МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОКОГНИТИВНЫХ ТЕХНОЛОГИЙ

Поткин Олег Александрович

Информационная технология распознавания жестов для человеко-машинного взаимодействия на базе сверточных нейронных сетей

Выпускная квалификационная работа

Направление:
«Информатика и вычислительная техника» — 09.04.01
Образовательная программа
«Компьютерная лингвистика и искусственный интеллект» — «CLAIM»

Научный руководитель
к.т.н., доцент Филиппович А.Ю.

МОСКВА, 2019 г.

Аннотация

В диссертации предлагается анализ и реализация алгоритмов детектирования, трекинга и классификации статических жестов Русского Жестового Языка в видеопотоке с использованием методов компьютерного зрения и машинного обучения. Решение является платформонезависимым. Обучающая выборка для нейронной сети не имеет аналогов, включает в себя 10 классов и состоит более чем из 2000 уникальных изображений. Программное обеспечение включает модуль детектирования кистей рук на изображении с выделением региона изображения, модуль трекинга, модуль классификации статичных жестов на выделенном регионе изображения, а так же вспомогательный модуль предварительной обработки, расширения обучающей выборки и обучения нейронной сети.

Abstract

The thesis proposes an analysis and implementation of algorithms for detecting, tracking and classification of the static gestures for Russian Sign Language in a video stream using computer vision and machine learning techniques. The solution is platform independent. The dataset has no analogues, includes 10 classes and consists of more than 2000 unique images. The software includes a hand detection module with the hand segmentation, a gesture tracking module, a static gestures classification module in the detected region of the image, as well as an auxiliary preprocessing module, dataset augmentation module and the neural network maintanance module for its training and evaluation.

СПИСОК ОТСЫЛОЧНЫХ ДОКУМЕНТОВ

1. Язык русский жестовый. Услуги по переводу для инвалидов по слуху. Основные положения — ГОСТ Р 57636-2017
2. Устройства и методы на базе жестовых интерфейсов (Gesture-based interfaces across devices and methods) — ISO/IEC FDIS 30113-5

СПИСОК ОСНОВНЫХ ТЕРМИНОВ И ИХ ОПРЕДЕЛЕНИЙ

Алгоритм — это пошаговый метод решения той или иной проблемы. Обычно используется для задач обработки данных, расчетов и других компьютерных и математических операций. Также используется для операций с данными (вставка нового элемента в контейнер, поиск определенного элемента, сортировка и другие).

Градиентный спуск — алгоритм итерационной оптимизации первого порядка для нахождения минимума функции.

Детектирование (объектов) — это компьютерная технология, связанная с компьютерным зрением и обработкой изображений, целью которой является обнаружение экземпляров семантических объектов определенного класса (таких как люди, здания или автомобили) в цифровых изображениях и видео.

Искусственный интеллект — это интеллект, демонстрируемый машинами, в отличие от естественного интеллекта, демонстрируемого людьми и животными. В разговорной речи термин «искусственный интеллект» используется для описания машин, которые имитируют «когнитивные» функции, а так же такие человеческие способности как «обучение» и «решение проблем».

Классификатор — алгоритм, который решает задачу классификации.

Машинное зрение — это междисциплинарная научная область, которая исследует принципы, методы и алгоритмы, дающие возможность вычислительным системам понимать абстракции высокого уровня из цифровых изображений или видео. С точки зрения инженерии, это автоматизация задач, которые может выполнять зрительная система человека.

Машинное обучение — это направление науки, которое занимается исследованием алгоритмов и статистических моделей, которые используются компьютерными системами для эффективного выполнения конкретной задачи без использования явных инструкций, вместо этого опираясь на шаблоны и выводы.

Нейронная сеть (искусственная) — вычислительная система, которая имеет сходства, но не обязательно идентична биологическим нейронным сетям, формирующим мозг животных. Такие системы «учатся» выполнять задачи, рассматривая примеры и при этом они не запрограммированы какими-либо правилами для решения конкретных задач.

Персептрон — алгоритм обучения с учителем (supervised learning) бинарных классификаторов. Бинарный классификатор — это функция, которая может решить, принадлежит ли вход, представленный вектором чисел, некоторому определенному классу.

Промышленный робот — роботизированная система, используемая на производстве.

Промышленные роботы автоматизированы, программируются и могут перемещаться по трем или более осям.

Распознавание — классическая проблема в компьютерном зрении, обработке изображений и машинном зрении, состоящая в том, чтобы определить, содержат ли данные изображения какой-либо конкретный объект, свойство или действие.

Трекинг — процесс определения местоположения движущегося объекта (или нескольких объектов) с течением времени в потоке данных (например, видео).

СОДЕРЖАНИЕ

Введение....8
1 Аналитический раздел....12
1.1 Анализ и формализация объекта исследования....12
1.2 Обзор и анализ существующих решений для распознавания жестов при человеко-машинном взаимодействии....16
1.2.1 Высокоточная система жестового управления....17
1.2.2 Volkswagen Golf R Touch Gesture Control....18
1.2.3 DICE − система жестового управления автомобилем от Mercedes-Benz....18
1.2.4 Патент US8634980B1: Driving pattern recognition and safety control....19
1.2.5 Патент US20060136846A1: User interface apparatus using hand gesture recognition and method thereof....20
1.2.6 Патент US20140254864A1: System and method for gesture detection through local product map....21
1.3 Выводы....23
2 Конструкторско-технологический раздел....24
2.1 Обзор и анализ существующих решений для детектирования и распознавания объектов на изображении....24
2.1.1 Методы выделения сюжетной части в системах распознавания изображений. .24
2.1.2 Применение искусственных нейронных сетей в задачах распознавания изображений....28
2.1.3 Обучение искусственных нейронных сетей....33
2.1.4 Применение методов глубокого обучения в задачах распознавания изображений....37
2.1.5 Подходы к обучению сверточных нейронных сетей....39
2.1.6 Краткий обзор фреймворков для задач глубокого обучения....41
2.1.7 Особенности подготовки набора данных для обучения нейронной сети....45
2.1.8 Методы оценки качества работы классификатора....46
2.2. Выбор технологии для программной реализации алгоритма классификации жестовых команд....48
2.2.1 Разработка технического задания....48
2.2.2 Выбор фреймворка для разработки модели нейронной сети....50
2.3 Выводы....50
3 Специальный раздел....52

3.1 Программная реализация алгоритма распознавания жестовых команд.....52
3.1.1 Предварительная обработка изображений и детектирование кисти руки.....53
3.1.2 Подготовка набора данных для классификатора.....54
3.1.3 Архитектура классификатора на основе сверточной нейронной сети для распознавания статических жестовых команд.....57
3.1.4 Ресурсные требования к программной и аппаратной реализации.....60
3.2 Экспериментальные исследования.....61
3.2.1 Сравнительный анализ работы классификаторов жестовых команд.....61
3.3 Выводы.....62
Заключение.....63
Список литературы.....67
Приложение А. Доклад по теме ВКР (русская версия).....71
Приложение Б. Доклад по теме ВКР (английская версия).....74
Приложение В. Графический материал работы (макеты плакатов).....76
Приложение Г. Программная документация.....81
Приложение Д. Реферативный обзор основной литературы.....90
Приложение Е. Статья для конференции AIST 2018.....93
Приложение Ж. Статья для конференции AIST 2019.....99

ВВЕДЕНИЕ

Роботизированные системы стали ключевыми компонентами в различных отраслях промышленности. В последнее время концепция Human-Robot Collaboration (далее HRC) привлекла внимание исследователей. Литературные примеры предполагают, что человек обладает несравненными навыками решения проблем в значительной степени благодаря продвинутым сенсорно-двигательным способностям, но имеет ограниченную силу и точность [1]. Однако роботизированные системы имеют стойкость к усталости, высокую скорость, точность и производительность, но серьезные ограничения в гибкости. HRC может освободить человека от тяжелых задач посредством интуитивного и надежного интерфейса взаимодействия для повышения общей эффективности. В представленном исследовании разработан прототип такого интерфейса, где в основе лежат жестовые команды.

Жесты являются одним из способов обмена информацией, общения. Информация, лежащая в основе мимики, жестов рук и позы тела лежит в основе эффективного канала связи при взаимодействии людей [2][3].

Распознавание жестов относится к математической интерпретации человеческих движений вычислительным устройством. Чтобы взаимодействовать с человеком, роботизированные системы должны правильно понимать человеческие жесты и выполнять соответствующие команды в достаточной степени точности.

В настоящее время такие отраслевые гиганты как Google, Apple, Kuka Robotics, BMW, Facebook, Netflix и другие активно развивают направление перспективных интерфейсов человеко-машинного взаимодействия, где жестовое взаимодействие одно из наиболее востребованных, а задача качественного и уверенного распознавания жестовых команд является одной из основных. В дополнение, создание эффективных каналов взаимодействия, в том числе на базе жестовых команд, может освободить людей от тяжелых и потенциально опасных задач.

Актуальность проведенных исследований заключается в разработке прототипа системы распознавания жестовых команд для его последующего усовершенствования и коммерциализации.

В рамках диссертации реализованы следующие модули системы:

— уникальный набор данных (изображения статических жестов) для обучения и тестирования алгоритмов машинного обучения;
— программное обеспечение для предварительной подготовки данных и обучения классификатора на базе сверточной нейронной сети;

— программное обеспечение для классификации статических жестов и визуализации результата.

Основными задачами исследования являются:

— анализ методов классификации изображений, основанных на использовании искусственных нейронных сетей;
— выбор оптимального метода выделения информативной части на изображениях;
— выбор оптимальной архитектуры нейронной сети для распознавания статических жестов в видеопотоке;
— оптимизация параметров используемой нейронной сети;
— выбор методов и параметров аугментации (расширения) набора данных для обучения классификатора;
— реализация и исследование работоспособности и эффективности алгоритма распознавания статических жестов, основанного на использовании искусственной нейронной сети.

В диссертационной работе при решении поставленных задач использованы методы теории искусственных нейронных сетей, математического моделирования, теории вероятностей и математической статистики. Для разработки программных компонентов были использованы алгоритмы компьютерного зрения, а именно:

— морфологические преобразования изображения;
— поиск объекта (ROI — Region of Interest) по цвету и контуру;
— изменение размеров (Resize) и выделение фрагментов изображений.

В качестве одного из ключевых компонентов программного модуля были использованы алгоритмы машинного обучения, а именно сверточные нейронные сети. Также были разработаны вспомогательные программные компоненты для предварительной обработки данных, обучения нейронной сети и проверки точности классификатора. Проверка точности заключается в вычислении погрешности, точности и полноты.

Научная новизна заключается в способах оптимизации методов обучения нейронной сети, повышении качества и увеличении объема набора данных, увеличения точности классификации.

В качестве способов оптимизации применяются:

— оптимизация набора данных для обучения;
— аугментация набора данных для обучения;
— подбор параметров классификатора.

Уникальный набор данных, разработанная архитектура сверточной нейронной сети, алгоритмы предварительной обработки данных и обучения классификатора являются основой для создания программной системы распознавания жестовых команд.

Диссертация состоит из трех основных разделов:

— аналитический раздел, где осуществляется анализ предметной области и приводятся примеры уже разработанных систем и алгоритмов распознавания жестов и жестовых команд, патентов, связанных с распознаванием жестовых команд, производится анализ и сравнение актуальных разработок, представленных в статьях российских и зарубежных авторов;

— конструкторско-технологический раздел, где приводится описание и анализ современных методов и алгоритмов распознавания объектов на изображении, производится обзор и сравнение популярных библиотек машинного обучения содержащих рассматриваемые методы и соответствующий набор инструментов, а также осуществляется выбор метода и библиотеки для разработки программного обеспечения для детектирования, трекинга и классификации жестов;

— специальный раздел, где приведены алгоритмы и подходы к задаче обработки набора графических данных, продемонстрирована программная реализация алгоритма распознавания жестовых команд, разобраны различные архитектуры нейронных сетей, их достоинства и недостатки, а также представлены качественные результаты работы классификатора жестовых команд.

Целью разрабатываемого программного комплекса является внедрение в мультимедийные системы транспортных средств (Infotainment Systems), а также обеспечение возможности отдельным разработчикам и исследователям использовать разрабатываемые алгоритмы и набор данных в целях доработки и использования в собственных исследованиях.

Практическая значимость заключается в разработке системы распознавания жестовых команд, которую можно использовать в качестве интерфейса для человеко-машинного взаимодействия, а также в обеспечении возможности тестирования алгоритмов классификации и применении пользовательских наборов данных и параметров нейронной сети (метод обучения, топология нейронной сети и др.) для решения смежных задач.

Тема и материалы диссертации были представлены в трех научных работах, опубликованных в «CEUR-WS series»:

— статья «Static gestures classification using Convolutional Neural Networks on the example of the Russian Sign Language» для конференции AIST-2018;
— постер с графическими материалами для статьи «Static gestures classification using Convolutional Neural Networks on the example of the Russian Sign Language» для конкурса постеров на конференции AIST-2018;
— статья «Hand gestures detection, tracking and classification using Convolutional Neural Network» для конференции AIST-2019;

1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

1.1 Анализ и формализация объекта исследования

Анализ — это процедура мысленного или материального разделения целостного объекта (предмета, явления, процесса) на составляющие части (признаки, свойства, отношения) с целью их изучения [4].

Формализация — это совокупность познавательных операций, обеспечивающая отвлечение от значения понятий и смысла выражений формализованной теории с целью исследования её логических особенностей, дедуктивных и выразительных возможностей. В символической логике и математике, где формализация наиболее развита, под ней понимают реконструкцию содержательной научной теории в виде формализованного языка — искусственной знаковой системы, предназначенной для представления некоторой научной теории в процессе научного поиска [5].

Формализованная информационная модель — это определенные совокупности знаков (символов), которые существуют отдельно от объекта моделирования, могут подвергаться передаче и обработке. Реализация информационной модели на компьютере сводится к ее формализации в форматы данных, с которыми работает компьютер.

Формализация информации о жестах, а именно ручных жестовых командах, заключается в классификации жестов рук, выделении ряда параметров, присущих всей совокупности жестов, группе ручных жестов, каждому ручному жесту в отдельности. Формализация информации о ручных жестах необходима для создания максимально простого и быстродействующего алгоритма распознавания жестовых команд.

Основой для построение системы жестовых команд в диссертации являются дактилемы Русского Жестового Языка (далее РЖЯ).

Основная единица жестового языка — жест, который характеризуется иконичностью, то есть возможностью обозначения денотата (обозначаемого предмета) путем изображения каких-либо визуальных параметров предмета. Жест в РЖЯ не является неделимым. В настоящее время выделяют пять компонентов жеста: конфигурация (форма руки / рук), место исполнения (локализация), направление движения, характер движения и немануальный компонент (выражение лица и артикуляция) [6]. Следовательно, жесты русского жестового языка могут быть классифицированы по различным основаниям.

Визуально-кинестетический канал передачи информации определяет специфику передачи денотата, поэтому первой должна быть классификация жестов РЖЯ по

особенностям передачи денотата.

В РЖЯ существуют указательные жесты, которые активно используются для описания пространственных и определительных отношений. Использование указательного жеста оказывается возможным благодаря общности предварительных сведений участников разговора, их знаний об окружающей жизни.

Известна классификация иконических жестов РЖЯ [7]:

— рисующие жесты, обрисовывающие контур обозначаемого предмета (например, БАКЕНБАРДЫ: большой и указательный пальцы обрисовывают бакенбарды);
— пластические жесты, дающие пластическое изображение денотата (например, ГРОБ: согнутые кисти рук, направленные кончиками пальцев от себя, кладутся одну на другую сторонами ладоней);
— жесты, имитирующие действия (например, ЖАРИТЬ: на обращенную кверху ладонь левой руки кладется тыльной стороной ладонь правой руки, которая затем переворачивается и кладется ладонью).

Так как РЖЯ — беззвучный язык, то жесты невозможно классифицировать по фонетическому признаку, то есть по наличию и месту в них ударения.

По морфологическому словообразовательному признаку классифицируют простые, производные и сложные слова. Жесты РЖЯ также могут быть простыми (например, ЧАШКА), производными (например, в жесте ИСПОЛЬЗОВАТЬ, сначала дактилируются знаки И, С, затем показывается жест ПОЛЬЗА). Проблемой является название третьей группы жестов — сложные или составные. Например, только что приведенный жест ИСПОЛЬЗОВАТЬ. Не меньшую трудность представляет определение статуса жеста ПТЕНЕЦ, который представляет собой сочетание жестов ПТИЦА и МАЛЕНЬКИЙ [8].

