Международный университет информационных технологий

УДК: 004.85

На правах рукописи

МУХАНОВ САМАТ БАКЫТЖАНОВИЧ

Разработка и применение высокоточных методов распознавания образов

6D070400 – Вычислительная техника и программное обеспечение

Диссертация на соискание степени
доктора философии (PhD)

Научные консультанты
Доктор технических наук,
Профессор, Ускенбаева Р.К.
Зарубежный консультант
PhD, Профессор Енг И.Ч.

Республика Казахстан
Алматы, 2024

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Pattern recognition – Распознавание образов

CV – Computer Vision (компьютерное зрение)

CNN – Convolutional Neural Network, разновидность сетевой архитектуры для алгоритмов глубокого обучения.

KSL – Kazakh Sign Language (Казахский Язык Жестов)

ASL – American Sign Language (Американский язык жестов)

TSL – Turkish Sign Language (Турецкий язык жестов)

RNN – Рекуррентные нейронные сети (Рекуррентные нейронные сети)

LSTM – Long short-term memory – (Длинная цепь элементов краткосрочной памяти)

DL (Deep learning) – Глубокое обучение

ML (Machine Learning) – Машинное обучение

Python – Язык программирования общего назначения высокого уровня

MediaPipe – Алгоритм для применения скелетизации кисти рук суставов

HSV (Hue, Saturation, Value) – альтернативные представления цветовой модели RGB

RGB (Red Green Blue) – Аддитивная цветовая модель, в которой основные цвета света красный, зеленый и синий складываются различными способами

HGR (Hand gesture recognition) – Распознавание жестов рук

SVM (Support Vector Machine) – Машина опорных векторов

NN (Neural Network) – Нейронная сеть (НС), компьютерная программа (система), смоделированная на основе человеческого мозга и нервной системы.

Модель обучения – Набор данных, который используется для обучения алгоритма ML

DOF (Degree of Freedom) – степень свободы

Dataset – Набор данных, совокупность связанных наборов информации, состоящая из отдельных элементов, но которой компьютер может манипулировать как единым целым

Softmax – Функция активации

Cross-entropy – Перекрестная энтропия, мера разницы между двумя распределениями вероятностей для данной случайной величины или набора событий

Django – Расширенная веб-инфраструктура, написанная на Python, использующая архитектурный шаблон контроллера представления модели (MVC)

HOG (Histogram of Oriented Gradients) – гистограмма ориентированных градиентов

ВВЕДЕНИЕ

В настоящее время всё больше исследований направлено на решение задач с применением библиотек компьютерного зрения и инструментов искусственного интеллекта. Наиболее частыми являются решения и подходы с использованием моделей машинного и глубокого обучения искусственных нейронных сетей для распознавания жестов казахского жестового алфавита на основе методов обучения с учителем и глубокого обучения для обработки последовательных данных. Объектом исследования является казахский жестовый алфавит для построения коммуникации между людьми с ограниченными возможностями. Предметом исследования – методы машинного обучения и модели искусственных нейронных сетей и глубокого обучения для классификации и распознавания жестов. Методы исследования – Data Science, Machine Learning, Deep Learning, Neural networks и Computer Vision.

Распознавание образов — это изображение, на котором расположен объект. Так как объект является абстрактным (объектом может быть любая форма, изображенная на картине). Мы решили исследовать одно из актуальных направлений – распознавание жестов. Для распознавания казахского жестового языка, первым делом нужно изучить казахский жестовый алфавит. Для того, чтобы обучить нейронную сеть распознавать казахский язык жестов, необходимо собрать данные (датасеты) в формате изображений обозначенные жестами рук. Распознавания жестов – это задача классификации, которая является одним из направлений видов распознавания образов. Фундаментальной основой распознавания – является теория распознавания образов.

Целью исследования. Разработать и применить высокоточные методы распознавания образов. Для этого необходимо использовать инструменты искусственного интеллекта, а именно алгоритмы и библиотеки (язык программирование Python) для работы с машинным обучением и нейронными сетями глубокого обучения. Собрать данные для обучения и предварительно обработать для обучения нейронных сетей. Важно произвести математические расчеты для вычисления градиента функций и алгоритма обучения обратного распространения ошибки (backpropagation) и функций активации для каждого нейрона. Применить эмпирическиq подход к разработке собственной архитектуры моделей нейронных сетей для обучающих параметров и гиперпараметров. Архитектура данной модели может варьироваться как глубиной, так шириной нейронной сети, таким образом мы произвольно определяем количество обучающихся параметров. В диссертационной работе основное внимание уделяется изучению методов обучения с учителем, метода глубокого обучения, задаче классификации и распознавания жестов, обученных на собственных данных (изображения, полученные и разделенные на кадры из видеопоследовательности, снятые веб-камерой или мобильным устройством). Эти методы позволяют значительно расширить спектр задач, которые можно

эффективно решать в реальном времени в области распознавания жестов. Разработка программного обеспечения позволит протестировать эффективность работы обученной модели и применять ее в лабораторных целях, вносить корректировки для улучшения данной модели. Достижение поставленной цели приведет к повышению эффективности и расширению возможностей современных систем компьютерного зрения и распознавания жестов.

Для достижения поставленной цели необходимо было решить следующие **задачи:**

1. Собрать данных (изображения букв, представленные в виде жестов рук казахского жестового алфавита) и предварительная обработка в целях подготовки обучающего набора для методов машинного и глубокого обучения.
2. Анализ архитектур глубокого обучения и применение методов для распознавания жестов.
3. Обучить нейронную сеть распознавать казахский жестовый алфавит на собственных данных (изображениях).
4. Сделать сравнительный сверточной и рекуррентной нейронной сети глубокого обучения.
5. Разработать программное обеспечение для распознавания жестов (казахского жестового алфавита) в реальном времени.

Объектом исследования является обучение искусственной нейронной сети для распознавания казахского жестового алфавита с помощью инструментов искусственного интеллекта, а именно методов машинного обучения.

Предметом исследования является программное обеспечение системы распознавания жестов, разработанное на базе (основе) обученной модели глубоких нейронных сетей.

Научная новизна исследования определяется:

1. Собраны данные в формате изображений жестов рук и для обучения моделей искусственных нейронных сетей.
2. Обучены модели глубокого обучения для распознавания казахского жестового (дактильного) алфавита.
3. Построена собственная архитектура гибридной модели на основе метода обучения с учителем сверточной нейронной сети и метода глубокого обучения рекуррентной нейронной сети.
4. Разработано программное обеспечение для распознавания казахского жестового алфавита.

Научная положения, выносимые на защиту:

1. Собраны данные в формате изображений жестов рук для обучения моделей искусственных нейронных сетей.
2. Обучены модели машинного и глубокого обучения для распознавания казахского жестового алфавита.
3. Построена собственная архитектура гибридной модели на основе метода обучения с учителем сверточной нейронной сети и метода глубокого обучения рекуррентной нейронной сети.

4. Разработано программное обеспечение для распознавания казахского жестового алфавита.

Теоретическая и практическая значимость. Теоретическая значимость исследования заключается в разработке и обосновании нового подхода к решению задачи классификации для распознавания казахского жестового алфавита, обученного на собственных собранных данных для обучения модели искусственной нейронной сети на алгоритмах/методах машинного обучения.

Практическая значимость заключается в разработке программного обеспечения для распознавания алфавита казахского жестового языка в реальном времени.

Достоверность результатов. Достоверность результатов диссертационной работы подтверждается правильностью поставленных исследовательских задач и применением математических формул и расчетов, а также вычисления и получения результатов посредством обучения собственной нейронной сети на принципе работы архитектуры многослойного перцептрона, разработав гибридную архитектурную модель сверточной и рекуррентной нейронной сети и получив обученные и тестовые результаты на основе таких метрик как: accuracy, recall, precision и f1-score, confusion matrix (матрицы ошибок), а также программной реализацией и вычислительными экспериментами, проведенными в среде разработки PyCharm и Jupyter Notebook на языке программирования Python и библиотек для работы с компьютерным зрением и машинным обучением.

Апробация диссертационной работы. Получены акты внедрения по результатам исследования диссертационной работы в таких организациях как ТОО «Verigram» и ТОО «Smart-edu.kz». ТОО «Verigram» создает комплексные решения в области распознавания и верификации документов, лиц, объектов, а также внедряет технологии OCR и биометрии для улучшения качества обслуживания клиентов, защиты от мошенничества. ТОО «Smart-edu.kz» в свою очередь представляет образовательные курсы, предназначенные для профессионального развития и обучения взрослых лиц в различных областях и профессиях, например таких как компьютерная графика, Web-программирование и т. д. В данных компаниях в исследовательских целях пройдены тестирование программного продукта. Результаты о проделанной работе докладывались на следующих конференциях:

1. Uskenbayeva R.K., Mukhanov S.B. Contour analysis of external images, ACM International Conference Proceeding Series, 3410811, 2020.

2. Mukhanov S.B., Uskenbayeva R.K. Pattern Recognition with Using Effective Algorithms and Methods of Computer Vision Library, Advances in Intelligent Systems and Computing, 2020.

3. Mukhanov S.B., Tursunov S.A., Izteleuov N.E., Tazabekov A. (2019) data science and machine learning «Changing Kazakhstan Society Using Smart Technologies».

4. Aitulen A.D., Mukhanov S.B., Khassenova G.I. (2019) data science and machine learning «Face Recognition Through Various Facial Expressions».

5. Slyamkhan S.M., Yembergenov A.A., Bordousov N.S., Mukhanov S.B. (2019) data science and machine learning «Game Application with Machine Learning Elements».

Связь с государственными программами. Обзорная статья диссертационной работы была опубликована в Материалах Всемирного конгресса по глобальной оптимизации: 6th World Congress on Global Optimization, WCGO 2019 (Metz, France) в рамках реализации проекта программно-целевого финансирования (ИРН №BR05236517). Статья с полученными результатами была опубликована в журнале Eastern-European Journal of Enterprise Technologies, 5 (2-113), pp. 44-54 (процентиль 34) в рамках реализации проекта грантового финансирования (ИРН №AP08053034). Задачи, поставленные в данной диссертации, имеют высокую практическую значимость и непосредственно связаны с процессами внедрения цифровых технологий в производство. Вопросы цифровизации активно обсуждаются и получают особое внимание в выступлениях президента Республики Казахстан Токаева К.К. и в стратегических документах правительства, таких как "Стратегия "Казахстан-2050"" и "Государственная программа "Цифровой Казахстан"

Структура и объем диссертации. Диссертация состоит из введения, 4 основных глав, заключения, списка литературы и приложений.

Во введении дано обоснование актуальности исследования темы диссертационной работы. Поставлены и сформулированы цель, объект, предмет, методы и задачи научно-исследовательской работы. Описаны результаты экспериментов, получены результаты исследований, показаны их научная новизна, практическая значимость и апробация результатов исследования диссертационной работы.

В первой главе диссертации описывается теория распознавания образов для задач классификации. Рассмотрены системы распознавания образов, среда и область применения систем распознавания и требования к развитию и улучшению данных систем. Представлены задачи распознавания автомобильных номеров, распознавания лиц и распознавания жестов. Освещена область компьютерного зрения в распознавании изображений, а именно жестов рук.

Во второй главе описаны общие понятия языка жестов и разновидность международных языков жестов, а также дактильный алфавит жестового языка. Освящен и представлен казахский язык жестов, состоящий из 42 (сорока двух) букв в формате изображений. Теоретически описаны построение жеста руки по принципу векторной модели в трехмерном пространстве Эйлера. Математически представлена конструкция модели жестов в виде уравнений в Евклидовом пространстве. Описаны обозначения ключевых точек и обнаружение кончиков пальцев по контуру ладони, а также метод построения каркасного скелета кисти рук.

Третья глава диссертации содержит применение алгоритмов/методов распознания жестов. Применена и описана задача классификации на основе машинного обучения с использованием машины опорных векторов,

рекуррентных нейронных сетей на основе последовательности, сверточных нейронных сетей, а также глубокого обучения. Применена архитектура нейронных сетей для распознавания изображений, а именно жестов рук. Освящена важная часть структуры нейронной сети, в которой применяются картины в роли объектов для обучения, матричные фильтры, параметры свертки и композиция фильтров. Математически описана (вычислено формулами) и доказана (расчетами матричных произведений векторов для каждого слоя параметров – весов и смещений и применений сложных дифференцированных уравнений) важность применения алгоритма обучения (backpropagation) метода обратного распространения ошибок для вычисления градиента (gradient descent) и необходимость применения функции активаций для минимизации функции потерь (loss function).

Четвертая глава описывает эксперименты исследовательской работы, а именно как распознается казахский жестовый алфавит. Получены результаты с применением метрики для тестирования моделей глубокого обучения. Доказаны точность распознавания данных моделей на основе метрик точности (precision), полноты (recall), F1-score (мера) и поддержки (support) для каждого класса, а также общая точность (accuracy) и средние оценки (average). Протестирована построенная гибридная архитектурная модель нейронной сети с использованием слоев рекуррентной и сверточной нейронной сети. Разработано программное обеспечение, которое распознает жесты в режиме реального времени.

В заключениях каждой главы изложены основные результаты работы, выводы по диссертации и будущие шаги исследования.

1 ТЕОРИЯ РАСПОЗНАВАНИЯ ОБРАЗОВ. КОМПЬЮТЕРНОЕ ЗРЕНИЕ. ЗАДАЧИ РАСПОЗНАВАНИЯ ОБРАЗОВ

1.1 Теория о распознавании образов

Теория распознавания образов является главной предметной областью в теме диссертации.

Теория распознавания образов — это область исследования в сфере искусственного интеллекта и компьютерного зрения, которая занимается разработкой методов и алгоритмов для автоматического распознавания и классификации объектов и образов на основе их характеристик и признаков. Эта область имеет широкий спектр приложений, включая распознавание текста, изображений, звуков, голоса, лиц, жестов и многих других типов данных.

В отечественной литературе, образовании и науке в области распознавания образов автор книги «Теория распознавания образов и кластерного анализа» Амиргалиев Е.Н. представил в математическом виде задачи классификации, кластерного анализа, алгоритмы распознавания образов и методы машинного обучения [1]. В данной книге описываются основные понятия, теоремы, определения, а также математические формулы уравнений алгоритмов. Зарубежным аналогом данной работы является книга Кристофера М. Бишопа, по распознавания образов и машинного обучения.

Другой автор Мамырбаев О.Ж. внес значительный вклад и добился успеха в научно-исследовательских работах в распознавании речи казахского языка (аудиосигнал) [2], развивая и популяризируя государственный язык Республики Казахстан, что является одним из актуальных направлений в задачах распознавания на основе обработки временных рядов.

Процесс распознавания образов представляет собой когнитивный механизм, рассматриваемый в области психологии и когнитивной нейробиологии. Этот процесс включает в себя сопоставление входной информации, полученной из стимула, с данными, хранящимися в памяти [3]. Суть заключается в том, что информация из окружающей среды передается в кратковременную память, что автоматически активирует соответствующее содержимое долговременной памяти. Например, при изучении алфавита ребенок может предсказывать следующую букву, услышав "A, B", а также самостоятельно произносить любую букву. Этот процесс позволяет нам прогнозировать будущие события и требует повторения опыта, активируя семантическую память подсознательно. Распознавание образов играет важную роль в пространственном ориентировании, запоминании, а также обнаружении опасностей и ресурсов для обеспечения выживания [4].

1.1.1 Системы распознавания образов

Настоящая Система распознавания включает в себя методы и технические средства, которые взаимодействуют для выполнения процессов анализа и синтеза образов. Прежде чем создать систему распознавания, необходимо провести анализ доступной информации об объектах исследования. В системе

распознавания физические характеристики образов могут быть как простыми, так и сложными. Примером простой системы распознавания является классификация пикселей в мультиспектральных сканерах или оцифрованных изображениях, где признаками выступают спектральные свойства отражения объектов на земной поверхности. Однако, при использовании других типов данных в процессе классификации, система становится более сложной. Комплексные системы распознавания могут быть одноуровневыми или многоуровневыми. [5, 6].

Одноуровневые системы используют единый словарь признаков и унифицированный алгоритм распознавания, в то время как многоуровневые системы используют результаты распознавания на одном этапе в качестве входных данных для последующего этапа. Большинство технологий обработки данных в области распознавания образов используют сложные многоуровневые системы при решении задач, которые разделяются на уровни в процессе декомпозиции [6]. Интерактивный режим обработки информации может потребоваться на этапе синтеза образов из-за недостатка исходной информации или сложностей ее формального описания. Важной функцией аналитики данных в многоуровневых системах является оценка качества распознавания и потери информации на последующих этапах.

Системы распознавания могут быть классифицированы в зависимости от полноты имеющейся информации на системы с обучением, без обучения и самообучающиеся системы [7, 8]. В системах распознавания без обучения предполагается, что имеющаяся информация и выбранный принцип распознавания позволяют разделить все необходимые классы без ошибок. Примерами таких систем являются те, которые основаны на принципах сопоставления с образцом или кластеризации [9]. Однако, если данных недостаточно для точного разделения объектов, то распознавание без обучения может привести к значительным ошибкам. В таких случаях используются обученные системы, которые широко применяются в тематической обработке данных [10, 11]. Процедура обучения проводится аналитиком данных в интерактивном режиме. Однако следует отметить, что процедуры классификации с обучением требуют больше труда, и качество распознавания может зависеть от личного опыта аналитика данных и его понимания поставленной задачи, а не только от эффективности алгоритма распознавания.

1.1.2 Среда и область применения систем распознавания

Для обучения системы распознавания объектов на изображениях требуется предоставить ей данные/датасет, содержащие тысячи изображений с явно размеченными объектами. Существует разнообразие методов распознавания объектов на изображениях, но одним из перспективных подходов является использование метода гистограмм ориентированных градиентов (HOG). В данном методе изображение деструктурируется, а затем система вычисляет вектор градиента в блоках 16x16 пикселей, создавая карту этих векторов по всему изображению [12]. Эти векторы остаются постоянными независимо от

позы, положения и освещения объекта. Улучшенная версия алгоритма, известная как CoHOG, учитывает границы объектов, что позволяет распознавать не только векторы градиентов, но и форму объекта. Toshiba дополнила метод CoHOG, представив технологию гистограмм совместного присутствия ориентированных градиентов (ECoHOG) [13, 14]. Эта технология определяет человека с использованием дополнительного анализа размеров и направлений его очертаний, включая голову, ноги, руки и плечи. Это обеспечивает эффективное распознавание объектов даже в условиях недостаточного освещения, когда традиционный метод CoHOG [15, 16, 17] может быть менее эффективным. В области рекламы и маркетинга все более перспективным становится распознавание образов. С использованием нейросетей можно значительно ускорить процесс, который ранее требовал большой команды и множества недель исследований.

Например, сервис YouScan [18] в России применяет систему мониторинга социальных медиа для отслеживания упоминаний брендов в социальных сетях. Эта система не только анализирует тексты постов, но и обрабатывает фотографии, что позволяет делать выводы о продукте. Распознавание образов на фотографиях позволяет выявить интересные закономерности, на которые обычно не обращают внимание. Например, анализ фотографий выявил, что коты чаще связываются с техникой Apple, в то время как собаки ассоциируются с брендом Adidas. Эта информация может быть полезной для более целевой рекламы. Ниже приведен рисунок 1.1, демонстрирующий распознавание логотипа Adidas на одежде.

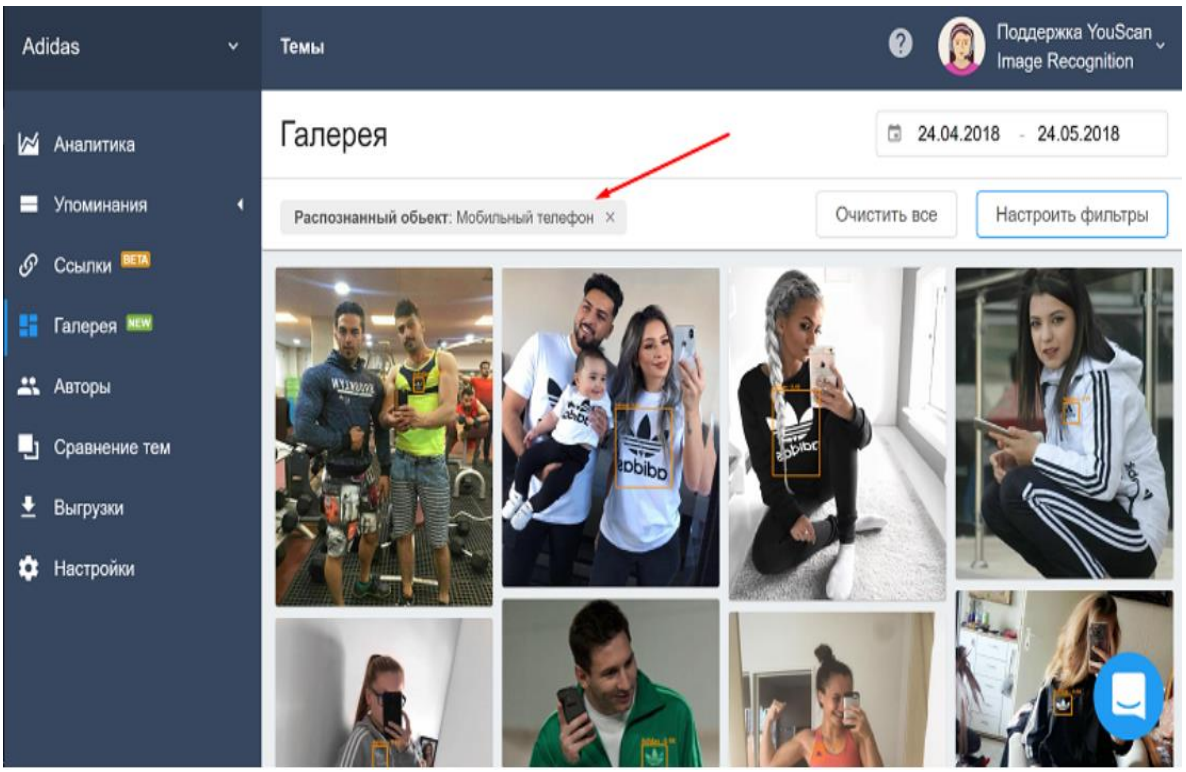


Рисунок 1.1 – Сервис YouScan

Активное использование машинного зрения находит перспективное применение в распознавании образов на камерах видеонаблюдения в городах. С 2017 года в Москве внедрена система умного видеонаблюдения, способная идентифицировать преступников в местах с высокой плотностью людей. Эта система основана на технологии, предоставленной российской компанией NTechLab, которая уже успешно помогла задержать несколько десятков нарушителей [19, 20]. Подобные системы видеонаблюдения также существуют в Китае, которые не только распознают лица, но также, но также способны определять марки автомобилей и даже одежду. Это может быть полезным для проведения маркетинговых исследований.

Рисунок 1.2 – Система распознавания образов и лиц от SenseTime

В сфере медицины внедрение компьютерных систем для распознавания образов представляет собой значительный прогресс, поскольку во многих случаях они способны выявлять аспекты, которые могли бы быть упущены даже опытными медицинскими специалистами [19, 20]. Эти системы действуют в качестве содействия, подтверждая гипотезы врачей и предоставляя дополнительные данные для более глубоких исследований. В России в настоящее время создаются программные комплексы для диагностики раковых заболеваний на изображениях КТ, МРТ и ПЭТ с использованием нейросетей для анализа тысяч размеченных снимков, что обеспечивает точность распознавания на уровне 95–97%. Один из таких комплексов разрабатывается Департаментом информационных технологий Москвы с использованием открытой библиотеки Google TensorFlow [21, 22].

1.1.3 Требования к развитию и улучшению систем распознавания

Исследовав алгоритмы и рабочие системы распознаваний на рынке IT-технологий и в научно-исследовательской области, мы изучили какие именно требования поставлены к развитию, улучшению производительности и эффективности данных систем. Системы распознавания жестов продолжают развиваться и совершенствоваться с целью улучшения точности и эффективности распознавания [19, с.13]. На пути развития и улучшения этих систем выделяются несколько ключевых направлений:

Расширение базы данных: для улучшения процесса обучения моделей требуется постоянное расширение баз данных, содержащих изображения и видео жестов.

Разработка новых методов обучения: Научные исследования должны сосредотачиваться на создании новых методов обучения, способствующих повышению точности распознавания жестов.

Улучшение архитектуры моделей: Разработчики регулярно работают над улучшением архитектуры моделей распознавания жестов, чтобы повысить точность и скорость распознавания.

Увеличение количества параметров: Увеличение параметров в моделях может улучшить их точность, однако это может сопровождаться увеличением времени обучения и размеров моделей.

Использование глубоких нейронных сетей: Применение глубоких нейронных сетей, таких как сверточные и рекуррентные сети, способствует улучшению точности распознавания жестов.

Использование комбинации методов: Комбинирование различных методов распознавания жестов, таких как машинное обучение и антропологические методы, может эффективно повысить точность распознавания.

Адаптация к контексту: Учет контекста и сцен, в которых происходят жесты, может существенно улучшить точность распознавания.

Разработка новых приложений: Создание новых приложений, использующих распознавание жестов, способствует улучшению и расширению возможностей систем распознавания.

Повышение точности распознавания требует разработки новых алгоритмов и методов машинного обучения, способных обеспечивать более высокую точность в процессе распознавания [9, 10, 11, с.11]. Дополнительно, сфокусированность на распознавании более сложных жестов включает в себя создание систем, способных идентифицировать не только простые жесты, но и более сложные варианты, такие как жесты, состоящие из нескольких компонентов или комбинаций нескольких простых жестов.

Следующие требования направлены на совершенствование систем распознавания жестов:

1. Распознавание в реальном времени: разработка быстрых алгоритмов и оптимизация вычислительной производительности для обеспечения работы системы в реальном времени.

2. Адаптивность к различным условиям освещения: создание систем, которые могут работать в различных условиях освещения, включая яркий солнечный свет, темноту и т. д.

3. Улучшение производительности на больших наборах данных: оптимизация алгоритмов и улучшение производительности на больших наборах данных, таких как видео с большим количеством фреймов или изображений с большим разрешением.

4. Адаптивность к различным камерам и устройствам: создание систем, которые могут работать с различными типами камер и устройств, включая камеры на мобильных устройствах.

5. Интерпретируемость результатов: обеспечение возможности интерпретировать результаты распознавания жестов, чтобы пользователи могли понимать, как система принимает решения и на каких основаниях.

6. Обеспечение конфиденциальности и безопасности: разработка методов для защиты персональных данных пользователей и обеспечения безопасности системы от злоумышленников.

1.2 Компьютерное зрение в распознавании

Компьютерное зрение занимает ключевую роль в искусственном интеллекте. Данная область представляет собой раздел искусственного интеллекта, фокусирующийся на создании методов и алгоритмов для автоматического анализа изображений и видеоматериалов. Эта область активно развивается, что приводит к постоянному расширению и улучшению возможностей компьютерного зрения. В рамках компьютерного зрения одним из важных направлений является распознавание жестов. Эта технология позволяет автоматически обрабатывать и анализировать изображения и видео в реальном времени [22, с.13], применяясь в различных областях, таких как медицина, робототехника, автомобильная промышленность, безопасность и другие. В контексте распознавания жестов компьютерное зрение автоматически определяет жесты на видео и использует их в разнообразных приложениях, включая управление устройствами, робототехнику, медицинские применения и др. [23, 24].

Эволюция компьютерного зрения включает постоянное совершенствование алгоритмов обработки изображений, разработку новых методов и архитектур нейронных сетей, улучшение качества обучающих выборок, а также интеграцию компьютерного зрения с другими технологиями, такими как голосовые и жестовые интерфейсы [25]. Одним из ключевых требований для развития и улучшения систем распознавания жестов является повышение точности распознавания и уменьшение числа ложных срабатываний. Важным аспектом также является увеличение скорости распознавания жестов, чтобы системы могли оперативно обрабатывать большие объемы данных в реальном времени [26, 27]. Кроме того, необходимо усовершенствовать устойчивость систем к различным условиям, таким как

изменения освещения, позы и масштаба объектов, а также расширять способности систем в обработке разнообразных типов жестов и их комбинаций.

Компьютерное зрение играет ключевую роль в распознавании жестов и действий, поскольку позволяет автоматически анализировать видео и изображения. Это особенно важно в областях мониторинга безопасности, управления роботами, интерфейсов виртуальной реальности и многих других. Для улучшения систем распознавания жестов следует продолжать развивать и совершенствовать методы компьютерного зрения, такие как глубокие нейронные сети и методы обучения с подкреплением, а также повышать качество обучающих данных [28]. Важно также учитывать разнообразие жестов и действий, чтобы системы распознавания могли успешно решать различные задачи и применяться в различных сценариях. Компьютерное зрение обладает огромным потенциалом в области распознавания жестов и действий, и его дальнейшее развитие будет способствовать созданию более эффективных и точных систем распознавания, что может оказать существенное влияние на множество аспектов жизни людей [29].

1.3 Задачи распознавания

Одной из задач диссертационной работы является задача распознавания образов в целом и конкретно – жестов. Задача распознавания, или распознавание образов (pattern recognition), заключается в идентификации объектов или явлений на основе набора характеристик или признаков. Эта задача является ключевой и активно исследуется в различных областях науки и техники, таких как компьютерное зрение, машинное обучение, робототехника, биометрия и других [30]. В общем случае процесс распознавания включает следующие этапы:

- Подготовка данных: этап включает в себя сбор и подготовку данных для последующей обработки. Здесь могут использоваться различные методы, такие как препроцессинг изображений (нормализация, уменьшение шума, фильтрация и т. д.), преобразование признаков (например, применение PCA или других методов сокращения размерности данных) и т. д.
- Выбор и обработка признаков: на этом этапе выбираются наиболее значимые признаки для описания объектов, и проводится их обработка, включая выделение границ, детекцию особых точек, описание текстур, применение фильтров и т. д.
- Выбор алгоритма классификации: происходит выбор алгоритма, который будет использоваться для классификации объектов на различные классы. В данном случае могут использоваться как классические методы (например, метод ближайших соседей, метод опорных векторов, логистическую регрессию), так и более сложные методы машинного обучения, включая нейронные сети.
- Тестирование и оценка качества: алгоритм тестируется на тестовых данных, и его качество оценивается с использованием различных метрик, таких

как точность, полнота, F1-мера и др. Эта процедура позволяет оценить эффективность алгоритма на новых данных.

- Улучшение качества: в случае недостаточного качества классификации применяются различные методы улучшения, такие как увеличение размера выборки, оптимизация параметров алгоритма, использование более сложных признаков и др.

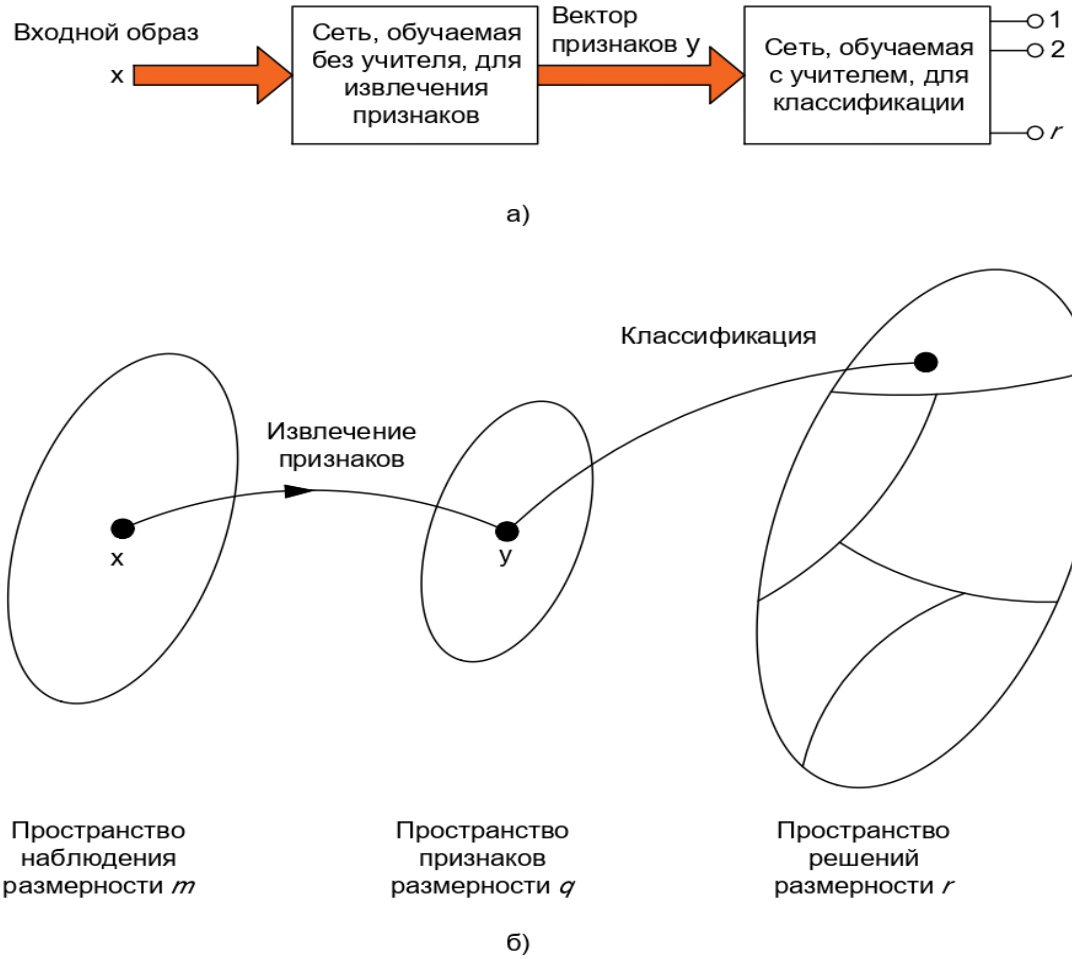
Задачи распознавания могут иметь разные цели и применения, такие как распознавание объектов на изображениях и видео, биометрическая идентификация людей и многое другое. Некоторые примеры задач распознавания с использованием компьютерного зрения включают распознавание лиц, жестов, объектов, текста, животных, эмоций, рукописного текста, цветов и флоры. Ниже представлены различные задачи распознавания [29, с.16], осуществляемого с использованием компьютерного зрения:

1. Идентификация или верификация лиц – задача распознавания человеческих лиц на изображениях или видео.
2. Классификация жестов – определение и классификация жестов, выполняемых на изображениях или в видео.
3. Классификация и обнаружение объектов – определение и обнаружение различных объектов на изображениях или видео, таких как автомобили, дома, люди и другие.
4. Распознавание текста – определение текста на изображениях и его преобразование в текстовый формат для последующего анализа.
5. Идентификация видов животных – определение и классификация видов животных на изображениях или видео.
6. Определение эмоций – задача распознавания эмоций на лицах людей на изображениях или видео.
7. Распознавание рукописного текста – идентификация и конвертация рукописного текста на изображениях в текстовый формат для последующего анализа.
8. Классификация цветов и растений – определение и классификация цветов и растений на изображениях или видео.

Распознавание образов формально определяется как процесс, в котором полученный образ/сигнал должен быть отнесен к одному из предопределенных классов (категорий). Чтобы нейронная сеть могла решать задачи распознавания образов, ее сначала необходимо обучить, подавая последовательность входных образов наряду с категориями, которым эти образы принадлежат [31, 32, 33]. После обучения сети на вход подается ранее не виденный образ, который принадлежит тому же набору категорий, что и множество образов, использованных при обучении. В целом, машины распознавания образов, созданные на основе нейронных сетей, можно разделить на следующие два типа.

1. Система состоит из двух компонентов: сети извлечения признаков (без учителя) и сети классификации (с учителем), как показано на рисунке 1.3а. Этот метод соответствует традиционному подходу к статистическому

распознаванию образов [34]. В терминах концепции образ представляет собой набор из *m* наблюдений, каждое из которых можно рассматривать как точку *x* в *m*-мерном пространстве наблюдений (данных). Извлечение признаков описывается преобразованием, которое переводит точку x в промежуточную точку *y* в *q*-мерном пространстве признаков, где $q < m$ (рисунок 1.3b). Это преобразование можно рассматривать как операцию снижения размерности (то есть сжатие данных), что упрощает задачу классификации. Сам процесс классификации описывается как преобразование, которое отображает промежуточную точку u в один из классов в *r*-мерном пространстве решений (где *r* - количество выделенных классов).



Рисунке 1.3 – представляет собой классический подход к распознаванию образов.

2. Система разрабатывается/проектируется в форме единой многослойной сети прямого распространения, применяя один из алгоритмов обучения с учителем. В данном контексте процесс извлечения признаков выполняется вычислительными узлами скрытого слоя сети.

Какой из этих двух подходов применять на практике – зависит от конкретной предметной области. Мы в своем исследовании применили

эмпирический подход, то есть экспериментировать с моделями обучения нейронных сетей с помощью алгоритмов/методов машинного обучения.

1.3.1 Распознавание автомобильных номеров

Одним из видов задач распознавания является задача распознавания автомобильных номеров.

Распознавание автомобильных номеров (Automatic Number Plate Recognition или сокращенно ANPR) – это технология из области компьютерного зрения [34, с.18], которая позволяет автоматически считывать номера автомобилей с помощью камер и специализированных программных алгоритмов. Распознавание автомобильных номеров – это задача компьютерного зрения, которая заключается в автоматическом распознавании текста на номерных знаках автомобилей [35]. Распознавание автомобильных номеров находит широкое применение в различных областях, таких как контроль скорости, системы безопасности, контроль транспорта на дорогах и автостоянках, розыск угнанных автомобилей, регистрация нарушений правил дорожного движения и т.д. [36].

Задача распознавания автомобильных номеров обычно включает в себя следующие шаги:

- Захват изображения – использование камеры для получения изображения автомобиля с номером. Это может быть сделано с помощью алгоритмов детектирования объектов, таких как метод каскадных классификаторов, а также с помощью методов, основанных на нейронных сетях, таких как Faster R-CNN, YOLO и SSD [37, 38, 39].
- Предобработка изображения – применение фильтров и других операций для улучшения качества изображения, к примеру уменьшение шума или улучшение контраста.
- Выделение номерного знака – выделение номерного знака из остальной части изображения, например, путем обнаружения прямоугольной формы и определения его положения и размера на изображении.
- Сегментация символов. С помощью этого этапа изображение номерного знака разбивается на отдельные символы, что позволяет упростить задачу распознавания текста. Для сегментации символов часто используются алгоритмы, основанные на анализе границ, такие как Canny edge detection, Sobel edge detection, а также методы, основанные на морфологических операциях [40, 41].
- Распознавание символов – определение символов, составляющих номерной знак, с помощью методов обработки изображений и машинного обучения. Существует множество алгоритмов для распознавания текста, в том числе традиционные методы, такие как шаблонное сопоставление, методы, основанные на признаках, такие как Histogram of Oriented Gradients (HOG). Также применяются нейронные сети, такие как CNN и LSTM.

- Проверка номера – сравнение распознанного номера с базой данных, чтобы проверить, является ли автомобиль украденным или подозрительным.
- Принятие решения – принятие решения, то есть отправка сообщения на полицейский участок или регистрацию данных об автомобиле в базе данных.

Работа с изображениями и распознавание текста на номерных знаках представляют собой сложные задачи, для решения которых используются разнообразные методы и алгоритмы [42, 43, 44]. Результаты распознавания могут зависеть от нескольких факторов, таких как качество изображения номера, угол обзора, освещение, наличие шума и другие. Для повышения точности распознавания применяются различные техники обработки изображений, такие как улучшение качества изображения с использованием фильтров, коррекция перспективы и угла обзора, выравнивание изображения, адаптивная бинаризация для сегментации номера, удаление шума, фильтрация контуров [45-48] и т.д. Для распознавания номеров на изображениях используются разнообразные алгоритмы машинного обучения, включая классификацию и сегментацию изображений. Алгоритмы классификации могут базироваться на методах распознавания шаблонов, машинного обучения и нейронных сетях. Алгоритмы сегментации могут использовать методы, такие как кластерный анализ, графические модели и методы машинного обучения. Для распознавания номеров на видео применяются алгоритмы отслеживания объектов, которые позволяют следить за движением автомобилей и изменениями номерных знаков в течение времени. Также используются алгоритмы машинного обучения для классификации номеров на изображениях, полученных с видеокамер. Распознавание автомобильных номеров широко используется в системах видеонаблюдения, на парковках с контролем доступа, в системах оплаты дорожных сборов и т.д. Кроме того, это также применяется в правоохранительных органах для идентификации автомобилей, связанных с преступными деяниями.

Существует система для распознавания номеров автомобилей компании ТОО «S-V.KZ». Это программный продукт NomerOk Lite для распознавания автомобильных номерных знаков с функцией отправки результатов детекции во внешние приложения [21, 22, с.13]. Базовая версия NomerOk Lite рассчитана на работу с одной камерой, для которой можно настроить 4 зоны распознавания. Принцип работы NomerOk Lite: при фиксации автомобильного номера в видеопотоке программа NomerOk сохраняет скриншот транспортного средства и делает запись в базу данных о событии (дата/время) с результатами распознавания. Версия программного обеспечения Lite включает модуль распознавания автомобильных номеров и предполагает возможность последующей интеграции с системами контроля доступа.

Рисунок 1.4 – Распознавание автомобильных номеров

В целом в последнее время технологии видеонаблюдения претерпели значительные инновационные изменения. Одной из наиболее востребованных задач в этой сфере является распознавание номеров автомобилей. Эта возможность видеотехнологий применяется не только для выявления преступников, но и для учета автомобилей, въезжающих на платные территории. Современные средства видеонаблюдения значительно снижают трудозатраты персонала при организации доступа на охраняемую территорию и в управлении парковкой. Более того, они также уменьшают риски и потери. Автоматизация системы с помощью распознавания номеров позволяет избежать человеческого вмешательства в различные процессы, которые ранее требовали его участия.

Рисунок 1.5 – Принцип распознавания автомобильных номеров

При въезде автомобиля на охраняемую территорию с использованием системы автоматизации камеры передают полученную информацию на видеосервер, где программное обеспечение осуществляет поиск и распознавание номеров. Затем полученный номер сравнивается с организованной базой данных [19, с.13]. В случае подтверждения данных система генерирует сигнал для разрешения доступа на объект. Процедура для въезда идентична алгоритму, используемому при выезде. Качество и четкость изображения напрямую зависят от технических характеристик камеры. Кроме того, настройка параметров и качественный монтаж также играют важную роль. Разрешение для распознавания номеров автомобилей должно быть достаточным, не слишком низким и не излишне высоким. Низкое разрешение может привести к неточностям в распознавании, в то время как слишком высокое разрешение может создать дополнительную нагрузку на сервер и увеличить светочувствительность из-за высокой плотности пикселей. Это, в свою очередь, может вызвать проблемы с чтением номеров в условиях ночного времени.

1.3.2 Распознавание лиц

Одним из важных этапов исследования диссертационной работы является область распознавания лиц. Данная область стала актуальной вначале 2000-х годов, когда произошел бум технологического прорыва. Технические устройства усложнились, и стало возможным интегрировать программное обеспечение современные девайсы и смартфоны. Далее данная область будет описана подробно и будут выделены ключевые особенности работы данного алгоритма/метода.

Распознавание лиц представляет собой задачу компьютерного зрения, заключающуюся в выявлении и идентификации лиц на изображениях или в потоке видео в режиме реального времени [29, с.15]. Эта технология широко применяется в системах безопасности, аутентификационных системах и автоматической классификации фотографий. Для распознавания лиц используются различные методы, включая те, что основаны на признаках, модельных методах и глубоких нейронных сетях [48, 49]. Методы на основе признаков направлены на извлечение характеристик лица, таких как его форма, расположение глаз и рта. Затем проводится сопоставление этих признаков с изображением в базе данных. Методы на основе моделей используют статистические модели, обученные на больших наборах изображений, чтобы определить характеристики, связанные с лицами. Глубокие нейронные сети используются для изучения обширных наборов данных с целью автоматического выявления характеристик, связанных с лицами, и последующего распознавания на основе этих характеристик [50-53].

Результаты распознавания могут зависеть от различных факторов, таких как качество изображения, уровень освещения, угол обзора, а также таких параметров, как возраст и пол. При использовании технологии распознавания

лиц также необходимо учитывать вопросы приватности и этики, поскольку это может привести к нарушению личных данных людей.

Системы распознавания лиц используют алгоритмы компьютера, чтобы выделить уникальные черты лица человека, такие как расстояние между глазами или форма подбородка [54, 55]. Эти черты конвертируются в математические данные и сравниваются с шаблонами лиц, которые хранятся в базе данных. Шаблоны лиц содержат только те детали, которые могут использоваться для отличия одного человека от другого и отличаются от обычных фотографий. Некоторые системы не предлагают однозначный результат, а вместо этого сравнивают неизвестное лицо с несколькими потенциальными шаблонами лиц, упорядоченными по вероятности правильной идентификации [7, 8, с.12].

Системы распознавания лиц могут иметь разную точность в условиях плохого освещения, низкого качества изображения и неправильного угла обзора. Ошибки систем распознавания лиц могут проявляться в форме ложных минусов (когда система не распознает человека) и ложных срабатываний (когда система ошибочно идентифицирует человека). При оценке эффективности системы важно учитывать как ложноположительные, так и ложноотрицательные результаты.

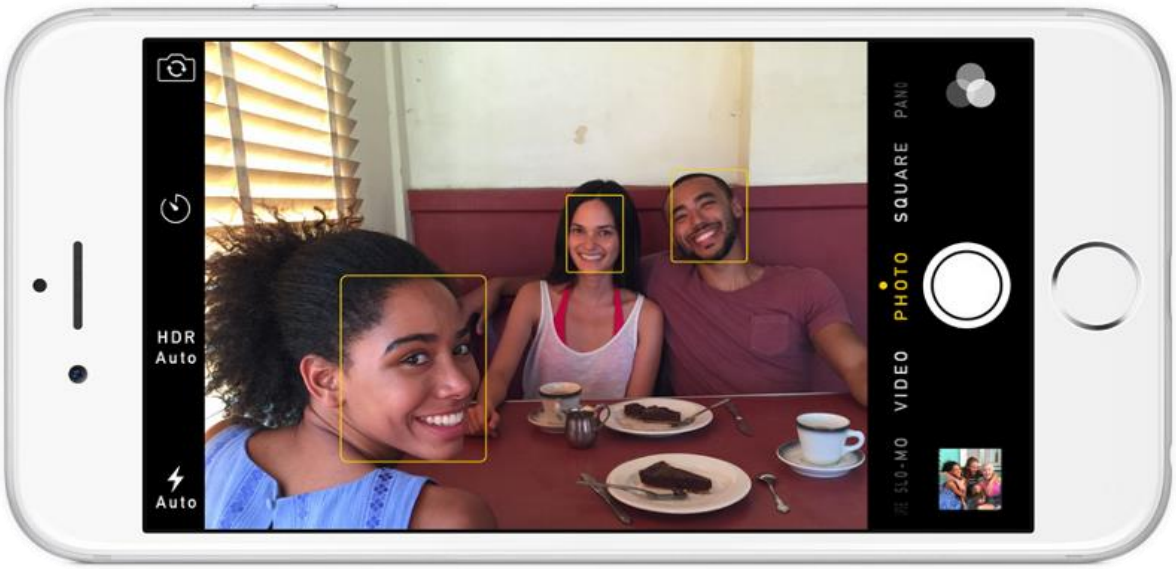


Рисунок 1.6 – Обнаружение лиц на основе алгоритма Viola-Jones

В последние годы полиция США активно применяет технологию распознавания лиц в повседневной практике. Она собирает фотографии арестованных, и сопоставляет их с базами данных по распознаванию лиц. Фотографии сохраняются в базах данных, и используются при расследовании уголовных случаев. Кроме того, правоохранительные органы могут обращаться к этим базам данных, для идентификации людей. Фотографии, полученные из социальных сетей, систем видеонаблюдения, дорожных камер или отснятых в полевых условиях. Технология распознавания лиц [7, 8, с.11] также может

использоваться в реальном времени для сравнения с "горячими списками", подозреваемых в преступной деятельности. С использованием мобильных устройств, таких как смартфоны и планшеты, полиция может фотографировать людей на месте и проверять их личность в базах данных по распознаванию лиц. Эта технология успешно применялась в аэропортах, пограничных пунктах и на масштабных мероприятиях, таких как Олимпийские игры. Кроме того, распознавание лиц может находить применение в частных местах, таких как магазины и спортивные стадионы, однако здесь могут действовать дополнительные правила. На текущий момент существует множество баз данных, которые поддерживают применение технологии распознавания лиц, и статистика указывает на то, что более 25% всех государственных и местных правоохранительных органов в США могут проводить поиски по распознаванию лиц в своих базах данных [58, 59]. По данным журнала "Управляющий" за 2015 год, не менее чем в 39 штатах использовали программное обеспечение распознавания лиц в базах данных Департамента транспорта (DMV) для выявления случаев мошенничества [24, с.6].

1.3.3 Распознавание жестов

Распознавание жестов представляет собой процесс, при котором компьютерная система анализирует движения и позы человека для определения выполняемого им действия или жеста [60, 61]. Эта технология находит широкое применение в различных областях, включая управление компьютерами и машинами, контроль производства, медицинские и спортивные приложения, а также в играх и других сферах. Разнообразные методы используются для распознавания жестов, такие как анализ движения, машинное обучение и компьютерное зрение. Камеры и датчики часто применяются для отслеживания движений человека и передачи информации на компьютер для дальнейшей обработки. Результаты распознавания жестов могут применяться в управлении системами умного дома, автоматизации производства, контроле качества продукции, диагностике заболеваний и других областях. Примером системы распознавания жестов является Kinect от Microsoft, которая позволяет пользователям управлять играми и приложениями с использованием жестов тела [62, 63].

