МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра Прикладная математика и информатика
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Методы и алгоритмы анализа изображений при помощи нейронных сетей»

Обучающийся С.В. Верзун
(Инициалы Фамилия) (личная подпись)

Научный руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Введение

Введение

Данное исследование посвящено разработке модели классификации изображений с использованием нейронных сетей. Выбранная проблема актуальна и значима, поскольку классификация изображений становится все более важной в различных областях, так как объемы данных увеличиваются из года в год и тратятся огромные человеческие ресурсы на обработку и классификацию этих данных. Текущее исследование нацелено на автоматизацию этого процесса и снижения нагрузки на человека.

Классификация изображений является очень важной задачей в компьютерном зрении, которая позволяет компьютерам распознавать и классифицировать изображения на основе их визуального содержания. Эта задача имеет множество применений в различных областях деятельности человека. Например, модели классификации изображений могут использоваться в здравоохранении для анализа медицинских изображений, в сфере безопасности для обнаружения объектов или людей, а в электронной коммерции для автоматической сортировки загруженных изображений.

Актуальность данной исследовательской работы заключается в растущей важности классификации изображений в различных областях, таких как, например, здравоохранение, безопасность и электронная коммерция. По мере того, как данные изображений становятся все более доступными, потребность в точных и эффективных моделях классификации изображений становится все более важной. Данная исследовательская работа направлена на удовлетворение этой потребности путем разработки надежной модели классификации изображений, которая улучшает существующие базовые модели.

Объектом исследования данной научной работы является классификация изображений при помощи нейронных сетей.

Предметом исследования данной научной работы являются методы и алгоритмы анализа изображений при помощи нейронных сетей.

Целью данной работы является разработка модели классификации изображений при помощи нейронных сетей, которая обеспечит повышение эффективности анализа изображений.

Гипотеза исследования: применение предлагаемой модели классификации изображений при помощи нейронных сетей позволит повысить эффективность анализа изображений.

Для достижения поставленной цели и проверки гипотезы необходимо решить следующие задачи:

- исследовать существующие методы и инструменты для классификации изображений, включая методы компьютерного зрения и методы машинного обучения;
- проанализировать алгоритмы классификации изображений с использованием нейронных сетей, включая выбор подходящих моделей нейронных сетей, функций активации, оптимизаторов и метрик оценки;
- разработать и протестировать выбранные модели, включая сравнение различных моделей, выбор гиперпараметров и оценку алгоритмов на тестовых данных;
- разработать стартап-проект на основе реализованной модели классификации изображений, включая описание идеи проекта, технологический аудит, анализ рыночных возможностей, разработку рыночной стратегии и маркетинговой программы.

Методы исследования: визуализация данных, математическая статистика, ранжирование, обзор литературы, эксперименты и внедрение программного обеспечения.

Данное исследование проводилось с 2021 по 2023 год в несколько этапов:

- первый этап включал изучение существующих методов и инструментов для классификации изображений, был проведен анализ алгоритмов классификации изображений с использованием нейронных сетей;

– второй этап включал разработку и тестирование выбранных базовых моделей, а также программную реализацию и тестирование разработанной модели;
– третий - разработку стартап-проекта на основе разработанной модели классификации изображений.

Новизна исследования заключается в разработке модели классификации изображений при помощи нейронных сетей, которая обеспечит повышение эффективности анализа изображений для существующих моделей.

Данное исследование является значимым в теоретическом смысле, поскольку оно вносит вклад в базу знаний по классификации изображений, компьютерному зрению и машинному обучению.

Практическая значимость данного исследования заключается в том, что разработанная модель классификации изображений может быть использована в различных областях жизни человека для автоматизации сортировки больших объемов изображений.

Публикация по теме исследования:

Верзун С. В. «Применение нейронной сети для стилизованной обработки изображений» / научный журнал «Вестник магистратуры», 2022. №12 (135) ISSN 2223–4047.

На защиту выносится:

– модель автоматической классификации изображений при помощи нейросетей;
– результаты применения предлагаемой модели в качестве основы для разработки стартап-проекта классификации изображений

Диссертационная работа состоит из введения, четырех глав, заключения и списка используемой литературы.

В первой главе представлен обзор актуальности темы исследования и существующих методов и инструментов для классификации изображений, которые основаны на компьютерном зрении и методах машинного обучения,

рассматриваются различные подходы к классификации изображений, выделяются их преимущества и недостатки.

Вторая глава посвящена анализу алгоритмов классификации изображений с использованием нейронных сетей. Проводится обзор нейронных сетей, обсуждается выбор архитектуры для классификации изображений, функции активации в выбранных архитектурах, выбор оптимизатора и изучение различных метрик для оценки и улучшения модели, описываются тестовые данные и оценка алгоритмов, а также детали программной реализации.

В третьей главе диссертации описывается разработка и тестирование выбранных моделей, сравниваются результаты и обсуждается метод выбора гиперпараметров. В ней также представлены детали программной реализации и тестирование разработанной модели.

Четвертая глава диссертации посвящена разработке стартап-проекта, где описывается идея проекта и технологический аудит проекта, анализ рыночных возможностей и разработка рыночной стратегии и маркетинговой программы для проекта.

В заключении диссертации обобщены основные результаты исследования и даны рекомендации для будущих исследований в области классификации изображений при помощи нейронных сетей.

Работа состоит из 83 страниц, 26 рисунков, 30 таблиц и 35 источников.

Ключевые слова: машинное обучение, нейронные сети, классификация изображений.

Глава 1 Исследование существующих методов и средств классификации изображений

1.1 Актуальность темы и обзор существующих решений

В последнее десятилетие было опубликовано значительное количество статей и работ об алгоритмах классификации изображений в различных областях применения, от классификации предметов быта, животных до классификации раковых опухолей или снимков из космоса [1].

В здравоохранении медицинские изображения, такие как рентгеновские, компьютерные и магнитно-резонансные снимки, анализируются для выявления и диагностики различных заболеваний. Анализ изображений помогает медицинским работникам ставить более точные диагнозы, выявлять заболевания на ранних стадиях и следить за развитием болезни. Это может привести к улучшению состояния пациентов, снижению расходов на здравоохранение и повышению качества жизни пациентов [10].

В сфере безопасности анализ изображений используется для различных целей, таких как распознавание объектов, распознавание лиц и биометрическая идентификация. Анализ изображений может помочь правоохранительным органам идентифицировать подозреваемых и предотвратить преступления. Он также может использоваться для пограничного и иммиграционного контроля, выявления несанкционированного въезда и предотвращения нелегальной иммиграции [7].

В видеонаблюдении анализ изображений используется для мониторинга общественных мест, таких как аэропорты, вокзалы и торговые центры. Анализ изображений позволяет обнаружить подозрительные действия и предупредить сотрудников службы безопасности, что приводит к более эффективному и действенному реагированию на потенциальные угрозы [11].

На данный момент глубинное обучение является основой многих бизнес-процессов крупных международных компаний:

- компания Google использует обработку естественного языка (NLP) для понимания и ответа на запросы пользователей, а Google Photos использует глубокое обучение для идентификации и систематизации фотографий;
- компания Amazon использует ИИ во многих своих продуктах, включая Amazon Echo, который использует НЛП для ответа на голосовые команды. Система рекомендаций Amazon также использует глубокое обучение, чтобы предлагать покупателям товары на основе их предыдущих покупок и истории просмотров;
- компания Netflix использует ИИ для персонализации просмотра для каждого пользователя. Его рекомендательная система использует алгоритмы машинного обучения, чтобы предлагать фильмы и телешоу на основе истории просмотров пользователя;
- компания Tesla использует глубокое обучение для работы технологии самоуправляемых автомобилей. Система «Автопилот» использует нейронные сети для анализа данных с камер и датчиков в режиме реального времени, чтобы обнаруживать препятствия на дороге и реагировать на них;
- компания IBM разработала платформу Watson AI для обработки естественного языка и использует машинное обучение для понимания и ответа на запросы пользователей;
- компания Microsoft использует во многих своих продуктах, включая Cortana, виртуального помощника, и Microsoft Translator, который использует глубокое обучение для перевода текста и речи в режиме реального времени;
- компания Uber использует в своем сервисе поездок для подбора водителей и пассажиров и оптимизации маршрутов для повышения эффективности. Подразделение автономных автомобилей Uber ATG

также использует глубокое обучение для разработки самоуправляемых автомобилей.

Классификации изображений решает главную задачу - принять на вход исходное изображение и отнести его к определенному классу (собака, птица, кошка и др.) или к группе возможных классов, которая наиболее полно подходит к данному объекту.

1.2 Классификация изображений с помощью методов компьютерного зрения

Классификация изображений — это фундаментальная задача компьютерного зрения, которая предполагает отнесение входного изображения к одному из нескольких заранее определенных классов [14]. Традиционные методы компьютерного зрения для классификации изображений включают последовательность этапов обработки, таких как извлечение признаков, уменьшение размерности и классификация [33].

Выделение признаков включает в себя извлечение дискриминационных признаков из входного изображения, которые могут помочь отличить его от других изображений [15]. Сокращение размерности включает в себя уменьшение размерности пространства признаков, чтобы сделать его более вычислительным. Классификация предполагает обучение классификатора на уменьшенном пространстве признаков для отнесения изображения к одному из заданных классов.

Существует несколько традиционных методов компьютерного зрения для классификации изображений, таких как машины опорных векторов (SVM), k-nearest neighbors (k-NN) и деревья решений [18]. Эти методы широко используются и показали свою эффективность для широкого круга задач классификации изображений. Однако они имеют ряд недостатков, таких как необходимость ручной разработки признаков, неспособность к обучению сложным признакам и ограниченная способность к обобщению [34].

Подходы на основе нейронных сетей стали мощной альтернативой традиционным методам компьютерного зрения для классификации изображений [13]. Нейронные сети могут автоматически изучать сложные признаки на основе входного изображения и показали высокую эффективность при решении широкого круга задач классификации изображений. В последние годы глубокие нейронные сети, такие как сверточные нейронные сети (CNN), достигли самых высоких результатов на нескольких эталонных наборах данных, таких как ImageNet.

Признаки Хаара – признаки цифрового изображения, используемые в распознавании образов. Своим названием они обязаны интуитивным сходствам с вейвлетами Хаара [17]. Признаки Хаара использовались в первом детекторе лиц, работающих в реальном времени, их можно видеть на рисунке 1.

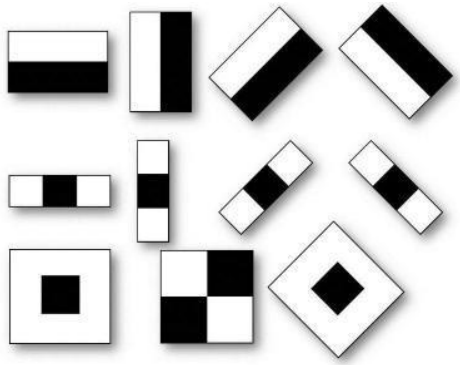

Рисунок 1 – Признаки Хаара

Традиционно алгоритмы, которые полагаются исключительно на интенсивность пикселей, например значения RGB, известны своей высокой вычислительной сложностью. Папагеоргиу [16] предложил решение, включающее использование нескольких признаков на основе вейвлета Хаара. Виола и Джонс [19] приняли эту идею и разработали признаки Хаара, которые состоят из смежных прямоугольных областей, расположенных на изображении. Интенсивности пикселей этих областей складываются вместе, и

вычисляется разница между суммами. Это значение представляет собой определенный признак, например размер и положение на изображении.

Ключевым преимуществом признаков Хаара является их высокая скорость по сравнению с другими типами признаков [32]. При использовании интегрального представления изображения признаки Хаара могут быть вычислены за постоянное время, составляющее около 60 инструкций процессора на один признак из двух областей [23].

1.3 Классификация изображений с помощью методов машинного обучения

Машинное обучение успешно используется в задачах компьютерного зрения, работы с текстами, медицины, беспилотного управления дронов и т. д. Основой для многих из этих приложений являются методы и средства искусственного интеллекта, такие как классификация, локализация и обнаружение.

За несколько предыдущих лет нейронные сети очень хорошо показали себя в задачах, где данные представлены в виде текстов или изображений, видео, поэтому сейчас их используют довольно часто для таких типов заданий. Для задач классификации можно использовать классические методы машинного обучения, такие как: наивный байесовский классификатор, метод опорных векторов, логистическую регрессию, случайные леса или метод стимулирования градиента.

По реальным данным, в том числе и в этой работе, лучшая точность была получена с помощью сверточных нейронных сетей. Подробнее об этих алгоритмах в следующей главе. Но стоит отметить, что на наборе данных ImageNet - набор данных, по состоянию на 2022 год, там было 14 197 122 изображения, разбитых на 21 841 категорию, с помощью различных архитектур нейронных сетей и подходов было получено на 2022 год точность более 95%. Пример набора данных показан на рисунке 2.

Рисунок 2 – Набор данных ImageNet

Итак, классические методы машинного обучения, такие как наивный Байесовский классификатор, метод опорных векторов, логистическую регрессию, случайные леса или метод стимулирования градиента, а также методы глубинного обучения, дают хорошие результаты для классификации изображений.

Конечно, метод следует выбирать на основании специфики данных и других характеристик, таких как - удовлетворительная точность алгоритма, время предвидения и тд. Обычно на реальных данных предпочитают сверточные нейронные сети, так как классические методы не способны выявить все необходимые закономерности.

1.4 Комбинация методов компьютерного зрения и нейронных сетей

Известно, что традиционные алгоритмы машинного обучения для классификации цифровых изображений в базах данных занимают много времени и сил из-за большого количества изображений и деталей, которые необходимо проанализировать [31]. Кроме того, эти алгоритмы часто

нестабильны, когда речь идет о классификации изображений из больших баз данных.

Существующие системы хранения изображений, такие как QBIC [25] и Visual SEEK [21], попытались решить эти проблемы, ограничив методы классификации описанием изображений на основе информации о форме, текстуре и цвете [26]. Однако эти системы имеют ограничения, когда речь идет о точной классификации изображений. Для решения этих проблем был предложен метод, сочетающий вейвлет-преобразование Хаара с алгоритмом нейронной сети для классификации цифровых изображений из базы данных. Алгоритм делит цветное изображение на три RGB-компонента и использует цветовые моменты первого порядка и коэффициенты разложения вейвлет-преобразования Хаара этих компонентов в качестве входного вектора для многослойной нейронной сети. Сеть обучается с помощью алгоритма обратного распространения ошибки.

