном случае являлись непосредственная передача данных и функция прямого прохода сети [66]. В процессе обучения прямой проход используется множество десятков и даже сотен тысяч раз. Таким образом, даже небольшой выигрыш в производительности отдельного прохода может дать существенную прибавку в скорости в контексте всего процесса обучения.

В таблице 5.4 приведены данные о скорости выполнения алгоритмов нейронных сетей различных архитектур на векторно-матричном процессоре, а также число эквивалентных операций умножения и время выполнения в тактах процессора. Используются полносвязные нейронные сети с 512 и 1024 нейронами на скрытом слое [20], сверточные нейронные сети первого и второго порядка с использованием нейронов второго порядка на одном и двух слоях сети.

Таблица 5.4 – Замеры производительности для разных архитектур нейронных сетей

Значение	Число эквивалентных операций умножения элементов	Время выполнения в тактах процессора	Время выполнения в миллисекундах
512-NN	12288	13150	0,33 мсек
1024-NN	24576	25800	0,65 мсек
CNN	2526308	450587	17,84 мсек
CNN^2 1 слой	5911908	1054437	26,06 мсек
CNN^2 2 слоя	6995300	1247669	29,7 мсек

При использовании сравнительно небольшой выборки размером 24 000 обучающих примеров и 8-ми эпох обучения, только на прямые проходы сети потребуется 24 000 * 8 * 0,068 = 13056 секунд = 217,6 минут = 3,63 часов. Если же нейронная сеть обучается на выборке NORB, содержащей 291 600 снятых с двух камер примеров, то время обработки возрастает до 291 600 * 8 * 2 * 0,068 = 317260,8 секунд = 5287,68 минут = 88,128 часов.

Для стандартной сети ускорение прямого прохода за счет использования векторно-матричной архитектуры составляет примерно 6%, для сверточной – 12% [33].

Конечно, оценка ускорения сети с помощью векторных процессоров не может быть однозначной, для разных процессоров, имеющих разные характеристики, время выполнения единичного цикла обучения будет

различным. Поэтому сравнение проводилось с конкретным процессором Intel Core i3-2130, CPU = 3,50 GHz.

Таблица 5.5 – Сравнение производительности для разных размеров выборок

	Процессор	NM	Ускорение
1 проход	0,075 с	0,067 с	0,008 с
1000 проходов	75 с	67 с	8 с
Полное обучение	240 мин	214,5 мин	25,6 мин
NORB	5832 мин	5209 мин	622 мин

Для того чтобы посчитать количество эквивалентных операций умножения для классической нейронной сети с одним скрытым слоем, достаточно умножить число входов на число нейронов скрытого слоя и сложить с результатом аналогичной операции для выходов сети.

$$N_{multiple} = N_{input} \cdot N_{hidden} + N_{hidden} \cdot N_{output} \quad (5.1)$$

В рассматриваемом случае сеть содержала 12 входов, 12 выходов и 512 (либо 1024) нейронов в скрытом слое.

В случае сверточных нейронных сетей вместе с усложнением архитектуры сети, усложняется и вычисление количества операций умножения [78].

Для того чтобы получить общее количество операций умножения, необходимо сложить эквивалентные величины для всех сверточных и субдискретизирующих слоев сети, а также выходного слоя:

$$N_{multiple} = N_{c1} + N_{s1} + N_{c2} + N_{s2} + N_{c3} + N_{out}. \quad (5.2)$$

Для первого сверточного слоя количество операций умножения равно произведению числа всех возможных позиций плавающего окна относительно вектора входного изображения на размер и количество ядер сверточного слоя:

$$N_{c1} = N_{corec1}^2 \cdot \left(n_{input} - N_{corec1} + 1\right)^2 \cdot N_{coresC1}, \quad (5.3)$$

где N_{corec1} – размер ядра C1-слоя, $N_{input} = n_{input} \cdot n_{input}$ – размер входов сети, $N_{coresC1}$ – количество ядер в слое.

Для слоя S1 формула будет выглядеть следующим образом:

$$N_{s1} = N_{coresS1} \cdot N_{coresC1} \cdot N_{cores1}^2 \cdot N_{mapS1}^2, \quad (5.4)$$

где $N_{coresS1}$ – количество ядер в S1-слое, $N_{coresC1}$ – количество ядер предыдущего слоя, N_{cores1} – размер ядра S1-слоя, n_{c1} – размер карты предыдущего слоя.

Для последующих слоев (C2, S2, C3) используются аналогичные формулы. Для выходного слоя вычисления будут сложи с алгоритмом для выходного слоя классической сети:

$$N_{out} = N_{coresC3} \cdot n_{c3}^2 \cdot n_{output}. \quad (5.5)$$

Если в архитектуре сети присутствуют слои второго порядка, то для каждого такого слоя количество операций нужно умножить на три, для третьего порядка – на четыре и т.д.

Таблица 5.6 – Структура СНС архитектуры «LeNet-5»

Слой	Размер карт	Количество карт	Размер ядер
Вход 96x96			
C1	92x92	8	5x5
S1	23x23	8	4x4
C2	18x18	24	6x6
S2	6x6	24	3x3
C3	1x1	100	6x6
Выход 5			

Модули обратного прохода имеют еще большую алгоритмическую сложность по сравнению с прямым проходом [100], поэтому их выполнение занимает большее время и вносит более существенный вклад в ускорение процесса обучения нейронной сети.

Важно отметить, что процессоры NeuroMatrix предусматривают также параллельное соединение нескольких модулей в одну вычислительную структуру, за счет чего обработка распределяется между платами и скорость

вычислений возрастает в разы [42]. Также возможно еще большее ускорение за счет использования в некоторых частях программы директив .branch и .wait, которые позволяют включать режим параллельного выполнения инструкций процессора [46]. Процессоры NeuroMatrix имеют два устройства генерации адресов, за счет чего возможна адресаций сразу по двум адресам во внешней памяти. Адресные регистры разделены на две группы: ar0 – ar3 и ar4 – ar7, необходимым условием является использование параллельно выполняющимися командами регистрами из разных групп.

Таким образом, векторно-матричная архитектура процессоров NeuroMatrix при правильном применении может давать увеличение производительности при реализации алгоритмов нейросетевого типа. Во многом этому способствует то, что структура нейронных сетей в большинстве случаев хорошо ложится на архитектуру процессоров, которая включает в себя аппаратную реализацию функции нелинейного преобразования, ориентацию на работу с векторными массивами, а также операцию взвешенного суммирования в качестве своей основы. Данная операция является краеугольным камнем нейросетевых алгоритмов и широко в них используется [23].

Таблица 5.7 – Сравнение ускорения для разных размеров выборок

Примеров	24 000	100 000	NORB (291 600)	Среднее ускорение
1 эпоха	10,5 %	11,5 %	10 %	11 %
2 эпохи	9,5 %	10 %	10,5 %	10 %
8 эпох	10 %	10,5 %	11 %	10,5 %

Использование процессоров векторно-матричной архитектуры для реализации нейронных сетей может дать прибавку в скорости выполнения алгоритма в среднем 10,5%. В таблице 5.7 приведены данные об ускорении для выборок разных размеров для одной, двух и восьми эпох обучения, а также средние показатели ускорения.

5.4 Тестирование разработанной методики создания выборок

5.4.1 Описание созданных для эксперимента обучающих множеств

Набор обучающих данных имеет большое значение для любой системы, основанной на искусственных нейронных сетях. Целью данного эксперимента было тестирование созданного программного комплекса для распознавания объектов десяти классов при помощи сверточной нейронной сети второго порядка, а также разработанной методики создания эффективных обучающих множеств. Для этого было создано *три типа обучающих выборок*, в различной степени соответствующие разработанной методике.

Базовые параметры включали десять ракурсов каждого объекта и двадцать пять фонов. При этом второй и третий тип обучающих выборок в большей степени соответствовали разработанной методике создания обучающих множеств, что привело к большей эффективности и лучшим результатам распознавания образов нейронной сетью.

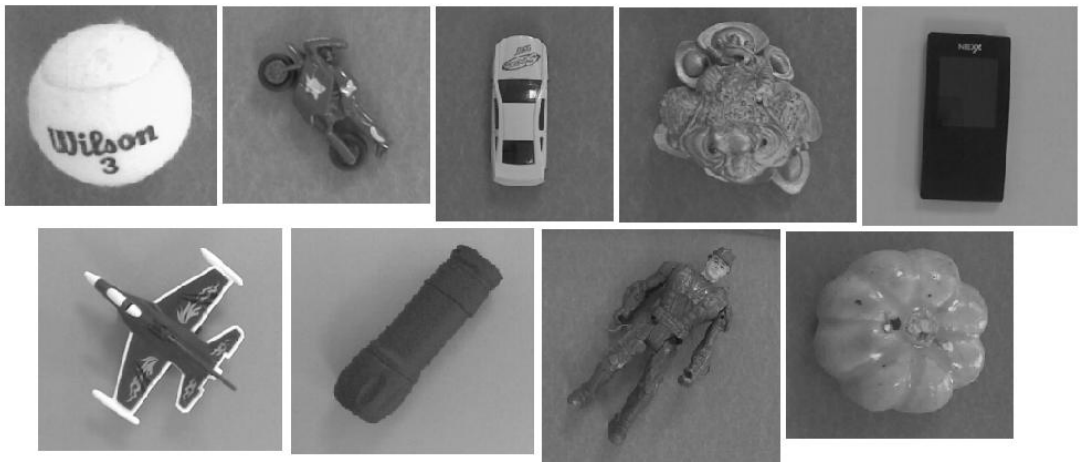


Рисунок 5.4 – Предметы, использованные для создания обучающего множества

Обучающие выборки первого типа содержали 5 сложных и 5 простых объектов. На каждый объект приходилось по 8 ракурсов и 4 точки отдаления,

всего 32 состояния. Объекты накладывались на фоны, которые составляли изображения нескольких типов: мебель, стены и обои, одежда. Кроме этого, был принят во внимание тот факт, что недостаточное качество изображений обучающей выборки может сказаться на процессе обучения и распознавания за счет того, что данные для распознавания отличаются от примеров, использованных при обучении нейронной сети.

Для создания *обучающих множеств второго и третьего типов* использовалось большее количество пунктов разработанной методики. Главное изменение заключалось в том, что в выборки был добавлен «пустой» класс, то есть изображения, на которых не содержится ни один из предметов, не относящиеся ни к одному из классов. Такое решение было принято, потому что нейронная сеть, обученная на стандартной выборке, всегда пытается отнести к определенному классу любое изображение, поступающее на вход, даже если на нем не присутствует искомый объект.

На каждый объект приходилось 8 ракурсов и 3 степени отдаления, всего 24 изображения. Коллекция фонов для формирования выборок включала следующие множества:

1. 25 фонов из предыдущей выборки.

2. 25 дополнительных текстур.

3. 10 основных цветов, но не простого цифрового заполнения фона пикселям и определенных характеристик, а фотографии однотонных цветных поверхностей, имеющие естественную неравномерность.

4. 25 текстур комнаты, в которой проводились исследования.

5. 15 текстур одежды.

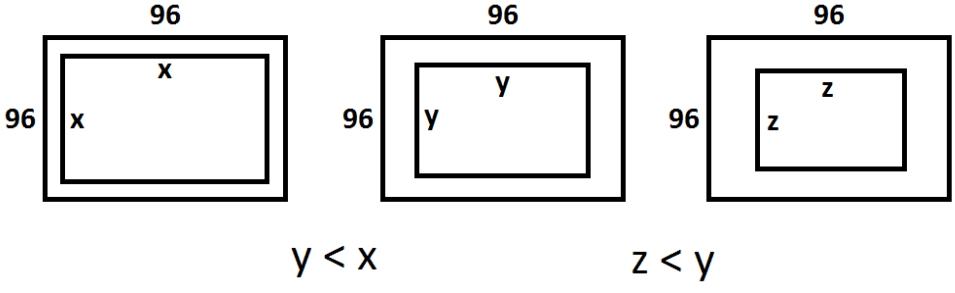


Рисунок 5.5 – Степени отдаления объектов

Количество ракурсов для каждого класса объектов равнялось восьми и было фиксировано. Каждый из восьми ракурсов имел три степени отдаления. Таким образом, общее количество состояний для каждого класса равнялось двадцати четырем. Количество фонов также было увеличено до ста. Помимо текстур и сцен разной степени сложности, в новую выборку было добавлено большое количество фоновых изображений, выделенных из той среды, в которой предполагалось использование комплекса распознавания. Набор классов объектов был дополнен «пустым» классом, не содержащим никаких объектов, но имеющим собственный идентификатор.

Таким образом, выборки содержали 2400 примеров для каждого из десяти классов, 900 примеров для пустого класса, общий объем равнялся 24 900. В качестве фоновых изображений было использовано 10 основных цветов, 50 текстур и 40 снимков рабочего пространства кабинета.

В выборках *третьего типа* для большей стабильности распознавания при смене освещенности рабочей сцены в коллекцию изображений объектов были добавлены снимки при трех разных вариантах освещения, а каждый пример обучающей выборки был подвергнут преобразованию с изменением параметра интенсивности каждого пикселя изображения и применением пороговой функции. Использованные значения параметра изменения интенсивности:-20, 0, 20, 40.

5.4.2 Тестирование методики создания обучающих выборок

Эксперимент включал в себя следующие шаги: на предварительном этапе был составлен список из десяти классов объектов, участвующих в эксперименте. Приоритет был отдан объектам, имеющим существенные отличия друг от друга, а также цветовую однородность для облегчения процесса сегментации. После этого с помощью редактора, входящего в состав программного комплекса, были сохранены необходимые ракурсы объектов с автоматической генерацией масштабирования. Часть фонов была

выделена из заранее подготовленной коллекции текстур, а также десяти основных цветов, часть – автоматически сгенерировалась непосредственно из среды распознавания. Модуль создания выборки позволил объединить все полученные данные в обучающее множество. В заключение была произведена побайтовая проверка обучающей выборки, обучение и тестирование сверточной нейронной сети [92].