Распознавание жестов представляет собой сложную задачу, требующую использования различных методов и технологий. Широко распространенные методы включают в себя машинное обучение, нейронные сети и антропологические подходы [64 - 68]. Машинное обучение применяется для создания моделей, способных распознавать жесты на основе предоставленных образцов. Нейронные сети используются для классификации жестов и определения траекторий движений рук. Антропологические методы основаны на изучении анатомии и движений человеческого тела.

Важным аспектом в распознавании жестов является использование различных алгоритмов и методов для сегментации и классификации жестов. Сегментация включает в себя выделение рук и других объектов на видео, а

классификация используется для определения типа выполняемого жеста. Существует множество подходов к решению этой задачи, включая машинное обучение, нейронные сети, анализ движения и антропологические методы. Каждый из этих подходов имеет свои преимущества и недостатки, и для достижения оптимальных результатов часто используется их комбинация.

Технология распознавания жестов на основе трекинга также заслуживает внимания. Она включает в себя метод, связывающий фрагменты границ в контуры с использованием внешнего скелета и радиальную функцию для анализа противоположных частей границ объекта и последующей сегментации его.

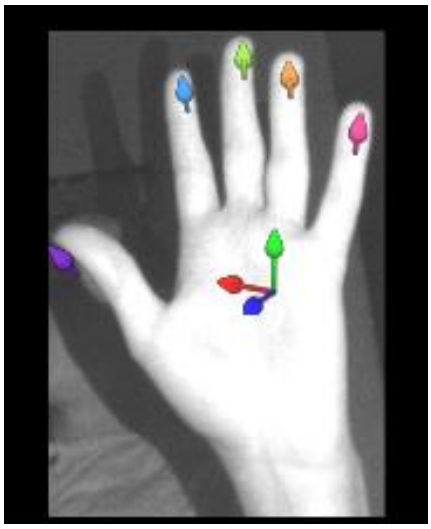


Рисунок 1.7 – Изображение кисти RealSense IR stream

На рисунке 1.7 показана технология RealSense IR stream, предоставляемая компанией Intel (Intel RealSense). Эта технология включает в себя различные виды датчиков, включая инфракрасные (IR) камеры, глубинные камеры и обычные RGB-камеры, которые работают вместе для создания трехмерных изображений и обеспечения функций распознавания жестов [69, 70, 71], отслеживания движений и других возможностей взаимодействия человека с устройством.

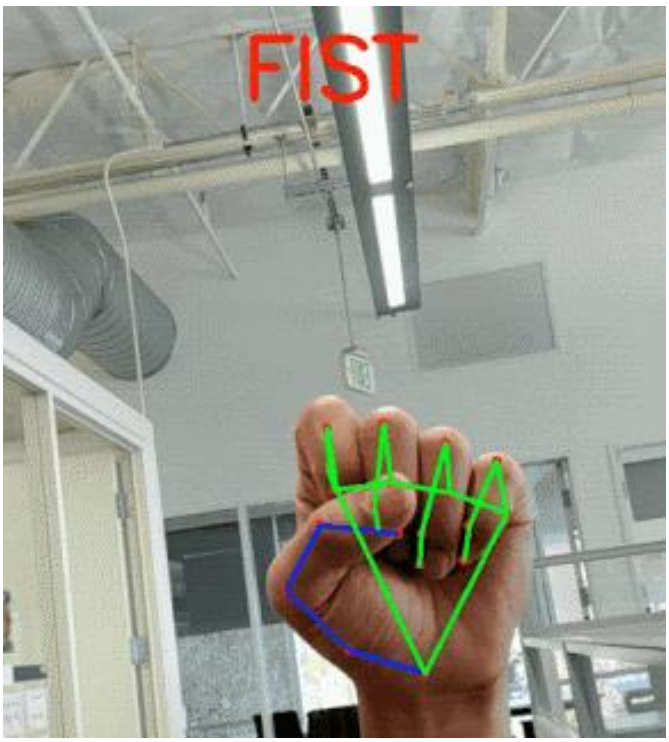


Рисунок 1.8 – Построение скелетной модели кисти рук

Данное изображение (рисунок 1.8) показывает построение скелетной модели кисти и распознавание данного жеста с набора обученных данных. Существует различные аналоги по распознаванию жестов ведущих компаний кремниевой долины, таких IT-гигантов как Google, Facebook (Meta) и др. [72, 73, 74]. Исследуя и изучая их опыт можно построить собственную нейронную сеть, которая в свою очередь будет распознавать казахский жестовый алфавит на собранной собственной коллекции данных изображений.

1.3.4 Кластерный анализ

Кластерный анализ — это метод машинного обучения, направленный на группировку схожих объектов данных в кластеры. Объекты внутри одного кластера считаются более похожими друг на друга, чем на объекты из других кластеров. Кластеризация является одним из методов обучения без учителя, поскольку в процессе не требуется размеченных данных с определенными классами [75]. Если углубиться в проблемы кластерного анализа, сущность кластерного анализа, а также разновидности кластерного анализ в первую очередь нужно сделать несколько предостережений общего характера.

1. Некоторые из методов кластерного анализа являются очень простыми процедурами, которые в свою очередь, не имеют достаточно статистического обоснования. Это означает, что большинство методов кластерного анализа [76] является эвристическими (то, что в работе используют программисты-разработчики). То есть не более чем правдоподобные алгоритмы, которые используется для создания кластеров объектов.

2. Методы кластерного анализа созданы и разработаны для многих научных дисциплин, поэтому представляют из себя признаки специфики данных дисциплин.

3. В результате анализа можно сказать, что разные кластерные методы [77] могут порождать и порождают разные решения для одних и тех же данных. Это означает обычное явление в большинстве прикладных исследований.

Процесс кластерного анализа включает в себя следующие шаги:

- Выбор метода кластеризации - существует множество методов кластеризации, таких как k-средних, иерархическая кластеризация, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), агломеративная кластеризация и др. [78, 79, 80]. Выбор метода зависит от характера данных и требований конкретной задачи.

- Определение метрики расстояния - выбор подходящей метрики расстояния между объектами является важным шагом. Распространенные метрики включают евклидово расстояние, манхэттенское расстояние, косинусное расстояние и т.д.

- Определение числа кластеров (при необходимости) - некоторые методы кластеризации, такие как k-средних, требуют определения количества кластеров заранее. Существуют методы [81, 82], такие как метод локтя (elbow method) или метод силуэта (silhouette method), которые могут помочь выбрать оптимальное количество кластеров.

- Стандартизация данных (при необходимости) - в случае использования методов, чувствительных к масштабу, может потребоваться стандартизация данных.

- Применение алгоритма кластеризации - применение выбранного метода кластеризации к данным и получение результирующих кластеров.

- Анализ результатов - Оценка качества кластеризации, например, с использованием внутренних (внутрикластерные расстояния) и внешних (межкластерные расстояния) метрик [83]. Визуализация результатов также может быть полезной.

- Интерпретация и использование результатов - интерпретация полученных кластеров в контексте конкретной задачи. Кластеризация может использоваться, например, для сегментации пользователей, выявления паттернов или упрощения структуры данных.

В кластерном анализе существует следующие меры (метрики) сходства [26, с.16]:

1. Евклидова метрика представляет собой наиболее распространенную меру сходства, часто используемую в анализе данных. Если, к примеру, объект описывается двумя параметрами, то его можно представить как точку на плоскости, где расстояние между объектами вычисляется как расстояние между соответствующими точками при помощи теоремы Пифагора. Квадратично возводятся расстояния по каждой координате, суммируются и из полученной суммы извлекается квадратный корень.

Расстояние $(x, y) = \sqrt{\sum(x_i - y_i)^2}$, (1.1)

2. Вычисление расстояния при использовании квадрата евклидовой метрики.

$$(x, y) = \sum(x_i - y_i)^2, \quad (1.2)$$

3. Манхэттенское расстояние, также известное как 'расстояние городских кварталов', рассчитывается путем взятия абсолютных значений покоординатных расстояний и их суммирования. Название этой метрики связано с тем, что она моделирует расстояние, пройденное человеком в городе, где перемещение возможно только по улицам, и невозможно, например, пройти квартал по диагонали. Аналогия на декартовой плоскости означает движение только вдоль линий, параллельных осям координат, что соответствует манхэттенскому расстоянию.

Расстояние $(x, y) \sum(x_i - y_i)^2$, (1.3)

4. Чебышевская метрика. $(x, y) = \max |(x_i - y_i)|$, (1.4)

На вопрос о том, какую меру сходства выбрать, нет однозначного ответа. Выбор зависит от типа данных и характера решаемой задачи. Кластерный анализ широко используется в различных областях, таких как анализ данных, обработка изображений, биоинформатика, маркетинг и другие.

1.4 Выводы по разделу

В первом разделе проведены исследования, направленные на обзор теории распознавания образов и систем распознавания. Проведен анализ существующих задач распознания. Рассмотрены задачи распознавания и их применение в диссертационной работе. Описана теоретическая значимость темы диссертационной работы и поставлены задачи научного исследования.

2 ЯЗЫК ЖЕСТОВ. ВЕКТОРНАЯ МОДЕЛЬ ЖЕСТОВ РУК. КЛЮЧЕВЫЕ ПРИЗНАКИ ДЛЯ РАСПОЗНАВАНИЯ

2.1 Языки жестов

По всему миру существует обширное разнообразие языков жестов, применяемых различными сообществами и культурами. Некоторые из этих языков обладают официальным статусом и признаются правительствами стран, тогда как другие представляют собой неофициальные или местные варианты языков жестов.

2.1.1 Разновидность языков жестов мира

Один из хорошо известных языков жестов – американский язык жестов (American Sign Language, ASL), представленный на рисунке 2.1. Этот язык применяется в США и Канаде, обладает своими уникальными грамматическими правилами и лексикой. ASL является основным средством общения для людей с ограниченными возможностями, страдающих от слуховой или речевой недостаточности в этих странах [83, 84, 85]. Еще одним примером языка жестов является британский язык жестов (British Sign Language, BSL), широко используемый в Великобритании. Этот язык также обладает своей грамматикой и лексикой и часто используется параллельно с американским языком [86, 87]. Кроме того, существуют местные варианты языков жестов, развивающиеся в определенных регионах и общинах. Возможно, что некоторые из этих языков имеют общие черты с другими языками жестов, но также могут включать уникальные элементы, связанные с культурой и историей соответствующего сообщества. Существует множество методов коммуникаций для людей с ограниченными возможностями, среди которых выделяется использование языка жестов, основанного на жестах рук.

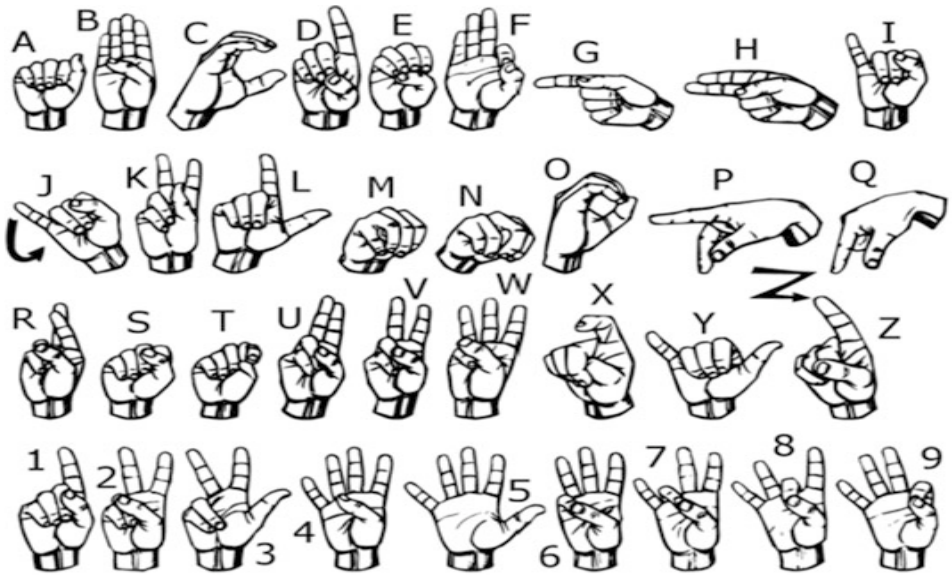


Рисунок 2.1 – Жесты рук американского языка жестов (ASL). Представлены буквы и цифры в количестве 36 жестов

Ниже представлен рисунок 2.2, где для каждой буквы («Z», «M», «G», «N») жестового алфавита показан уникальный жест руки, представленный в разных формах степеней свободы кисти рук, а точнее в различных позах.

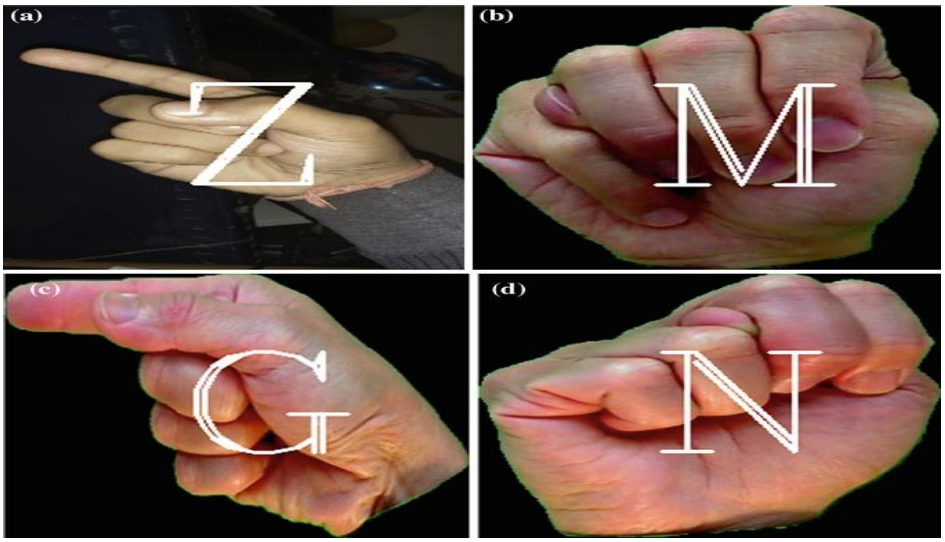


Рисунок 2.2 – Прогноз CNN, наложенный на тестовые изображения (a – d)

Распознавание турецкого языка жестов [89]. Собрано много данных для обучения жестов турецкого языка. Ниже представлено изображение (рисунок 2.3) алфавита турецкого жестового языка, состоящего из 29 букв.

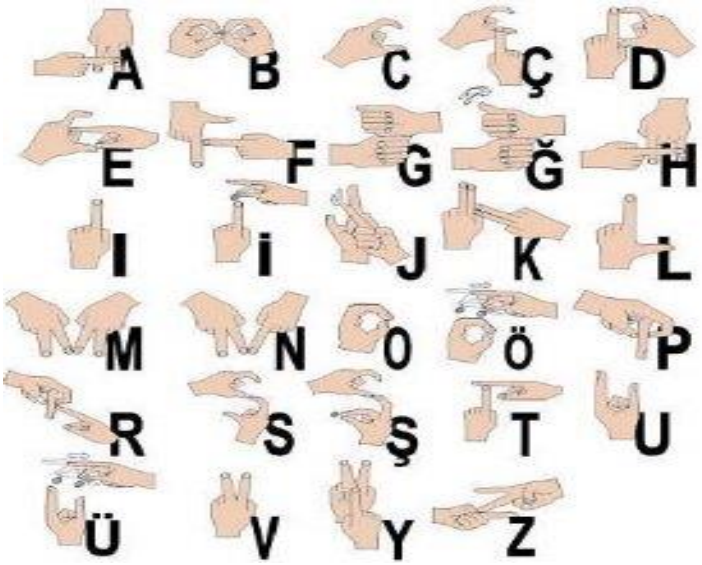


Рисунок 2.3 – Турецкий алфавит языка жестов

На рисунке 2.3 изображены жесты рук, представленные в разных обозначениях и положениях кисти рук турецкого жестового алфавита.

2.1.2 Казахский жестовый алфавит

Казахский дактильный (жестовый) алфавит представляет собой систему жестовой коммуникации, используемую глухонемыми и слабослышащими людьми на казахском языке. В этой системе 42 жеста обозначают буквы казахского алфавита и знаки препинания. Каждый жест формируется с использованием комбинаций движений пальцев, рук и запястья. Например, для обозначения буквы "А" используется жест с разведенными пальцами и поднятыми руками, в то время как для буквы "Б" выполняется круговое движение сжатой правой руки. Различные варианты казахского дактильного алфавита применяются в различных регионах, где проживают казахи. Обучение этой системе проводится в специализированных школах и центрах для слабослышащих и глухонемых людей.

В Казахстане примерно 200 000 человек имеют нарушения слуха. Согласно законодательству Республики Казахстан [39, с.10], каждому лицу с нарушением слуха предоставляются бесплатные услуги специалиста по языку жестов на 60 часов в год, однако это может быть недостаточно. Важность данной темы обусловлена общим интересом государства, представителей языка жестов и родителей детей с нарушениями слуха [39, с.10]. Исследования в этой области имеют социальную направленность, например, на оказание помощи людям с ограниченными возможностями.

Настоящее исследование фокусируется на создании системы перевода обычного языка на язык жестов, с возможностью передачи семантического значения перевода с казахского языка на казахский жестовый язык (КЖЯ). Результаты перевода представляются с использованием анимированного персонажа, что обеспечивает удобное визуальное представление жестов [90, 91]. Для достижения этой цели проект предлагает разрешить неоднозначность слов в предложениях во время морфологического анализа, и создание анимированного персонажа, управляемого анимациями на основе нотаций жестов КЖЯ, созданных с использованием системы нотаций Л.С. Димскиса.

Компьютерные системы перевода жестового языка, разработанные по всему миру, не всегда учитывают морфологические и синтаксические особенности языка. А также семантические аспекты жестового языка, которые существенны для повышения качества перевода [92, 93]. В контексте создания компьютерной системы перевода с казахского на казахский жестовый язык, особое внимание следует уделить учету семантической составляющей для обеспечения более качественного и понятного перевода. Процесс передачи семантической составляющей вводимого предложения в компьютерной системе перевода на казахском жестовом языке осуществляется путем разрешения омонимии на этапе морфологического анализа с использованием формализованного словаря лексических значений омонимов и многозначных слов. Учитывая уникальные особенности казахского языка, было принято решение разработать формализованный словарь лексических значений омонимов и многозначных слов на казахском языке. Этот словарь, в определенной степени аналогичный семантическому словарю В. Тузова,

предназначен для облегчения семантического анализа текстов. Однако, поскольку словарь В. Тузова охватывает перевод на семантический язык слов русского языка и предназначен для создания работающей компьютерной системы семантического анализа текстов на русском языке с его флективной структурой, который противопоставлен агглютинативной структуре казахского языка, перенос алгоритма семантического анализа по словарю В. Тузова на казахский язык невозможен. Тем не менее, можно воспользоваться некоторыми методами формализации для построения аналогичной системы, как предлагает сам автор. Ниже представлен Казахский жестовый алфавит в формате изображений для каждой буквы:

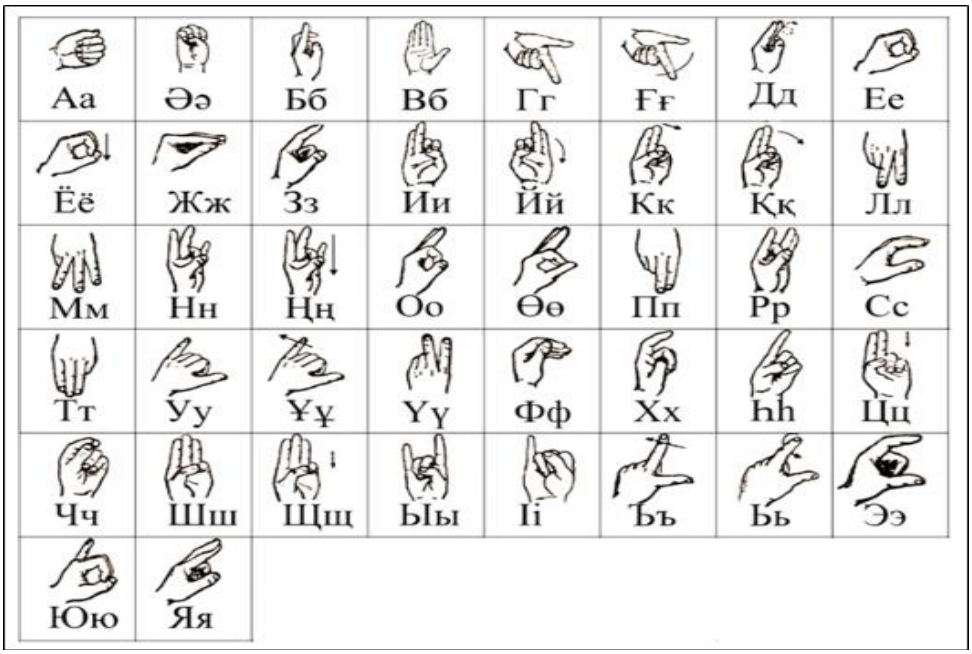


Рисунок 2.4 – Казахский дактильный (жестовый) алфавит

Рисунок 2.4 представляет Казахский алфавит жестов в виде изображений для каждой буквы [94]. Всего в алфавите 42 букв, а в свою очередь это означает 42 класса для задач классификации жестов. Данный алфавит был применен для обучения моделей глубоких нейронных сетей [11, с.13]. Он состоит из букв в виде изображений и является бесспорно отличной возможностью для сбора и подготовки данных (датасетов) для обучения нейросетей.

2.2 Векторная модель для описания и построения жестов руки

Векторная модель для описания и построения жестов руки представляет собой абстрактную модель, в которой каждый жест представлен вектором ключевых признаков и их координат [97]. Подробнее опишем эту модель с использованием математических формул и уравнений.

Для описания жестов руки мы будем использовать набор ключевых точек, которые представляют собой заметные анатомические места на руке, такие как кончики пальцев, суставы и другие особенности. Предположим, у нас есть n

ключевых точек на руке, и каждая точка будет представлена трехмерной координатой x, y, z в пространстве.

Определение вектора ключевых точек.

Для каждой точки определяем вектор ключевой точки P_i , где i – индекс точки, а

$$P_i = (x_i, y_i, z_i) \quad (2.1)$$

Определение вектора признаков жеста.

Вектор признаков жеста F будет представлен вектором, включающим координаты всех ключевых точек руки:

$$F = \{P_1, P_2, \dots P_n\} \quad (2.2)$$

Определение класса жеста. Каждому вектору признаков F соответствует определенный класс жеста. Этот класс можно определить с использованием методов машинного обучения, таких как классификация.

Обучение модели. Для распознавания жестов и обучения модели мы используем обучающий набор данных, в котором каждый жест представлен вектором признаков F и соответствующим классом. Обучение модели может включать в себя методы машинного обучения, такие как SVM CNN, LSTM и другие.

Распознавание жестов. После обучения модели мы можем использовать её для распознавания жестов руки, подавая на вход вектор признаков F. Модель будет классифицировать жест, определяя соответствующий класс.

Таким образом, векторная модель для описания жестов руки включает в себя вектор ключевых точек, вектор признаков жеста и методы машинного обучения для обучения и распознавания жестов. Эта модель позволяет абстрагировать и описывать различные жесты руки в математическом представлении.

Построение контуров для жеста руки – это процесс выделения границы объекта на изображении, который может быть использован для последующего анализа жестов. Этот процесс может быть выполнен с помощью различных алгоритмов компьютерного зрения. Один из методов построения контуров – это метод Canny [45, с.14], который широко используется в компьютерном зрении для обработки изображений.

Контуры, полученные после применения стандартных алгоритмов обнаружения границ на изображениях, часто не образуют замкнутых контуров и содержат разрывы. Эти разрывы часто возникают при использовании популярных детекторов границ, таких как Canny [45, с.16], который считается одним из лучших алгоритмов для обнаружения границ. Один из этапов алгоритма Canny включает в себя процесс прослеживания связанных границ, однако локальные критерии прослеживания, используемые в Canny, недостаточно эффективны для формирования замкнутых контуров. Самым простым методом объединения краёв является использование морфологических

операций, таких как расширение или морфологическое закрытие (последовательное расширение и эрозия) на двоичном изображении краёв. Однако такой подход может увеличить толщину краёв и исказить форму объектов, особенно если размеры промежутков превышают 1-2 пикселя. Ещё одним способом объединения краёв является поиск пар граничных точек контуров, которые находятся близко друг к другу, и их соединение отрезком. Существующие решения этой проблемы обычно основаны на локальном анализе контуров и не учитывают общую структуру изображения, что может привести к ошибкам в случаях, когда невозможно однозначно различить разрыв в контуре от узкой части объекта. Предлагается метод решения этой проблемы с использованием непрерывного скелета, который учитывает целостную структуру объекта. Для решения этой проблемы рассматривается изображение, полученное после применения оператора обнаружения краёв.

В результате применения разработанного метода объединения краёв улучшаются результаты обнаружения границ, полученные при использовании оператора Canny [45, с.5], и обеспечивается возможность получения непрерывных контуров, которые можно использовать для сегментации объектов на изображении. Кроме того, эффективные алгоритмы построения непрерывного медиального представления обеспечивают высокую скорость работы данного метода, что делает его пригодным для применения в системах компьютерного зрения реального времени.

2.3 Жесты рук в трехмерном пространстве Эйлера

Векторная модель жеста в пространстве Эйлера может использоваться для описания ориентации и движения объекта (например, жеста руки) с учетом угловой ориентации [98]. Пространство Эйлера обычно включает три угла, такие как угол крена, угол тангажа и угол курса (yaw, pitch, roll).

Математически представленная векторная модель жеста в пространстве Эйлера может быть выражена следующим образом:

- Угол курса (yaw), обозначается как ψ;
- Угол тангажа (pitch), обозначается как θ;
- Угол крена (roll), обозначается как ϕ.

Векторная модель описывает эти углы в виде вектора $E = [\phi, \theta, \psi\ ]$, где каждый угол представлен в радианах. Данй вектор E может представлять ориентацию объекта в трехмерном пространстве Эйлера.

Модель также может включать в себя скорости изменения углов, что позволяет описывать движение объекта в данном пространстве.

Пример векторной модели в пространстве Эйлера:

$$E = [\phi, \theta, \psi\ ], \tag{2.3}$$

где:

ϕ – угол крена (roll) в радианах;

θ – угол тангажа (pitch) в радианах;

ψ – угол курса (yaw) в радианах.

Это позволяет описывать ориентацию и движение объекта с учетом угловых параметров.

2.3.1 Построение единого и непрерывного каркаса

Определение непрерывного скелета вместе со способом его получения кратко описано в этом разделе. Концепция непрерывного скелета бинарных изображений, используемая в данном исследовании, следует общей теории, описанной в [21, с.11]. Более детальную информацию можно найти в книге [21, с.12].

Для многоугольной фигуры F максимальный пустой круг определяется как любой круг B, полностью содержащийся внутри фигуры F, такой, что любой другой круг B', содержащийся внутри фигуры F, не содержит в себе круга B, то есть $\forall B' \subset F B' \neq B : B \not\subset B'$. Скелет многоугольной фигуры F представляет собой множество центров ее максимальных пустых кругов. На скелете определена радиальная функция $R(x, y)$, которая присваивает каждой точке (x, y) скелета значение радиуса максимального пустого круга с центром в этой точке. Можно доказать [99], что скелет многоугольной фигуры состоит из объединения конечного числа отрезков и параболических дуг. Таким образом, скелет многоугольной фигуры можно рассматривать как геометрический граф – плоскую фигуру, состоящую из вершин – точек на плоскости – и ребер – линий, соединяющих некоторые пары вершин. Вершины этого графа – это точки пересечения отрезков и параболических дуг, а ребра – именно отрезки и параболические дуги, составляющие скелет. Степень любой вершины в таком графе будет равна 1, 2 или 3. Свойства скелета будут дополнительно использоваться как граф и как объединение кривых. При анализе видеопоследовательностей работа идет не с многоугольниками, а с растровыми двоичными изображениями, где один из двух цветов представляет принадлежность соответствующего пикселя к объекту. Следовательно, для построения непрерывного скелета сначала и прежде всего необходимо построить многоугольную аппроксимацию исходной фигуры. В [100] описаны эффективные алгоритмы для построения такой аппроксимации, используемые в рамках исследования и экспериментов. Впоследствии скелет можно построить для полученного многоугольника. Существующие эффективные алгоритмы [100] позволяют строить скелет за время $O(N \log N)$, где N - количество вершин в многоугольнике. После построения скелета обычно выполняется его дополнительная обработка, называемая "стрижкой", с целью удаления незначимых и шумовых ветвей. На рисунке 2.5 показан процесс построения скелета на примере изображения ладони. Для лучшей визуализации изображение ладони имеет низкое разрешение.

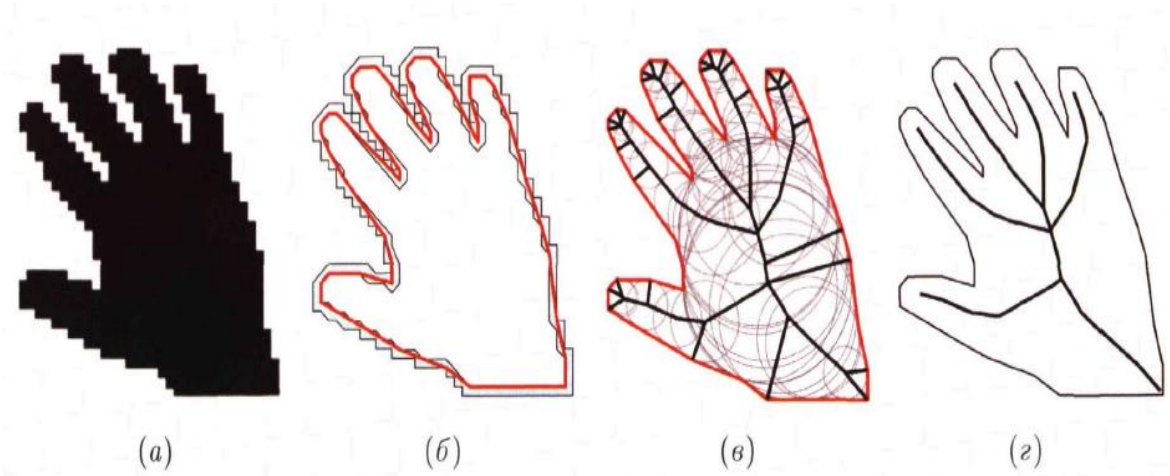


Рисунок 2.5 - Этапы формирования скелета: оригинальное бинарное изображение (а); приближенный многоугольник границы объекта (б); многоугольный скелет (в) и скелет после обрезки (г)

Каркас ладони, применяемый в жестовых языках, представляет собой структуру, используемую для выражения пространственных взаимоотношений и указания направлений движения.

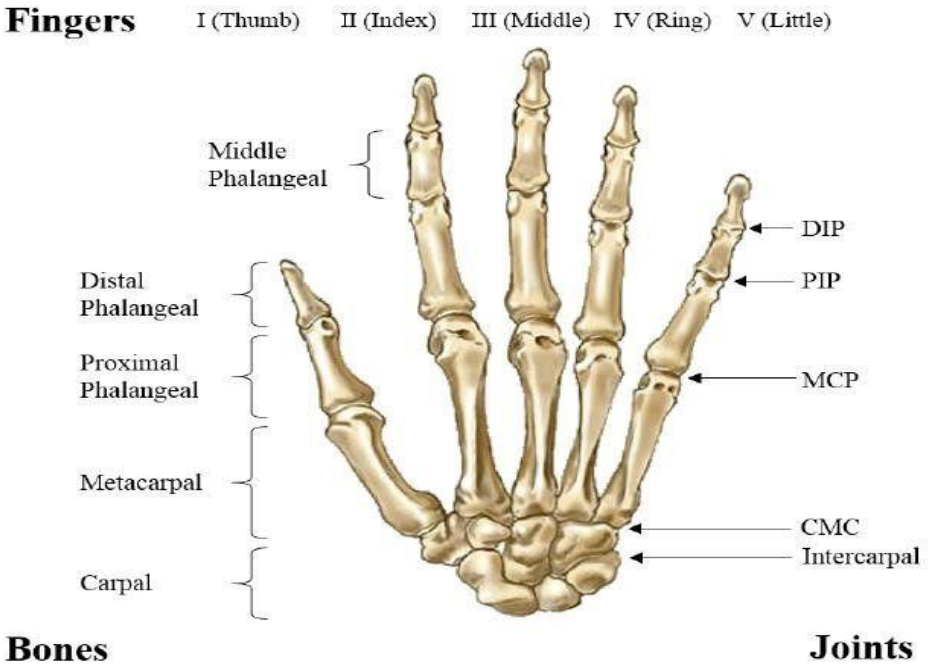


Рисунок 2.6 - Скелетная анатомия человеческой руки

Приведённая скелетная анатомия человеческой руки (рисунок 2.6) применяется как основной каркас для построения модели кисти рук. Освоение навыка создания единого и непрерывного каркаса ладони в жестовых языках представляет собой ключевой элемент для эффективного взаимопонимания с другими участниками жестового общения. Чтобы сформировать такой каркас, следует следовать определенным шагам:

1. Разместить ладонь на уровне плеч или выше, чтобы облегчить видимость для других;

2. Соединить кончики пальцев для создания структуры каркаса ладони;

3. Гарантировать непрерывность каркаса, перемещаясь вдоль контура каждого пальца от основания до кончика и переходя к следующему пальцу. Обеспечить сохранение целостности каркаса при переходе между пальцами;

4. Проверить свой каркас ладони, воспользовавшись другой рукой или зеркалом, чтобы удостовериться в его непрерывности и правильном положении каждого пальца.

Важно помнить, что единый каркас ладони охватывает не только непрерывную структуру пальцев, но также включает в себя плоскость ладони, которая должна быть направлена в соответствующем направлении. Правильное положение ладони способствует точному пониманию жеста и влияет на восприятие окружающими. Участие в классах жестового языка или общение с носителями жестового языка представляет собой эффективный способ улучшить навыки формирования каркаса ладони.

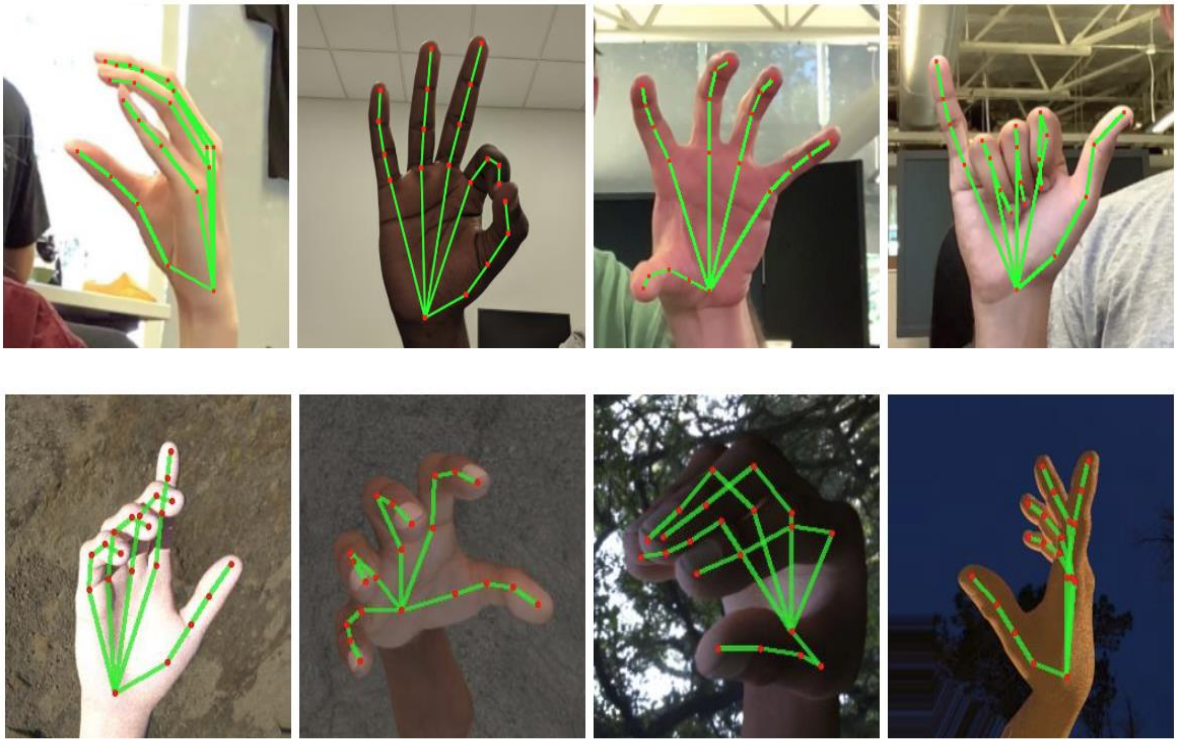


Рисунок 2.7 - Построение единого и непрерывно каркаса ладони

2.3.2 Свойства и ветка конструкции ладони

Каждая ветвь скелета представляет собой отдельный элемент структуры, рассматриваемый как непрерывная линия. Формальное определение данной концепции приведено ниже. Скелетная ветвь, или просто ветвь скелета, представляет собой отрезок структуры, соединяющий два разветвления (узла) в общей композиции скелета [99, 100]. Обычно ветвь обладает начальной и конечной точками, соответствующими начальному и конечному узлу. Свойства

ветви скелета могут быть использованы для анализа морфологии объекта, такого как лист, стебель или корень. Ряд характеристик ветви скелета включает:

1. Длину: используется для измерения длины сегмента объекта;
2. Угол: анализ морфологических особенностей объекта через угол между ветвями скелета;
3. Кривизну: измерение изгибов объекта и оценка его формы с использованием кривизны ветви скелета;
4. Толщину: оценка размеров объекта с помощью толщины ветви скелета;
5. Связность: определение связности объекта и его структуры на основе связности ветви скелета.

Эти характеристики извлекаются из скелета объекта с использованием алгоритмов компьютерного зрения и применяются для классификации объектов и анализа их формы и структуры [98]. В свою очередь, каркасная модель кисти представляет собой ветвь в конструкции ладони, состоящей из костей, связок и мышц. Эта ветвь обеспечивает поддержку и возможность движения руки, а также обладает следующими особенностями:

1. Гибкость: способность конструкции ладони адаптироваться к различным позам и формам, обеспечивая широкий спектр движений;
2. Прочность: выдерживание веса и силы руки во время движений и выполнения задач;
3. Компактность: удобство помещения конструкции ладони внутри камеры или на экране устройства для распознавания жестов;
4. Многоразовость: способность использования каркасной модели кисти для различных задач, включая распознавание жестов и управление устройствами;
5. Конструкция суставов: множество суставов обеспечивает множество направлений движения руки и выполнение различных жестов.

Эти свойства делают ладонь универсальным инструментом для разнообразных задач в ручном труде, спорте, музыке и других областях.

2.4 Жесты рук в Евклидовом пространстве

Жесты рук могут быть описаны с точки зрения Евклидова пространства, где каждый жест представлен вектором в данном пространстве [99, 100]. Рассмотрим более подробное математическое описание жестов рук с использованием формул и уравнений. Предположим, что у нас есть n ключевых точек на руке, и каждая точка представлена трехмерной координатой (x, y, z) в трехмерном геометрическом пространстве.

Определение векторов ключевых точек:

Для каждой точки определяем вектор ключевой точки $P_i = (x_i, y_i, z_i)$, где i - индекс точки.

Определение вектора признаков жеста:

Вектор признаков жеста F будет представлен вектором, включающим координаты всех ключевых точек руки:

$$F = (p_1, p_2, \dots, p_n), \tag{2.4}$$

Пространство жестов: Пространство жестов будет представлять собой линейное подпространство трехмерного геометрического пространства, определяемое векторами признаков жестов F.

Классификация жестов: Каждому жесту соответствует определенный класс или категория, которая определяется на основе анализа вектора признаков F.

Обучение модели: для обучения модели классификации жестов мы используем обучающий набор данных, в котором каждый жест представлен вектором признаков F и соответствующим классом. Обучение модели может включать в себя методы машинного обучения, такие как SVM, CNN, LSTM и другие [101, 102, 103].

Распознавание жестов: после обучения модели мы можем использовать её для распознавания жестов руки, подавая на вход вектор признаков F. Модель будет классифицировать жест, определяя соответствующий класс. Таким образом, линейная алгебра в геометрическом пространстве позволяет абстрагировать и описывать различные жесты рук в математическом представлении с использованием векторов и линейных операций. Это позволяет обучать модели для распознавания и классификации жестов.

Вычисление расстояний между ключевыми точками:

Для анализа жестов можно вычислить расстояния между ключевыми точками. Расстояние между двумя точками P_i, и P_j вычисляется с использованием формулы Евклидового расстояния:

$$d(P_i, P_j) = (x_i, x_j)^2 + (y_i, y_j)^2 + (z_i, z_j)^2, \tag{2.5}$$

Это расстояние может быть использовано как один из признаков для анализа жестов.

Углы между суставами руки: для оценки углов между суставами руки можно использовать тригонометрию. Например, угол между векторами $P_i - P_j$ и $P_j - P_k$ можно вычислить с использованием скалярного произведения и норм векторов:

$$cos(\theta) = \left|\left|P_i - P_j\right|\right| * \left|\left|P_j - P_k\right|\right| (P_i - P_j) * (P_j - P_k), \tag{2.6}$$

где θ - угол между векторами.

Распознавание жестов в геометрической плоскости:

Для упрощения анализа жестов можно проектировать координаты ключевых точек руки на двумерную геометрическую плоскость. Это позволяет работать с двумерными векторами и упрощает вычисления.

Методы машинного обучения. Для классификации жестов руки в геометрической плоскости или в трехмерном пространстве можно использовать

различные методы машинного обучения, такие как SVM, CNN или LSTM [103 - 106], в зависимости от специфики задачи.

Обучение и распознавание. Обучение модели включает в себя подачу обучающих данных в модель, где каждый жест представлен вектором признаков. Модель обучается определять классы жестов. После обучения модели мы можем использовать её для распознавания жестов руки, подавая на вход вектор признаков. Модель определит соответствующий класс жеста. Таким образом, описание жестов рук с использованием линейной алгебры, математического анализа и геометрической плоскости позволяет абстрагировать и анализировать жесты с учетом расстояний, углов и других геометрических характеристик. Модели машинного обучения помогают классифицировать жесты на основе их признаков.

Основным компонентом предлагаемого подхода является создание вектора признаков на основе координат ключевых точек. Предложенный алгоритм выполняет поиск ключевых точек среди висячих вершин скелета объекта. Каждая висячая вершина скелета вместе с ее ветвью классифицируется на два типа, обозначаемых как 1 и 0. Точки, принадлежащие классу 1, считаются ключевыми. Затем рассматриваются различные варианты этого алгоритма в зависимости от типа объекта, который рассматривается.

Создание основных характеристик жеста руки представляет собой ключевую задачу в области распознавания жестов [107,108]. Основные характеристики представляют собой параметры, описывающие форму, расположение и движение руки в момент выполнения жеста. Их извлечение позволяет представить жест в виде числового вектора, который может быть использован для обучения алгоритмов машинного обучения.

Существует множество методов формирования основных характеристик жеста руки. Одним из наиболее распространенных методов является использование базисных функций, заключающееся в представлении жеста как линейной комбинации базисных функций, описывающих форму руки в различных положениях. Другой метод включает в себя использование опорных точек, выделяя ключевые точки на руке, которые могут быть использованы для описания положения и формы руки во время жеста [109, 110]. Опорные точки могут быть выделены с использованием методов компьютерного зрения, таких как детекторы ключевых точек или алгоритмы сегментации.

Кроме того, существуют методы, основанные на анализе движения, такие как определение скорости и ускорения движения руки во время жеста. Эти характеристики могут быть использованы для определения динамических свойств жеста, таких как скорость, длительность и плавность движения. В современных исследованиях в области распознавания жестов также активно используется глубокое обучение для извлечения основных характеристик. Например, нейронные сети, такие как сверточные нейронные сети, применяются для извлечения признаков изображений руки, а рекуррентные нейронные сети используются для анализа последовательности движений руки

во времени. Формирование ключевых характеристик представляет собой важный этап обработки данных жестов, повышая точность их распознавания.

2.4.1 Обрисовка и обозначение (маркировка) ключевых точек

Для построения жеста руки в распознавании жестов ключевые точки могут быть представлены как векторы координат [100, 101, с.35]. Рассмотрим математическое и векторное представление для описания ключевых точек жеста руки:

Математическое представление (уравнения):

Допустим, у нас есть некоторое количество ключевых точек, обозначим их как $p_1, p_2, \dots, p_n$, где n - количество ключевых точек. Каждая ключевая точка может быть описана тройкой координат (x, y, z) в трехмерном пространстве:

Ключевая точка 1: $P_1 = (x_1, y_1, z_1)$

Ключевая точка 2: $P_2 = (x_2, y_2, z_2)$

...

Ключевая точка n: $P_n = (x_n, y_n, z_n)$

Векторное представление $-$ для векторного представления каждая ключевая точка будет представлена вектором вида $P_i = [x_i, y_i, z_i]$ $-$ в трехмерном пространстве [100, 101, с.35]. Все ключевые точки объединяются в единый вектор признаков, который описывает жест руки:

$$V = [P_1, P_2, \dots, P_n], \tag{2.7}$$

где $P_i = [x_i, y_i, z_i]$ $-$ вектор, представляющий координаты i-й ключевой точки. Этот вектор V описывает положение всех ключевых точек в жесте. Таким образом, вектор V представляет собой вектор признаков для данного жеста руки, и его можно использовать для распознавания и классификации жестов с помощью методов машинного обучения, нейронных сетей и других техник анализа данных. Для описания ключевых признаков жестов руки в математическом представлении формул мы можем использовать следующие параметры:

Положение ключевых точек относительно друг друга $-$ пусть $P_i = [x_i, y_i, z_i]$ $-$ координаты i-й ключевой точки в трехмерном пространстве.

Углы между суставами руки $-$ для описания углов между суставами руки можно использовать тригонометрические функции, такие как синус и косинус, чтобы выразить углы между векторами, соединяющими ключевые точки.

Расстояние между ключевыми точками $-$ P_i и P_j в трехмерном пространстве можно выразить с помощью 3D-формулы расстояния, такой как расстояние между двумя точками в трехмерном пространстве:

$$\text{Расстояние} \quad (d_{ij}) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{2.8}$$

Скорость и ускорение движений руки можно определить как изменение положения ключевых точек во времени, например, разницу между текущим и предыдущим положением.

Ускорение (*A*) можно определить как изменение скорости во времени, то есть разницу между текущей и предыдущей скоростью. Таким образом, ключевые признаки могут быть описаны с использованием указанных параметров в математических формулах. Вектор признаков для жеста руки будет включать в себя значения этих параметров, позволяя оценивать и классифицировать жесты.

В области распознавания жестов рук формирование ключевых характеристик начинается с выделения и обозначения основных точек на руке с использованием методов компьютерного зрения, таких как обнаружение объектов, сегментация изображения, а также выделение контуров и точек интереса [111]. После получения данных определяются ключевые точки, которые могут включать кончики пальцев, кисть, запястье, локоть и плечо. Для каждой точки задаются координаты в пространстве и другие характеристики, такие как углы, расстояния и относительные положения между ними. Вспомогательные данные, такие как показания давления, акселерометра и гироскопа, могут использоваться для получения дополнительной информации о положении и движении руки. После установления ключевых точек они могут быть использованы для вычисления различных характеристик жеста, таких как направление и скорость движения, форма кисти, а также позиция и ориентация руки в пространстве и т. д. Эти характеристики могут быть использованы для обучения алгоритмов машинного обучения и для распознавания жестов руки в режиме реального времени. После выделения и обозначения ключевых точек на изображении происходит формирование ключевых характеристик, описывающих положение руки и ее движения [112, 113, 114]. Эти характеристики могут включать в себя различные параметры, такие как:

1. Положение ключевых точек относительно друг друга;
2. Углы между суставами руки;
3. Расстояние между ключевыми точками;
4. Скорость и ускорение движений руки.

Для формирования ключевых характеристик могут применяться различные методы, такие как методы геометрического моделирования, машинного обучения и алгоритмы глубокого обучения. Один из таких подходов – использование метода главных компонент (PCA) [115, 116], который позволяет уменьшить размерность данных и выделить наиболее значимые характеристики. Для этого проводится анализ главных компонент, который выявляет взаимосвязь между ключевыми точками и позволяет сократить размерность пространства характеристик.

Другим подходом является применение сверточных нейронных сетей [117 - 120], которые способны автоматически извлекать характеристики из изображений. Сначала нейронная сеть обучается на обширном наборе данных, содержащем изображения различных жестов рук. Затем нейросеть может быть

использована для извлечения ключевых характеристик из новых изображений. Формирование ключевых характеристик представляет собой важный этап в обработке данных жестов, поскольку они позволяют детально описать положение и движения руки. Полученные ключевые характеристики могут использоваться для обучения алгоритмов машинного обучения и нейронных сетей, способных распознавать жесты рук на основе этих характеристик.

2.4.2 Обнаружение кончиков пальцев по контуру ладони

Обнаружение кончиков пальцев по контуру ладони может быть выполнено с использованием различных методов и алгоритмов обработки изображений. Одним из подходов может быть анализ контура ладони и поиск точек, представляющих кончики пальцев. Общий алгоритм описания этого процесса представлен математическими формулами. Предположим, что у нас есть контур ладони, который можно представить как последовательность точек контура $-$ $C = \{(x_i - y_i)\}$ где i - индекс точки контура.