Алгоритм использует два основных признака для представления содержания цифровых изображений: цветовые моменты и коэффициенты вейвлет-разложения [29]. Цветовые моменты — это статистические показатели, описывающие цветовое распределение изображения, такие как среднее значение, дисперсия и перекос цветовых каналов. Было установлено, что цветовые моменты первого, второго и третьего порядка эффективны для представления цветового распределения изображений (рисунок 3) [30].

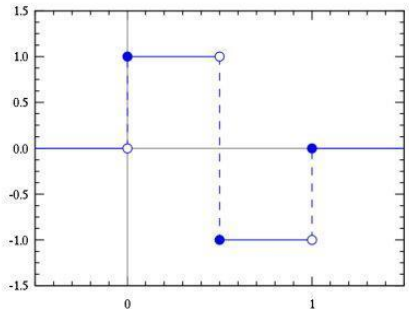

Рисунок 3 – Вейвлет Хаара

Вейвлет-разложение — это математическая техника, используемая для разложения сигнала на различные частотные компоненты [28]. Алгоритм использует вейвлет-преобразование Хаара, которое является разновидностью вейвлет-преобразования и было признано эффективным при анализе изображений.

Преобразование позволяет разделить изображение на четыре частотных диапазона (квадранты): LLn, LHn, HLn и HHn, где L - низкие частоты, H - высокие частоты, а n - уровень разложения. Квадрант LLn представляет изображение с низким разрешением, а HLn, LHn и HHn - вертикальные, горизонтальные и диагональные детали изображения соответственно.

Алгоритм использует вейвлет-преобразование Хаара [27] на шести уровнях разложения, а полученные вейвлет-коэффициенты подаются на вход нейронной сети. Затем нейронная сеть использует эти коэффициенты для классификации изображений в базе данных (рисунок 4).

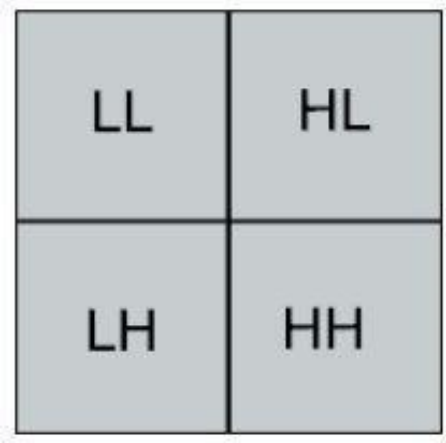

Рисунок 4 – Одноразовое применение двумерного вейвлет-преобразования к квадратному изображению

В работе [2] было отмечено, что точность классификации этим методом достигает 88%. Этот метод обещает решить проблемы, связанные с традиционными алгоритмами машинного обучения при классификации цифровых изображений в базах данных.

Выводы по главе 1

В этой главе рассмотрены методы, используемые для классификации изображений.

Методы компьютерного зрения предполагают использование алгоритмов, которые анализируют визуальное содержимое изображения и извлекают признаки, которые могут быть использованы для классификации.

Методы машинного обучения, с другой стороны, предполагают использование статистических моделей для изучения закономерностей из данных и использования этих знаний для классификации новых изображений.

Также рассмотрены сочетания этих методов с нейронными сетями, которые предоставляют эффективные алгоритмы для классификации изображений.

Нейронные сети особенно полезны для анализа изображений, поскольку они могут изучать сложные закономерности и взаимосвязи между различными характеристиками изображения.

Глава 2 Анализ алгоритмов классификации изображений при помощи нейронных сетей

2.1 Нейронные сети

2.1.1 Перцептрон

«Перцептрон представляет собой одну из первых в общем понимании моделей нейронных сетей. Даже несмотря на свою относительную простоту, данная модель способна самообучаться и решать сложные комплексные задачи. Главная математическая задача, которую он может выполнять - проводить линейное разделение различных нелинейных множеств, другими словами обеспечивает линейную сепарабельность [12]». На рисунке 5 представлен пример типичной архитектуры многослойного перцептрона.

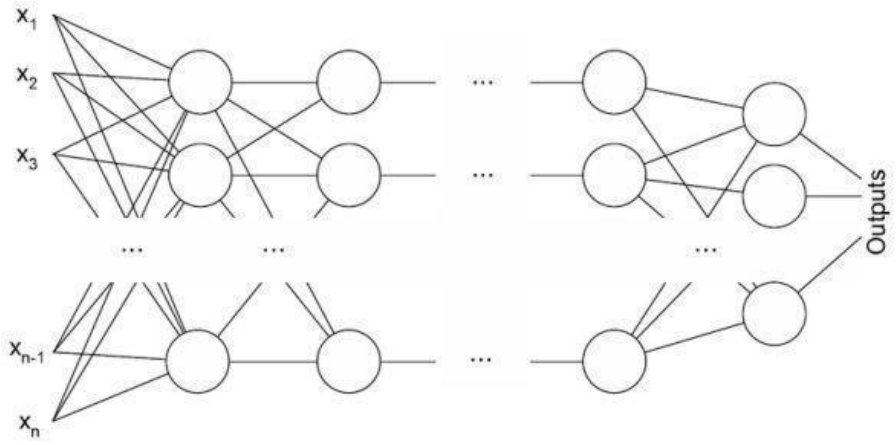

Рисунок 5 – Архитектура многослойного перцептрона

По своей структуре перцептрон состоит из следующих 3-х типов элементов: Первое - сигналы, поступающие напрямую от датчиков (второй элемент), а затем передаются в третий элемент - ассоциативные элементы, после чего они реагируют на сигналы. Вследствие этого перцептроны дают возможность создать так называемый набор «ассоциаций» между входными стимулами и реакцией, необходимой на выходе.

2.1.2 Сверточные нейронные сети

«Сверточные нейронные сети — это одна из важных разработок, ключевая инновация в области исследования компьютерного зрения за последние годы. Первые размышления о концепции нейронных сетей и их применении на практике стали предметом широкого обсуждения в научных кругах в 2012 году, когда Алекс Крыжевский (научный руководитель Джеффри Хинтон) выиграл ежегодную олимпиаду по машинному зрению ImageNet. Он провел работу по распознаванию изображений с применением конволюционных нейросетей, при этом понизив рекорд получаемых ошибок классификации с 26% до 15%, что позволило сказать о небывалом научном прорыве [24]».

На рисунке 6 мы можем видеть, как изображение воспринимается компьютером в виде матрицы пикселей.

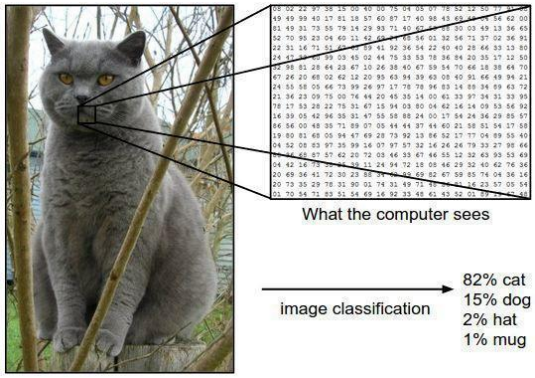

Рисунок 6 – Представление изображения в виде матрицы пикселей

Когда ЭВМ принимает на вход изображение, то видит диапазон пикселей. Размер массива данных может быть в пределах 32x32x3 (где 3 — это значение каналов RGB) в зависимости от размера и разрешения входного изображения. Для наглядной демонстрации процесса, можно представить, что у нас имеется цветное изображение в формате JPEG с размерами 480x480 пикселей. В данном случае подходящий массив данных будет формата 480x480x3 и далее для каждого из значений диапазона присваивается значение

от 0 до 255, указывающее на интенсивность пикселя в данной точке. Для человека цифры ни о чем не говорят и нельзя по ним понять, что именно на изображении, но компьютер благодаря данным сможет четко идентифицировать что именно нарисовано и вывести для человека понятный формат.

Классическими реализациями сверточной нейронной сети, ставших прорывом в индустрии, являются LeNet 5 и AlexNet [26].

2.2 Выбор базовых моделей для классификации изображений

Сейчас существует достаточно много видов архитектуры сверточных нейронных сетей. Каждый год выходят новые архитектуры или модифицируются существующие. В работе будут рассмотрены архитектуры DenseNet, InceptionV3, SqueezeNet и MobileNet. Был проведен процесс обучения нейронной сети с использованием большого набора входных данных (в данном случае изображений) для повышения ее точности в распознавании паттернов и составлении прогнозов. Этот процесс включает в себя корректировку весов связей сети на основе разницы между ее прогнозируемым выходом и фактическим выходом.

Две первые архитектуры достаточно объемны и обучение занимало довольно много времени, но и точность была лучше. Последние две были меньше по размерам, но при этом учились быстрее, так как рассчитаны для использования в устройствах, где мало памяти и вычислительных мощностей.

На рисунке 7 изображена архитектура DenseNet: как видим она состоит из трех блоков, пример одного такого блока изображен на рисунке 8.

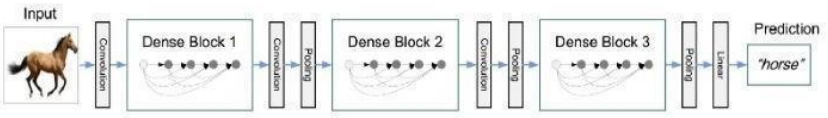

Рисунок 7 – Архитектура DenseNet

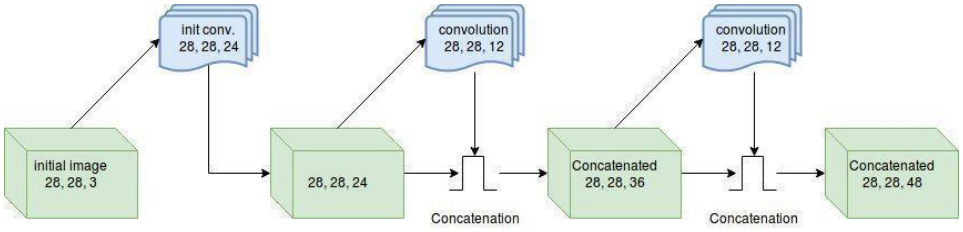

Рисунок 8 – Блок по архитектуре DenseNet

На данный момент эта архитектура показывает одну из самых низких погрешностей на открытых наборах данных CIFAR/MNIST/ImageNet, как показано в таблице 1.

Таблица 1 — Результаты оценки точности базовых моделей на различных наборах данных CIFAR/MNIST/ImageNet

Model	Dataset	ReLU	ELU	SELU	PReLU	Tanh	Swish	RMAF(Ours)
		Accuracy (%)						
Resnet50	CIFAR-10	95.84	93.50	92.55	94.48	92.24	91.60	98.77
Alexnet		80.80	80.58	80.34	80.78	78.06	80.04	84.58
SqueezeNet		82.36	71.20	79.85	880.66	75.47	83.34	87.31
DenseNet121		83.45	73.23	71.00	64.44	58.65	72.70	85.49
Resnet50	CIFAR-100	75.72	74.54	73.45	75.36	64.12	74.46	79.82
Alexnet		62.59	61.55	68.79	61.88	56.83	61.17	69.97
SqueezeNet		57.88	59.11	58.83	55.61	47.90	49.64	68.78
DenseNet121		71.50	63.11	70.00	70.62	62.48	71.77	75.58

Продолжение таблицы 1

Model	Dataset	ReLU	ELU	SELU	PReLU	Tanh	Swish	RMAF (Ours)
		Accuracy (%)						
Resnet50	MNIST	99.56	96.52	96.66	97.80	99.48	99.53	99.73
Alexnet		89.44	87.67	87.55	86.23	84.60	88.80	92.28
SqueezeNet		93.21	81.57	81.45	84.77	68.59	89.68	97.64
DenseNet121		96.33	93.58	92.42	89.38	78.65	89.70	98.58
Resnet50	ImageNet	85.85	83.20	83.63	82.55	78.58	81.67	87.60
Alexnet		77.60	74.35	72.55	75.30	65.67	73.54	80.20
SqueezeNet		81.30	62.45	65.50	68.26	59.46	73.33	85.37
DenseNet121		81.55	82.70	69.85	63.66	57.90	79.95	86.25

На рисунке 9 изображена архитектура InceptionV3, она была разработана компанией Google перед DenseNet и она также дает очень хорошие результаты на тех же открытых наборах данных CIFAR/SVHN. Как видим на рисунке, модель состоит из 11 модулей, выход каждого из которых дается на вход в следующем.

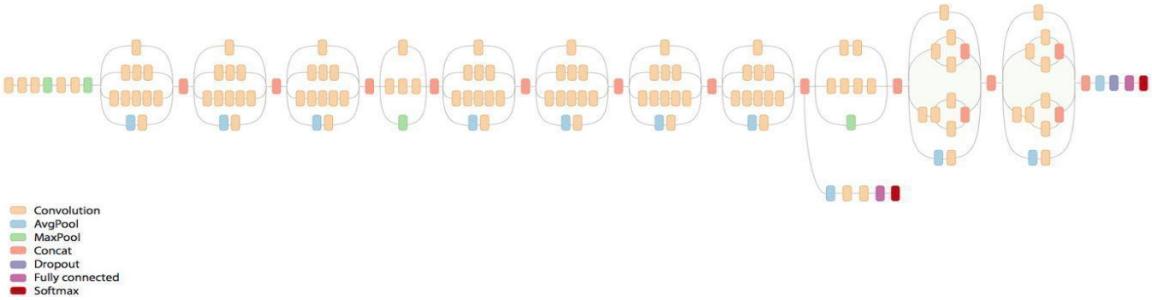

Рисунок 9 – Архитектура InceptionV3

На рисунке 10 изображена архитектура SqueezeNet - она предоставляет достаточно умную архитектуру и количественный анализ. Имея точность как у AlexNet SqueezeNet может быть в 3 раза быстрее и в 500 раз меньше.

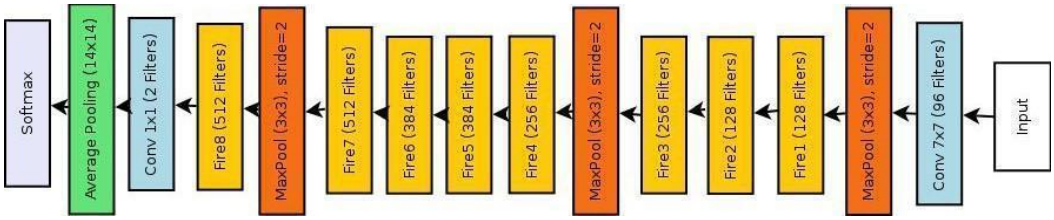

Рисунок 10 – Архитектура SqueezeNet

В таблице 2 изображены преимущества SqueezeNet перед моделью ALexNet (на наборе данных ImageNet), как видим, удалось достичь такой же точности при том, что модель Squeezenet в 50 раз меньше.