В таблице 5.8 приведены результаты эксперимента и работы комплекса распознавания объектов для нескольких классов в четырех случаях, представляющих из себя комбинации двух условий:

1) использование одного из трех типов обучающих множеств, описанных ранее;

2) использование стандартной сверточной нейронной сети или сверточной нейронной сети с двумя слоями второго порядка, описанной в данной диссертационной работе.

Во всех случаях учитывалось изменение степени освещенности объекта [38]. Тестовая выборка состояла из ста изображений каждого класса, выделенных со снимков камеры мобильного робота и преобразованных в формат обучающей выборки. Нейронная сеть после обучения тестировалась на тестовой выборке и генерировала файл, состоящий из номеров классов, полученных на выходе, который сравнивался с эталонной заранее известной последовательностью классов.

Для того чтобы проверить эффективность разработанных методики создания выборок и использования СНС второго порядка, были созданы обучающие выборки трех типов, отличающиеся степенью использования разработанной методики: использованием программы для полуавтоматического создания обучающих примеров, добавлением пустого класса в выборку с расширением коллекции фоновых изображений, и расширением обучающего множества за счет изменения параметра освещенности.

Все три типа выборок были использованы при обучении СНС первого и второго порядков. В таблице 5.8 приведены результаты распознавания для шести классов объектов в результате обучения нейронных сетей двух видов на каждой из созданных выборок.

Таблица 5.8 – Оценка обобщающей способности (%) для разных типов обучающих выборок с использованием СНС первого и второго порядков

Объект		Жаба	Человек	Машина	Самолет	Степлер	Плеер
СНС	Тип ОМ №1	73	72	74	82	84	84
	Тип ОМ №2	76	74	77	86	86	85
	Тип ОМ №3	80	77	81	87	90	92
СНС 2-го порядка	Тип ОМ №1	78	75	79	86	88	88
	Тип ОМ №2	81	78	81	87	90	89
	Тип ОМ №3	84	84	86	92	95	94

Таким образом, среднее увеличение обобщающей способности при использовании разработанной СНС второго порядка составляет в среднем 4% по сравнению с классической СНС на том же валидационном множестве. Также, при обучении на выборках, созданных с использованием разработанной методики, обобщающая способность СНС возрастает в среднем на 6% [39].

Также из результатов эксперимента можно сделать еще несколько выводов:

1) Для качественного обучения и распознавания объектов в реальном времени в обучающую выборку в обязательном порядке должен быть включен пустой класс объектов.

2) Увеличение размера выборки, а также разнообразия масштабирования изображений объектов, положительно влияет на качество обучения сверточной нейронной сети.

3) Использование кадров той среды, в которой планируется проводить распознавание объектов, является более предпочтительным, чем использование абстрактного набора текстур.

Выводы

1. Разработан программный комплекс, состоящий из отдельных модулей: модуля подготовки исходного материала для обучающей выборки, модуля создания обучающих выборок, модуля генерации гипотез о местоположении объекта на изображении, модуля распознавания объектов с помощью СНС второго порядка. Разработанный программный комплекс позволяет создавать и обрабатывать материал для обучающих выборок, формировать в полуавтоматическом режиме большие визуальные обучающие выборки, обучать СНС, выделять и распознавать объекты на изображениях с помощью обученного классификатора. Выполнена четвертая подзадача диссертационного исследования.

2. Экспериментально доказано, что разработанный численный метод отсеивания гипотез по низкочастотной структуре уменьшает количество генерируемых гипотез в среднем на 16 % и повышает скорость выделения и распознавания объектов на изображениях в среднем на 14%. Эти показатели могут отличаться в зависимости от размера обрабатываемого изображения.

3. Доказано, что разработанный параллельный алгоритм обработки данных в сверточных нейронных сетях второго порядка обеспечивает ускорение 10,5% по сравнению с известным алгоритмом.

4. Определено, что использование разработанной методики создания визуальных обучающих выборок повышает обобщающую способность нейронной сети в среднем на 6%. Также, важно отметить, что формирование обучающей выборки происходит в полуавтоматическом режиме, за счет чего на ее создание затрачивается меньше сил и времени.

5. Доказано, что использование нейронов с сумматорами высокого порядка в разработанной архитектуре СНС второго порядка повышает обобщающую способность сверточной нейронной сети в среднем на 4% по сравнению со стандартной архитектурой СНС типа «LeNet-5».

ЗАКЛЮЧЕНИЕ

1. Анализ научно-методического аппарата по теме диссертационной работы показал, что распознавание многопараметрических объектов на изображениях является на сегодня не решенной, актуальной задачей.

2. Выявлено, что для решения задачи распознавания объектов на изображениях, в общем виде необходимо решить две отдельных подзадачи: выделение и распознавание объектов. В результате анализа математических методов распознавания и выделения объектов, была выбрана нейросетевая модель R-CNN в качестве решения обеих подзадач.

3. Разработан метод выделения и распознавания объектов на изображениях, базирующийся на модели R-CNN, отличающийся от известных использованием СНС высокого порядка и численного метода отсеивания гипотез.

4. Обосновано, что наиболее оптимальной модификацией архитектуры СНС «LeNet-5» является архитектура с нейронами второго порядка на первом и втором сверточных слоях сети. Использование нейронов второго порядка позволяет повысить обобщающую способность нейронной сети в среднем на 4%.

5. Разработан численный метод отсеивания гипотез по низкоуровневой структуре. Использование разработанного численного метода позволяет снизить число обрабатываемых нейронной сетью гипотез в среднем на 8-26% до распознавания, с использованием более простых и быстрых алгоритмов. За счет этого, общая скорость выделения и распознавания объектов на изображениях увеличивается на 5-22%, в зависимости от размера изображения.

6. Предложен параллельный алгоритм обработки данных в СНС второго порядка, ориентированный на процессоры с векторно-матричной архитектурой. Использование разработанного параллельного алгоритма

повышает скорость обработки данных в СНС второго порядка в среднем на 10,5%.

7. Предложена методика полуавтоматического создания визуальных обучающих выборок для нейронных сетей. В методике учтены: оптимальные способ создания, тип фона и параметры изображений обучающей выборки (размер и сдвиг объектов, сложность фона), влияние освещенности, а также деформации и фильтрации изображений на качество обучения и обобщающую способность нейронной сети. Использование разработанной методики повышает обобщающую способность СНС в среднем на 6%.

8. Разработан программный комплекс для решения задачи выделения и распознавания объектов, позволяющий создавать обучающие выборки по эффективной методике, выделять объекты, распознавать объекты с помощью СНС высокого порядка.

Список литературы

1. Агеев, А. Д. Нейроматематика. Книга 6: учебное пособие для вузов [Текст] / А. Д. Агеев, А. Н. Балухто, А. В. Бычков, С. А. Верещагин и др. – М.: ИПРЖР, 2002. – 448 с.

2. Архитектура NeuroMatrix NM6403. Руководство пользователя. – М.: Модуль, 2004. – 36 с.

3. Барский, А. Б. Нейронные сети: распознавание, управление, принятие решений [Текст] / А. Б. Барский. – М.: Финансы и статистика, 2004. – 176 с.

4. Библиотека нижнего уровня для работы с таймерами, коммуникационными портами и прерываниями. – М.: Модуль, 2004. – 54 с.

5. Блейхут, Р. Теория и практика кодов, контролирующих ошибки [Текст] / Р. Блейхут. – М.: Мир, 1986. – 576 с.

6. Боровиков, В. П. Искусство анализа данных, 2-е издание [Текст] / В. П. Боровиков. – СПБ.: Питер, 2005. – 412 с.

7. Васенков, Д. В. Методы обучения искусственных нейронных сетей [Текст] / Д. В. Васенков // Информатизация образования, 2007. – С. 20-29.

8. Васильев, В. И. Распознающие системы. Справочник. 2-е издание [Текст] / В. И. Васильев. – Киев: Наукова думка, 1983. – 424 с.

9. Ватолин, Д. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео [Текст] / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин // Диалог-МИФИ, 2003. – С. 102-112.

10. Воронцов, К. В. Комбинаторный подход к оценке качества обучаемых алгоритмов [Текст] / К. В. Воронцов // Математические вопросы кибернетики. – 2004. – №13. – С. 5-36.

11. Воронцов, К. В. Комбинаторные оценки качества обучения по прецедентам [Текст] / К.В. Воронцов // Доклады РАН. – 2004. – Т. 394. – №2. – С. 175–178.

12. Галушкин, А. И. Нейрокомпьютеры и их применение: учебное пособие для ВУЗов. Книга 1 – Теория нейронных сетей [Текст] / А. И. Галушкин. – М.: ИПРЖР, 2000. – 416 с.

13. Гельфанд, И. М. Лекции по линейной алгебре [Текст] / И. М. Гельфанд. – М.: Добросвет, 2009. – 320 с.

14. Гонсалес, Р. Цифровая обработка изображений [Текст] / Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1007 с.

15. Горбань, А. Н. Нейроинформатика [Текст] / А. Н. Горбань, В. Л. Дунин-Барковский, А. Н. Кирдин. – Новосибирск: Наука. Сибирское предприятие РАН, 1998. – 296 с.

16. Горбань, А. Н. Обучение нейронных сетей [Текст] / А. Н. Горбань. – М.: ПараГраф, 1990. – 160 с.

17. Горбашевский, Д. Ю. Параллельная фильтрация в системе визуализации параллельных вычислений [Текст] / Д. Ю. Горбашевский, А. Ю. Казанцев // Новосибирск: ГрафиКонб, 2006. – С. 333-336.

18. Грузман, И. С. Цифровая обработка изображений в информационных системах: учебное пособие [Текст] / И. С. Грузман, В. С. Киричук, В. П. Косых. – Новосибирск: Изд-во НГТУ, 2000. – 168 с.

19. Желтов, С. Ю. Обработка и анализ изображений в задачах машинного зрения [Текст] / С. Ю. Желтов. – М.: Физматкнига, 2010. – 672 с.

20. Забалуев, М. Описание реализации программы. Оценка производительности эмуляции нейронной сети [Текст] / М. Забалуев. – М.: Модуль, 1999. – 22 с.

21. Заенцев, И. В. Нейронные сети. Основные модели [Текст] / И. В. Заенцев. – Воронеж: ВГУ, 1999. – 76 с.

22. Качановский, Ю. П. Предобработка данных для обучения нейронной сети [Текст] / Ю. П. Качановский, Е. А. Коротков // Фундаментальные исследования. – 2011. – №12. – С. 117-120.

23. Козадаев, А. С. Принципы реализаций искусственной нейронной сети [Текст] / А. С. Козадаев // Вестник Тамбовского университета. –2010. – № 1. – Т. 15. – С. 108-110.

24. Комарцова, Л. Г. Нейрокомпьютеры: учебное пособие [Текст] / Л. Г. Комарцова, А. В. Максимов. – М.: МГТУ, 2002. – 318 с.

25. Корн, Г. Справочник по математике для научных работников и инженеров [Текст] / Г. Корн, Т. Корн. – М.: Наука, 1970. – 246 с.

26. Короткий, С. Нейронные сети: основные положения [Текст] / С. Короткий. – М.: Наука, 1989. – 231 с.

27. Косоруков, Д. Е. Система на кристалле 1879ХК1 для цифровой обработки аналоговых сигналов в радиотехнических системах и спутниковой навигации [Текст] / Д. Е. Косоруков, А. Л. Эйсымонт, В. Г. Осипов. – М.: 2010. – 80 с.

28. Крисилов, В. А. Методы ускорения обучения нейронных сетей [Текст] / В. А. Крисилов, Д. Н. Олешко. – М.: Гардарика, 2005. – 1042 с.

29. Кросс-средство разработки программ: многоцелевой подключаемый отладчик. Руководство пользователя. – М.: Модуль, 2004. – 48 с.

30. Круглов, В. В. Искусственные нейронные сети. Теория и практика. – 2-е издание [Текст] / В. В. Круглов, В. В. Борисов. – М.: Горячая линия – Телеком, 2002. – 382 с.

31. Лагунов, Н. А. Влияние параметров визуальных обучающих выборок на качество распознавания нейронных сетей [Текст] / Н. А. Лагунов // Academic science – problems and achievements. – 2014. – Vol. IV. – С. 134-137.

32. Лагунов, Н. А. Выделение и распознавание объектов с использованием оптимизированного алгоритма селективного поиска и сверточной нейронной сети высокого порядка [Текст] // Фундаментальные исследования. – 2015. – №5. – С. 511-516.

33. Лагунов, Н. А. Обобщенная схема реализации сверточной нейронной сети на аппаратной платформе Neuromatrix [Текст] / Н. А. Лагунов, А. С. Якшин // Актуальные проблемы современной науки: материалы II международной научно-практической конференции. – Ставрополь, 2013. – С. 85-89.

34. Лагунов, Н. А. Применение сверточных нейронных сетей в задаче распознавания многопараметрических объектов [Текст] / Н. А. Лагунов // Пространство и время. – 2013. – №3(13). – С. 194-197.

35. Лагунов, Н. А. Разработка модуля создание визуальных обучающих выборок для нейронных сетей комплекса распознавания изображений с камеры мобильного робота [Текст] / Н. А. Лагунов // Фундаментальные и прикладные аспекты компьютерных технологий и информационной безопасности: материалы I Всероссийской научно-технической конференции. – Ставрополь, 2015. – С. 42-46.