Выделение выпуклых областей. Выделим выпуклые области контура, которые могут соответствовать пальцам. Для этого можно использовать алгоритмы поиска выпуклых оболочек, например, алгоритм Грэхэма или алгоритм Джарвиса [121, 122].

Определение кончиков пальцев. Найдем точки на контуре, которые находятся на некотором расстоянии от вершин выпуклых оболочек. Для этого можно использовать следующую формулу для расстояния между двумя точками (x_1, y_1) и (x_1, y_1) в двумерном пространстве:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.9}$$

Если точка на контуре находится на достаточном расстоянии от вершины выпуклой оболочки, то она может быть классифицирована как кончик пальца.

Алгоритмическое описание. Пройдемся по всем точкам контура ладони и найдем расстояния от каждой точки до вершин выпуклых оболочек. Если расстояние от точки до ближайшей вершины выпуклой оболочки превышает некоторый порог, то данная точка считается кончиком пальца. Полученные точки кончиков пальцев могут быть сохранены для дальнейшего использования. Этот алгоритм описывает обнаружение кончиков пальцев по контуру ладони на изображении [123, 124, 125]. Точные значения порогов и параметров могут варьироваться в зависимости от конкретной реализации и задачи.

Для каждой точки контура ладони $P_i = (x_i, y_i,)$ вычислим расстояние d_i от этой точки до каждой вершины выпуклой оболочки. Мы будем сравнивать расстояния и находить точку с максимальным расстоянием.

Пусть d_{ij} - расстояние от точки P_i до j-й вершины выпуклой оболочки, где j - индекс вершины выпуклой оболочки. Найдем индекс вершины с максимальным расстоянием:

$$j_{max} = arg j_{max} d_{ij}, \quad (2.10)$$

где $arg j_{max}$ означает индекс вершины, для которой расстояние максимально.

Определим пороговое расстояние T, при котором точка P_i считается кончиком пальца. Если $d_{ijmax} > T$, то точка P_i считается кончиком пальца. Все точки, удовлетворяющие этому условию, могут быть собраны в список или множество, представляющее кончики пальцев. Этот подход позволяет найти кончики пальцев, исходя из максимального расстояния до вершин выпуклой оболочки [126, 127]. Пороговое значение T может быть настроено в зависимости от конкретной задачи и характеристик изображения.

Для выявления кончиков пальцев на контуре ладони могут быть задействованы алгоритмы компьютерного зрения. Один из таких методов, известный как "Fingertips Detection Algorithm", базируется на анализе структуры контура ладони [128, 129]. Суть этого алгоритма заключается в выявлении выпуклых областей на контуре ладони, которые соответствуют кончикам пальцев. В начале необходимо обнаружить контур ладони, используя алгоритм выделения контуров, например, алгоритм Кэнни (Canny) [89, с.30]. Затем, на основе этого контура, выделяются выпуклые области, применяя алгоритм выделения выпуклых оболочек. После выделения выпуклых областей необходимо определить, какие из них представляют собой кончики пальцев. Это может быть достигнуто путем использования нескольких критериев, таких как длина выпуклой оболочки, угол между соседними выпуклыми областями и другие.

2.4.3 Метод построения каркасного скелета кисти руки

Метод построения каркасного скелета кисти руки в векторном представлении включает в себя определение координат суставов и соединений между ними [130]. Подробно опишем этот метод с использованием математических формул и уравнений. Предположим, что у нас есть набор ключевых точек, обнаруженных на кисти руки. Каждая точка представлена трехмерной координатой x, y, z в пространстве. Ниже приведены шаги по построению каркасного скелета.

Определение координат суставов:

Для начала определим координаты основных суставов кисти руки, таких как запястье (*Wrist*) и кончики пальцев (*FingerTips*). Эти координаты могут быть получены на основе обработки изображения.

Определение векторов соединений:

Определим векторы, соединяющие суставы. Например, вектор *VSE* соединяет плечо и локоть, вектор *VEW* соединяет локоть и запястье, и так далее.

Вектор между двумя точками $A(x_1, y_1, z_1)$ и $B(x_2, y_2, z_2)$ может быть определен как:

$$VAB = ||x_2 - x_1 y_2 - y_1 z_2 - z_1||, \quad (2.11)$$

Построение каркасного скелета:

Каркасный скелет будет представлен набором суставов и соединений. Это можно представить в виде вектора, содержащего координаты суставов и векторы соединений. Например, каркасный скелет можно представить вектором S:

$$S = \{Wrist, FingerTips, VSE, VEW, \dots\}, \quad (2.12)$$

Таким образом, каркасный скелет кисти руки представляет собой абстрактную модель, включающую координаты суставов и векторы соединений. Это векторное представление можно использовать для анализа жестов и движений руки, а также для обучения моделей машинного обучения в задачах распознавания жестов и управления интерактивными системами. Метод построения каркасного скелета кисти руки является одним из подходов к анализу жестов в компьютерном зрении. Этот метод использует алгоритмы компьютерного зрения для обнаружения и сегментации кисти руки на изображении, а затем строит каркасный скелет на основе расположения ключевых точек на изображении [131].

Данный метод находит широкое применение в различных сферах, таких как системы управления с использованием жестов, игры с жестовым управлением, системы жестового контроля для людей с ограниченными возможностями и другие.

2.4.4 Задача классификации

Для задач классификации нейронные сети с успехом решают огромное множество различных задач. Простая задача классификации предсказывает к какому классу относится тот или иной объект. Например, возможно по признакам пациента предсказать болен он или здоров, или же по картине с изображением цифры предсказать ноль это или единица или другая цифра. Нейронная сеть – это некоторая функция, у которой есть на входе и выходе некоторые числовые данные. Рассмотрим простейший пример. Пусть даны три числовые характеристики некоторого объекта x_1, x_2, x_3, и необходимо по ним сделать простое бинарное предсказание – да или нет. Например, это может быть давление, температура, возраст некоторого пациента, и нам нужно предсказать болен он или нет.

Классификация в машинном обучении представляет собой процесс присвоения объекту или экземпляру данных одной или нескольких категорий или классов на основе его признаков [132]. Обычно учебный набор данных используется для обучения модели, а затем модель применяется к новым данным для определения их принадлежности к определенным классам. Задача классификации в машинном обучении — это процесс разделения множества объектов на заранее определенные категории или классы. Основной целью является создание модели, способной автоматически присваивать новым, ранее не виденным объектам соответствующий класс на основе их характеристик или

признаков. Подходы к решению задачи классификации включают в себя использование обширного обучающего набора данных, который содержит объекты с известными метками классов, и обучение модели на основе этих данных. Задачи классификации в распознавании образов или жестов охватывают множество приложений, включая компьютерное зрение, робототехнику, взаимодействие человека с машиной и другие области [133].

Задача классификации в распознавании жестов заключается в том, чтобы классифицировать или определить конкретный жест, сделанный пользователем, и присвоить его определенному классу или категории. Жесты могут представлять собой различные движения, положения или конфигурации рук и пальцев, и задача состоит в том, чтобы научить модель распознавать эти жесты и относить их к определенным действиям или командам.

Допустим, у нас есть система умного дома, и мы хотим, чтобы пользователь мог управлять ей с помощью жестов. Задача классификации в этом контексте заключается в том, чтобы обучить модель распознавать жесты пользователя и соотносить их с определенными командами, такими как включение/выключение света, регулирование громкости музыки и другие управляющие действия.

Этапы решения задачи:

– Сбор данных – необходимо записать набор данных, содержащий различные жесты, которые пользователь может сделать. Каждый жест должен быть помечен соответствующим классом;

– Предварительная обработка данных – изображения или видео с жестами могут потребовать предварительной обработки, такой как изменение размера, выделение ключевых точек или фильтрация шума;

– Обучение модели – с использованием подготовленных данных модель машинного обучения (например, сверточная нейронная сеть) обучается классифицировать различные жесты;

– Валидация и тестирование – модель проверяется на отложенных данных (валидация) и далее на новых данных (тестирование), чтобы оценить ее способность обобщения в реальных условиях;

– Интеграция в систему – после успешного обучения и тестирования модель может быть интегрирована в систему умного дома, где она будет классифицировать жесты пользователя в реальном времени.

Такой подход к распознаванию жестов позволяет создавать интуитивные и удобные интерфейсы для взаимодействия с устройствами и системами, особенно в случаях, когда другие методы ввода могут быть неудобными или невозможными [134].

2.5 Выводы по разделу

Второй раздел содержит описание языка жестов, общих понятий языка жестов, а также разновидностей международных языков жестов. Описан казахский язык жестов. Рассмотрены теоретические основы, описывающие построение жеста руки векторной моделью в трехмерном измерении Эйлера.

Описаны математические формулы и представлены конструкции модели жестов в геометрическом (Евклидовом) пространстве. Рассмотрены и освещены обозначения ключевых точек и обнаружение кончиков пальцев по контуру ладони, а также метод построения каркасного скелета кисти рук.

3 КЛАССИФИКАЦИЯ ЖЕСТОВ НА ОСНОВЕ ИЗОБРАЖЕНИЙ. АРХИТЕКТУРА НЕЙРОННЫХ СЕТЕЙ

3.1 Алгоритмы распознавания жестов

В данной работе были применены различные алгоритмы распознавания жестов и методы машинного обучения нейросетей глубокого обучения. Далее будут описаны, какие именно алгоритмы были исследованы и применены для данной задачи.

Существует разнообразие алгоритмов распознавания жестов, применяемых в различных контекстах. Некоторые из наиболее популярных:

1. K-ближайших соседей (K-Nearest Neighbors, KNN): Этот метод опирается на обучающий набор данных для определения класса нового входного сигнала на основе его ближайших соседей в обучающем наборе [101, с.35];
2. Метод опорных векторов (Support Vector Machine, SVM): Алгоритм использует гиперплоскости для разделения данных на различные классы. SVM стремится найти гиперплоскость, максимизирующую расстояние между классами, что делает его эффективным для классификации жестов [101, с.35];
3. Нейронные сети: Этот класс алгоритмов машинного обучения пытается эмулировать работу нейронной системы человека и успешно применяется для распознавания жестов благодаря своей способности извлекать признаки из данных [103, с.3];
4. Метод главных компонент (Principal Component Analysis, PCA): Этот метод направлен на снижение размерности данных через преобразование их в пространство меньшей размерности. Это может улучшить производительность классификатора и сократить время обработки данных [104, с.35].

Каждый из этих методов имеет свои сильные и слабые стороны, и оптимальный выбор зависит от конкретной цели распознавания жестов.

3.2 Классификация жестов на основе машинного обучения

Классификация жестов при помощи машинного обучения осуществляется путем обучения алгоритма распознавания жестов на заранее подготовленных данных. Эти данные могут включать в себя изображения, видеозаписи или другие сведения, соотнесенные с различными жестами. Один из наиболее распространенных методов классификации жестов с использованием машинного обучения – метод опорных векторов (Support Vector Machines, SVM). SVM применяется для разделения данных на классы и может успешно использоваться для классификации жестов. Работа алгоритма SVM заключается в поиске гиперплоскости, наилучшим образом разделяющей данные по классам [78, с.27; 97, 98, 99, с.30].

Еще одним востребованным методом классификации жестов с применением машинного обучения является метод нейронных сетей (Neural Networks). Нейронные сети, состоящие из взаимосвязанных нейронов, могут обучаться на данных и применяться для классификации жестов. Другие методы

машинного обучения, такие как решающие деревья, случайный лес, алгоритм k-ближайших соседей и так далее, также могут использоваться для классификации жестов [101, с.35].

Значительным аспектом классификации жестов, основанной на машинном обучении, является подготовка и обработка данных. Для эффективного обучения алгоритма необходимо собрать достаточное количество данных, соответствующих различным жестам. Предварительная обработка данных может включать в себя улучшение качества изображений или выделение характерных признаков. Также важным этапом в процессе классификации жестов с применением машинного обучения является выбор подходящих признаков для обучения модели. Этот метод остается одним из наиболее востребованных в области распознавания жестов, поскольку он опирается на алгоритмы машинного обучения для создания моделей, способных классифицировать жесты на основе извлеченных из изображений признаков [111, с.42].

3.3 Архитектура нейронных сетей для распознавания изображений

Структура нейронной сети, или ее архитектура, определяет взаимосвязь и организацию нейронов (или узлов) в сети, а также функциональные задачи, выполняемые каждым слоем сети в контексте машинного обучения [120 – 123, с.42]. Вот основные элементы и понятия, связанные с архитектурой нейронных сетей:

Слои (Layers). Нейронные сети разделены на различные слои, каждый из которых выполняет определенные операции. Основные типы слоев включают входной слой, скрытые слои и выходной слои. Входной слой (Input Layer) принимает входные данные и передает их внутрь сети. Число нейронов в этом слое соответствует размерности входных данных. Скрытые слои (Hidden Layers), находящиеся между входным и выходным слоями, выполняют вычисления и извлекают признаки из входных данных. Количество и размерность этих слоев может изменяться в зависимости от архитектуры сети. Выходной слой (Output Layer) представляет конечный результат работы нейронной сети. Количество нейронов в выходном слое зависит от задачи, например, для классификации может быть несколько нейронов, соответствующих числу классов [123, с.42].

Связи определяют, как нейроны одного слоя соединены с нейронами другого слоя. Каждая связь обладает весом, регулируемым в процессе обучения.

Функции активации (Activation Functions) управляют тем, какой выходной сигнал передается от одного нейрона к следующему. Эти функции придают нейронной сети нелинейность, что позволяет моделировать более сложные функции.

Сверточные нейронные сети изначально разработаны для обработки изображений. Они используют комбинацию сверточных слоев и слоев субдискретизации (пулинг). Каждый фрагмент изображения умножается на

элементы сверточной матрицы, а затем результат суммируется и записывается в соответствующее положение на выходном изображении [135]. Метод обратного распространения ошибки широко используется для их обучения. Важно понимать, что RGB-изображение представляет собой матрицу с тремя плоскостями, содержащими значения пикселей.

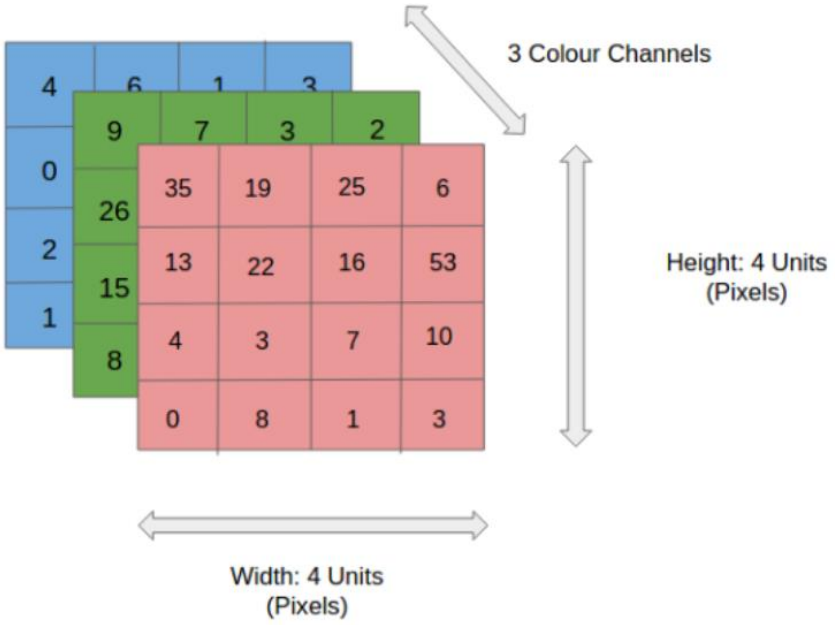


Рисунок 3.1 – Структура RGB-изображения

Однако у изображения в оттенках серого есть только одна плоскость. Рисунок 3.2 демонстрирует работу CNN для изображения в оттенках серого.

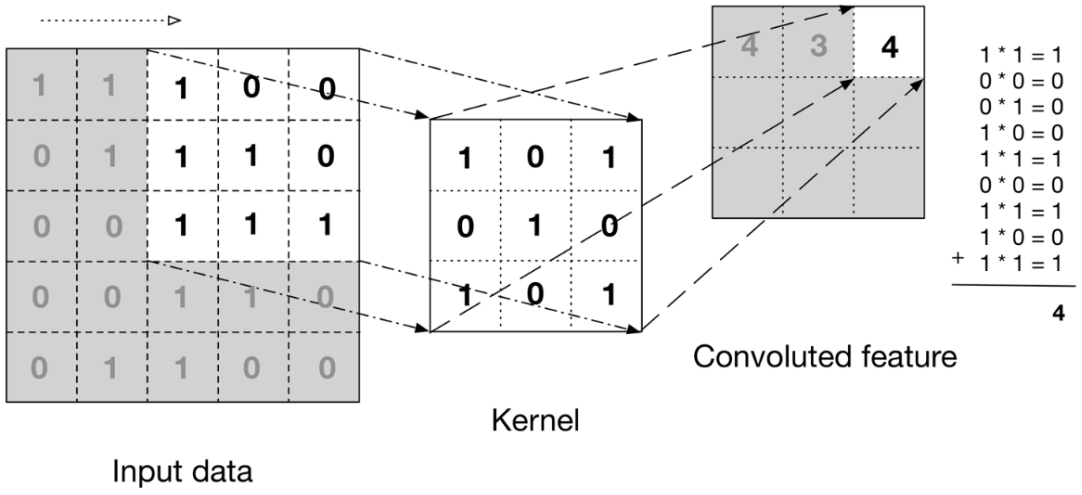


Рисунок 3.2 – Принцип работы сверточных нейронных сетей для изображения в оттенках серого

Далее, используем ядро, представленное в виде матрицы 3x3, и применяем его к входному изображению, после чего получаем свернутую функцию. Затем эта функция передается на следующий слой.

Сверточные нейронные сети содержат несколько слоев искусственных нейронов [117, с. 42]. Данные нейроны, как бы подражая своим биологическим аналогам, представляют собой математические функции, которые вычисляют взвешенную сумму нескольких входов и выдают активационное значение. Когда изображение вводится в ConvNet, каждый слой создает несколько активационных функций, которые передаются на следующий слой. Первый уровень обычно получает начальные характеристики, такие как горизонтальные или диагональные края. Результат из первого слоя передается на следующий слой, где находятся более сложные детали, такие как углы или более сложные края, как показано на рисунке 3.3. И чем глубже мы продвигаемся в сети, тем более сложные характеристики она может выявлять, такие как объекты, лица, жесты и так далее.

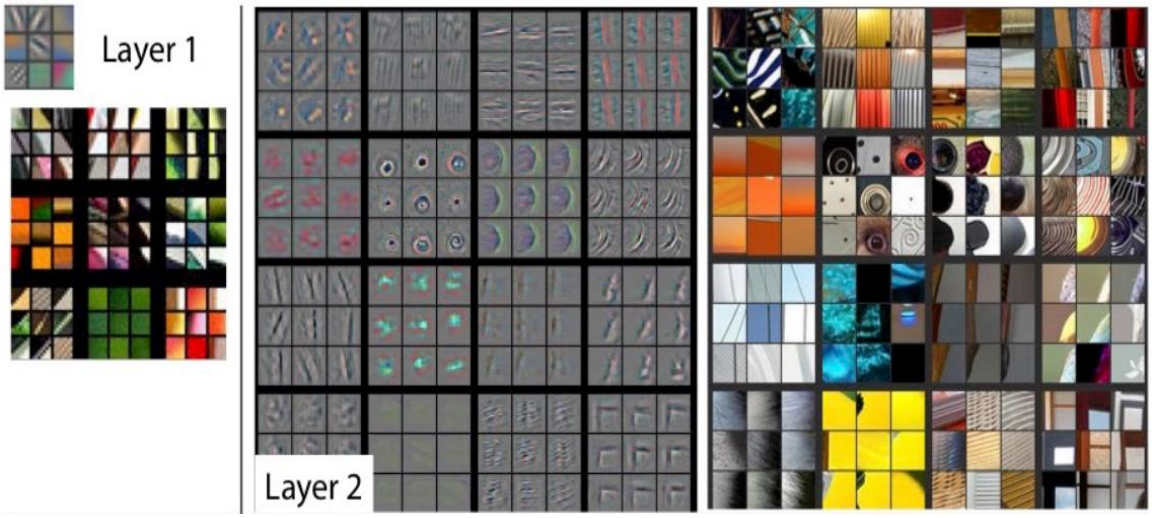


Рисунок 3.3 - Структура слоев и их содержание

На основе активационной карты последнего слоя слой классификации выдает набор оценок уверенности, то есть значения от 0 до 1. Эти значения указывают, насколько вероятно, что изображение принадлежит к определенному классу. Например, ConvNet, который идентифицирует части тела человека, и вывод последнего слоя в такой сети будет вероятностью того, что на изображении присутствует конкретная часть тела [136]. Рисунок 3.4 показывает определение класса изображения путем постепенного определения его специфических деталей через слои CNN.

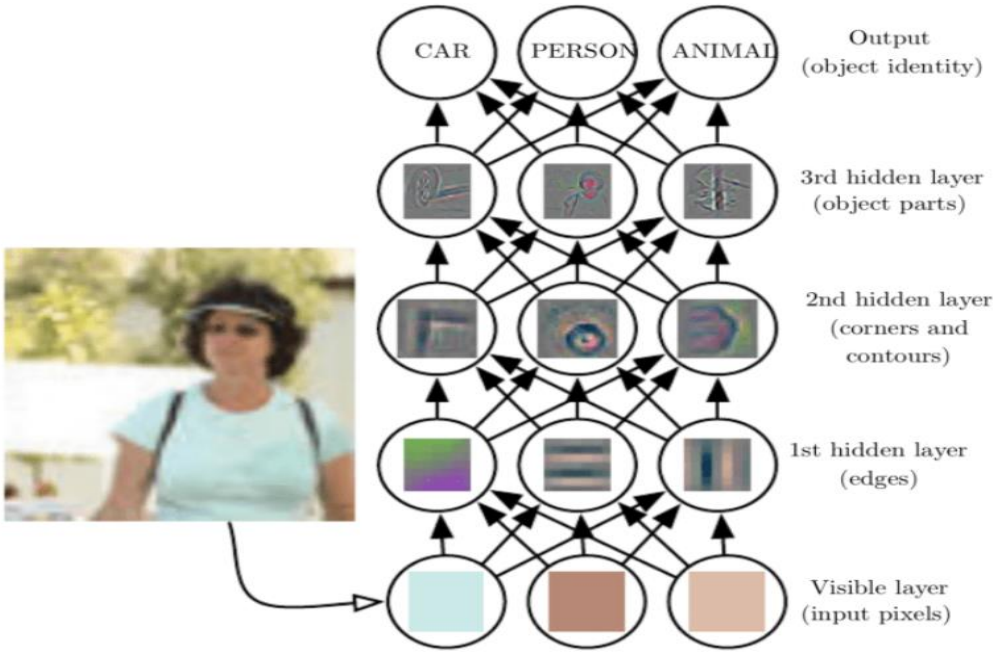


Рисунок 3.4 – Последовательность и параметры определения класса изображения с использованием сверточной нейронной сети (CNN)

Архитектура модели CNN для распознавания жестов рук предназначена для эффективного извлечения признаков из входных изображений и их классификации в один из семи классов жестов [118, с. 42]. Два сверточных слоя используют набор фильтров для извлечения функций из входных изображений. Уровень пакетной нормализации нормализует выходные данные сверточных слоев, чтобы улучшить стабильность модели во время обучения. Размер выходных данных сверточных слоев уменьшается за счет максимального слоя пула, что снижает вычислительную сложность модели. Слой исключения используется для уменьшения переобучения путем случайного исключения некоторых нейронов в плотных слоях во время обучения. Два полностью связанных плотных слоя используют функции, извлеченные из сверточных слоев, для классификации входных изображений по одному из семи классов жестов. Выходной слой имеет семь узлов, по одному для каждого класса жестов. Выходной уровень использует функцию активации softmax для перевода выходных значений в оценки вероятности для каждого класса жестов. Оптимизатор Адама используется для обновления весов модели во время обучения. Оптимизатор Адама адаптирует скорость обучения для каждого весового параметра, что приводит к более быстрой сходимости и повышению производительности [137,138]. Разница между ожидаемым результатом и фактическими метками изображений рассчитывается с использованием категориальной функции перекрестной энтропийной потери. Метрика категориальной точности используется для оценки производительности модели во время обучения и тестирования. Выходные векторы создаются путем горячего кодирования семи классов жестов. Горячее кодирование – это способ представления двоичных категориальных данных. В этом случае каждый

выходной вектор имеет семь значений, одно из которых равно единице, а остальные равны нулю, что представляет вероятность каждого класса жестов.

Архитектура модели CNN для распознавания жестов рук предназначена для эффективного извлечения функций из входных изображений и классификации их в один из семи классов жестов. Использование методов оптимизации, таких как оптимизатор Адама, функция потерь и метрика точности, а также использование горячего кодирования, позволяет эффективно обучать и оценивать модель. Необходимы дальнейшие исследования и разработки новых методов и архитектур для повышения производительности и способности к обобщению систем распознавания жестов рук на основе CNN.

3.3.1 Картины в роли объектов для обучения нейронных сетей

Монохромные изображения представлены в виде векторов чисел, где каждый пиксель представлен числом от 0 (белый) до 1 (черный). Такой способ упрощен и теряет информацию о смежности пикселей. Цветные изображения кодируются в формате RGB, где каждый пиксель представлен тройкой чисел (красный, зеленый, синий) из диапазона 0–255. Это позволяет сохранить информацию о цветах. Также можно преобразовать цветные изображения в вектора чисел для ввода в нейронную сеть. Такие методы цифровизации [105, с.40] могут быть простыми, но они все равно имеют свое применение, особенно в простых задачах.

Существует проблема, связанная с использованием полносвязных нейронных сетей для задачи распознавания изображений [91, с. 31]. В данном контексте объясняется структура классической полносвязной нейронной сети, включая входной и выходной слои, и обсуждаются сложности, связанные с большим числом связей (весов) между нейронами. Для небольших изображений, например, размером 3x3 пикселя, полносвязная нейронная сеть требует большого количества весов и связей между нейронами, что делает ее тяжеловесной и требующей много времени и данных для обучения [137]. Большая часть весов может оказаться неиспользуемой для конкретной задачи.

Рассмотрим альтернативный подход из биологии, раздел анатомии человека, где нейроны в мозге не обрабатывают информацию от всего изображения, а только от узких областей. Этот подход может позволить снизить количество связей и весов в нейронной сети, делая ее более эффективной для распознавания изображений. Далее необходимо применить архитектуру, аналогичную той, которая развилась в человеческом мозге для обработки изображений, к искусственным нейронным сетям для достижения более эффективных результатов в задачах распознавания изображений.

Специальные архитектуры нейронных сетей, которые отличаются от стандартной архитектуры [100, с. 38]. Изображения – это не числа, и не вектор чисел, это достаточно сложный объект для того, чтобы подать их в нейронную сеть. Упрощенный способ подать изображение в нейронную сеть заключается в его цифровизации, а именно в преобразовании в вектор чисел. Для простоты рассмотрим самый обычный случай, когда все изображения монохромны, то

есть каждый пиксель – это определенный оттенок серого (как показано на рисунке 3.6).

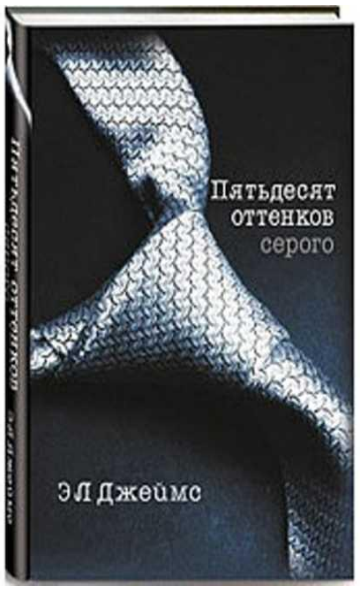


Рисунок 3.6 – Gray scale (оттенок серого)

Мы можем представить каждый пиксель картинки числами от 0 до 1. Причем, ноль будет соответствовать белому цвету, а единица черному, а все что между ними – это оттенок серого цвета (рисунок 3.7).

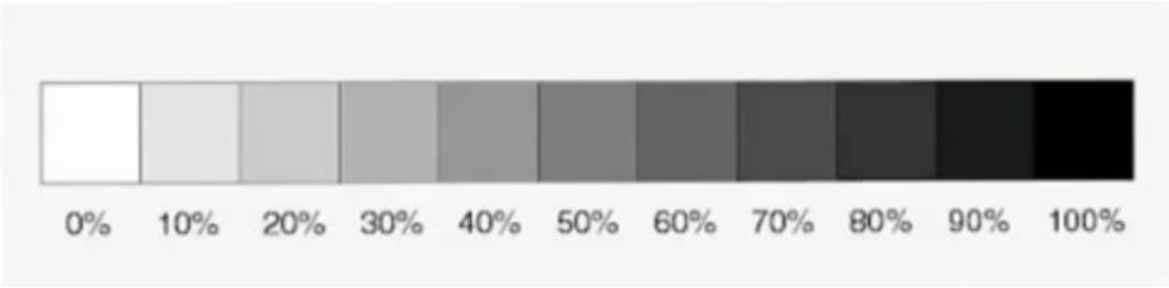


Рисунок 3.7 – Gray scale tones (черно-белые тон цветов)

Таким образом, черно-белое изображение преобразуется в таблицу, а точнее в матрицу чисел, который подается в качестве значений входного слоя нейронной сети (рисунок 3.8):

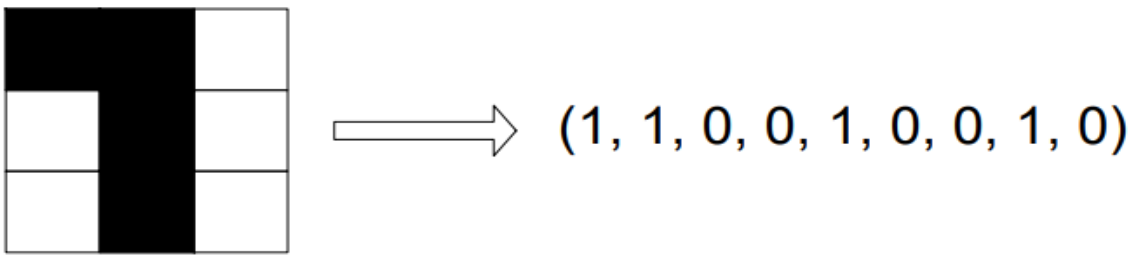


Рисунок 3.8 – Преобразование пикселей в вектора

То есть, если пиксель черный, это 1, а если белый, то 0. Таким образом, изображение превращается в длинный вектор. У такого способа цифровизации изображений есть и свои недостатки. Самый очевидный это то, что способ слишком простой. То есть, когда мы оцифровываем изображение, теряется часть информации, которая заложена в ней. Изображение, переходя в вектор теряет информацию о смежности пикселей. А именно, верхний левый пиксель преобразуется в первую единицу вектора. Пиксель, который стоит под ним, преобразуется не в соседнее число, а число, которое отстоит от первого на три позиции. Расстояние между этими пикселями в векторе равно размеру изображения. Если размер изображения был 100*100, расстояние между пикселями было бы соответственно равно 100. То есть превращая изображение в векторы чисел, мы теряем информацию о смежности пикселей. Но такой подход [138] имеет место быть, и для простых задач он может быть применяем.

Теперь рассмотрим более сложный случай, когда изображение цветное. Важный момент заключается в том, как закодировать с помощью чисел цвета. Для этого нужно использовать формат RGB (рисунок 3.9). Фактически каждый пиксель – это тройка целых чисел из интервала (0, 255). Числа соответствуют интенсивности красного, зеленого и синего цветов, которые наблюдаются в конкретном оттенке.

Named	Numeric	Color name	Hex rgb	Decimal
		black	#000000	0,0,0
		silver	#C0C0C0	192,192,192
		gray	#808080	128,128,128
		white	#FFFFFF	255,255,255
		maroon	#800000	128,0,0
		red	#FF0000	255,0,0
		purple	#800080	128,0,128
		fuchsia	#FF00FF	255,0,255
		green	#008000	0,128,0
		lime	#00FF00	0,255,0
		olive	#808000	128,128,0
		yellow	#FFFF00	255,255,0
		navy	#000080	0,0,128
		blue	#0000FF	0,0,255
		teal	#008080	0,128,128
		aqua	#00FFFF	0,255,255

Рисунок 3.9 – Цветовая гамма в формате RGB

Например, если выбрать полностью черный цвет, в нем нет ни красного, ни зеленого, ни других цветов. Ему соответствует тройка чисел 0, 0, 0. Белый цвет содержит в себе и зеленый, и красный, и синий. Поэтому белому цвету соответствует, по формату RGB, тройка максимальной интенсивности 255, 255,

255. Таким образом, цвета можно преобразовать в вектора чисел [137,138], только здесь используется не одно число, а три. Фактически в цветном изображении каждый пиксель – это тройка чисел от 0 до 255. Картина состоит из трех матриц (рисунок 3.10).

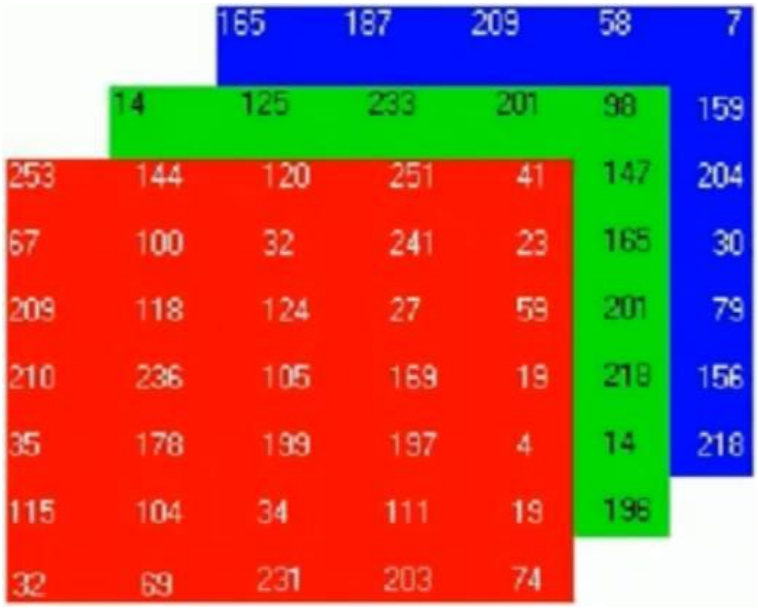


Рисунок 3.10 – Картина, состоящая из матриц

Таблица содержит информацию об интенсивности цвета. Если использовать этот простой способ, то все таблицы можно “вытянуть” в линию, в результате чего получится один большой вектор, который подается на вход в нейросеть. Будем использовать классические нейронные сети (рисунок 3.11).

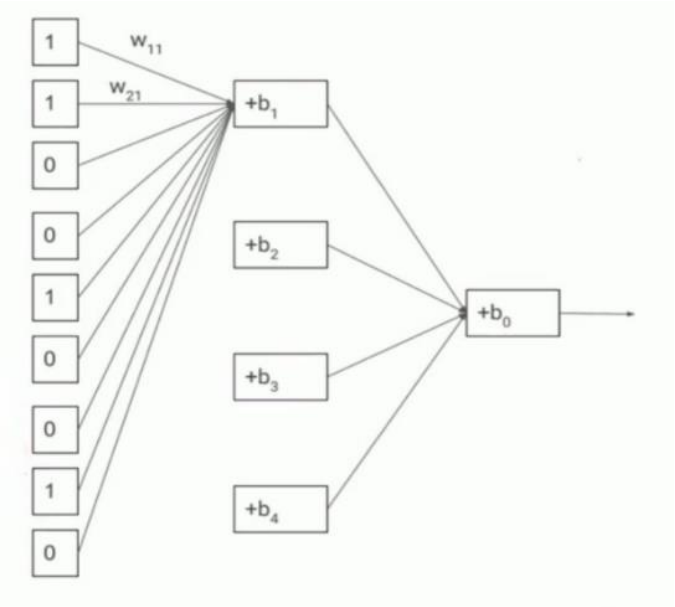


Рисунок 3.11 – Параметры в структуре нейронной сети – веса и смещения

Первый вертикальный ряд цифр – входной слой. То есть это изображения в виде векторов. Средний слой – сами нейронные сети. Третье – это выходной

слой нейронной сети. Все стрелки между нейронами имеют веса, которые нужно тренировать. Проблема в том, что даже для самого маленького изображения 3*3 нам понадобится размерность входного слоя равная к 9 пикселям, и более того, из каждого пикселя нужно проводить связь в каждый нейрон следующего слоя, так как слои полносвязные. То есть в данном случае будет 36 весов-связей, на первом слое нейронной сети будет 4 весов-смещения. Функция активации на данном рисунке не указана.

Таким образом, даже для маленькой картинки 3*3 в результате имеем большое количество весов. Это главный недостаток полносвязной архитектуры для задачи распознавания картинок. Все веса необходимо тренировать, но в конечном счете их все невозможно использовать, и неизвестно, будет ли полученная нейронная хорошо распознавать изображения [139]. Чтобы избежать этой проблемы необходимо экономить веса. Практика показала, что когда речь идет о распознавании изображений, даже если они представлены в виде векторов, так много весов иметь не обязательно. Каждому нейрону из первого слоя нейронной сети не обязательно знать информацию обо всей картинке. Так же устроен и человеческий глаз. Наша сетчатка состоит из клеточек, в одну клеточку приходит информация не обо всей картинке, а только от очень узкой области изображения. С нейронными сетями можно сделать точно так же.

3.3.2 Сверточная архитектура и ее матричные фильтры

Рассмотрим процесс распознавания изображений в биологическом человеке, исследуя структуру сетчатки глаза и первых слоев зрительных нервов. Нейроны в сетчатке и зрительном центре не имеют информации обо всем изображении, а работают в маленьких группах. Это приводит к идее сверхточной архитектуры, вдохновленной анатомией глаза, где все слои нейронной сети двумерны и связаны только с небольшими группами нейронов предыдущего слоя [95, с. 32].

Есть методы экономии весов в сверточных нейронных сетях. Количество связей в сети можно значительно уменьшить, используя параллельный перенос весов между различными областями изображений. Это приводит к сокращению количества обучаемых весов до минимума. Это представляется в виде матрицы весов, которая прикладывается к разным областям изображения. Веса используются для суммирования входных значений и вычисления выходных значений в нейронах следующего слоя. Этот подход обеспечивает экономию весов, что упрощает обучение и улучшает эффективность сверточных нейронных сетей.

Рассмотрим, какие процессы происходят в голове у человека, когда он распознает изображение. На рисунок 3.12 приведен срез сетчатки глаза и первых слоев зрительных нервов зрительного центра.

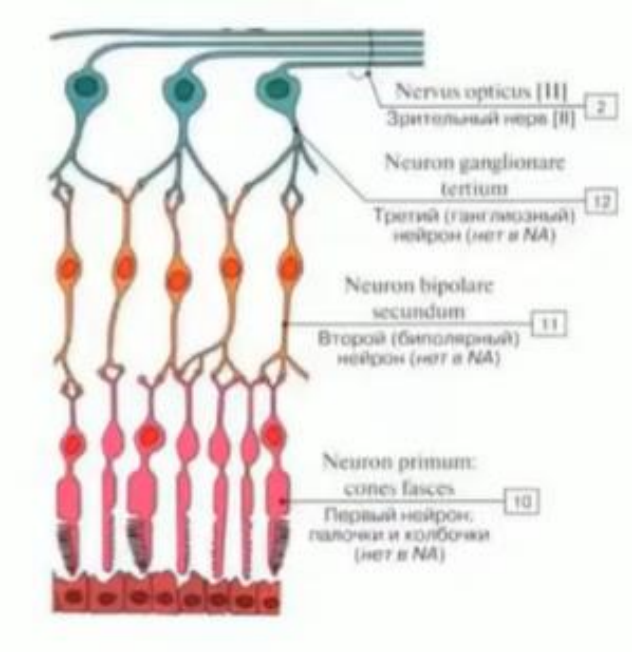


Рисунок 3.12 – Биологические нейроны зрительного нерва человека

Ни один из нейронов сетчатки зрительного центра не имеет информации обо всем изображении, которое расстилается перед глазами. В нейроны первого уровня приходит световой поток. Далее они группируются по парам или по три в маленькие группы. Нейроны следующего уровня тоже соединяются по группам, и дальше сигнал идет в следующий уровень головного мозга. То есть никакой полной связности в зрительном центре мозга нет. Потому что нейроны очередного слоя соединяются не со всеми нейронами предыдущего, а только с небольшой группой. Таким образом, возникает идея сверточной архитектуры [139, 140], на которой построены нейронные сети, распознающие изображения. Эта идея происходит из анатомии человеческого тела. Более того, в полносвязных слоях мы привыкли что каждый слой – это вертикальный столбец, но есть одномерный. А в сверточной архитектуре все слои нейронной сети должны быть двумерными. Потому что сама сетчатка глаза – это двумерная поверхность, у которой есть ширина и высота. Поэтому все слои сверточной архитектуры будут двумерными. Возникает полная аналогия между структурой сверточной нейронной сети и строением сетчатки человеческого глаза (рисунок 3.13).

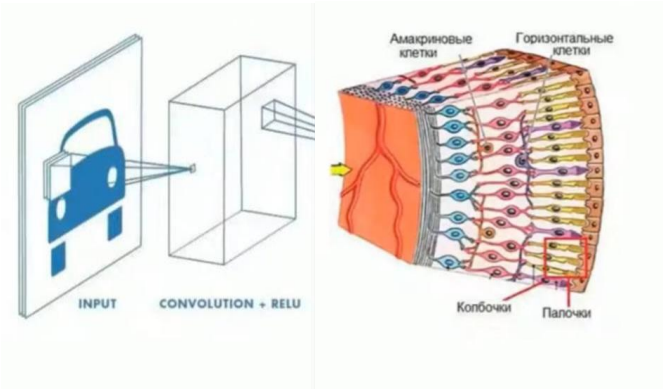


Рисунок 3.13 – Аналогия между искусственными и биологическими нейронами

Все слои сверточной нейронной сети представлены в виде матрицы. Каждый нейрон последующего слоя соединен только с нейронами определенной области предыдущего слоя. Размер этой области определяется при создании сверточных нейронных сетей. В нашем примере размеры области 2*2 (рисунок 3.14).

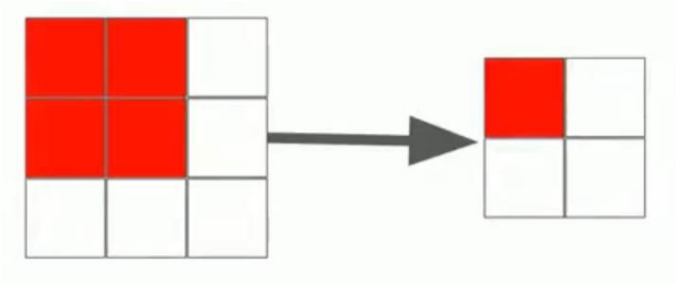


Рисунок 3.14 – Картина 2x2

Область из 4 нейронов будет отображаться как 1 нейрон следующего слоя. Нейроны во втором слое могут иметь области, перекрывающиеся (пересекающиеся) с предыдущим слоем (рисунок 3.15).

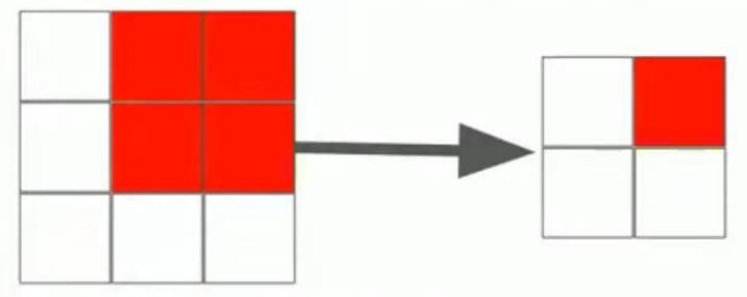


Рисунок 3.15 – Картина 2x2

Именно так выглядит сверточный слой (рисунок 3.16), который совершенно не похож на полносвязный.

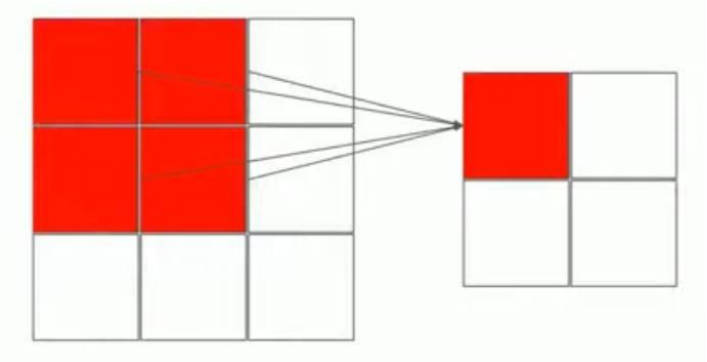


Рисунок 3.16 – Сверточный слой

Связи нейрона следующего слоя исходят только из определенной области предыдущего слоя. Нейроны в сверточном слое – это как слепые мудрецы, ощупывающие слона (рисунок 3.17).

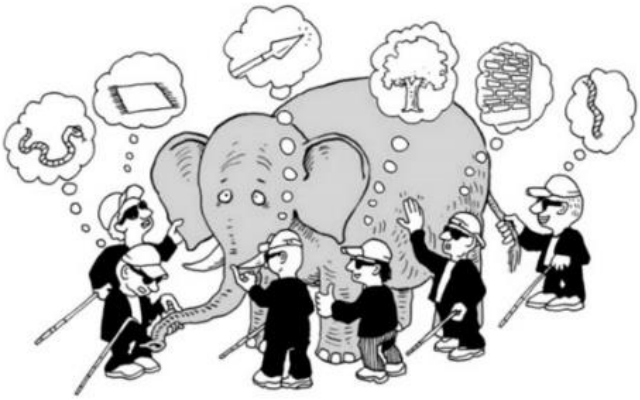


Рисунок 3.17 – Изображение, представленное в контексте из концепции, слепые мудрецы

Рисунок 3.17 описывает философскую концепцию, что каждый мудрец, ощупывая разные части тела слона думает, что это что-то похожее, например на ковер, стену или змею. Фактически, слепые мудрецы – это не выделенные нейроны. Каждый из этих нейронов имеет дело лишь с ограниченной областью изображения. Поэтому они могут иногда ошибаться. Проблема слепых мудрецов: нет “мудреца 2-го уровня”, который бы смог обработать информацию от каждого мудреца, после чего смог бы сказать, что этот объект – слон. В нейронных сетях мы можем внедрить еще один слой, который анализирует информацию, полученную с предыдущего слоя. Если размер области составляет 2*2, то каждый нейрон второго слоя будет иметь 4 веса связей. Поскольку в этом слое всего 4 нейрона (см. рисунок 3.18), то общее количество весов связей между сверточными слоями будет составлять 4 * 4 = 16. Что касается веса-смещения, это свободные члены, которые жили внутри каждого искусственного нейрона в полносвязной архитектуре. А в сверточной архитектуре веса-смещения нет.

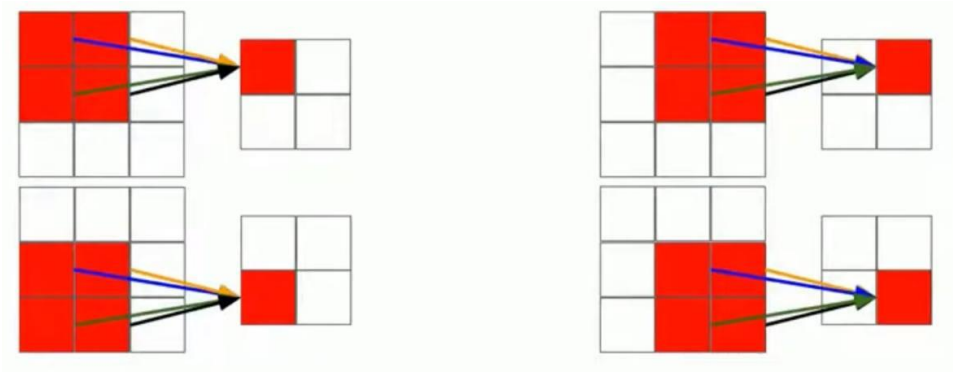


Рисунок 3.18 – Матричные слои изображения

На рисунке 3.18 мы видим 4 пучка связей, ведущих в разные нейроны второго слоя. Здесь должно быть 16 разных весов для тренировки. Можно еще сэкономить: считать, что, соответствующие веса на картинках равны друг другу. То есть здесь будет только 4 различных веса для тренировки (идентичные веса показаны одним цветом). Все это очень упрощает работу. Это позволяет веса записывать в виде матрицы, как слои нейронной сети (рисунок 3.19).

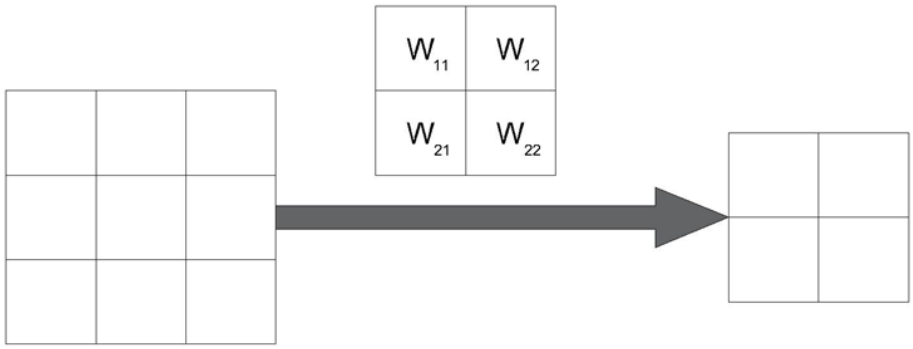


Рисунок 3.19 – Веса из слоев нейронной сети

По факту матрица весов сама прикладывается к различным областям изображения, и результат будет записываться в следующий слой. Вспоминаем что происходит с весами в нейросети. Они доумножаются на входные значения, и результат суммируется.