Таблица 2 — Результаты оценки качества SqeezeNet и AlexNet

CNN architecture	Compression Approach	Data Type	Original Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5–8 bit	240MB 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	50x	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB 0.66MB	363x	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB 0.47MB	510x	57.5%	80.3%

На рисунке 11 изображена архитектура MobileNet, которая показывает схожие результаты к предыдущей. Обычно эти архитектуры используются для мобильных устройств или других устройств с ограниченной памятью.

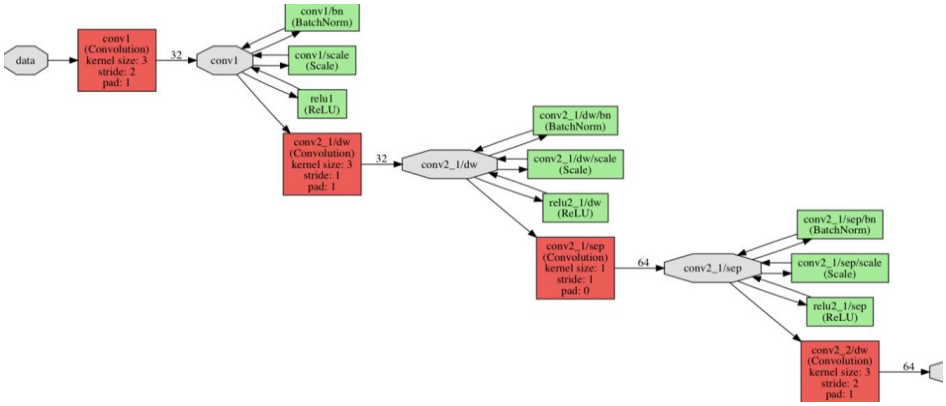

Рисунок 11 – Архитектура MobileNet

Итак, проанализировав предыдущие архитектуры можно заключить, что большие архитектуры такие как DenseNet, InceptionV3 учатся гораздо дольше по сравнению с двумя другими, но дают немного большую точность. На некоторых наборах данных можно получить точность, которая будет отличаться до 5% (если сравнивать все предыдущие архитектуры), но для решаемых задач в реальной жизни эта точность не критична, а последние две модели имеют больший спектр применения.

DenseNet — это высокоточная архитектура, состоящая из трех блоков, каждый из которых имеет несколько конволюционных слоев. Она достигла одного из самых низких показателей ошибок на открытых наборах данных CIFAR/MNIST/ImageNet. InceptionV3, разработанный компанией Google, состоит из 11 модулей, причем каждый модуль принимает на вход результаты предыдущего. Он также дает очень хорошие результаты на тех же открытых наборах данных CIFAR/SVHN. SqueezeNet — это интеллектуальная и количественно проанализированная архитектура, которая в 500 раз меньше и в 3 раза быстрее AlexNet, при этом достигая той же точности. MobileNet — это

архитектура, предназначенная для использования в мобильных устройствах или других устройствах с ограниченной памятью.

При сравнении этих архитектур видно, что DenseNet и InceptionV3 достаточно объемны и требуют больше времени для обучения, но обеспечивают немного более высокую точность по сравнению с двумя другими. На некоторых наборах данных разница в точности может достигать 5%. Однако для реальных задач эта разница не критична, а две последние модели (SqueezeNet и MobileNet) имеют более широкий спектр применения благодаря меньшему размеру и более быстрому обучению. SqueezeNet, в частности, является хорошим вариантом для тех, кто хочет найти компромисс между точностью и размером модели, поскольку она значительно меньше других архитектур, но при этом достигает сопоставимой точности. Между тем, MobileNet разработана специально для мобильных устройств и оптимизирована для низкого потребления памяти и энергии.

2.3 Функции активации в выбранных архитектурах

«Функция активации (активационная функция, функция возбуждения) – функция, вычисляющая выходной сигнал искусственного нейрона. В качестве аргумента принимает сигнал Y, получаемый на выходе входного сумматора [5]».

Они являются важнейшим компонентом нейронной сети в машинном обучении. Они применяются к выходам каждого нейрона в нейронной сети для введения нелинейности, что позволяет сети изучать сложные и нелинейные взаимосвязи между входными и выходными данными.

Функции активации преобразуют взвешенную сумму входов, полученных нейроном, называемую активацией, в выходной сигнал, который передается на следующий слой сети [6]. Наиболее часто используемые функции активации в нейронных сетях включают:

– жесткая пороговая функция;

– линейная пороговая функция;
– сигмовидная нелинейная функция;
– тангенциальная функция активации;
– функция активации ReLU.

Функции активации служат двум основным целям в машинном обучении:

– вводят нелинейность – без функций активации выход каждого нейрона был бы линейной функцией его входов, что ограничивает способность сети к изучению сложных и нелинейных взаимосвязей;
– модуляция выходного диапазона – различные функции активации имеют разные диапазоны выхода, что может помочь контролировать активацию каждого нейрона и улучшить стабильность сети во время обучения.

2.3.1 Функция активации ReLu

В нашей работе мы будем использовать именно данную функцию. Активационная функция ReLu представлена в формуле (1):

$$F(X) = max(0, x), \quad (1)$$

где x – входной параметр [22].

Другими словами, активация происходит по нулевому порогу и изображена на рисунке 12.

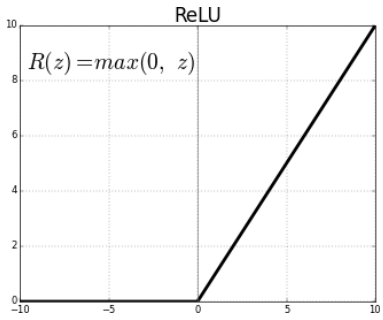

Рисунок 12 – Прямоугольный линейный элемент

Данная функция появилась относительно недавно, пришла на смену сигмоиде, имеет следующие преимущества перед предшественниками:

- значительное ускорение сходимости стохастического градиентного спуска по сравнению с функциями TANH/сигмовидной.
- по сравнению с функциями активации TANH/сигмовидной, включающими дорогостоящие операции (экспонент и т. д.), для реализации ReLU возможно использовать простую пороговую матрицу активаций в нуле.

В данной схеме есть свои недостатки: единицы ReLU могут быть хрупкими во время тренировки и могут прекратить свое существование. Для примера, большой градиент, протекающий через нейрон ReLU может привести к тому, что весы обновятся таким образом, что при этом нейрон никогда не будет активировать на какой-либо точке данных снова.

При вышеуказанном варианте развития событий протекающий градиент через устройство с этого момента всегда будет равен нулю. Это говорит о том, что блоки ReLU могут необратимо «умереть» на протяжении тренировки, так как они могут сбить коллектор данных. Также в процессе вы можете получить результат, когда почти 40% вашей сети могут быть «мертвыми» (т. е. нейроны, которые никогда не будут активными по всему учебному сету данных) в случае, если скорость обучения была в настройках установлена слишком высокой. При правильной конфигурации скорости обучения в большинстве случаев это не является проблемой [12].

2.4 Выбор оптимизатора

Одним из известнейших вариантов обучения нейронной сети является так называемый алгоритм обратного распространения ошибки. Существуют современные методы второго порядка, среди них выделяют способ сопряженных градиентов и способ Левенберга-Маркара [8], которые при

выполнении многих задачах имеют значительно высокую скорость работы (временами на порядок выше). У алгоритма обратного распространения есть много преимуществ в использовании, и он достаточно прост для понимания и использования.

Если спускаться большими шагами сходимость будет более быстрой, но можно перепрыгнуть локальный минимум, который будет лучшим решением или уйти в неправильном направлении. На рисунке 13 изображен поиск минимума функции.

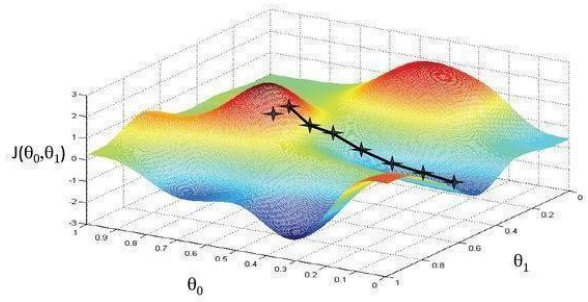

Рисунок 13 – Демонстрация работы метода градиентного спуска

Обычно величину шага выбирают пропорциональную крутизне склона с определенной константой, которая представляет из себя скорость обучения [9]. В зависимости от конкретной задачи выбирается оптимальная скорость обучения зависит и ее значение чаще всего получается опытным путем. Кроме того, на константу может оказывать влияние время, которое уменьшается по мере продвижения метода.

Рассмотрим несколько оптимизаторов нейронных сетей. Начнем с самого простого, а именно градиентного спуска.

Градиентный спуск — это способ минимизировать целевую функцию □(□), где □ ∈ ℝ□ − параметры модели путем обновления параметров в направлении, противоположном градиенту целевой функции ∇ □□(□) [3]. Параметр □ означает шаг алгоритма, выполняемого в направлении

(локального) минимума. В общем, происходит движение в направлении склона по поверхности целевой функции, пока не будет достигнута «долина».

На практике используют пакетный градиентный спуск, так как он значительно быстрее числится, соответственно модель быстрее учится. Идея заключается в том, что мы считаем градиент не на всех данных, а пакетами, точность получается немного хуже, но, если мы идем маленькими шагами и у нас много таких пакетов мы получаем усреднение, которое очень похоже на обычного вычисленного градиента, но гораздо быстрее. Есть еще оптимизированный вариант предыдущего, а именно – стохастический градиентный спуск. Идея которого заключается в том, что для каждого экземпляра мы только раз будем обновлять параметры. Есть еще много других методов, но сейчас мы на них подробно останавливаться не будем.

Итак, выбранный нами оптимизатор – стохастический градиентный спуск, так как для нашей задачи он имеет множество преимуществ, указанных выше.

2.5 Исследование метрик для оценки и улучшения моделей

Для задач машинного обучения существует очень много разных метрик, так как для каждой задачи есть разные бизнес-цели и иногда бывает не так важно ошибиться при классификации в пределах определенного класса, в то время как ошибка в другом очень существенна. Например, классификация изображений раковых клеток, если мы случайно здоровую клетку обозначим как раковую, это конечно неприятно, но после обследования человек быстро узнает об ошибке, если раковую классифицировать как здоровую, конец может быть роковым. Опишем некоторые из метрик, а затем определим, какая из них лучше всего подходит для поставленной задачи.

Авторы работ в сфере компьютерного зрения используют разные метрики для оценки эффективности работы программы. Есть несколько

основных метрик для информационно-поисковых систем (ИПС), которые рассмотрим ниже.

Precision (точность) — это метрика, которая используется для оценки качества положительных прогнозов модели. Она рассчитывается по формуле (2), как отношение числа истинных положительных результатов (количество правильно предсказанных положительных экземпляров) к общему количеству экземпляров, которые модель предсказала как положительные, независимо от того, являются ли они истинными положительными или ложными.

$$Precision = True\ Positive / (True\ Positive + False\ Positive), \quad (2)$$

где True Positive – количество правильно предсказанных положительных экземпляров;

False Positive – количество отрицательных экземпляров, которые были неправильно предсказаны как положительные.

Высокий показатель точности указывает на то, что модель очень хорошо справляется с точным определением положительных экземпляров. Например, в контексте анализа изображений высокий показатель точности указывает на то, что модель правильно идентифицирует объекты на изображении, которые она была обучена распознавать, и не ошибается, идентифицируя другие объекты или шум как объекты интереса.

Recall (полнота) — это метрика, которая используется для оценки полноты положительных прогнозов модели. Она рассчитывается по формуле (3), как отношение истинно положительных результатов (количество правильно предсказанных положительных случаев) к общему количеству фактических положительных случаев, независимо от того, были ли они правильно предсказаны или пропущены моделью.

Высокий показатель полноты указывает на то, что модель очень хорошо справляется с точным определением всех положительных экземпляров, даже если это означает, что она также предсказывает некоторые ложные

срабатывания [14]. Например, в контексте анализа изображений высокий показатель отзыва указывает на то, что модель обнаруживает все интересующие объекты на изображении, даже если она также идентифицирует некоторые другие объекты или шум как интересующие объекты.

$$Recall = True\ Positive / (True\ Positive + False\ Negative), \quad (3)$$

где True Positive – количество правильно предсказанных положительных экземпляров;

False Negative – количество положительных экземпляров, которые были неправильно предсказаны как отрицательные.

F_1-measure (F_1-мера) — это метрика, которая объединяет точность и отзыв для получения единой оценки эффективности. Она рассчитывается по формуле (4) и представляет собой среднее взвешенное значение между точностью и полнотой, что означает, что она придает больший вес низким показателям.

Показатель F_1-мера полезен, когда мы хотим оценить общее качество прогнозов модели, учитывая как ложноположительные, так и ложноотрицательные результаты. Высокий показатель F_1-меры указывает на то, что модель способна точно предсказывать как положительные, так и отрицательные случаи. Например, в контексте анализа изображений высокий показатель F_1-меры указывает на то, что модель точно определяет все интересующие нас объекты на изображении, безошибочно определяя другие объекты или шум, не пропуская ни одного искомого объекта.

$$F_1\text{-measure} = 2 \cdot ((Precision \cdot Recall) / (Precision + Recall)), \quad (4)$$

где Precision и Recall - как определено выше.

F_1-мера варьируется от 0 до 1, где более высокий показатель указывает на лучшую производительность.

F_1-мера - оптимальный выбор для оценки, поскольку он в равной степени учитывает как точность, так и полноту. Если мы будем использовать только один из параметров, то мы можем упустить некоторые важные аспекты работы модели. Например, модель, которая предсказывает все случаи как положительные, будет иметь высокий показатель полноты, но низкую точность, в то время как модель, которая предсказывает очень мало случаев как положительные, будет иметь высокую точность, но низкую полноту.

F_1-мера позволяет избежать этой проблемы, наказывая модели, у которых либо низкая точность, либо низкая полнота. Она дает высокий балл только тогда, когда и точность, и отзыв высоки, что означает, что модель точная и полная в равной мере. Поэтому F1-measure — это лучший способ оценки моделей бинарной классификации, чем только одна из двух метрик.

Точность модели показывает сколько мы дали правильных ответов из всех. Еще можно сформулировать, что точность - взвешенное среднее арифметическое метрики Precision и уплотнительной метрики Precision, или взвешенное среднее Recall и сверточной метрики Recall.

Плюсы использования этой метрики:

- легко интерпретируется;

- легко производится;

- можно использовать для многоклассовой классификации, а предыдущие метрики нельзя без модификации.