36. Лагунов, Н. А. Способы улучшения качества визуальных обучающих выборок [Текст] // Вычислительные и информационные технологии в науке, технике и образовании: материалы I Всероссийской научно-технической конференции. – Ставрополь, 2014. – С. 25-28.

37. Лагунов, Н. А. Численный метод отсеивания гипотез при селективном поиске объектов на изображении [Текст] / Н. А. Лагунов // Студенческая наука для развития информационного общества: материалы III Всероссийской научно-технической конференции, 2015. – С. 159-161.

38. Мезенцева, О. С. Анализ и экспериментальное исследование зависимости качества обучения нейронных сетей от параметров обучающих выборок [Текст] / О. С. Мезенцева, Н. А. Лагунов // Вестник СКФУ. – 2014. – №5. – С. 15-21.

39. Мезенцева, О. С. Влияние предобработки изображений на качество обучения нейронной сети для их распознавания [Текст] / О. С. Мезенцева, Н. А. Лагунов // Вестник СКФУ. – 2015. – №1. – С. 51-58.

40. Мезенцева, О. С. Реализация нестандартных моделей нейронов на векторном процессоре NeuroMatrix [Текст] / О. С. Мезенцева, Н. А. Лагунов, Д. В. Мезенцев, Н. С. Савченко // Известия ЮФУ. – 2012. – №6. – С. 178-182.

41. Мерков, А. Распознавание образов. Введение в методы статистического обучения [Текст] / А. Мерков. – М.: Едиториал, УРСС, 2011. – 256 с.

42. Микросхема интегральная 1879ВМ2. Кросс-средства разработки программ. Описание языка ассемблера. – М.: Модуль, 2004. – 35 с.

43. Одрин, В. М. Морфологический анализ систем. Построение морфологических таблиц [Текст] / В. М. Одрин, С. С. Картавов. – Киев: Наукова думка, 1977. – 498 с.

44. Омату, С. Нейрокомпьютеры и их применение: учебное пособие для ВУЗов, книга 1 – Нейроуправление и его приложения [Текст] / С. Омату, М. Халид, Р. Юсоф. – М.: ИПРЖР, 2000. – 272 с.

45. Осовский, С. Нейронные сети для обработки информации [Текст] / С. Оссовский. – М.: Финансы и статистика, 2002. – 344 с.

46. Первые шаги в разработке программ для NeuroMatrix NM6403, Версия 1.1. – М.: Модуль, 2004. – 48 с.

47. Потапов, А. А. Новейшие методы обработки изображений [Текст] / А. А. Потапов, А. А. Пахомов, С. А. Никитин. – М.: Физматлит, 2008. – 496 с.

48. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы [Текст] / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия. – Телеком, 2006. – 147 с.

49. Сергиенко, А. Б. Цифровая обработка сигналов: учебник для вузов [Текст] / А. Б. Сергиенко. – СПб.: Питер, 2002. – 608 с.

50. Сойфер, В. А. Методы компьютерной обработки изображений [Текст] / В. А. Сойфер. – М.: Физматлит, 2003. – 459 с.

51. Тархов, Д. А. Нейронные сети. Модели и алгоритмы. Книга 18 [Текст] / Д. А. Тархов. – М.: Радиотехника, 2005. – 256 с.

52. Тархов, Д. А. Нейросетевые модели и алгоритмы. Справочник [Текст] / Д. А. Тархов. – М.: Радиотехника, 2014. – 352 с.

53. Ту, Дж. Принципы распознавания образов [Текст] / Дж. Ту, Р. Гонсалес. – М.: Мир, 1978. – 412 с.

54. Фомин, Я. А. Статистическая теория распознавания образов [Текст] / Я. А. Фомин, Г. Р. Тарловский. – М.: Радио и связь, 1986. – 624 с.

55. Форсайт, Д. А. Компьютерное зрение. Современный подход [Текст] / Д. А. Форсайт, П. Джин. – М.: Вильямс, 2004. – 928 с.

56. Фурман, Я. А. Введение в контурный анализ. Приложения к обработке изображений и сигналов [Текст] / Я. А. Фурман. – М.: ФИЗМАТЛИТ, 2003. – 592 с.

57. Хайкин, С. Нейронные сети: полный курс, 2-е издание [Текст] / С. Хайкин. – М.: Вильямс, 2006. – 1104 с.

58. Царегородцев, В. Г. Оптимизация предобработки данных: константа Липшица обучающей выборки и свойства обученных нейронных сетей [Текст] / В. Г. Царегородцев // Нейрокомпьютеры: разработка, применение, 2003. – 3-8 с.

59. Царегородцев, В. Г. Оптимизация предобработки признаков выборки данных: критерии оптимальности [Текст] / В. Г. Царегородцев // Нейрокомпьютеры. – 2005. – №4. – С. 32-40.

60. Царегородцев, В. Г. Рождение сложности (о новых типах слоёв и нейронов в искусственной нейронной сети) [Электронный ресурс] // URL: http://neuropro.ru/memo333.shtml (дата обращения 10.11.2015).

61. Царегородцев, В. Г. Свёрточные нейронные сети с полиномиальными (high-order) сумматорами нейронов [Электронный ресурс] // URL: http://neuropro.ru/memo334.shtml (дата обращения 10.11.2015).

62. Шапиро, Л. Компьютерное зрение [Текст] / Л. Шапиро, Дж. Стокман. – М.: Бином. Лаборатория знаний, 2006. – 752 с.

63. Щеглов, И. Н. Алгоритм формирования репрезентативной обучающей выборки искусственной нейронной сети [Текст] / И. Н. Щеглов,

С. А. Демченко, А. В. Богомолов, А. А. Подлесских // Нейрокомпьютеры и их применение: материалы V Всероссийской конференции. – Москва, 1999. – С. 405-407.

64. Abrial, P. Color detection for vision machine defect inspection on electronic devices [Text] / P. Abrial // Electronic Manufacturing Technology Symposium. – Melaka, 2010. – pp. 180-192.

65. Agostinelli, F. Learning Activation Functions to Improve Deep Neural Networks [Text] / F. Agostinelli, M. Hoffman, P. Sadowski, P. Baldi // International Conference on Learning Representations. – Puerto Rico, 2015. – pp. 1024-1032.

66. Agrawal, P. Analyzing the Performance of Multilayer Neural Networks for Object Recognition [Text] / P. Agrawal, R. Girshick, J. Malik // Lecture Notes in Computer Science. – 2014. – Vol. 8695. – pp. 329-344.

67. Alexe, B. Measuring the objectness of image windows [Text] / B. Alexe, T. Deselaers, V. Ferrari // Transactions on pattern analysis machine intelligence. – 2012. – Vol. 34. – pp. 2189-2202.

68. Alvarez, J. Road scene segmentation from a single image [Text] / J. Alvarez, T. Gevers, Y. LeCun, A. M. Lopez // European Conference on Computer Vision. – Firenze, 2012. – pp. 376-389.

69. Arbelaez, P. Multiscale Combinatorial Grouping Computer Vision and Pattern Recognition [Text] / P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques // Computer Vision and Pattern Recognition. – Columbus, 2014. – pp. 328-335.

70. Arenzon, J. J. Neural networks with high order connections [Text] / J. J. Arenzon, R. M. C. de Almeida // Physical review. – 1993. – Vol. 48. – №5. – pp. 4060-4069.

71. Arnab, A. Higher Order Potentials in End-to-End Trainable Conditional Random Fields [Text] / A. Arnab, S. Jayasumana, S. Zheng, P. Torr // Computer Vision and Pattern Recognition. – Boston, 2015. – pp. 402-411.

72. Arora, S. Performance Comparison of SVM and ANN for Handwritten Devnagari Character Recognition [Text] / S. Arora, D. Bhattacharjee, M. Nasipuri // International Journal of Computer Science Issues. – 2010. – Vol. 7. – pp. 59-72.

73. Bengio, Y. Learning deep architectures for AI [Text] / Y. Bengio // Foundations and Trends in Machine Learning. – 2009. – Vol. 2. – pp. 1-127.

74. Ben-Hur, A. Support vector clustering [Text] / A. Ben-Hur, D. Horn, H. Siegelmann, V. Vapnik // Journal of Machine Learning Research. – 2001. – Vol. 3. – pp. 125-137.

75. Bin, M. Standardization and Its Effects on K-Means Clustering Algorithm [Text] / M. Bin, D. Usman // Research Journal of Applied Sciences, Engineering and Technology. – 2013. – Vol. 4. – pp. 412-422.

76. Bottou, L. Global training of document processing systems using graph transformer networks [Text] / L. Bottou, Y. LeCun, Y. Bengio // Computer Vision and Pattern Recognition. – Puerto-Rico, 1997. – pp. 280-291.

77. Brunelli, R. Template Matching Techniques in Computer Vision: Theory and Practice [Text] / R. Brunelli. – Wiley, 2009. – 346 p.

78. Cadieu, C. F. Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition [Text] / C. F. Cadieu, H. Hong, D. L. K. Yamins, N. Pinto // Computational Biology. – 2014. – Vol. 10. – pp. 1-18.

79. Caffe. Deep learning framework by the BVLC [Электронный ресурс] // URL: http://caffe.berkeleyvision.org/ (дата обращения 05.11.2015).

80. Canny, J. A Computational Approach to Edge Detection [Text] / J. A. Canny // Transactions on Pattern Analysis and Machine Intelligence. – 1986. – Vol. 8. – pp. 679-698.

81. Carreira, S. Constrained parametric min-cuts for automatic object segmentation [Text] / S Carreira, C. Sminchisescu // Transactions on pattern analysis machine intelligence. – 2012. – Vol. 34. – pp.1312 -1328.

82. Cheng, M.-M. BING: Binarized Normed Gradients for Objectness Estimation at 300fps [Text] / M.-M. Cheng, Z. Zhang, W. Y. Lin, P. Torr // Computer Vision and Pattern Recognition. – Puerto-Rico, 2014. – pp. 260-275.

83. Cheng, M.-M. SalientShape: Group Saliency in Image Collections [Text] / M.-M. Cheng, N. J. Mitra, X. Huang, S. M. Hu // The Visual Computer. – 2014. – Vol. 30(4). – pp. 488-495.

84. Cireşan, D. C. High-Performance Neural Networks for Visual Object Classification [Text] / D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber // Intelligent Fill in Form. – 2011. – Vol. 1. – pp. 1-11.

85. Ciresan, D. C. Deep big simple neural nets for handwritten digit recognition [Text] / D.C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber // Neural Computation. – 2010. – Vol. 22. – pp. 311-333.

86. Ciresan, D. C. Handwritten Digit Recognition with a Committee of Deep Neural Nets on GPUs [Text] / D. C. Ciresan, U. Meier, L. M. Gambardella, J. Schmidhuber // International Conference on Computer Vision. – Portugal, 2011. – pp. 240-254.

87. Coates, A. An Analysis of Single-Layer Networks in Unsupervised Feature Learning [Text] / A. Coates, A. Ng, H. Lee // International Conference on Artificial Intelligence and Statistics. – Lauderdale, 2011. – pp. 215-223.

88. Cuong, N. K. Face Detection using Variance based Haar-Like feature and SVM [Text] / N. K. Cuong, H. P. Ju, J. Ho-Youl // Proceedings of the Fourth International Conference on Informatics in Control, Automation and Robotics. – Madeira, 2008. – pp. 222-243.

89. Dollar, P. A Seismic Shift in Object Detection [Электронный ресурс] // URL: https://pdollar.wordpress.com/2013/12/10/a-seismic-shift-in-object-detection/ (дата обращения 10.11.2015).

90. Dushnik, D. Video Segmentation via Diffusion Bases [Text] / D. Dushnik, A. Schclar, A. Averbuch // Computer Vision and Pattern Recognition. – Portland, 2013. – pp. 135-193.

91. Erhan, D. Scalable Object Detection using Deep Neural Networks [Text] / D. Erphan, C. Szegedy, A. Toshev, D. Anguelov // Computer Vision and Pattern Recognition. – Columbus, 2014. – pp. 2155-2162.

92. Everitt, B. S. Cambridge Dictionary of Statistics, 4th edition [Text] / B. S. Everitt. – Cambridge University Press, 2010. – 480 p.

93. Fairchild, M. D. Color Appearance Models, 2nd edition [Text] / M. D. Fairchild. – Addison-Wesley, 2005. – 422 p.

94. Fast Artificial Neural Network Library [Электронный ресурс] // URL: http://fann.sourceforge.net/ (дата обращения 05.10.2015).

95. Fei-Fei, L. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories [Text] / L. Fei-Fei, R. Fergus, P. Perona // Workshop on Generative-Model Based Vision. – Washington DC, 2004. – pp. 802-814.

96. Felzenszwalb, F. Efficient Graph-Based Image Segmentation [Text] / F. Felzenszwalb, D. P. Huttenlocher // International Journal of Computer Vision. – 2004. – Vol. 59. – pp. 145-157.

97. Fink, G. A. Models for Pattern Recognition From Theory to Applications [Text] / G. A. Fink, B. M. Markov. – Berlin Heidelberg: Springer-Verlag, 2008. – 424 p.

98. Fukushima, K. Neocognitron: A model for visual pattern recognition. The Handbook of Brain Theory and Neural Networks [Text] / K. Fukushima. – MIT Press, 1995. – 220 p.

99. Girshick, R. Rich feature hierarchies for accurate object detection and semantic segmentation [Text] / R. Girshick, J. Donahue, T. Darrell, J. Malik // Computer Vision and Pattern Recognition. – Columbus, 2014. – pp. 580-587.