Проблемы классификаций жестов русского жестового языка определяются его спецификой, недостаточной изученностью и слабой степенью документированности РЖЯ, некоторыми традициями к его описанию [8].

Дактилемы (буквы дактильного алфавита) широко представлены в дактилологии.

Термин дактилология (от греческого dactilos — палец, logos — слово, учение) используется в двух основных значениях. Во-первых, так называют алфавит воспроизведённый пальцами руки (рук). В этом значении слово употребляется в высказываниях: «русская дактилология — одноручная», «английская дактилология — двуручная» и т. п. Во-вторых — общение при помощи ручной азбуки, т. е. дактильной речи [9].

Русский дактильный алфавит был опубликован в первой русской книге об

обучении глухих Флёри [10]. В этом дактильном алфавите отчётливо прослеживается принцип пальцевого воспроизведения графического изображения буквы.

При разработке русской дактилологии стремились, чтобы дактилема (буква дактильного алфавита) была максимально похожа на свой оригинал. Во многом это удалось, но некоторые дактилемы всё же весьма условные (рисунок 1).

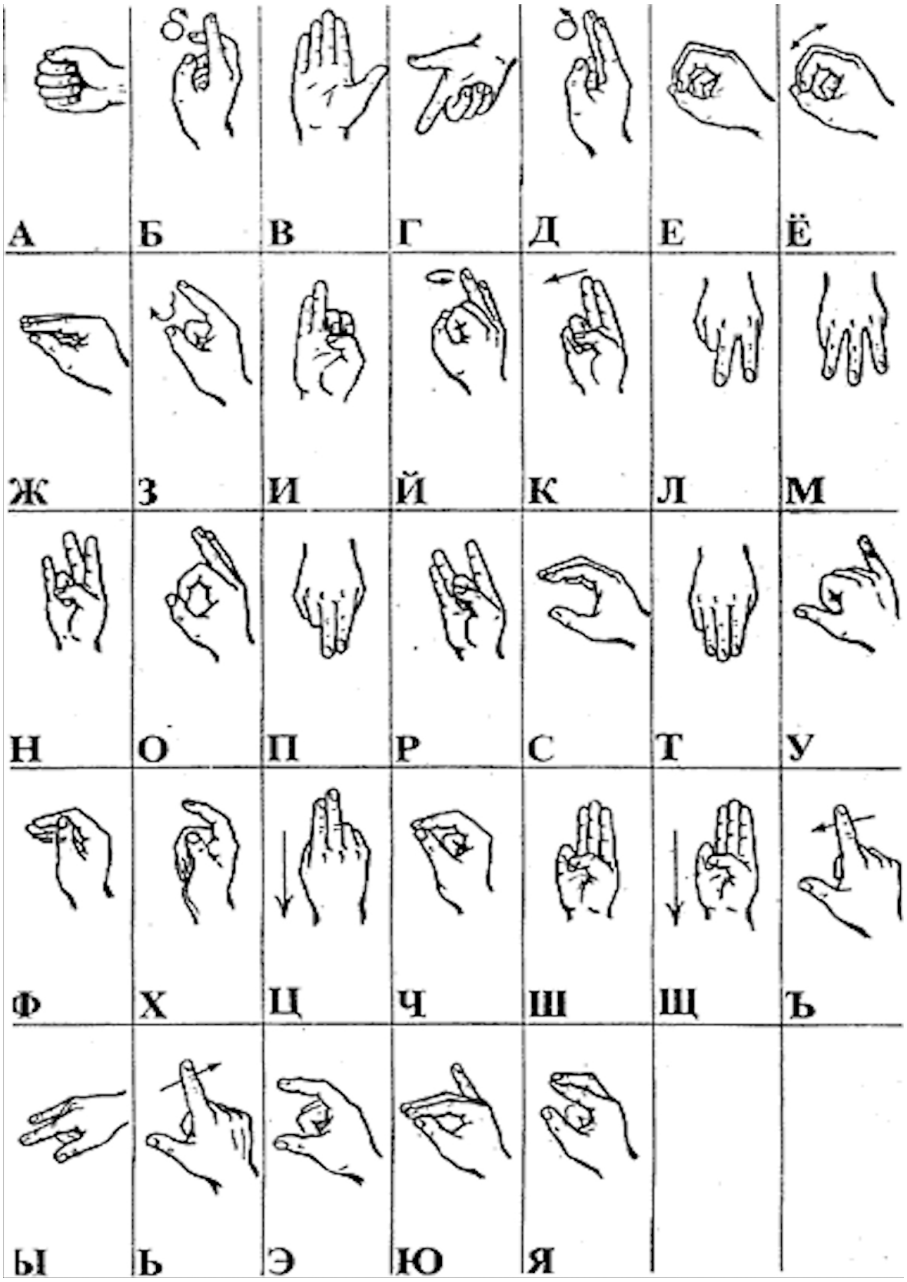

Рисунок 1 – Примеры дактилем Русского Жестового Языка

Например, в дактилемах «Л», «М», «З», «С» — чётко видна буква русского алфавита, а дактилемы «Р», «Ч», «Ж», «Ф» — весьма условны.

Общаясь при помощи дактилологии следуют нормам орфографии русского языка. В то же время дактилирование обязательно сопровождается устной речью — артикулированием (во время перевода — без голоса). Произношение дактилируемых слов и словосочетаний должно соответствовать орфоэпическим нормам. На первых порах расхождение при дактилировании и произношении (проговаривании) существенное и очень мешает. Но при усердных занятиях возможно преодолеть возникающие трудности. Ещё одна трудность — одновременно видна только одна дактилема, а все предыдущие нужно помнить [9].

Формализация в случае жестовых команд на основе дактилем РЖЯ является некоторой формой классификации, но не привычной для понимания человека, а удобной для последующей обработки жестовой информации как изображения при распознавании. Исходя из этого принципа, формализуем два класса жестовых команд из множества жестов, в основе которых лежат дактилемы РЖЯ:

— статические — жестовые команды на базе дактилем, которые полностью описываются одним фреймом, например: А, Б, В, Г, Е и др.;

— динамические — жестовые команды на базе дактилем, которые не могут быть полностью описаны одним фреймом, например: Ё, З, Й, Ц и др.;

При решении задачи распознавания изображений необходимо формализовать объект, который необходимо распознать, выделить ключевые особенности объекта и оперировать этими особенностями.

Формализация информации о жесте является одним из важнейших условий при решении задачи распознавания жестовых команд. Рассмотрим конструктивные особенности, а именно визуальные характеристики статичестких жестов в качестве объектов формализации.

Человеческие руки и тело обладают уникальными визуальными особенностями. В задаче распознавания жестов на основе изображений жесты состоят из фрагментов изображений рук и/или тела. Поэтому использование таких визуальных признаков в идентификации жестов вполне обоснованно.

Цвет – это простая визуальная функция для идентификации жестов из фоновой информации. Однако в сложной среде HRC на системы распознавания жестов на основе цветов сильно влияют освещение и тени [11]. Еще одна распространенная проблема в обнаружении цвета кожи заключается в том, что цвет кожи человека сильно различается среди человеческих рас (рисунок 2).

Рисунок 2 − Палитра цветов кожи

Из-за вышеперечисленных проблем, в современных подходах, цвет кожи рассматривается только как один из многих параметров при идентификации жестов.

Другой интуитивный и простой способ идентификации жестов — использование уникальной формы и контура человеческого тела. Существенный вклад в формализацию отношения форм и соответствий был внесен Сержем Белонги и его коллегами [12]. Они разработали метод дескриптора контекста формы. Дескриптор контекста формы используется для обнаружения похожих фигур на разных изображениях. Датчики глубины позволили точно измерять форму поверхности. 3D-модели, созданные на основе таких технологий, позволяют очень детально представлять форму человеческого тела [13].

В распознавании жестов на основе изображения условия освещения сильно влияют на качество идентификации жестов. Поэтому многие исследователи используют методы локальных признаков, которые не чувствительны к условиям освещения. К таким методам относятся, например, SURF и ORBare [14][15]. Как правило, локальные характеристики также как и цветовые рассматриваются как один из множества параметров при идентификации жестов. Несколько методов идентификации, таких как методы формы и контура, методы движения и методы обучения, основаны на локальных признаках.

Совмещая перечисленные конструктивные особенности получаем формализованную модель, использование которой упростит задачу распознавания статичных (а в перспективе и динамических) жестов. Формализованная модель строится путем последовательного детектирования определенной визуально-конструктивной особенности жеста. Наиболее общей конструктивной особенностью является цвет, затем форма и контур. Наименее общим признаком для жеста является содержимое его частей (неполное представление). Эти условия позволяют составить последовательность алгоритмов детектирования и распознавания статических жестовых команд.

1.2 Обзор и анализ существующих решений для распознавания жестов при человеко-машинном взаимодействии

В настоящее время ведется довольно много исследований по созданию методов

распознавания образов, позволяющих бесконтактно взаимодействовать с компьютером посредством жестов рук. В результате мы видим множество новых прототипов, готовых к массовому внедрению продуктов и патентов, в основе которых лежит технология бесконтактного жестового взаимодействия.

1.2.1 Высокоточная система жестового управления

Начинающая компания 3Gear из Сан-Франциско (США) создаёт новую систему жестового управления, способную отслеживать малейшие изменения положения кистей рук и пальцев [16].

Аппаратная составляющая комплекса включает две 3D-камеры, подвешенные на специальной рамке над поверхностью рабочего стола (рисунок 3).

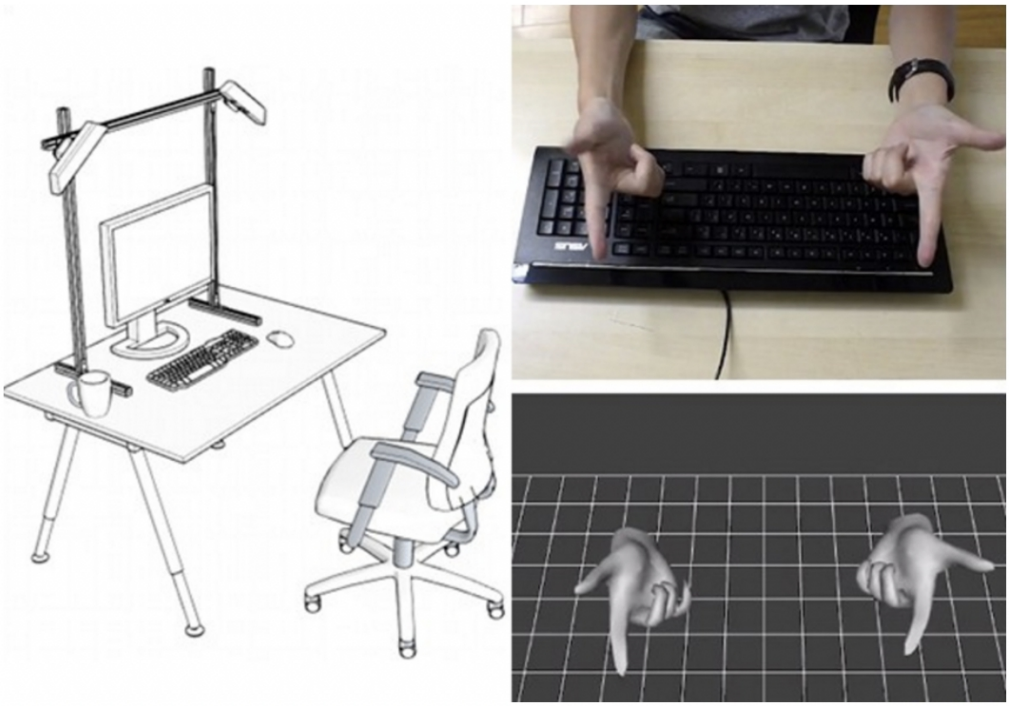

Рисунок 3 − Высокоточная система жестового управления

Изображения, снятые со скоростью 30 кадров в секунду, обрабатываются специализированным программным обеспечением. Этот процесс предполагает поиск совпадения в обширной базе данных, содержащей информацию о 30 тысячах возможных положений пальцев и рук.

Разработчики подчеркивают, что система способна реагировать на малейшие изменения — когда пальцы двигаются всего на один миллиметр. В этом комплексе высокая скорость реакции и распознавание жестов занимает в среднем 33 миллисекунды.

Возможные сферы применения: игровые и развлекательные комплексы, CAD-системы и др.

1.2.2 Volkswagen Golf R Touch Gesture Control

Компания Volkswagen на примере электрического автомобиля e-Golf Touch продемонстрировала некоторые особенности нового бортового информационно-развлекательного комплекса. Презентация состоялась на выставке CES 2016 в Лас-Вегасе (Невада, США) [17]. Бортовой центр поддерживает управление при помощи жестов (рисунок 4) и голосовых команд.

Рисунок 4 − Жестовое управление мультимедийной системой Volkswagen

Такой подход должен минимально отвлекать водителя от управления транспортным средством, делая взаимодействие с бортовой электроникой интуитивно понятным и максимально удобным.

1.2.3 DICE − система жестового управления автомобилем от Mercedes-Benz

Система Dynamic & Intuitive Control Experience (DICE) использует ветровое стекло автомобиля как большой интерактивный дисплей (рисунок 5).

Рисунок 5 − Пользовательский интерфейс системы DICE

Этот дисплей функционально разбит на три зоны: музыкальную, информационную и зону автомобильной навигации [18]. Простое помахивание рукой заставит активироваться некоторые функции системы и выполнить определенные действия, при этом пользователь освобождается от необходимости искать для этого определенную кнопку или переключатель.

1.2.4 Патент US8634980B1: Driving pattern recognition and safety control

Правообладатель: Google Inc

Статус: US Grant, 2014

Google предлагает [19] отслеживать движения рук водителя и преобразовывать определённые жесты в команды для контроля бортовой электроники (рисунок 6).

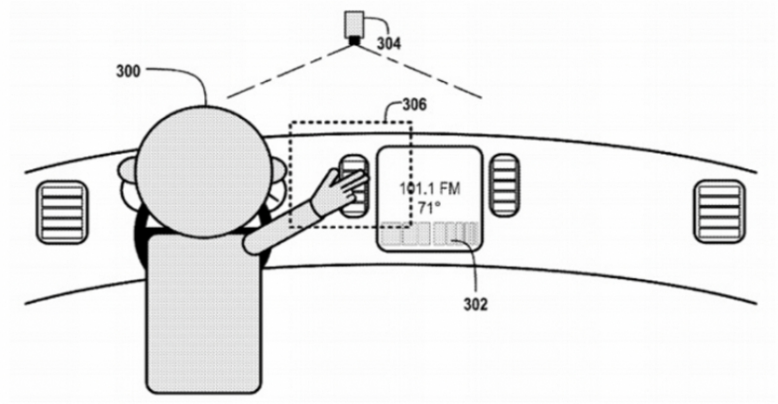

Рисунок 6 − Схема работы системы, описанной в патенте US8634980B1

К примеру, взмахнув ладонью рядом с окном, автомобилист сможет поднять или опустить стекло. А аналогичный жест рядом с мультимедийной системой позволит увеличить или уменьшить громкость.

Для отслеживания положения рук водителя предлагается использовать специальную камеру на потолке транспортного средства, позволяющую формировать карту глубины пространства внутри салона. Кроме того, данные будут собираться при помощи лазерного сканера.

1.2.5 Патент US20060136846A1: User interface apparatus using hand gesture recognition and method thereof

Правообладатель: Intellectual Discovery Co Ltd

Статус: US Grant, 2010

В последнее время автомобили оснащаются различными устройствами при этом улучшается качество и функциональность самих автомобилей. Встраиваемые устройства становятся функциональнее; такие устройства как компакт-диски (CD) и телевизионные системы (ТВ) стали типичной комплектацией стерео-систем. Кроме того, в комплектацию современных автомобилей включены различные датчики а так же камеры, навигационная система (CNS), система глобального позиционирования (GPS) и географическая информационная система (GID).

По мере того, как автомобильные телематические службы становятся разнообразнее и сложнее в использовании, управление ими так же усложняется и базовых кнопочных операций недостаточно. Кроме того, учитывая, что большинство телематических служб используется во время вождения, требуется больше простых интерфейсов управления.