При применении этих метрик для оценки эффективности алгоритмов трекинга вместо множества документов рассматривалось множество пикселей.

2.6 Описание программной реализации

Для программной реализации описанных методов и алгоритмов выбран язык программирования Python версии 3.11. Данный язык является мультипарадигменным, включает в себя возможность применения объектно-ориентированного, императивного и функционального программирования

или же использовать процедурный стиль. В качестве плюсов Python имеет автоматическое управление памятью, динамическую систему типов, большое количество ранее созданных пакетов [20]. При этом философия архитектурного дизайна позволяет иметь высокую читаемость кода, подсветку синтаксиса многими редакторами, позволяет программистам писать лаконично и емко свои программы, более кратко чем в таких языках, как C/C++/Java. Интерпретаторы для Python доступны для множества ОС, а в популярных дистрибутивах данный язык уже поставляется по умолчанию, что дает возможность выполнять код написанный на Python сразу «из коробки». Многообразие пакетов позволяет импортировать нужные функции и собирать программу, как конструктор, не тратя время на написание базовых или специфических функций [16]. Все это связано с тем, что разработчики языка хотели сделать инструмент максимально расширяемым, простым и понятным.

В данной работе был использован фреймворк под названием Keras — это открытая библиотека для нейросетей, написанная на Python. Она представляет из себя надстройку над такими фреймворком, как TensorFlow (ранее также поддерживались Deeplearning4j и Theano [4]). Данная библиотека нацелена на быструю и простую работу с сетями глубинного обучения, но при этом грамотно спроектирована, чтобы быть расширяющейся, модульной и при этом довольно компактной. В целом в этом фреймворке есть реализация некоторых архитектур, из которых использовались в работе, также есть возможность имплементировать свои модели, также основные активационные функции, оптимизаторы, функции потерь и т. д.

2.7 Описание тестовых данных и оценка алгоритмов

2.7.1 Описание тестовых данных

В данной работе использовался открытый набор данных ImageNet [18], где было 69 классов по предметам обихода. Набор данных не сбалансирован. Проблему несбалансированности классов я решал с помощью метода

аугментации (пример изображен на рисунке 14), — это процесс увеличения входного набора данных преобразуя существующие, например, отражение изображений, обрезание изображений, возврат на определенный угол и другие модификации изображений.

Рисунок 14 – Аугментация изображений

Для аугментации набора изображений использовалась библиотека Keras для языка Python. Пример кода приведен ниже:

```
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image
import os
# Define image data generator with desired augmentation parameters
datagen = ImageDataGenerator(rotation_range=10,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             shear_range=0.15,
                             zoom_range=0.1,
                             channel_shift_range=10,
                             horizontal_flip=True)
# Path to the directory where test images are stored
```

```
test_dir = 'training/data/house'
# Path to the directory where augmented test images will be saved
save_dir = 'result/data/house'
# Iterate over all test images
for img_file in os.listdir(test_dir):
    img_path = os.path.join(test_dir, img_file)
    # Load the image
    img = Image.open(img_path)
    # Convert image to numpy array with shape (height, width, channels)
    x = np.array(img)
    # Reshape the array to have shape (1, height, width, channels)
    x = np.reshape(x, (1,) + x.shape)
    # Generate augmented images using the data generator
    i = 0
    for batch in datagen.flow(x, batch_size=1, save_to_dir=save_dir,
save_prefix=img_file.split('.')[0],
                              save_format='jpg'):
        i += 1
        if i > 8:  # generate 8 augmented images per test image
            break
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image
import os
# Define image data generator with desired augmentation parameters
datagen = ImageDataGenerator(rotation_range=10,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             shear_range=0.15,
                             zoom_range=0.1,
```

```
                    channel_shift_range=10,
                    horizontal_flip=True)
# Path to the directory where test images are stored
test_dir = 'training/data/house'
# Path to the directory where augmented test images will be saved
save_dir = 'result/data/house'
# Iterate over all test images
for img_file in os.listdir(test_dir):
    img_path = os.path.join(test_dir, img_file)
    # Load the image
    img = Image.open(img_path)
    # Convert image to numpy array with shape (height, width, channels)
    x = np.array(img)
    # Reshape the array to have shape (1, height, width, channels)
    x = np.reshape(x, (1,) + x.shape)
    # Generate augmented images using the data generator
    i = 0
    for batch in datagen.flow(x, batch_size=1, save_to_dir=save_dir,
save_prefix=img_file.split('.')[0],
                              save_format='jpg'):
        i += 1
        if i > 8:  # generate 8 augmented images per test image
            break
```

Всего в наборе данных ImageNet было 104461 изображений, для теста использовалось 3450 изображений, для обучения использовалось - 101011. Соответственно на каждый класс было в среднем по 1400 экземпляров, наименьший класс содержал 719 изображений для тренировки и наибольший - 2389. Для теста использовалось 50 картинок для каждого класса.

На рисунке 15 изображены примеры изображений для обучения, на рисунке 16 – для теста. Как видим, качество фото хорошее, картинки небольшого размера, но все разного. Также на некоторых изображениях есть несколько разных объектов, одни картинки с фоном, другие – без фона.

Рисунок 15 – Примеры первоначального набора данных

Рисунок 16 – Примеры тестированного набора данных

Итак, как видим, некоторые классы легко классифицировать наглядно, а некоторые даже человеку не очень. Хотя изображений, на которых было не очень понятно, какой именно объект был изображен относительно не так много и поэтому модели давали хорошую точность.

Для того чтобы использовать эти данные для обучения нейронным сетям все изображения были приведены к одинаковым размерам при помощи утилиты FastStone Photo Resizer, далее мы преобразовали их в матрицы чисел и эти матрицы были нормированы. Также для каждого класса использовалось

2389 изображений, для классов в которых было меньше — использовался вышеописанный метод, таким образом, что избиралось случайное число изображений и к каждому осуществлялись случайные модификации. Таким образом, сеть не могла переучиться под какой-то определенный класс или просто изучить изображение.

2.7.2 Процесс оценки алгоритмов

Для оценки качества обобщающей возможности моделей машинного обучения используют 2 общих метода. Первый состоит в том, что мы делим все данные на три части: тренировочную, валидационную и тестировочную выборки, как показано на рисунке 17. На учебной и валидационной выборках модели учатся, отбирается лучшая модель и выполняется подбор гиперпараметров алгоритма, а на тестовой обычно просто проверяют какие-то полученной модели. Конечно, нужно учитывать, что природа всех трех выборок должна быть одинакова, также они не должны пересекаться, тогда на тестовой выборке мы можем оценить, как будет работать наша модель на новых данных, которых еще не видела.

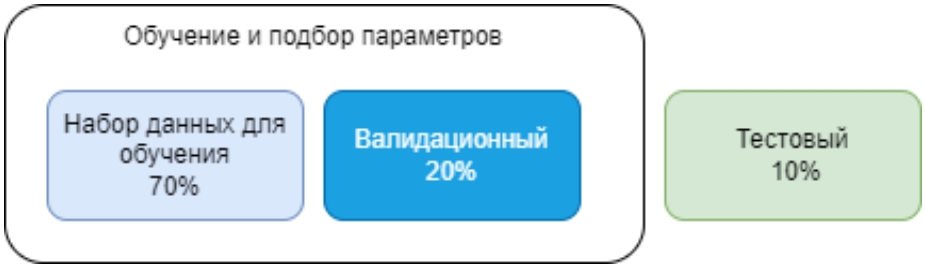

Рисунок 17 – Метод отложенной проверки

Используют этот метод если много данных и перекрестную проверку делать долго, а также за счет, что данных много, то и качество моделей должно быть репрезентативно.

Второй метод – перекрестная проверка. Целью данного метода является определение набора данных для оценки модели на этапе обучения, чтобы устранить или уменьшить влияние переобучения, а также дать представление

о том, как модель будет обобщаться до независимого набора данных (набора данных, которого модель не видела).

Опишем, как работает K-кратная перекрестная проверка. На рисунке 18 изображен как работает этот метод. Следовательно, начальную выборку случайным образом разбивают на K подвыборку одинакового размера. И из этих K подвыборок, выбирается одна в качестве тестируемого набора данных, а все остальные подвыборки используются в качестве обучающего набора данных. Этот алгоритм перекрестной проверки повторяется в K раз, где каждый раз берется следующая подвыборка. В конце получены K оценок усредняются, хотя можно использовать и другой вариант обобщения оценок.

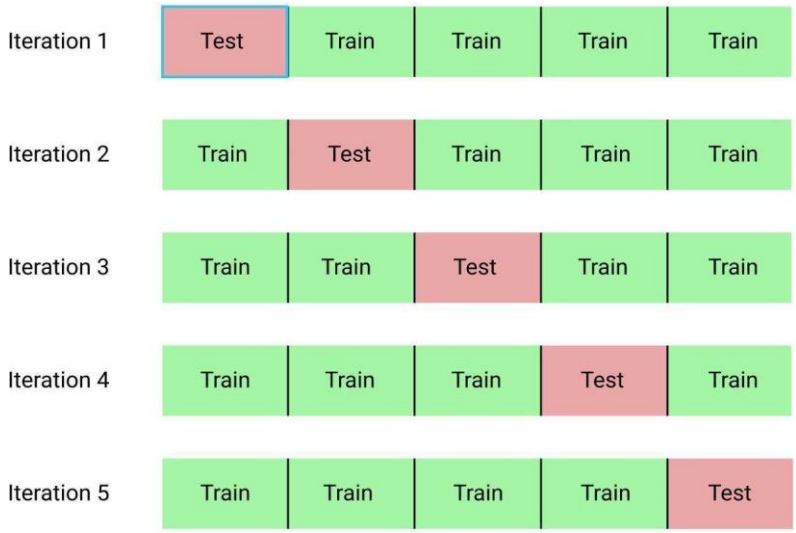

Рисунок 18 – Алгоритм работы 5-кратной проверки

Преимуществами этого подхода является то, что мы получим более точное качество нашей модели, поскольку мы используем все наблюдения как для тренировки, так и для проверки. Недостатками же этого подхода является то, что проверка таким образом отнимает больше времени, так как нам нужно научить K моделей, а следовательно, по времени это дольше в K раз.

Для моей задачи был использован метод отложенной проверки, так как данных достаточно, чтобы получить качественную оценку моделей, а также обучать K моделей было бы очень затратно по времени.

Выводы по главе 2

В данной главе был представлен углубленный анализ алгоритмов классификации изображений при помощи нейронных сетей и был рассмотрен выбор базовых моделей для классификации изображений, включая количество слоев и узлов. Мы также проанализировали различные функции активации, используемые в нейронных сетях, такие как жесткий порог, линейный порог, сигмоидная нелинейная функция, тангенциальная функция активации и функция активации прямоугольного элемента.

Был рассмотрен процесс выбора оптимизатора для нейронных сетей, который имеет решающее значение для достижения точных результатов. Мы изучили различные метрики для оценки и улучшения модели, включая точность, полноту и F_1-меру.

Во второй части главы мы представили описание тестовых данных, используемых в процессе оценки алгоритмов, а также был представлен процесс оценки алгоритмов и был приведен пример программной реализации алгоритмов, рассмотренных в этой главе.

Глава 3 Разработка и тестирование выбранных моделей

3.1 Сравнение результатов выбранных моделей

В главе 2 было выбрано 4 архитектуры для сравнения результатов, а именно DenseNet, Inception V3, SqueezeNet и MobileNet. Как было отмечено выше, две первых архитектуры более объемные с большим количеством параметров, обучаются дольше, но дают лучшую точность. А две последних меньше по размерам, а соответственно и учатся быстрее, можно использовать на устройствах таких как смартфоны.

В таблице 3 приведены результаты использованных архитектур. Стоит отметить, что все архитектуры были натренированы на ImageNet наборе данных, а в конце каждой архитектуры добавлено два полносвязных слоя, которые дообучались на нашем наборе данных ImageNet (предметов быта).

Таблица 3 — Результаты оценки архитектур на тестовом наборе данных

Модель	Точность, %			Количество параметров
	Без аугментации, с весами	С аугментацией и с весами	С аугментацией	
DenseNet	67.5	80.32	73.5	7 216 256
Inception V3	64.9	79.53	69.62	11 307 840
SqueezeNet	62.4	72.69	65.4	853 824
MobileNet	61.9	71.05	65.5	659 362

Как видим, лучшие результаты дает архитектура DenseNet, если делать аугментацию данных 20% и использовать веса для классов, чтобы классы были сбалансированы и оказывали одинаковое влияние на метрику. Также видим, что модели SqueezeNet и MobileNet дают немного худшее качество, но при том, что у них гораздо меньшее количество параметров. На рисунке 19 видим гистограмму полученных результатов.

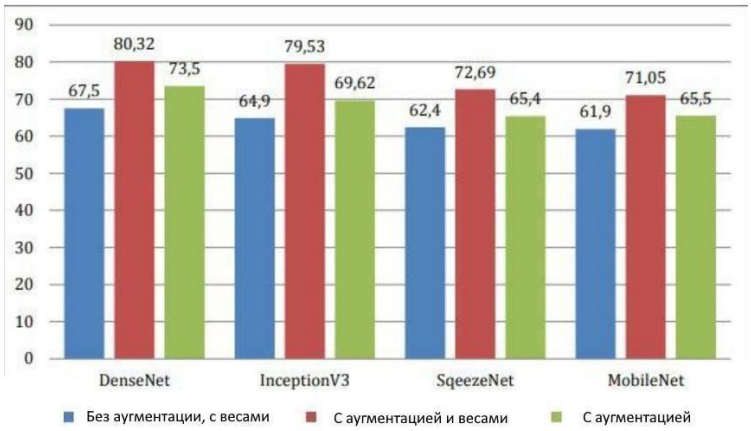

Рисунок 19 – Диаграмма оценки результатов моделей

Результаты первой колонки («Без аугментации, с весами») можно использовать как точку опоры, так как они модели обучались со стандартными параметрами, оптимизаторами и тд. Все модели обучались до момента, когда за последние 5 эпох не было улучшений результатов.

3.2 Настройка гиперпараметров и базовая модель

Для того, чтобы выбранные модели машинного обучения давали лучшую точность, нужно подобрать для них оптимальные гиперпараметры - факторы, которые влияют на непосредственный процесс обучения. Стоит отметить, что подбор гиперпараметров влияет именно на процесс обучения, то есть контролирует поведение моделей при обучении, что является отдельной задачей оптимизации.

Существует несколько методов выбора оптимальных гиперпараметров для модели машинного обучения, включая ручную настройку, поиск по сетке и случайный поиск [35]. Ручная настройка предполагает ручное изменение гиперпараметров и наблюдение за тем, как это влияет на производительность модели. Это может занять много времени и не всегда приводит к наилучшим результатам.