100. González, R. C. Digital Image Processing [Text] / R. C. González , R. E. Woods. – Prentice Hall, 2007. – 85 p.

101. Goyal, S. Object Recognition Using Deep Neural Networks: A Survey [Text] / S. Goyal, P. Benjamin // Neural and Evolutionary Computing. –2014. – Vol. 2. – pp. 1-16.

102. Gu, J. Recent Advances in Convolutional Neural Networks [Text] / J. Gu, Z. Wang, J. Kuen, L. Ma // Transactions on Cybernetics. – 2015. – Vol. 2. – pp. 180-186.

103. Hassoun, M. Fundamentals of artificial neural networks [Text] / M. Hassoun. – A Bradford Book, 2003. – 511 p.

104. Haykin, S. Adaptive Filter Theory, 3rd Edition [Text] / S. Haykin. – Prentice-Hall, 1996. – 420 p.

105. Held, J. B. From a Few Cores to Many: A Tera-scale Computing Research Overview [Text] / J. B. Held, J. K. Sean // White Paper Research at Intel, 2006. – pp. 228-234.

106. Hoiem, E. Category independent object proposals [Text] / E. Hoiem, D. Hoiem // Lecture Notes in Computer Science. – California, 2010. – pp. 575-588.

107. Jack, W. Invariant pattern recognition using higher-order neural networks [Text] / W. Jack, J. Chang // Proceedings of 1993 International Joint Conference on Neural Networks. – Nagoya, 1993. – pp. 420-434.

108. Jasper, R. R. Segmentation As Selective Search for Object Recognition [Text] / R. R. Jasper, E. A. Koen van de Sande, G. Theo // International Conference on Computer Vision. – Barcelona, 2011. – pp. 1879-1886.

109. Jasper, R. R. Selective Search for Object Recognition [Text] / R. R. Jasper, E. A. K. van de Sande, G. Theo, A. W. M. Smeulders // International Journal of Computer Vision. – 2013. – Vol. 104. – pp. 154-171.

110. Keysers, D. Comparison and Combination of State-of-the-art Techniques for Handwritten Character Recognition: Topping the MNIST Benchmark [Text] / D. Keysers // Computer Vision and Pattern Recognition. – 2006. – pp. 110–113.

111. Krizhevsky, A. ImageNet classification with deep convolutional neural networks [Text] / A. Krizhevsky, I. Sutskever, G. E. Hinton // Advances in

Neural Information Processing Systems 25. – Harrahs and Harveys, 2012. – pp. 1106-1114.

112. Krizhevsky, A. ImageNet Classification with Deep Convolutional Neural Networks [Text] / A. Krizhevsky, I. Sutskever, G. Hinton // Advances in Neural Information Processing Systems 25. – Harrahs and Harveys, 2012. – pp. 2146-2153.

113. Lawrence, S. Face Recognition: A Convolutional Neural Network Approach [Text] / S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back // Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition. – 1997. – Vol. 8. – pp. 1-24.

114. LeCun, Y. Convolutional networks for images, speech, and timeseries [Text] / Y. LeCun, Y. Bengio // The Handbook of Brain Theory and Neural Networks. – 1995. – Vol. 1. – pp. 255-258.

115. LeCun, Y. Efficient BackProp [Text] / Y. LeCun, L. Bottou, G. B. Orr // Neural Networks: Tricks of the trade. – 1998. – Vol. 1. – pp. 5-50.

116. LeCun, Y. Gradient-based learning applied to document recognition [Text] / Y. LeCun, L. Bottou, Y. Bengio, P. Haffner // Proceedings of the IEEE. – 1998. – Vol. 86. – pp. 2278-2324.

117. LeCun, Y. Handwritten digit recognition with a backpropagation neural network / Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel // Advances in Neural Information Processing Systems. – 1990. – Vol. 2. – pp. 396-404.

118. LeCun, Y. The MNIST Database of handwritten digits [Text] / Y. LeCun, C. Cortes, C.J.C. Burges // Advances in Neural Information Processing Systems. – 1998. – Vol. 1. – pp. 361-375.

119. LeCun, Y. The NORB Dataset [Электронный ресурс] / Y. LeCun, J. Huang // URL: http://www.cs.nyu.edu/~ylclab/data/norb-v1.0/ (дата обращения 05.10.2015).

120. Lin, L. Complex Background Subtraction by Pursuing Dynamic Spatio-Temporal Models [Text] / L. Lin, Y. Xu, X. Liang, J. Lai // Image Processing. – 2015. – Vol. 23. – pp. 3191-3202.

121. Muller, K. An Introduction to Kernel-Based Learning Algorithms [Text] / K. Muller, S. Mika, G. Ratsch, K. Tsuda, B. Scholkopf // Neural Networks. – 2001. – Vol. 1. – pp. 181-201.

122. Neon. Python based Deep Learning Framework by Nervana [Электронный ресурс] // URL: https://github.com/NervanaSystems/neon (дата обращения 05.11.2015).

123. Neural Designer [Электронный ресурс] // URL: https://www.neuraldesigner.com/ (дата обращения 05.11.2015).

124. Nissen, S. Neural Networks Made Simple [Text] / S. Nissen // Artificial Intelligence. – Pittsburgh, 2005. – pp. 14-19.

125. Nixon, S. M. Feature Extraction and Image Processing [Text] / S. M. Nixon, A. S. Aguado. – Academic Press, 2008. – 88 p.

126. Parthey, J. Porting GCC to the TMS320-C6000 DSP Architecture [Text] / J. Parthey, R. Baumgartl // Proceedings of GSPx’04. – Santa Clara, 2004. – pp. 112-120.

127. Philip, K. Geometric Tools for Computer Graphics [Text] / K. Philip, H. David. – The Morgan Kaufmann Series in Computer Graphics, 2003. – 98 p.

128. Pinto, N. Why is real-world visual object recognition hard? [Text] / N. Pinto, D. D. Cox, J. J. DiCarlo // Computation Biology. – 2008. – Vol. 4. – pp. 27-31.

129. Ranzato, M. A. What is the best multi-stage architecture for object recognition? [Text] / M. A. Ranzato, K. Jarrett, K. Kavukcuoglu, Y. LeCun // Computer Vision 12th International Conference. – Florida, 2009. – pp. 2146-2153.

130. Reed, S. Training Deep Neural Networks on Noisy Labels with Bootstrapping [Text] / S. Reed, D. Anguelov, C. Szegedy, D. Erhan, A. Rabinovich // Neural and Evolutionary Computing. – 2014. – Vol. 4. – pp. 412-428.

131. Riesenhuber, M. Neural mechanisms of object recognition [Text] / M. Riesenhuber, T. Poggio // Current Opinion in Neurobiology. – 2002. – Vol. 12. – pp. 162-168.

132. Salomon, D. Assemblers and Loaders [Text] / D. Salomon. – UK: Ellis Horwood Ltd, 1993. – 294 p.

133. Sermanet, P. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks [Text] / P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun // Computer Vision and Pattern Recognition. – Columbus, 2013. – pp. 1082-1090.

134. Serre, T. Object recognition with features inspired by visual cortex [Text] / T. Serre, L. Wolf, T. Poggio // Computer Vision and Pattern Recognition Conference. – Minneapolis, 2007. – pp. 232-246.

135. Shaika, K. B. Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space [Text] / K. B. Shaika, P. Ganesana, V. Kalista, B. S. Sathisha // Computer Science. – 2015. – Vol. 57. – pp. 41-48.

136. Simard, P. Transformation invariance in pattern recognition – tangent distance and tangent propagation [Text] / P. Simard, Y. LeCun, J. Denker, B. Victorri // International Journal of Imaging Systems and Technology. – 2000. – Vol. 11. – pp. 181-197.

137. Simard, P.Y. Best practices for convolutional neural networks applied to visual document analysis [Text] / P. Y. Simard, D. Steinkraus, J. C. Platt // Seventh International Conference on Document Analysis and Recognition. – Edinburgh, 2003. – pp. 541-555.

138. Stavros, J. Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers [Text] / J. Stavros, Paulo J. G. Lisboa // Transactions on neural networks. – 1992. – Vol. 3. – pp. 123-134.

139. Steven, W. S. The Scientist and Engineer’s Guide to Digital Signal Processing, Second Edition [Text] / W. S. Steven. – California Technical Publishing, 1999. – 432 p.

140. Taigman, Y. DeepFace: Closing the gap to human-level performance in face verification [Text] / Y. Taigman, M. Yang, M. Aurelio, L. Wolf // Conference on Computer Vision and Pattern Recognition. – Columbus, 2014. – pp. 344-354.

141. TensorFlow. An Open Source Software Library for Machine Intelligence [Электронный ресурс] // URL: https://www.tensorflow.org/ (дата обращения 05.11.2015).

142. The VELES. Distributed platform for rapid Deep learning application development [Электронный ресурс] // URL: https://velesnet.ml/ (дата обращения 05.11.2015).

143. Torch. A scientific computing framework for LuaJIT [Электронный ресурс] // URL: http://torch.ch/ (дата обращения 05.11.2015).

144. Triantaphyllou, E. Multi-Criteria Decision Making: A Comparative Study [Text] / E. Triantaphyllou. – Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000. – 320 p.

145. Viola, P. Rapid object detection using a boosted cascade of simple features [Text] / P. Viola, M. Jones // Computer Vision and Pattern Recognition. – Columbus, 2001. – pp. 511-518.

146. Wang, M. Y. Regionlets for generic object detection [Text] / M. Y. Wang, S. Zhu, Y. Lin // International Conference on Computer Vision. – Sydney, 2013. – pp. 17-24.

147. Wiatowski, T. A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction [Text] / T. Wiatowski, H. Bölcskei // International Symposium on Information Theory. – Hong Kong, 2015. – pp. 2048-2054.

148. Widrow, B. Learning phenomena in layered neural networks [Text] / B. Widrow, R. Winter, R. Baxter // First Int. Conf. Neural Networks. – San Diego, 1987. – pp. 411-429.

149. Wilson, D. R. The general inefficiency of batch training for gradient descent learning [Text] / D. R. Wilson, T. R. Martinez // Neural Networks. – 2003. – Vol. 14. – pp. 1429-1451.

150. Wilson, D. R. The inefficiency of batch training for large training sets [Text] / D. R. Wilson, T. R. Martinez // International Joint Conference Neural Networks. – Como, 2000. – pp. 113-117.

151. Yang, G. Human Face Detection in Complex Background [Text] / G. Yang, T. S. Huang // Pattern Recognition. – 1994. – Vol. 24. – pp. 53-63.

152. Zheng, Y. Object Recognition using Neural Networks with Bottom-up and Top-down Pathways [Text] / Y. Zheng, Y. Meng, Y. Jin // Neurocomputing. – 2011. – Vol. 74. – pp. 3158-3169.

153. Ziou, D. Edge detection techniques: An overview [Text] / D. Ziou, S. Tabbone // International Journal of Pattern Recognition and Image Analysis. – 1998. – Vol. 8. – pp. 537-559.

Приложение А – Методика полуавтоматического создания визуальных обучающих выборок для нейронных сетей

Разработанная методика создания визуальных обучающих выборок для нейронных сетей включает следующие шаги:

1. Выбор объектов для распознавания. Рекомендуется, чтобы объекты имели яркие отличительные свойства (цвет, форма, структура, мелкие детали). Методика направлена на создание обучающих выборок для распознавания конкретного набора физических объектов малого и среднего размеров.

2. Создание коллекции фонов для обучающей выборки. Предпочтительно использование сложных сцен, вроде пейзажей, а также текстуры, 8-10 основных цветов, но не цифровых, а в виде неоднородных текстур. Желательно наличие в коллекции фонов частей фотографий комнаты, в которой предполагается проводить распознавание. Рекомендуемое количество фонов – не меньше 100, при наличии 10 классов объектов. Соотношение сложных фонов к простым: 25 сложных фонов-сцен реального мира, 25 текстур, 10 основных цветов, 40 фонов из помещения, где проводится распознавание.

3. Создание фотографий выбранных объектов при трех разных уровнях освещенности (рекомендуется 5-10 люкс, 10-15 люкс и 15-20 люкс), сверху, постепенно смещая объект на 45 градусов, по 8 ракурсов на каждый объект. Рекомендуется исключать сложные, неоднозначные ракурсы, заметно отличающиеся от прочих состояний рассматриваемого объекта; а объекты, имеющие простую структуру, вроде мяча, рекомендуется наклонять для изменения распределения тени по поверхности объекта.

4. Выделение областей, соответствующих объектам из полученных фотографий (ручное, автоматическое или полуавтоматическое). Интерполяция фоновых изображений до размера 200x200 пикселей, а изображений объектов до 100x100 пикселей, 150x150 пикселей, 180x180

пикселей. Перевод всех изображений в оттенки серого, запись в файлы текстового формата, представляющие из себя квадратные матрицы, в которых элементы соответствуют либо значению интенсивности изображения в данной точке, либо содержат отрицательное значение «-1» в точках, не содержащих части целевого объекта, для простоты манипуляций и наложения объектов на фоны.

5. Подготовка изображений для пустого класса: включает все имеющиеся фоны и автоматически выделенные участки комнаты разных размеров, изменение размеров до 96x96 пикселей, перевод в оттенки серого. Изображения для пустого класса сразу масштабируются, т. к. для них нет необходимости совершать дополнительную обработку.

6. Автоматическое наложение всех имеющихся объектов на все фоны, осуществляемое программой; запись обучающей выборки в бинарный файл, добавление пустого класса. Рекомендуется не сохранять файлы изображений на диск для увеличения скорости обработки данных. Рекомендуется также осуществлять запись каждого изображения последовательно, без формирования большого массива данных.

7. Интерполяция всех полученных изображений до размеров 96x96 пикселей. Такой подход дает более качественные обучающие примеры, чем использование изначально уменьшенных объектов и фонов, и способствует повышению качества обучения нейронной сети и распознавания объектов.