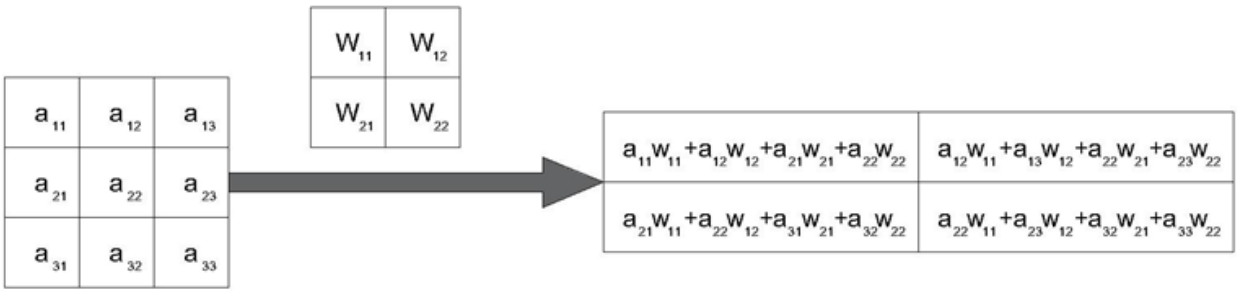


Рисунок 3.20 – Умножение весов в слоях нейронных сетей

У нас есть матрица весов, она прикладывается к первой области изображения 2*2, соответствующие числа в изображениях перемножаются. Затем все числа суммируются и возникает число, которое нужно записать в ячейку следующего слоя (рисунок 3.20). Точно так же, сдвигая матрицу цветов на соседнюю область, мы перемножаем их с весами, которые лежат в этих ячейках, затем получаем новое выражение для записи в другой новой ячейке следующего слоя [139, 140, с.58].

Здесь возникает известная теория изображении – идея фильтра. Матричные фильтры появились задолго до возникновения продвинутых нейронных сетей. Сверточная архитектура нейронных сетей использует именно матричные фильтры. Данные фильтры применялись и до эпохи нейронных сетей в задачах обработки изображений:

– нахождение границ предметов на изображении;

– размытие изображений;

– увеличение контрастности, и т. д.

Фильтр – это матрица с числами. Он прикладывается к каждой области изображения и результат операции свертки записывается в итоговое изображение. Операция свертки полностью повторяет преобразование данных в нейронной сети. Например, у нас есть черно-белое изображение, которое представлено в виде матрицы (рисунок 3.21), а также есть фильтр (рисунок 3.22).

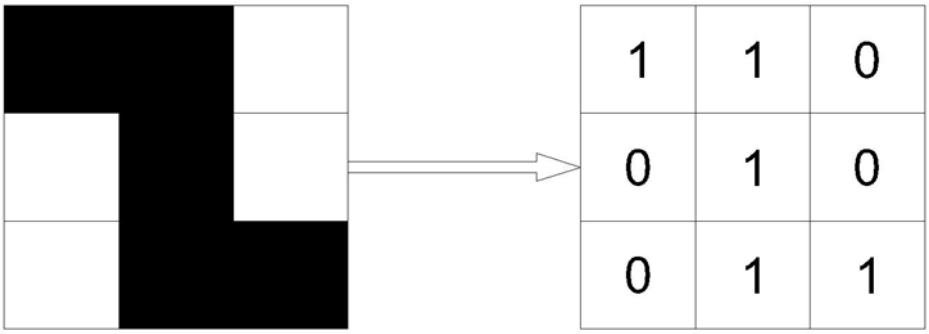


Рисунок 3.21 – монохромные изображения, представленные в цифрах 0 и 1

Фильтр:

0	1
0	1

Рисунок 3.22 – Фильтр

В этом фильтре “прогоняется” изображение. Сначала оно прикладывается к первой области (рисунок 3.23).

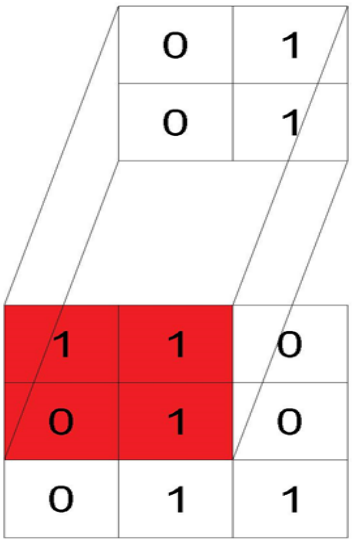


Рисунок 3.23 – Работа фильтра

Происходит точно такая же операция, как и в сверточной архитектуре. Соответствующие ячейки перемножаются, затем все суммируется. После всех операций выводим итоговое изображение:

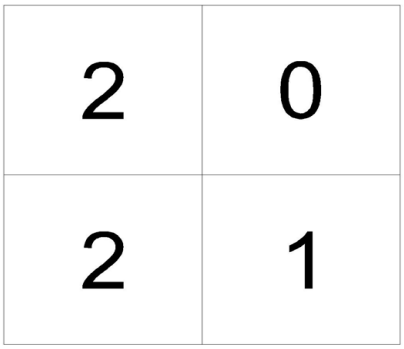


Рисунок 3.24 – фильтр итогового изображения

Когда мы прогоняем фильтр по изображению, фильтр выделяет области изображения, которые наиболее тесно соответствуют маске фильтра. Из данного примера мы видим, что свертка уменьшает размер изображения. У нас есть изображение n*n, фильтр m*m. После наложения фильтра размер матрицы изображения будет:

$$(n - m + 1) * (n - m + 1). \quad (3.1)$$

3.3.3 Параметры свертки сетей и композиция фильтров

В данном параграфе обсуждаются операции свертки в сверточных нейронных сетях (CNN). Рассматривается возможность тонкой настройки операции свертки с помощью параметров, таких как величина шага и добавление пустой рамки вокруг изображения (padding). Также обсуждается, как эти параметры влияют на области изображения, к которым применяется фильтр. Также подчеркивается важность добавления рамки для более эффективной обработки пикселей на границе изображения [142]. Этот параграф предлагает понимание тонкой настройки операции свертки и ее влияния на обработку изображений в сверточных нейронных сетях. Обсуждаются параметры, такие как величина шага и добавление пустой рамки вокруг изображения (padding), которые могут влиять на области изображения, к которым применяется фильтр. Приводятся примеры, как изменение этих параметров может влиять на операцию свертки. Также обсуждается польза добавления пустой рамки для улучшения обработки пикселей на границе изображения. Рассматривается концепция композиции фильтров в сверточных нейронных сетях (CNN) [142]. Основное внимание уделяется последовательному применению нескольких фильтров к изображению, где каждый следующий фильтр использует результат предыдущего. Этот метод позволяет распознавать сложные зависимости между пикселями в изображении, даже если фильтры сами по себе просты и примитивны. Приводится пример композиции двух примитивных фильтров для нахождения определенного фрагмента в изображении. Главный вывод заключается в том, что первые фильтры в композиции распознают базовые закономерности, а последующие фильтры способны выявить более сложные и абстрактные зависимости в изображении.

Фильтр прикладывается к областям изображения. Этот процесс можно настроить с помощью параметров, отвечающих за:

- величину шага (как сильно смещается фильтр по сравнению с предыдущей итерацией). По умолчанию шаг составляет 1 единицу как по вертикали, так и по горизонтали;
- добавление пустой границы вокруг изображения. Это нужно для того, чтобы фильтр приложился к крайним точкам изображения.

Величину шага задают два параметра: *stride_x, stride_y* – на сколько пикселей сдвигается фильтр по горизонтали и вертикали после очередной итерации. По умолчанию *stride_x* = *stride_y* = 1, то есть фильтр в ряду изображения каждый раз сдвигается на 1 пиксель, а при окончании ряда фильтр смещается на 1 пиксель вниз. Но в некоторых задачах можно эти параметры задать отличными от 1, что зависит от специфики самой задачи.

Например, вот так перемещается фильтр 3*3 по изображению 6*6 (рисунок 3.25), если *stride_x* = 1, *stride_y* = 2:

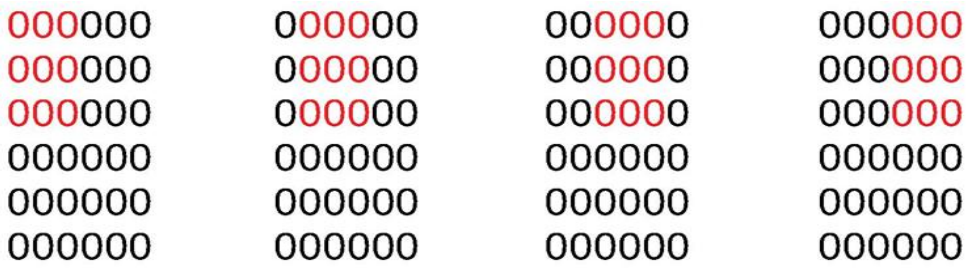


Рисунок 3.25 – Фильтр изображения 6x6

Второй способ, как можно настраивать операцию свертки – это применение Паддинга (Padding). Паддинг предполагает добавление пустой рамки вокруг изображения. Не выполнение этого действия может привести к проблемам, таким как редкое использование краевых пикселей фильтром и уменьшение размерности результата по сравнению с исходным изображением. Например, если padding = 2, то это означает, что вокруг изображения добавляется рамка из белых пикселей шириной 2. Например:

Изображение:	После padding = 1:
110	00000
010	01100
111	00100
	01110
	00000

Композиция фильтров [143]. Далее, после применения одного фильтра можно применить второй фильтр. Это позволяет распознавать более сложные изображения, имея в своем распоряжении лишь слабые объекты. Например, научимся распознавать области изображения, в которых присутствует фрагмент (рисунок 3.26):

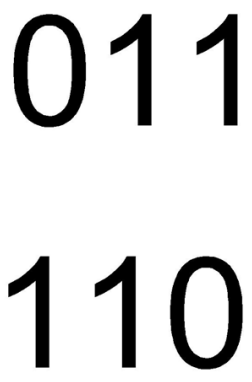


Рисунок 3.26 – Фрагмент на области изображения

Данное изображение необходимо выбрать в качестве фильтра. Но мы используем более простые фильтры. Таким образом получается, что этот фрагмент (рисунок 3.26) можно детектировать с помощью двух простых фильтров. Сначала применяем фильтр (рисунок 3.27):

Рисунок 3.27 – Применение фильтра

А затем еще один (рисунок 3.28):

Рисунок 3.28 – Подходящий фрагмент

Эти фильтры очень примитивны, но тем не менее их композиция позволяет найти весьма нетривиальный фрагмент в нашем изображении.

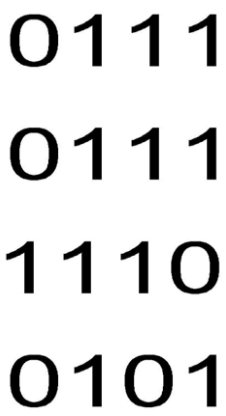


Рисунок 3.29 – Фрагмент в изображении

Итак, подаем изображение (рисунок 3.29), и далее с помощью фильтров (рисунок 3.27, рисунок 3.28) получаем после первого фильтра (рисунок 3.30):

Рисунок 3.30 – Первый фильтр

После второго фильтра (рисунок 3.30):

Рисунок 3.31 – Второй фильтр

Далее, посмотрим, чему соответствуют числа (рисунок 3.31), которые мы получили после второго фильтра. Максимальные значения соответствуют областям в исходном изображении, которые содержали исходное изображение (рисунок 3.27). Действительно, мы получили (рисунок 3.32):

Рисунок 3.32 – Соответствующие числа, выделенные красным цветом

Мы видим и понимаем, что максимальные числа, выделенные красным, соответствуют областям, которые мы искали в исходном изображении (рисунок

3.30). Первая красная четверка в первом ряду говорит о том, что в самом исходном изображении в верхнем правом углу был нужный нам фрагмент. Далее, проверим данные фрагменты (рисунок 3.33):

0111 0111 1110 0101	0111 0111 1110 0101	0111 0111 1110 0101

Рисунок 3.33 – Фрагменты данных

Таким образом, после композиции двух фильтров (рисунок 3.29, рисунок 3.30), мы нашли все области, в которых имеется интересующий нас фрагмент (рисунок 3.31). Здесь возникает полная аналогия со «слепыми мудрецами» (рисунок 3.17). Когда «все мудрецы пощупали слона», это первый фильтр. То есть мы знаем, что их информация противоречива. Таким образом, возвращаясь к философской концепции, когда один «сверх-мудрец» опросил «слепых мудрецов и понял, что это был слон», это второй фильтр. Это означает, что если к изображению последовательно применяются несколько фильтров, то первые фильтры выявляют самые фундаментальные характеристики. Более сложные характеристики распознаются только последними фильтрами. Можно заключить, что сверточные нейронные сети гораздо лучше справляются с задачей распознавания изображений.

3.3.4 Распознавания изображений и фильтры в нейронных сетях

Сверточные нейронные сети применяют фильтры для обработки изображений, однако коэффициенты фильтра определяются самой сверточной нейронной сетью. Сверточная нейросеть, прежде всего, это тренируемые объекты. В ней есть фильтры, но числа, которые стоят внутри ячеек фильтра, определяет сама нейронная сеть в процессе тренировки. В целом, искусственный интеллект (ИИ) сам подбирает “слепых для ощупывания слона” (рисунок 3.17). То есть, «для ощупывания слона нужен разный жизненный опыт, иметь некоторые знания для нужных поверхностей». ИИ тренирует «собственных мудрецов для распознавания слона».

Теперь рассмотрим, как данный процесс описывается математически [146, 147]. На рисунке 3.34 показана упрощенная общая архитектура СНС.

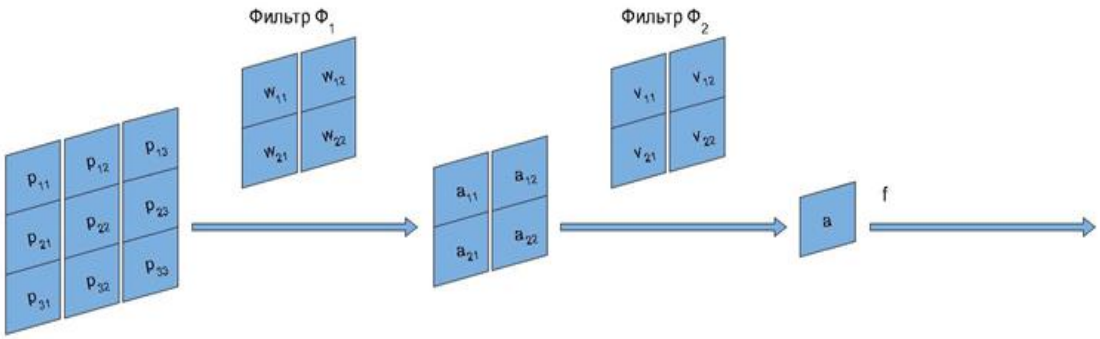


Рисунок 3.34 – Архитектура сверточной нейронной сети

На вход будут подаваться изображения 3*3. Пиксели изображения будут обозначены как p_{ij}. Вначале к этому изображению применяется фильтр Φ_1. Получится результат с пикселями a_{ij}. Действительно, когда на изображение 3*3 применяется фильтр 2*2, то полученный результат, изображение должно быть 2*2. После этого к полученному изображению 2*2 используем фильтр Φ_2. Когда размер изображения и фильтра совпадают, в результате получается одно число.

Кроме того, как и в любой нейронной сети в СНС применяются функции активации. В данном случае f - функция активации. Когда в сверточной архитектуре функция активации написана после определенного слоя, это означает что функция f применяется ко всем ячейкам слоя.

Далее рассмотрим какие выражения получаются. После первого фильтра видим выражения a_{ij}. Они являются результатом свертки, можно их записать следующим образом:

$$a_{11} = p_{11}w_{11} + p_{12}w_{12} + p_{21}w_{21} + p_{22}w_{22}, \quad (3.2)$$

$$a_{12} = p_{12}w_{11} + p_{13}w_{12} + p_{23}w_{21} + p_{23}w_{22}, \quad (3.3)$$

$$a_{13} = p_{21}w_{11} + p_{22}w_{12} + p_{21}w_{21} + p_{32}w_{22}, \quad (3.4)$$

$$a_{14} = p_{22}w_{11} + p_{23}w_{12} + p_{32}w_{21} + p_{33}w_{22}, \quad (3.5)$$

Зная выражения для a_{ij}, можно выписать выражения для a. Для этого к полученному изображению после первого фильтра Φ_1, нужно применить функцию активации f, затем наложить фильтр Φ_2:

$$a = f(a_{11})v_{11} + f(a_{12})v_{12} + f(a_{21})v_{21} + f(a_{22})v_{22}\ , \quad (3.6)$$

Таким образом, в соответствии с определением операции свертки получается результат a. К полученному выражению a нужно применить еще одну функцию активации f, и это будет результатом работы нейронной сети. Тогда СНС вычисляет выражение:

$$F_{NN}(P) = f(a) = f(f(a_{11})v_{11} + f(a_{12})v_{12} + f(a_{21})v_{21} + f(a_{22})v_{22}) =$$

$$= f(f(p_{11}w_{11} + p_{12}w_{12} + p_{21}w_{21} + p_{22}w_{22})v_{11} + \\ +f(p_{12}w_{11} + p_{13}w_{12} + p_{23}w_{21} + p_{23}w_{22})v_{12} + \\ +f(p_{21}w_{11} + p_{22}w_{12} + p_{21}w_{21} + p_{32}w_{22})v_{21} + \\ +f(p_{22}w_{11} + p_{23}w_{12} + p_{32}w_{21} + p_{33}w_{22})v_{22}), \quad (3.7)$$

Где $F_{NN}(P)$ - функция нейронной сети. А далее, по тренировочной выборке мы выписываем функцию потерь и минимизируем ее относительность весов нейронной сети.

Рассмотрим, как тренируются нейронные сети для простой задачи распознавания изображения. Все начинается с тренировочной выборки. В качестве примера возьмем три изображения (таблица 3.2).

Таблица 3.2 – Изображения в бинарном формате

Изображение	Y
P1:	1
011	
001	
P2:	2
110	
011	
P3:	0
101	
010	

Предположим, у нас есть обучающий набор данных, состоящий из изображений, а для каждого из объектов в этом наборе данных у нас есть целевой признак. Мы определили целевой признак Y как "количество фрагментов с образом 11". Поскольку признак Y является количественным, перед нами стоит задача регрессии.

В задаче регрессии функция потерь представляет собой сумму квадратов разности между выходом нейронной сети $F_{NN}(P)$ и истинным значением признака Y. Таким образом, нам нужно минимизировать эту функцию потерь:

$$L_{(w)} = (F_{NN}(P1) - 1)^2 + (F_{NN}(P2) - 2)^2 + (F_{NN}(P3) - 0)^2 \quad (3.8)$$

Теперь нужно выбрать подходящую архитектуру нейронной сети (рисунок 3.35).

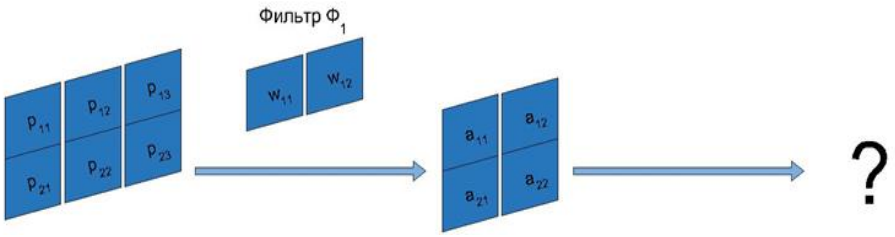


Рисунок 3.35 – Выбор подходящей архитектуры

В нашей тренировочной выборке все изображения имели размеры. У них было два ряда пикселей, по 3 столбца в каждом ряду. То есть на вход нейронной сети будет подаваться изображение, состоящее из шести пикселей. Это изображение проводим по фильтру Φ_1, в результате получаем изображение 2*2. В любой СНС сверточные слои рано или поздно заменяются на (обычные) полносвязные. Поскольку у нас задача простая, то мы добавим первый полносвязный слой сразу после сверточного слоя (рисунок 3.36).

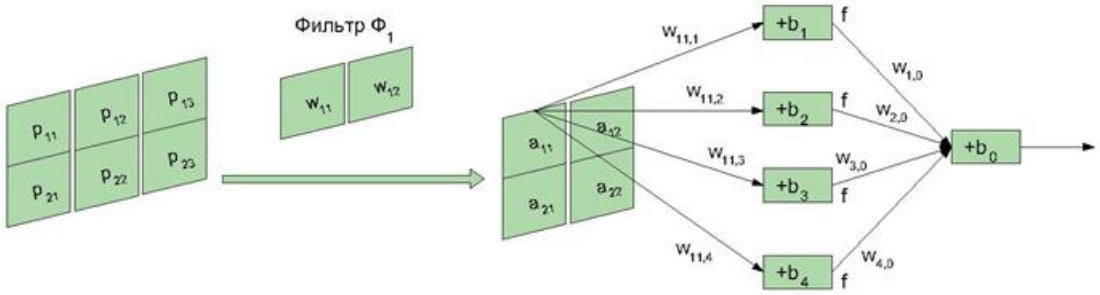


Рисунок 3.36 – Добавление полносвязного слоя

После сверточного слоя у нас получается 4 выражения a_{ij}, каждый пиксель соединяем с каждым из четырех пикселей полносвязного слоя. Этот слой будет последним, потому что дальше стоит результирующии нейрон, который выдает окончательно решения для решения задачи регрессии. Здесь мы будем использовать функции активации $Relu(x)$.

После получения результата, в выражении функции потерь получатся вхождения весов w_{ij} которые принадлежат фильтру Φ_1, а также весов, которые соответствуют полносвязным слоям. Мы применили стандартную процедуру градиентного спуска, находим оптимальное значение весов. Но на данном примере оптимальные веса легко подобрать без использования градиентного спуска (рисунок 3.37).

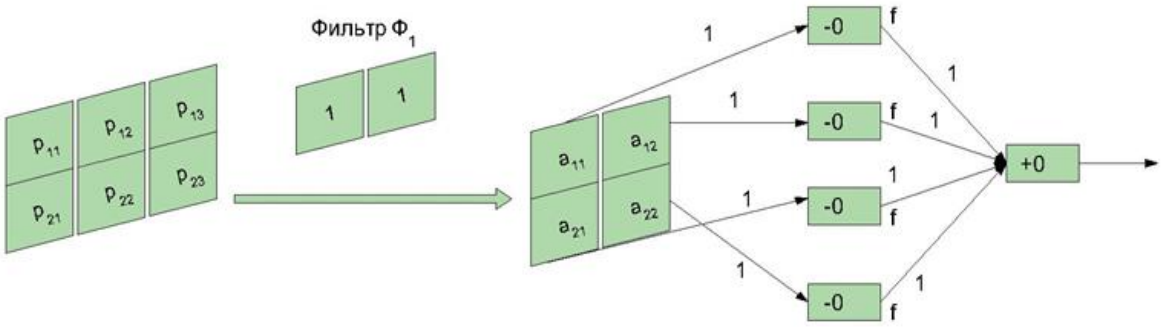


Рисунок 3.37 – Подбор оптимальных весов

Получились оптимальные веса для Φ_1 полносвязного слоя. Сама нейронная сеть нашла “подходящего слепого мудреца”, который имеет именно такие числа. Веса-связи, равные 0, не показаны.

После того, как стали известны оптимальные веса, снова можно выписать функцию, которую вычисляет нейронная сеть:

$$F_{NN}(P) = f(a_{11} - 1) + f(a_{12} - 1) + f(a_{21} - 1) + f(a_{22} - 1) = \\ = f(p_{11} + p_{12} - 1) + f(p_{12} + p_{13} - 1) + f(p_{21} + p_{22} - 1) + \\ + f(p_{22} + p_{23} - 1) \quad (3.9)$$

С помощью этой формулы можно проверить правильность выбранных весов на конкретных примерах. Поэтому, для изображения (0, 0, 0, 0, 0, 0) получаем:

$$F_{NN}(P) = f(-1) + f(-1) + f(-1) + f(-1) = \\ = 0 + 0 + 0 + 0 = 0 \quad (3.10)$$

Для функции $Relu(x)$ выполнено $Relu(-1) = 0$.

Теперь рассмотрим другой пример, изображение (1, 1, 1, 1, 1, 1). Получаем:

$$F_{NN}(P) = f(2 - 1) + f(2 - 1) + f(2 - 1) + f(2 - 1) = \\ = 1 + 1 + 1 + 1 = 4 \quad (3.11)$$

Это верные ответы, поскольку мы загадали признак $Y =$ "количество вхождений элемента 11". Таким образом, сверточная нейросеть была натренирована правильно.

Из такого простого примера следуют очень далеко идущие выводы. Конкретно, мы рассмотрели общую структуру сверточных нейронных сетей. Как мы установили, вначале идут фильтры (сверточные слои), которые приводят к уменьшению размеров каждого последующего слоя, а в конце сети следуют несколько полносвязных слоев (рисунок 3.38).

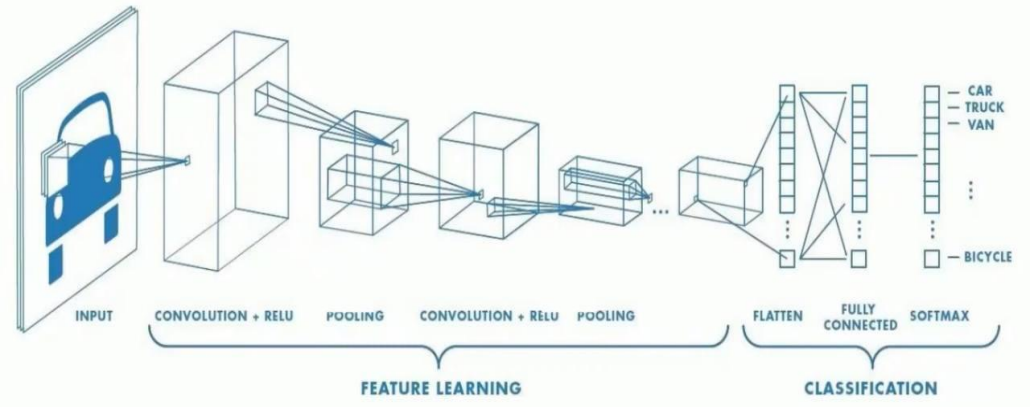


Рисунок 3.38 – Архитектура сверточных нейронных сетей с полносвязными слоями

На рисунке 3.38 изображение, заданное на входе, проходит этапы обучения (feature learning) и классификацию (classification) изображения на выходе.

3.3.5 Метод обратного распространения ошибок

Одним из важных этапов обучения нейронных сетей является применение метода (алгоритма) обратного распространения ошибки, или backpropagation. Метод обратного распространения ошибки представляет собой конкретную реализацию градиентного спуска в контексте многослойных сетей прямого распространения [149]. Основная концепция этого метода заключается в эффективном вычислении частных производных (partial derivative) функции сети F (w, x) по всем компонентам настраиваемого вектора весов w для данного входного вектора x. Этот алгоритм направлен на поиск градиента ошибки по всем обучаемым параметрам нашей модели [147 - 150].

Мы будем рассматривать обычные полносвязанные сети для задачи классификации. Однако многие принципы применимы также и для других типов нейронных сетей, а также в общем для произвольных дифференцируемых вычислительных графов. При выполнении вычислений в одном полносвязанном слое, в первую очередь, предположим, что мы работаем с векторами в виде строк, а не столбцов:

$$x = [x_1 \;\; x_2], \tag{3.12}$$

$$h = [h_1 \;\; h_2 \;\; h_3], \tag{3.13}$$

$$b = [b_1 \;\; b_2 \;\; b_3], \tag{3.14}$$

Таким образом, выходной вектор h вычисляется с использованием нелинейной функции активации:

$$h = F(xW + b), \tag{3.15}$$

Чтобы рассчитать, к примеру, первый компонент (элемент) вектора h_1 необходимо выполнить следующие шаги:

x_1 и x_2 на w_{11}и w_{21} из матрицы:

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}, \tag{3.16}$$

Получим,

$$h_1 = F(x_1 w_{11} + x_2 w_{21} + b_1), \tag{3.17}$$

Таким же образом вычисляются и h_2 и h_3.

Разделим вычисления одного слоя на два этапа: линейный и нелинейный. Предположим, что после линейной части мы получаем вектор t. Затем нелинейная функция, примененная к каждому элементу, преобразует его в конечный вектор h тогда, $t = xW + b$, $h = F(t)$. Разложим это на составляющие для одного элемента:

$$t_1 = x_1 w_{11} + x_2 w_{21} + b_1, \tag{3.18}$$

$$h_1 = F(t_1), \tag{3.19}$$

В нашем случае важно осознавать, что полносвязанные слои нейронной сети представляют собой всего лишь частные случаи вычислительных графов. Рассмотрим пример визуализации графа для рассматриваемого

полносвязанного слоя. Это позднее поможет нам понять, каким образом мы перемещаемся по графу в процессе обратного распространения ошибки.

Граф вычислений для полносвязанного слоя выглядит следующим образом. Узлы с данными x, t и h, и для вычисления узла t нам нужны узлы с параметрами w и b.

Для оптимизации параметров нейронной сети при использовании оптимизационного алгоритма нам требуется вектор градиента ошибки по всем обучаемым параметрам нашей модели:

$$\frac{\partial E}{\partial \Omega} = \left\{\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \frac{\partial E}{\partial w_2}, \dots\right\}, \tag{3.20}$$

Количество элементов в градиенте соответствует количеству обучаемых параметров, и для удобства мы можем разделить градиент на группы, соответствующие различным объединениям параметров. Например, если у нас есть матрица весов W, мы можем рассматривать её как отдельный объект со своим набором параметров, и, следовательно, необходимо иметь часть градиента того же размера для неё. Обозначим его $\frac{\partial E}{\partial W}$, то есть как частная производная ошибки E по матрице W для нашего текущего слоя. Аналогично, для вектора b –соответствующая ему часть градиента $\frac{\partial E}{\partial b}$. Для каждого слоя и для каждого объекта обучаем ими параметрами: для b_1 будет $\frac{\partial E}{\partial b_1}$, для W_2 будет $\frac{\partial E}{\partial W_2}$, для b_2 будет $\frac{\partial E}{\partial b_2}$. Таким образом, это скаляр, вектор, матрица или тензор.

Важно осознавать, что в большинстве случаев алгоритм обучения, такой как градиентный спуск (Stochastic Gradient Descent), и алгоритм вычисления градиента (BACKPROP), могут быть полностью независимыми. Алгоритм обучения стремится получить градиент, и ему не важно, каким образом он был вычислен. В процессе вычисления градиента нам не важно, как он будет использоваться во время обучения. Поэтому, сфокусируемся на вычислении градиента. Предположим, что этот слой является частью определенной модели, которую мы обучаем и нам надо найти $\frac{\partial E}{\partial W}$ и $\frac{\partial E}{\partial b}$. Для этого слоя используем цепное правило (Chain Rule) из математического анализа и двигаемся в обратном направлении. Пусть нам уже дано $\frac{\partial E}{\partial h}$ в численном виде. То есть, при заданном входе в сеть при заданных весах мы достали $\frac{\partial E}{\partial h}$. Размерность у него такая же, как и у вектора h, то есть это вектор строка из трёх элементов:

$$\left[\frac{\partial E}{\partial h_1} \quad \frac{\partial E}{\partial h_2} \quad \frac{\partial E}{\partial h_3}\right], \tag{3.21}$$

тогда мы можем вычислить $\frac{\partial E}{\partial t}$. Это тоже будет вектор из трёх элементов, как и вектор t. Далее, понимая $\frac{\partial E}{\partial t}$, мы можем вычислить $\frac{\partial E}{\partial w}$ и $\frac{\partial E}{\partial b}$, те самые элементы, которые нам нужно найти. Размеры такие же как исходных матриц и

векторов. $\frac{\partial E}{\partial W}$ – это матрица два на три (2x3), $\frac{\partial E}{\partial b}$ – это вектор из трёх элементов. Однако, кроме того, мы можем вычислить градиент по входу $\frac{\partial E}{\partial x}$.

Это необходимо, чтобы мы могли передать градиент на предыдущий слой и повторить те же манипуляции для его параметров. Вектор x – это вход для нашего слоя, h – выход для предыдущего слоя. В данном примере, $\frac{\partial E}{\partial x}$ – это векторы из двух элементов.

Теперь все это необходимо выводить по очереди:

Начнём с $\frac{\partial E}{\partial t}$, при условии, что $\frac{\partial E}{\partial h}$ нам дана данная производная. Самый корректный способ все это вывести – отслеживать конкретные компоненты вектора градиента. Рассмотрим $\frac{\partial E}{\partial t_1}$. Это число частной производной функции ошибки E по переменной t_1. Воспользуемся тем, что ошибка зависит от h_1., а h_1зависит от t_1. Тогда, при использовании цепного правила:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1}, \tag{3.22}$$

Всё это сводится к числам. В действительности было бы целесообразно полностью выписать производную, используя правило дифференцирования сложной функции для нескольких переменных и учесть, какие промежуточные переменные связывают между собой t_1 с ошибкой E. Но даже на схеме мы видим, что эта связь проходит только через h_1, а h_2 и h_3 никак от t_1 не зависят. Следовательно, здесь дополнительные пояснения не требуются. Теперь рассмотрим, что у нас получилось:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1} = \frac{\partial E}{\partial h_1}, \tag{3.23}$$

Что значит $\frac{\partial h_1}{\partial t_1}$. Как видно $\frac{\partial h_1}{\partial t_1}$ связаны через скалярную функцию $h_1 = F(t_1)$. Это означает что $\frac{\partial h_1}{\partial t_1}$ это просто производная функции в точке t_1:

$$\frac{\partial E}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot \frac{\partial h_1}{\partial t_1} = \frac{\partial E}{\partial h_1} \cdot F'(t_1), \tag{3.24}$$

впоследствии аналогичным образом,

$$\frac{\partial E}{\partial t_2} = \frac{\partial E}{\partial h_2} \cdot F'(t_2) \text{ и } \frac{\partial E}{\partial t_3} = \frac{\partial E}{\partial h_3} \cdot F'(t_3), \tag{3.25}$$

Этого достаточно для расчета вектора ∂E по ∂t. Однако данное выражение представим в более компактном видеВ данном случае поэлементно умножаем два вектора:

$$\frac{\partial E}{\partial h} = \left[\frac{\partial E}{\partial h_1} \quad \frac{\partial E}{\partial h_2} \quad \frac{\partial E}{\partial h_3}\right], \tag{3.26}$$

на вектор, состоящий из проиводных функции на разных точках:

$$F'(t_1) = [F(t_1)' \quad F(t_2) \quad 'F(t_3)')], \tag{3.27}$$

Это равносильно, если бы применили функцию $F'(t)$ к вектору t по элементно. Тогда финальный вектор имеет вид:

$$\frac{\partial E}{\partial t} = \frac{\partial E}{\partial h} \odot F'(t) \tag{3.28}$$

Этот вектор также имеет три элемента. Такое поэлементное произведение часто называют произведением Адамара. Таким образом, мы получили выражение для вычисления вектора $\frac{\partial E}{\partial t}$.

Теперь рассмотрим градиент по матрице W. Аналогично следим за отдельными элементами матрицы и применяем цепное правило дифференцирования. Начнём с $\frac{\partial E}{\partial w_{11}}$. Вес w_{11} связывает вход x_1 и выход t_1. То есть, в t_2 и t_3 он никакой вклад не даёт. Тогда:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{11}}, \tag{3.29}$$

Чему равна $\frac{\partial t_1}{\partial w_{11}}$. Это дифференциал t_1:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{11}} = \frac{\partial E}{\partial t_1} \cdot x_1, \tag{3.30}$$

Рассмотрим производную для другого элемента матрицы. Начнем с изменения первого индекса, то есть спустимся вниз:

$$\frac{\partial E}{\partial w_{21}} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial w_{21}} = \frac{\partial E}{\partial t_1} \cdot x_2, \tag{3.31}$$

Если мы перемещаемся вдоль столбцов и увеличиваем второй индекс, то этот вес теперь соединяет вход и выход t_2. Поэтому, такая производное будет равна:

$$\frac{\partial E}{\partial w_{12}} = \frac{\partial E}{\partial t_2} \cdot \frac{\partial t_2}{\partial w_{12}} = \frac{\partial E}{\partial t_2} \cdot x_1, \tag{3.32}$$

Точно так же мы можем поступить со всеми остальными элементами градиента $\frac{\partial E}{\partial w}$. Надо заметить, что есть корреляция индекса веса w индексом x и t. Первый индекс соответствует индексу x а второе с индексом t. Это заметно напоминает процесс умножения матриц. Действительно, мы можем компактно представить финальную матрицу $\frac{\partial E}{\partial w}$ через матричное произведение. Пусть

$$x^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \tag{3.33}$$

$$\frac{\partial E}{\partial t} = \begin{bmatrix} \frac{\partial E}{\partial t_1} & \frac{\partial E}{\partial t_2} & \frac{\partial E}{\partial t_3} \end{bmatrix}, \tag{3.34}$$

тогда

$$\frac{\partial E}{\partial W} = x^T \cdot \frac{\partial E}{\partial t}, \tag{3.35}$$

Теперь у нас есть градиент по матрице весов. Далее, рассмотрим вектор $\frac{\partial E}{\partial b}$. Вклад в первый элемент дает только один из них в t_1:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial b_1} = \frac{\partial E}{\partial t_1} \cdot 1. \tag{3.36}$$

Аналогично для двух других элементов:

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial t_2} \text{ и } \frac{\partial E}{\partial b_3} = \frac{\partial E}{\partial t_3}, \tag{3.37}$$

тогда

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial t} \tag{3.38}$$

Оба вектора состоят из трёх элементов.

Теперь самый важный момент: это градиент по входу. Здесь процесс вычисления будет сложнее. Нам нужно получить $\frac{\partial E}{\partial x}$ вектор из двух элементов зная $\frac{\partial E}{\partial t}$. Начнём с $\frac{\partial E}{\partial x_1}$, так как x_1 уже даёт вклад во все элементы вектора t.

Далее, необходимо применить правило дифференцирования сложной функции нескольких переменных. Это приведет к появлению суммы по промежуточным переменным.

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial t_1} \cdot \frac{\partial t_1}{\partial x_1} + \frac{\partial E}{\partial t_2} \cdot \frac{\partial t_2}{\partial x_1} + \frac{\partial E}{\partial t_3} \cdot \frac{\partial t_3}{\partial x_1}, \tag{3.39}$$

Заметим, что появилась сумма по всем элементам, в которую внесен вклад $\frac{\partial E}{\partial x_1}$. Если воспользоваться теми же самыми соотношениями, мы получим соответствующие веса W:

$$\frac{\partial t_1}{\partial x_1} = w_{11}, \frac{\partial t_2}{\partial x_1} = w_{12}, \frac{\partial t_3}{\partial x_1} = w_{13}, \tag{3.40}$$

Именно эти веса связывают x_1с t_1, t_2и t_3:

$$\frac{\partial E}{\partial x_1} = \frac{\partial E}{\partial t_1} \cdot w_{11} + \frac{\partial E}{\partial t_2} \cdot w_{12} + \frac{\partial E}{\partial t_3} \cdot w_{13}, \tag{3.41}$$

Аналогично для x_2там уже будут другие веса w:

$$\frac{\partial E}{\partial x_2} = \frac{\partial E}{\partial t_1} \cdot w_{21} + \frac{\partial E}{\partial t_2} \cdot w_{22} + \frac{\partial E}{\partial t_3} \cdot w_{23}, \tag{3.42}$$

Также обратим внимание на скрытое матричное умножение здесь и запишем все в компактной форме:

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial t} \cdot w^T. \tag{3.43}$$

Теперь у нас есть понимание того, как находить градиенты по параметрам полносвязного слоя, и, более того, мы знаем, как передать градиент к предыдущему слою, чтобы проделать аналогичные шаги с его параметрами и так далее. Для этого мы вычислили $\frac{\partial E}{\partial x}$. Однако, не ясно что мы предположили, что мы знаем $\frac{\partial E}{\partial h}$. Если мы знаем $\frac{\partial E}{\partial h}$ для последнего слоя, то мы можем постепенно вычислить все градиенты для всех предшествующих ему слоёв. Теперь как нам получить $\frac{\partial E}{\partial h}$, с которого нам надо начать. Поскольку это последний слой, его вывод прямо связан с конечной ошибкой E. Нужно вычислить соответствующую производную. В последнем слое мы не используем функцию активации, поэтому наша задача сводится к нахождению $\frac{\partial E}{\partial t}$, и мы будем исходить из этого, двигаясь в обратном направлении. Для этого надо понять как связан последний вектор t и ошибка E. После выполнения определенных операций мы должны получить одномерный массив (скаляр) E, ошибку, которая всегда представляет собой единственное число, независимо от каких-либо факторов. Для начала получим окончательные предсказания нашей модели:

$$Z = Softmax(t) = S(t) = \left\{\frac{e^{t_i}}{\sum_j e^{t_i}}\right\}, \tag{3.44}$$

Применили экспоненту к каждому элементу вектора, переводя их монотонно в диапазон от нуля до плюс бесконечности. Затем разделили на сумму, чтобы обеспечить единичную сумму финальных вероятностей. Теперь, имея вероятности, предоставленные нейронной сетью в качестве ответа, мы можем рассчитать ошибку предсказания. Для этого нам также понадобится известный правильный ответ, который мы обозначим как y. Напомним, что y – это вектор, состоящий из нулей и одной единицы в позиции, соответствующей индексу правильного класса (в данном случае это 0, 1 или 2). Это истинное распределение, которое мы стремимся достичь при соответствующем входе в нейросеть. Тогда ошибку можно вычислить следующим образом:

$$E = CrossEntropy(z, y) = -\sum_i y_i \ln z_i, \tag{3.45}$$

y – правильный ответ, z – выход из Softmax.

Таким образом, мы имеем сочетание $Softmax$ и $CrossEntropy$ и данном конкретном случае мы можем подставить одно в другое, упростить, и процесс дифференцирования будет упрощен.

$$E = CrossEntrophy(S(t), y) = -\sum_i y_i \ln \frac{e^{t_i}}{\sum_j e^{t_j}} = -\sum_i y_i \left(t_i - \ln \sum_j e^{t_j}\right) =$$
$$-\sum_i y_i t_i + \sum_i y_i \ln \sum_j e^{t_j} = -\sum_i y_i t_i + \ln \sum_j e^{t_j}, \tag{3.46}$$

Получили простую зависимость E от t:

$$E = CrossEntropy(S(t), y) = -\sum_i y_i t_i + \ln \sum_j e^{t_j}, \tag{3.47}$$

Можем вычислить нужный вектор $\frac{\partial E}{\partial t}$. Распишем один его элемент:

$$\frac{\partial E}{\partial t_k} = -y_k + \frac{1}{\sum_j e^{t_k}} \cdot e^{t_k}, \tag{3.48}$$

Если внимательно посмотрим, то это наш элемент $Softmax$:

$$\frac{\partial E}{\partial t_k} = -y_k + \frac{1}{\sum_j e^{t_k}} \cdot e^{t_k} = S(t)_k - y_k, \tag{3.49}$$

А теперь запишем для вектора:

$$\frac{\partial E}{\partial t} = S(t) - y = Z - y, \tag{3.50}$$

Функция активации. Будем использовать одну из самых простых и популярных функций $ReLU$:

$$F(t) = ReLU(t) = \max(o, t), \tag{3.51}$$

$$F'(t) = \begin{cases} 1, t \geq 0 \\ 0, t < 0 \end{cases}, \tag{3.52}$$

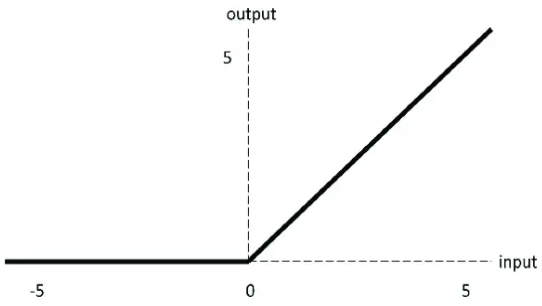


Рисунок 3.39 – Функция активации ReLU

Теперь у нас есть все фрагменты информации, которые мы можем объединить. Допустим, у нас есть конкретная нейронная сеть с двумя полносвязанными слоями. Если слоев больше, процесс выполняется аналогично. Создадим вычислительный граф для такой нейросети – граф прямого распространения. Вход в сеть представляет собой x, он преобразуется в t_1. Для этого нужны матрица W_1 и вектор b_1:

$$t_1 = xW_1 + b_1, \tag{3.53}$$

Индекс здесь уже означает номер слоя, а не номер элемента как раньше. t_1 преобразуется в h_1, выход первого слоя, которой идёт сразу на вход во второй слой и преобразуется в t_2 путём применения матрицы W_2 и вектора смещений b_2:

$$h_1 = F(t_1), \tag{3.54}$$

$$t_2 = h_1W_2 + b_2 \tag{3.55}$$

В последнем слое функция активации не используется. Сразу получаем вероятности Z через $Softmax$. Затем считаем ошибку, используя $CrossEntropy$:

$$Z = S(t_2), \tag{3.56}$$

$$E = CE(z, y). \tag{3.57}$$

После получения ошибки идет расчет в обратном направлении по графу, попутно вычисляются градиенты по соответствующим параметрам. Это и есть алгоритм обратного распространения ошибки.

Мы знаем, как получить $\frac{\partial E}{\partial t_2}$, t_2 – это последний вектор t в самом последнем слое. Зная $\frac{\partial E}{\partial w_2}$, мы можем получить $\frac{\partial E}{\partial b_2}$, два параметра второго слоя. Также мы можем получить $\frac{\partial E}{\partial h_1}$ – градиент по выходу из первого слоя, затем $\frac{\partial E}{\partial t_1}$, из которого уже можем получить $\frac{\partial E}{\partial w_1}$ и $\frac{\partial E}{\partial b_1}$.

$\frac{\partial E}{\partial x}$ вычислять не имеет смысла, так как он дальше не передается. Теперь выпишем все, что мы вывели до этого:

$$\frac{\partial E}{\partial w_2} = S(t_2) - y = Z - y \tag{3.58}$$

$$\frac{\partial E}{\partial w_2} = h_1^T \cdot \frac{\partial E}{\partial t_2} \tag{3.59}$$

Следующее действие – это вычисление производной для вектора для второго слоя:

$$\frac{\partial E}{\partial b_2} = \frac{\partial E}{\partial t_2} \tag{3.60}$$

Далее вычисляем производную, умножив на веса для первого слоя:

$$\frac{\partial E}{\partial h_1} = \frac{\partial E}{\partial t_2} \cdot W_2^T \tag{3.61}$$

Далее вычисляем производную с произведением Адамара для производной функции вектора *t* первого слоя:

$$\frac{\partial E}{\partial h_1} = \frac{\partial E}{\partial h_1} \odot F'(t_1) \tag{3.62}$$

Затем вычисляем *x* транспонированную на производную для второго слоя:

$$\frac{\partial E}{\partial w_2} = x^T \cdot \frac{\partial E}{\partial t_2} \tag{3.63}$$

Наконец, завершаем последнее действие производными для второго слоя:

$$\frac{\partial E}{\partial b_1} = \frac{\partial E}{\partial t_1} \tag{3.64}$$

Таким образом мы провели математические расчеты для вычисления градиента с помощью уравнений частных производных и сложных дифференциальных уравнений (Chain rule) для подсчета матрицы и векторов. Это является сложным и очень объёмным процессом работы алгоритма обратного распространения. В последующих главах описывается, как он технически реализуется в Python с помощью библиотек машинного обучения (сверточных и рекуррентных нейронных сетях).

Далее визуализируем все в графиках, ниже приведен рисунок функции потерь (loss function):

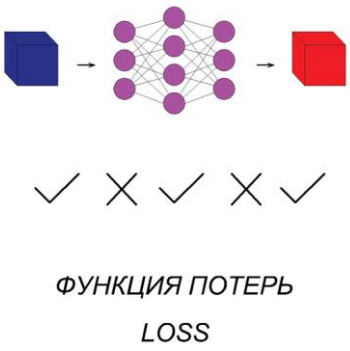


Рисунок 3.39 – Функция потерь в структуре нейросетей

Следующий график показывает нахождение функции ошибки:

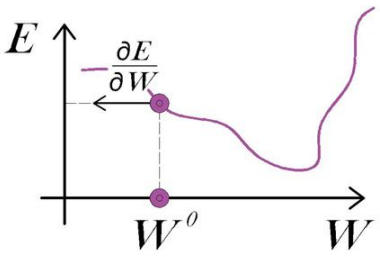


Рисунок 3.40 – Функция ошибки в градиенте

E – функция ошибки (кросс-энтропия), w – веса.

Затем мы вычисляем градиентный спуск по следующей формуле 3.65.

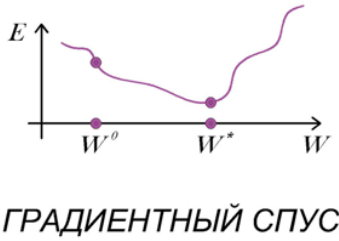


Рисунок 3.41 – градиентный спуск

$$W^{t+1} = W^t - \alpha * \frac{\partial E}{\partial W} \cdot (W^t), \tag{3.65}$$

где W – веса, α – темп обучения (learning rate), $\frac{\partial E}{\partial W}$ – производная ошибки, t – вектор.