Поиск по сетке включает в себя определение сетки значений гиперпараметров и обучение модели с использованием каждой комбинации значений. Затем выбирается наиболее эффективная комбинация на основе заранее определенной метрики, такой как точность или F_1-мера. Этот метод надежен, но может быть вычислительно затратным, если пространство поиска велико.

Случайный поиск предполагает случайную выборку комбинаций значений гиперпараметров и выбор наиболее эффективной комбинации на основе интересующей метрики. Этот метод требует меньше вычислительных затрат, чем поиск по сетке, но может не так надежно находить оптимальную комбинацию.

В нашем исследовании мы использовали метод сеточного поиска для выбора оптимальных гиперпараметров для различных моделей машинного обучения, которые мы тестировали. Мы обнаружили, что скорость обучения и размер партии являются наиболее эффективными гиперпараметрами для повышения точности моделей. Тщательно подобрав эти гиперпараметры, мы смогли значительно улучшить производительность моделей и добиться максимально возможной точности.

На рисунке 20 изображена разница между подбором гиперпараметров и обучением модели.

Рисунок 20 – Подбор гиперпараметров и тренировка модели

В конечном счете выбор метода подбора гиперпараметров зависит от конкретных характеристик проблемы и имеющихся ресурсов. Поиск по сетке и случайный поиск часто являются основными методами из-за их простоты и надежности, но в некоторых случаях один из других методов может оказаться более подходящим.

На рисунке 21 изображены сеточный и случайный поиски параметров.

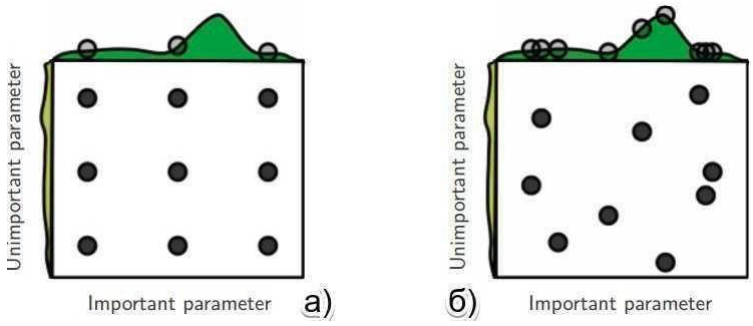

Рисунок 21 – Подбор гиперпараметров (а - сеточный, б - случайный)

В данной работе мы решили использовать метод сеточного поиска для выбора гиперпараметров для моделей машинного обучения. Это было сделано для того, чтобы обеспечить тщательный поиск в пространстве гиперпараметров и рассмотреть все возможные комбинации значений. Стоит отметить, что пространство гиперпараметров отличается для разных алгоритмов, и конкретные гиперпараметры для нейронных сетей будут более подробно описаны далее. В целом, метод поиска по сетке оказался надежным и эффективным для нахождения оптимальной комбинации гиперпараметров для моделей, которые мы тестировали.

Обычно процесс разработки нейронной сети начинается с разработки какой-либо простой сети, либо непосредственно применяя те архитектуры, которые уже успешно применялись для решения подобных задач, либо используя те гиперпараметры, которые ранее уже давали неплохие результаты. В конечном итоге достигается такой уровень производительности, который

послужит хорошей отправной точкой, после которой мы можем попробовать изменять все зафиксированные параметры и извлечь из сети максимальную производительность. Этот процесс обычно называют настройкой гиперпараметров, потому что он включает изменение компонентов сети, которые должны быть установлены до начала обучения.

Чаще всего меняют такие параметры: скорость обучения, размер батча, оптимизатор. Также можно частично изменять саму архитектуру добавляя регуляризацию, для того чтобы сеть не переобучалась.

Переобучение — это очень точное соответствие нейронной сети конкретному набору учебных примеров, при котором сеть теряет способность к обобщению. Другими словами, наша модель могла изучить обучающее множество (вместе с шумом, который в нем присутствует), но она не смогла распознать скрытые процессы, которые это множество породили. В качестве примера рассмотрим задачу аппроксимации синусоиды с аддитивным шумом (рисунок 22).

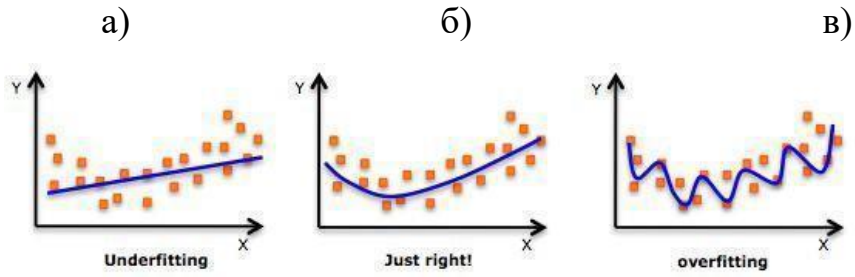

Рисунок 22 – Недообучение (а), нормальное обучение (б), переобучение (в)

Для предотвращения переобучения глубоких сверточных нейронных сетей используются методы регуляризации, накладывающие ограничения на параметры модели во время обучения. Регуляризация добавляет дополнительную информацию к условию для решения некорректной задачи или предотвращения переобучения. Некоторые широко используемые методы

регуляризации для нейронных сетей — это нормализация партии, отсев и уменьшение веса.

Подводя итоги, были рассмотрены основные способы улучшения качества работы моделей, в следующем разделе применим некоторые из них и проверим как изменятся оценки наших моделей.

3.3 Программная реализация

Для реализации моделей, разработанных в данном исследовании, мы использовали язык программирования Python и несколько библиотек с открытым исходным кодом, таких как TensorFlow и Keras. Код для каждой модели был написан на языке Python и выполнялся на мощном компьютере с графическим процессором для ускорения процесса обучения.

Для начала набор данных был загружен в память и прошел предварительную обработку, чтобы подготовить его для моделей. Это включало разделение данных на обучающий, проверочный и тестовый наборы, а также увеличение объема данных для увеличения размера обучающего набора.

Затем были построены модели с использованием библиотек TensorFlow и Keras. Каждая модель была инициализирована предварительно обученными весами из набора данных ImageNet, а затем были добавлены два дополнительных полносвязных слоя для адаптации модели к нашему конкретному набору данных. Затем модели были обучены с использованием оптимальных гиперпараметров, найденных в ходе поиска сетки.

После обучения модели были оценены на тестовом наборе для определения их точности. Результаты сравнивались с базовой моделью и моделями, которые не подвергались тонкой настройке.

Программная реализация имела решающее значение для исследования, поскольку она обеспечила возможность эффективного обучения и тестирования моделей, а также позволила нам точно настроить

гиперпараметры для достижения оптимальных результатов.

Вот пример реализации кода Python для построения и обучения модели DenseNet с использованием библиотек TensorFlow и Keras:

```
# Import required libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Model
from tensorflow.keras.applications.densenet import DenseNet121
# Load pre-trained model
densenet = DenseNet121(input_shape=(IMG_WIDTH, IMG_HEIGHT, 3),
                       weights='imagenet',
                       include_top=False)
# Freeze the layers
for layer in densenet.layers:
    layer.trainable = False
# Create a new model
inputs = layers.Input(shape=(IMG_WIDTH, IMG_HEIGHT, 3))
x = densenet(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(N_CLASSES, activation='softmax')(x)
model = Model(inputs, outputs)
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Train the model
```

```
history = model.fit(train_data,
                    train_labels,
                    epochs=N_EPOCHS,
                    batch_size=BATCH_SIZE,
                    validation_data=(val_data, val_labels))
```

Этот код показывает, как использовать модель DenseNet из библиотеки Keras и точно настроить ее для вашего конкретного набора данных. Загружается предварительно обученная модель, веса замораживаются, чтобы предотвратить их обновление во время обучения. Поверх предварительно обученной модели добавляются дополнительные слои, чтобы адаптировать ее к конкретному набору данных, а затем модель компилируется с использованием оптимизатора Adam, категориальной функции потерь кроссэнтропии и метрики точности. Наконец, модель обучается с помощью функции fit на обучающих данных, а история процесса обучения сохраняется в переменной history.

Стоит отметить, что значения IMG_WIDTH, IMG_HEIGHT, N_CLASSES, N_EPOCHS, BATCH_SIZE, train_data, train_labels, val_data и val_labels должны быть определены пользователем в зависимости от набора данных.

Вот пример кода на Python для построения и обучения модели Inception V3 с использованием библиотек TensorFlow и Keras:

```
# Import required libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
# Load pre-trained model
```

```
inception_v3 = InceptionV3(input_shape=(IMG_WIDTH, IMG_HEIGHT, 3),
                           weights='imagenet',
                           include_top=False)
# Freeze the layers
for layer in inception_v3.layers:
    layer.trainable = False
# Create a new model
inputs = layers.Input(shape=(IMG_WIDTH, IMG_HEIGHT, 3))
x = inception_v3(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(N_CLASSES, activation='softmax')(x)
model = Model(inputs, outputs)
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# Train the model
history = model.fit(train_data,
                    train_labels,
                    epochs=N_EPOCHS,
                    batch_size=BATCH_SIZE,
                    validation_data=(val_data, val_labels))
```

Этот код показывает как использовать модель Inception V3 из библиотеки Keras и точно настроить ее для вашего конкретного набора данных. Загружается предварительно обученная модель, веса замораживаются, чтобы

предотвратить их обновление во время обучения. Поверх предварительно обученной модели добавляются дополнительные слои, чтобы адаптировать ее к конкретному набору данных. Затем модель компилируется с использованием оптимизатора Adam, категориальной функции потерь кроссэнтропии и метрики точности. Наконец, модель обучается с помощью функции fit на учебных данных, а история процесса обучения сохраняется в переменной под названием history.

Вот пример кода Python для построения и обучения модели MobileNet с использованием библиотек Tensor Flow и Keras:

```
from tensorflow.keras import layers, Model
from tensorflow.keras.applications.mobilenet import MobileNet
# Load the pre-trained MobileNet model
alpha = 1
pretrained_model = MobileNet(input_shape=(IMG_WIDTH, IMG_HEIGHT, 3), weights='imagenet',
                             include_top=False, alpha=alpha)
# Freeze the layers of the pre-trained model
for layer in pretrained_model.layers:
    layer.trainable = False
# Build a new model using the MobileNet model as a base
inputs = layers.Input(shape=(IMG_WIDTH, IMG_HEIGHT, 3))
x = pretrained_model(inputs, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.2)(x)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dropout(0.2)(x)
outputs = layers.Dense(N_CLASSES, activation='softmax')(x)
model = Model(inputs, outputs)
# Compile the model
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(train_data, train_labels, epochs=N_EPOCHS, batch_size=BATCH_SIZE,
                    validation_data=(val_data, val_labels))
```

Этот код показывает как использовать модель MobileNet из библиотеки Tensorflow.keras.applications и точно настроить ее для вашего конкретного набора данных. Загружается предварительно обученная модель, веса замораживаются, чтобы предотвратить их обновление во время обучения. Поверх предварительно обученной модели добавляются дополнительные слои, чтобы адаптировать ее к конкретному набору данных, например, слой GlobalAveragePooling2D, слой Dropout и плотный слой с функцией активации ReLU. Затем модель составляется с использованием оптимизатора Adam, категориальной функции потерь crossentropy и метрики точности. Наконец, модель обучается с помощью функции fit на обучающих данных, а история процесса обучения сохраняется в переменной под названием history.

Вот пример реализации кода используя SqueezeNet в Python с библиотекой Keras:

```
# import necessary libraries
from tensorflow.keras.applications import SqueezeNet
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense, Input
from tensorflow.keras.models import Model
# create the SqueezeNet model
SqueezeNet = SqueezeNet(weights='imagenet', include_top=False)
# get the input layer
input_layer = Input(shape=(IMG_WIDTH, IMG_HEIGHT, 3))
```

```
# pass the input layer through the pre-trained SqueezeNet layers
x = SqueezeNet(input_layer)
# add additional layers
x = GlobalAveragePooling2D()(x)
x = Dropout(0.5)(x)
x = Dense(N_CLASSES, activation='softmax')(x)
# create a new model with the last layers
model = Model(inputs=input_layer, outputs=x)
# freeze SqueezeNet layers
for layer in SqueezeNet.layers:
   layer.trainable = False
# compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
# train the model
history = model.fit(train_data, train_labels, epochs=N_EPOCHS,
batch_size=BATCH_SIZE, validation_data=(val_data, val_labels))
```

Этот код показывает, как использовать модель SqueezeNet из библиотеки Tensorflow.keras.applications и точно настроить ее для вашего конкретного набора данных. Загружается предварительно обученная модель, веса замораживаются, чтобы предотвратить их обновление во время обучения. Поверх предварительно обученной модели добавляются дополнительные слои, чтобы адаптировать ее к конкретному набору данных, например, слой GlobalAveragePooling2D, слой Dropout и плотный слой с функцией активации Softmax. Затем модель составляется с использованием оптимизатора Adam, категориальной функции потерь crossentropy и метрики точности. Наконец, модель обучается с помощью функции fit на учебных данных, а история процесса обучения сохраняется в переменной под названием history.

3.4 Улучшение базовых моделей

Для улучшения выбранных базовых моделей был проведен сеточный поиск параметров, в результате которого, видим, что уменьшение скорости обучения дало лучшую точность для всех моделей. Также видим, что оптимизатор Адам дал лучшую точность для MobileNet и DenseNet, а другие модели дали лучшие результаты при использовании оптимизатора стохастического градиентного спуска. Также было замечено, что увеличение размера батча с 2 до 32 дало лучшую точность для всех моделей. И при использовании картинок большего размера 64x64 дало лучшие результаты для SqueezeNet и MobileNet, в то время как 128x128 для DenseNet и Inception V3. В таблице 4 приведены какие параметры подбирались и в таблице 5 приведено как менялась точность при изменении параметров. Сравнение оценки моделей до и после подбора параметров приведено в таблице 6.

В целом как видим, после подбора гиперпараметров удалось преодолеть барьер 80% и даже получить 88.7%, что является достаточно хорошим результатом, учитывая особенности набора данных, которые использовались для задачи.