Для ввода данных в телематических системах могут использоваться интерфейсы, подобные интерфейсам ввода в мобильных устройствах, такие как голосовой (распознавание речи), сенсорный (использование сенсорного ввода). Однако, голосовое управление имеет ряд существенных недостатков: низкая скорость распознавания речи, фильтрация посторонних шумов.

В то же время, сенсорный экран очень удобен в мобильных устройствах, например, в персональных цифровых ассистентах (PDA). Тем не менее, так как терминал телематики в основном используется во время вождения, сенсорный экран требует тщательного внимания, что является отвлекающим фактором для водителя и потенциальной угрозой безопасности.

Таким образом, требуется удобный и простой интерфейс управления телематическими средствами, не отвлекающий водителя от управления транспортным средством.

Intellectual Discovery Co Ltd. в патенте US7702130B2 предлагает реализацию интерфейса управления мультимедийными системами автомобиля с применением жестового управления (рисунок 7).

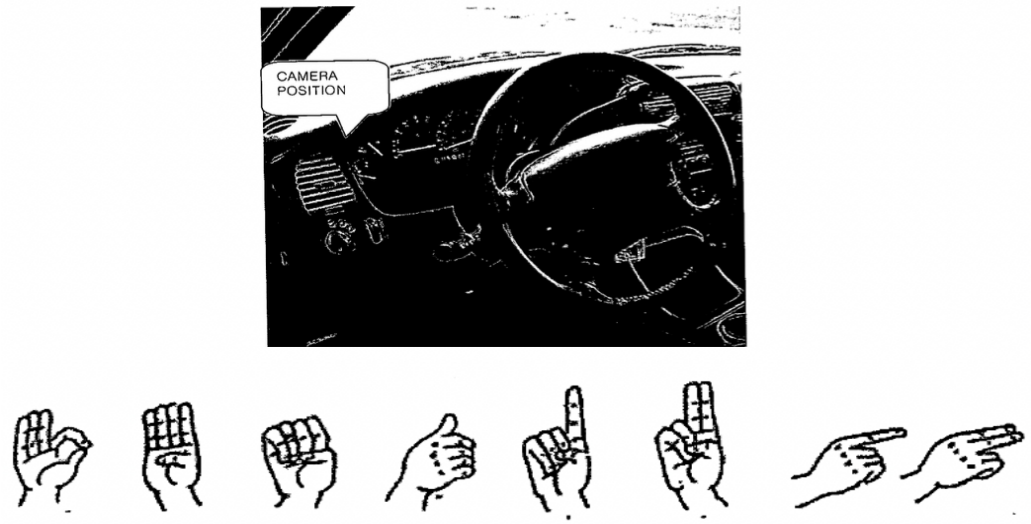

Рисунок 7 − Схема системы, описанной в патенте US20060136846A1

Устройство, использующее распознавание жестов рук [20], включает в себя:

— блок ввода для приема и регистрации команд;
— блок распознавания жестов рук для сохранения изображения жестов рук в ассоциации с конкретной командой и преобразования изображений жестов рук в соответствующую команду путем распознавания изображения жестов рук из изображения, полученного в блоке приема и регистрации команд;
— блок выполнения команд для выполнения операции, соответствующей команде, преобразованной в блоке распознавания жестов рук.

1.2.6 Патент US20140254864A1: System and method for gesture detection through local product map

Правообладатель: Google LLC

Статус: Abandoned, 2019

Настоящее изобретение относится в целом к области обнаружения жестов, а более конкретно — к новой и полезной системе и способу для обнаружения жестов посредством

локальных карт продукта в поле обнаружения жеста [21].

Операторы локального двоичного шаблона (Local Binary Pattern, LBP) обычно использовались для классификации текстур в области компьютерного зрения. LBP маркирует пиксели изображения пороговыми пикселями в поддерживаемом регионе, окружая пиксель-ядро (центральный пиксель). Если пиксель больше, чем центральный пиксель, то ему присваивается значение 1, в противном случае пикселю присваивается значение 0. Затем вычисляется гистограмма частоты двоичных значений пикселя. Затем гистограмма может быть нормализована. Полученная гистограмма затем используется в качестве признака в процессах классификации (рисунок 8).

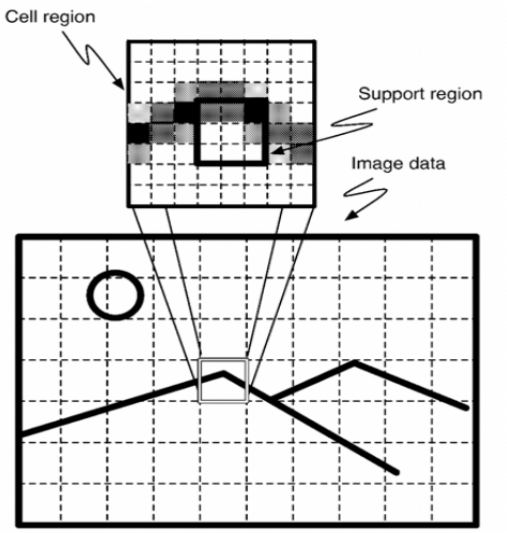

Рисунок 8 − Фрагмент локальной карты продукта

Операторы LBP стали характерной чертой для приложений; тем не менее, негативной особенностью LBP является чувствительность к изменяющимся условиям освещения и может оказаться критичной для некоторых приложений.

Таким образом, в области обнаружения жестов существует необходимость создать новую и полезную систему и способ для обнаружения жестов посредством локальных карт продукта. Настоящее изобретение обеспечивает такую новую и полезную систему и способ.

В патенте US20140254864A1 описаны система и метод для детектирования жестов через локальную карту продукта. Для достижения результата выполняют следующие шаги:

— сбор графических данных;

— на обработчике по множеству опорных областей изображения вычисляются

размерности компонента поддерживаемой области данных изображения, который характеризуется относительной величиной пикселя-ядра к пикселю не-ядра опорной области;

— вычисление вероятностной модели множества опорных регионов в вероятностной функции распределения для, по меньшей мере, одной ячейки данных изображения;
— применение классификатора вероятностной функции распределения;
— обнаружение объекта в данных изображения в соответствии с результатом примененного классификатора.

1.3 Выводы

В разделе рассматривается актуальность проводимого исследования и его практическая значимость. Также осуществляется анализ предметной области, где приводятся примеры уже разработанных систем и алгоритмов распознавания жестов и жестовых команд, патентов, связанных с распознаванием жестовых команд, производится анализ и сравнение актуальных разработок, представленных в статьях российских и зарубежных авторов. В разделе подтверждается актуальность разработки программного модуля для детектирования и классификации статических жестовых команд.

2 КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ

2.1 Обзор и анализ существующих решений для детектирования и распознавания объектов на изображении

Широкое распространение цифровых камер и вычислительных мощностей привело к активному развитию методов детектирования и распознавания объектов на изображении.

Детектирование (в задачах компьютерного зрения) — определение (обнаружение) местоположения (локализация) всех объектов заданного класса на изображении [22].

Распознавание образов — научное направление, связанное с разработкой принципов и построением систем, предназначенных для определения принадлежности данного объекта к одному из заранее выделенных классов объектов [23].

В настоящее время существует множество технологий и методов детектирования и распознавания изображений: от классических алгоритмов компьютерного зрения до нейронных сетей. Методы распознавания и детектирования объектов широко применяются в системах видеонаблюдения, медицине, робототехнике, автомобилестроении и других областях, где в основе лежат цифровые изображения, которые несут полезную информацию.

2.1.1 Методы выделения сюжетной части в системах распознавания изображений

В задачах распознавания изображений выделение сюжетной части изображения представляет собой обнаружение и локализацию целевых объектов на изображении.

Обнаружение и локализация объектов на изображении — один из основных этапов при решении задачи поиска и распознавания объектов на изображениях. Эти задачи характеризуются высокой сложностью для вычислительных машин, которая обусловлена:

— разнообразием цветов и форм распознаваемых объектов;

— наличие шумов: блики, повороты, искажения и другие;

— разная уровень освещенности детектируемых объектов.

Решение задач обнаружения и локализации позволяет провести анализ качественного состава сцены, представленной на изображении, а также получить информацию о расположении детектируемого объекта или взаимном расположении объектов.

Множество всех методов решения задачи детектирования можно разделить на три

основные группы [24]:

— методы, которые для описания объекта используют признаки, наиболее характерные для объектов: точечные особенности объекта, либо признаки, построенные для изображения, содержащего только объект;

— методы поиска объектов, соответствующих шаблону – некоторому описанию объектов;

— методы детектирования движения объектов – выделение движущихся объектов на основании нескольких изображений или кадров видео одной и той же сцены.

Один из возможных подходов к решению задачи детектирования состоит в том, чтобы использовать алгоритмы машинного обучения [25] для построения моделей классов объектов и алгоритмы вывода для поиска объектов на изображении.

Построение модели состоит из двух этапов (рисунок 9):

— извлечение признаков, характерных для объектов класса, – построение характеристических векторов-признаков для ключевых точек объекта (углов, ребер или контуров объектов) или для всего объекта;

— обучение модели на полученных признаках для последующего распознавания объектов.

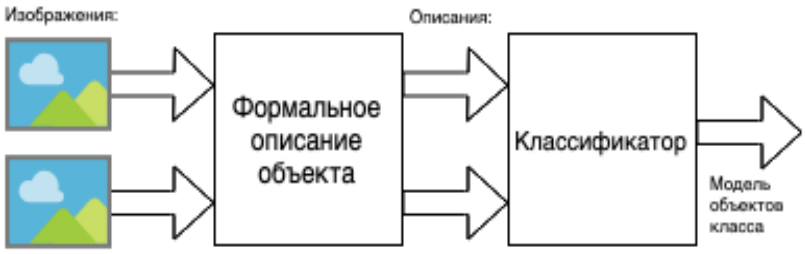

Рисунок 9 − Схема построения модели класса

Техники данной группы описывают объект с использованием векторов-признаков. Вектора строятся на основе цветовой информации (гистограмма ориентированных градиентов (Histogram of Oriented Gradients или HOG) – один из наиболее популярных способов) [26]. Также может быть использована контекстная информация, а в некоторых случаях – данные о геометрии и взаимном расположении частей объекта. Тем не менее, все эти методы строят некоторую математическую модель объекта на каждом изображении обучающей выборки, содержащем объект. Формально признак x_i – это числовая характеристика. Для каждой ключевой точки алгоритмы данной группы строят вектор признаков $(x_1, x_2, \ldots, x_n)$ Таким образом, объект описывается набором векторов

признаков в характерных точках. В результате тренировки строится модель, содержащая «усредненные» вектора признаков [27].

Алгоритм вывода (поиска) включает следующие этапы:

— извлечение признаков объекта из тестового изображения;

— поиск объектов на изображении (рисунок 10).

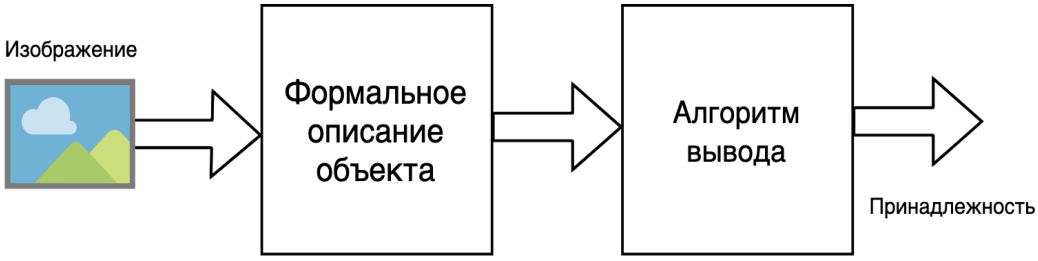

Рисунок 10 − Схема поиска объектов

Входными данными алгоритма поиска являются формальное описание объекта – набор признаков, которые выделены из тестового изображения, – и модель класса объектов. На основании этой информации классификатор принимает решение о принадлежности объекта классу [28]. Некоторые методы поиска также оценивают степень достоверности того, что объект принадлежит рассматриваемому классу.

При извлечении признаков могут возникнуть следующие проблемы:

— На изображении может быть много объектов одного класса, а необходимо найти всех представителей. Поэтому необходимо просматривать все части изображения, проходя «бегущим» окном от левого верхнего до правого нижнего угла [29]. При этом размер окна определяется размером изображений обучающей выборки.

— Объекты на изображении могут иметь разный масштаб. Самое распространенное решение – масштабирование изображения.

Качество рассматриваемых методов в основном зависит от того, насколько хорошо выбраны признаки, т.е. насколько хорошо эти признаки дифференцируют классы объектов.

Детектирование объектов на основании некоторого шаблона предполагает, что имеется изображение объекта с выделенными признаками (шаблон) и тестовое изображение, которое сопоставляется этому шаблону.

Результатом такого сопоставления (matching) является мера сходства (рисунок 11). Считается, что если эта мера больше некоторого порога, то тестовое изображение – это

изображение объекта [30].

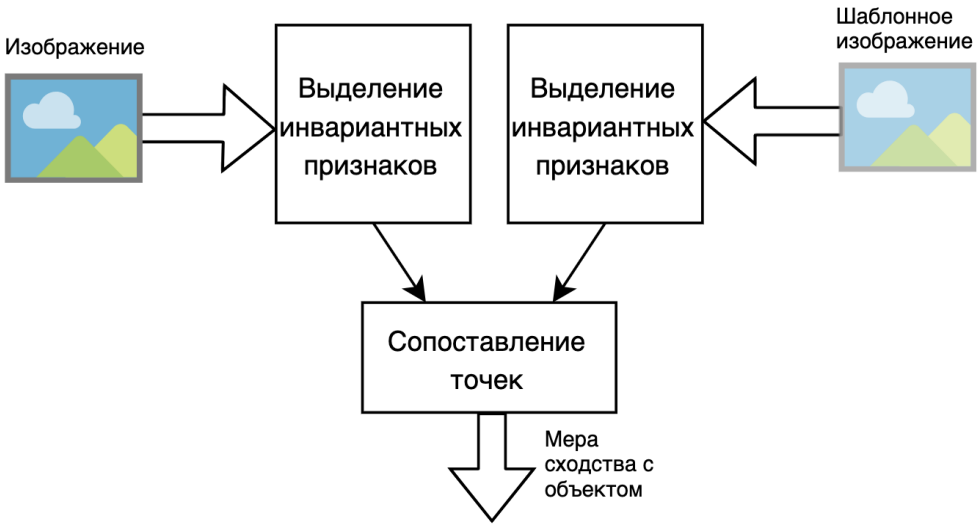

Рисунок 11 − Детектирование объектов с поиском объектов по шаблону

В простейшем случае в качестве шаблона применимо изображение объекта – матрица интенсивности цветов, которые наиболее характерны для объекта. Более сложные методы в качестве шаблона могут использовать наборы векторов признаков (дескрипторов), геометрическое описание объекта или вероятностные модели объектов, содержащих информацию о распределении интенсивностей пикселей.

В процессе поиска осуществляется итерация методом скользящего окна (Sliding Window). Окно имеет размеры шаблона (области), полученного из изображения. Далее выполняется сравнение описания части исходного изображения, покрываемого окном, и самого шаблона. Сопоставление с шаблоном подразумевает сравнение описания тестового и шаблонного изображений по некоторому заранее выбранному критерию (метрике), как правило, выбирается Евклидово расстояние [31], норма L_1 , взвешенная свертка квадратичных ошибок, либо корреляция.

Например, задано шаблонное описание объекта $I_0(X)$ в дискретном пространстве пикселей $\{X_i=(x_i, y_i)\}$. В этом случае задача поиска объекта сводится к задаче минимизации суммарной ошибки. Если в качестве меры сходства принимается Евклидово расстояние, то задача может быть формализована следующим образом:

$$E(u)=\sum_i (I(X_i+u)-I_0(X_i))^2=\sum_i e_i^2 \to min \quad ,$$

где u — смещение шаблонного описания в системе координат исходного изображения. В конечном итоге, независимо от выбранной метрики, решается задача

оптимизации.

Для алгоритмов, которые используют сопоставление дескрипторов ключевых точек, одним из наиболее важных является вопрос выбора порога, используемого в качестве критерия соответствия (в простейшем случае, если расстояние между дескрипторами меньше данного порога, точки считаются соответствующими). Увеличение данного порога приводит, с одной стороны, к увеличению числа найденных совпадений, с другой стороны, к увеличению числа ложных срабатываний. Уменьшение же порога наряду с ростом числа правильно продетектированных несовпадений ведет к росту числа правильных соответствий, которые были отброшены. Данная зависимость графически отображается с помощью ROC-кривой [32] – по величине площади под данной кривой (Area Under Curve, AUC) можно судить о качестве выбранного алгоритма построения соответствий между ключевыми точками на разных изображениях.