Ниже показаны записи математических расчетов, формулы уравнений, изученные с книг линейной алгебры и аналитической геометрии, математического анализа и статистики, математической логики и теории алгоритмов и применяемые в нейронных сетях.

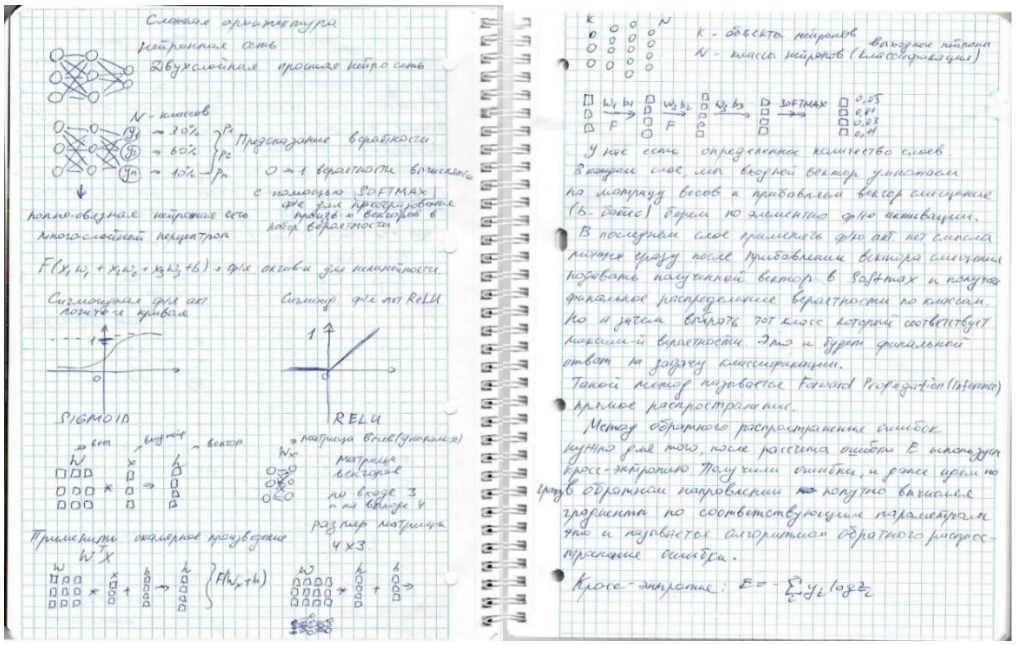


Рисунок 3.42 – Расчеты сложной архитектуры в слоях нейронных сетях

В завершении визуализировали все проделанные математические расчёты [146 - 149] в каждом слое нейронных сетей. Проведенные теоретические и аналитические исследования далее будут апробированы экспериментально. Также будет разработано программное обеспечение для реализации распознавания.

3.4 Выводы по разделу, классификация жестов и алгоритмы распознавания

В третьем разделе описаны применения различных алгоритмов распознавания жестов. Выполнена задача классификации на основе машинного обучения, где применяется машина опорных векторов, рекуррентные нейронные сети, сверточные нейронные сети, а также глубокое обучение. Применена архитектура нейронных сетей для распознавания изображений, а именно жестов рук. Освящена важная часть структуры нейронной сети, в которой применяются изображения в роли объектов для обучения, матричные фильтры, параметры свертки и композиция фильтров. Математически описана и доказана важность применения алгоритма обучения метода обратного распространения ошибок для вычисления градиента и необходимость применения функции активаций для минимизации функции потерь.

4 ЭКСПЕРИМЕНТЫ И МЕТОДЫ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ. РАСПОЗНАВАНИЕ КАЗАХСКОГО ЖЕСТОВОГО АЛФАВИТА

В этой главе мы собрали все необходимые данные для обучения нейронных сетей казахского жестового алфавита. Для проведения экспериментов мы применили методы обучения моделей нейронных сетей. Далее последовательно описываются этапы экспериментов, тестов и реализация программного обеспечения для распознавания казахских букв жестового алфавита в реальном времени.

4.1 Сбор данных (датасеты) для обучения

Для английского и русского жестовых языков доступны базы изображений жестов в хорошем качестве, чего нельзя сказать о казахском языке. Поэтому мы собрали свои данные для обучения. Этот набор данных состоит из 42 классов, каждый класс представляет собой букву казахского алфавита. И для каждого такого класса есть в среднем около 300-400 изображений. В общей сложности набор наших собранных данных состоит из более чем 15 тысяч изображений. Сами изображения представляют собой черно-белые снимки руки с разрешением 1280x720p, отображающие определенный жест, снятый с большого числа углов.

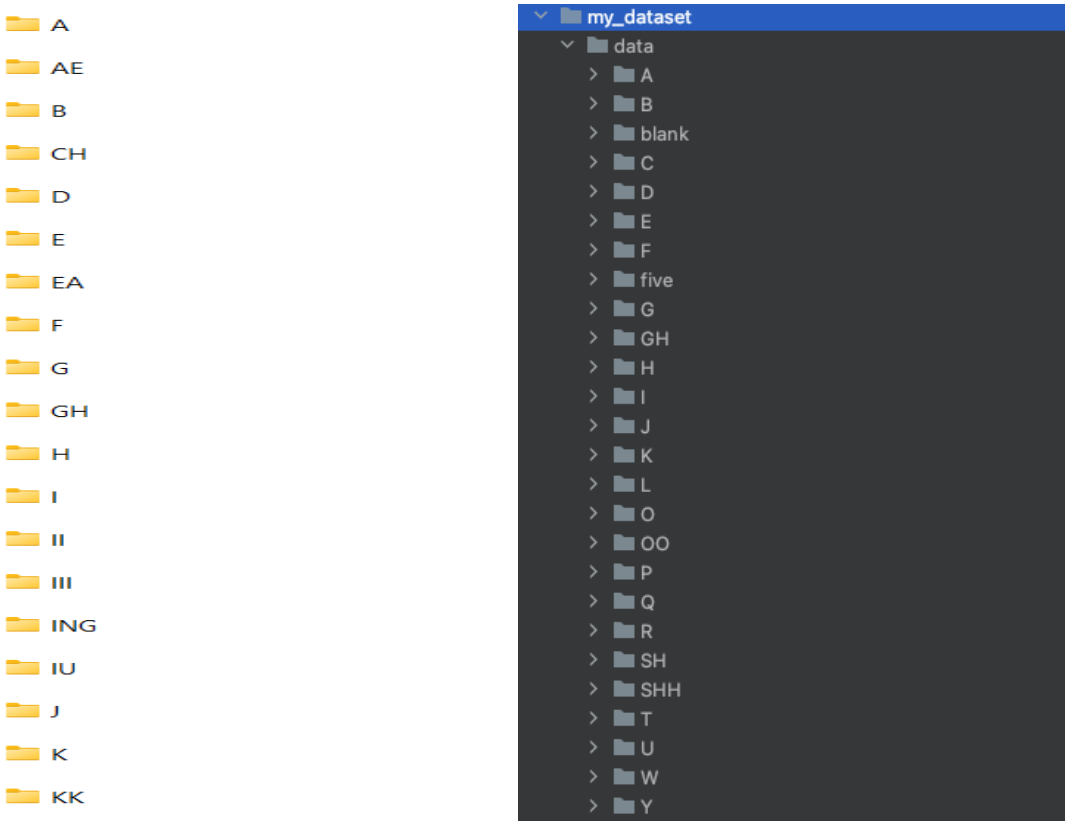


Рисунок 4.1 – Набор данных для обучения нейронных сетей

Для всех букв собраны наборы данных из изображений рук, представленных для каждого жеста. Для каждого жеста сделаны снимки при разных условиях освещения и расстояний от видеокамеры. После сбора мы отсеяли изображения низкого качества и привели все оставшиеся изображения к одному виду и размеру. После того как мы собрали данные, следующим шагом была предварительная обработка и очистка данных для удаления любых ошибок, несоответствий или ненужной информации. Это такие задачи, как нормализация данных, разработка функций и увеличение данных. Наборы данных показаны на рисунке 4.1.

На рисунке 4.2 показаны примеры изображений и соответствующих символов из этого набора данных.

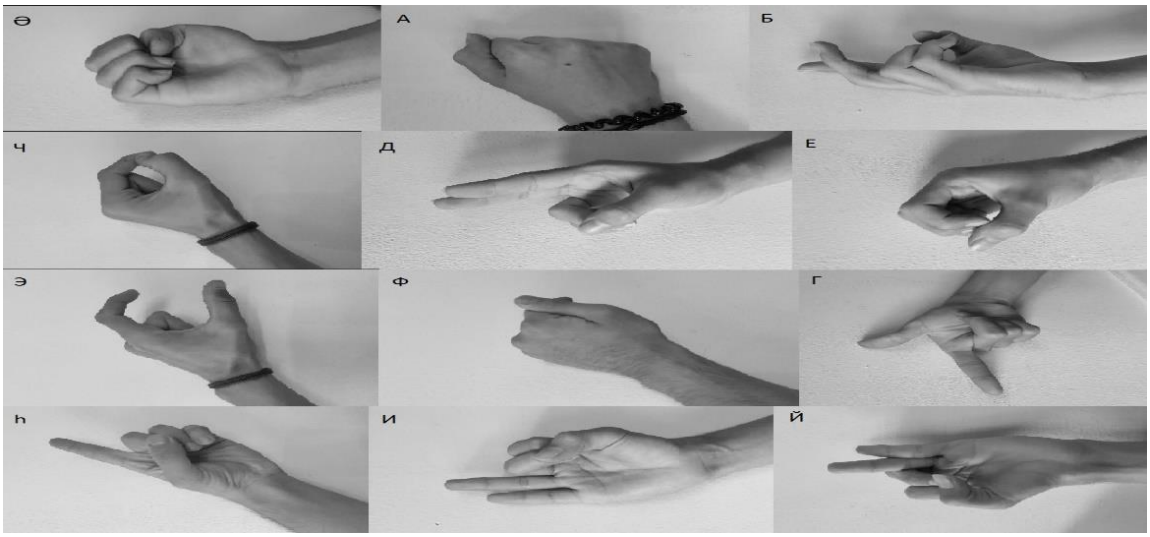


Рисунок 4.2 – Изображения некоторых жестов казахского жестового языка из используемого набора данных

Рисунок 4.3 – Изображение жеста буквы «Ә»

На рисунке 4.3 показано количество изображений для казахской буквы «Ә». Всего было собрано около 300 изображений для данной буквы.

Для обучения нейронных сетей для распознавания жестов рук мы разметили и подготовили данные для обучения.

Первым шагом в подготовке данных применили сегментацию, которая включала изоляцию области жестов рук от фона с использованием алгоритма определения порога. Алгоритм вычислял абсолютную разницу между фоном и текущим кадром, а затем определял пороговое значение полученного изображения для получения переднего плана. Затем алгоритм идентифицировал контуры на изображении порогового значения и выбрал максимальный контур, представляющий область жеста руки.

После сегментации мы преобразовали полученные изображения в оттенки серого. Изображения в оттенках серого имеют только один канал, что упрощает обработку изображений сверточных нейронных сетей. Кроме того, изображения в оттенках серого имеют более высокий контраст, что облегчает CNN выявление закономерностей на изображениях. Изменение размера изображений до стандартного размера также имело решающее значение для эффективного сравнения в процессе обучения. Выбор размера изображения зависит от доступных вычислительных ресурсов и типа используемых данных.

Маркировка набора данных правильной информацией о классе была необходима для того, чтобы нейросеть научилась ассоциировать определенные жесты рук с соответствующими метками. Маркировка может оказаться трудоемкой и сложной задачей, но она имеет решающее значение для разработки точной и надежной системы распознавания жестов рук. Сбор, нормализация и маркировка наборов данных для обучения может занять значительное количество времени и усилий, но это важно для разработки системы, которая сможет точно распознавать жесты рук. Мы исключили из нашего набора данных изображения низкого качества, которые были искажены из-за неблагоприятных условий (рисунок 4.4). В результате сохранились только изображения высокого качества, подобные изображенному на рисунке 4.3.

Рисунок 4.4 – Наборы данных низкого качества

Модель обучения с полученными наборами данных.

Разработка надежной системы распознавания жестов рук требует важного этапа обучения модели CNN с нашим набором данных казахского языка жестов. Чтобы обеспечить разнообразие и точность, мы собрали изображения разных людей с разными размерами рук, оттенками кожи и условиями освещения. Это не только создало более разнообразный набор данных, но и позволило модели точно распознавать жесты рук людей с разными физическими характеристиками. Чтобы предотвратить переобучение и увеличить разнообразие набора данных, мы использовали различные методы увеличения изображений, такие как случайное вращение, сдвиг, переворачивание и масштабирование. Этот метод дополнения привел к созданию более крупного и разнообразного набора данных, что улучшило обобщение модели на новые и ранее неизвестные данные.

Следующим этапом было подготовить данные для обучения. Был отобран набор данных из 2100 изображений для 42 классов жестов, по 50 изображений для каждого класса. Для обучения и валидации моделей сверточных нейронных сетей этого количества изображений достаточно. Оно позволит избежать промежуточного этапа работы по аугментации набора данных.

Модели машинного обучения с учителем обычно обучаются на подмножестве всего набора данных, а затем выполняют его на тестовом (другом подмножестве). Набор данных случайным образом разделяется на обучающие, валидационные и тестовые наборы данных, где 80% данных использовались для обучения модели (1680 изображений), 10% использовались для валидации модели (210 изображений) и 10% использовались для проверки точности модели (210 изображений). Для работы нейронных сетей требуется определенное количество данных, на основе которых данная нейронная сеть, фактически, будет учиться понимать информацию (обучение на примерах с правильными ответами), которую ей нужно определить.

Будущие работы будут направлены на дополнение и расширение набора данных собственных изображений, а также увеличения количества данных для обучения с целью улучшения работы модели.

4.2 Методы обучения моделей глубоких нейронных сетей

Глубокое обучение – это подраздел машинного обучения, который фокусируется на создании и обучении искусственных нейронных сетей, имеющих несколько слоев (глубокие сети) для решения сложных задач в различных областях, таких как компьютерное зрение, обработка естественного языка, рекомендации и другие [150]. Основные характеристики глубокого обучения включают:

Многослойные нейронные сети – глубокие модели обычно состоят из множества слоев, включая входной, скрытые и выходные слои. Эти слои позволяют модели извлекать иерархические и абстрактные признаки из входных данных.

Обучение на больших объемах данных – глубокие нейронные сети часто требуют больших объемов данных для обучения. Это позволяет моделям обучаться на разнообразных примерах и достигать высокой производительности.

Сложные архитектуры – существует множество архитектур глубокого обучения, включая сверточные нейронные сети для обработки изображений, рекуррентные нейронные сети для работы с последовательными данными и генеративные модели, такие как генеративно-состязательные сети [150, 151].

Глубокое обучение проявило себя как эффективный метод решения сложных задач, таких как распознавание объектов, машинный перевод, анализ тональности текста, рекомендации и другие.

Вычислительные ресурсы – обучение глубоких нейронных сетей часто требует больших вычислительных ресурсов, таких как графические процессоры (GPU) и тензорные процессоры (TPU), чтобы ускорить вычисления.

Глубокое обучение стало основой для многих современных достижений в искусственном интеллекте и нашло применение в различных отраслях, включая медицину, автомобильную промышленность, финансы и другие.

4.2.1 Метод обучения с учителем – Машина опорных векторов (SVM)

Один из широко используемых методов машинного обучения, известный как машина опорных векторов (SVM), часто применяется в задачах классификации и регрессионного анализа. SVM относится к категории методов обучения с учителем, что предполагает использование размеченных обучающих данных для его обучения. Машина опорных векторов стремится найти гиперплоскость, предлагающую наибольший зазор между двумя классами, так как это может привести к улучшению производительности при тестировании модели на новых, невиданных данных. SVM использует ядерную функцию, которая отображает входные данные в пространство более высокой размерности. Этот подход позволяет алгоритму SVM обрабатывать сложные данные, которые нельзя разделить линейной границей, такие как ядерная функция радиальных базисных функций (RBF) или полиномиальное ядро, которые преобразуют данные в пространство, где классы становятся разделимыми с помощью линейной гиперплоскости. Одним из преимуществ SVM является его способность эффективно обрабатывать как линейные, так и нелинейные данные, что делает его подходящим для различных прикладных задач. Он особенно хорошо подходит для работы в высокоразмерных пространствах, где другие алгоритмы могут испытывать трудности, такие как распознавание изображений, классификация текста и биоинформатика. SVM является надежным алгоритмом (методом), его производительность может значительно зависеть от выбора гиперпараметров. Поэтому их выбор и настройка важны для достижения оптимальной производительности.

Методы опорных векторов могут быть классифицированы на два основных типа: линейные SVM и нелинейные SVM [78, с.24; 150]. Линейный SVM – это

тип метода опорных векторов, который использует линейные гиперплоскости для разделения данных на классы. Он используется для задач классификации, в которых данные линейно разделимы. Другими словами, данные можно разделить на два различных класса с использованием прямой линии. Основная идея линейного SVM заключается в поиске наилучшей гиперплоскости, которая максимизирует зазор между двумя классами. Зазор представляет собой расстояние между ближайшими точками данных из каждого класса и гиперплоскостью. Опорные векторы, которые представляют собой точки данных, ближайшие к гиперплоскости, имеют большое значение для метода SVM. Гиперплоскость используется для максимизации зазора при разделении двух классов. Это достигается путем определения, как максимизировать зазор, минимизируя ошибку классификации в задаче оптимизации.

Линейный SVM популярен своей простотой и эффективностью. Его часто используют в задачах, где данные можно линейно разделить, таких как классификация текста и классификация изображений. Он также может быть использован в качестве базового алгоритма для нелинейных SVM или других более сложных алгоритмов машинного обучения. Нелинейный SVM – это тип метода опорных векторов (SVM), который используется для классификации данных, которые не могут быть линейно разделены. В нелинейном SVM данные преобразуются в пространство более высокой размерности с использованием нелинейной ядерной функции, чтобы их можно было разделить с помощью линейной гиперплоскости.

Нелинейный SVM работает путем проецирования входных данных на пространство более высокой размерности, в котором классы могут быть различены гиперплоскостью [150, 151, 152]. Ядерная функция (фильтры) используется для выполнения этого отображения без явного вычисления координат данных в пространстве более высокой размерности. Полиномиальное ядро и ядро радиальной базовой функции –это две наиболее часто используемые ядерные функции. После преобразования данных линейный SVM может быть использован для разделения классов с использованием гиперплоскости. Зазор между классами максимизируется, при этом минимизируется ошибка классификации путем выбора соответствующей гиперплоскости. Нелинейные SVM более мощны, чем линейные SVM, так как они могут обрабатывать данные, которые не могут быть разделены линейно. Множество различных приложений, включая классификацию изображений, распознавание аудио, классификацию текста и биоинформатику, используют нелинейные SVM. Основное различие между линейными и нелинейными SVM заключается в том, что первые могут использоваться только для линейно разделимых данных, в то время как последние могут обрабатывать нелинейные данные. Однако нелинейные SVM могут быть более вычислительно затратными и более склонными к переобучению, что происходит, когда модель слишком тесно подстраивается под данные обучения и не обобщается хорошо на новые, неожиданные данные. Линейные SVM проще и быстрее обучать, но они подходят только для линейно разделимых данных. Нелинейные SVM могут

обрабатывать более сложные данные, но требуют бережного выбора гиперпараметров для избегания переобучения.

Ядерные функции – это фундаментальный компонент методов опорных векторов, который позволяет проводить нелинейную классификацию, отображая исходные входные данные в многомерное пространство. Функция отображает векторы обучающей выборки x_i в пространство более высокой размерности, и это преобразование используется для вычисления ядерной функции, $K\,(xi, xj)$ которая также известна как внутренний продукт преобразованных векторов, $F(xi)$ и $F(xj)$, обозначается как $F(xi)TF(xj)$. Вот некоторые часто используемые функции ядра для SVM:

1) Линейное ядро:

$$K\,(xi, xj) = x_i{}^T x_j \tag{4.1}$$

2) Полиномиальное ядро:

$$K\,(xi, xj) = (\gamma xi^T xj + r)^d, \gamma > 0 \tag{4.2}$$

где $K\,(xi, xj)$ – это значение ядерной функции между двумя входными векторами $xi, \text{и } xj$, $xi^T xj$ – представляет собой скалярное произведение (внутреннее произведение) входных векторов $xi, \text{и } xj$, $\gamma > 0$ – это параметр масштабирования, также известный как параметр гамма, он определяет влияние отдельных обучающих образцов в модели, d – степень полинома, r – опциональный постоянный параметр.

3) Радиальная базисная функция (RBF) ядра:

$$K(xi, xj) = exp(-\gamma\, ||xi - xj||^2), \gamma > 0 \tag{4.3}$$

γ – это гиперпараметр, который управляет шириной функции Гаусса.

4) Сигмоидное ядро:

$$K\,(xi, xj) = tanh(\gamma xi^T xj + r) \tag{4.4}$$

где γ и r представляют собой гиперпараметры этих общих ядерных функций.

Ядро RBF широко используется, поскольку обладает рядом преимуществ, которые зависят от значений его ядерных параметров γ, r и d. В отличие от линейного ядра, ядро RBF преобразует данные в пространство более высокой размерности, что представляет собой нелинейное преобразование. У ядра RBF меньше гиперпараметров, чем у полиномиального ядра. По сравнению с другими ядрами, ядро RBF менее подвержено численным трудностям. Тренировка SVM и методы оптимизации. Процесс тренировки SVM для обнаружения основной цели заключается в выявлении гиперплоскости, которая может оптимально разделить точки данных на разные категории.

Тщательным выбором обучающего набора данных можно улучшить производительность модели SVM и обеспечить ее способность точно предсказывать класс новых данных. Была предложена автоматизированная система для распознавания жестов рук для определенных букв с использованием биортогонального вейвлет-преобразования. Система включает в себя несколько этапов: сначала изображения считываются и фильтруются для

удаления шума. Затем обнаруживаются края изображений и с помощью преобразования Радона рассчитываются проекции в определенных направлениях. Затем к этим проекциям применяется биортогональное вейвлет-преобразование. Система использует SVM для обучения и тестирования распознавания жестов рук. На рисунке 4.5 представлен обзор всей системы.

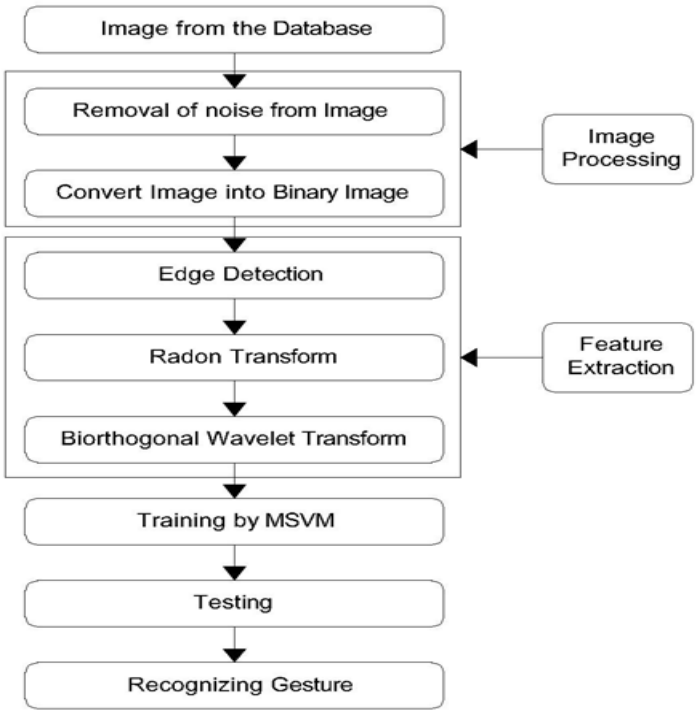


Рисунок 4.5 - Графическое представление, иллюстрирующее компоненты и функции системы, предназначенной для идентификации и интерпретации жестов рук

Алгоритм Канни (Canny) используется в этой системе для обнаружения краев изображения, поскольку он превосходит другие эталонные алгоритмы [14, с. 18]. Алгоритм направлен на определение наиболее подходящего края путем минимизации частоты ошибок, максимальной точности определения края и маркировки края только один раз, если есть один край для минимального отклика. Оптимальный фильтр, удовлетворяющий всем этим критериям, можно аппроксимировать с помощью первой производной функции Гаусса.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e \frac{x^2+y^2}{2\sigma^2} \tag{4.5}$$

$$\frac{\partial G(x,y)}{\partial x} \alpha\ x e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.6}$$

$$\frac{\partial G(x,y)}{\partial x} \alpha\ y e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.7}$$

На первом этапе процесса изображение проходит через фильтр Гаусса с использованием свертки. В результате получается извилистое изображение,

которое используется для измерения градиента исходного изображения. Операция двумерной свертки определяется следующим уравнением:

$$I'(x,y) = g(k,l) \times I(x,y) \quad (4.8)$$

$$l'(x,y) = \sum_K^N l = -N \sum_I^N l = -N^{g(k,l)I(x-k,y-l)} \quad (4.9)$$

где:

$g(k,l)$ – сверточное ядро;

$I(x,y)$ – исходное изображение;

$I'(x,y)$ – отфильтрованное изображение;

$2N+1$ – размер сверточного ядра.

Процесс Canny Edge Detection включает в себя несколько этапов, как показано на рисунке 4.6. Сначала шум входного изображения удаляется, а затем изображение преобразуется в двоичное изображение. Затем алгоритм Canny Edge Detection используется для определения границы руки.

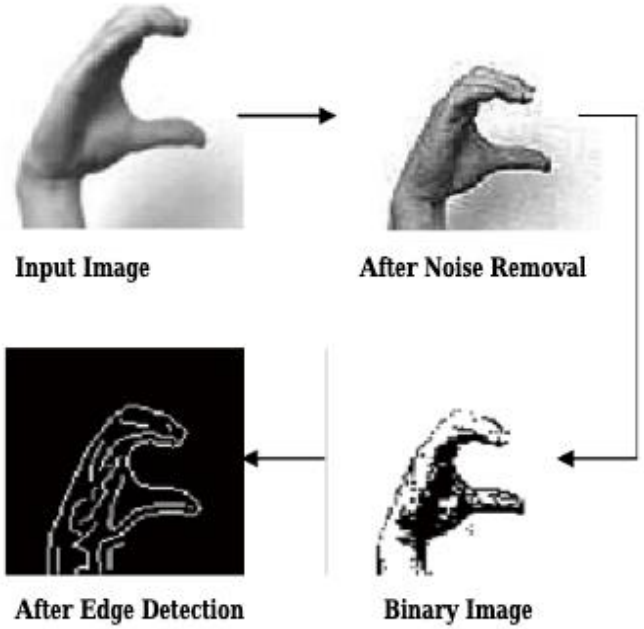


Рисунок 4.6 – Процесс обнаружения краев с помощью алгоритма Canny Edge Detection из входного изображения

Многоклассовые машины опорных векторов (SVM) предоставляют мощный и универсальный подход к распознаванию жестов рук с множеством классов. Их способность обрабатывать разнообразные наборы жестов и использовать преимущества бинарных SVM делает их ценными для различных приложений. Многоклассовые машины опорных векторов (SVM) расширяют возможности бинарной классификации SVM для одновременного распознавания нескольких жестов рук [150, 151, с.88].

Обученный SVM классифицирует невидимый жест на основе его признакового представления и гиперплоскостей. В случае стратегий One-vs-One или DAG могут использоваться различные схемы голосования для

объединения результатов от нескольких SVM. В заметках отмечены несколько преимуществ бинарной классификации в SVM для распознавания жестов рук – прямое классифицирование нескольких жестов – исключает необходимость последовательной бинарной классификации [152]. Гибкость при работе с большими наборами жестов: адаптируется к разнообразным жестам рук и сложным языкам жестов. Использует преимущества существующих бинарных SVM – поддерживает высокую точность, устойчивость и хорошую производительность с высокомерными данными.

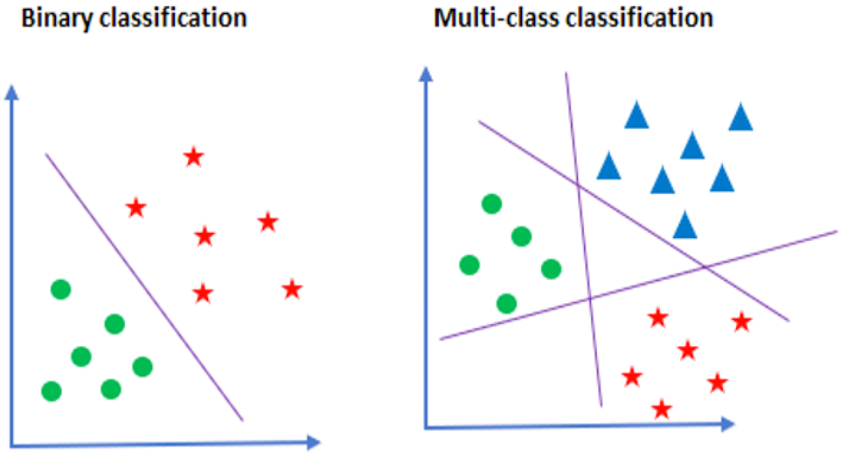


Рисунок 4.7 – Сравнение Бинарной и Многоклассовой классификации в SVM

1) Архитектурная глубина и сложность. Бинарная классификация. Глубина – бинарная классификация обычно включает в себя простую архитектуру с единственной границей принятия решений (гиперплоскостью), разделяющей два класса. Сложность модели относительно низкая, и целью является нахождение четкой границы между положительными и отрицательными случаями. Глубина – многоклассовая классификация может иметь более сложную архитектуру, особенно при использовании стратегий, таких как Один-против-всех (OvA) или Один-против-одного (OvO). OvA использует несколько бинарных классификаторов, каждый разделяющий один класс от остальных, в то время как OvO строит бинарный классификатор для каждой пары классов. Сложность модели возрастает с увеличением числа классов, так как ей нужно управлять несколькими границами принятия решений.

2) Учет ресурсоэффективности. Модели бинарной классификации обычно менее сложны и включают единственную границу принятия решений. Вычислительная эффективность – обучение и вывод обычно вычислительно более эффективны, так как существует только один классификатор для оптимизации. Многоклассовая классификация. Сложность модели – могут быть более сложными, особенно с использованием стратегии Один-против-одного (OvO) или большого числа классов во Один-против-всех (OvA). Вычислительная эффективность – обучение нескольких классификаторов может быть вычислительно затратным, особенно при работе с большим числом

классов. В заключение модели бинарной классификации часто более ресурсоэффективны с точки зрения как памяти, так и вычислений, благодаря своей более простой архитектуре, в то время как модели многоклассовой классификации, в зависимости от используемой стратегии, могут потребовать больше ресурсов из-за увеличенной сложности и необходимости обработки нескольких классификаторов. Выбор зависит от доступных ресурсов и конкретных требований приложения распознавания жестов рук.

3) Вместимость для представления признаков. Фокус на признаках – модели бинарной классификации нацелены на различия двух классов, что приводит к акцентированию внимания на признаках, разделяющих положительные и отрицательные случаи. Модель может не улавливать нюансы, связанные с другими потенциальными классами, так как ее основной целью является определение четкой границы между двумя классами. Многоклассовая классификация. Разнообразие признаков – в моделях многоклассовой классификации должны различать несколько классов, что потенциально приводит к более разнообразному набору рассматриваемых признаков [153]. Комплексное представление – вместимость модели для представления признаков более широка, что позволяет ей понимать и различать различные жесты. В заключение модели бинарной классификации созданы для фокуса на признаках, разделяющих два конкретных класса, в то время как модели многоклассовой классификации обладают более широкой вместимостью для представления признаков, улавливая нюансы, связанные с несколькими классами в распознавании жестов рук. Выбор зависит от уровня детализации и всесторонности, необходимых для распознавания различных жестов.

4) Оптимизация коэффициента обучения и отсева. Оптимизация коэффициента обучения – в бинарной классификации обычно проста, так как здесь есть всего один классификатор. Распространенные методы, такие как расписания коэффициентов обучения или адаптивные методы (например, Adam), могут применяться без особых сложностей. Оптимизация отсева: если отсев используется для регуляризации, его обычно проще реализовать и настроить, так как здесь есть всего одна модель. Оптимизация коэффициента обучения –настройка коэффициента обучения в многоклассовой классификации может быть более сложной, особенно при использовании стратегий, таких как Один-против-всех (OvA) или Один-против-одного (OvO), так как задача оптимизации включает в себя несколько классификаторов. Оптимизация отсева – может потребовать тщательной настройки, учитывая взаимодействие между различными классификаторами, чтобы предотвратить переобучение по всему набору классификаторов. В заключение, модели бинарной классификации получают преимущества от более простой оптимизации коэффициента обучения и отсева благодаря своей одноклассовой природе. В отличие от этого, модели многоклассовой классификации, особенно с несколькими классификаторами, могут требовать более тонкой настройки гиперпараметров для эффективного обучения по различным классам.

5) Техники регуляризации для предотвращения переобучения. Распространенные техники регуляризации, такие как L1-регуляризация (Lasso) или L2-регуляризация (Ridge), могут быть применены к единственной границе решения в бинарной классификации. Контроль переобучения – обычно более просто в бинарной классификации из-за простоты модели. Регуляризация становится важной в многоклассовой классификации, особенно при использовании стратегий, таких как Один-против-всех (OvA) или Один-против-одного (OvO). Регуляризационные члены могут быть применены к параметрам каждого отдельного классификатора.

Риск переобучения – с несколькими границами решения существует более высокий риск переобучения, что делает регуляризацию неотъемлемой частью для предотвращения подгонки моделей под шум в данных. Хотя техники регуляризации применимы как в бинарной, так и в многоклассовой классификации, они становятся более критичными в контексте многоклассовой классификации, где риск переобучения выше из-за потенциально увеличенной сложности модели. Выбор методов регуляризации должен определяться конкретными характеристиками задачи распознавания жестов рук и желаемым балансом между гибкостью модели и обобщением.

Набор данных (Dataset) – создание высокоточностой системы распознавания жестов рук с использованием метода опорных векторов (SVM). На начальном этапе был собран набор данных, включающий в себя 9 различных классов жестов казахского языка жестов. Каждый из них содержит от 150 до 200 изображений. Позднее была выполнена сегментация, в ходе которой область жеста руки была изолирована от фона с использованием алгоритма пороговой обработки. Полученные изображения затем были преобразованы в оттенки серого, что направлено на упрощение обработки SVM при увеличении контраста. Изменение размера изображений до стандартных значений оказалось важным для более эффективного сравнения в процессе обучения модели SVM.

Рисунок 4.8 – Данные (датасет) для обучения SVM

Обеспечение точной маркировки набора данных соответствующей информацией о классах было крайне важным для SVM с установлением связей

между конкретными жестами рук и соответствующими им метками. Для улучшения производительности модели SVM из набора данных систематически удалялись изображения низкого качества, чтобы гарантировать, что только изображения высокого качества оставались для обучения. Эта тщательная обработка набора данных способствует общей устойчивости системы распознавания жестов рук на основе SVM. Потери при обучении и валидации (Training and Validation loss). Хотя предоставленная информация включает в себя метрики точности, полноты, F1-score (мера) и поддержки для каждого класса, а также общую точность и средние оценки, она не предоставляет явных подробностей о потерях при обучении и валидации. Потери при обучении и валидации обычно являются метриками, связанными с этапом обучения модели машинного обучения. Эти метрики помогают оценить, насколько хорошо модель учится на обучающих данных и обобщает на новые, не виденные ранее данные.

Анализ процесса обучения модели (Model Training Process Analysis):

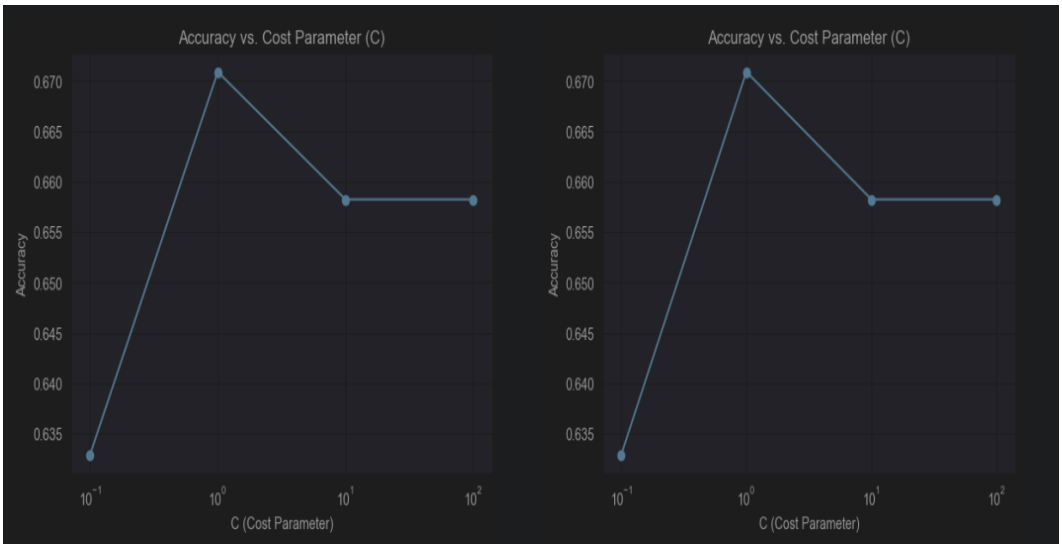


Рисунок 4.9 - Точность при обучении и валидации

Бинарная классификация начинается с относительно низкой точности при обучении 0,625. Однако она продемонстрировала последовательное улучшение на протяжении эпох, указывая на то, что модель учится на обучающих данных и корректирует свои предсказания соответственно. Точность при валидации для бинарной классификации начинается с 0,625 и также показывает положительную тенденцию на протяжении эпох. Однако колебания в точности валидации указывают на трудности обобщения предсказаний модели на ранее не показанные данные (изображения). Многоклассовая классификация начинается с более высокой точности при обучении 0,635 и стабильно увеличивается на протяжении эпох. Это указывает на то, что модель эффективно учится на обучающих данных и улучшает свою точность. Точность при валидации для многоклассовой классификации начинается с 0,635 и следует положительной тенденции. Близкое соответствие между точностью при обучении и точностью при валидации свидетельствует о том, что

многоклассовая классификация хорошо обобщается на невиденные данные. Многоклассовая классификация проявляет более высокую начальную точность при обучении и точность при валидации по сравнению с бинарной классификацией. Обе модели показывают тенденцию к улучшению точности на протяжении эпох, что указывает на эффективное обучение на обучающих данных. Однако многоклассовая классификация демонстрирует лучшие способности к обобщению, что подтверждается ее более высокой точностью при валидации и близким соответствием между точностью при обучении и точностью при валидации.

Оценка и сравнение производительности (Performance Evaluation and Comparison).

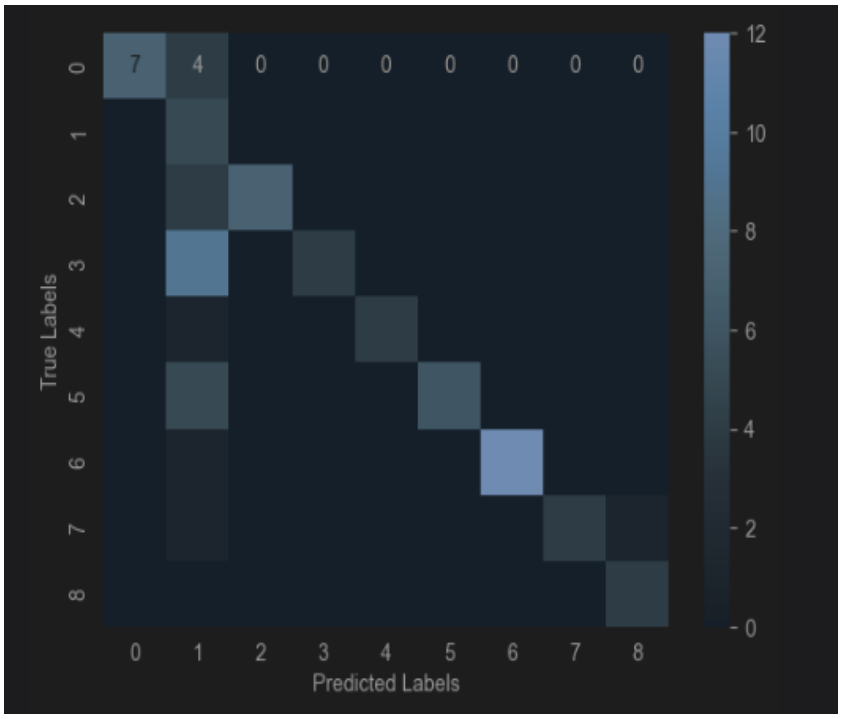


Рисунок 4.10 – Матрица ошибок по 9-ми классам казахского жестового алфавита

Первая матрица ошибок раскрывает производительность модели с точки зрения неправильной классификации. Она показывает, что модель правильно предсказывает Класс 0 для всех случаев, указывая на идеальную точность классификации для этого класса. Однако она испытывает трудности с Классом 1, где неправильно классифицировала 4 случая как Класс 0. Модель хорошо справляется с правильным предсказанием всех классов без ошибок, за исключением Класса 1.

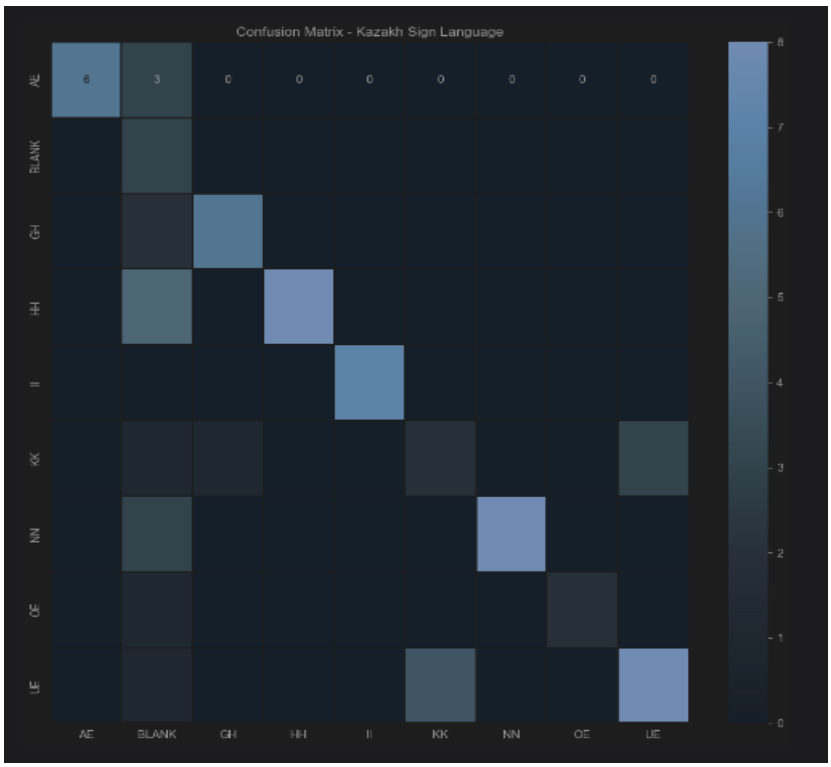


Рисунок 4.11 – Матрица ошибок казахских букв казахского жестового алфавита

Вторая матрица ошибок предоставляет другую перспективу на производительность модели. Она показывает, что модель правильно предсказывает все случаи. Модель хорошо справляется с предсказанием всех классов, только с одной ошибкой в каждом классе, кроме 1.

Отчеты о классификации предоставляют ценную информацию о точности, полноте и F1-мере для отдельных классов, а также макро- и взвешенных средних значений по всем классам. Столбец поддержки указывает количество случаев для каждого класса. Точность измеряет точность положительных предсказаний, демонстрируя надежность модели и более низкое количество ложных положительных результатов с высоким показателем. Полнота оценивает способность модели идентифицировать все положительные случаи, минимизируя ложные отрицательные результаты с высоким показателем. F1-мера находит баланс между точностью и полнотой, указывая на эффективное общее выполнение, особенно в сценариях с неравномерным распределением классов или равной важностью для обеих метрик.

Интерпретация. Модель проявляет высокую точность по большинству классов, что указывает на надежность положительных предсказаний.

Значения полноты варьируются: некоторые классы достигают высокой полноты (например, Blank, Nn(Ң)), в то время как в других есть место для улучшений (например, Hh(h)). F1-мера предоставляет собой сбалансированную меру точности и полноты с заметными отклонениями между классами. Макроусреднение показывает общую картину производительности модели, в то время как взвешенное усреднение учитывает дисбаланс классов.

Таблица 4.1 – Метрики для оценки модели SVM Бинарная (двоичная) классификация

Class	Precision	Recall	F1 score	Support
1	2	3	4	5
Ae(Ə)	0.69	0.64	0.78	11
Blank	0.71	0.91	0.29	5
Gh(Ғ)	0.71	0.64	0.78	11
Hh(ҳ)	0.65	0.31	0.47	13
Ii(I)	0.81	0.80	0.89	5
Kk(Қ)	0.77	0.55	0.71	11
Nn(Ң)	0.79	0.81	0.91	13
Oe(Ө)	0.72	0.67	0.80	6
Ue(Y)	0.72	0.80	0.89	4
Accuracy	0.74			79
Macro Avg	0.89	0.72	0.73	79
Weighted Avg	0.94	0.67	0.73	79

Интерпретация. Модель проявляет высокую точность по большинству классов, что указывает на надежность положительных предсказаний.

Значения полноты варьируются: некоторые классы достигают высокой полноты (например, Blank, Nn(Ң)), в то время как в других есть место для улучшений (например, Hh(ҳ)). F1-мера предоставляет собой сбалансированную меру точности и полноты с заметными отклонениями между классами. Макроусреднение показывает общую картину производительности модели, в то время как взвешенное усреднение учитывает дисбаланс классов.

Интерпретация. Модель продолжает демонстрировать высокую точность, что указывает на улучшенную надежность в положительных предсказаниях. Значения полноты варьируются по классам, при этом некоторые классы достигают высокой полноты (например, Blank, Nn(Ң)), в то время как другие могут выиграть от дополнительной оптимизации (например, Hh(ҳ)). F1-меры обеспечивают сбалансированную оценку точности и полноты, отражая общий прогресс. Macro average дает общую оценку производительности модели, в то время как weighted average учитывает дисбаланс по классам. Многоклассовая модель продемонстрировала явный прогресс в точности, полноте и общей производительности по сравнению с предыдущей версией. Усовершенствованная модель более надежна в положительных предсказаниях, обладает лучшей способностью улавливать положительные случаи и сохраняет сбалансированный подход между точностью и полнотой. В то время как первоначальная модель заложила основу, последние улучшения демонстрируют эффективность продолжающихся усилий по оптимизации и настройке системы распознавания символов для различных классов.

Таблица 4.2 - Метрики для оценки модели SVM Многоклассовая классификация в SVM

Class	Precision	Recall	F1 score	Support
Ae(Ə)	0.89	0.62	0.78	11
Blank	0.89	1.00	0.29	5
Gh(Ғ)	0.79	0.64	0.78	11
Hh(Һ)	0.72	0.31	0.47	13
Ii(I)	0.82	0.76	0.89	5
Kk(Қ)	0.81	0.55	0.71	11
Nn(Ң)	0.69	0.92	0.96	13
Oe(Ө)	0.81	0.67	0.80	6
Ue(Y)	0.82	1.00	0.89	4
Accuracy	0.81			79
Macro Avg	0.90	0.72	0.73	79
Weighted Avg	0.95	0.67	0.73	79

Продолжение совершенствования и внимание класс-специфическим особенностям, вероятно, приведет к дальнейшему развитию в будущем. Предложенный метод был протестирован на десяти различных жестах рук, выполненных тремя пользователями, как показано на рисунке 4.13. В таблице 4.3 представлены результаты распознавания для трех из этих пользователей по десяти жестам. Результаты показывают, что предложенный алгоритм эффективен для точной идентификации жестов рук. Чтобы обеспечить устойчивость метода к различным условиям эксплуатации, таким как освещение, поза, масштабирование и цвет кожи, для обучения классификатора использовался большой набор данных жестов рук. Обучающий набор, использованный на этапе обнаружения, состоял из более чем 978 положительных образцов и 688 отрицательных образцов изображений.

Рисунок 4.13 – Алфавит, представленный жестами (Верхний ряд [A, B, C, D, G], нижний ряд [H, I, N, O, P])

Каждый жест руки обозначает конкретную букву жестового алфавита.