8. Внесение дополнительного разнообразия в выборку за счет изменения освещенности всех полученных примеров. Рекомендуется преобразовывать интенсивность пикселей в диапазоне от -20 до +40, используя четыре состояния (-20, 0, +20, +40) для каждого изображения выборки.

9. Опциональное использование фильтров повышения четкости или выделения границ. Фильтр повышения четкости представляет собой простое применение квадратной матрицы свертки 3x3 элемента и показывает

ощутимое влияние на процесс обучения нейронной сети на изображениях обучающей выборки.

10. Опциональное расширение выборки за счет использования аффинных преобразований и эластичных искажений. Описанный подход к созданию выборок и без этого обеспечивает необходимое разнообразие и полноту обучающих данных. Может потребоваться для небольших обучающих выборок, при небольшом количестве исходного материала. Также следует учитывать, что обучающая выборка значительно увеличивается за счет изменения параметра освещенности.

11. Запись файлов меток и информационного файла, необходимых для контроля и правильного протекания процесса обучения. Файл меток имеет расширение «.cat» и содержит последовательность чисел типа «int», в точности соответствующую порядку классов примеров обучающей выборки; записывается в бинарном формате. Информационный файл используется для удобства и простоты использования обучающей выборки, записывается в текстовом формате и содержит информацию о размере выборки, количестве и последовательности номеров входящих в нее классов объектов.

12. Опционально преобразование массива данных выборки в изображения и сохранение их на диск. Рекомендуется осуществлять сохранение обучающей выборки в виде изображений после завершения процесса создания, т. к. это сильно замедляет процесс обработки данных, и по сути не является обязательным этапом – правильность генерации изображений можно проверить и другими способами, например, с помощью модуля тестирования визуальных обучающих множеств.

Приложение Б – Листинг программных кодов

Б1. Листинг основных функций, реализующих СНС второго порядка

Функция обратного прохода от S к C-слою

```
void BackPropagationFromSLayerToCLayerHigh(StructForCards ***S, int ColS, int SizeS,
StructForCardsHigh ***C, int ColC, int SizeC,  StructForSLayerKernel *KernelsForS, int SizeRFOfS)
{
  int i, j, q;
  for (i = 0; i < ColS; i++) {
    int t1 = 0, t2 = 0;
    for (j = 0; j < SizeS; j++) {
      for (q = 0; q < SizeS; q++) {
        for (int p = 0; p < SizeRFOfS; p++) {
          for (int pp = 0; pp < SizeRFOfS; pp++) {
            C[i][t1][t2].error = S[i][j][q].error * KernelsForS[i].u *
DerivativeOfActivationFunction(C[i][t1][t2].output);
            C[i][t1][t2].uerror = S[i][j][q].error * KernelsForS[i].u *
DerivativeOfActivationFunction(C[i][t1][t2].output * C[i][t1][t2].output);
            t2++;  }
          t1++;
          t2 = q * SizeRFOfS;
        }
        t1 = j * SizeRFOfS;
        t2 = (q + 1) * SizeRFOfS;
      }
      t1 = (j + 1) * SizeRFOfS;
      t2 = 0;      }   }
  for (i = 0; i < ColC; i++)
    for (j = 0; j < SizeC; j++)
      for (q = 0; q < SizeC; q++) {
        //C[i][j][q].error *= KernelsForS[i].u;
        //C[i][j][q].error *= DerivativeOfActivationFunction(C[i][j][q].output);
      }}
```

Функция создания ядер для C-слоя высокого уровня

```
void CreateKernelsForCLayerHigh(float ***KernelsForCLayer, float ***KernelsForCLayer2,  int
ColCards, int *rows, int columns) {
  for (int i = 0; i < ColCards; i++) {
    KernelsForCLayer[i] = new float *[rows[i]];
    for (int j = 0; j < rows[i]; j++) KernelsForCLayer[i][j] = new float [columns];
  }}
```

Функция создания и инициализации слоя высокого уровня

```
void CreateLayerHigh(StructForCardsHigh ***mas, int ColCards, int size){
  for (int i = 0; i < ColCards; i++) {
    mas[i] = new StructForCardsHigh *[size];
    for (int j = 0; j < size; j++) mas[i][j] = new StructForCardsHigh [size];    }}
```

Функция удаления слоя высокого уровня

```
void DelLayerHigh(StructForCardsHigh ***mas, int ColCards, int size) {
  for (int i = 0; i < ColCards; i++) {
    for (int j = 0; j < size; j++) delete [] mas[i][j];
    delete [] mas[i];  }   delete [] mas; }
```

Функция прямого распространения от C к S-слою

```
void FromCLayerToSLayerHigh(StructForCards ***S, int ColCards, int SizeS, StructForCardsHigh ***C,
StructForSLayerKernel *KernelForS, int SizeRFOfS) {
```

```
  for (int i = 0; i < ColCards; i++) {
    int t1 = 0, t2 = 0;
    for (int j = 0; j < SizeS; j++) {
      for (int q = 0; q < SizeS; q++) {
        float w_sum = 0;
        for (int p = 0; p < SizeRFOfS; p++) {
          for (int pp = 0; pp < SizeRFOfS; pp++) {
            w_sum += C[i][t1][t2].output;
            t2++;
          }
          t1++;
          t2 = q * SizeRFOfS;
        }
        w_sum *= KernelForS[i].u;
        w_sum += KernelForS[i].b;
        S[i][j][q].output = ActivityFunction(w_sum);
        t1 = j * SizeRFOfS;
        t2 = (q + 1) * SizeRFOfS;
      }
      t1 = (j + 1) * SizeRFOfS;
      t2 = 0;  }}}
```

Функция прямого распространения от входов к первому С-слою высокого уровня

```
void FromInputToC1High(StructForCardsHigh ***C1, int ColCards, int SizeC1, float ***Input,
        float ***KernelsForC1, float ***UKernelsForC1, float *BiasForC1, int *ColCardsForC1, int
**IndCardsForC1, int SizeRFOfC1) {
  for (int i = 0; i < ColCards; i++)
    for (int j = 0; j < SizeC1; j++)
      for (int q = 0; q < SizeC1; q++)        {
        float w_sum = 0;
        int shift = 0;
        for (int z = 0; z < ColCardsForC1[i]; z++)
        {
          for (int p = 0; p < SizeRFOfC1; p++)
            for (int pp = 0; pp < SizeRFOfC1; pp++)
              w_sum    +=    Input[IndCardsForC1[i][z]][j+p][q+pp]    *KernelsForC1[i][shift+p][pp]    +
Input[IndCardsForC1[i][z]][j+p][q+pp] * Input[IndCardsForC1[i][z]][j+p][q+pp] * UKernelsForC1[i][shift+p][pp];
          shift += SizeRFOfC1; }
        w_sum += BiasForC1[i];
        C1[i][j][q].output = ActivityFunction(w_sum);   }}
```

Функция инициализации слоя высокого уровня

```
void InizLayerHigh(StructForCardsHigh ***mas, int ColCards, int size) {
  for (int i = 0; i < ColCards; i++)
    for (int j = 0; j < size; j++)
      for (int q = 0; q < size; q++)
      {
        mas[i][j][q].error = 0;
        mas[i][j][q].uerror = 0;
      }}
struct StructForCardsHigh {
  float output;
  float error;
  float uerror;};
```

Функция обновления весовых коэффициентов для обоих ядер

```
void UpdateWeightsForCLayerHigh(StructForCardsHigh ***C, int ColC, int SizeC,
          float ***Input, float ***KernelsForC, float ***UKernelsForC, float *BiasForC,
          int *ColCardsForC, int **IndCardsForC, float **DeltaCoreForC, float **UDeltaCoreForC,
int SizeRFOfC, float eta) {
```

```
int p, pp, j, q;
InizDeltaCore(DeltaCoreForC, SizeRFOfC);
InizDeltaCore(UDeltaCoreForC, SizeRFOfC);
for (int i = 0; i < ColC; i++) { // Цикл по картам
  int shift = 0;
  float sum = 0;
  float sum2 = 0;
  for (j = 0; j < SizeC; j++)
    for (q = 0; q < SizeC; q++)
    {
      sum += C[i][j][q].error;
      sum += C[i][j][q].uerror;
    }
  BiasForC[i] += sum * eta;
  for (int z = 0; z < ColCardsForC[i]; z++) {
    for (j = 0; j < SizeC; j++)
      for (q = 0; q < SizeC; q++) {
        for (p = 0; p < SizeRFOfC; p++)
          for (pp = 0; pp < SizeRFOfC; pp++)
          {
            DeltaCoreForC[p][pp] += Input[IndCardsForC[i][z]][j+p][q+pp] * C[i][j][q].error;
            UDeltaCoreForC[p][pp] += Input[IndCardsForC[i][z]][j+p][q+pp] * C[i][j][q].uerror;  } }
      for (p = 0; p < SizeRFOfC; p++)
        for (pp = 0; pp < SizeRFOfC; pp++)
        {
          KernelsForC[i][shift+p][pp] += DeltaCoreForC[p][pp] * eta;
          UKernelsForC[i][shift+p][pp] += UDeltaCoreForC[p][pp] * eta;
        }
      shift += SizeRFOfC;
      InizDeltaCore(DeltaCoreForC, SizeRFOfC);
      InizDeltaCore(UDeltaCoreForC, SizeRFOfC); } }
```

Б2. Листинг кода, реализующего отсеивание гипотез по низкочастотной структуре

```
// функция для рассчета первого 64-битного вектора
__int64 calcImageHash1(IplImage* image, bool show_results=false);
// функция для рассчета второго 64-битного вектора
__int64 calcImageHash2(IplImage* image, bool show_results=false);
// функция для рассчёта расстояния Хэмминга
__int64 calcHammingDistance(__int64 x, __int64 y);

int main()
{
  cvNamedWindow( "etalon");
  cvNamedWindow( "hyp");

  IplImage *etalon=0, *hyp=0;
  __int64 porog = 0;
  bool hyps[number_of_hyps];

   char* etalon_filename = etalon_name;
  etalon = cvLoadImage(etalon_filename, 1);

  if(!etalon){
           printf("[!] Error: cant load image: %s\n", etalon_filename);
           return -1;
     }

  __int64 ve1 = calcImageHash1(etalon, true);
  __int64 ve2 = calcImageHash2(etalon, true);
```

```
  for(int n = 0; n < number_of_hyps; n++)
  {
      char* hyp_filename = hyp_names[n];
      hyp = cvLoadImage(hyp_filename, 1);

      //printf("[i] etalon: %s\n", etalon_filename);
      //printf("[i] hyp: %s\n", hyp_filename);

      if(!hyp){
            printf("[!] Error: cant load image: %s\n", hyp_filename);
            return -1;
      }

      // покажем изображение (если требуется)
      //cvShowImage( "etalon", etalon );
      //cvShowImage( "hyp", hyp );

      // рассчитаем 64-битрые вектора гипотезы с помощью функций
      __int64 vh1 = calcImageHash1(hyp, false);
      __int64 vh2 = calcImageHash2(hyp, false);

      // рассчитаем расстояние Хэмминга
      __int64 dist1 = calcHammingDistance(ve1, vh1);
    __int64 dist2 = calcHammingDistance(ve2, vh2);

    // рассчитаем результирующую разность (дельта)
    __int64 delta = dist1 + dist2;

    if(delta < porog)
      hyps[n] = 1;
    else
      hyps[n] = 0;

  }

      // ждём нажатия клавиши
      cvWaitKey(0);

      // освобождаем ресурсы
      cvReleaseImage(&etalon);
      cvReleaseImage(&hyp);

      // удаляем окна
      cvDestroyAllWindows();

      return 0;
}

// функция для рассчета первого 64-битного вектора
__int64 calcImageHash1(IplImage* src, bool show_results)
{
      if(!src){
            return 0;
      }

    // результирующее изображение, изображение в оттенках серого, бинарное изображение
      IplImage *res=0, *gray=0, *bin =0;

      res = cvCreateImage( cvSize(8, 8), src->depth, src->nChannels);
      gray = cvCreateImage( cvSize(8, 8), IPL_DEPTH_8U, 1);
      bin = cvCreateImage( cvSize(8, 8), IPL_DEPTH_8U, 1);

      // уменьшаем картинку
```

```
	cvResize(src, res);
	// переводим в оттенки серого
	cvCvtColor(res, gray, CV_BGR2GRAY);
	// вычисляем среднее
	CvScalar average = cvAvg(gray);
	printf("[i] average: %.2f \n", average.val[0]);
	// получаем бинарное изображение относительно среднего
	cvThreshold(gray, bin, average.val[0], 255, CV_THRESH_BINARY);

	// построим хэш
	__int64 hash = 0;

	int i=0;
	// пробегаемся по всем пикселям изображения
	for( int y=0; y<bin->height; y++ ) {
		uchar* ptr = (uchar*) (bin->imageData + y * bin->widthStep);
		for( int x=0; x<bin->width; x++ ) {
			// 1 канал
			if(ptr[x])
		  {
			   hash |= 1i64<<i;
			}
			i++;
		}
	}

	printf("[i] hash: %I64X \n", hash);

	if(show_results){
		// увеличенные картинки для отображения результатов
		IplImage* dst3 = cvCreateImage( cvSize(128, 128), IPL_DEPTH_8U, 3);
		IplImage* dst1 = cvCreateImage( cvSize(128, 128), IPL_DEPTH_8U, 1);

		// показываем картинки
		cvNamedWindow( "64");
		cvResize(res, dst3, CV_INTER_NN);
		cvShowImage( "64", dst3 );
		cvNamedWindow( "gray");
		cvResize(gray, dst1, CV_INTER_NN);
		cvShowImage( "gray", dst1 );
		cvNamedWindow( "bin");
		cvResize(bin, dst1, CV_INTER_NN);
		cvShowImage( "bin", dst1 );

		cvReleaseImage(&dst3);
		cvReleaseImage(&dst1);
	}

	// освобождаем ресурсы
	cvReleaseImage(&res);
	cvReleaseImage(&gray);
	cvReleaseImage(&bin);

	return hash;
}

// функция для рассчета второго 64-битного вектора
__int64 calcImageHash2(IplImage* src, bool show_results)
{
	if(!src){
		return 0;
	}
```

```
    // результирующее изображение, изображение в оттенках серого, бинарное изображение
    IplImage *res=0, *gray=0, *bin =0;

    res = cvCreateImage( cvSize(8, 8), src->depth, src->nChannels);
    gray = cvCreateImage( cvSize(8, 8), IPL_DEPTH_8U, 1);
    bin = cvCreateImage( cvSize(8, 8), IPL_DEPTH_8U, 1);

    // уменьшаем картинку
    cvResize(src, res);
    // переводим в оттенки серого
    cvCvtColor(res, gray, CV_BGR2GRAY);

    // построим хэш
    __int64 hash = 0;

    int i=0;
    // пробегаемся по всем пикселям изображения
    for( int y=0; y<gray->height; y++ ) {
        uchar* ptr = (uchar*) (gray->imageData + y * gray->widthStep);
      ptr[0]=0; i =1;
        for( int x=0; x<gray->width; x++ ) {

            if(ptr[x] < ptr[x-1])
          {
              hash |= 1i64<<i;
            }
            i++;
        }
    }

    printf("[i] hash: %I64X \n", hash);

    if(show_results){
        // увеличенные картинки для отображения результатов
        IplImage* dst3 = cvCreateImage( cvSize(128, 128), IPL_DEPTH_8U, 3);
        IplImage* dst1 = cvCreateImage( cvSize(128, 128), IPL_DEPTH_8U, 1);

        // показываем картинки
        cvNamedWindow( "64");
        cvResize(res, dst3, CV_INTER_NN);
        cvShowImage( "64", dst3 );
        cvNamedWindow( "gray");
        cvResize(gray, dst1, CV_INTER_NN);
        cvShowImage( "gray", dst1 );
        cvNamedWindow( "bin");
        cvResize(bin, dst1, CV_INTER_NN);
        cvShowImage( "bin", dst1 );

        cvReleaseImage(&dst3);
        cvReleaseImage(&dst1);
    }

    // освобождаем ресурсы
    cvReleaseImage(&res);
    cvReleaseImage(&gray);
    cvReleaseImage(&bin);

    return hash;
}

// функция для рассчёта расстояния Хэмминга
__int64 calcHammingDistance(__int64 x, __int64 y)
{
```

```
    __int64 dist = 0, val = x ^ y;

    while(val)
    {
        ++dist;
        val &= val - 1;
    }

    return dist;
}
```