Методы детектирования по заданному шаблону эффективно работают при поиске одиночных объектов. При возникновении перекрытий в «бегущем окне» исчезают некоторые признаки. Поэтому при сопоставлении окна шаблону вводится порог, по которому отсекаются неперспективные окна, заведомо не содержащие объектов.

2.1.2 Применение искусственных нейронных сетей в задачах распознавания изображений

Искусственные нейронные сети (ИНС) — совокупность моделей биологических нейронных сетей. Представляют собой сеть элементов — искусственных нейронов — связанных между собой синаптическими соединениями. Сеть обрабатывает входную информацию и в процессе изменения своего состояния во времени формирует совокупность выходных сигналов [33].

Работа сети состоит в преобразовании входных сигналов во времени, в результате чего меняется внутреннее состояние сети и формируются выходные воздействия. Обычно НС оперирует цифровыми, а не символьными величинами.

Существуют два подхода к созданию искусственных нейронных сетей:

— информационный подход — безразлично, какие механизмы лежат в основе работы искусственных нейронных сетей, важно лишь, чтобы при решении задач информационные процессы в ИНС были подобны биологическим [34];

— биологический — при моделировании важно полное биологическое подобие, и необходимо детально изучать работу биологического нейрона.

Нервная система и мозг человека состоят из нейронов, соединенных между собой

нервными волокнами. Нервные волокна способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от нашей кожи, ушей и глаз к мозгу, процессы мышления и управления действиями – все это реализовано в живом организме как передача электрических импульсов между нейронами.

Биологический нейрон (Cell) имеет ядро (Nucleus), а также отростки нервных волокон двух типов (рисунок 12) – дендриты (Dendrites), по которым принимаются импульсы (Carries signals in), и единственный аксон (Axon), по которому нейрон может передавать импульс (Carries signals away). Аксон контактирует с дендритами других нейронов через специальные образования – синапсы (Synapses), которые влияют на силу передаваемого импульса.

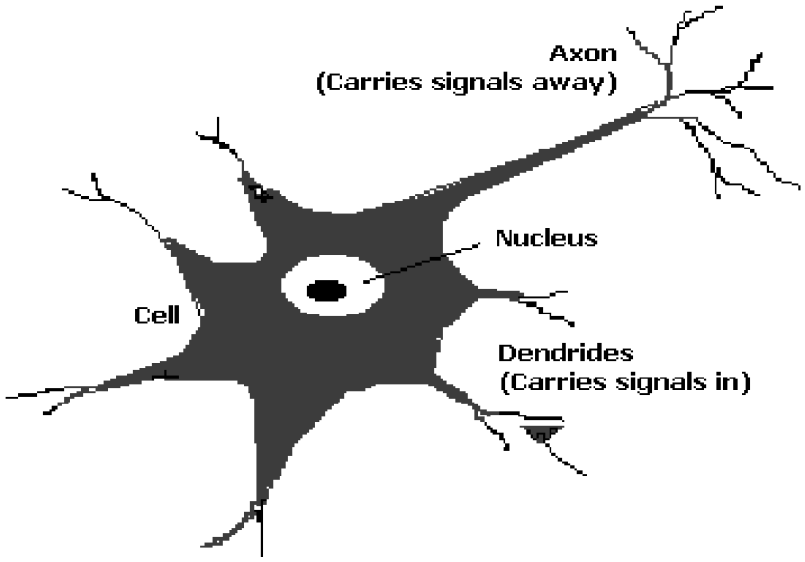

Рисунок 12 − Биологический нейрон

Искусственный нейрон (далее – нейрон) является основой любой искусственной нейронной сети. Нейроны представляют собой относительно простые, однотипные элементы, имитирующие работу нейронов мозга. Каждый нейрон характеризуется своим текущим состоянием по аналогии с нервными клетками головного мозга, которые могут быть возбуждены и заторможены [35].

Искусственный нейрон, также как и его естественный прототип, имеет группу синапсов (входов), которые соединены с выходами других нейронов, а также аксон – выходную связь данного нейрона – откуда сигнал возбуждения или торможения поступает на синапсы других нейронов.

Общий вид искусственного нейрона представлен на рисунке 13.

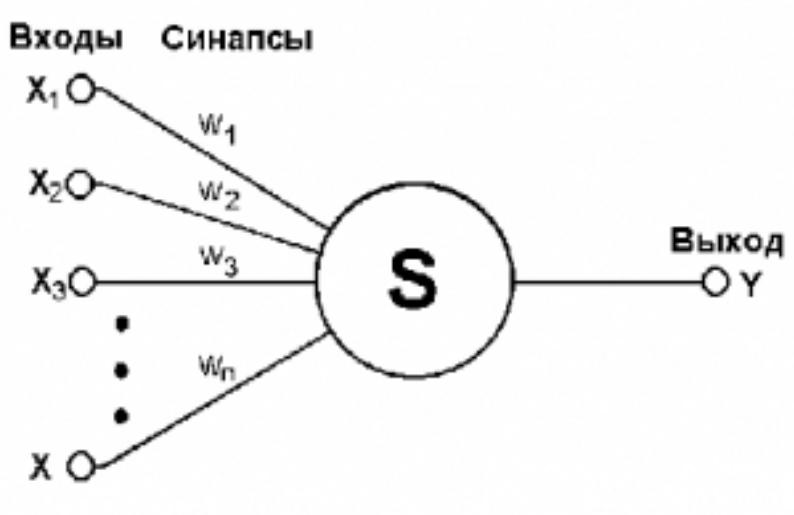

Рисунок 13 − Искусственный нейрон

Каждый синапс характеризуется величиной синаптической связи или весом w_i , который по своему физическому смыслу эквивалентен электрической проводимости.

Текущее состояние нейрона определяется как взвешенная сумма его входов: $s = \sum_{i=1}^{n} x_i w_i$, где x – вход нейрона, а w – соответствующий этому входу вес.

В настоящее время существует большое количество топологий ИНС. Полный перечень топологий ИНС составить крайне трудно, так как исследования в этой области ведутся очень активно. Многие из топологий ИНС были разработаны под конкретные задачи, а некоторые из них являются универсальными и используются в различных задачах.

Одна из первых моделей ИНС это персептрон [36]. ИНС основанные на этой топологии просты в реализации, но используются чаще совместно с другими ИНС. Сети прямого распространения передают информацию от входа к выходу. Нейроны не связаны между собой в одном слое, но связаны с соседними слоями каждый с каждым (рисунок 14).

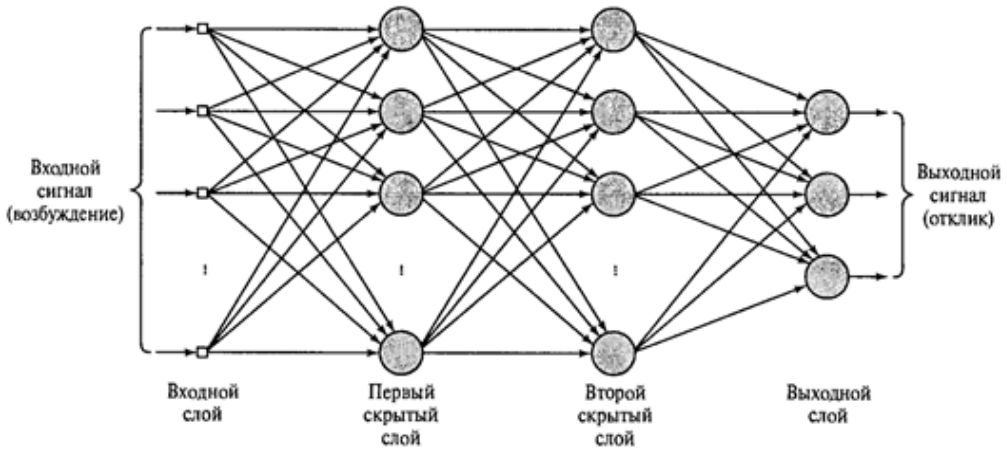

Рисунок 14 − Схема персептрона

Основным алгоритмом обучения персептрона до настоящего времени остается алгоритм обратного распространения ошибки (Back-propagation algorithm) [37] и его многочисленные модификации.

Другая распространенная модель — сети радиально-базисных функций. Это обычные сети прямого распространения, но использующие радиально-базисную функцию в качестве функции активации (рисунок 15).

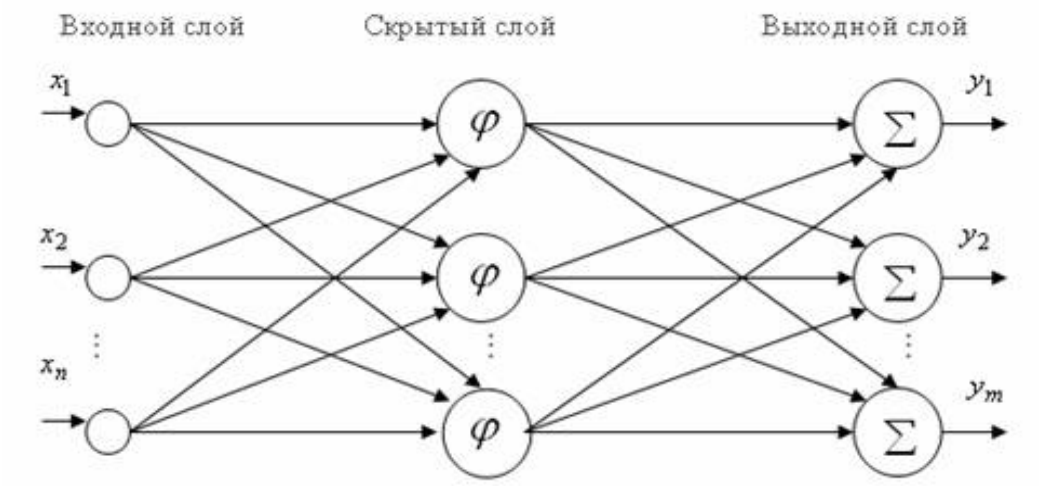

Рисунок 15 − Сеть прямого распространения с радиально-базисной функцией

Входной слой — это вектор из n элементов, подключенных напрямую ко всем узлам скрытого слоя. Каждый узел скрытого слоя представляет собой радиально-базисную функцию и преобразовывает расстояние от данного входного вектора до соответствующего ему «центра» по некоторому нелинейному закону [38].

Нейронные сети Хопфилда — полносвязные сети, где каждый нейрон является

входным до обучения, скрытым во время него и выходным после (рисунок 16).

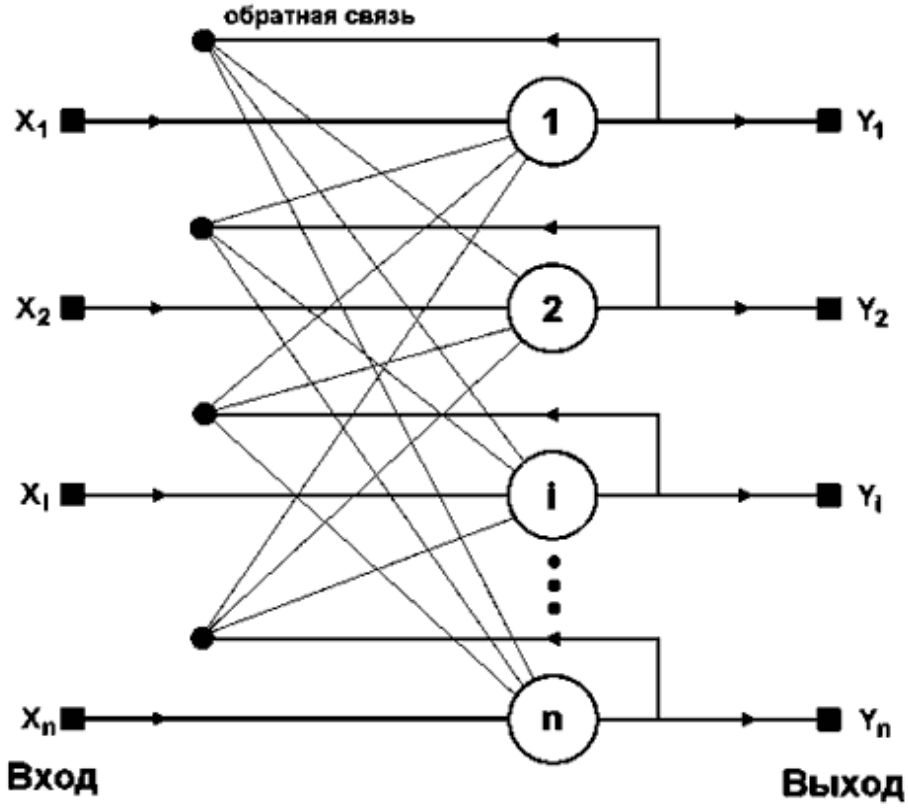

Рисунок 16 − Структурная схема сети Хопфилда

Матрица весов подбирается таким образом, чтобы все «запомненные» вектора являлись бы для нее собственными. Один раз обученная одному или нескольким образам система будет сходиться к одному из известных ей образов, так как только одно из этих состояний является стационарным [39].

Следующая известная модель — машина Больцмана (Boltzmann machines), которая во многом похожа на сеть Хопфилда, но в ней некоторые нейроны помечены как входные, а некоторые остаются скрытыми [40].

Входные нейроны становятся выходными, когда все нейроны в сети обновляют свои состояния (рисунок 17). Сначала весовые коэффициенты присваиваются случайным образом, затем происходит обучение методом обратного распространения, или в последнее время все чаще с помощью алгоритма Contrastive Divergence (когда градиент вычисляется при помощи Марковской цепи).

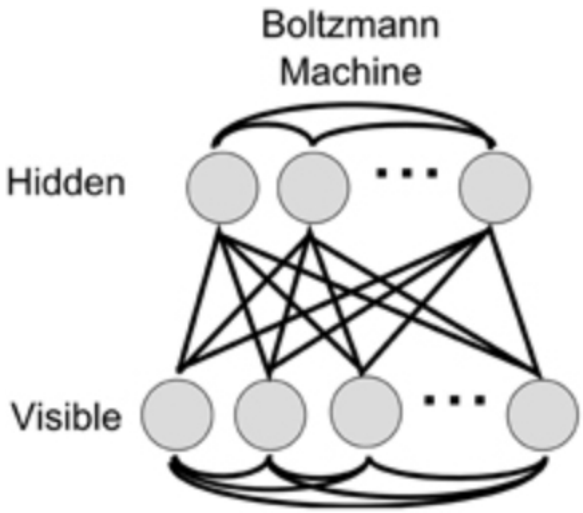

Рисунок 17 − Модель машины Больцмана

Машина Больцмана — стохастическая нейронная сеть, так как в обучении задействована цепь Маркова. Процесс обучения и работы здесь почти такой же, как в сети Хопфилда: нейронам присваивают определенные начальные состояния, а затем цепь начинает свободно функционировать. В процессе работы нейроны могут принимать любое состояние, и постоянно происходит перемещение между входными и скрытыми нейронами. Активация регулируется значением общей температуры, при понижении которой сокращается и энергия нейронов. Сокращение энергии вызывает стабилизацию нейронов. Таким образом, если температура задана правильно, система достигает равновесия [41].

2.1.3 Обучение искусственных нейронных сетей

Способность к обучению является главной отличительной чертой мозга. В рамках искусственных нейронных сетей обучение может представлять собой подбор правильной архитектуры сети и весовых коэффициентов связей, чтобы выполнение задачи, поставленной перед сетью, было эффективным. Искусственная нейронная сеть обычно подбирает веса, основываясь на данных, предоставленных ей для обучения, что даёт ей преимущество перед системами, где данные заложены изначально. В зависимости от степени процесса обучения, происходящему по определенному алгоритму, сеть учится эффективно реагировать на входные данные.

Существующие на данный момент методы обучения нейронных сетей делятся на следующие классы:

— детерминированный метод – такой, при котором параметры нейронной сети изменяются итеративно, в зависимости от значений параметров, входных

данных, а также основываясь на значениях выхода: тех, которые необходимо получить и тех, которые получились в процессе вычислений. Наиболее ярким примером служит метод обратного распространения ошибки.