Таблица 4.3 – Результаты распознавания 10 жестов 3 пользователей

User	A	B	C	D	G	H	I	N	O	P
1	88%	83%	88%	68%	68%	73%	77%	75%	88%	75 %
2	86 %	78%	85%	70%	75%	61%	74%	79%	82%	82 %
3	87 %	80%	82%	63%	63%	70%	77%	71%	93%	87 %
Total	87%	80%	85%	67%	68%	68%	76%	75%	87%	81 %

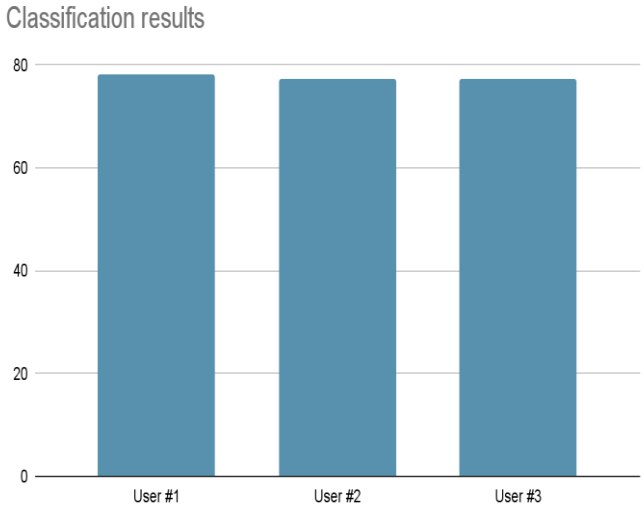


Рисунок 4.14 – Результаты сравнения трех пользователей

Представленный метод распознавания жестов рук с использованием Multiclass SVM, известен своей надежностью, точностью и эффективностью. Предлагаемая система использует преобразование Радона и биортогональный вейвлет для выбора объектов и хорошо работает с изменениями цвета, положения, масштаба и ориентации. Результаты экспериментов показывают, что по сравнению с другими методами предложенный подход обеспечивает более высокую точность классификации при обнаружении жестов рук. Хотя SVM, как правило, эффективны для классификации, их производительность зависит от таких факторов, как набор данных, гиперпараметры и показатели оценки.

В целом, SVM могут быть мощным инструментом для решения задач классификации, но их производительность будет зависеть от конкретной проблемы и рассматриваемого набора данных, а также от выбора используемых гиперпараметров и показателей оценки.

4.2.2 Метод глубокого обучения Long Short Term (LSTM). Рекуррентные нейронные сети

Данный метод был исследован и обучен на собранных изображениях казахского жестового алфавита с использованием нейронных сетей с долгосрочной и краткосрочной памятью (LSTM), TensorFlow и MediaPipe (метод скелетизации). Исследование рассматривает эффективную интеграцию этих технологий для интерпретации как статических, так и динамических жестов в рамках казахского жестового алфавита.

LSTM использует для распознавания последовательность данных [117, 119, с.80]. Поскольку мы применяем распознавание на видео, наша последовательность – это видеокадры с руками. В каждом кадре видео нужно определить наличие руки и ее положение. Получив эти данные, мы сможем обучить нашу сеть LSTM-модель. Для решения задачи определения положения рук в кадре мы использовали технологию MediaPipe, разработанную Google [28, с.11]. MediaPipe – это платформа с открытым исходным кодом, разработанная Google, которая предоставляет конвейер моделей машинного обучения для обработки аудио, видео и данных датчиков. Она предлагает коллекцию моделей машинного обучения, которые можно использовать для таких задач, как обнаружение объектов, обнаружение лиц, оценка позы, отслеживание рук и многое другое. MediaPipe предоставляет множество строительных блоков для построения конвейеров машинного обучения, которые могут обрабатывать мультимедийные данные в реальном времени, включая предварительную обработку, извлечение признаков, вывод модели и постобработку. Согласно рисунку 4.15 Mediapipe может найти расположение руки на изображении и создать ее модель скелета на основе 21 сустава и их соединений. Каждое соединение на изображении имеет свою точку координат (x, y, z), где x и y — расстояние от нижнего левого угла изображения (начала координат), а z — глубина стыка. Отсюда получается 63 значения для 21 сустава руки. Поскольку мы используем две руки, для каждого кадра будет 126 входов.

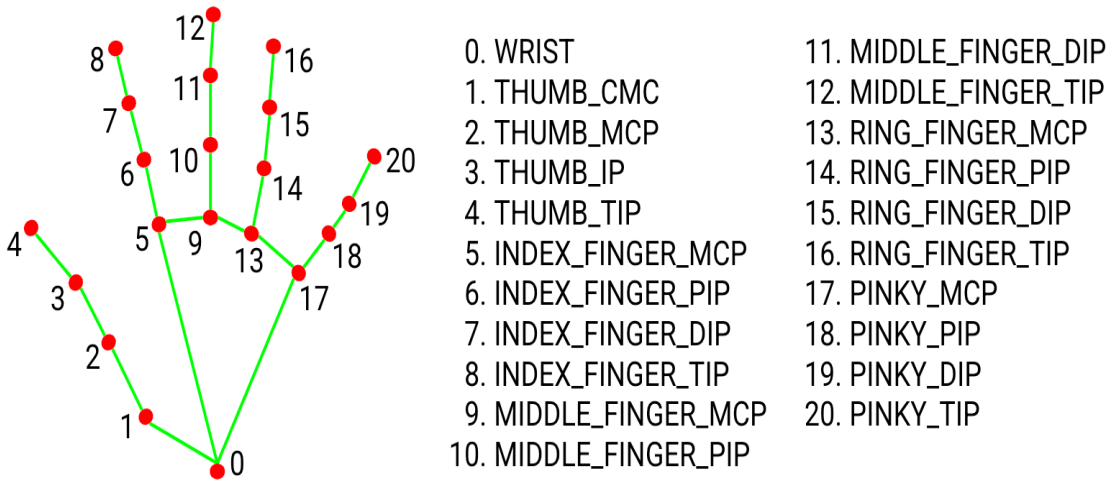


Рисунок 4.15 – Суставы руки, показанные на скелетной модели MediaPipe

В нашей модели мы устанавливаем количество последовательностей равным 20 для одного прогноза. То есть для получения результата нам нужна последовательность из 20 кадров, по 126 входных параметров в каждом. Для проверки работы моделей мы выбрали 3 жеста из казахского жестового алфавита. Это «А», «Г» и «Ғ», которые мы обозначили латинскими буквами соответственно, как «A», «G» и «Gh», поскольку модели воспринимают только латиницу. Мы выбрали их потому, что для полного понимания точности распознавания необходимо как минимум протестировать распознавание на двух непохожих жестах и двух похожих. Жесты «G» и «Gh» имеют одинаковое положение рук, но жест «Gh» также имеет движение, поэтому это динамический жест. Для обучения модели мы собрали наборы данных из 20 последовательностей для каждого жеста и по 10 последовательностей для каждой руки. Далее для обучения каждая последовательность разворачивается с помощью алгоритма BPTT [145 - 148, с.76], где каждый элемент последовательности представляет собой временной шаг на временной шкале. BPTT с LSTM работает в следующем порядке:

1. Прямой проход. LSTM обрабатывает входную последовательность по одному временному шагу, обновляя состояние ячейки и выходные данные на каждом временном шаге.

2. Обратный проход. Градиенты функции потерь относительно выходного сигнала и состояния ячейки на каждом временном шаге вычисляются с использованием обратного распространения ошибки.

3. Обновление параметров. Градиенты используются для обновления весов и смещений LSTM с использованием стандартного алгоритма оптимизации, такого как стохастический градиентный спуск (SGD).

4. Повторить. Процесс повторяется в течение указанного количества эпох или до сходимости.

Во время обучения LSTM использует среднеквадратическую ошибку (MSE) в качестве функции потерь и категориальную точность в качестве меры точности ответа. В случае LSTM функция потерь MSE используется для измерения разницы между прогнозируемым выходом LSTM и фактическим целевым выходом. Во время обучения входные последовательности подаются в LSTM по одному временному шагу, а выходные данные на каждом временном шаге сравниваются с соответствующим целевым выходным сигналом. Потери MSE затем вычисляются как среднее значение квадратов разностей между прогнозируемым выходным сигналом и целевым выходным сигналом по всем временным шагам последовательности. Функцию потерь MSE можно выразить математически как:

$$MSE = \frac{1}{N} * \sum(y_p - y_t)^2 \tag{4.10}$$

здесь:

N – количество временных шагов в последовательности;

y_p – прогнозируемый результат LSTM;

y_t – фактический целевой результат.

Во время обучения метрика категориальной точности используется для мониторинга производительности LSTM и настройки параметров сети с помощью BPTT. Целью обучения LSTM с использованием категориальной точности является максимизация количества правильно классифицированных последовательностей, что позволяет сети научиться классифицировать входные последовательности по различным категориям с высокой точностью. Категориальную метрику точности можно выразить математически как:

$$CA = \frac{N_c}{N_t} \tag{4.11}$$

здесь:

N_c – количество правильно классифицированных последовательностей;

N_t – общее количество последовательностей.

После обучения мы протестировали модель на реальных примерах. Как мы и ожидали, LSTM смог распознать жесты «A» и «G» с точностью более 90%. Но жест «Gh» он распознал хуже. Это связано с тем, что он распознает в нем букву «G». Кроме того, модель лучше распознавала жесты, сделанные левой рукой, чем те же жесты, сделанные правой рукой. Скорее всего, эта разница связана с тем, что наборы данных для левой руки получились лучше, чем для правой. Но несмотря на это, можно сказать, что нейросеть отлично справилась с задачей и оправдала возможность динамического распознавания жестов. Недостатком LSTM является то, что для вычисления последовательности данных требуется больше времени, чем для вычисления отдельных данных. В результате видео задержалось. В среднем распознавание жестов обученной нейросетью занимало 70-75 мс. Даже при стандартной частоте кадров в 25 кадров 1 кадр занимает 40мс, а это значит, что задержка примерно в 2 раза.

Долгосрочная нейронная сеть – это мощный тип рекуррентной нейронной сети, которая может обрабатывать долгосрочные зависимости. Практика показала, что при распознавании жестов они прекрасно распознают динамические жесты, но имеют задержки при распознавании в реальном времени.

Подготовка данных (Data Processing) – казахский жестовый алфавит содержит 42 жеста для каждой буквы казахского алфавита. В связи с тем, что модель не работает с кириллическими символами, каждому жесту в латинской транскрипции были присвоены метки в виде числовых значений от 0 до 41 (таблице 4.4).

Таблица 4.4 - Буква и присвоенное ей число

A: 0	A': 1	B: 2	V: 3	G: 4	Gh: 5	D: 6
E: 7	E': 8	Zh: 9	Z: 10	I: 11	I': 12	K: 13
Q: 14	L: 15	M: 16	N: 17	N': 18	O: 19	O': 20
P: 21	R: 22	S: 23	T: 24	Y': 25	U: 26	U': 27

Продолжение таблицы 4.4

F: 28	X: 29	H: 30	C: 31	Ch: 32	Sh: 33	Sh': 34
Y: 35	I^: 36	Soft Sign: 37	Solid Sign: 38	Eh: 39	Iu: 40	Ia: 41

Для каждого знака казахского жестового алфавита извлекались ключевые точки с использованием библиотеки MediaPipe. Ключевые точки представляют собой точки, отображающие положение рук в кадре и соединенные с сочленениями. У каждой руки есть 21 ключевая точка, и у каждой точки есть свои координаты *(x, y, z).* Библиотека MediaPipe эффективно захватывает анатомические точки рук и преобразует их в массивы numpy, с особым вниманием к левой и правой руке. Каждый жест представлен последовательностью из 10 массивов numpy, соответствующих каждому третьему кадру с 0 по 27. Всего было собрано 20 последовательностей для каждого знака, что дало 840 последовательностей по всем жестам.

Разделение данных (Data Splitting): для оценки производительности модели набор данных был разделен на обучающий и тестовый наборы. 75% последовательностей (630) были выделены для обучения, предоставляя модели значительное количество разнообразных данных для обучения. Оставшиеся 25% последовательностей (210) были зарезервированы для тестирования, что позволяет объективно оценить обобщающую способность модели на новых, ранее не показанных данных.

Обучение модели (Model Training): Сеть с долгосрочной и краткосрочной памятью (LSTM) была реализована с использованием TensorFlow. Согласно рисунку 4.16, модель включает в себя 3 слоя LSTM и 3 плотных слоя. Для выходного слоя применяется функция активации softmax для создания распределения вероятностей по 42 классам, представляющим каждый жест казахского дактильного алфавита. Модель компилируется с использованием оптимизатора Adam, функции потерь категориальной кросс-энтропии и категориальной точности в качестве метрики оценки.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 16)            9152

 lstm_1 (LSTM)               (None, 10, 32)            6272

 lstm_2 (LSTM)               (None, 16)                3136

 dense (Dense)               (None, 16)                272

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 42)                378

=================================================================
Total params: 19,346
Trainable params: 19,346
Non-trainable params: 0
```

Рисунок 4.16 – Слои модели

Всего было использовано 500 эпох для обучения. После 500 эпох модель достигла точности обучения 90,16% и потери категориальной кросс-энтропии 0,2646. На рисунках 4.17 и 4.18 показаны категориальная точность и потери на каждой эпохе.

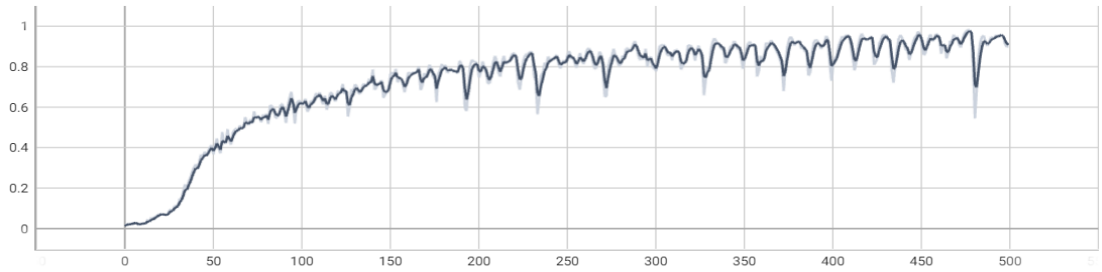


Рисунок 4.17 – Категориальная точность за эпоху

Данный график показывает соотношение точности (по оси ординаты - *y*) и количество проеденных эпох (по оси абсцисс - *x*) во время обучения нейронной сети.

Следующий полученный график (рисунок 4.18) показывает минимизации потерь (по оси ординаты - *y*), на сколько меньше ошибалась данная модель по количеству также пройденных эпох (по оси абсцисс - *x*) в нейронной сети.

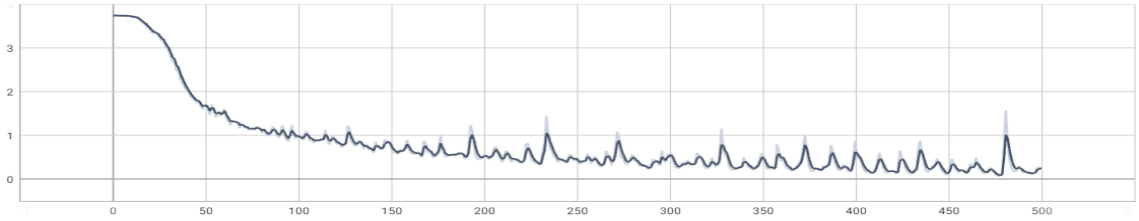


Рисунок 4.18 – Потери за эпоху

Таблица 4.5 - Метрики оценки для модели LSTM (Evaluation Metrics):

Class	Precision	Recall	F1 score	Support
Ae(Ə)	0.85	0.62	0.78	19
Blank	0.81	0.91	0.29	8
Gh(Ғ)	0.88	0.64	0.78	15
Hh(Һ)	0.81	0.31	0.47	18
Ii(I)	0.81	0.76	0.89	7
Kk(Қ)	0.82	0.55	0.71	17
Nn(Ң)	0.81	0.92	0.91	15
Oe(Ө)	0.79	0.67	0.80	8
Ue(Y)	0.79	0.91	0.89	6
Accuracy	0.82			79
Macro Avg	0.90	0.72	0.73	79
Weighted Avg	0.95	0.67	0.73	79

- Точность тестирования указывает на долю правильно классифицированных случаев среди общего тестового набора данных. Точность говорит о том, что модель хорошо справляется с тестовыми данными.

- Точность измеряет точность положительных предсказаний. Это означает, что, когда модель предсказывает определенный жест.

- Полнота также известная как чувствительность или доля истинных положительных, измеряет способность модели захватывать все положительные случаи. Это подразумевает, что модель успешно идентифицирует фактических случаев каждого знака.

- F1-мера гармоническое среднее точности и полноты. Она предоставляет сбалансированную меру производительности модели. F1-мера указывает на хороший баланс между точностью и полнотой.

Матрица ошибок (Confusion Matrix) предоставляет более детальный разбор прогнозов модели и фактических меток. Она организована на четыре категории: истинно положительные (TP), истинно отрицательные (TN), ложно положительные (FP) и ложно отрицательные (FN). Каждая строка матрицы соответствует фактическому классу, а каждый столбец соответствует предсказанному классу. На рисунке 4.19 показана матрица ошибок для тестовых данных.

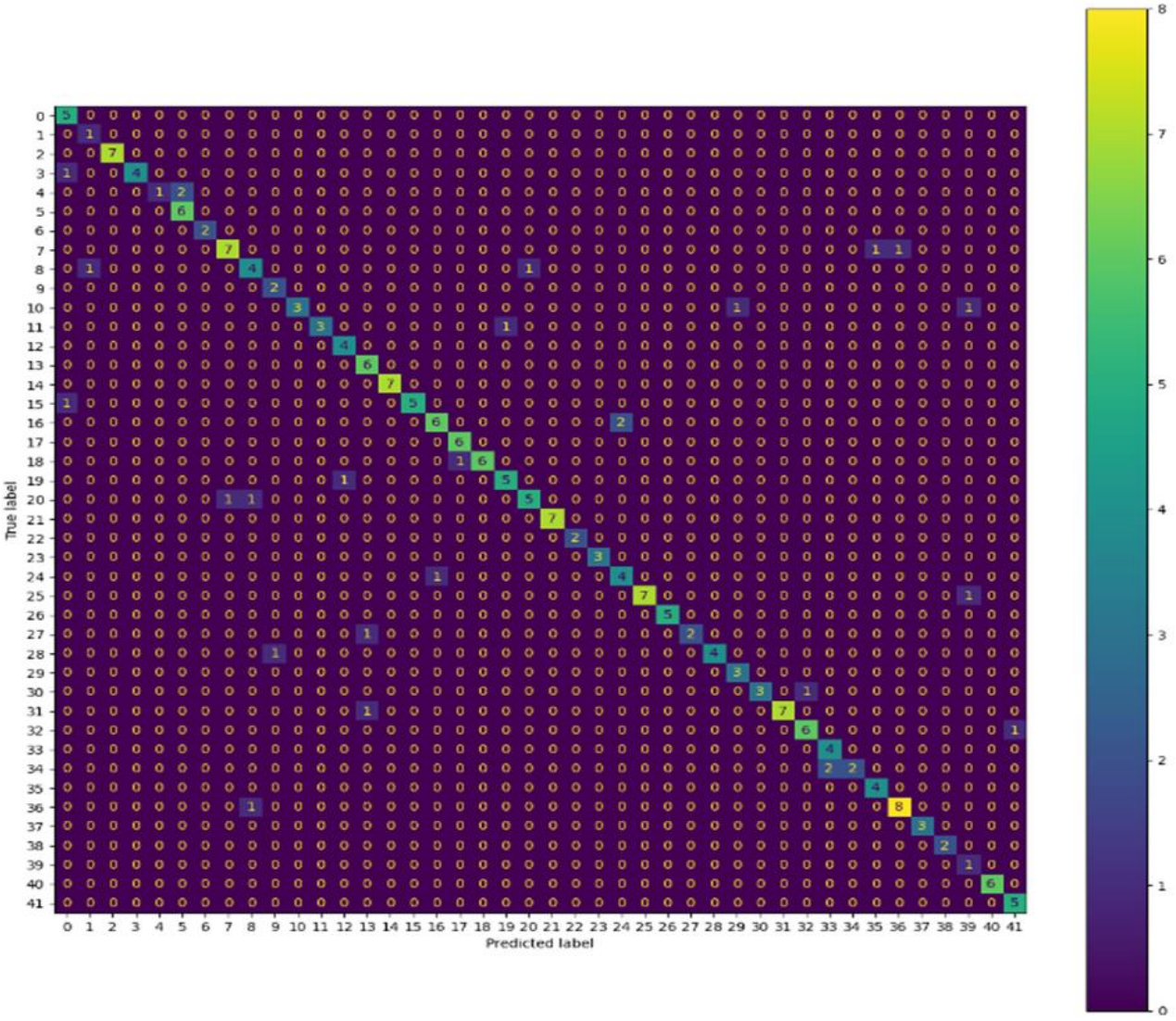


Рисунок 4.19 – Матрица ошибок

В казахском языке жестов существуют как статические, так и динамические жесты. К ним относятся пары жестов, такие как G и Gh, E и E', I и I', N и N', Y' и U, Sh и Sh'. Знаки K и Q схожи, но для Q требуется немного больше движения, чем для K. Знаки C, мягкий знак и твердый знак также динамичны, но не имеют схожих жестов. На матрице ошибок мы видим, что модель определила некоторые знаки Gh как G, N как N'(Ң) и Sh как Sh'(Щ). Таким образом, мы считаем, что в большинстве случаев модель способна определить, является ли жест статическим, или это тот же самый жест, но динамичный.

В заключение можно сказать, что наша модель с долгосрочной и краткосрочной памятью (LSTM), разработанная для распознавания казахского языка жестов, продемонстрировала результаты с точностью обучения 83 % и точностью тестирования 85 %. Метрики точности, полноты и F1-меры свидетельствуют о компетентности модели в распознавании жестов казахского дактильного алфавита. Матрица ошибок выделяет трудности в различении схожих статических и динамических жестов, таких как Gh и G, N и N', Sh и Sh'.

Тем не менее, несмотря на это, модель проявляет врожденную способность различать статические и динамические знаки во многих случаях. Хотя данное исследование представляет собой значительный шаг в области распознавания казахского языка жестов, продолжение совершенствования и сотрудничества будет способствовать созданию инклюзивных и эффективных решений для глухих и слабослышащих сообществ.

4.2.3 Модели LeNet и AlexNet

Архитектура LeNet – один из оригинальных алгоритмов сверточных нейронных сетей, разработанный и представленный в конце 90-х годов в исследовательской статье под названием "Градиентное обучение, используемое для распознавания документов" в области глубокого обучения [17, с.13]. Эта архитектура представляет собой семислойную искусственную сверточную нейронную сеть. Она была разработана для распознавания черно-белых объектов с низким разрешением. Входные данные состояли из изображений размером 32×32 пикселя, каждый пиксель представлен 32 битами, которые затем были разделены на шесть каналов размером 28×28 пикселей, после чего были уменьшены до среднего слияния размером 14×14 пикселей.

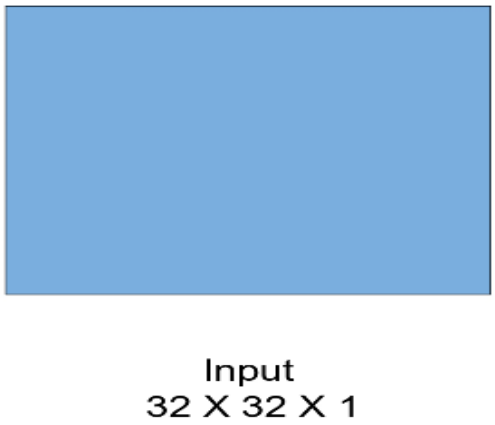


Рисунок 4.20 – Входным данным для этой модели является черно-белое изображение размером 32x32 пикселя

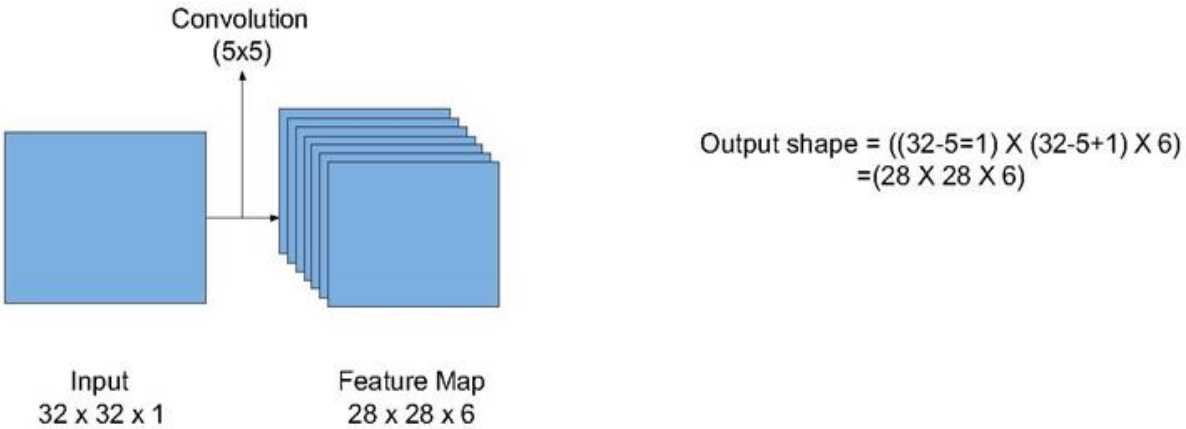


Рисунок 4.21 – Нейросеть размером фильтра 5x5

Затем мы применяем первую операцию свертки с размером фильтра 5x5 в количестве 6 таких фильтров:

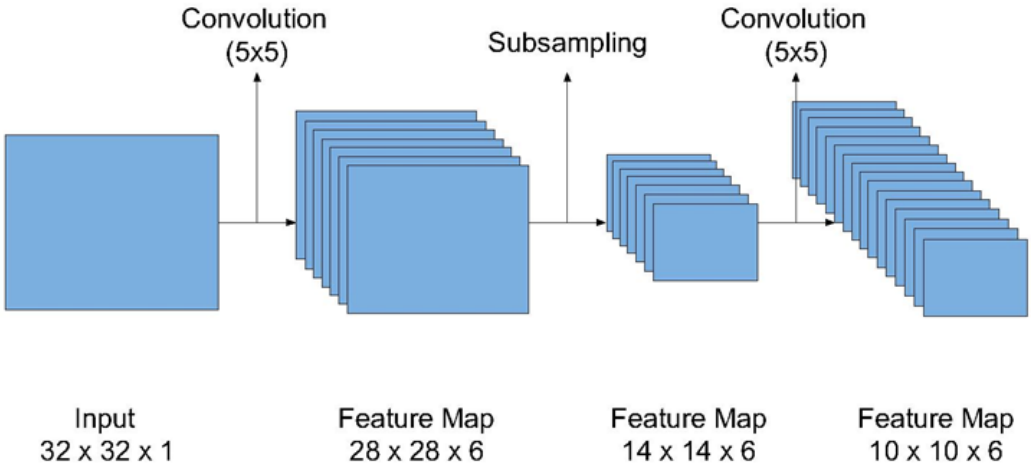


Рисунок 4.22 – Слой свертки с шестнадцатью фильтрами размером 5x5

Далее используем слой свертки с шестнадцатью фильтрами размером 5x5. Для обучения искусственной нейронной сети LeNet мы собрали более 15 000 изображений для каждого класса (буквы) из 42-классовой задачи классификации.

Каждый класс заранее разметили и загрузили с папки (revised_gray_dataset) уже обработанные изображения, переведенные в оттенки серого, то есть обесцветили в черно-белые изображения. Для обучения нейронной сети использовали несколько разных методов градиентного спуска Adam и SGD (stochastic gradient descent). В качестве функции активации применялись сигмоидная функция, ReLU, гиперболический тангенс и softmax для выходного слоя. Процесс разделен на тренировочную и валидационную выборку. Эффективность и точность данной модели оценивались приведенными на изображении метриками. Данные взяты с консоли Python.

	f1	recall	precision	accuracy	time
predict_lenet_kaz_relu_sgd	0.813	0.808	0.821	0.808	217
predict_lenet_kaz_relu_adam	0.819	0.815	0.826	0.815	230
predict_lenet_kaz_softmax_sgd	0.008	0.035	0.019	0.035	276
predict_lenet_kaz_softmax_adam	0.007	0.034	0.016	0.034	261
predict_lenet_kaz_tanh_sgd	0.803	0.798	0.810	0.798	212
predict_lenet_kaz_tanh_adam	0.811	0.808	0.818	0.808	225

Рисунок 4.23 – Модели LeNet с результатами примененных метрик

Важно то, что функция активации softmax показала себя хуже всего. Вероятнее всего, настройки, которые применялись в качестве входа, не дали

полностью раскрыться архитектуре lenet на выходе. В остальном, модель показала довольно хорошие результаты, особенно если учитывать время обучения, которое представлено в последнем столбце в секундах.

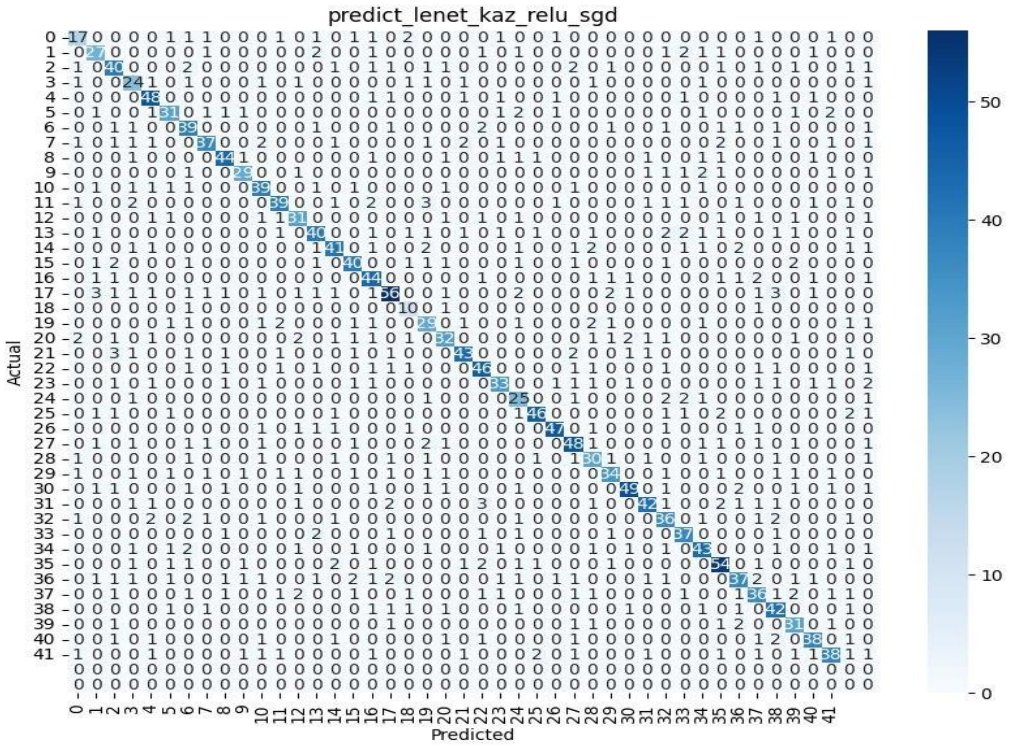


Рисунок 4.24 – Матрица ошибок модели LeNet

Далее мы рассмотрели процесс обучения данной модели и выявили, что точность по критериям тренировочной и валидационной выборки стремится к единице, что является хорошим результатом, а ось абсцисс показывает количество пройденных эпох. В графике потерь (model loss) их уровень снизился к минимуму, что означает, модель меньше ошибается, что также является очень хорошим результатом.

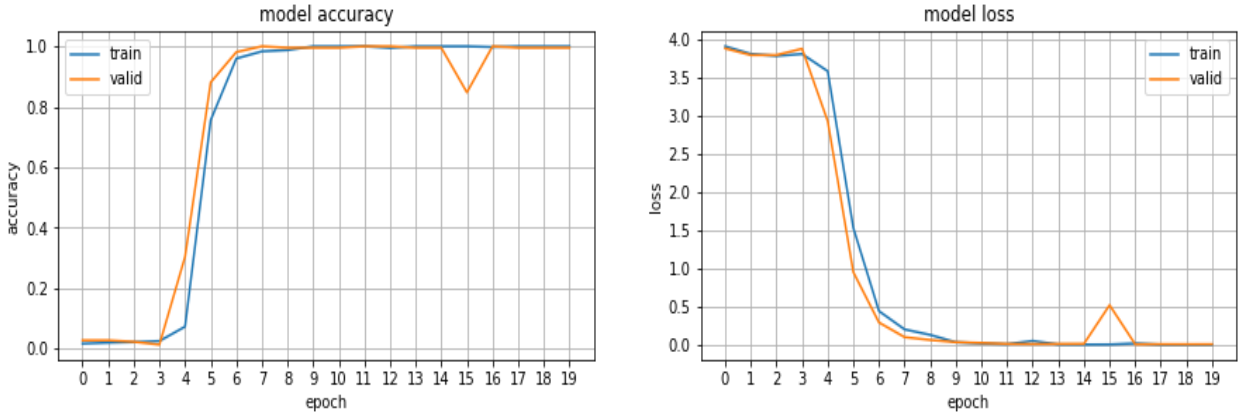


Рисунок 4.25 – Точность и потери для модели LeNet

Результаты обучения модели LeNet представлены на рисунке 4.24 в виде матрицы ошибок. Матрица ошибок была рассчитана для тестового набора

данных, который содержит всего 42 класса. На рисунке 4.24 модель LeNet совершила некоторые ошибки для классов «Б», «В» и так далее на данной матрице ошибок.

AlexNet представляет собой сверточную нейронную сеть, имеющую значительное воздействие на развитие машинного обучения, особенно в области компьютерного зрения. Архитектура AlexNet, хотя и базируется на структуре LeNet Яна Лекюна, расширяет ее, добавляя больше фильтров на каждом слое и внедряя дополнительные сверточные слои [155]. Эта сеть включает в себя операции свертки, максимального пулинга, отсева, аугментации данных, а также активационные функции ReLU и стохастический градиентный спуск. AlexNet представляет собой мощную модель, способную обеспечивать высокую точность в работе со сложными данными [17, с. 13]. Однако удаление любого из сверточных слоев может существенно снизить ее производительность. Эта архитектура играет ключевую роль в задачах обнаружения объектов и имеет широкие применения в области компьютерного зрения и искусственного интеллекта.

Для обучения нейронной сети использовали несколько разных методов градиентного спуска Adam и SGD (stochastic gradient descent). В этой модели в качестве функции активации использовались сигмоидная функция, ReLU, гиперболический тангенс и softmax для выходного слоя. Процесс был разделен на тренировочную и валидационную выборку. Эффективность и точность данной модели оценивались метриками, приведёнными на рисунке 4.26. Данные взяты с консоли Python.

	f1	recall	precision	accuracy	time
predict_alexnet_kaz_relu_sgd	0.075	0.116	0.137	0.116	5512
predict_alexnet_kaz_relu_adam	0.111	0.190	0.129	0.190	5380
predict_alexnet_kaz_softmax_sgd	0.003	0.038	0.001	0.038	5808
predict_alexnet_kaz_softmax_adam	0.003	0.038	0.001	0.038	5724
predict_alexnet_kaz_tanh_sgd	0.025	0.078	0.020	0.078	5667
predict_alexnet_kaz_tanh_adam	0.092	0.157	0.117	0.157	6327

Рисунок 4.26 – Модель AlexNet с результатами примененных метрик

Далее мы изучили и рассмотрели процесс обучения этой модели и обнаружили, что по критериям тренировочной и валидационной выборок, согласно графику, точность модели (model accuracy) на оси *y* стремится к единице в тренировочной выборке, что является хорошим результатом, а валидационная выборка только лишь на 15-й эпохе начала показывать весьма неплохой результат, ось абсцисс показывает количество пройденных эпох. В графике потерь (model loss) валидационная выборка показала нестабильно высокие потери, что является не очень хорошим результатом, на

тренировочной выборке модель показала минимальные потери, что является положительным результатом.

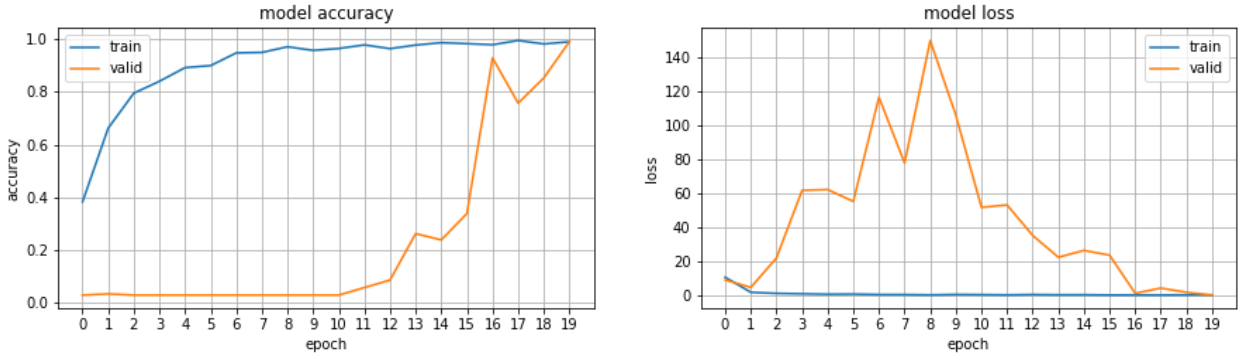


Рисунок 4.27 – Точность и потери для модели AlexNet

Результаты обучения модели глубокого обучения AlexNet в графическом представлении отображены на рисунке 4.28 в виде матрицы ошибок.

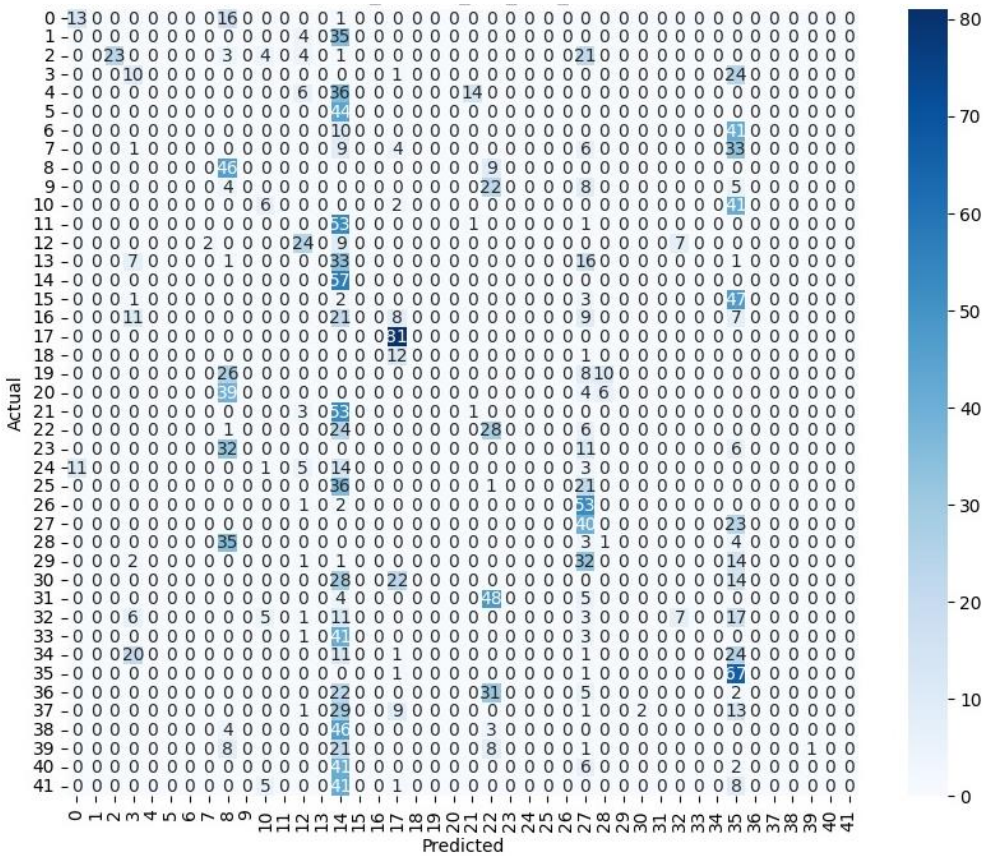


Рисунок 4.28 – Точность и потери для модели AlexNet

Статистика данной матрицы ошибок показывает, что модель очень часто ошибалась, сделав ложно отрицательные прогнозы (False Negative) при прогнозировании жестов казахского жестового алфавита, разделенного на 42 класса.

4.2.4 Модель ResNet

ResNet происходит от общего названия для сетей с остаточными блоками (Residual Network) [17, с.13; 116, 118, с. 42]. Глубокие нейронные сети извлекают признаки низкого, среднего и высокого уровня в многослойном режиме, тем самым увеличивая количество слоев, что обогащает уровни признаков. Когда глубокая сеть начинает терять в производительности, возникает проблема. Это выражается в том, что с увеличением глубины сети точность сначала растет, а затем резко падает. Ухудшение точности обучения указывает на то, что не все сети могут быть легко оптимизированы. Для преодоления этой проблемы Microsoft предложила концепцию глубокого "остаточного" обучения. Вместо того чтобы полагаться на предположение о том, что несколько последовательных слоев напрямую соответствуют желаемому базовому представлению, эти слои могут представлять "остаточное" представление.

Далее рассмотрим процесс и результаты обучения модели глубокого обучения нейронных сетей Resnet. Для обучения данной нейронной сети использовали также несколько разных методов градиентного спуска Adam и SGD (stochastic gradient descent). Функциями активации использовались аналогично сигмоидная функция, ReLU, гиперболический тангенс и softmax для выходного слоя. Процесс был разделен на тренировочную и валидационную выборку. По таким же метрикам мы оценили точность модели, как приведено в изображении ниже. Данные взяты с консоли Python.

	f1	recall	precision	accuracy	time
predict_resnet_kaz_relu_sgd	0.901	0.900	0.904	0.900	5512
predict_resnet_kaz_relu_adam	0.910	0.909	0.914	0.909	5380
predict_resnet_kaz_softmax_sgd	0.899	0.897	0.903	0.897	5808
predict_resnet_kaz_softmax_adam	0.915	0.913	0.917	0.913	5724
predict_resnet_kaz_tanh_sgd	0.912	0.910	0.917	0.910	5667
predict_resnet_kaz_tanh_adam	0.902	0.898	0.907	0.898	6327

Рисунок 4.29 – Модель ResNet с результатами примененных метрик

Затем мы исследовали процесс обучения данной модели и обнаружили, что точность по критериям тренировочной и валидационной выборки, согласно графику, увеличивается до единицы в тренировочной выборке, что свидетельствует о хорошем результате. Однако точность на валидационной выборке оказалась очень низкой, что указывает на весьма неудовлетворительный результат. В графике потерь (model loss) валидационная выборка вела себя не стабильно. Это означает, что модель меньше ошибалась на тренировочной выборке. Это является удовлетворительным результатом.

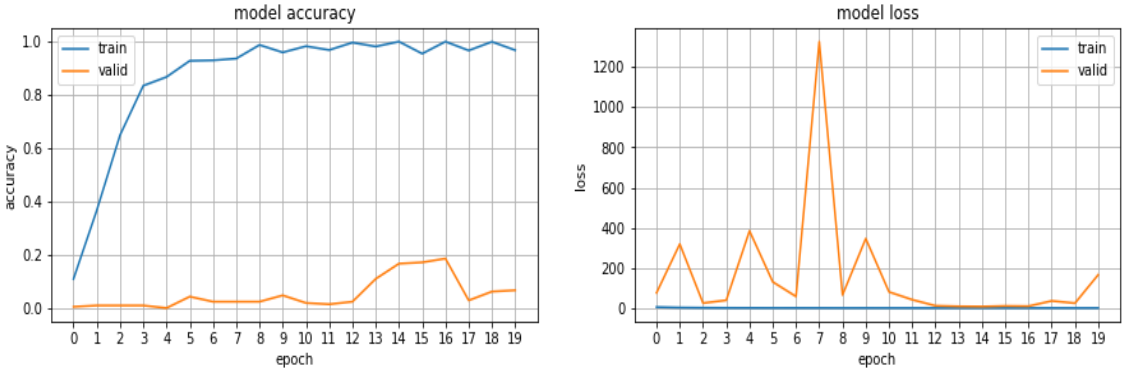


Рисунок 4.30 – Точность и потери для модели ResNet50

Результаты обучения модели глубокого обучения ResNet50 в графическом представлении.

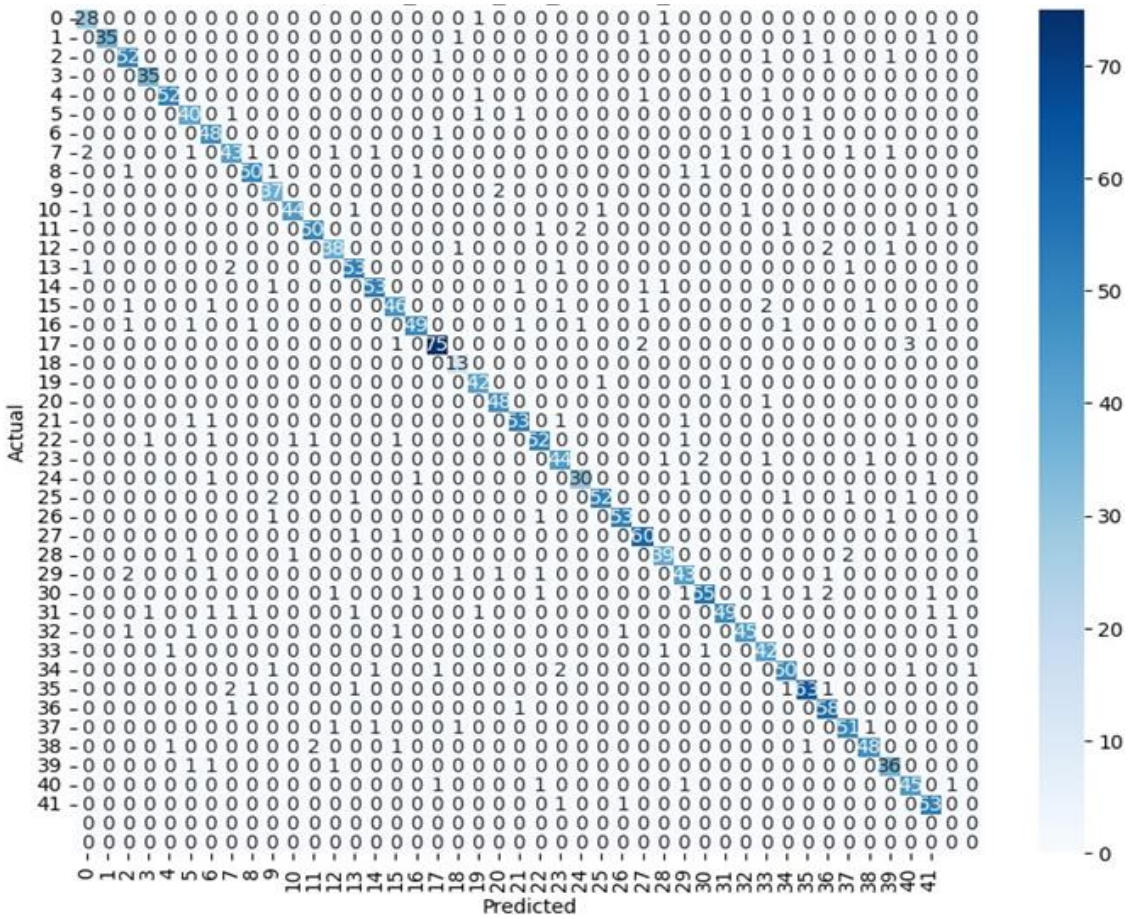


Рисунок 4.31 – Матрица ошибок для модели ResNet

Визуализация матрицы ошибок модели ResNet показана на рисунке 4.31. В некоторых классах модель ошибалась, дав ложно отрицательные прогнозы (False Negative). В целом результат удовлетворительный.

4.2.5 Предложенная Модель гибридной архитектуры на основе сверточных и рекуррентных нейронных сетей

Исследовав и изучив свойства работы всех моделей и методов машинного обучения и нейронных сетей, проведя полный цикл исследовательской работы,

обучая нейронную сеть распознавать жесты, а именно казахский жестовый алфавит, мы пришли к идее и необходимости разработать собственную архитектуру нейронной сети глубокого обучения по принципу работы многослойного перцептрона с комбинированием слоев из рекуррентных нейронных сетей, так как полученные результаты обучения моделей были недостаточно точны в распознавании жестов.

Модель Машина опорных векторов в общей точности распознает казахские жестовые буквы с результатом около 83%, что доказано на основе метрик F1-score, precision, recall, accuracy и confusion matrix. Модель сверточных нейронных сетей распознавала жесты с общей точностью более 90%, но ошибалась при распознавании динамичных жестов, так как в казахском жестовом алфавите существует около 10-динамичных жестов. Модель ошибочно распознавала (False Negative) или не могла распознавать конкретный жест. Некоторые изображения жестов в казахском дактильном алфавите практически одинаково выглядят, но разница заключается в движении рук под разным углом градуса. Поэтому был изучен и применен метод глубокого обучения обработки последовательных данных – сеть рекуррентных нейронных сетей модель LSTM.

Данная модель ранее была предназначена для распознавания речи или синтеза речи на основе работы с временными рядами. Она, как оказалось, подходит для работы с видеопоследовательностями. Наборы данных собирались с помощью записи с веб-камеры. Алгоритм LSTM разделяет данные видео на кадры и преобразовывает в вектора с координатами (ключевые особенности). Так получаются необходимые элементы с пикселей изображения. Это ресурсоемкий процесс, так как на обработку и время выполнения уходит большое количество времени.

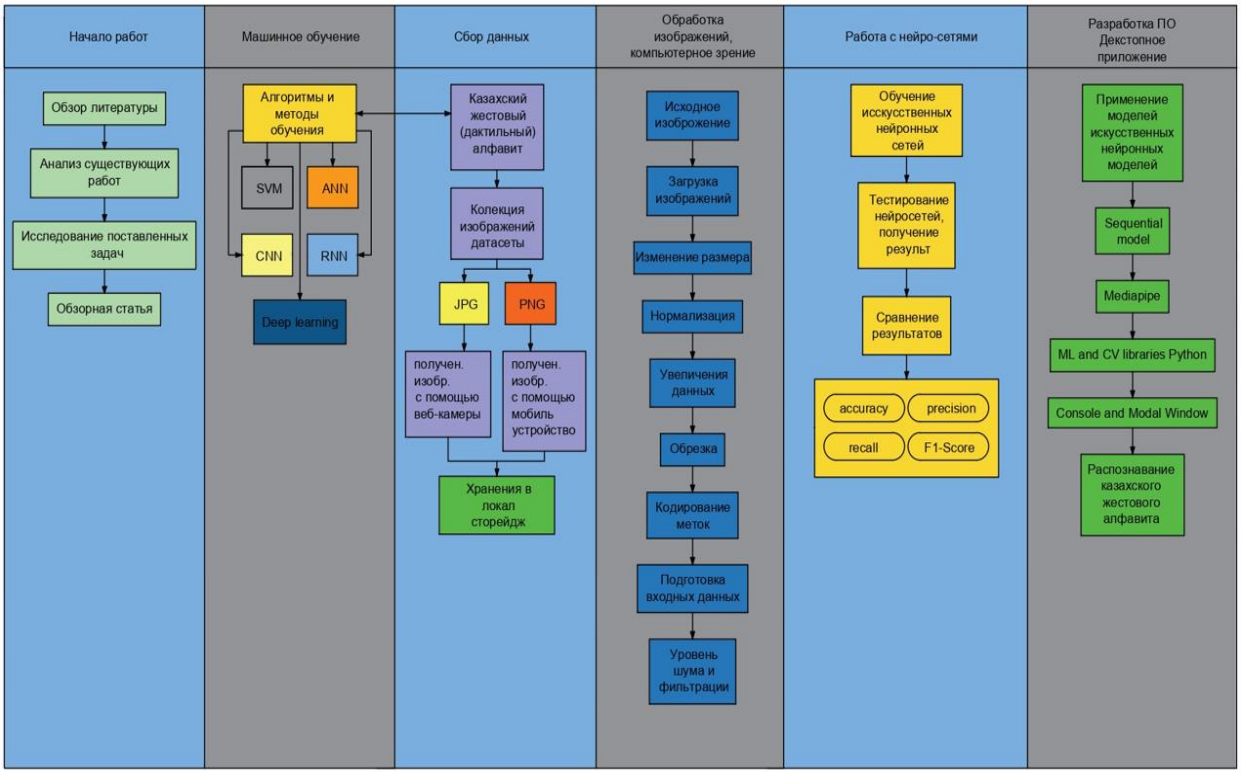


Рисунок 4.32 – Полный цикл исследовательской работы

Также была изучена модель глубокого обучения – YOLO (You Only Look Once), новейшая сеть обнаружения объектов, которая использует регрессионную модель и алгоритм пространственно разделенных ограничивающих рамок (bounding boxes) для обнаружения объектов на изображении. Модель известна своей высокой точностью и скоростью обработки в реальном времени.

Таким образом, проанализировав свойства работы использованных нейросетей, было принято решение скомбинировать данные методы и предложить собственную модель гибридной архитектуры нейросетей глубокого обучения.

Для собственной архитектуры, изучив свойства работы сетей, мы скомбинировали и разработали архитектуру гибридной модели из слоев рекуррентной (LSTM) и сверточной (YOLO) нейронных сетей для распознавания жестов казахского жестового языка.

Для обучения нашей модели мы использовали набор данных жестов казахского жестового языка. Набор данных состоял из жестов рук, снятых на камеру с помощью технических устройств в лабораторных условиях с сохранением видеозаписей. Для каждого жеста мы сделали соответствующие метки. Сбор и разметка набора данных осуществлялись вручную. Перед обучением нашей модели мы предварительно обработали набор данных, преобразовав видеозаписи в последовательность кадров и изменяя их размер до стандартного размера. Мы также применили техники аугментации данных,

такие как случайное вращение, масштабирование и обрезка, чтобы увеличить изменчивость данных и предотвратить переобучение.

Принцип работы нашей нейронной сети строится на следующих слоях. Входной слой определяет количество нейронов, полученных с изображения, после слой признаков отвечает за выделения (извлечение) признаков из входного изображения, затем сохраненные веса поступают в слой LSTM, и далее плотный слой определяет подходящие нейроны в выходной слой, который в свою очередь классифицирует нейроны на заранее определенные классы. На рисунке 4.33 показан принцип работы предложенной архитектуры.

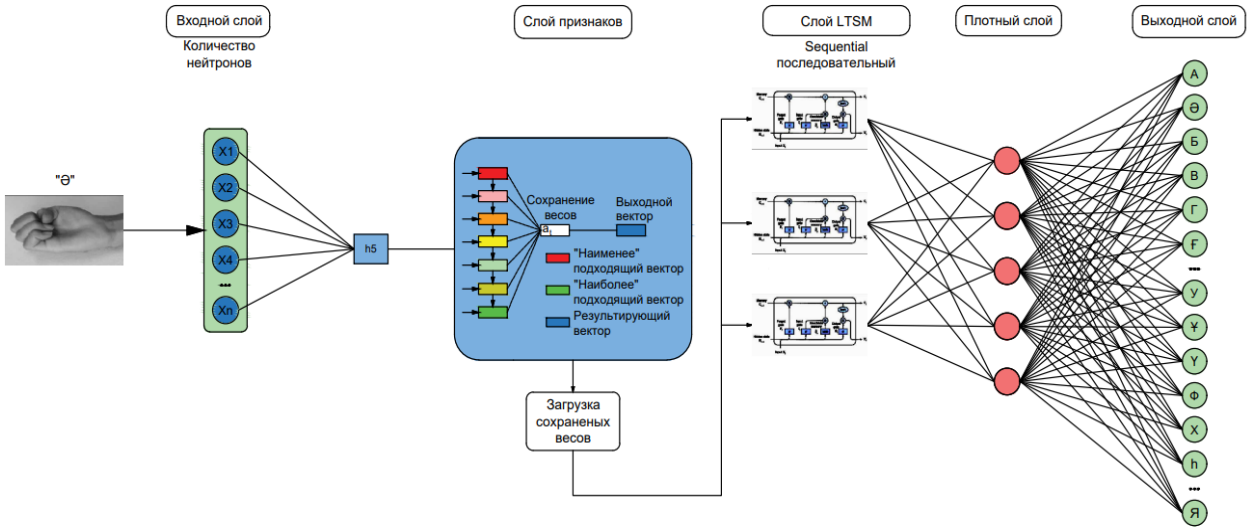


Рисунок 4.33 – Принцип работы предложенной гибридной модели нейронной сети

На рисунке 4.34 показана полносвязная нейронная сеть предложенной модели (гибридной) на рекуррентных и плотных слоях нейронных сетей. Архитектура модели состоит из рекуррентной нейронной сети (LSTM) с 3 слоями, за которой следует плотный слой (DENSE) также с 3 слоями.

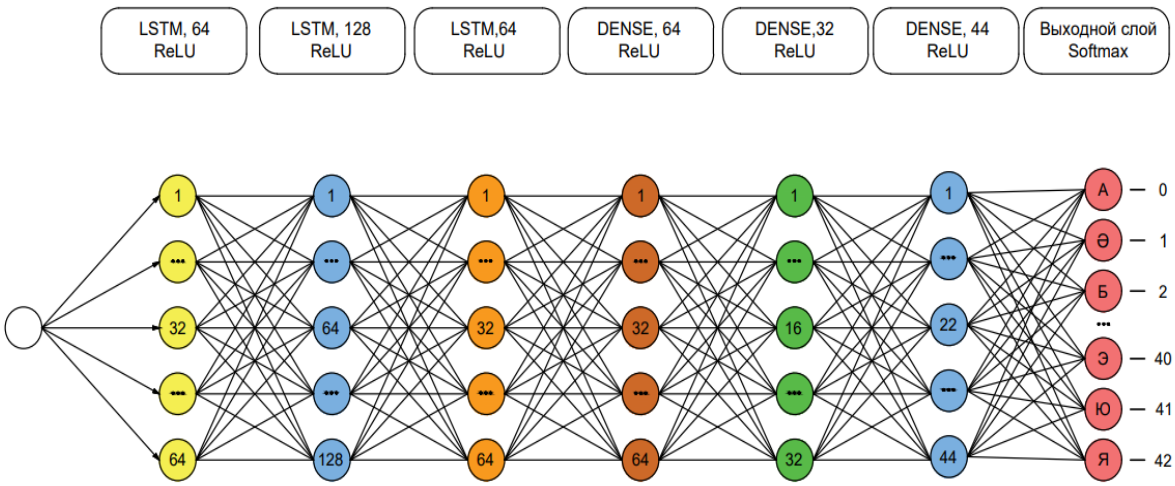


Рисунок 4.34 – Полносвязная нейронная сеть предложенной модели

Для сбора набора данных был написан скрипт на Python, который запускает камеру и с использованием LSTM распознает руку, выстраивая кости в 21 точке, на местах, где рука и пальцы согнуты. Затем, с некоторой задержкой, необходимо показать жест, указывающий на один определенный символ из 42 возможных, повторяя эту процедуру несколько раз и слегка меняя положение руки, чтобы собрать наиболее разнообразный набор данных. После этого скрипт автоматически сохраняет данные кадр за кадром, полученные с камеры, в формате .h5 (hdf5). В нашем случае мы собрали 100 последовательностей из 30 кадров для каждого класса. В результате мы получили по 3000 кадров для каждого класса.

Обучение модели происходило на наборе данных жестов жестового языка, используя размер пакета 64 и скорость обучения 0,001. Также применялся весовой коэффициент step decay (пошаговое затухание) для предотвращения переобучения со значением 0,0005. Для оценки производительности модели использовались стандартные метрики, такие как средняя точность Mean Average Precision (MAP) и пересечение объединения Intersection Over Union (IoU). Мы использовали инструменты визуализации, такие как кривые точности-полноты и матрицы ошибок для анализа производительности модели на различных классах жестов жестового языка. Полученные результаты показали, что модель способна точно обнаруживать жесты жестового языка с высокими значениями MAP и IoU.

Далее перейдем непосредственно к экспериментам и библиотекам машинного обучения, которые мы применили в обучении нейронной сети глубокого обучения. Ниже приведен список необходимых библиотек для работы с машинным обучением и обучением нейросети.