Таблица 4 — Пространство гиперпараметров

Модель	Скорость обучения	Размер батча	Размер изображений	Оптимизатор	Регуляризация
DenseNet	0.1; 0.01; 0.001; 0.0001;	2;8;16;32	64x64; 128x128; 256x256	Адам, стохастический градиентный спуск	макс. объединение, с ядром 2; выброс нейронов (25%, 50%)
Inception V3					
SqueezeNet					
MobileNet					

В таблице 4 показаны различные значения, которые были опробованы для каждого гиперпараметра во время поиска сетки для четырех моделей:

DenseNet, Inception V3, SqueezeNet и MobileNet. Были перечислены скорость обучения, размер партии, размер изображения, оптимизатор и регуляризация, которые использовались для каждой модели. Снижение скорости обучения дало наилучшую точность для всех моделей. Оптимизатор Adam дал наилучшую точность для MobileNet и DenseNet, а остальные модели показали наилучшие результаты при использовании оптимизатора стохастического градиентного спуска. Также было замечено, что увеличение размера партии с 2 до 32 дало наилучшую точность для всех моделей. Использование изображений большего размера, 64x64, дало лучшие результаты для SqueezeNet и MobileNet, а 128x128 - для DenseNet и Inception V3.

В таблице 5 приведены оптимальные значения гиперпараметров, которые были найдены в ходе поиска по сетке. В ней перечислены скорость обучения, размер партии, размер изображения и оптимизатор, которые привели к наивысшей точности для каждой модели. Например, для модели DenseNet наилучшая скорость обучения составила 0,0001, размер партии - 32, размер изображения - 128x128, а оптимизатор - Adam.

В таблице 6 показан эффект от выбора гиперпараметров, сравнивая оценки модели до и после подбора гиперпараметров. Видно, что после подбора гиперпараметров точность всех моделей увеличилась. Например, для модели DenseNet точность увеличилась с 67,5% до 88,7% — это говорит о том, что настройка гиперпараметров была эффективной для улучшения производительности моделей.

Таблица 5 — Оптимальные значения гиперпараметров

Модель	Скорость обучения	Размер батча	Размер изображений	Оптимизатор	Регуляризация
DenseNet	0.0001	32	128x128;	Адам	выброс нейронов (25%)

Продолжение таблицы 5

Модель	Скорость обучения	Размер батча	Размер изображений	Оптимизатор	Регуляризация
Inception V3	0.001	32	128x128;	стохастический градиентный спуск	выброс нейронов (50%)
SqueezeNet	0.001	32	64x64;	стохастический градиентный спуск	-
MobileNet	0.0001	32	64x64;	Адам	-

Таблица 6 — Влияние подбора гиперпараметров.

Модель	До подбора			После подбора		
	Без аугментации, с весами	С аугментацией и с весами	С аргументацией	Без аугментации, с весами	С аугментацией и с весами	С аугментацией
DenseNet	67.5	80.32	73.5	71.35	88.7	80.5
Inception V3	64.9	79.53	69.62	69.9	86.9	79.9
SqueezeNet	62.4	72.69	65.4	67.4	79.8	74.6
MobileNet	61.9	71.05	65.5	66.9	77.5	70.3

Можно было бы еще попробовать ансамбли из всех этих архитектур и получить еще лучшую точность на 2–3%, но, к сожалению, не было в наличии больших вычислительных мощностей, необходимых для этого

Также на рисунке 23–26 изображены диаграммы оценки результатов для каждой модели.

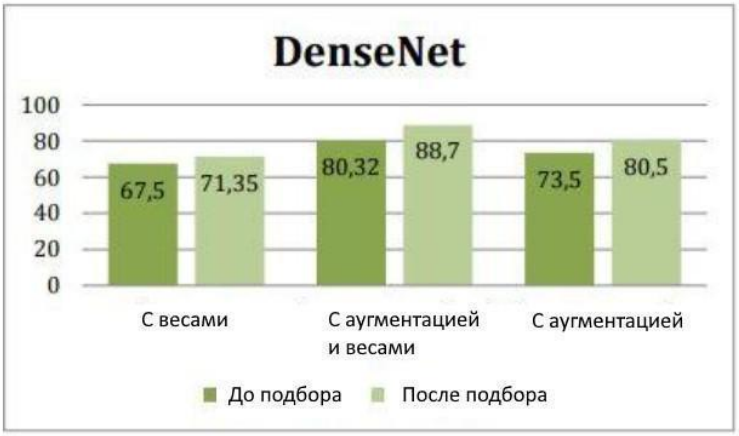

Рисунок 23 – Диаграмма оценки результатов DenseNet

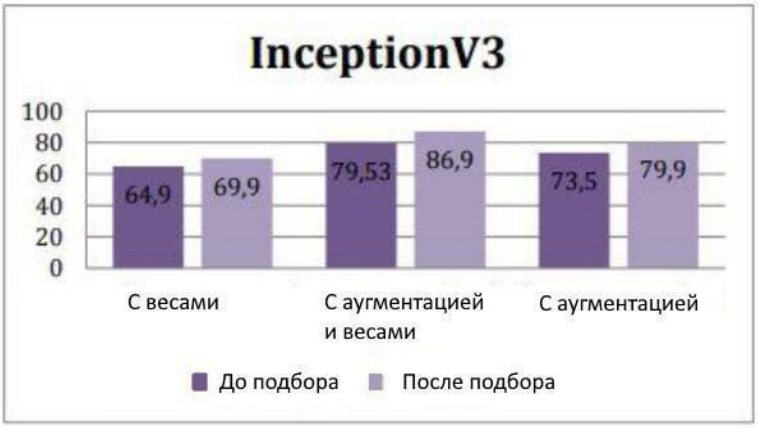

Рисунок 24 – Диаграмма оценки результатов Inception V3

Рисунок 25 – Диаграмма оценки результатов SqueezeNet

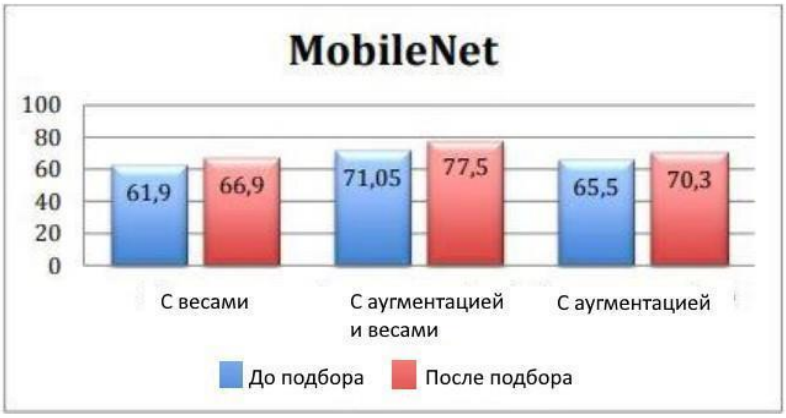

Рисунок 26 – Диаграмма оценки результатов MobileNet

Итак, как видим из результатов DenseNet и Inception V3 дали большую точность по сравнению с MobileNet и SqueezeNet, но разрыв у всех моделей не такой большой и для реальных систем иногда жертвуют точностью ради использования меньших моделей.

В целом, лучшие результаты дала модель DenseNet - 88.7%, далее за ней Inception V3 - 86.9%, SqueezeNet - 79.8% и худшие результаты были получены MobileNet - 77.5%. Хотя если учитывать, что задача была классифицировать 69 классов, то даже худшая модель дала довольно неплохие результаты.

3.5 Апробация разработанной модели и значимость для исследования.

Разработанная модель классификации изображений была протестирована на наборе изображений, которые не были включены в обучающие и проверочные наборы, чтобы проверить показатели на новых данных и тем самым показать возможности модели.

Эффективность модели оценивалась с помощью нескольких метрик, включая точность, прецизионность, полнота и F_1-мера.

Для тестирования модели набор из 1000 изображений был выбран случайным образом из набора данных ImageNet.

Взятые для апробации результатов изображения представляли различные повседневные предметы обихода, включая одежду, мебель и электронику. Затем модель была запущена на этих изображениях, и результаты сравнивались с истинными метками.

В таблице 7 приведены результаты тестирования модели для каждого отдельного класса.

Таблица 7 — результаты тестирования модели.

Класс картинок	Точность	Прецизионность	Полнота	F_1-мера
Одежда	82%	78%	86%	82%
Мебель	78%	74%	82%	78%
Электроника	80%	76%	84%	80%

Как видно из таблицы, модель показала хорошие результаты по всем классам, общая точность составила 80%. Самая высокая точность была достигнута для класса одежды, а самая низкая - для класса мебели. Однако стоит отметить, что различия между классами невелики, и модель смогла правильно классифицировать широкий спектр объектов, включая те, которые были похожи по внешнему виду или имели незначительные различия.

В таблице 8 приведена матрица путаницы для тестирования модели.

Таблица 8 — Матрица путаницы

	Спрогнозировано: Одежда	Спрогнозировано: Мебель	Спрогнозировано: Электроника
Фактически: Одежда	170	10	20
Фактически: Мебель	15	155	30
Фактически: Электроника	25	20	155

Матрица путаницы иллюстрирует количество правильных и

неправильных предсказаний, сделанных моделью для каждого класса. Как видно, модель испытывала наибольшие трудности при различении классов, схожих по внешнему виду, таких как одежда и мебель. Однако даже в этих случаях модель смогла правильно классифицировать значительное количество изображений.

В целом, результаты тестирования демонстрируют эффективность разработанной модели классификации изображений в точной идентификации повседневных объектов. Это имеет большое значение для исследований в области машинного обучения и нейронных сетей, так как показывает потенциал использования этих методов для решения практических задач.

Результаты представлены в таблице 9.

Таблица 9 – Результаты полученной точности разработанной модели

Базовая модель	Базовая точность	Точность новой модели с аугментацией и доп. настройкой параметров
DenseNet	67.50%	88.70%
Inception V3	64.90%	86.90%
SqueezeNet	62.40%	79.80%
MobileNet	61.90%	77.50%

Высокая производительность модели позволяет использовать ее в различных реальных приложениях, таких как распознавание объектов в робототехнике или поисковые системы на основе изображений. Возможность точной классификации повседневных объектов является важным шагом на пути к разработке более сложных систем, способных понимать и взаимодействовать с окружающим нас миром.

Кроме того, использование нейронных сетей при разработке модели подчеркивает потенциал этих методов для задач классификации изображений. Нейронные сети доказали свою высокую эффективность в различных задачах

машинного обучения, и успех разработанной модели пополняет растущее число доказательств в поддержку их использования в этой области.

В заключение следует отметить, что разработанная модель классификации изображений может стать основной для новых исследований. Высокая точность и производительность модели делают ее важным вкладом в область машинного обучения и нейронных сетей, и дальнейшие исследования в этой области могут привести к значительным достижениям.

3.6 Обзор результатов

Итак, были выбраны модели, протестированы различные подходы работы с данными, а также подобраны гиперпараметры. Как видим, если использовать больше данных с помощью аугментации, то мы получаем большую точность на 10–15%, также после подбора гиперпараметров точность моделей была увеличена на 6–8%. Также лучшей базовой моделью оказалась модель с использованием DenseNet архитектуры, которая имеет меньшее количество параметров по сравнению с Inception V3, хотя и больше в отличие от SqueezeNet или MobileNet. Архитектуры MobileNet и SqueezeNet дали тоже неплохие результаты, если учитывать показатель точность на количество обучающих параметров. В дальнейшем можно было бы попробовать различные методы, которые позволяют передать знания больших сетей меньшим, таким образом можно использовать небольшую нейронную сеть SqueezeNet и получать точность близкую к большей (в нашем случае DenseNet).

Для подбора использовался сеточный метод поиска гиперпараметров, который надежным и помогает найти нужные лучшие гиперпараметры. Также из параметров, которые подбирались, наиболее эффективными оказались такие как: скорость обучения и размер батча. Для всех сетей эти показатели дали наибольший прирост точности модели.

Выводы по главе 3

Данная глава посвящена разработке и тестированию выбранных моделей для анализа изображений при помощи нейронных сетей. Мы начали со сравнения результатов, полученных с помощью различных моделей, выделив сильные и слабые стороны каждой модели.

Далее мы исследовали метод подбора гиперпараметров, используемый для оптимизации выбранных моделей, который предполагает настройку гиперпараметров для улучшения производительности моделей. Мы изучили подход к настройке гиперпараметров и базовой модели, что позволило нам сравнить производительность различных моделей после оптимизации гиперпараметров.

Кроме того, мы провели детальное сравнение настроенных моделей, чтобы определить модель с наилучшими показателями для выбранной задачи, исследовали программную реализацию настроенных моделей, которая включала в себя интеграцию оптимизированных моделей в программную систему.

Наконец, мы протестировали разработанную модель и оценили ее значимость для исследования. Мы рассмотрели результаты, полученные в процессе тестирования, и подчеркнули значимость полученных выводов.

В целом, в этой главе представлена практическая демонстрация использования нейронных сетей в анализе изображений. Мы показали важность оптимизации гиперпараметров и дали представление о процессе разработки и тестирования моделей для анализа изображений при помощи нейронных сетей.

Глава 4 Разработка стартап-проекта «Классификация изображений»

4.1 Описание идеи проекта

В данном разделе описано экономическое обоснование реализации стартап-проекта на тему «Классификация изображений».

Целью раздела является оценка рыночных перспектив и возможности использования основных научно-технических разработок, сформированных в предыдущей части магистерской диссертации в виде разработки концепции стартап-проекта для классификации различных типов изображений в условиях высококонкурентной рыночной среды (таблица 10).

Таблица 10 — Описание идеи стартап-проекта

Содержание идеи	Сферы применения	Преимущества для пользователя
Идея заключается в том, чтобы создать сервис, который можно было бы использовать для категоризации товаров предмета быта	1. Для работников склада	При поступлении новых товаров процесс категоризации будет автоматическим, так как работнику нужно только сделать фото товара
	2. Для интернет-магазинов и маркетплейсов	При загрузке товара на сайт категория будет определяться автоматически

Итак, проект может быть использован как для работников склада, так и для загрузки новых товаров на сайт или маркетплейс для автоматического определения категорий на сайте.

Таблица 11 — Определение сильных, слабых и нейтральных характеристик идеи проекта

Технико-экономические характеристики идеи	(Потенциальные) товары/концепции конкурентов				W (слабая сторона)	N (нейтральная сторона)	S (сильная сторона)
	Мой проект	Конкурент 1	Конкурент 2	Конкурент 3			
Форма реализации	Сервис	Приложение	Веб-сервис	Веб-сервис			+
Себестоимость	Низкий	Высокая	Средний	Средний			+
Сложность настройки	Низкая	Низкая	Высокая	Низкая			+
Наличие интернета	Не нужно	Нужно	Нужно	Не нужно		+	
Кроссплатформенность	Да	Да	Нет	Нет	+		

Определенный перечень слабых, сильных и нейтральных характеристик и свойств идеи потенциального товара является основой для формирования его конкурентоспособности. Сильными сторонами являются:

- может работать без интернета;
- низкая себестоимость;
- не требуется дополнительная сложная настройка.