— в стохастических методах параметры сети меняются без каких-либо закономерностей, случайно. Изменения принимаются лишь в том случае, когда они приводят к улучшению результатов. К известным проблемам данного метода обучения относится так называемая «ловушка локального минимума» (рисунок 18).

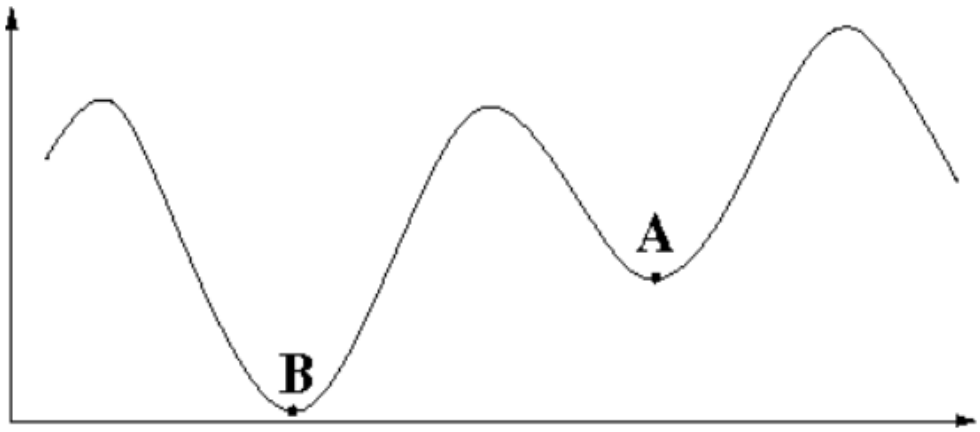

Рисунок 18 − Проблема локального минимума

Предположим, что значение ошибки изначально близко к точке M_1 , либо равно её значению. При малых произвольных шагах коррекции параметров любые отклонения от нее увеличат значение ошибки, и, следовательно, не будут приняты. Получается, что невозможно найти наименьшее значение ошибки в точке M_2 . Если же случайные шаги корректировки достаточно велики, ошибка не сможет установиться в одном из двух минимумов, так как будет меняться слишком резко [42].

В качестве одного из вариантов решения данной проблемы предполагается постепенное и монотонное уменьшение среднего размера случайных корректирующих шагов. С равной вероятностью значение ошибки принимает все возможные значения при больших произвольных шагах. В случае, когда размер случайных шагов корректировки изменяется постепенно, значение ошибки будет замедлять рост в точке M_2 в течение некоторого времени. При последующем уменьшении шага корректировки значение ошибки может «застревать» не только в точке M_2 но и в M_1 . Дальнейшее непрерывное уменьшение шага может привести к результату, когда преодолевается локальный минимум M_1 , при этом попадая в локальный минимум M_2 .

Все основные алгоритмы обучения строятся на функции, оценивающей качество работы нейронной сети. При этом имеем некий алгоритм, подстраивающий параметры системы, которые меняются в зависимости от значений оценки, которая была получена.

Теория об обучении нейронных сетей включает в себя рассмотрение трёх основных характеристик процесса обучения по примерам: сложность вычислений, сложность обучающей выборки и ёмкость [43]. Последняя представляет собой информацию о том, как много образцов из выборки сеть способна запомнить, а также о функциях и границах в принятии решений.

Сложность же обучающей выборки показывает, какое число образцов необходимо сети для получения способности к обобщению. Если набор примеров будет слишком малым, то сеть может оказаться «переобученной»: на обучающей выборке она будет хорошо функционировать, при этом плохо функционируя на тестовых образцах, подверженных сходному статистическому распределению [44].

Рассмотрим концептуальные подходы к обучению нейронных сетей.

Обучение с учителем.

Данная модель впервые была предложена Фрэнком Розенблаттом в 1957 году [45].

Процесс обучения с учителем полагает, что для каждого из множества входных векторов будет существовать целевой вектор, который и представляет из себя желаемый выход. Другими словами, задаётся массив пар векторов $\{(X^M, s^M)\}$, где $X^M \in X$ – описывающий условие задачи вектор, а $s^M \in Y$ – уже известное для заданного вектора X^M решение задачи. Такая пара называется обучающей.

Для обучения сети обычно требуется некоторое количество таких пар. Оно происходит следующим образом: вычисляется выход сети, основываясь на предоставленной информации о выходном векторе, после чего выход и соответствующий ему целевой вектор сравниваются. Разность этих значений (ошибка) считывается сетью с помощью обратной связи, после чего весовые коэффициенты меняются по принципу алгоритма, выбранного для минимизации ошибки. Иначе говоря, сеть меняет свои параметры так, чтобы получалось требуемое отображение $X \to Y$. При этом важно учитывать, что размер набора $\{(X^M, s^M)\}$ должен быть достаточно большим, дабы алгоритм сформировал желаемое отображение [46].

Векторы из обучающего набора представляются сети поочередно, после чего происходит вычисление ошибок и подстраивание весового коэффициента каждого вектора до того момента, пока общая ошибка на всём обучающем наборе не будет на достаточно низком уровне. Стоит отметить, что коррекция весовых коэффициентов

происходит только при условии ошибочного ответа [47].

Обучение без учителя.

С точки зрения биологии, обучение без учителя представляет собой более правдоподобную модель. В процессе исследования этой модели обучения Тойво Кохоненом [48], а также многими другими, установлено, что она не требует целевого вектора выходов, значит, нет необходимости в сравнении полученного выхода с изначально определенными эталонными ответами.

Массив обучающих данных представляет собой лишь входные векторы. Наилучшее значение для выхода определяется самим алгоритмом. Обычно он подстраивает весовые коэффициенты таким образом, чтобы на выходе векторы получались согласованными, другими словами, если входные векторы будут достаточно близкими, то их предъявление алгоритму должно давать схожие выходные векторы. Во время обучения сетью выделяются характерные черты обучающей выборки, на основе чего схожие векторы формируются в некоторые группы. При поступлении входного вектора из данной группы даёт какой либо конкретный вектор выхода, но до начала процесса обучения неясно, какой именно выход получится при направлении в сеть данной группы входных векторов выборки. То есть важен вопрос интерпретации выхода сети, основанный на процессе обучения. Это представляется вполне возможным, так как определить связь входа и выхода, установленную сетью, обычно несложно.

Метод обратного распространения ошибки.

При работе с многослойной нейронной сетью отсутствует информация об оптимальных величинах выходов нейронов каждого из слоёв, исключая последний. Соответственно, если нейронная сеть имеет более одного слоя, обучение, основанное лишь на значениях ошибок выходов из сети будет невозможным.

Данную проблему можно решить, определив наборы сигналов на выходе из каждого слоя сети, что представляет собой ресурсоемкий и не всегда возможный процесс. Кроме того, возможно подстраивать веса динамически, при этом обычно выбирают самые слабые связи и подвергаются незначительным изменениям, которые сохраняются лишь в том случае, когда сеть стала работать лучше, то есть уменьшилась ошибка на выходе. Данный эмпирический метод является трудоёмким вычислительным процессом, несмотря на простоту описываемых действий.

Наиболее привлекательным вариантом является распространение ошибки от выходов искусственной нейронной сети к её входам, что является обратным направлением относительно обычного рабочего режима сети. Подобный алгоритм носит название

«метод обратного распространения ошибки» [49].

При обучении нейронной сети методом обратного распространения ошибки по всем слоям сети производится два прохода: прямой и обратный. При первом производится приём входных данных, а также распространение их в направлении выходов. Во время второго этапа вычисляется ошибка, которая впоследствии распространяется обратно, а также корректируются весовые коэффициенты. Когда процесс обучения сети будет закончен, сигналы будут распространяться лишь в прямом направлении. Стоит отметить, что хотя обучение сети может длиться достаточно долго, но при этом уже обученная сеть способна быстро вычислять результаты по полученным данным. Для метода обратного распространения ошибки имеются различные модификации, направленные на уменьшение времени обучения сети.

2.1.4 Применение методов глубокого обучения в задачах распознавания изображений

Недостатком полносвязных нейронных сетей является большое количество весов для обучения. Даже для маленьких изображений нейронная сеть может содержать более миллиона весов. Следовательно, чтобы обучить такую сеть необходимо огромное количество входных данных. Другой недостаток связан с тем, что изображения представлены в виде вектора – одномерного массива. Таким образом происходит потеря топологической информации, важной для обработки изображений. Анализ соседних пикселей происходит только по горизонтали, но не по вертикали, а при работе с изображениями важно учитывать и то и другое. Чтобы устранить эти недостатки была предложена новая архитектура нейронных сетей, которая называется сверточные нейронные сети. Сейчас преимущественно сверточные нейронные сети используются при распознавании изображений [50].

Сверточные сети – это просто нейронные сети, в которых вместо общей операции умножения на матрицу, по крайней мере в одном слое, используется свертка.

Свертка – это особый вид линейной операции (рисунок 19). Если в полносвязной сети каждый нейрон подключается к каждому пикселю входного изображения, то в сверточных сетях нейрон получает на вход ограниченное количество пикселей – участки изображения $3*3$ пикселя $5*5$ пикселей и другие. Следующий нейрон работает со следующим участком изображения, который может частично пересекаться с участком соседнего нейрона.

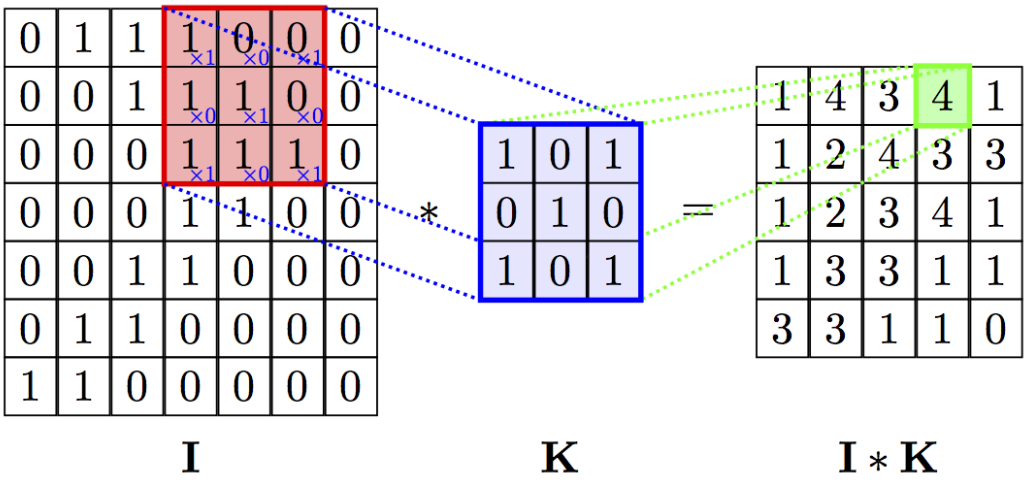

Рисунок 19 − Пример операции свертки

Со сверткой связаны три важные идеи, которые способствуют улучшению процесса машинного обучения: разреженные взаимодействия, разделение параметров и эквивариантные представления. Кроме того, свертка предоставляет средства для работы со входами переменного размера [51].

В слоях традиционной нейронной сети применяется умножение на матрицу параметров, в которой взаимодействие между каждым входным и каждым выходным блоками описывается отдельным параметром. Это означает, что каждый выходной блок взаимодействует с каждым входным блоком. Напротив, в сверточных нейронных сетях взаимодействия обычно разреженные (это свойство называют разреженной связностью, или разреженными весами). Достигается это за счет того, что значение ядра меньше значения входа. Например, входное изображение может содержать тысячи или миллионы пикселей, но небольшие значимые признаки, например границы или переходы, можно обнаружить с помощью ядра, охватывающего площади с размерами порядка десяти или сотни пикселей. Следовательно, нужно хранить меньше параметров, а это, в свою очередь, снижает требования модели к ресурсам памяти и повышает ее статистическую эффективность. Кроме того, для вычисления выхода потребуется меньше операций. Комплексное применение данных методов значительно повышает эффективность нейронной сети.

При работе с изображениями свертка создает двумерную карту появления определенных признаков во входном изображении. Если переместить объект во входном изображении, то его представление на выходе переместится на такую же величину. Это бывает нужно, когда мы знаем, что некоторая функция от небольшого числа пикселей полезна при применении к нескольким участкам входа. Например, в случае обработки

изображений полезно обнаруживать границы в первом слое сверточной сети (рисунок 20).

Рисунок 20 − Эффективность обнаружения границ

Одни и те же границы встречаются более-менее везде в изображении, поэтому имеет смысл разделять параметры по всему изображению. Но в некоторых случаях такое глобальное разделение параметров нежелательно. Например, если обрабатываются изображения, которые были кадрированы так, чтобы в центре оказалась рука человека в определенной конфигурации, то задача состоит в выделении разных признаков в разных точках – часть сети будет обрабатывать верхнюю часть руки в поисках границ пальцев и фона, а другая часть – искать границу локтевого сустава в нижней части изображения.

Свертка не эквивариантна относительно некоторых других преобразований, например масштабирования или поворота. Для обработки таких преобразований нужны другие механизмы.

Наконец, существуют типы данных, которые нельзя обработать с помощью нейронных сетей, определяемых путем умножения на матрицу фиксированной формы. Свертка позволяет обрабатывать некоторые данные такого рода.

2.1.5 Подходы к обучению сверточных нейронных сетей

Алгоритм обратного распространения ошибки может быть адаптирован для сверточных слоев в сверточных нейронных сетях. Алгоритм обратного распространения ошибки подобен используемым в полносвязных нейронных сетях. Отличие лишь в том, что частная производная ошибки по отношению к весу представляет собой сумму выражений связывающих правил для всех нейронов, на которые влияет этот вес. Это происходит из-за общих весов. Ошибка будет использоваться для вычисления частной

производной предыдущих слоев как функции веса по отношению к каждому выходу нейрона [52].

Поскольку полносвязные слои в сверточных нейронных сетях включают в себя большинство параметров сети, они склонны к переобучению. Переобучение – это излишне точное соответствие обученной нейронной сети набору данных, на которых она была обучена (рисунок 21).

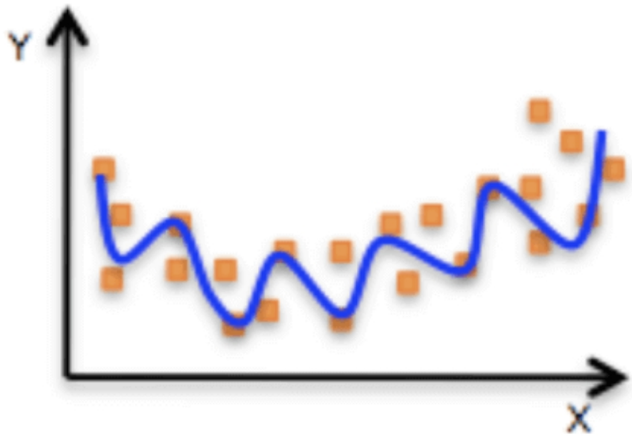

Рисунок 21 − Графическое представление переобучения (линейная регрессия)

При этом нейронная сеть теряет способность к обобщению [53]. Переобучение может проявиться если обучающих примеров недостаточно, маленькая часть нейронов станет более важной для большинства вычислений, а остальные нейроны будут избыточными.

Большие нейронные сети как правило очень медленные, что затрудняет предотвращение проблемы переобучения путем комбинирования прогнозов множества различных крупных нейронных сетей во время тестирования. Dropout — это один из методов решения данной проблемы. Ключевой идеей является случайное удаление юнитов вместе с их связями из нейронной сети во время обучения (рисунок 22).