```
import numpy as np
import os
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import keras
from keras.utils import to_categorical
from keras.callbacks import Callback
from keras import backend as K
from keras.optimizers import RMSprop
from tensorflow.keras import models, layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten, Dense, Dropout
from datetime import datetime
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from tqdm import tqdm
import pickle
```

Рисунок 4.35 – Подключение необходимых библиотек Python

Описание каждой подключенной библиотеки:

- numpy, необходима для математических операций, упрощает и ускоряет работу, так как работает на принципах векторных вычислений;
- os и PIL, необходимы для работы с файлами и открытия изображения;
- matplotlib, используется для визуализации изображений и графиков;
- tensorflow.keras, встроенная версия Keras, предоставляемая библиотекой TensorFlow. В данном контексте используется для построения и обучения моделей глубокого обучения;
- datetime, используется для измерения времени выполнения определенных операций, что может быть полезно при оценке производительности моделей;
- sklearn.model_selection train_test_split из библиотеки scikit-learn применяется для разделения данных на обучающий и тестовый наборы, что является важным этапом при оценке производительности модели;
- sklearn.metrics f1_score из scikit-learn используется как метрика для измерения точности модели в задачах классификации;
- tqdm предоставляет индикатор выполнения, что упрощает визуализацию процесса обучения, особенно при работе с большими объемами данных;
- pickle используется для сериализации объектов, что может быть полезно для сохранения обученных моделей или других важных данных.

Ниже приведен фрагмент программного кода.

```
x_data = []
y_data = []
datacount = 0
for j in os.listdir('resized_gray_dataset/'):
    if not j.startswith('.'):
        count = 0
        for k in os.listdir('resized_gray_dataset/' + j + '/'):
            img = Image.open('resized_gray_dataset/' + j + '/' + k).convert('L')
            img = img.resize((320, 120))
            arr = np.array(img)
            x_data.append(arr)
            count = count + 1
        y_values = np.full((count, 1), lookup[j])
        y_data.append(y_values)
        datacount = datacount + count

        print(datacount, '-xxx')
x_data = np.array(x_data, dtype = 'float32')
print(len(y_data))
y_data = np.vstack(y_data)


y_t_data = []

for _ in y_data:
    for __ in _:
        y_t_data.append(__)
y_data = np.array(y_t_data)

print(len(y_data))
```

Рисунок 4.36 – Фрагмент кода на Python

1. Цикл пробегается по каждой папке внутри директории 'resized_gray_dataset/'.
2. Для каждой папки (каждого класса) собирает изображения и их метки:

- загружает изображения, преобразует их в оттенки серого и изменяет размер до 320x120 пикселей;
- преобразует изображения в массивы и добавляет их в список **x_data**;
- создает метки **y_values** в виде массивов с метками классов и добавляет их в список **y_data**;
- подсчитывает количество данных в **datacount**.

3. Конвертирует списки **x_data** и **y_data** в массивы numpy соответствующих типов данных.

4. Выводит количество уникальных классов в **y_data**.

5. Преобразует структуру **y_data** для удобства использования.

6. Выводит количество данных в **y_data**.

Ниже приведен пример изображений, которые использовались для обучения.

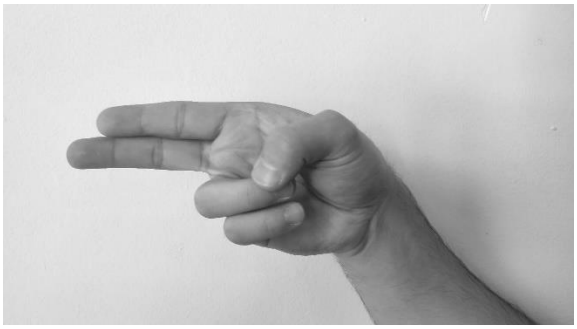


Рисунок 4.37 – Изображение жеста

Следующий этап – это разделение выборки на тренировочную, валидационную и тестовую. На рисунке 4.38 приведено их соотношение. Выборки должны быть стратифицированы, то есть в каждой из них должно быть похожее с оригинальной выборкой распределение целевых классов (букв).

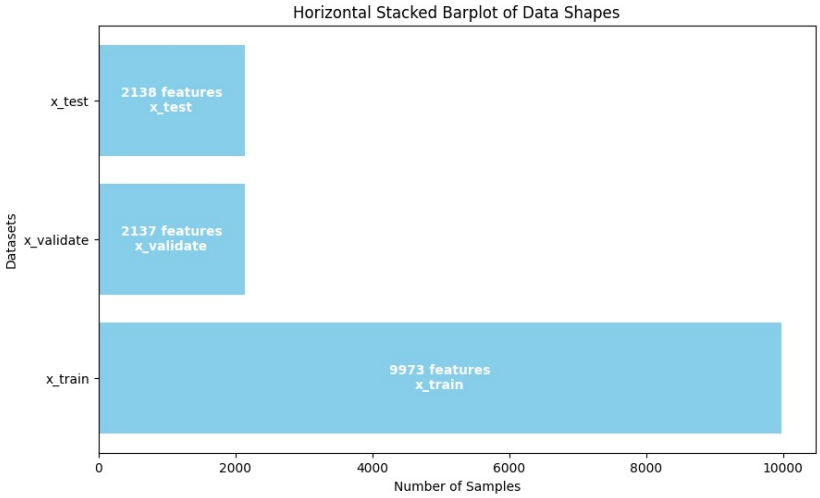


Рисунок 4.38 – Разделение данных на обучаемые, валидационные и тестовые выборки

Далее, после проведения экспериментов, обученных на собственных данных, получили метрики оценки для нашей сверточной нейронной сети, отображенной в таблице 4.6.

Таблица 4.6 - Метрики оценки для модели CNN (Evaluation Metrics):

Class	Precision	Recall	F1 score	Support
Ae(Ә)	0.91	0.62	0.78	11
Blank	0.99	0.91	0.29	5
Gh(Ғ)	0.93	0.64	0.78	11
Hh(Һ)	0.89	0.31	0.47	13
Ii(І)	0.94	0.76	0.89	5
Kk(Қ)	0.95	0.55	0.71	11
Nn(Ң)	0.91	0.92	0.91	13
Oe(Ө)	0.89	0.67	0.80	6
Ue(Ү)	0.94	0.91	0.89	4
Accuracy	0.93			79
Macro Avg	0.90	0.72	0.73	79
Weighted Avg	0.95	0.67	0.73	79

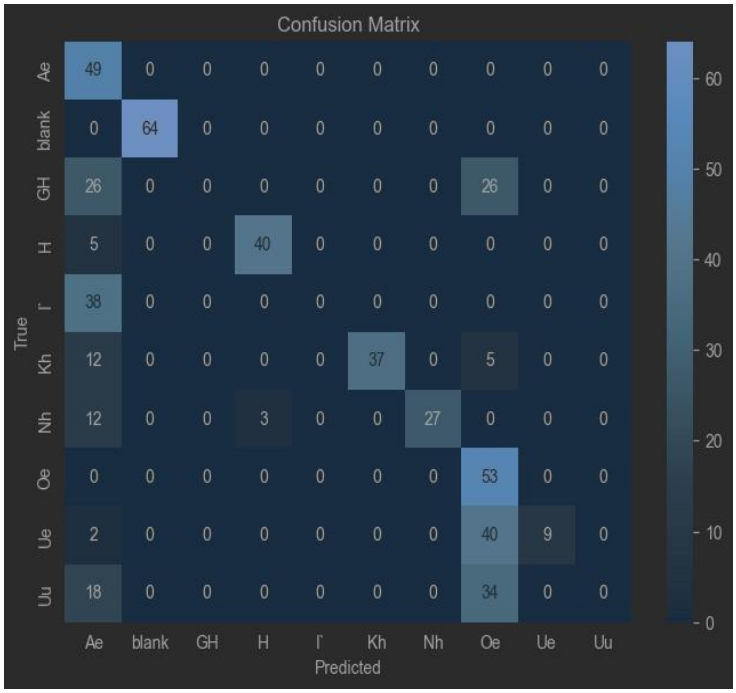


Рисунок 4.39 – Результат матрицы ошибок

Как показано на рисунке 4.39 модель ошибалась при распознавании казахских жестовых букв в некоторых классов.

Далее были проведены эксперименты только по уникальным буквам казахского алфавита, то есть 9 классам (буквам) казахского жестового языка. Затем проводилась сегментация, включающая выделение региона жеста руки из фона с использованием алгоритма пороговой обработки. Полученные изображения затем преобразовывались в оттенки серого, что упрощает

обработку и улучшает контрастность. Изменение размера изображений до стандартного значения было критически важным для эффективного сравнения в процессе обучения. Маркировка датасета с правильной информацией о классах была необходима для того, чтобы модель могла ассоциировать конкретные жесты рук с соответствующими метками. Этот процесс, хотя и занимает время, является ключевым для разработки точной и надежной системы распознавания жестов рук. Из датасета были удалены изображения низкого качества, что привело к сохранению только высококачественных изображений.

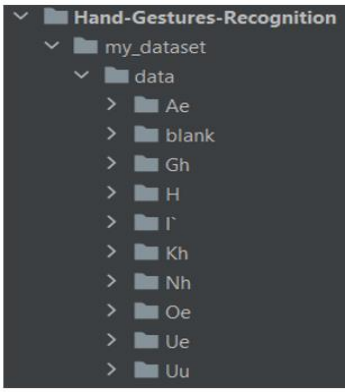


Рисунок 4.40 – Набор данных казахских букв

При применении аугментации данных с использованием ImageDataGenerator данная библиотека генерирует увеличенные версии исходного датасета на лету во время обучения. Увеличенные изображения используются для обучения модели, эффективно увеличивая разнообразие тренировочного датасета. Это помогает модели лучше обобщаться на изменения во входных данных, делая ее более устойчивой к неизвестным примерам. В нашем случае ImageDataGenerator настроен с различными методами аугментации, такими как вращение, сдвиги по ширине и высоте, сдвиг, масштабирование и горизонтальное отражение. По мере обучения модели она не видит одно и то же изображение несколько раз, а сталкивается с различными увеличенными версиями оригинальных изображений в каждой эпохе. Таким образом, физический датасет не изменяется; вместо этого в каждую эпоху обучения модель видит разные варианты оригинальных изображений, что симулирует более крупный и разнообразный датасет. Этот процесс помогает улучшить способность модели справляться с изменениями и хорошо проявлять себя на неизвестных данных.

Обучение и потери Валидации (Training and Validation loss).

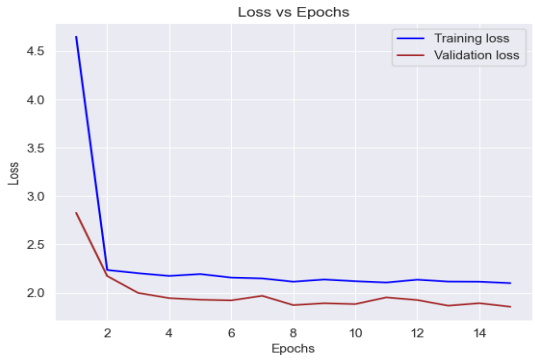


Рисунок 4.41 – Потери и эпохи

Модель начинает с относительно высокой потери на этапе обучения, равной 4,65, что указывает на то, что начальные предсказания значительно отличаются от истинных значений. Однако на протяжении последующих эпох модель продемонстрировала стабильную тенденцию к снижению потерь в процессе обучения. Это снижение говорит о том, что модель успешно извлекает знания из обучающих данных и вносит улучшения.

В отличие от потерь на этапе обучения, потери на валидации для данной модели проявляют более значительные флуктуации. Несмотря на общую тенденцию к уменьшению, заметное расхождение между потерями на этапе обучения и валидации указывает на потенциальную проблему переобучения или недостаточной обобщаемости. Увеличение разрыва между этими двумя потерями указывает на то, что модель может хорошо справляться с обучающими данными, но испытывает трудности в обобщении на неизвестные примеры.

Оценка производительности и сравнение (Performance Evaluation and Comparison).

Матрица ошибок раскрывает производительность модели с точки зрения неверных классификаций. Она показывает, что модель верно предсказывает Класс 0 для всех случаев (63 из 63), указывая на идеальную точность классификации для этого класса. Однако у нее возникают трудности с Классом 1, где она неправильно классифицирует 44 случая как Класс 0, 9 случаев как сам Класс 1 и 8 случаев как Класс 8. Кроме того, она неверно классифицирует 4 случая Класса 2 как Класс 9. Модель хорошо справляется с правильным предсказанием Классов 3, 4, 5, 6, 7, 8 и 9 без неверных классификаций.

На рисунке 4.42 приведена матрица ошибок.

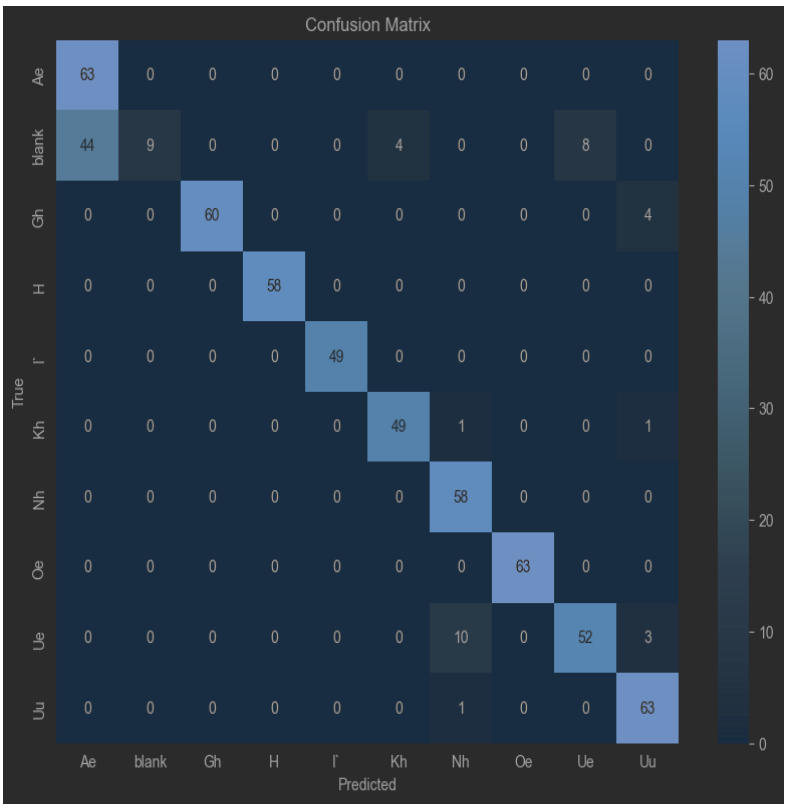


Рисунок 4.42 – Матрица ошибок по 9-классам казахских букв в жестовом алфавите

Далее, мы получаем данные по следующим метрикам:

Точность (Precision). Данная модель последовательно достигает высокой точности для всех классов, причем каждое значение точности равно 1.00, за исключением Класса 7 (0.95). Это указывает на то, что модель редко делает ложноположительные предсказания.

Полнота (Recall). Модель демонстрирует отличные показатели полноты 1.00 для всех классов, за исключением Класса 2, где она немного ниже и составляет 0.97. Это свидетельствует о том, что модель улавливает практически все экземпляры каждого класса.

F1-мера (F1-Score). Значения F1-меры для модели универсально высоки для всех классов, самое низкое из них – 0.98 в Классе 2. Это указывает на сбалансированную производительность с точки зрения точности и полноты для каждого класса.

Точность (Accuracy). Модель достигает впечатляющей общей точности 0.96, что указывает на высокий уровень правильности в его предсказаниях.

Таким образом, расчеты показали, что по точности, полноте и F1-мере для всех классов модель демонстрирует общую высокую точность (0.96).

Macro и weighted средние значения для точности, полноты и F1-меры выше для модели, что указывает на более сбалансированную и точную производительность для всех классов.

Таблица 4.7 – Метрики оценки предложенной модели рекуррентной и сверточной нейронной сети LSTM+CNN (Evaluation Metrics):

Class	Precision	Recall	F1 score	Support
Ae(Ә)	0.99	1.00	1.00	63
blank	0.99	1.00	1.00	65
Gh(Ғ)	0.99	0.97	0.98	64
H(Һ)	0.97	1.00	1.00	58
I`(I)	0.97	1.00	1.00	49
Kh(Қ)	1.00	1.00	1.00	51
Nh(Ң)	0.98	0.98	0.99	58
Oe(Ө)	0.95	0.97	0.98	63
Ue(Ү)	0.96	1.00	1.00	65
Uu(Ұ)	0.98	1.00	1.00	64
Accuracy	0.96			600
Macro Avg	1.00	1.00	1.00	600
Weighted Avg	1.00	0.99	1.00	600

Анализ производительности, согласно данной таблице, показал высокую точность распознавания казахских букв по применённым метрикам.

На рисунке 4.43 приведена матрица ошибок после применения модели для распознавания всех букв (жестов) казахского алфавита (42 класса).

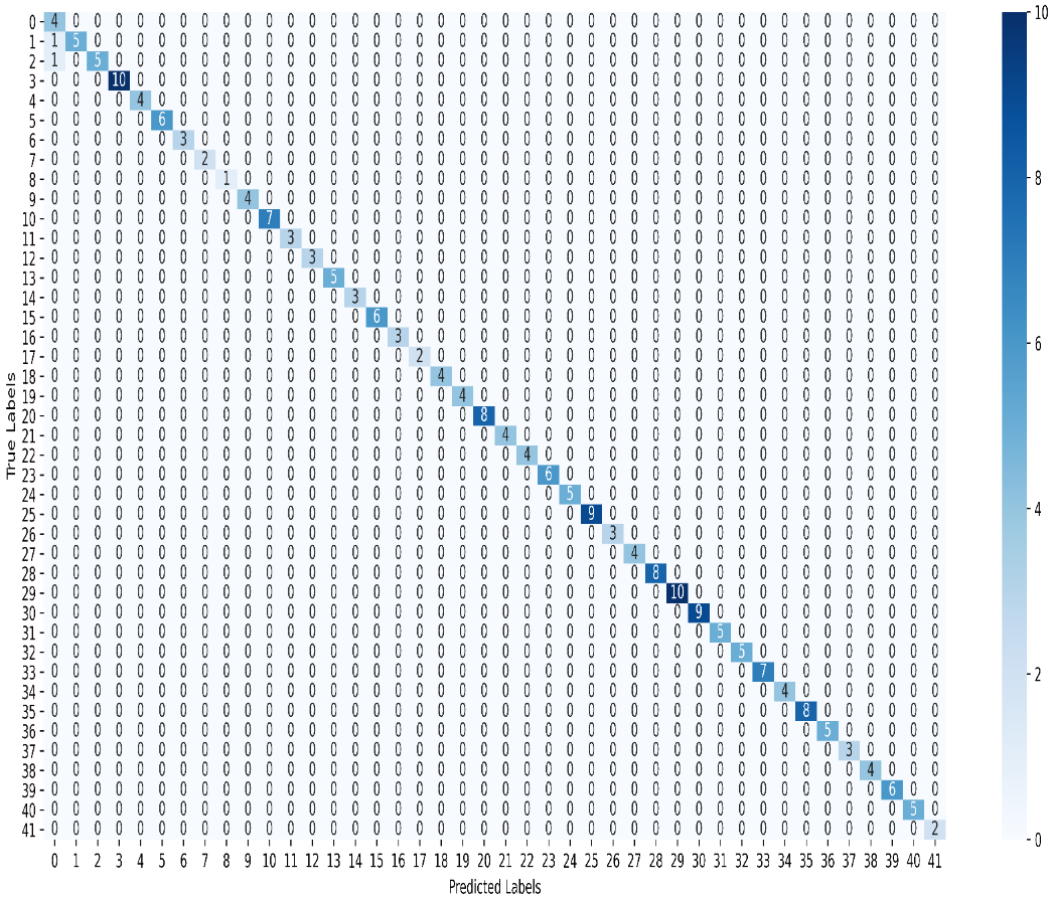


Рисунок 4.43 – Матрица ошибок по 42 классам казахских букв в жестовом алфавите

Далее сравним все полученные результаты всех моделей по следующим метрикам:

- вертикальная ось – accuracy/общая точность;
- горизонтальная ось – precision/правильные предсказания, recall/положительные примеры и f1-score/среднее гармоническое, отображенные снизу вверх.

Результаты сравнения приведены на рисунке 4.44 в виде графика столбцов, стремящихся к 1, то есть к 100% точности предсказания правильных ответов (true positive) по распознаванию жестов казахских букв данных моделей:

Рисунок 4.44 – Сравнение моделей по распознаванию казахских букв

На рисунке 4.45 построен общий график эффективности всех моделей, включая предложенную модель, основанную на гибридной архитектуре с использованием слоев рекуррентных и сверточных нейронных сетей (LSTM + CNN). Точность распознавания казахских жестов более чем 96%.

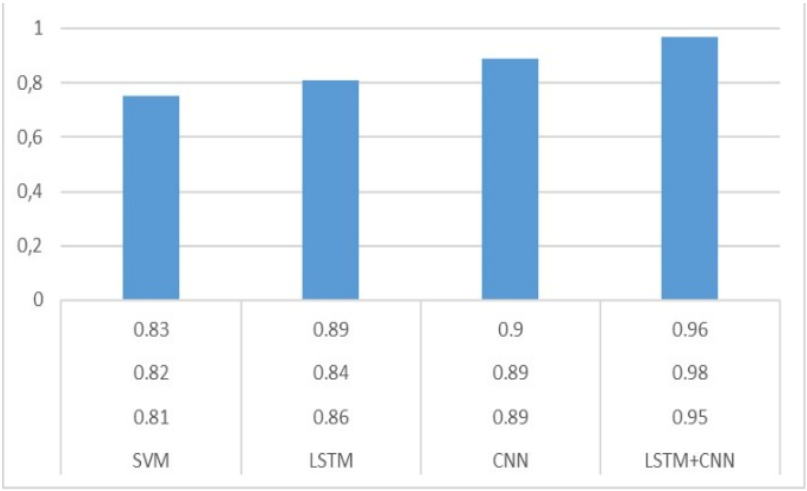


Рисунок 4.45 – Общая точность распознавания моделей

Эксперименты показали, что точность распознавания казахских букв и в целом казахского жестового алфавита доказана вычислениями во время обучения, тестирования и обоснования результатов по примененным метрикам, которые описаны и применены выше, а также визуализацией матрицы ошибок (confusion matrix). Таким образом, можно заключить, что предложенная модель гибридной архитектуры является более высокоточной в сравнении с остальными моделями машинного и глубокого обучения нейронных сетей.

4.3 Сравнительный анализ архитектур глубоких нейронных сетей

Сравнительный анализ таких как моделей LeNet, AlexNet, ResNet использовали пакетную нормализацию, оптимизатор rmsprop и категориальную перекрестную энтропию в качестве функции потерь. Результаты эксперимента были представлены в таблице 4.7 для сопоставления. Обучение продолжалось 20 эпох с целью достижения сходимости оптимальной модели CNN для распознавания жестов рук. Таблица 4.7 показывает сравнение производительности с точки зрения матрицы точности, потерь и штрафов. Полученные результаты показывают схожие и сопоставимые характеристики для всех моделей. Это объясняется наличием разнообразных изображений в обучающих данных и эффективностью CNN для данного типа задач. Из результатов обучения LeNet, AlexNet и ResNet на протяжении 20 эпох видно, что LeNet показал лучшие результаты со схожей скоростью обучения, точностью и потерями. LeNet, при меньшем количестве параметров, проявил себя более эффективно. AlexNet и ResNet50 проявили признаки переобучения, видимые в уменьшении точности и увеличении потерь на проверочном наборе данных. Анализ матрицы ошибок выявил некоторые ошибки CNN при распознавании сложных жестов, таких как "Ә" и "Б". Эти ошибки обусловлены спецификой проявления данных жестов при движении вверх, что представляет сложность для систем распознавания изображений. Однако CNN справилась с этой проблемой, улучшив точность. Важным направлением для будущих исследований может быть интеграция обученных нейронных сетей в мобильные устройства. Учитывая, что мобильные устройства являются ключевым средством использования систем распознавания языка жестов в повседневной жизни, вопрос о размере и ресурсах моделей на мобильных устройствах становится актуальным. Возможные решения включают в себя техники сжатия для нейронных сетей или использование легковесных архитектур, таких как LeNet.

Таблица 4.8 - Сравнение производительности моделей глубокого обучения с точки зрения матрицы ошибок, потерь и штрафов

Model	Total params (M)	Accuracy	Loss	Penalty
LeNet	1,07	0.990	0.0453	-3.017
AlexNet	35,5	0.990	0.1343	-3.021
ResNet50	23,7	0,642	1,8544	-3.130

Результаты показали, что сверточные нейронные сети эффективно справляются с задачами распознавания объектов на изображениях, в данном случае жестах. Они широко используются в компьютерном зрении, особенно благодаря своей способности выявлять границы объектов, текстуры, грани и углы с использованием локальных шаблонов.

Архитектура сверточной нейронной сети, состоящая из нескольких слоев, извлекает все более абстрактные детали изображения по мере продвижения вглубь сети. Одной из ключевых характеристик сверточной нейронной сети является инвариантность к трансформациям, что позволяет им распознавать объекты на изображениях независимо от масштаба, поворота и сдвига.

Сравнительный анализ моделей Lenet, ResNet, AlexNet с разными функциями активации (ReLU, Softmax, Tanh, Sigmoid) и разными оптимизаторами градиентного спуска (ADAM, SGD) для задачи распознавания жестов в казахском языке. В качестве метрик использовались точность и F1-мера, учитывающие ошибки 1 и 2 рода.

Lenet показал хорошие результаты, особенно при использовании ReLU и Tanh. Softmax как функция активации на выходном слое дала худшие результаты.

ResNet произвел впечатляющие результаты, демонстрируя высокую точность. Лучшие результаты были достигнуты с использованием ReLU и Tanh.

AlexNet – несмотря на простоту архитектуры, результаты были неудовлетворительными, особенно по сравнению с Lenet и ResNet. Softmax также показал себя неэффективным.

Общие наблюдения:

Softmax на выходном слое сети дал худшие результаты, возможно, из-за своей вероятностной природы.

ResNet требовало значительно больше времени обучения, чем Lenet и AlexNet. Выбор функции активации оказывает значительное влияние на результаты. ReLU и Tanh часто показывали хорошие результаты. Внутреннее строение сетей, особенно ResNet, представляет собой сложную структуру с множеством параметров. Матрица ошибок предназначена для визуализации результатов для лучших конфигураций гиперпараметров. Время обучения для AlexNet было высоким, а результаты — неудовлетворительными.

Таким образом, построение и выбор архитектуры, функций активации и оптимизаторов важен для достижения хороших результатов в задачах распознавания жестов на изображениях.

4.4 Разработка программного обеспечения и распознавание в реальном времени

Распознавание жестов в реальном времени — это процесс, который позволяет компьютерам и другим устройствам распознавать и интерпретировать жесты, сделанные человеком, в режиме реального времени. Этот процесс позволяет людям взаимодействовать с компьютерами, используя жесты рук или другие движения тела, вместо традиционных устройств ввода, таких как клавиатура и мышь. Для распознавания жестов в реальном времени компьютер использует различные методы. Например, один из наиболее распространенных методов — это оптическое распознавание, при котором камера или другое устройство оптической передачи изображения снимает жесты и отправляет их в компьютер для анализа.

Разработанное программное обеспечение предлагает пользователю распознавать жесты рук людей с ограниченными возможностями. Основная цель программного обеспечения — предложить потребителю взаимодействовать и построить коммуникацию с людьми, которые не могут говорить вследствие каких-либо нарушений. Взаимодействие системы распознавания жестов рук с пользователями показано на рисунке 4.46. Система получила на вход жест руки и на выходе выполнила определенное действие, связанное с этим жестом. Алгоритм работы системы приведен ниже.

Начало: запуск камеры

Шаг 1. Снимите изображение или видео с камеры.

Шаг 2: Извлеките область интереса.

Шаг 3. Функция сопоставления.

Шаг 4. Функция перевода.

Шаг 5: Отобразите результат

На первом этапе пользователь предоставляет доступ к камере, и камера начинает работать. Камера фиксирует положение руки, далее происходит выделение интересующей области. В нашем случае это та область, которая находится в положении руки. Далее жест руки обрабатывается по написанному алгоритму и выдает результат. После осуществляется проверка результата на точность. Если показанный жест соответствует набору данных, результат будет показан на экране.

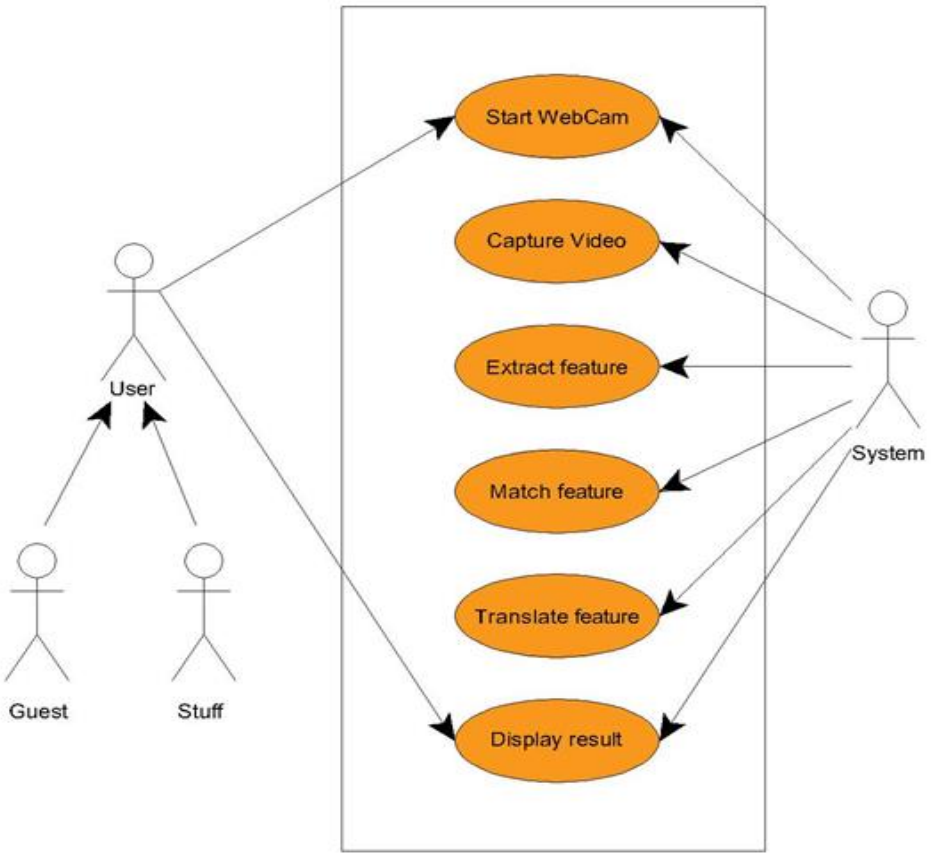


Рисунок 4.46 - Диаграмма вариантов использования

На рисунке 4.47 представлена ER диаграмма предлагаемой системы. Схема состоит из 9 таблиц, в которых показаны взаимосвязи всех сущностей, объектов и их поведение внутри системы. Есть Потребитель и Администратор, которые унаследованы от Пользователя и имеют определенные функции, доступные только им. Например, Администратор может добавлять и удалять данные из базы данных и, соответственно, обучать модели, а пользователь может использовать систему только для распознавания жестов рук.

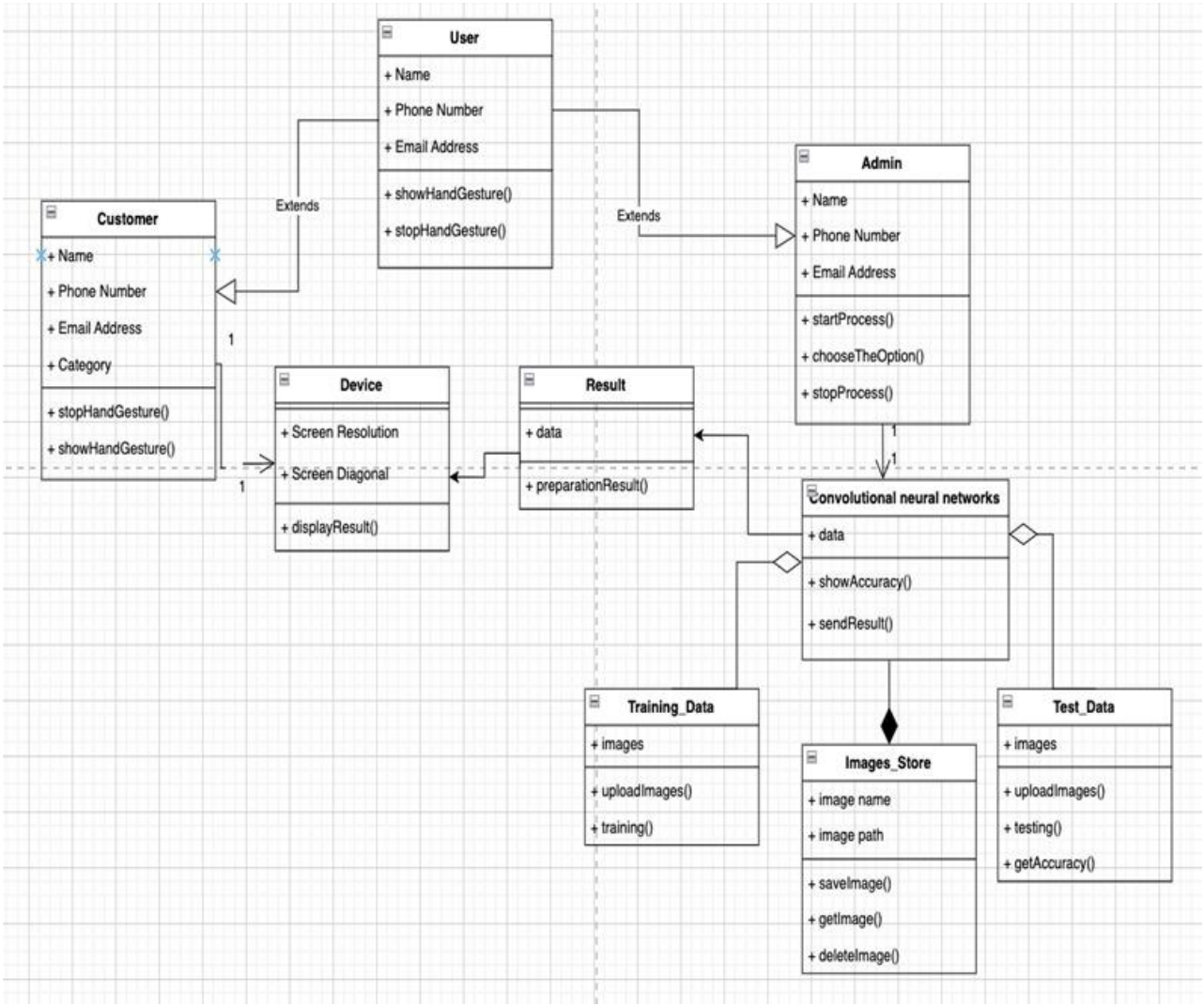


Рисунок 4.47 – ER диаграмма

Система включает в себя множество этапов вычислений, результатом которых будет классификация жеста. Стоит отметить, что систему необходимо обучить, а затем проверить точность обученной модели. База данных хранилища изображений – это оборудование, с которым пользователь будет взаимодействовать с системой, а также устройство, на котором установлено приложение или имеется доступ к нему.

Python – это язык программирования высокого уровня, широко используемый для машинного обучения и научных вычислений. Он предоставляет богатый набор библиотек и инструментов для манипулирования, анализа и визуализации данных. Django – популярный веб-фреймворк, который позволяет разработчикам быстро и эффективно создавать веб-приложения. Обеспечивает чистый и прагматичный дизайн, подчеркивающий возможность повторного использования и модульность. Keras – это API нейронной сети высокого уровня, написанный на Python. Предоставляет простой и интуитивно понятный интерфейс для создания и обучения моделей глубокого обучения, включая CNN. Scikit-learn – это библиотека Python, предоставляющая инструменты для машинного обучения и анализа данных. Включает в себя различные алгоритмы классификации, регрессии, кластеризации и уменьшения

размерности. Numpy – фундаментальная библиотека для научных вычислений на Python. Предоставляет мощные структуры данных для многомерных массивов и матриц, а также большую коллекцию математических функций для операций с массивами. Matplotlib – это библиотека Python для визуализации данных. Предоставляет множество функций построения графиков для создания статических, анимированных и интерактивных визуализаций. SciPy – библиотека для научных вычислений на Python. Включает в себя модули оптимизации, интеграции, интерполяции, обработки сигналов и многое другое. Наконец, Pandas – это библиотека для манипулирования и анализа данных на Python. Предоставляет структуры данных для эффективной обработки и анализа данных, а также множество функций для очистки, агрегирования и преобразования данных.

Далее, пошагово проделываем все этапы, сначала загружаем нашу обученную модель для интеграции в программное обеспечение (рисунок 4.48).

```python
model.save('actions_test_(44 letters 30 FRAMES 100 seq REFINED 31 epochs).h5')
```

Python

Рисунок 4.48 – Интеграция модели

На рисунке 4.49 приведена собственная гибридная модель, состоящая из слоев рекуррентной (LSTM) и сверточной (Dense) нейронных сетей.

```python
# +
model = Sequential()
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,126)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(44, activation='softmax'))
model.load_weights('actions_test_(44_letters_30_FRAMES_100_seq_REFINED_31_epochs).h5')
```

Python

Рисунок 4.49 – Слои гибридной модели

Ниже приведены некоторые фрагменты программного кода разработанного приложения.