Б3. Листинг параллельного алгоритма СНС

```
global _NN: label;

data ".myData"
end ".myData";

nobits ".my_data"
  pointers: word[51*51];
  pointers2: word[22*22];
  pointers3: word[8*8];
    map: word[51*51*4];
    map2: word[22*22*8];
    map3: word[8*8*16];
  n:word;
  window: word[50*50];
  window2: word[30*30];
  window3: word[8*8];
  output: word[100];
  buf: long;
  cc:word;
  c1: word;
  c2: word;
  c3: word;
  w1: word;
  w2: word;
end ".my_data";

begin ".text"
<_NN>
   ar5 = ar7 - 2;
   push ar0, gr0;
   push ar1, gr1;
  push ar2, gr2;
  push ar3, gr3;
  push ar4, gr4;
  push ar5, gr5;

   ar0 = [--ar5];     // input image
  ar1 = [--ar5];     // cores1
  ar2 = c1;
  [ar2] = ar1;

  ar1 = [--ar5]; //cores2
  ar2 = c2;
  [ar2] = ar1;

  ar1 = [--ar5]; //cores3
  ar2 = c3;
  [ar2] = ar1;

  ar1 = [--ar5]; //weights
  ar2 = w1;
  [ar2] = ar1;

  ar1 = [--ar5]; //weights2
  ar2 = w2;
  [ar2] = ar1;

  gr1 = 0;
  ar2 = pointers;

  // fill pointers

  gr3 = 50;
  gr5 = 49;
  <Pointer_Loop1>

    gr2 = 50;
    <Pointer_Loop2>
    [ar2] = gr1;
    gr1++;
    ar2++;
    gr2--;
    if >= goto Pointer_Loop2;

  gr3--;
  if >= delayed goto Pointer_Loop1;
  gr4 = gr1;
  gr1 = gr4 + gr5;


  // create windows and core

  ar6 = n;
  gr1 = 4;
  [ar6] = gr1;

  <Loop4maps>

  ar6 = pointers;
  gr6 = 51*51;
  ar5 = map;

    <Pointers>

      //create window
      ar2 = window; //window
      ar3 = ar0; //image
```

```
                //Skip to right position with pointer
                gr6 = [ar6];
                <Pointers_Skip>
                ar3++;
                gr1--;
                if > goto Pointers_Skip;

                // ar6 pointers

                gr1 = 50;
                <Window_Loop1>
                  gr2 = 50;
                  <Window_Loop2>
                  gr3 = [ar3];
                  [ar2] = gr3;
                  ar2++;
                  ar3++;

                  gr2--;
                  if > goto Window_Loop2;

                    //Skip 50 times
                    gr1 = 50;
                    <Skip50>
                    ar3++;
                    gr1--;
                    if > goto Skip50;

                  gr6--;
                if > goto Window_Loop1;


                // multiple

                sb = 03030303h;

                ar2 = window; //window
                ar3 = [c1]; //cores
                gr1 = 250;

                rep 8 wfifo  = [ar2++], ftw, wtw;
                rep 1 data  = [ar2] with vsum, data, 0;

                <Multiple_Loop>

                  rep 8 wfifo  = [ar2++], ftw, wtw;
                  rep 1 data  = [ar2] with vsum, data,
afifo;


                gr1--;
                if > goto Multiple_Loop;

                ar4 = buf;
                rep 1 [ar4] = afifo;

                  //Add afifo
                  gr2 = [ar4++];
                  gr3 = [ar4];
                  gr4 = gr2 + gr3;
                  [ar5++] = gr4;


                gr6--;
                  if > goto Pointers;

                ar6 = n;
                gr1 = [ar6];
                gr1--;
                [ar6] = gr1;
                if > goto Loop4maps;

                //create windows from maps 2

                // fill pointers

                gr3 = 30;
                gr5 = 29;
                <Pointer_Loop2_1>

                  gr2 = 30;
                  <Pointer_Loop2_2>
                  [ar2] = gr1;
                  gr1++;
                  ar2++;
                  gr2--;
                  if >= goto Pointer_Loop2_2;

                gr3--;
                if >= delayed goto Pointer_Loop2_1;
                gr4 = gr1;
                gr1 = gr4 + gr5;


                // create windows and core

                ar6 = n;
                gr1 = 2;
                [ar6] = gr1;

                <Loop8maps>

                ar6 = pointers2;
                gr6 = 22*22;
                ar5 = map2;

                  <Pointers2>

                    //create window
                    ar2 = window2; //window
                    ar3 = map; //image

                    //Skip to right position with pointer
                    gr6 = [ar6];
                    <Pointers_Skip2>
                    ar3++;
                    gr1--;
                    if > goto Pointers_Skip2;

                    // ar6 pointers

                    gr1 = 30;
                    <Window_Loop2_1>
                      gr2 = 30;
                      <Window_Loop2_2>
                      gr3 = [ar3];
                      [ar2] = gr3;
```

```
                ar2++;
                ar3++;

                gr2--;
                if > goto Window_Loop2_2;

                  //Skip 30 times
                  gr1 = 30;
                  <Skip30>
                  ar3++;
                  gr1--;
                  if > goto Skip30;

                gr6--;
              if > goto Window_Loop2_1;


              // multiple

              sb = 03030303h;

              ar2 = window2; //window
              ar3 = [c2]; //cores2
              gr1 = 90;

              rep 8 wfifo  = [ar2++], ftw, wtw;
              rep 1 data  = [ar2] with vsum, data, 0;

              <Multiple_Loop2>

                rep 8 wfifo  = [ar2++], ftw, wtw;
                rep 1 data  = [ar2] with vsum, data,
afifo;


              gr1--;
              if > goto Multiple_Loop2;

              ar4 = buf;
              rep 1 [ar4] = afifo;

                //Add afifo
                gr2 = [ar4++];
                gr3 = [ar4];
                gr4 = gr2 + gr3;
                [ar5++] = gr4;


              gr6--;
            if > goto Pointers2;

          ar6 = n;
          gr1 = [ar6];
          gr1--;
          [ar6] = gr1;
          if > goto Loop8maps;

          //create windows from maps 3

          // fill pointers

          gr3 = 15;
          gr5 = 14;
          <Pointer_Loop3_1>
            gr2 = 15;
            <Pointer_Loop3_2>
            [ar2] = gr1;
            gr1++;
            ar2++;
            gr2--;
            if >= goto Pointer_Loop3_2;

          gr3--;
          if >= delayed goto Pointer_Loop3_1;
          gr4 = gr1;
          gr1 = gr4 + gr5;


          // create windows and core

          ar6 = n;
          gr1 = 2;
          [ar6] = gr1;

          <Loop16maps>

          ar6 = pointers3;
          gr6 = 8*8;
          ar5 = map3;

            <Pointers3>

              //create window
              ar2 = window3; //window
              ar3 = map2; //image

              //Skip to right position with pointer
              gr6 = [ar6];
              <Pointers_Skip3>
              ar3++;
              gr1--;
              if > goto Pointers_Skip3;

              // ar6 pointers

              gr1 = 15;
              <Window_Loop3_1>
                gr2 = 15;
                <Window_Loop3_2>
                gr3 = [ar3];
                [ar2] = gr3;
                ar2++;
                ar3++;

                gr2--;
                if > goto Window_Loop3_2;

                  //Skip 15 times
                  gr1 = 15;
                  <Skip15>
                  ar3++;
                  gr1--;
                  if > goto Skip15;

                gr6--;
              if > goto Window_Loop3_1;
```

```
                // multiple

                sb = 03030303h;

                ar2 = window3; //window
                ar3 = [c3]; //cores2
                gr1 = 20;

                rep 8 wfifo  = [ar2++], ftw, wtw;
                rep 1 data  = [ar2] with vsum, data, 0;

                <Multiple_Loop3>

                  rep 8 wfifo  = [ar2++], ftw, wtw;
                  rep 1 data  = [ar2] with vsum, data,
afifo;

                gr1--;
                if > goto Multiple_Loop3;

                ar4 = buf;
                rep 1 [ar4] = afifo;

                  //Add afifo
                  gr2 = [ar4++];
                  gr3 = [ar4];
                  gr4 = gr2 + gr3;
                  [ar5++] = gr4;


                gr6--;
              if > goto Pointers3;

            ar6 = n;
            gr1 = [ar6];
            gr1--;
            [ar6] = gr1;
            if > goto Loop16maps;

            //calculate final vector

            ar0 = output;

            gr5 = 100;

            <NNLayer>
              ar1 = map3;
              ar2 = [w1];

              sb = 03030303h;
              rep 8 wfifo = [ar2++], ftw, wtw;
              ar1++;
              rep 1 data = [ar1] with vsum , data, 0;

              gr4 = 12;
              <NeuronLoop>
                rep 8 wfifo = [ar2++], ftw, wtw;
                rep 1 data = [ar1++] with vsum , data,
afifo;
              gr4--;
              if > goto NeuronLoop;

                ////////////////////////////////////function

                rep 1 [ar0++] = afifo;
              gr5--;
              if > goto NNLayer;

              ////////////// LAST
              gr5 = 100;
              <LastLayer>

              gr5--;
              if > goto LastLayer;

              pop ar5, gr5;
              pop ar4, gr4;
              pop ar3, gr3;
              pop ar2, gr2;
               pop ar1, gr1;
               pop ar0, gr0;
               return;


            end ".text";
```