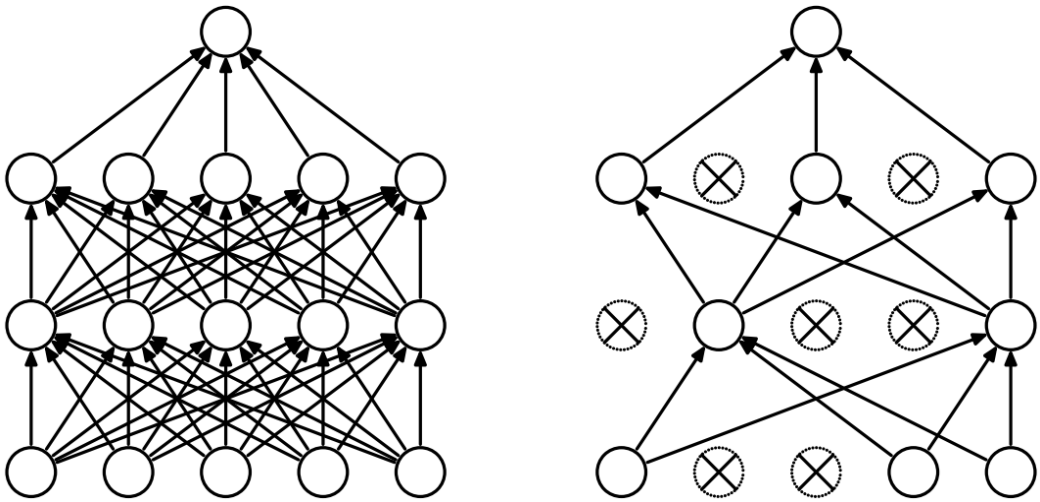

Рисунок 22 − Полносвязная сеть и сеть с применением метода Dropout

Это предотвращает адаптацию модели к набору данных, что, в свою очередь, помогает нейронной сети обрабатывать ошибки, не полагаясь на существование некоторых нейронов [54].

2.1.6 Краткий обзор фреймворков для задач глубокого обучения

В настоящее время существует большое число фреймворков для работы с тензорами и нейронными сетями, в том числе сверточными нейронными сетями.

Фреймворк глубокого обучения — это интерфейс, библиотека или инструмент, который позволяет разработчику легче и быстрее создавать модели глубокого обучения, не вдаваясь в детали базовых алгоритмов. Фреймворк обеспечивает четкий и лаконичный способ определения моделей с использованием набора предварительно созданных и оптимизированных компонентов.

Вместо того, чтобы писать сотни строк кода, мы можем использовать подходящую среду, чтобы быстро построить работающую модель.

Наиболее популярными фреймворками являются TensorFlow, Theano, PyTorch, CNTK, Keras, nnForge и другие (таблица 1).

Таблица 1: Характеристики программных средств глубокого обучения

№	Название	Язык	Открытый код	Архитектура
1	Tensor Flow	Python, C++	+	CPU, GPU, Mobile, Embedded, Web
2	Microsoft CNTK	C++	+	CPU, GPU
3	Caffe	C++	+	CPU, GPU
4	Theano	Python	+	CPU, GPU
5	PyTorch	Python, Lua	+	CPU, GPU, оптимизация для большого числа математических операций

Для реализации практической части работы необходимо выбрать наиболее подходящий фреймворк. Далее представлен краткий обзор наиболее популярных фреймворков.

Tensor Flow, разработанная компанией Google, — это надежная платформа с открытым исходным кодом, поддерживающая глубокое обучение, доступ к которой можно получить даже со смартфона.

Tensor Flow — это отличный инструмент для создания и разработки статистических программ. Поскольку фреймворк предлагает распределенное обучение — все модели ИИ будут обучаться намного эффективнее на любом уровне абстракции, который предпочитает пользователь [55].

Особенности:

— масштабируемый интерфейс для комфортного программирования;
— постоянное развитие платформы за счет открытого исходного кода;
— обширные и хорошо задокументированные мануалы.

Достоинства:

— интерфейс для работы с Python;
— способен развивать огромную вычислительную мощность;
— система использует вычислительную графическую абстракцию для создания моделей ИИ.

Недостатки:

— для принятия решения или прогнозирования, фреймворк передает входные данные через несколько узлов — этот процесс занимает много времени;
— в системе отсутствует множество уже существующих моделей ИИ.

Microsoft CNTK — это быстрый и универсальный фреймворк с открытым исходным кодом, основанный на нейронных сетях с поддержкой текста, сообщений и ремоделирования голоса. Платформа представляет из себя эффективную среду масштабирования.

Особенности:

— фреймворк хорошо оптимизирован, что обеспечивает высокую эффективность, масштабируемость, отличную скорость работы и высокоуровневую интеграцию;
— платформа включает в себя различные компоненты, такие как: настройки гиперпараметра, контроль моделей и их усиление, CNN, RNN и т. д.;
— фреймворк эффективно использует вычислительные мощности компьютера для обеспечения лучшей работоспособности фреймворка.

Достоинства:

— так как платформа поддерживает языки Python и C++, фреймворк позволяет работать с несколькими сервисами одновременно, что, в свою очередь, значительно ускоряет процесс обучения;
— фреймворк разрабатывался с учетом последних событий в мире искусственного интеллекта. Архитектура Microsft CNTK поддерживает GAN, RNN и CNN.

Недостатки:

— в фреймворке отсутствуют панель визуализации и поддержка мобильной архитектуры микропроцессоров ARM.

Caffe — платформа, включающая в себя предустановленные наборы обучаемых нейронных сетей. Этот фреймворк известен своими возможностями обработки изображений, также в платформу была включена поддержка пакета прикладного ПО MATLAB [56].

Особенности:

— все модели фреймворка имеют открытый исходный код;
— фреймворк обеспечивает высокую скорость и эффективность работы;
— сообщество, которое модифицирует код фреймворка.

Достоинства:

— поддержка языков C, C++ и Python;
— специализируется на решении различных вычислительных задач.

Недостатки:

— не способен обрабатывать комплексные массивы данных.

Theano — это расширение языка Python, позволяющее эффективно вычислять математические выражения, содержащие многомерные массивы. Theano разработана в лаборатории LISA для поддержки быстрой разработки алгоритмов машинного обучения. Библиотека реализована на языке Python, поддерживается на операционных системах Windows, Linux и Mac OS. В состав Theano входит компилятор, который переводит математические выражения, написанные на языке Python в эффективный код на C или CUDA [57].

Особенности:

— процесс оценки выражений быстрее из-за динамической генерации кода;

— обеспечивает превосходную точность, даже при минимальных значениях;

— модульное тестирование.

Достоинства:

— обеспечивает эффективную поддержку всех приложений с интенсивным использованием данных, но требует объединения с другими библиотеками;

— платформа отлично оптимизирована для работы как с CPU, так и с GPU.

Недостатки:

— для текущей версии не запланирован выпуск обновлений.

PyTorch — это фреймворк с открытым исходным кодом, предназначенный прежде всего для выполнения большого количества числовых операций. Платформа предлагает множество алгоритмов для быстрого развития сетей глубокого обучения. Данный фреймворк нашел применение в лабораториях искусственного интеллекта Facebook и Twitter [58].

Особенности:

— фреймворк включает себя множество подпрограмм для индексирования, фрагментирования, транспонирования с N-мерной моделью массива;

— платформа включает в себя процедуры оптимизации, в основном для числовых операций, основанных на нейронных сетях;

— платформа поддерживает GPU;

— фреймворк взаимодействует с iOS и Android.

Достоинства:

— высокая гибкость платформы за счет взаимодействия с различными языками программирования и интеграции с мобильными ОС;

— обеспечивает эффективное использование графического процессора;

— включает в себя уже существующие модели ИИ для обучения.

Недостатки:

— неструктурированная документация.

2.1.7 Особенности подготовки набора данных для обучения нейронной сети

Для обучения нейронной сети необходимо подготовить набор данных.

Данные для обучения должны быть репрезентативными. Обучающая выборка должна полно и разносторонне представлять описываемый объект, включать в себя различные возможные сценарии.

Чем точнее обучающая выборка аппроксимирует генеральную совокупность изображений, которые будут поступать на вход нейронной сети, тем выше предельно возможное качество обучения нейронной сети. Правильно составленная обучающая выборка – это часть конкретизации технического задания [59].

Обучающая выборка должна содержать изображения с различным освещением, сделанные с помощью разных моделей камер, что усложняет подбор необходимого числа примеров для обучения нейронной сети. Существует несколько возможных способов подготовки изображений для обучающей выборки. Одним из таких примеров может быть создание обучающей выборки из естественных изображений. Примеры для такой выборки создаются на основе реальных данных и состоит из следующих этапов:

— сбор данных – фотографирование, выделение части в видеопотоке;

— фильтрация – проверка на наличие распознаваемого объекта, уровня освещенности, соответствия объекта формальным требованиям;

— подготовка инструментария для разметки;

— разметка – выделение интересующих областей изображения;

— labeling — присвоение метки каждому классу.

Еще одним подходом подготовки изображений является искусственная генерация изображений – некоторое искусственное увеличение количества данных для обучающей выборки [60]. Смысл такого подхода состоит в выборе нескольких шаблонных примеров и создании на основе этих примеров новых изображений путем различных операций искажения или трансформации. Обычно используются следующие операции:

— геометрические преобразования;

— яркостные и цветовые преобразования;

— замена фона;

— специальные, характерные для конкретной задачи искажения.

Генерация искажений осуществляется с помощью различных библиотек для

работы с изображениями или специальных программ, которые позволяют создавать искусственные изображения.

Такой подход не требует большого количества времени, поскольку нет необходимости вручную размечать и собирать большие объемы данных. Главным недостатком подхода является слабая связь качества обучения на сгенерированных данных с работой на реальных данных.

Схожий подход – это генерация примеров на основе реальных естественных изображений [61]. При таком подходе вместо шаблонных изображений используются реальные изображения. Чтобы определить, какие именно искажения необходимо применять часть данных используется для валидации. По ним можно оценивать наиболее часто встречающиеся типы ошибок и добавлять изображения с соответствующими искажениями в обучающую выборку.

Такой метод требует высоких временных затрат, но позволяет подготовить достаточно большую обучающую выборку. Сложность такого подхода заключается в правильном определении параметров, по которым строятся искажения изображений. Параметры искажения можно подбирать путем анализа ошибочных параметров нейронной сети, обученной на реальных данных.

В каждом конкретном случае обучения необходимо подбирать индивидуальные параметры обучающей выборки в зависимости от наличия и количества данных, условий получения изображений и возможных искажений на реальных изображениях.

2.1.8 Методы оценки качества работы классификатора

Определение целей в терминах метрики ошибок – обязательный первый шаг, потому что от выбранной метрики ошибок зависят все последующие действия. Кроме того, следует понимать, какой уровень качества желателен.

Для большинства приложений невозможно достичь полной безошибочности. Байесовская частота ошибок дает минимальную частоту, на которую мы можем рассчитывать, даже если объем обучающих данных бесконечен и есть возможность восстановить истинное распределение вероятности [62]. Причина может заключаться в том, что входные признаки содержат неполную информацию о выходной величине, или в том, что система принципиально стохастическая. Кроме того, критическим ограничением является объем обучающих данных, который конечен.

В академических исследованиях обычно имеется некоторая оценка частоты ошибок, основанная на ранее опубликованных результатах эталонного тестирования. При

разработке коммерческого продукта у разработчика должно быть представление о том, какая частота ошибок приемлема, чтобы приложение можно было считать безопасным, рентабельным или привлекательным для пользователей. После того как реалистические требования к частоте ошибок сформулированы, в основе всех дальнейших решений должно быть стремление к достижению этой частоты.

Помимо целевого значения показателя качества, есть еще вопрос о выборе показателя. Для измерения эффективности полного приложения, включающего компонент машинного обучения (классификатор в данном случае), есть несколько показателей качества.

Иногда одни ошибки обходятся гораздо дороже других. Например, в системе обнаружения почтового спама возможны ошибки двух типов: неправильная классификация нормального сообщения как спама и неправильный пропуск спама во входящую почту. Заблокировать нормальное сообщение гораздо хуже, чем пропустить сомнительное. Вместо того чтобы измерять частоту ошибок классификатора спама, хотелось бы измерить некую полную стоимость, в которой стоимость блокировки нормальных сообщений выше стоимости пропуска спама [63].

Иногда требуется обучить бинарный классификатор, обнаруживающий редкие события, например тест для диагностики редкого заболевания. Предположим, что заболевание встречается у одного человека на миллион. Мы легко можем получить для этой задачи распознавания верность 99.9999, просто заставив классификатор всегда сообщать об отсутствии заболевания. Очевидно, что верность не годится для оценки качества такой системы. Решить проблему можно, если измерять не верность, а точность (precision) и полноту (recall).

Точностью называется доля правильных ответов модели, а полнотой – доля обнаруженных истинных событий. Детектор, утверждающий, что нет ни одного больного, имеет идеальную точность, но нулевую полноту. Детектор, утверждающий, что больны все, достигает идеальной полноты, но его точность равна процентной доле людей, страдающих заболеванием (0.0001 в случае, когда заболеванием страдает один человек на миллион).

При использовании точности и полноты часто строят ТП-кривую, когда по оси y откладывается точность, а по оси x – полнота (рисунок 23).

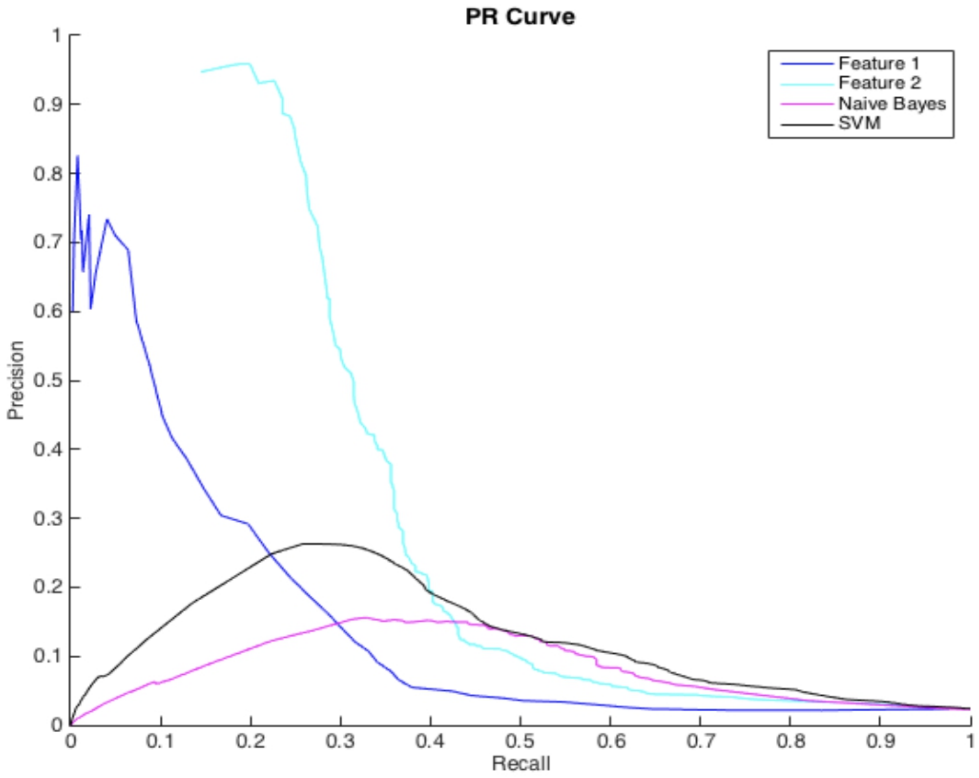

Рисунок 23 − Пример ТП-кривой

Порождаемая классификатором оценка выше, если подлежащее обнаружению событие действительно произошло.

Изменяя значение порога, можно отдавать предпочтение либо точности, либо полноте. Во многих случаях желательно охарактеризовать качество классификатора одним числом, а не кривой. Для этого точность p и полнота r объединяются в F-меру:

$$F = \frac{2 * p * r}{p + r}$$

Важно заранее определить, какой показатель качества будет улучшаться, а затем сконцентрироваться на нем. Не имея четко сформулированных целей, трудно сказать, принесло некое изменение системы успех или нет.

2.2. Выбор технологии для программной реализации алгоритма классификации жестовых команд

2.2.1 Разработка технического задания

Разработка программного модуля детектирования, трекинга и классификации статических жестов является основой для комплексной системы взаимодействия человека и роботизированных систем, которые должны правильно понимать человеческие жесты и

выполнять соответствующие команды в достаточной степени точности. Создание эффективных каналов взаимодействия, в том числе на базе жестовых команд, может освободить людей от тяжелых и потенциально опасных задач.

Программный модуль должен включать в себя следующие вспомогательные модули:

— модуль предварительной обработки изображений;

— модуль детектирования региона жестов рук на изображении;

— модуль классификации жестов рук на заданном регионе изображения;

— инструментальные модули для обучения нейронной сети и обработки данных.

В результате работы вспомогательных модулей, основной программный модуль должен осуществлять отображение информации о наличии жеста, обнаруженного на изображении, его принадлежность к определенному классу и значение.

Схема работы и обмена данными для компонентов системы представлена на рисунке 24.