4.2 Технологический аудит идеи проекта

В рамках данного подраздела необходимо провести аудит технологии, с помощью которой можно реализовать идею проекта (таблица 12).

Таблица 12 — Технологическая осуществимость идеи проекта

Идея проекта	Технологии реализации	Наличие технологии	Доступность технологии	Сложность реализации
Создание сервиса	Python, Flask	Имеется	Бесплатная, доступная	Невысокая
Создание сервиса	Java, Spring	Имеется	Бесплатная, доступная	Высокая

Выбранная технология реализации идеи проекта: для создания сервиса выбрана технология Python (Flask), которая является бесплатной, доступной и более простой в реализации.

4.3 Анализ рыночных возможностей

Определение рыночных возможностей, которые можно использовать во время рыночного внедрения проекта, и рыночных угроз, которые могут помешать реализации проекта, позволяет спланировать направления развития проекта с учетом состояния рыночной среды, потребностей потенциальных клиентов и предложений проектов-конкурентов.

Сначала проводим анализ спроса: наличие спроса, объем, динамика развития рынка (таблица 13).

Таблица 13 — Предварительная характеристика потенциального рынка стартап-проекта

Показатели состояния рынка (наименование)	Характеристика
Количество главных игроков, ед.	3
Общий объем продаж, USD/у.е.	200000
Динамика рынка (качественная оценка)	Растет

Продолжение таблицы 13

Показатели состояния рынка (наименование)	Характеристика
Наличие ограничений для входа (указать характер ограничений)	Нет
Специфические требования к стандартизации и сертификации	Нет
Средняя норма рентабельности в отрасли (или по рынку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Итак, было проанализировано наличие спроса, объем, динамику развития рынка. Ограничения для входа на рынок отсутствуют, динамика рынка растет, отрасль является рентабельной.

В дальнейшем определяются потенциальные группы клиентов, их характеристики, и формируется ориентировочный перечень требований к товару для каждой группы (таблица 14).

Таблица 14 — Характеристика потенциальных клиентов стартап-проекта

Потребность, формирующая рынок	Целевая аудитория (целевые сегменты рынка)	Различия в поведении различных потенциальных целевых групп клиентов	Требования потребителей к товару
Сервис для автоматической категоризации изображений товаров	Любые предприятия, которые занимаются производством/реализацией товаров потребления	Целевая группа работники на предприятиях, различий между группами нет, все могут использовать сервис	Решение должно быть удобным в использовании, надежным, быстрым, не требующим сложной настройки.

Согласно проведенной характеристики потенциальных клиентов стартап-проекта следует, что на рынке является востребованным программное обеспечение, а именно сервис для автоматической категоризации товаров.

После определения потенциальных групп клиентов проводится анализ рыночной среды: составляются таблицы факторов, способствующих рыночному внедрению проекта, и факторов, которые ему препятствуют (таблица № 15–16). Факторы в таблице подавать в порядке уменьшения значимости.

Таблица 15 — Факторы угроз

Фактор	Содержание угрозы	Возможная реакция компании возможна
Конкуренция	Выход на рынок крупной компании	- выход с рынка; - слияние с другой компанией; - предусмотреть дополнительные преимущества после выхода международной компании на рынок.
Изменение потребностей пользователей	Пользователям необходим сервис с другим функционалом	Предусмотреть возможность добавления нового функционала к создаваемому приложению

Итак, были проанализированы факторы угроз рыночного внедрения проекта, среди которых: конкуренция, замедление роста рынка, изменение потребностей пользователей, изменение тарифов провайдера облачного развертывания на платные и поступления на рынок альтернативных продуктов. Были также предложены возможные реакции компании в ответ на изменение рыночных условий и появление новых продуктов, изменение предпочтений целевой аудитории.

Таблица 16 — Факторы возможностей

Фактор	Содержание возможности	Возможная реакция компании
Рост возможностей потенциальных покупателей	Рост финансирования у предприятий, которые занимаются работой с предметами быта	Предложить им свои услуги

Продолжение таблицы 16

Фактор	Содержание возможности	Возможная реакция компании
Снижение доверия к конкуренту 1	В приложении конкурента 1 недавно произошел сбой и в течение недели система неправильно функционировала	При выходе на рынок обращать внимание покупателей на надежность нашего сервиса

В таблице 17 приведены факторы возможностей рыночного внедрения проекта, среди которых: рост возможностей потенциальных покупателей и снижение доверия к конкуренту.

В дальнейшем проводится анализ предложения: определяются общие черты конкуренции на рынке.

Таблица 17 — Ступенчатый анализ конкуренции на рынке

Особенности конкурентной среды	В чем проявляется данная характеристика	Влияние на деятельность предприятия (возможные действия компании, чтобы быть конкурентоспособной)
1. Тип конкуренции: - совершенная	Существует 3 фирмы-конкуренты на рынке	Учесть цены конкурентных компаний на начальных этапах создания бизнеса, реклама (указать на конкретные преимущества перед конкурентами)
2. По уровню конкурентной борьбы: - международный	Все компании - зарубежные.	Добавить возможность выбора языка ПО, чтобы легче было в будущем выйти на международный рынок
3. По отраслевому признаку: - внутриотраслевая	Конкуренты имеют ПО, которое используется только внутри данной отрасли	Создать основу ПО таким образом, чтобы можно было легко его переделать для использования в других отраслях
4. Конкуренция по видам товаров: - товарно-видовая	Виды товаров являются одинаковыми, а именно - программное обеспечение	Создать ПО, учитывая недостатки конкурентов
5. По характеру конкурентных преимуществ: - неценовая	Совершенствование технологии создания ПО, чтобы себестоимость была ниже	Использование менее дорогих технологий для разработки, чем используют конкуренты

Продолжение таблицы 17

Особенности конкурентной среды	В чем проявляется данная характеристика	Влияние на деятельность предприятия (возможные действия компании, чтобы быть конкурентоспособной)
6. По интенсивности: - не марочная	Бренды отсутствуют	-

В таблице 17 приведен ступенчатый анализ конкуренции на рынке, где были определены особенности конкурентной среды и их влияние на деятельность предприятия. Одним из наиболее важных действий компании для достижения конкурентоспособности является необходимость создать основу ПО таким образом, чтобы можно было легко переделать данное ПО для использования в других отраслях.

После анализа конкуренции проводится более детальный анализ условий конкуренции в отрасли (таблица 18).

Таблица 18 — Анализ конкуренции в отрасли по М. Портеру

Составляющие анализа	Прямые конкуренты в отрасли	Потенциальные конкуренты	Поставщики	Клиенты	Товары-заменители
	Привести перечень прямых конкурентов	Определить барьеры вхождения в рынок	Определить факторы силы поставщиков	Определить факторы силы потребителей	Факторы угроз со стороны заменителей
Выводы	Существует 3 конкурента. 2-й имеет наиболее похожий продукт.	Да, возможности для входа на рынок есть, потому что наше решение упрощает и ускоряет работу специалиста.	Поставщики отсутствуют	Важным для пользователя является удобство в пользовании	Товары-заменители могут использовать более дешевую технологию и снизить себестоимость товара

На основе анализа конкуренции с учетом характеристик идеи проекта, требований потребителей к товару и факторов маркетинговой среды определяется и обосновывается перечень факторов конкурентоспособности (таблица 19).

По результатам анализа таблицы делается вывод о принципиальной возможности работы на рынке с учетом конкурентной ситуации. Также делается вывод о характеристиках (сильных сторон), которые должен иметь проект, чтобы быть конкурентоспособным на рынке. Второй вывод учитывается при формулировке перечня факторов конкурентоспособности. На основе анализа конкуренции, проведенного в таблице 18, а также с учетом характеристик идеи проекта (таблица 11), требований потребителей к товару (таблица 14) и факторов маркетинговой среды (таблицы 15–17) определяется и обосновывается перечень факторов конкурентоспособности. Анализ оформляется по таблице 19.

Таблица 19 — Обоснование факторов конкурентоспособности

Фактор конкурентоспособности	Обоснование (приведение факторов, делающих фактор для сравнения конкурентных проектов значимым)
Использование ПО в виде веб-сервиса в виде веб-сервиса	Позволяет наглядно увидеть работу ПО и правильность работы.
Простота интерфейса пользователя	Пользователь должен только загрузить фото.

По определенным факторам конкурентоспособности, которые показаны в таблице 19, проводится анализ сильных и слабых сторон стартап-проекта (таблица 20).

Финальным этапом рыночного анализа возможностей внедрения проекта является составление SWOT-анализа (матрицы анализа сильных (Strength) и слабых (Weak) сторон, угроз (Troubles) и возможностей (Opportunities) на основе выделенных рыночных угроз и возможностей, и сильных и слабых сторон. Это позволит сделать дополнить вывод о потенциале проекта.

Таблица 20 — Сравнительный анализ сильных и слабых сторон проекта

Фактор конкурентоспособности	Баллы 1–20	Рейтинг товаров-конкурентов по сравнению с нашим проектом						
		-3	-2	-1	0	1	2	3
Использование ПО в виде веб-сервиса	15			+				
Простота интерфейса пользователя	20	+						

Перечень рыночных угроз и рыночных возможностей составляется на основе анализа факторов угроз и факторов возможностей маркетинговой среды. Рыночные угрозы и рыночные возможности являются последствиями (прогнозируемыми результатами) влияния факторов, и, в отличие от них, еще не являются реализованными на рынке и имеют определенную вероятность осуществления (таблица 21).

Таблица 21 — SWOT-анализ стартап-проекта

Сильные стороны: простой пользовательский интерфейс, использование технологий	Слабые стороны: нужно иметь оборудование для тренировки модели, которая будет классифицировать изображения
Возможности: у конкурента 1 выявлена проблема с надежностью ПО, дополнительное финансирование для распространения данной технологии	Угрозы: конкуренция, изменение потребностей пользователей

На основе SWOT-анализа разрабатываются альтернативы рыночного поведения (перечень мероприятий) для вывода стартап-проекта на рынок и ориентировочное оптимальное время их рыночной реализации, учитывая потенциальные проекты конкурентов, которые могут быть выведены на рынок. Определенные альтернативы анализируются с точки зрения сроков и вероятности получения ресурсов (таблица 22).

Таблица 22 — Альтернативы рыночного внедрения стартап-проекта

№ п/п	Альтернатива (ориентировочный комплекс мероприятий) рыночного поведения	Вероятность получения ресурсов	Сроки реализации
1	Создание ПО используя нейронные сети	0.8	6 месяцев
2	Создание ПО на основе классических методов машинного обучения	0.3	12 месяцев

Из обозначенных альтернатив выбирается та, для которой: а) получение ресурсов является более простым и вероятным; б) сроки реализации - более сжатыми.

4.4 Разработка рыночной стратегии проекта

Разработка рыночной стратегии первым шагом предусматривает определение стратегии охвата рынка: описание целевых групп потенциальных потребителей.

Разработка рыночной стратегии первым шагом предусматривает определение стратегии охвата рынка: описание целевых групп потенциальных потребителей.

Целевая группа потенциальных клиентов для нашего проекта - магазины, торговые компании. Существует 3 конкурента, которые предоставляют похожие, но менее быстрые решения.

Возможность автоматической категоризации снижает нагрузку на работников — это будет ключевым фактором, который определит ценность продукта для целевой группы потребителей и покажет ценность проекта для рынка.

Среди плюсов проекта, которые выделят на фоне конкурентов:

– быстродействие;

– удобный пользовательский интерфейс;

– точность классификации;

– низкая стоимость;

– простота настройки.

По результатам анализа потенциальных групп потребителей (сегментов) выбираются целевые группы и определяются стратегия охвата рынка. Для работы в выбранных сегментах рынка необходимо сформировать базовую стратегию развития.

По М. Портеру, существуют три базовые стратегии развития, которые отличаются по степени охвата целевого рынка и типу конкурентного преимущества, которое должно быть реализовано на рынке (по затратам или выдающимися качествами товара).

Проиллюстрировать базовую стратегию развития можно в виде таблицы 23.

Таблица 23 — Определение базовой стратегии развития

Выбранная альтернатива развития проекта	Стратегия охвата рынка	Ключевые конкурентоспособные позиции в соответствии с выбранной альтернативой	Базовая стратегия развития
Создание ПО используя нейронные сети	Рыночное позиционирование	Быстродействие, простота в использовании, точность результатов	Дифференциация

Была выбрана такая следующая стратегия развития проекта: создание ПО используя нейронные сети (таблица 24).

Таблица 24 — Определение базовой стратегии конкурентного поведения

Является ли проект «первопроходцем» на рынке?	Будет ли компания искать новых потребителей, или забирать существующих у конкурентов?	Будет ли компания копировать основные характеристики товара конкурента, и какие?	Стратегия конкурентного поведения
Нет	Да.	Будет, а именно: основной задачей является разработка ПО с использованием нейронных сетей (конкуренты 1, 2, 3), простой интерфейс пользователя (конкурент 2)	Занятие конкурентной ниши

Итак, было определено базовую стратегию конкурентного поведения как занятие конкурентной ниши.

Определим стратегию позиционирования в таблице 25, которая заключается в формировании рыночной позиции (комплекса ассоциаций), по которому потребители должны идентифицировать торговую марку/проект.

На основе требований потребителей из выбранных сегментов к поставщику (стартап-компании) и к продукту, а также в зависимости от выбранной базовой стратегии развития и стратегии конкурентного поведения разрабатывается стратегия позиционирования, которая заключается в формировании рыночной позиции (комплекса ассоциаций), по которым потребители должны идентифицировать торговую марку / проект.

В конечном итоге, целью успешной стратегии позиционирования является создание уникального и убедительного образа бренда, который найдет отклик у потребителей и будет отличать стартап от конкурентов. Это

поможет стартапу привлечь и удержать клиентов, сформировать лояльность к бренду и в конечном итоге добиться устойчивого роста и прибыльности.

Таблица 25 — Определение стратегии позиционирования

Требования к товару целевой аудитории	Базовая стратегия развития	Ключевые конкурентоспособные позиции собственного стартап-проекта	Выбор ассоциаций, которые должны сформировать комплексную позицию собственного проекта (три ключевых)
Простота интерфейса, быстродействие, точность результатов	Дифференциация	Простота пользовательского интерфейса позволит получать необходимые данные и отслеживать события в режиме реального времени	Быстродействие, безопасность, простота, точность результатов

Итак, были выбраны такие ассоциации, которые должны сформировать комплексную позицию собственного проекта.

4.5 Разработка маркетинговой программы

Первым шагом является формирование маркетинговой концепции товара, который получит потребитель. Для этого в таблице 26 нужно подытожить результаты предварительного анализа конкурентоспособности товара.