```python
# +
from PIL import Image, ImageDraw, ImageFont
font_size = 60
font = ImageFont.truetype('arial.ttf', font_size)
text_color = (255, 255, 255)
bg_color = (255, 0, 0)
def prob_viz(res, actions, input_frame):
    output_frame = input_frame.copy()
    max_idx = np.argmax(res)
    letter = actions[max_idx]
    img_pil = Image.new('RGB', (font_size, font_size), color=bg_color)
    draw = ImageDraw.Draw(img_pil)
    draw.text((0, 0), letter, font=font, fill=text_color)
    img_np = np.array(img_pil)
    x_offset = 10
    y_offset = output_frame.shape[0] - font_size - 10
    output_frame[y_offset:y_offset+font_size, x_offset:x_offset+font_size] = img_np
    return output_frame
```

Python

Рисунок 4.50 – Подключение отрисовки Image, ImageDraw, ImageFont

```python
# +
import cv2
import numpy as np
import mediapipe as mp
# 1. New detection variables
sequence = []
sentence = []
predictions = []
threshold = 0.5
cap = cv2.VideoCapture(0)

def blur_region(image, x, y, w, h, blur_intensity=31):
    sub_img = image[y:y+h, x:x+w]
    sub_img = cv2.GaussianBlur(sub_img, (blur_intensity, blur_intensity), 30)
    image[y:y+sub_img.shape[0], x:x+sub_img.shape[1]] = sub_img
    return image


# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()

        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)
```

Рисунок 4.51 – Использование алгоритма mediapipe

```
        # Draw landmarks
        draw_styled_landmarks(image, results)

        # 2. Prediction logic
        keypoints = extract_keypoints(results)
        sequence.append(keypoints)
        sequence = sequence[-30:]

        if len(sequence) == 30:
            res = model.predict(np.expand_dims(sequence, axis=0))[0]
            predicted_action = actions[np.argmax(res)]
            print(predicted_action)
            predictions.append(np.argmax(res))

            # 3. Viz logic
            if np.unique(predictions[-10:])[0]==np.argmax(res):
                if res[np.argmax(res)] > threshold:

                    if len(sentence) > 0:
                        if actions[np.argmax(res)] != sentence[-1]:
                            sentence.append(actions[np.argmax(res)])
                    else:
                        sentence.append(actions[np.argmax(res)])

            if len(sentence) > 5:
                sentence = sentence[-5:]
```

Рисунок 4.52 – Применение Draw landmarks, prediction logic и visualization

```
                if results.pose_landmarks:
                    hand_landmarks = []
                    if results.left_hand_landmarks:
                        hand_landmarks.extend(results.left_hand_landmarks.landmark)
                    if results.right_hand_landmarks:
                        hand_landmarks.extend(results.right_hand_landmarks.landmark)

                    if hand_landmarks:  # Check if hand_landmarks is not empty
                        min_x = min([lm.x for lm in hand_landmarks])
                        max_x = max([lm.x for lm in hand_landmarks])
                        min_y = min([lm.y for lm in hand_landmarks])
                        max_y = max([lm.y for lm in hand_landmarks])

                        h, w, _ = image.shape
                        min_x, max_x, min_y, max_y = int(min_x * w), int(max_x * w), int(min_y * h), int(max_y * h)

                        padding = 20  # adjust the padding if needed
                        image = blur_region(image, min_x - padding, min_y - padding, max_x - min_x + 2 * padding, max_y - min_y + 2 * padding

                # Viz probabilities
                image = prob_viz(res, actions, image)

            # Show to screen
            cv2.imshow('OpenCV Feed', image)

            # Break gracefully
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

    cap.release()
    cv2.destroyAllWindows()
```

Рисунок 4.53 – Завершения программы и запуск камеры

Ниже приводится демонстрация работы приложения.

Рисунок 4.54 – Распознавание буквы «h»

Изображение выше демонстрирует результаты распознавания на собственной гибридной модели на слоях LSTM и CNN.

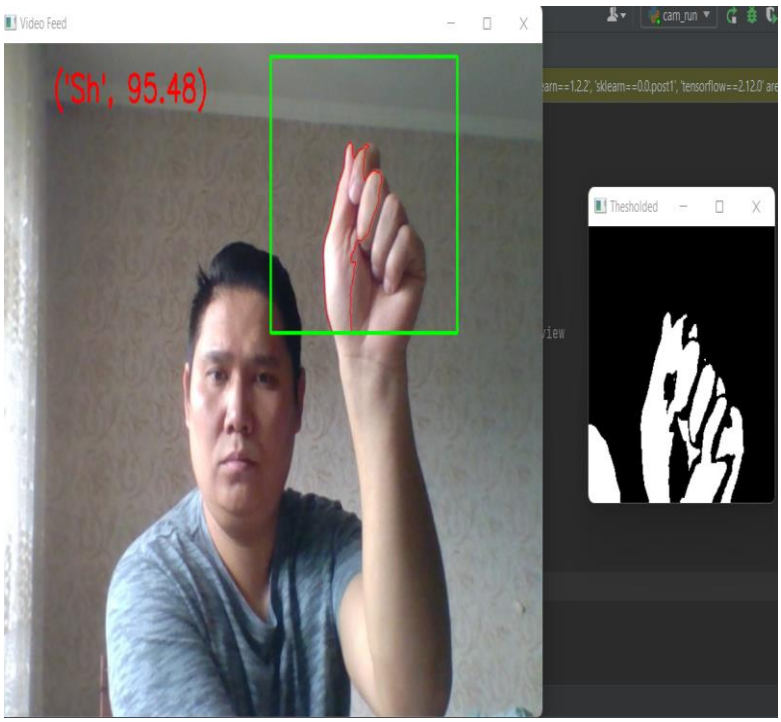


Рисунок 4.55 – Распознавание буквы «Ш»

Результаты распознавания сверточной нейронной сетью (CNN) приведены на рисунке 4.55.

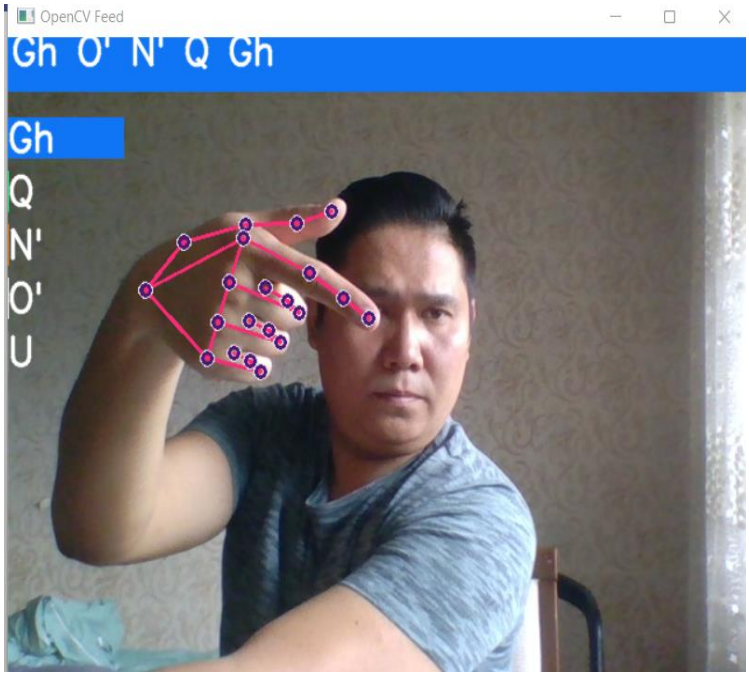


Рисунок 4.56 – Распознавание буквы «F»

На рисунке 4.56 показаны результаты распознавания с использованием рекуррентной нейронной сети Long-Short Term Memory.

4.5 Выводы по разделу

В четвертом разделе описываются проведенные эксперименты по распознаванию казахского жестового алфавита. Обучены модели нейронных сетей глубокого обучения и получены результаты обучения с применением метрик для тестирования моделей глубокого обучения. Доказаны точность распознавания данных моделей на основе метрик точности (precision), полноты (recall), F1-score (мера) и поддержки (support) для каждого класса, а также общая точность (accuracy) и средние оценки (average). Построена собственная гибридная модель глубокого обучения. Предложенная модель использует слои рекуррентной и сверточной нейронной сети. Разработано программное обеспечение, которое распознает жесты в режиме реального времени.

ЗАКЛЮЧЕНИЕ

Данная диссертационная работа была посвящена разработке и применению высокоточных методов распознавания образов. В качестве образов были использованы изображения с жестами рук казахского жестового алфавита.

В ходе исследования был проведен качественный анализ работ, посвященных распознаванию образов, а именно жестов рук. Каждая из работ использует свой собственный эмпирический подход для получения результатов в индивидуальных и специфических/узконаправленных условиях работы.

В процессе исследования были изучены разные архитектуры машинного и глубокого обучения сверточных и рекуррентных нейронных сетей с использованием моделей CNN (обучение с учителем) и модели LSTM (глубокое обучение) для обработки последовательных данных. Эти методы по метрикам F1-score, accuracy, precision, recall и матрицы ошибок (confusion matrix) показали точные результаты обучения и тестирования.

Была разработана собственная гибридная модель, которая решает сложную многоклассовую задачу классификации. Способ комбинирования слоев моделей LSTM и CNN показал эффективность предлагаемого подхода при распознавании казахских букв. Разработанная искусственная нейронная сеть распознает жесты с вероятностью более 95% согласно примененным метрикам, и уверенно их классифицирует.

Тем не менее все еще остаются актуальными такие проблемы как потребность в большом объеме обработанных и размеченных данных. Эталонных изображений жестов рук, полученных с казахского жестового языка, а также сурдопереводов недостаточно. Также существует проблема необходимости больших вычислительных ресурсах, которые помогут справиться с возрастанием данных и усложнением структур моделей.

Таким образом, проведенные теоретические, практические и экспериментальные исследования представляют собой полный цикл завершенной диссертационной работы с полученными результатами и их апробацией на конференциях и в организациях.

В диссертации были получены следующие результаты:

- Собраны изображения жестов рук для обучения моделей.
- Обучены искусственные нейронные сети для распознавания жестов казахского жестового алфавита.
- Разработана и построена архитектура гибридной модели на основе рекуррентных и сверточных нейронных сетей для глубокого обучения.
- Получены результаты распознавания жестов моделями глубокого обучения и сделан сравнительный анализ моделей нейронных сетей с визуализацией.
- Разработано программное обеспечение для распознавания жестов казахского жестового алфавита.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Амиргалиев Е.Д. Книга: Теория распознавания образов и кластерного анализа: учебник. Рекомендовано Министерством образования и науки Республики Казахстан. – Алматы, 2012.

2 Bekarystankyzy A., Mamyrbayev O., Mendes M., Zhumazhanov B., Fazylzhanova A. Automatic Speech Recognition Improvement for Kazakh Language with Enhanced Language Model // Communications in Computer and Information Science. - 2023. – Vol. 1863. – P. 538–545/

3 Кудрявцев, В. Б. Распознавание образов: учебное пособие для вузов / В. Б. Кудрявцев, Э. Э. Гасанов, А. С. Подколзин. — 2-е изд. — Москва: Издательство Юрайт, 2023. — 107 с. — (Высшее образование). — ISBN 978-5-534-15338-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/520462.

4 Панин, С. Д. Теория принятия решений и распознавание образов. Курс лекций: методические указания / С. Д. Панин. — Москва: МГТУ им. Н. Э. Баумана, 2017. — 239 с. — ISBN 978-5-7038-4482-3. — Текст: электронный // Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/103548.

5 Чабан Л.Н. Теория и алгоритмы распознавания образов: учебное пособие. – М. МИИГАиК (Московский государственный университет геодезии картографии)., 2004. – 102 с.

6 Нагоев З.В., Гуртаева И.А. Базовые элементы когнитивной модели и механизма восприятия речи на основе мультиагентного рекурсивного интеллекта, Информатика. Вычислительная техника, управление // Известия Кабардино-Балкарского научного центра РАН. - 2019. - №3(89). – С. 30-38.

7 Aitulen A.D., Mukhanov S.B., Khassenova G.I. Data science and machine learning «Face Recognition Through Various Facial Expressions», г. Алматы 2019. – 106 p.

8 Aitulen A.D., Mukhanov S.B. Обработка, идентификация и распознавание личности методом Виолы-Джонса // Вестник КазНИТУ. – 2019. - №6. – С. 16-24.

9 Uskenbayeva R.K., Mukhanov S.B. Contour analysis of external images // ACM International Conference Proceeding Series, г. Алматы 2020. – P. 3410811.

10 Mukhanov S.B., Uskenbayeva R.K. Pattern Recognition with Using Effective Algorithms and Methods of Computer Vision Library // Advances in Intelligent Systems and Computing. – 2020. - №1. – P. 31-37.

11 Kenshimov C., Mukhanov S., Merembayev T., Yedilkhan D. A Comparison of Convolutional Neural Networks for Kazakh Sign Language Recognition Eastern-European // Journal of Enterprise Technologies. – 2021. – Vol. 5, №2-113. - P. 44-54.

12 Муханов С.Б., Ли А.С., Жексенов Д.Б., Евдокимов Д.Д., Амиргалиев Е.Н., Калжигитов Н.К., Кеншимов Ш. Сравнительный анализ нейросетевых

моделей для методов распознавания жестов рук // Вестник НИА РК. Информационно-коммуникационные технологии. – 2023. - №2(88). – С. 15-27.

13 Mukhanov Samat, Uskenbayeva Raissa, Im Cho Young, Dauren Kabyl, Les Nurzhan, Amangeldi Maqsat. Gesture Recognition of Machine Learning and Convolutional Neural Network Methods for Kazakh Sign Language // Вестник Scientific Journal of Astana IT University. - 2023. – Vol. 15. – P. 16-27.

14 Mukhanov S.B., Tursunov S.A., Izteleuov N.E., Tazabekov A. Data Science and Machine Learning // Changing Kazakhstan Society Using Smart Technologies. – 2019. - №1. – P. 11-24.

15 Slyamkhan S.M., Yembergenov A.A., Bordousov N.S., Mukhanov S.B. Data Science and Machine Learning // Game Application with Machine Learning Elements. – 2019. - №1. – P. 16-28.

16 Mukhanov S.B., Aldanazar A.A., Uatbayeva A.M., Alimbekov A.Ye., Marat G.S Competitive learning in neural networks // International Journal of Information and Communication Technologies. – 2020. - Vol. 1, issue 3. - P. 70.

17 Mukhanov S.B., Alimbekov A.Ye., Marat G.S., Uatbayeva A.M., Aldanazar A.A. Automation of staff recruitment and assessment // International Journal of Information and Communication Technologies. – 2020. - Vol. 1, issue 3. - P. 117.

18 https://startpack.ru/application/youscan-smm 12.08.2022.

19 Hartley R.I., Zisserman A. Multiple View Geometry in Computer Vision. - 2nd edition. - Cambridge University Press, 2004. – 150 p.

20 https://netology.ru/blog/raspoznavanie-izobrajeniy 02.08.2022.

21 https://svkz.satu.kz/p62298227-nomerok-lite.html 11.05.2022.

22 https://videogorod.ru/blog/raspoznavanie-avtomobilnykh-nomerov-v-videonablyudenii/ 17.11.2022.

23 Куракин А. В. Распознавание жестов ладони с помощью непрерывного скелета // Труды 15-й всероссийской конференции «Математические методы распознавания образов». – М.: МАКС Пресс, 2011. - С. 428-431.

24 Gudmundsson S.A., Sveinsson J.R., Pardas M. et al. Model-Based Hand Gesture Tracking in ToF Image Sequences // 6th International Conference on Articulated motion and deformable objects. – AMDO, 2010. – 122 p.

25 Li L., Yeh H.H. Hand gesture recognition by 3D motion trails // Computer Vision and Image Understanding. - 2013. – Vol. 117, №4. – P. 384-395.

26 Местецкий Л.М. Непрерывная морфология бинарных изображений: фигуры, скелеты, циркуляры. - Физматлит, 2009. – 134 с.

27 Zhang H., Shan C., Gao W. Kinect-based hand gesture recognition for intelligent home care application // IEEE Transactions on Industrial Informatics. – 2013. - Vol. 9, №1. – P. 128-137.

28 Bazarevsky, V., Fan, Zh. (2019). On-device, real-time hand tracking with mediapipe. Google AI Blog. Available at: https:// ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html.

29 Wang Y.Q. An Analysis of Viola-Jones Face Detection Algorithm // IPOL Journal. - 2013. - №1. – P. 16-34.

30 Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection // 9th European Conference on Computer Vision. – 2006. - Vol. 1. - P. 430–443.

31 Shan C., Gao W. 3D human face reconstruction from monocular image and hand-held camera // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2013. - Vol. 35, №3. – P. 625-639.

32 Ko N., Yang H., Song H., Choi J. Real-time sign language recognition using RGB-D sensors for efficient human computer interaction // Sensors. - 2015. - Vol. 15, №8. – P. 20480-20500.

33 Viola P., Jones M.J. Robust real-time face detection // International Journal of Computer Vision. – 2004. - Vol. 57, №2. - P. 137–154.

34 Wu T.F., Lin C.J. Probability estimates for multi-class classification by pairwise coupling // The Journal of Machine Learning Research. - 2004. - №5. – P. 975-1005.

35 Машинное зрение и цифровая обработка изображений // Журнал «СТА» «Современные технологии автоматизации» http://www.cta.ru/cms/f/435961.pdf 30.10.2016.

36 Explaining AdaBoost http://rob.schapire.net/ http://rob.schapire.net/papers/explaining-adaboost.pdf 02.01.2017.

37 Cao Z., Simon T., Wei S.E., Sheikh Y. Realtime multi-person 2D pose estimation using part affinity fields // CVPR. IEEE Xplore. - 2016. - №1. – P. 7291-7299.

38 Cai Zheng et. al. A novel convolutional neural network based on improved inception modules for hand gesture recognition // Neurocomputing. - 2019. - №1. – P. 358.

39 Anna Vidyanova. In the USA, they are interested in the development of Kazakhs for the deaf // Capital. – 2022. - №1. – P. 15-27. |

40 Zhang Y., Cao C., Cheng J., Lu H. EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition. - 2018 // IEEE Transactions on Multimedia https://ieeexplore.ieee.org/document/8299578 27.03.2023.

41 Pisharady P.K., Trivedi M.M. Driver hand activity analysis in unconstrained environments // IEEE Transactions on Intelligent Transportation Systems. - 2012. - Vol. 13, №3. – P. 1419-1428.

42 Lai K., Yanushkevich S.N. CNN+RNN depth and skeleton based dynamic hand gesture recognition // In 24th International Conference on Pattern Recognition (ICPR). - IEEE, 2018 https://doi.org/10.1109/ICPR.2018.8545718 02.04.2023.

43 Gupta A., Kembhavi A., Davis L.S. Observing human-object interactions: Using spatial and functional compatibility for recognition // CVPR. IEEE Xplore. - 2009. - №1. – P. 198.

44 Xia L., Chen C.C., Aggarwal J.K. View invariant human action recognition using histograms of 3D joints // Computer Vision—ECCV. – 2012. - №1. – P. 20-34.

45 Fathi A., Mori G. Action recognition by learning mid-level motion features //CVPR 1-8. IEEE Xplore. - 2008. - №1. – P. 15-27.

46 Yin L., Wei X., Sun Y., Wang J., Rosato M. A 3D hand gesture recognition system using dynamic time warping and gesture normalization // Pattern Recognition. - 2013. - №46(6). – P. 1494-1506.

47 Grest D., Franz M., Mörwald T. Fast and robust 3D pose estimation of rigid objects from depth images. IEEE/RSJ // International Conference on Intelligent Robots and Systems. - 2012. - №1. – P. 1033-1039.

48 Kamath S., Vasan A. A comprehensive review of gesture recognition techniques // Egyptian Informatics Journal. - 2015. – Vol. 16, №2. – P. 97-111.

49 Li Y., Ye J., Ji Q., Shan Y. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network // Pattern Recognition. - 2013. - Vol. 46, №7. – P. 1943-1953.

50 Neff M., Koller D. Sign language recognition by combining vision and gyroscope data // Proceedings of the International Conference on Multimodal Interfaces. - 2003. - №1. – P. 68-73.

51 Li Y., Guan H. An efficient method for hand gesture recognition using depth and color features // Journal of Real-Time Image Processing. - 2018. - Vol. 15, №1. – P. 55-67.

52 Hu J., Zheng W.S., Lai J., Zhang J. Real-time hand gesture recognition for monocular natural image sequences // IEEE Transactions on Image Processing. - 2012. - Vol. 22, №5. – P. 1849-1858.

53 C. Zhu, J. Y. Yang, Z. P. Shao, and C. P. Liu, "Vision Based Hand Gesture Recognition Using 3D Shape Context," IEEE/CAA J. Autom. Sinica, vol. 8, no. 9, pp. 1600-1613, Sep. 2021. doi: 10.1109/JAS.2019.1911534.

54 López M.B., González P., Artés-Rodríguez A. A survey of deep learning techniques for RGB-D action recognition // Expert Systems with Applications. - 2017. - №72. – P. 327-336.

55 Cippitelli E., Spinsante S., Gambi E., Uccheddu F. A survey on vision-based hand gesture recognition // Sensors. - 2014. - Vol. 14, №10. – P. 2059-2089.

56 Ahlström D., Kristensen A.S. A survey of vision-based methods for action representation, segmentation and recognition // Computer Vision and Image Understanding. - 2014. - Vol. 125. – P. 3-16.

57 Jung H.Y., Nixon M.S. Hand gesture recognition using multi-scale colour space local features and bag-of-features // Computer Vision and Image Understanding. - 2015. - Vol. 135. – P. 1-13.

58 Chen X., Corso J.J. Action detection by implicit intentional motion clustering // CVPR. - 2015. - №1. – P. 4243-4252.

59 Cippitelli E., Spinsante S., Gambi E., Uccheddu F. Real time hand gesture recognition exploiting visual and depth data // IEEE/RSJ International Conference on Intelligent Robots and Systems. - 2014. - №1. – P. 3991-3996.

60 Ziaei A., Fathy M. A survey on human action and gesture recognition // Journal of Ambient Intelligence and Humanized Computing. - 2018. - Vol. 9, №1. – P. 75-96.

61 Grauman K., Shakhnarovich G. Spatial pyramid matching // Foundations and Trends® in Computer Graphics and Vision. - 2011. - Vol. 5, №3. – P. 239-274.

62 Zhu L., Shao L., Lin L., Wang X., Li X. A survey of pose recovery from 2D and 3D line drawings // IEEE Transactions on Cybernetics. - 2015. - Vol. 45, №3. – P. 469-482.

63 Deng J., Dong W., Socher R., Li L.J., Li K., Fei-Fei L. ImageNet: A large-scale hierarchical image database // IEEE Conference on Computer Vision and Pattern Recognition. - 2009. - №1. – P. 248-255.

64 Dalal N., Triggs B. Histograms of oriented gradients for human detection // CVPR. - 2005. - №1. – P. 886-893.

65 Shan C., Gong S., McOwan P.W. Robust facial expression recognition using local binary patterns // Image and Vision Computing. - 2008. - Vol. 27, №12. – P. 1787-1798.

66 Kjellström H., Kragic D. Learning visual object categories by combining features // ICML. - 2008. - №1. – P. 464-471.

67 Breiman L. Random forests // Machine Learning. - 2001. - Vol. 45, №1. – P. 5-32.

68 Yamato J., Ohya J., Ishii K. Recognizing human action in time-sequential images using hidden Markov model // CVPR. - 1992. - №1. – P. 379-385.

69 Chen C., Jafari R. Toward adaptive real-time gesture recognition for multimedia applications // IEEE Transactions on Multimedia. - 2007. - Vol. 9, №7. – P. 1402-1411.

70 Zhang L., Li Y., Qi G.J. A survey of network embedding-based approaches for natural language processing // Journal of Computer Science and Technology. - 2017. Vol. 32, №4. – P. 662-682.

71 Yan Y., Pollefeys M. The three Rs of computer vision-based hand gesture interface: Recognition, regression and new interfaces // Computer Vision and Image Understanding. - 2017. - Vol. 149. – P. 252-268.

72 Fang F., Chai X., Fu C.W. Exploiting semantic cues for multimodal video summarization // IEEE Transactions on Circuits and Systems for Video Technology. - 2017. - Vol. 29, №1. – P. 292-306.

73 Bobick A.F., Davis J.W. The recognition of human movement using temporal templates // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2001. - Vol. 23, №3. – P. 257-267.

74 Liu X., Xu C., Wang D. A survey on 3D hand gesture recognition // Signal Processing: Image Communication. - 2019. - Vol. 72. – P. 139-152.

75 Jiang M., Jia X., Zhang Y., Nevatia R. A linear programming approach for global non-rigid registration // CVPR. - 2012. - №1. – P. 2248-2255.

76 Bobick A.F., Davis J.W. The recognition of human movement using temporal templates // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2001. - Vol. 23, №3. – P. 257-267.

77 Cippitelli E., Gambi E., Spinsante S. Vision-based hand gesture recognition for humanoid robotic platforms: A review // Sensors. - 2012. - Vol. 12, №5. – P. 5309-5338.

78 Chang C.C., Lin C.J. LIBSVM: A library for support vector machines // ACM Transactions on Intelligent Systems and Technology (TIST). - 2011. - Vol. 2, №3. – P. 27.

79 Maji P., Berg A.C. Max-margin additive classifiers for detection // IEEE Transactions on Pattern Analysis and Machine Intelligence. - 2009. - Vol. 25, №5. – P. 1545-1557.

80 Nalpantidis L., Gasteratos A. Real-time and markerless hand gesture recognition for human–robot interaction // Robotics and Autonomous Systems. - 2013. - Vol. 61, №4. – P. 398-409.

81 Tang J., Deng L., Liu Z., Yang Y., Zhou M. Adaptive feature selection for gesture recognition using genetic algorithm // Pattern Recognition. - 2008. - Vol. 41, №1. – P. 259-268.

82 Lowe D.G. Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. - 2004. - Vol. 60, №2. – P. 91-110.

83 Poppe R. Vision-based human motion analysis: An overview // Computer Vision and Image Understanding. - 2007. - Vol. 108, №1-2. - P. 4-18.

84 Машинное зрение и цифровая обработка изображений // Журнал «СТА» «Современные технологии автоматизации» http://www.cta.ru/cms/f/435961.pdf 30.10.2016.

85 Обнаружение и локализация лица на изображении. Информационно-коммуникационные технологии в образовании http://ict.informika.ru/ft/002403/num2face.pdf 20.12.2016.

86 Explaining AdaBoost http://rob.schapire.net/ http://rob.schapire.net/papers/explaining-adaboost.pdf 02.01.2017.

87 Татаренков Д.А. Анализ методов обнаружения лиц на изображении // Молодой ученый. - 2015. - №4(84). - С. 270.

88 Местецкий Л.М. Математические методы распознавания образов. – М.: МГУ; ВМиК, 2002–2004. - С. 42–44.

89 Demircioglu, B., Bulbul, G., & Kose, H. (2016). Turkish Sign Language recognition with Leap Motion. 2016 24th Signal Processing and Communication Application Conference (SIU). doi:10.1109/siu.2016.7495809.

90 Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: a machine learning approach to corner detection" in IEEE Trans // Pattern Analysis and Machine Intelligence. – 2010. – Vol. 32. - P. 105-119.

91 Liukai Xu, Keqin Zhang, Genke Yang, Jian Chu. Gesture recognition using dual-stream CNN based on fusion of sEMG energy kernel phase portrait and IMU amplitude image // Biomedical Signal Processing and Control. – 2022. – Vol. 73. – P. 103364.

92 Yuanguo Zhou, Shan Shui, Yijun Cai, Chengying Chen, Yingshi Chen, Reza Abdi-Ghaleh. An improved all-optical diffractive deep neural network with less parameters for gesture recognition // Journal of Visual Communication and Image Representation. – 2023. – Vol. 90. – P. 103688.

93 Baiju Yan, Peng Wang, Lidong Du, Xianxiang Chen, Zhen Fang, YirongWu, “mmGesture: Semi-supervised gesture recognition system using mmWave radar”. – 2023. – Vol. 213, part B. – P. 119042.

94 https://yvision.kz/post/virtualnaya-realnost-v-pomoshch-izucheniyu-kazahskogo-yazyka-zhestov-871546.

95 Nuwan Munasinghe. Dynamic Hand Gesture Recognition Using Computer Vision and Neural Networks // IEEE 3rd International Conference for Convergence in Technology (I2CT). - 2018. https://ieeexplore.ieee.org/document/8529335 05.03.2023.

96 Pavlovic, V. I., Sharma, R., & Huang, T. S. (1997). Visual interpretation of hand gestures for human-computer interaction: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7), 677–695. doi:10.1109/34.598226.

97 Madeo, R. C. B., Lima, C. A. M., & Peres, S. M. (2013). Gesture unit segmentation using support vector machines. Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC ’13. doi:10.1145/2480362.2480373.

98 Goutsu, Y., Takano, W., & Nakamura, Y. (2015). Gesture recognition using hybrid generative-discriminative approach with Fisher Vector. 2015 IEEE International Conference on Robotics and Automation (ICRA). doi:10.1109/icra.2015.7139614.

99 Mestetskiy L. Skeleton representation based on composite Bezier curves // VISAPP 2010: Proceedings of the 5th International Conference on Computer Vision Theory and Applications. Vol. 1. INSTICC Press, 2010.

100 Местетцкий Л. М. Непрерывная морфология бинарных изображений: Фигуры, Скелеты, Окружности. Издательство Физматлит, 2009.

101 В. И. Игошин, Математическая логика и теория алгоритмов: учебное пособие для студентов высшего учебных заведений / 2-е издание, стер. – М.: Издательский центр «Академия», 2008. 448с.

102 Nahla Majdoub Bhiri, Safa Ameur, Ihsen Alouani, Mohamed Ali Mahjoub, Anouar Ben Khalifa. Hand gesture recognition with focus on leap motion: An overview, real world challenges and future directions // Expert Systems with Applications. – 2023. – Vol. 226. – P. 120-125.

103 Jaya Prakash Sahoo, Suraj Prakash Sahoo, Samit Ari, Sarat Kumar Patra, "DeReFNet: Dual-stream Dense Residual Fusion Network for static hand gesture recognition" // Displays. – 2023. – Vol. 77. – P. 102388.

104 Guoxiang Tong, Yueyang Li, Haoyu Zhang, Naixue Xiong. A Fine-grained Channel State Information-based Deep Learning System for Dynamic Gesture Recognition // Information Sciences. - 2023. – Vol. 636. – P. 118912.

105 Laura-Bianca Bilius, Ştefan-Gheorghe Pentiuc, Radu-Daniel Vatavu. TIGER: A Tucker-based instrument for gesture recognition with inertial sensors // Pattern Recognition Letters. – 2023. – Vol. 165. – P. 84-90.

106 Deniz Mengu, Yi Luo, Yair Rivenson, and Aydogan Ozcan. Analysis of Diffractive Optical Neural Networks and Their Integration with Electronic Neural Networks”, University of California, 2022, https://ieeexplore.ieee.org/document/8732486 19.12.2022.

107 Pinzón-arenas J.O., Jiménez-moreno R., Herrera-benavides J.E. Convolutional neural network for hand gesture recognition using 8 different emg signals // 2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA) IEEE. - Bucaramanga; Colombia, 2019. - P. 1-5.

108 Dhawan A., Honrao V. Implementation of Hand Detection based Techniques for Human Computer Interaction // Int. J. Comput. Appl. - 2013. - Vol. 72, №17. - P. 6–13.

109 Yeo H.S., Lee B.G., Lim H. Hand tracking and gesture recognition system for human-computer interaction using low-cost hardware // Multimed. Tools Appl. - 2013. https://link.springer.com/article/10.1007/s11042-013-1501-1 01.11.2022.

110 Okan Kopuklu, Ahmet Gunduz, Neslihan Kose, Gerhard Rigoll. Real-time Hand Gesture Detection and Classification Using Convolutional Neural Networks". - Institute for Human Machine Communication, 2019 https://www.researchgate.net/publication/331134940_Real-time_Hand_Gesture_Detection_and_Classification_Using_Convolutional_Neural_Networks 03.11.2022.

111 Jaya Prakash Sahoo, Allam Jaya Prakash, Paweł Pławiak. Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network". - National Institute of Technology, 2022 https://www.researchgate.net/publication/357914251_Real-Time_Hand_Gesture_Recognition_Using_Fine-Tuned_Convolutional_Neural_Network 13.12.2022.

112 Whoi-Yul Kim, Seunghyeok Shin. Skeleton-based Dynamic Hand Gesture Recognition Using a Part-based GRU-RNN for Gesture-based Interface". - IEEE Access, 2020 https://doi.org/10.1109/ACCESS.2020.2980128 09.01.2023.

113 Chunyong Ma, Shengsheng Zhang, Anni Wang, Yongyang Qi, Ge Chen "Skeleton-Based Dynamic Hand Gesture Recognition Using an Enhanced Network with One-Shot Learning". - College of Information Science and Engineering, 2020 https://doi.org/10.3390/app10113680 16.01.2023.

114 Van Houdt, Greg & Mosquera, Carlos & Nápoles, Gonzalo. A Review on the Long Short-Term Memory Model // Artificial Intelligence Review. – 2020. - №1 https://www.researchgate.net/publication/340493274_A_Review_on_the_Long_Short-Term_Memory_Model 20.02.2023.

115 Dinh-Son Tran, Ngoc-Huynh Ho, Hyung-Jeong Yang, Eu-Tteum Baek, Soo-Hyung Kim and Gueesang Lee. Real-Time Hand Gesture Spotting and Recognition Using RGB-D // Camera and 3D Convolutional Neural Network. - 2020. https://doi.org/10.3390/app10020722 14.02.2023.

116 Alejandro T.O., Juan J.T., Juan C.T., Alejandro P., Alexandro L.G., Rui A.C. LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals. - 2022 https://doi.org/10.3390/app12199700 19.02.2023.

117 Rehman Muneeb & Ahmed, Fawad & Khan, Muhammad & Tariq, Usman & Alfouzan, Faisal & Alzahrani, Nouf & Ahmad, Jawad. Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks. Computers // Materials and Continua. - - 2021

https://www.researchgate.net/publication/354463749_Dynamic_Hand_Gesture_Recognition_Using_3D-CNN_and_LSTM_Networks 27.11.2022.

118 Obaid F., Babadi A., Yoosofan A. Hand Gesture Recognition in Video Sequences Using Deep Convolutional and Recurrent Neural Networks. Applied computer systems. - 2020 https://doi.org/10.2478/acss-2020-0007 17.03.2023.

119 Xinpeng Zhang, Yifan Wang, and Yuxin Chen. Dynamic Hand Gesture Recognition Using 3D CNN and LSTM with FSM Context-Aware Model // IEEE Access. - 2021. - Vol. 9. - P. 102785-102796.

120 Köpüklü O., Kose N., Rigoll G. Motion fused frames: Data level fusion strategy for hand gesture recognition. – 2018 https://arxiv.org/abs/1804.07187 18.01.2023.

121 Zhang Y., Cao C., Cheng J., Lu H. EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition // IEEE Transactions on Multimedia. – 2018. - №1 https://ieeexplore.ieee.org/document/8299578 27.03.2023.

122 Lai K., Yanushkevich S.N. CNN+RNN depth and skeleton based dynamic hand gesture recognition // in 24th International Conference on Pattern Recognition (ICPR). - IEEE, 2018 https://doi.org/10.1109/ICPR.2018.8545718 02.04.2023.

123 Shi W., Cao J., Zhang Q., Li Y., Xu, L. Edge computing: Vision and challenges // IEEE internet of things journal. - 2016. – Vol. 3, №5. - P. 637-646.

124 Wong B.P., Kerkez B. Real-time environmental sensor data: An application to water quality using web services // Environmental Modelling & Software. - 2016. - Vol. 84. - P. 505-517.

125 Granell C., Havlik D., Schade S., Sabeur Z., Delaney C., Pielorz J., Mon J. L. Future Internet technologies for environmental applications // Environmental Modelling & Software. - 2016. - Vol. 78. - P. 1-15.

126 Alvarez M.A., Lawrence N.D. Computationally efficient convolved multiple output Gaussian processes // The Journal of Machine Learning Research. - 2011. - Vol. 12. - P. 1459-1500.

127 Futoma J., Hariharan S., Heller K. Learning to detect sepsis with a multitask Gaussian process RNN classifier // International conference on machine learning. - PMLR. - 2017. - №1. - P. 1174-1182.

128 Gelman A., Hill J. Data analysis using regression multilevel/hierarchical models. - Cambridge university press, 2006. – 122 p.

129 Li Y., Du N., Bengio S. Time-dependent representation for neural event sequence prediction. - 2017.

130 Lipton Z.C., Kale D., Wetzel R. Directly modeling missing data in sequences with rnns: Improved classification of clinical time series // Machine learning for healthcare conference. – PMLR, 2016. - C. 253-270.

131 Raissi M., Perdikaris P., Karniadakis G.E. Multistep neural networks for data driven discovery of nonlinear dynamical systems, https://doi.org/10.48550/arXiv.1801.01236, 2018. – 122 p.

132 Abadi M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Zheng X. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. https://doi.org/10.48550/arXiv.1603.04467, 2016. – 111 p.

133 Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lerer A. Automatic differentiation in pytorch. – California, USA, 2017. – 108 p.

134 Merembayev T., Kurmangaliyev D., Bekbauov B., Amanbek Y. A Comparison of machine learning algorithms in predicting lithofacies: Case studies from Norway and Kazakhstan // Energies. - 2021. - Vol. 14, №7. - P. 1896.

135 Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 284–294, Melbourne, Australia. Association for Computational Linguistics.

136 Howard R.A. Dynamic programming and Markov processes - 1960. Kijima M. Markov processes for stochastic modeling. - Springer, 2013. – 122 p.

137 Graves A. Generating sequences with recurrent neural networks. University of Toronto, 2013. – 114 p.

138 Irsoy O., Cardie C. Opinion mining with deep recurrent neural networks // Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). - 2014. - P. 720-728.

139 Chung J., Gulcehre C., Cho K., Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS 2014 Workshop on Deep Learning, December 2014. – 107 p.

140 Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. - 2012. - Vol. 25. – P. 18-27.

141 Mikolov T., Karafiát M., Burget L., Cernocký J., Khudanpur S. Recurrent neural network based language model // Interspeech. – Makuhari, 2010. - Vol. 2. - P. 1045-1048.

142 Zeng N., Zhang H., Song B., Liu W., Li Y., Dobaie A. M. Facial expression recognition via learning deep sparse autoencoders // Neurocomputing. - 2018. - Vol. 273. - P. 643-649.

143 Shuai Lv, Zhijun Li, Fellow I.E., Jin Huang Peng Shi, Fellow IEEE. A Novel Interval Type-2 Fuzzy Classifier Based on Explainable Neural Network for Surface Electromyogram Gesture Recognition // IEEE Transactions on human-machine systems. - 2023. - Vol. 53, №6. – P. 955.

144 П.С. Александров, книга “Курс аналитической геометрии и линейной алгебры”, учебник классическая учебная литература по математике 2-е издание., стер. – СПб.: Издательство «Лань», 2009. – 521с. ISBN 978-5-8114-0908-2.

145 В.И. Игошин, учебное пособие математическая логика и теория алгоритмов – 2-е изд., стер. – М.: Издательский центр «Академия», 2008 – 448 с., ISBN 978-5-7695-4593-1.

146 Learning Data Science by Sam Lau, Joseph Gonzaltz, Deborah Nolan, Released September 2023, Publisher(s): O'Reilly Media, Inc.,ISBN: 9781098113001.

147 Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter, by Wes McKinney, Sep 20, 2022.

148 “Математический анализ”, Киркинский А.С., Москва, Академический Проект 2006, УДК 517(075.8). ББК, К432.

149 Хайкин, Саймон, Нейронные сети: полный курс, 2-е издание: Первое с англ. -М.: Издательский дом “Вильясм”, 2006. – 1104с.:ил. – Парал.тит.анг., ISBN 5-8459-0890-6 (рус).

150 Численные методы, Н.С. Бахвалов, Н.П. Жидком, Г.М. Кобельников. – 7-е издание – М.: БИНОМ. Лаборатория знаний, 2011. – 636с.: ил. – (Классический университетский учебник). ISBN 978-5-9963-0449-3.

151 Quoc V., Jiquan N., Adam C., Abhik L. Bobby P. A. Y. On optimization methods for deep learning // Proceedings of the Twenty–Eighth International Conference on Machine Learning. – 2011.

152 Kingma D., Ba J. Adam A: A Method for Stochastic Optimization. ArXiv

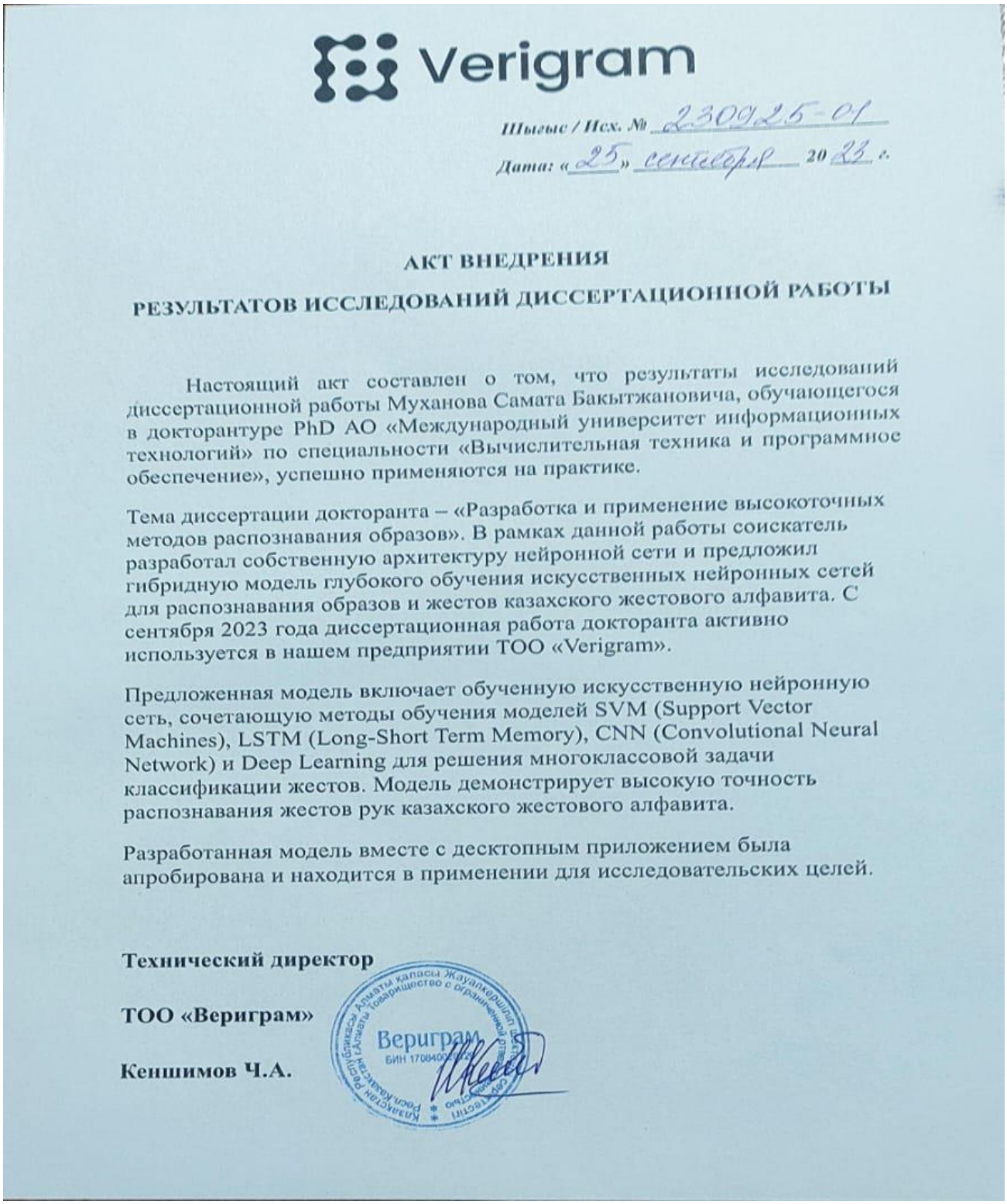
153 Duchi J., Hazan E., Singer Y. Adaptive subgradient methods for online learning and stochastic optimization // The Journal of Machine Learning Research. – 2011. – Vol.12. – P. 2121–2159.

154 Ye Y., Fan H., Li Y., Liu X., Zhang H. Deep neural network methods for solving forward and inverse problems of time fractional diffusion equations with conformable derivative // Neurocomputing. – 2022. – Vol. 509. – P. 177–192.

155 Le Q. V., Ngiam J., Coates A. On optimization methods for deep learning // Proceedings of the Twenty–Eighth International Conference on Machine Learning. – 2011.

ПРИЛОЖЕНИЕ А

Акт внедрения



Verigram

Шығыс / Исх. № 230925-01

Дата: «25» сентября 2023 г.

АКТ ВНЕДРЕНИЯ

РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЙ ДИССЕРТАЦИОННОЙ РАБОТЫ

Настоящий акт составлен о том, что результаты исследований диссертационной работы Муханова Самата Бакытжановича, обучающегося в докторантуре PhD АО «Международный университет информационных технологий» по специальности «Вычислительная техника и программное обеспечение», успешно применяются на практике.

Тема диссертации докторанта – «Разработка и применение высокоточных методов распознавания образов». В рамках данной работы соискатель разработал собственную архитектуру нейронной сети и предложил гибридную модель глубокого обучения искусственных нейронных сетей для распознавания образов и жестов казахского жестового алфавита. С сентября 2023 года диссертационная работа докторанта активно используется в нашем предприятии ТОО «Verigram».

Предложенная модель включает обученную искусственную нейронную сеть, сочетающую методы обучения моделей SVM (Support Vector Machines), LSTM (Long-Short Term Memory), CNN (Convolutional Neural Network) и Deep Learning для решения многоклассовой задачи классификации жестов. Модель демонстрирует высокую точность распознавания жестов рук казахского жестового алфавита.

Разработанная модель вместе с десктопным приложением была апробирована и находится в применении для исследовательских целей.

Технический директор

ТОО «Вериграм»

Кеншимов Ч.А.

Вериграм

БИН 170840

ПРИЛОЖЕНИЕ Б

Акт внедрения

ЖК SMART EDU
Қазақстан, Алматы,
Крамского к-сі, 30 үйі, 1 қенсе
БСН: 971222300992
Тел.: +7 700 992 04 79

ИП SMART EDU
Казахстан, Алматы,
ул. Крамского, дом 30, офис 1
БИН: 971222300992
Тел.: +7 700 992 04 79

№ 027 «1» 09 2023 г.

АКТ ВНЕДРЕНИЯ

РЕЗУЛЬТАТОВ ИССЛЕДОВАНИЙ ДИССЕРТАЦИОННОЙ РАБОТЫ

Настоящий акт составлен об использовании и внедрении результатов исследований диссертационной работы Муханова Самата Бакытжановича обучающийся в PhD докторантуре АО «Международный университет информационных технологий» по специальности «Вычислительная техника и программное обеспечение».

Темой диссертационного исследования является «Разработка и применение высокоточных методов распознавания образов».

Разработанная гибридная модель искусственных нейронных сетей для распознавания образов/жестов для казахского жестового алфавита используется в качестве десктопного приложения (ПО) в работе учебного центра «SMART EDU» с сентября 2023 года.

Данная модель представляет собой комбинирования методов обучения с учителем – на базе сверточных нейронных сетей (CNN) и обучения без учителя на основе слоев рекурентных нейронных сетей – LSTM для классификации жестов многоклассовой задачи.

Модель распознает с высокой точностью более 95% и прогнозирует угаданные жесты рук казахского жестового алфавита.

Данная разработанная модель с программным обеспечением проходит апробацию в условиях и в среде лабораторный и экспериментальной работы предприятия.

Директор Макаров И.А **подпись**

М.П.

ПРИЛОЖЕНИЕ В

Программный код

```
import cv2
import numpy as np
import os
from matplotlib import pyplot as plt
import time
import mediapipe as mp
mp_holistic = mp.solutions.holistic # Holistic model
mp_drawing = mp.solutions.drawing_utils # Drawing utilities
def mediapipe_detection(image, model):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # COLOR CONVERSION BGR 2 RGB
    image.flags.writeable = False                  # Image is no longer writeable
    results = model.process(image)                 # Make prediction
    image.flags.writeable = True                   # Image is now writeable
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR) # COLOR COVERSION RGB 2 BGR
    return image, results
def draw_landmarks(image, results):
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS) # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS) # Draw right hand connections
def draw_styled_landmarks(image, results):
    # Draw left hand connections
    mp_drawing.draw_landmarks(image, results.left_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(121,22,76), thickness=2,
circle_radius=4),
                              mp_drawing.DrawingSpec(color=(121,44,250),
thickness=2, circle_radius=2)
                              )
    # Draw right hand connections
    mp_drawing.draw_landmarks(image, results.right_hand_landmarks,
mp_holistic.HAND_CONNECTIONS,
                              mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=4),
                              mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)
                              )
cap = cv2.VideoCapture(0)
# Set mediapipe model
with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():

        # Read feed
        ret, frame = cap.read()
```

```
        # Make detections
        image, results = mediapipe_detection(frame, holistic)
        print(results)

        # Draw landmarks
        draw_styled_landmarks(image, results)

        # Show to screen
        cv2.imshow('OpenCV Feed', image)

        # Break gracefully
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
draw_landmarks(frame, results)
plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
def extract_keypoints(results):
    lh = np.array([[res.x, res.y, res.z] for res in
results.left_hand_landmarks.landmark]).flatten() if results.left_hand_landmarks
else np.zeros(21*3)
    rh = np.array([[res.x, res.y, res.z] for res in
results.right_hand_landmarks.landmark]).flatten() if results.right_hand_landmarks
else np.zeros(21*3)
    return np.concatenate([lh, rh])
# Path for exported data, numpy arrays
DATA_PATH = os.path.join('MP_Data')

# Actions that we try to detect
actions = np.array(["NONE", 'А', 'Ә', 'Б', 'В', 'Г', 'Ғ', 'Д', 'Е', 'Ё', 'Ж', 'З',
'И', 'Й', 'К', 'Қ', 'Л', 'М', 'Н', 'Ң', 'О', 'Ө', 'П', 'Р', 'С', 'Т', 'У', 'Ұ',
'Ү', 'Ф', 'Х', 'Һ', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'І', 'Ь', 'Э', 'Ю', 'Я', ])

# Thirty videos worth of data
no_sequences = 100

# Videos are going to be 30 frames in length
sequence_length = 30

# Folder start
start_folder = 1
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import TensorBoard
log_dir = os.path.join('Logs')
tb_callback = TensorBoard(log_dir=log_dir)
model = Sequential()
```

```
model.add(LSTM(64, return_sequences=True, activation='relu', input_shape=(30,126)))
model.add(LSTM(128, return_sequences=True, activation='relu'))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(44, activation='softmax'))
from sklearn.metrics import multilabel_confusion_matrix, accuracy_score
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(ytrue, yhat)
cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(15, 12))
sns.set(font_scale=1.2) # for label size
ax = sns.heatmap(cm_normalized, annot=True, annot_kws={"size": 10}, cmap="Blues",
fmt=".2f", xticklabels=actions, yticklabels=actions)
ax.set_title('Confusion Matrix', fontsize=20)
ax.set_xlabel('Predicted', fontsize=18)
ax.set_ylabel('True', fontsize=18)
plt.show()
```