Рисунок 24 - Схема обработки данных

Разрабатываемые программные модули должны соответствовать следующим требованиям:

— программный модуль должен работать в двух режимах: в режиме симуляции (с имеющимися видеофайлами) и в реальном времени (видеопоток с камеры);

— обучаемость – программный модуль должен иметь возможность обучения на новых выборках данных;

— портируемость — программный модуль должен поддерживать сценарии работы на разных целевых устройствах: информационно-развлекательный модуль автомобиля, компьютер, телевизор;

— энергоэффективность - система должна быть компактной и потреблять малое количества энергии в работе при максимальных нагрузках.

2.2.2 Выбор фреймворка для разработки модели нейронной сети

Для разработки нейронной сети был использован фреймворк машинного обучения PyTorch. Выбор обусловлен наличием открытого исходного кода, документации, возможностью программирования с использованием языка Python и рядом других параметров, описанных в разделе 2.1.6.

PyTorch прост в установке, имеет возможность интеграции с библиотекой математических вычислений NumPy, поддерживает множество инструментов и дополнений для упрощения и ускорения процесса разработки моделей нейронной сети.

Следующие достоинства фреймворка PyTorch стали решающими на этапе выбора инструментов для разработки:

— глубокая интеграция с языком Python и его стандартными библиотеками;

— вычислительный граф в PyTorch определяется во время выполнения, и, следовательно, многие популярные инструменты Python можно использовать в PyTorch, что является огромным преимуществом, потому что теперь такие инструменты отладки Python как pdb, ipdb и PyCharm, могут использоваться для отладки кода;

— PyTorch очень прост в использовании и дает возможность манипулировать вычислительными графами «на лету»;

— PyTorch значительно проще в освоении, чем любой другой фреймворк глубокого обучения, потому что он не сильно отличается от многих традиционных программных практик;

— PyTorch обладает одной из самых важных функций, известной как декларативный параллелизм данных (declarative data parallelism). Эта функция позволяет использовать torch.nn.DataParallel, чтобы обернуть любой модуль, который будет распараллелен по размеру пакета, а функция DataParallel поможет легко использовать несколько графических процессоров.

2.3 Выводы

В разделе приводится описание и анализ современных методов и алгоритмов распознавания объектов на изображении, производится обзор и сравнение популярных библиотек машинного обучения содержащих рассматриваемые методы и

соответствующий набор инструментов, а также осуществляется выбор метода и библиотеки для разработки программного обеспечения для детектирования, трекинга и классификации жестов.

3 СПЕЦИАЛЬНЫЙ РАЗДЕЛ

3.1 Программная реализация алгоритма распознавания жестовых команд

Программное обеспечение для задач детектирования, трекинга и классификации статических жестов состоит из следующих программных модулей:

— модуль предварительной обработки изображений;

— модуль детектирования региона жестов рук на изображении;

— модуль обучения классификатора на базе нейронной сети;

— модуль классификации жестов рук на заданном регионе изображения;

— модуль визуализации результата работы классификатора и диагностической информации;

— модуль извлечения изображений из видеопотока и генерации видеопотока на основе изображений.

Общий алгоритм работы системы детектирования, классификации и трекинга ручных жестов представлен на рисунке 25.

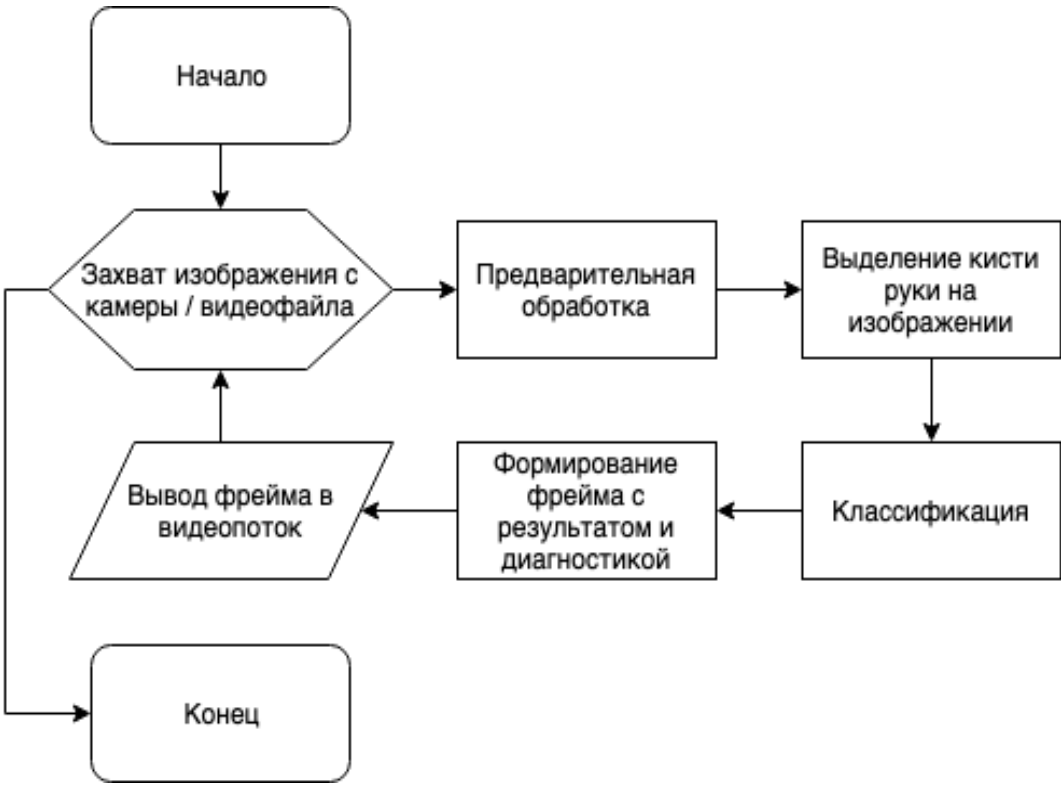

Рисунок 25 − Общий алгоритм работы системы

Программная реализация алгоритма является прикладным применением

нейронных сетей к задаче классификации изображений.

3.1.1 Предварительная обработка изображений и детектирование кисти руки

В задаче распознавания жестов на основе изображений, жесты состоят из фрагментов изображений рук. Поэтому использование этих фрагментов в качестве визуальных признаков при идентификации жестов вполне разумно.

Цвет — это простая визуальная функция для идентификации жестов по фоновой информации. Однако эффективность работы систем детектирования жестов рук на основе цветовой информации сильно зависит от освещения и теней.

Однако использование цветовых масок для таких задач оправдано ввиду относительной простоты самого метода, особенно на стадии создания прототипа [64] [65] [66].

Процесс нахождения региона интереса (Region of interest, ROI) — область кисти руки в данном случае, состоит из шагов, представленных на рисунке 26.

Рисунок 26 − Процесс нахождения региона кисти руки

Процесс нахождения ROI:

— первичное размытие изображения (было применено размытие по Гауссу с размером ядра, равным 3), чтобы избавиться от цветовых шумов;
— преобразование цветового пространства изображения из RGB в HSV [67];
— определение верхней и нижней границы интенсивности пикселей HSV, которые следует рассматривать как кожу в регионе кисти руки;
— удаление бинарных шумов при помощи функций эрозии и растяжений цветовой маски с эллиптическим ядром (функционал библиотеки OpenCV);
— поиск и выделение региона кисти руки (для упрощения задачи в данном случае ROI — это область с наибольшим количеством соседних белых пикселей);
— регион изображения готов для задачи классификации.

Результат преобразований и регион кисти руки представлены на рисунке 27.

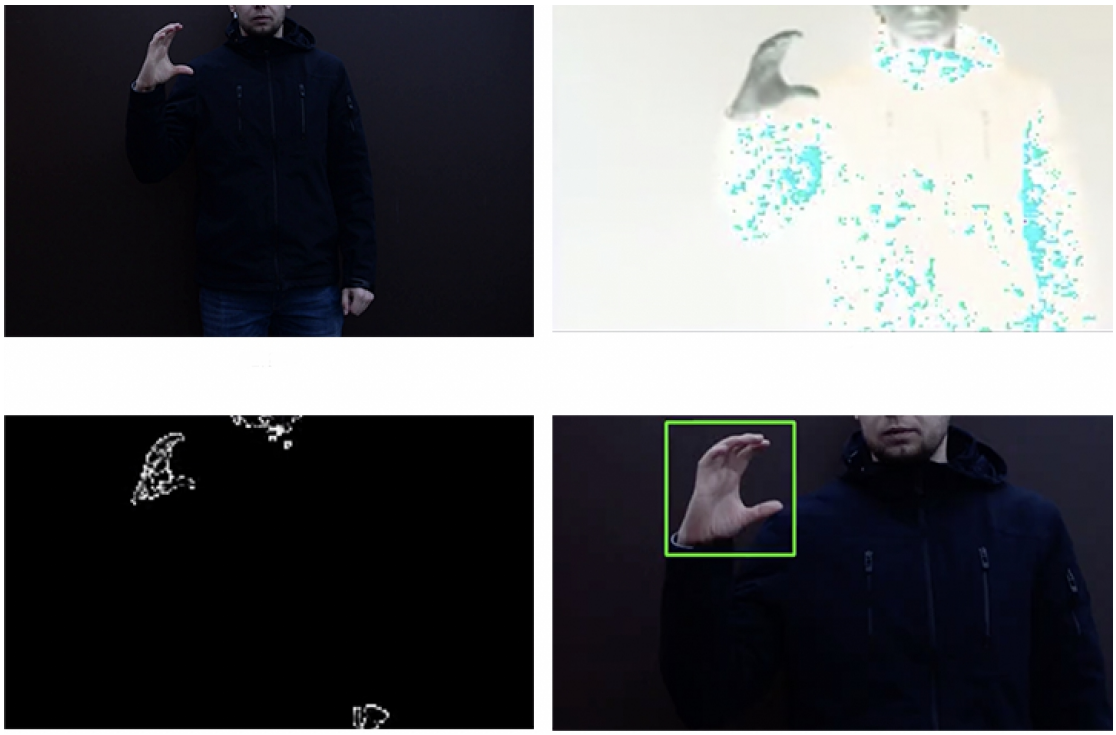

Рисунок 27 − Обнаружение региона руки с использованием маски цвета кожи

3.1.2 Подготовка набора данных для классификатора

Для обучения, кросс-валидации (перекрестной проверки) и тестирования классификатора был разработан оригинальный набор данных. Набор данных включает 2071 изображение рук в определенной жестовой конфигурации. Изображения были сделаны в разных условиях освещения и на небольшом расстоянии от объекта (0,4 - 0,7

метра). Каждая конфигурация руки является классом и соответствует определенному статическому жесту русского языка жестов. Всего представлено 10 классов (рисунок 28).

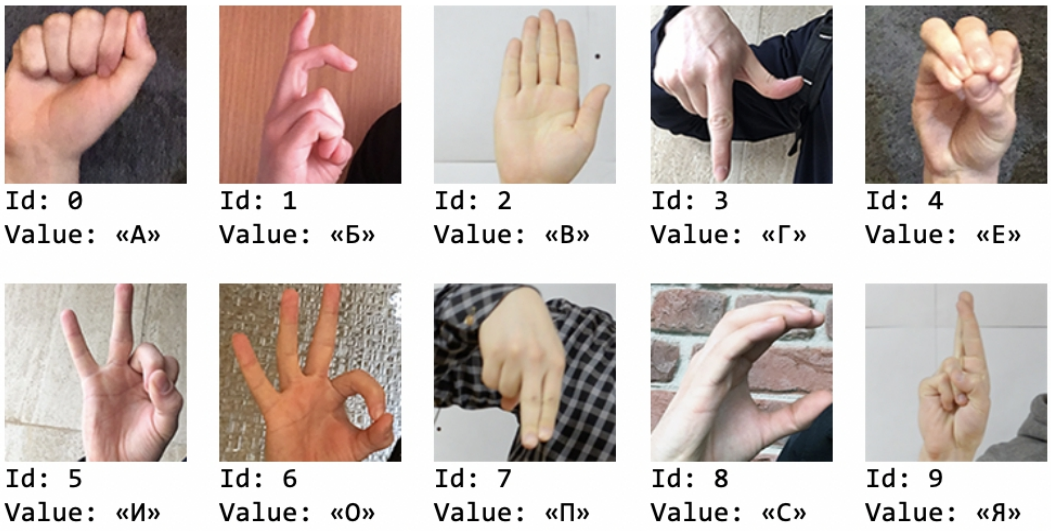

Рисунок 28 − Случайные изображения из набора данных

Изображения были сделаны с разными людьми, уровнем освещения и фоном.

Перед отправкой данных в классификатор необходимо выполнить предварительную обработку изображений — преобразования, которые помогут избавиться от второстепенных характеристик для классификатора, что, в свою очередь, повысит производительность классификатора. Предварительная обработка изображений уменьшает количество маловажных деталей (например, информация о цвете из пространства RGB) [68].

В таблице 2 представлены параметры изображения из набора данных до и после предварительной обработки.

Таблица 2: Параметры изображений из набора данных: до и после обработки

Параметр	Оригинальное изображение	Обработанное изображение
Формат	PNG	PNG
Ширина (пиксели)	128	64
Высота (пиксели)	128	64
Пространство	RGB	Grayscale
Глубина	3	1

В задаче классификации жестов рук, цвета не имеют отношения к идентификации объекта. В этом случае преобразование цветного изображения в изображение в градациях

серого не будет иметь значения, поскольку в конечном итоге модель будет учиться на основе геометрии, представленной на изображении (рисунок 29). Бинаризация изображения поможет повысить резкость изображения путем определения светлых и темных областей.

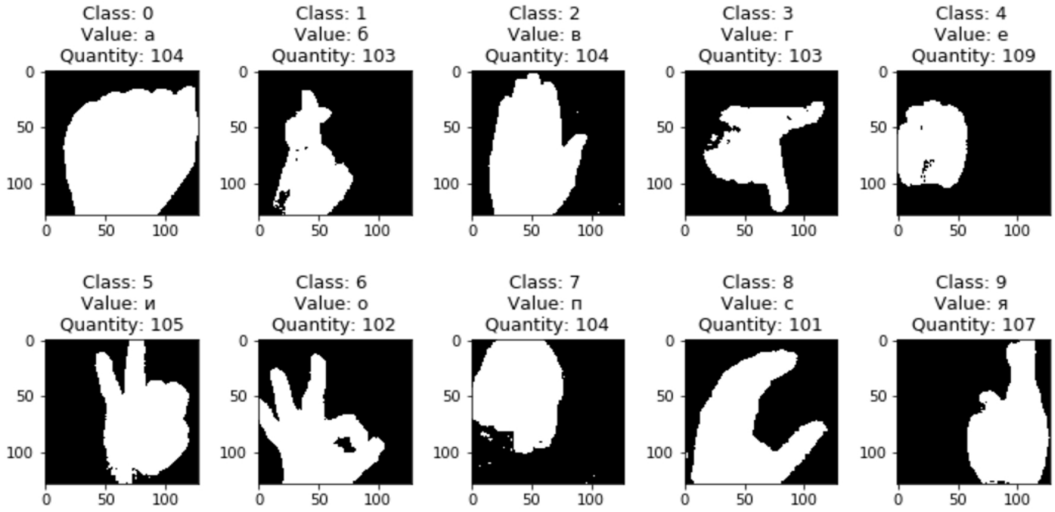

Рисунок 29 − Бинаризация изображений

Завершающим этапом предварительной обработки является нормализация. Значения пикселей находятся в диапазоне от 1 до 255, но для правильной работы нейронной сети требуются входные значения в диапазоне от 0 до 1. Для этого выполнено преобразование с использованием формулы $N = \frac{P}{P_{max}}$, где N — значение пикселя после нормализации, P — значение нормализованного пикселя, P_{max} — максимальное значение диапазона.

В тех случаях, когда набор данных небольшой, требуется больше синтетических данных. Для этого используются методы аугментации (расширения) данных. Общеизвестно, что чем больше данных имеет доступ к алгоритму глубокого обучения, тем эффективнее он может быть. Даже когда данные имеют низкое качество, алгоритмы могут на самом деле работать лучше, если полезные данные могут быть извлечены моделью из исходного набора данных [69].

Для получения синтетических изображений были применены следующие операции (рисунок 30):

— случайное вращение (все направления, +/- 10 градусов);

— случайное изменение размера с масштабированием.