Б4. Листинг функции создания обучающей выборки

```
namespace gabor
{
   public partial class mainDBForm : Form
   {
      private string path;
      private string directory_path;

      private Bitmap readAndresize(string fname, int w, int h)
      {
         Color def_clr = Color.FromArgb(100, 0, 0, 200);

         string path = fname;
         int x = 0; int y = 0;
         int[] mass;

         // fill array
         using (StreamReader sr = File.OpenText(path))
         {
            x = Convert.ToInt32(sr.ReadLine());
            y = Convert.ToInt32(sr.ReadLine());

            mass = new int[x * y];
            int position = 0;

            string s = "";
            string sub_s = "";
            s = sr.ReadLine();
            for (int i = 0; i < s.Length; i++)
            {
               if (s[i] != ' ')
               {
                  sub_s += s[i];
               }
               else
               {
                  mass[position] = Convert.ToInt32(sub_s);
                  sub_s = "";
                  position++;
               }
            }
            sr.Close();
         }
         //

         Bitmap img = new Bitmap(x, y);
         Bitmap mask = new Bitmap(img.Width, img.Height);

         // fill bitmap and mask

         int pointer = 0;

         for (int i = 0; i < x; i++)
            for (int j = 0; j < y; j++)
            {
               if (mass[pointer] == -1)
               {
                  img.SetPixel(i, j, Color.FromArgb(0, 0, 0, 0));
                  mask.SetPixel(i, j, Color.Empty);
                  pointer++;
```

```
        }
        else
        {
            img.SetPixel(i, j, Color.FromArgb(mass[pointer], mass[pointer], mass[pointer]));
            mask.SetPixel(i, j, def_clr);
            pointer++;
        }
    }
//

// создание уменьшенного изображения

Bitmap img_small = new Bitmap(w, h);
using (Graphics g = Graphics.FromImage(img_small))
{
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
    g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
    g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.HighQuality;
    Rectangle rect = new Rectangle(0, 0, w, h);
    g.DrawImage(img, rect, 0, 0, img.Width, img.Height, GraphicsUnit.Pixel);
}

//

// создание уменьшенной маски
Bitmap mask_small = new Bitmap(w, h);

using (Graphics g = Graphics.FromImage(mask_small))
{
    g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
    g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
    g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.HighQuality;
    Rectangle rect = new Rectangle(0, 0, w, h);
    g.DrawImage(mask, rect, 0, 0, img.Width, img.Height, GraphicsUnit.Pixel);
}

// нормализация уменьшенной маски
Bitmap mask_small_k = new Bitmap(w, h);
for (int i = 0; i < w; i++)
    for (int j = 0; j < h; j++)
    {
        if (mask_small.GetPixel(i, j).B > 10) { mask_small_k.SetPixel(i, j, def_clr); }
        else
        { mask_small_k.SetPixel(i, j, Color.Empty); }
    }

// save
Bitmap result = new Bitmap(w, h);

for (int i = 0; i < w; i++)
    for (int j = 0; j < h; j++)
    {
        if (mask_small_k.GetPixel(i, j) == def_clr)
        {
            result.SetPixel(i, j, img_small.GetPixel(i, j));
        }
        else
        {
            result.SetPixel(i, j, Color.FromArgb(0, 0, 0, 0));
        }
    }

return result;
```

```
}

private int readLength(string fname)
{
   int result = 0;

   using (StreamReader sr = File.OpenText(path))
   {
      result = Convert.ToInt32(sr.ReadLine());
      sr.Close();
   }

   return result;
}

private Bitmap resizeImageB(Bitmap img, int w, int h)
{
   Bitmap result = new Bitmap(w, h);

   using (Graphics g = Graphics.FromImage(result))
   {
      g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
      g.InterpolationMode = System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
      g.PixelOffsetMode = System.Drawing.Drawing2D.PixelOffsetMode.HighQuality;
      Rectangle rect = new Rectangle(0, 0, w, h);
      g.DrawImage(img, rect, 0, 0, img.Width, img.Height, GraphicsUnit.Pixel);
      g.Dispose();
   }

   return result;
}

public Bitmap readFromFile(string fname, int q)
{
   string path = fname;
   int x = 0; int y = 0;
   int[] mass;

   using (StreamReader sr = File.OpenText(path))
   {
      x = Convert.ToInt32(sr.ReadLine());
      y = Convert.ToInt32(sr.ReadLine());

      mass = new int[x * y];
      int position = 0;

      string s = "";
      string sub_s = "";
      s = sr.ReadLine();
      for (int i = 0; i < s.Length; i++)
      {
         if (s[i] != ' ')
         {
            sub_s += s[i];
         }
         else
         {
            mass[position] = Convert.ToInt32(sub_s);
            sub_s = "";
            position++;
         }
      }
      sr.Close();
```

```
            }

            Bitmap bitmap = new Bitmap(x, y);

            int w = bitmap.Width, h = bitmap.Height;
            int n = w * h;

            // Lock the bitmap's bits again.
            BitmapData bits = bitmap.LockBits(new System.Drawing.Rectangle(0, 0, bitmap.Width,
bitmap.Height), ImageLockMode.WriteOnly, bitmap.PixelFormat);
            IntPtr ptr = bits.Scan0;
            // Copy the array of color values into bitmap
            System.Runtime.InteropServices.Marshal.Copy(mass, 0, ptr, mass.Length);
            // Unlock the bits.
            bitmap.UnlockBits(bits);

            return bitmap;
        }

        public Bitmap readFromFile(string fname)
        {
            string path = fname;
            int x = 0; int y = 0;
            int[] mass;

            using (StreamReader sr = File.OpenText(path))
            {
                x = Convert.ToInt32(sr.ReadLine());
                y = Convert.ToInt32(sr.ReadLine());

                mass = new int[x * y];
                int position = 0;

                string s = "";
                string sub_s = "";
                s = sr.ReadLine();
                for (int i = 0; i < s.Length; i++)
                {
                    if (s[i] != ' ')
                    {
                        sub_s += s[i];
                    }
                    else
                    {
                        mass[position] = Convert.ToInt32(sub_s);
                        sub_s = "";
                        position++;
                    }
                }
                sr.Close();
            }

            Bitmap result = new Bitmap(x, y);

            int pointer = 0;

            for (int i = 0; i < x; i++)
                for (int j = 0; j < y; j++)
                {
                    if (mass[pointer] == -1)
                    {
                        result.SetPixel(i, j, Color.FromArgb(0, 0, 0, 0));
                        pointer++;
```

```
            }
            else
            {
                result.SetPixel(i, j, Color.FromArgb(mass[pointer], mass[pointer], mass[pointer]));
                pointer++;
            }
        }
    return result;
}

public string imageToString(Bitmap img)
{
    string result = "";
    int R = 0; int G = 0; int B = 0; int S = 0;

    for (int i = 0; i < img.Width; i++)
    {
        for (int j = 0; j < img.Height; j++)
        {
                R = img.GetPixel(i, j).R;
                G = img.GetPixel(i, j).G;
                B = img.GetPixel(i, j).B;
                S = (R + G + B) / 3;
                result += S.ToString();
                result += " ";
        }
    }
    return result;
}

public mainDBForm()
{
    InitializeComponent();

    path = System.IO.Directory.GetCurrentDirectory();
    string[] objects = System.IO.Directory.GetDirectories(path + "\\objects");
    string[] backgrounds = System.IO.Directory.GetDirectories(path + "\\backgrounds");
    MessageBox.Show(path + "\\backgrounds");

    int count = objects.Length;

    for (int i = 0; i < objects.Length; i++)
    {
        string old_str = objects[i];
        int last_pos = old_str.LastIndexOf('\\');
        old_str = old_str.Substring(last_pos + 1);
        objects[i] = old_str;
    }

    listBox1.SelectionMode = SelectionMode.MultiSimple;
    listBox1.BeginUpdate();
    for (int i = 0; i < count; i++)
    {
        listBox1.Items.Add(objects[i]);
    }
    listBox1.EndUpdate();

    count = backgrounds.Length;

    for (int i = 0; i < backgrounds.Length; i++)
    {
        string old_str = backgrounds[i];
        int last_pos = old_str.LastIndexOf('\\');
```

```
            old_str = old_str.Substring(last_pos + 1);
            backgrounds[i] = old_str;
        }

        listBox2.SelectionMode = SelectionMode.MultiSimple;
        listBox2.BeginUpdate();
        for (int i = 0; i < count; i++)
        {
            listBox2.Items.Add(backgrounds[i]);
        }
        listBox2.EndUpdate();
    }

    private void button2_Click(object sender, EventArgs e)
    {
        saveFileDialog1.InitialDirectory = path;
        saveFileDialog1.Filter = "object files (*.obj)|*.obj|All files (*.*)|*.*";
        saveFileDialog1.FilterIndex = 2;
        saveFileDialog1.FileName = "object";
        DialogResult dlg = saveFileDialog1.ShowDialog();
    }

    private void saveFileDialog1_FileOk(object sender, CancelEventArgs e)
    {
        textBox3.Text = saveFileDialog1.FileName;
        directory_path = saveFileDialog1.FileName;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (directory_path != null)
        {
            int progress = 0;
            progressBar1.Value = progress;

            #region nq
            string save_path = saveFileDialog1.FileName + "\\";

            //получить выбранные элементы
            ListBox.SelectedObjectCollection l1 = listBox1.SelectedItems;
            ListBox.SelectedObjectCollection l2 = listBox2.SelectedItems;

            string[] object_folders = new string[l1.Count];
            string[] background_folders = new string[l2.Count];
            int current = 0;

            foreach (Object selectedItem in listBox1.SelectedItems)
            {
                object_folders[current] = selectedItem as String;
                current++;
            }
            current = 0;
            foreach (Object selectedItem in listBox2.SelectedItems)
            {
                background_folders[current] = selectedItem as String;
                current++;
            }

            //определить переменные количества
            int N1 = object_folders.Length; int N2 = background_folders.Length; //количества папок об-в
и фонов
            int M1 = 200;
            int M2 = 180;
```

```
                    //создать общий массив

                    int NN1 = 0; //общее количество объектов
                    int NN2 = 0; // общее количество фонов

                    for (int i = 0; i < N1; i++)
                    {
                       NN1 += new DirectoryInfo(path + "\\objects\\" + object_folders[i]).GetFiles().Length;
                    }

                    for (int i = 0; i < N2; i++)
                    {
                       NN2 += new DirectoryInfo(path + "\\backgrounds\\" +
background_folders[i]).GetFiles().Length;
                    }

                    int empty_length = new DirectoryInfo(path + "\\empty\\").GetFiles().Length;

                    //создание массивов
                    byte[] data_array = new byte[96 * 96];
                    Bitmap data_image;
                    Bitmap data_image_small = new Bitmap(96,96);
                    Bitmap[] object_images = new Bitmap[NN1];
                    Bitmap[] background_images = new Bitmap[NN2];
                    Bitmap[] empty_images = new Bitmap[empty_length];

                    //заполнение массивов имен файлов
                    string[] object_names = new string[NN1]; //все имена объектов
                    int[] object_codes = new int[NN1];
                    int[] object_lengths = new int[NN1];
                    string[] background_names = new string[NN2]; //все имена фонов
                    string[] empty_names = new string[empty_length];

                    int[] data_array2 = new int[NN1 * NN2 + 1 + background_images.Length +
empty_images.Length];
                    data_array2[0] = (int)(NN1 * NN2 + empty_images.Length + background_images.Length);

                    current = 0;
                    for (int i = 0; i < N1; i++)
                    {
                       FileInfo[] dirs = new DirectoryInfo(path + "\\objects\\" + object_folders[i]).GetFiles();

                       foreach (FileInfo element in dirs)
                       {
                          object_names[current] = element.FullName;
                          object_codes[current] = i;
                          current++;
                       }
                    }

                    current = 0;
                    for (int i = 0; i < N2; i++)
                    {
                       FileInfo[] dirs = new DirectoryInfo(path + "\\backgrounds\\" +
background_folders[i]).GetFiles();

                       foreach (FileInfo element in dirs)
                       {
                          background_names[current] = element.FullName;
                          current++;
                       }
                    }
```

```
current = 0;
FileInfo[] empty_dirs = new DirectoryInfo(path + "\\empty\\").GetFiles();
foreach (FileInfo element in empty_dirs)
{
   empty_names[current] = element.FullName;
   current++;
}

//считывание и заполнение массива изображений объектов

for (int i = 0; i < NN1; i++)
{
   //object_lengths[i] = readLength(object_names[i]);
   //M2 = object_lengths[i];
   object_images[i] = readFromFile(object_names[i]);
   object_lengths[i] = object_images[i].Width;
}

//
for (int i = 0; i < NN2; i++)
{
   M1 = 200;
   //M1 = readLength(background_names[i]);
   background_images[i] = resizeImageB(readFromFile(background_names[i]), M1, M1);
}

/////
string image_save_path = directory_path + "\\images\\";

if (!Directory.Exists(image_save_path))
{
   Directory.CreateDirectory(image_save_path);
}
/////

//Генерация изображений выборки
current = 0;
int current2 = 1;
int diff = (M1 - M2) / 2;

//создание файлов выборок

byte[] _size = new byte[4];
//int intSize = NN1 * NN2 + empty_images.Length + background_images.Length;
int intSize = NN1 * NN2;

_size = BitConverter.GetBytes(intSize);

using (FileStream fs = new FileStream(save_path + ".dat", FileMode.Append))
{
   using (BinaryWriter bw = new BinaryWriter(fs))
   {
      bw.Write(_size);
      bw.Write((byte)96);
   }
}

int current_image = 0;

for (int i = 0; i < NN1; i++)
{
   for (int j = 0; j < NN2; j++)
   {
```

```
                        M2 = object_lengths[i];
                        M1 = 200;
                        diff = (M1 - M2) / 2;

                        data_image = new Bitmap(M1, M1);
                        //data_images[current] = background_images[j];
                        data_array2[current2] = (int)object_codes[i];
                        current2++;

                        //Цикл можно удалить
                        for (int ii = 0; ii < M1; ii++)
                           for (int jj = 0; jj < M1; jj++)
                           {
                              data_image.SetPixel(ii, jj, background_images[j].GetPixel(ii, jj));
                           }

                        for (int ii = 0; ii < M2; ii++)
                           for (int jj = 0; jj < M2; jj++)
                           {
                              if (object_images[i].GetPixel(ii, jj) != Color.FromArgb(0, 0, 0, 0))
                              {
                                 Color clr = object_images[i].GetPixel(ii, jj);
                                 data_image.SetPixel(ii + diff, jj + diff, clr);
                              }
                           }

                        // Уменьшение

                           data_image_small = new Bitmap(96, 96);
                           data_image_small = resizeImageB(data_image, 96, 96);
                           data_image_small.Save(image_save_path + current_image + ".png",
System.Drawing.Imaging.ImageFormat.Png);

                        //Сохранение в файл
                           current = 0;
                           using (FileStream fs = new FileStream(save_path + ".dat", FileMode.Append))
                           {
                              for (int ii = 0; ii < data_image_small.Width; ii++)
                              {
                                 for (int jj = 0; jj < data_image_small.Height; jj++)
                                 {
                                    int R = data_image_small.GetPixel(ii, jj).R;
                                    int G = data_image_small.GetPixel(ii, jj).G;
                                    int B = data_image_small.GetPixel(ii, jj).B;
                                    int S = (R + G + B) / 3;
                                    data_array[current] = (byte)S;
                                    current++;
                                 }
                              }

                              for (int cnt = 0; cnt < data_array.Length; cnt++)
                              {
                                 fs.WriteByte(data_array[cnt]);
                              }
                               using (BinaryWriter bw = new BinaryWriter(fs))
                                     {
                                        for (int cnt = 0; cnt < data_array.Length; cnt++)
                                        {
                                          bw.Write(data_array[cnt]);
                                        }
                                     }
                           }
```

```
                current++;
                current_image++;
            }
        }

        Bitmap empty_buffer = new Bitmap(200, 200);
        int empty_num = object_folders.Length;

        Bitmap empty_buffer_small = new Bitmap(96, 96);

        for (int i = 0; i < empty_names.Length; i++)
        {
            empty_buffer = readFromFile(empty_names[i]);
            empty_buffer_small = resizeImageB(empty_buffer, 96, 96);

            empty_buffer_small.Save(image_save_path + current_image + ".png",
System.Drawing.Imaging.ImageFormat.Png);

            using (FileStream fs = new FileStream(save_path + ".dat", FileMode.Append))
            {
                current = 0;
                for (int ii = 0; ii < empty_buffer_small.Width; ii++)
                {
                    for (int jj = 0; jj < empty_buffer_small.Height; jj++)
                    {
                        int R = empty_buffer_small.GetPixel(ii, jj).R;
                        int G = empty_buffer_small.GetPixel(ii, jj).G;
                        int B = empty_buffer_small.GetPixel(ii, jj).B;
                        int S = (R + G + B) / 3;
                        data_array[current] = (byte)S;
                        current++;
                    }
                }

                for (int cnt = 0; cnt < data_array.Length; cnt++)
                {
                    fs.WriteByte(data_array[cnt]);
                }
            }

            data_array2[current2] = (int)10;
            current2++;
            current_image++;
        }

        for (int i = 0; i < background_images.Length; i++)
        {
            empty_buffer = background_images[i];

            empty_buffer_small = resizeImageB(empty_buffer, 96, 96);

            empty_buffer_small.Save(image_save_path + current_image + ".png",
System.Drawing.Imaging.ImageFormat.Png);

            using (FileStream fs = new FileStream(save_path + ".dat", FileMode.Append))
            {
                current = 0;
                for (int ii = 0; ii < empty_buffer_small.Width; ii++)
                {
                    for (int jj = 0; jj < empty_buffer_small.Height; jj++)
                    {
                        int R = empty_buffer_small.GetPixel(ii, jj).R;
                        int G = empty_buffer_small.GetPixel(ii, jj).G;
```

```
                int B = empty_buffer_small.GetPixel(ii, jj).B;
                int S = (R + G + B) / 3;
                data_array[current] = (byte)S;
                current++;
            }
        }

        for (int cnt = 0; cnt < data_array.Length; cnt++)
        {
            fs.WriteByte(data_array[cnt]);
        }
    }

    data_array2[current2] = empty_num;
    current2++;
    current_image++;
}

byte[] data_array3 = new byte[data_array2.Length * 4];
current = 0;
for (int i = 0; i < data_array3.Length; i += 4)
{
    intSize = data_array2[current];
    _size = BitConverter.GetBytes(intSize);

    data_array3[i] = _size[0];
    data_array3[i+1] = _size[1];
    data_array3[i+2] = _size[2];
    data_array3[i+3] = _size[3];
    current++;
}

File.WriteAllBytes(save_path + ".cat", data_array3);

using (StreamWriter sw = File.CreateText(save_path + "2.cat"))
{
    for (int i = 0; i < data_array2.Length; i++)
    {
        sw.Write(data_array2[i]);
    }
}

using (StreamWriter sw = File.CreateText(save_path + ".info"))
{
    for (int i = 0; i < object_folders.Length; i++)
    {
        sw.WriteLine((i).ToString());
        sw.WriteLine(object_folders[i]);
    }
    sw.WriteLine(object_folders.Length.ToString());
    sw.WriteLine("empty class");
    sw.Close();
}
#endregion nq
}
else
{
    MessageBox.Show("Директория не задана");  }}}}
```