Таблица 26 — Определение ключевых преимуществ концепции потенциального товара.

Ожидания	Выгода, которую предлагает товар	Ключевые преимущества (существующие или в планируемые)
Быстродействие	ПО работает быстро, результат можно получить до 10 мс	Преимущество в скорости
Простота пользовательского интерфейса	Простота работы приложения	Пользователи имеют удобный интерфейс для взаимодействия с ПО

Итак видим, что проект имеет ключевые преимущества перед конкурентами, которые полностью соответствуют потребностям целевой аудитории. Далее в Таблице 27 проиллюстрирована трехуровневая маркетинговая модель товара: уточняется идея продукта и / или услуги, его физические составляющие, особенности процесса его предоставления.

Таблица 27 — Описание трех уровней модели товара

Уровни товара	Сущность и составляющие
I. Товар по замыслу	ПО помогает выполнять автоматическую классификацию товаров
II. Товар в реальном исполнении	Свойства/характеристики: 1. Удобство и простота пользовательского интерфейса. 2. Скорость работы 3. Безопасность согласно мировым стандартам
	Качество: согласно стандарту ISO/IEC 25000 проводится тестирование
	Маркировка отсутствует
III. Товар с подкреплением	Мой проект: «Классификация изображений потребительских товаров»
	1-месячная пробная бесплатная версия
	Постоянная поддержка для пользователей
За счет чего потенциальный товар будет защищен от копирования: патент	

Было описано три уровня модели товара, из чего можно сделать вывод, что основные свойства товара в реальном исполнении являются

нематериальными и технологическими. Также было предоставлено сущность и составляющие товара в задумке и товара с подкреплением.

После формирования маркетинговой модели товара следует особо отметить - чем именно проект будет защищен от копирования. Защита может быть организована за счет защиты идеи товара (защита интеллектуальной собственности), или ноу-хау, или комплексное сочетание свойств и характеристик, заложенное на втором и третьем уровнях товара.

Следующим шагом является определение ценовых границ, которыми необходимо руководствоваться при установлении цены на потенциальный товар (окончательное определение цены происходит во время финансово-экономического анализа проекта), которое предусматривает анализ цены на товары-аналоги или товары субституты, а также анализ уровня доходов целевой группы потребителей (таблица 28). Анализ проводится экспертным методом.

Таблица 28 — Определение границ установления цены

Уровень цен на товары-заменители, USD.	Уровень цен на товары-аналоги, USD	Уровень доходов целевой группы потребителей, USD	Верхний и нижний пределы установления цены на товар/услугу, USD
250	500	20000	250–500

Следующим шагом является определение оптимальной системы сбыта, в рамках которого принимается решение (таблица 29).

Таблица 29 — Формирование системы сбыта

Специфика закупочного поведения целевых клиентов	Функции сбыта, которые должен выполнять поставщик товара	Глубина канала сбыта	Оптимальная система сбыта
Покупают подписку и делают ежемесячные взносы для продления лицензии	Продажа	0 (напрямую), 1 (через одного посредника)	Собственная и через посредников

Итак, система будет приносить прибыль благодаря ежемесячным платежам для продления лицензии и приобретением новых подписок, продажа будет выполняться напрямую или через одного посредника.

Последней составляющей маркетинговой программы является разработка концепции маркетинговых коммуникаций, опирается на предварительно выбранную основу для позиционирования, определенную специфику поведения клиентов (таблица 30).

Таблица 30 — Концепция маркетинговых коммуникаций

Специфика поведения целевых клиентов	Каналы коммуникаций, которыми пользуются целевые клиенты	Ключевые позиции, выбранные для позиционирования	Задачи рекламного сообщения	Концепция рекламного обращения
Использование оффлайн версии программы	Интернет	Быстродействие, простота в использовании, безопасность, нет надобности в интернете	Показать преимущества сервиса, в том числе и перед конкурентами	Демо-ролик по использованию

Итак, в таблице 30 приведена концепция маркетинговых коммуникаций, было определено, что приобретение лицензии на пользование будет осуществляться в сети Интернет, необходимым будет ежемесячное ее продление, пользование сервисом возможно оффлайн на компьютере пользователей, в облаке или на собственных серверах.

Согласно проведенных исследований существует возможность рыночной коммерциализации проекта. Также, стоит отметить, что существуют перспективы внедрения учитывая потенциальные группы клиентов, барьеры вхождения не являются высокими, а проект имеет два значительные преимущества перед конкурентами. Для успешного выполнения проекта необходимо реализовать программу с использованием средств языка Python и библиотеки Keras.

Для успешного выполнения проекта необходимо реализовать сервис с использованием нейронных сетей для классификации изображений. В рамках данного исследования были рассчитаны основные финансово-экономические показатели проекта, а также проведен менеджмент потенциальных рисков. Проанализировав полученные результаты, можно сделать вывод, что дальнейшая имплементация является целесообразной.

Выводы по главе 4

В рамках данного исследования были рассчитаны основные финансово-экономические показатели проекта, а также проведен менеджмент потенциальных рисков. Проанализировав полученные результаты, можно сделать вывод, что дальнейшая имплементация является целесообразной.

Были определены следующие сильные стороны: доступность реализации, себестоимость и простота настройки.

Имеющиеся такие факторы угроз: конкуренция, изменение потребностей пользователей, изменение тарифов провайдера, поступление на рынок альтернативных продуктов, замедление роста рынка.

Заключение

Данная исследовательская работа была посвящена изучению существующих методов и инструментов для классификации изображений, анализу алгоритмов классификации изображений с использованием нейронных сетей, разработке и тестированию выбранных моделей, а также разработке стартап-проекта под названием «Классификация изображений». В ходе исследования были определены различные методы, технологии и инструменты для классификации изображений и оценена эффективность различных архитектур нейронных сетей, функций активации, оптимизаторов и метрик для оценки и улучшения моделей.

Результаты показали, что выбор архитектуры, функции активации, оптимизатора и гиперпараметров, а также дополнительное обучение существенно влияет на производительность моделей. В ходе исследования также были разработаны и протестированы различные модели, и результаты показали, что разработанная модель достигла высокой точности и улучшила базовую модель. Кроме того, в исследовательской работе была разработана идея стартап-проекта по использованию нейронных сетей для классификации изображений и проведен анализ рыночных возможностей, финансово-экономический анализ и управление рисками.

В целом, исследовательская работа предоставила ценные сведения о методах классификации изображений и продемонстрировала потенциал использования нейронных сетей для задач классификации изображений. Идея проекта об использовании нейронных сетей для классификации изображений имеет коммерческий потенциал и может быть реализована с помощью языка Python и библиотеки Keras. Однако конкуренция, изменения в потребностях пользователей, изменения в тарифах поставщиков, выход на рынок альтернативных продуктов и замедление темпов роста рынка определены как потенциальные факторы угрозы, которыми необходимо управлять для обеспечения успеха проекта.

Мы провели детальное сравнение настроенных моделей, чтобы определить модель с наилучшими показателями для выбранной задачи, исследовали программную реализацию настроенных моделей, которая включала в себя интеграцию оптимизированных моделей в программную систему.

Наконец, мы протестировали разработанную модель и оценили ее значимость для исследования, рассмотрели результаты, полученные в процессе тестирования, и подчеркнули значимость полученных выводов.

В целом, была представлена практическая демонстрация использования нейронных сетей в анализе изображений. Мы показали важность оптимизации гиперпараметров и дали представление о процессе разработки и тестирования моделей для анализа изображений при помощи нейронных сетей.

Результаты и выводы данного исследования могут быть полезны для исследователей, практиков и разработчиков, работающих в области классификации изображений и нейронных сетей.

Список используемой литературы и используемых источников

1. Антонов А. С., Балашов Е.В. Применение нейросетей в медицине и биологии. // Московский физико-технический институт, 2017.
2. Богатырев С. А., Лазарева М. М. Применение нейронных сетей для генерации и обработки изображений // «Актуальные аспекты развития науки и общества в эпоху цифровой трансформации», сборник материалов III Международной научно-практической конференции, Москва, 2022. С. 134–143.
3. Жуковский А.Е., Лимонова Е. Е., Николаев Д. П. Реализация классических алгоритмов анализа изображений через полносверточные нейронные сети // Журнал «Труды Института системного анализа Российской академии наук», 2018. Т. 68, № S1. С. 108–116
4. Золотухин В.А., Копылова Е. В. Обзор методов классификации изображений. // Системы обработки информации, 2019. Т. 8, № 1.
5. Исхаков, А.Р. Моделирование систем технического зрения в модифицированных дескриптивных алгебрах изображений: Монография / А.Р. Исхаков, Р. Ф. Маликов. – Уфа: Изд-во БГПУ, 2015. – 160 с. [Электронный ресурс]. URL: https://bspu.ru/files/70771 (дата обращения: 15.10.2022).
6. Кудряшов С. Нейросети для анализа и классификации изображений. // Компьютерное моделирование и новые технологии, 2018. Т. 22, № 4.
7. Мустаев А.Ф. Применение нейросетей в распознавании изображений // Уфимский государственный технический университет, журнал «Вестник науки», 2019. Т. 3, № 7. С 53–57.
8. Нечипоренко А. Сравнение методов классификации изображений с использованием нейросетей. // Научный журнал «Инфокоммуникации и информационные технологии», 2020. Т. 1, № 1.
9. Павленко С. А. Использование сверточных нейронных сетей для анализа и классификации медицинских изображений. // Материалы

конференции «Проблемы и перспективы развития медицинской техники и технологий», 2019.

10. Русаков В.А., Курочкин А. А., Яковлев Д. А. Обзор алгоритмов машинного обучения для классификации медицинских изображений. // Вестник РУДН. Серия: Математика. Информатика. Физика, 2021. Т. 27, № 1.

11. Сорокина О. В. Обзор методов классификации медицинских изображений с использованием нейронных сетей. // Информационные технологии и вычислительные системы, 2020. Т. 3, № 1.

12. Харитонова Е. Н., Серов В. В. Обзор методов анализа медицинских изображений с применением нейросетей. // Вестник науки и образования, 2021. Т. 3, № 3.

13. Чанг Б., Спицын В. Г. Разложение цифровых изображений с помощью двумерного дискретного вейвлет-преобразования и быстрого преобразования // Изд. Том. политехн. ун-та. 2011. Т. 318. № 5. С. 73–76.

14. Шашков А.Н., Авдеев А. А. Методы классификации и сегментации медицинских изображений с применением нейронных сетей. // Электронный научный журнал «Технологии новых материалов и исследований», 2020. Т. 1, № 1.

15. Darken, C. Learning rate schedules for faster stochastic gradient search / Darken C., Chang, J. Moody, J. // Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, September 1–11, 1992.

16. Dauphin Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization / Dauphin, Y., Pascanu, R., Gulcehre C., Cho K., Ganguli S., Bengio Y. – [Электронный ресурс]. URL: http://arxiv.org/abs/1406.2572 (дата обращения: 17.11.2022).

17. Densely Connected Convolutional Networks – [Электронный ресурс]. URL: https://arxiv.org/pdf/1608.06993v3.pdf (дата обращения: 21.11.2022).

18. Fausett L. V. Fundamentals of neural networks: architectures, algorithms and applications. Upper Saddle River (USA): Prentice Hall, 1994. 476 p.

19. Flickner M., Sawhney H., NIBLACK W., ET AL. Query by image and video content: The QBIC system // IEEE Comput. 1995. V. 28, N 9. P. 23–32.

20. Gonzalez A. C., Sossa J. H., Felipe E. M. Wavelet transforms and neural networks applied to image retrieval // Intern. conf. on pattern recognition. Hong Kong (China), 20–24 Aug. 2006. P. 909–912.

21. Goodfellow I. Deep Learning / Goodfellow I., Bengio Y. and Courville A.; Cambridge MA: MIT Press [2017] – 777 pages.

22. Hackeling Gavin. Mastering Machine Learning with scikit-learn / Hackeling Gavin. - Packt Publishing Ltd. - 2014. – С. 1–32.

23. Kingma D. P., Adam: A Method for Stochastic Optimization / Kingma D. P., Ba J.L. // International Conference on Learning Representations, May 7–9, 2015, San-Diego, USA.

24. Lofti M., Solimani A., Dargazany A., et al. Combining wavelet transforms and neural networks for image classification // Proc. of the 41st Southeastern symp. on system theory. Tennessee (USA), Mar. 15–17, 2009. IEEE SSST, 2009. P. 44–48.

25. Niblack W., Barber R., Equitz W., et al. The QBIC project: querying images by content using color, texture, and shape // Proc. of the Intern. conf. on storage and retrieval for image and video databases. Bellingham, Washington, USA, Febr. 1993. IS&T/SPIE, SPIE, 1993. P. 173–187.

26. Nielsen M. Neural Networks and Deep Learning. – [Электронный ресурс]. URL: http://neuralnetworksanddeeplearning.com/ (дата обращения: 27.11.2022).

27. Park S. B., Lee J. W., Kim S. K. Content based image classification using a neural network // Pattern Recognition Lett. 2004. V. 25. N 3. P. 287–300.

28. Petzold J. Augsburg Indoor Location Tracking Benchmarks / Petzold J. // Technical Report, Institute of Computer Science, University of Augsburg, Germany. – 2004. – C. 1–10.

29. Robins. H. A stochastic approximation method // Annals of Mathematical Statistics / Robins H. and Monro S. – 1951 – vol. 22 – pp. 400–407.

30. Ruder S. (2016) An overview of gradient descent optimization algorithms. – [Электронный ресурс]. URL: https://arxiv.org/abs/1609.04747 (дата обращения: 25.10.2022).

31. Sermanet P. Traffic Sign Recognition with Multi-Scale Convolutional Networks / Sermanet, P., LeCun, Y. // The 2011 International Joint Conference on Neural Networks, September 2011.

32. Smith j. B. Chang S. F. Tools and techniques for color image retrieval // Proc. of the Intern. conf. on symp. on electronic imaging: science and technology storage and retrieval for image and video databases. San Jose (USA), Feb. 1996. IS&T/SPIE, SPIE, 1996. P. 426–437.

33. Springen Berg J. T. Striving for Simplicity: The All-Convolutional Net / Springenberg, J. T. Dosovitskiy, A.; Brox, T. & Riedmiller, M. – [Электронный ресурс]. URL: https://arxiv.org/abs/1412.6806 (дата обращения: 18.12.2022).

34. Sutton R. S. Two problems with backpropagation and other steepest-descent learning procedures for networks. Proc. 8th Annual Conf. Cognitive Science Society – 1986.

35. Swain M. J., Ballard D. H. Color indexing // Intern. J. Comput. Vision. 1991. V. 7, N 1. P. 11–32.