Приложение В – Примеры того, как можно применить разработанные алгоритмы для распознавания объектов на изображениях

Визуализация выходных данных:

Объект: мяч (0)
Координаты: 196, 124, 368, 304

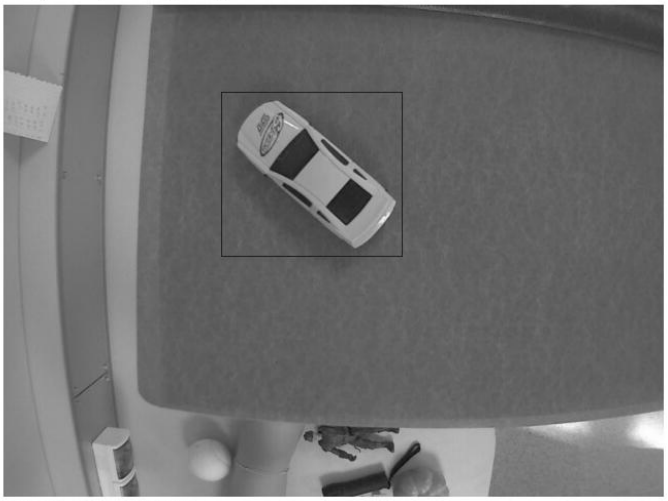
Рисунок В.1 – Пример распознавания №1

Визуализация выходных данных:

Объект: мяч (0)
Координаты: 197, 121, 366, 295

Рисунок В.2 – Пример распознавания №2

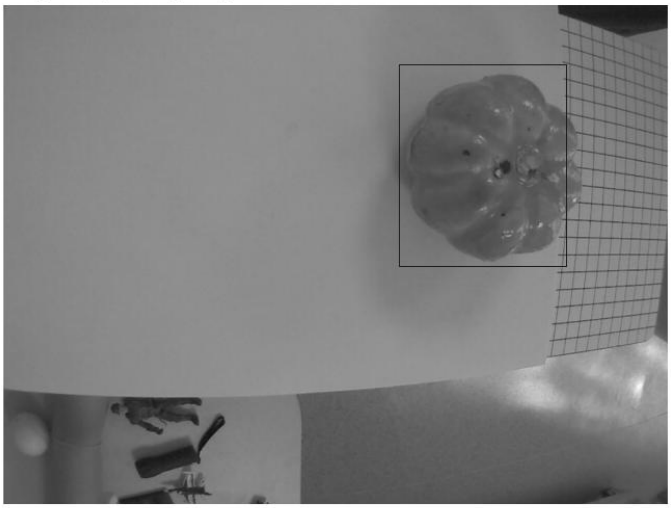
Визуализация выходных данных:



Объект: автомобиль (4)
Координаты: 221, 88, 392, 250

Рисунок В.3 – Пример распознавания №3

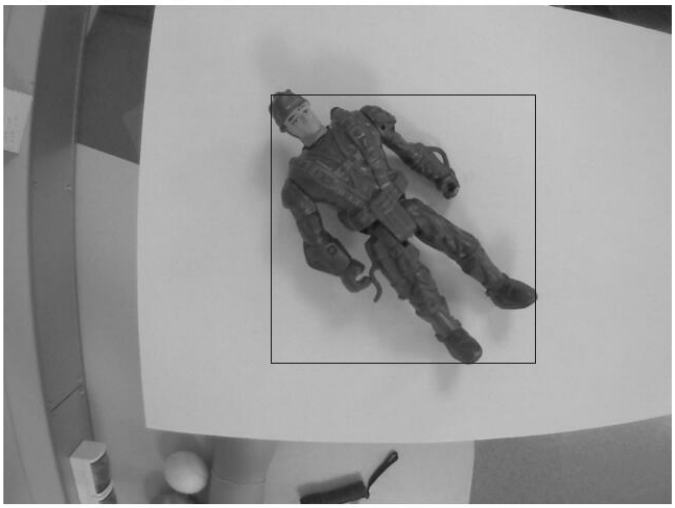
Визуализация выходных данных:



Объект: тыква (2)
Координаты: 388, 61, 532, 247

Рисунок В.4 – Пример распознавания №4

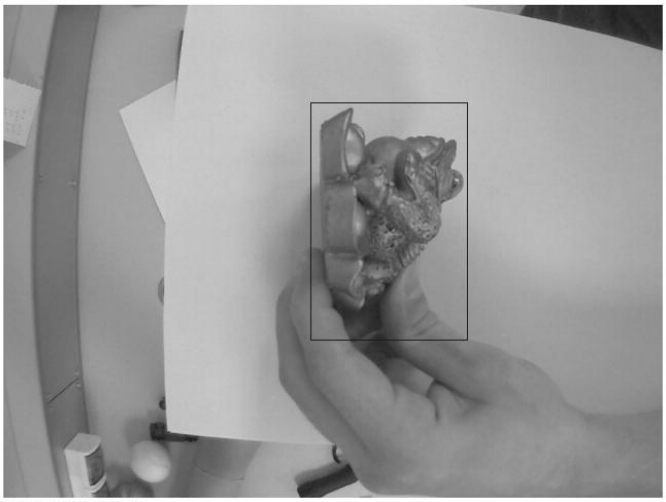
Визуализация выходных данных:



Объект: человек (6)
Координаты: 256, 83, 489, 342

Рисунок В.5 – Пример распознавания №5

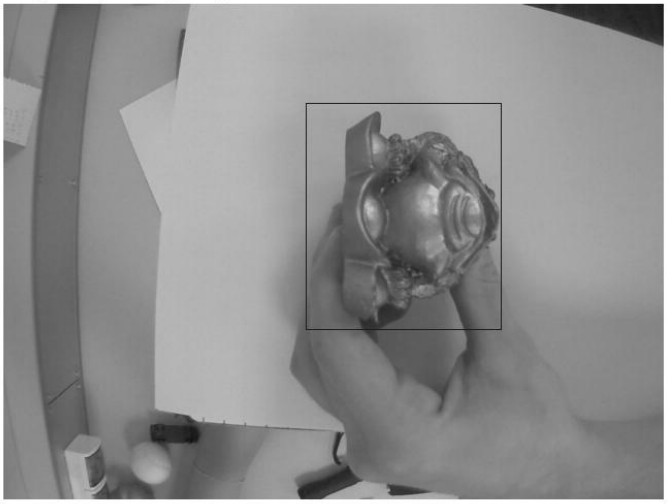
Визуализация выходных данных:



Объект: жаба (5)
Координаты: 307, 98, 427, 336

Рисунок В.6 – Пример распознавания №6

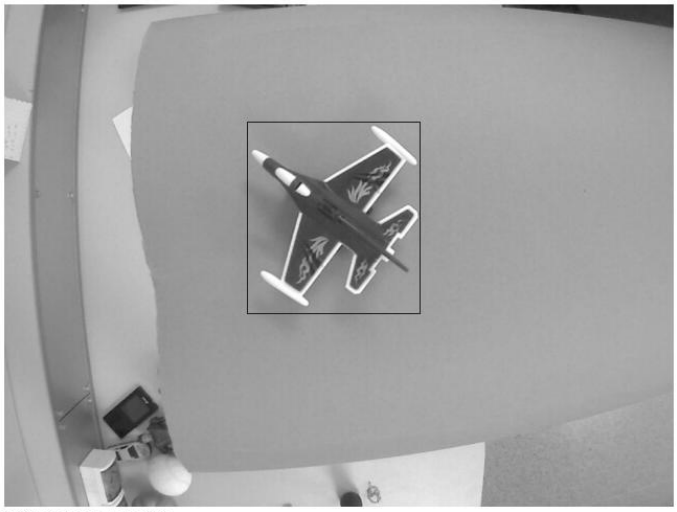
Визуализация выходных данных:



Объект: жаба (5)
Координаты: 307, 107, 481, 337

Рисунок В.7 – Пример распознавания №7

Визуализация выходных данных:



Объект: самолет (3)
Координаты: 227, 119, 385, 282

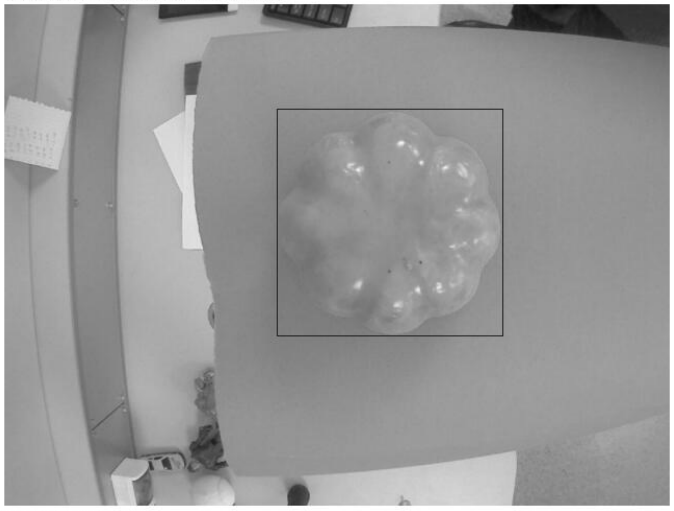
Рисунок В.8 – Пример распознавания №8

Визуализация выходных данных:

Объект: плеер (7)
Координаты: 298, 164, 498, 262

Рисунок В.9 – Пример распознавания №9

Визуализация выходных данных:



Объект: тыква (2)
Координаты: 261, 104, 460, 311

Рисунок В.10 – Пример распознавания №10