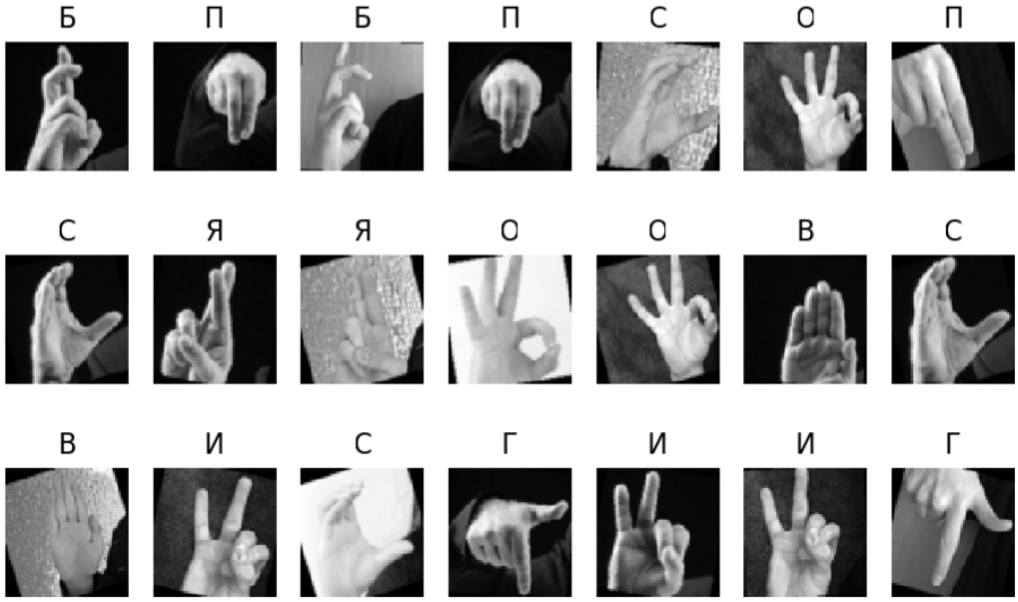

Рисунок 30 − Случайные изображения из набора данных

Чтобы минимизировать возможность переобучения классификатора, набор данных был разделен на 3 части [70]:

— обучающий набор, включающий в себя около 80% исходного набора данных (1671 изображение);
— тестовый набор, включающий в себя около 20% исходного набора данных (400 изображений), который используется для оценки классификатора (эти образцы не используются во время обучения и перекрестной проверки);
— набор для перекрестной проверки (кросс-валидации) содержит около 20% обучающего набора (300 изображений).

3.1.3 Архитектура классификатора на основе сверточной нейронной сети для распознавания статических жестовых команд

В этом разделе описана архитектура нейронной сети, применяемая для задачи классификации жестовых команд на абстрактном уровне. В качестве эталона были взяты две модели: архитектура LeNet-5 [71] (рисунок 31) и классификатор статических жестов, который был представлен на конференции AIST в 2018 году [72] (рисунок 32). Качественная оценка приведена в пункте 3.2.1.

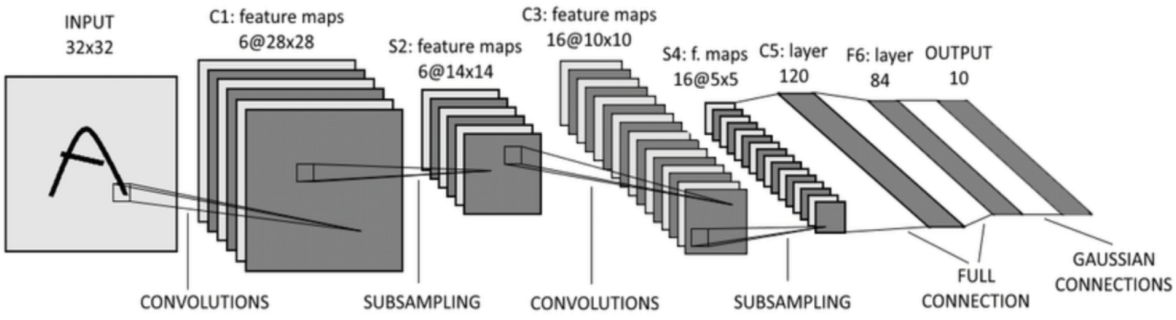

Рисунок 31 − Архитектура LeNet-5

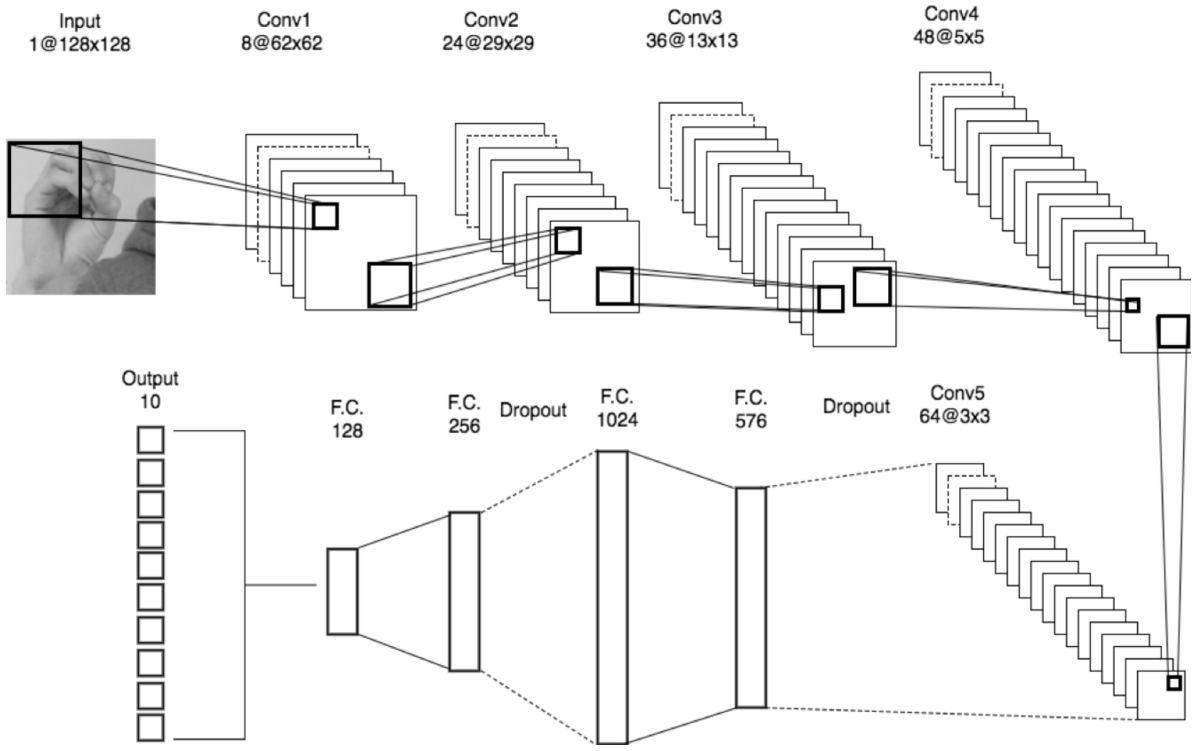

Рисунок 32 − Архитектура классификатора статических жестов (AIST 2018)

Новая усовершенствованная архитектура сети по сравнению с эталонными архитектурами включает в себя более глубокие свертки в сверточных слоях и значительно большее число элементов в полносвязных слоях (рисунок 33).

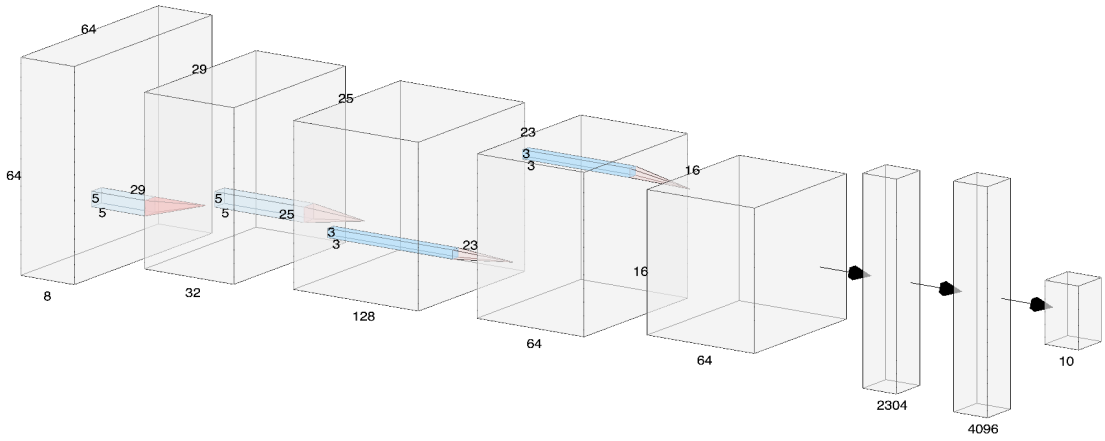

Рисунок 33 − Улучшенная архитектура сети для задачи классификации жестов

Нейронная сеть состоит из следующих слоев:

— входной слой (Input), размер вывода: 1*64*64;
— сверточный слой (Conv1), размер вывода: 8*64*64, размер окна свертки: 5*5, функция активации – ReLU (Rectified Linear Unit) [73];
— сверточный слой (Conv2), размер вывода: 32*29*29, размер окна свертки: 5*5, функция активации – ReLU;
— сверточный слой (Conv3), размер вывода: 128*25*25, размер окна свертки: 3*3, функция активации – ReLU;
— сверточный слой (Conv4), размер вывода: 64*23*23, размер окна свертки: 3*3, функция активации – ReLU;
— случайное удаление юнитов (Dropout 1), вероятность удаления = 0.25;
— сверточный слой (Conv5), размер вывода: 64*16*16, размер окна свертки: 3*3, функция активации – ReLU;
— случайное удаление юнитов (Dropout 2), вероятность удаления = 0.25;
— полносвязный слой 1, размер: 2304;
— случайное удаление юнитов (Dropout 3), вероятность удаления = 0.25;
— полносвязный слой 2, размер: 4096;
— выходной слой, размер: 10, функция активации – softmax.

Нейронная сеть разрабатывалась с использованием фреймворка PyTorch со следующими гиперпараметрами после оптимизации: скорость обучения составляет 0.001, размер одной связки (batch) составляет 20 элементов, количество эпох составляет 24, метрики — точность (accuracy), потери (loss), тип оптимизатора — Adam.

Динамика обучения, а именно изменение параметра потерь (loss), представлен на рисунке 34, где голубым цветом показан параметр для обучающего набора данных, а оранжевым — для валидационного набора данных.

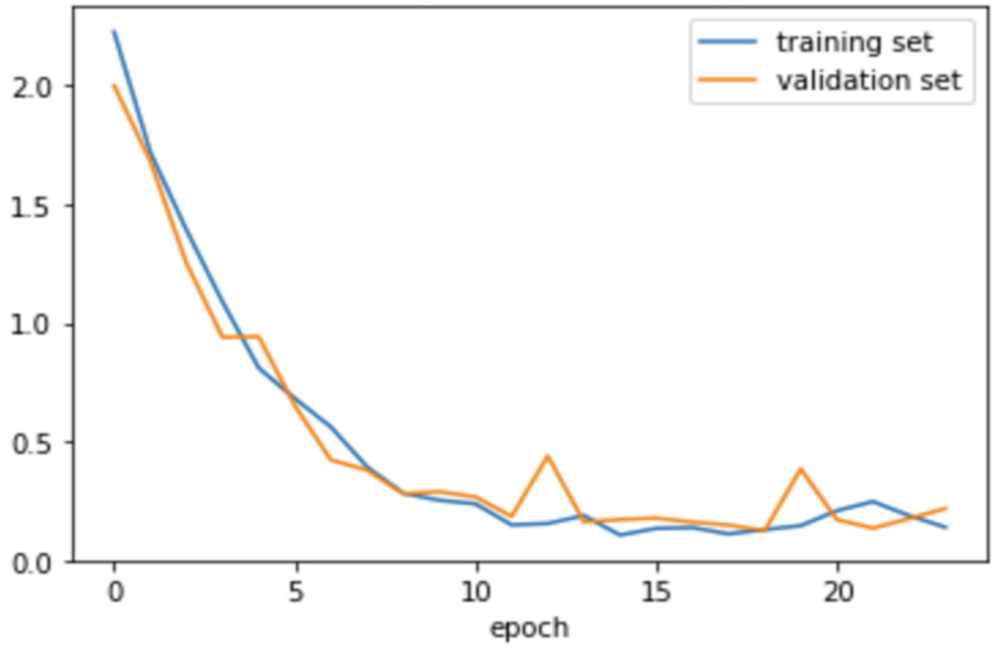

Рисунок 34 − Динамика изменения параметра loss при обучении классификатора

3.1.4 Ресурсные требования к программной и аппаратной реализации

Программный модуль системы включает основные модули (распознавания, детектирования, трекинга) и дополнительные модули (подготовка обучающей выборки, обучение нейронной сети). Язык разработки — Python версии 3.5.

Программное обеспечение:

— Python 3.6;
— ОС семейства Linux;
— дистрибутив Anaconda / Docker;
— Jupyter Notebook.

Зависимости:

— matplotlib – библиотека для построения графиков;
— openCV – библиотека алгоритмов компьютерного зрения;
— torch – библиотека для выполнения научных расчетов, включающая методы машинного обучения, такие как нейронные сети и глубокое обучение;
— numpy – библиотека для выполнения математических вычислений;
— scipy – библиотека для выполнения научных и инженерных расчетов;

— cpickle – библиотека сериализации и десериализации объектов в python;

— PIL – библиотека для работы с изображениями в формате pil;

— moviepy – библиотека для раскадровки видеопотоков.

Аппаратное обеспечение:

— компьютер с графическим процессором nvidia;

— web-камера (опция, для работе в режиме реального времени);

3.2 Экспериментальные исследования

В качестве эталона были взяты две модели: архитектура LeNet-5 и классификатор статических жестов, который был представлен на конференции AIST в 2018 году. Для оценки точности взяты параметры потери (loss) и точности (accuracy).

3.2.1 Сравнительный анализ работы классификаторов жестовых команд

Основным недостатком LeNet-5 является переобучаемость в некоторых случаях и отсутствие встроенного механизма, позволяющего избежать этого. Таким образом, эталонная архитектура была улучшена путем добавления выпадающих слоев со случайным удалением юнитов (dropout). Улучшенная архитектура LeNet-5 для задачи классификации жестов была представлена на конференции AIST в 2018 году [72]. Основным недостатком этой модели является сравнительно низкая производительность на тестовых данных (91,38%).

Новая усовершенствованная архитектура сети по сравнению с эталонными архитектурами включает в себя более глубокие свертки в сверточных слоях и значительно большее число элементов в полносвязных слоях (раздел 3.1.3).

Процедуры обучения и тестирования были выполнены на эталонных классификаторах. Результаты эталонных классификаторов и улучшенного классификатора приведены в таблице 3.

Таблица 3: Результаты классификации (контрольные показатели)

Архитектура классификатора	Потери при обучении (Training Loss)	Точность при обучении (Training accuracy)	Потери на тестовых данных (Test Loss)	Точность на тестовых данных (Test accuracy)
LeNet-5	**0.0681**	**0.9985**	0.5134	0.8708
Классификатор статических жестов (1 поколение)	0.1411	0.9369	0.2385	0.9138
Улучшенный классификатор статических жестов (2 поколение)	0.1310	0.9433	**0.2119**	**0.9360**

3.3 Выводы

Представленная версия усовершенствованного классификатора демонстрирует точность классификации в 93,6% по тестовому набору данных, которая превосходит результаты предыдущей версии — 91,38% и классификатор LeNet-5 — 87,08%. Полученные результаты точности являются достаточной основой для разработки промышленного прототипа интерфейса управления при помощи жестовых команд и дальнейших исследований в этом направлении.

ЗАКЛЮЧЕНИЕ

Актуальность проведенных исследований обусловлена проблемой взаимодействия между людьми и машинами (роботами), над решением которой работают такие известные компании как Boston Dynamics, Kuka Robotics, Google и другие. Основная идея в том, что создание эффективных каналов взаимодействия, в том числе на базе жестовых команд, может освободить людей от тяжелых и потенциально опасных задач.

В рамках диссертации реализованы следующие компоненты системы:

— программное обеспечение для предварительной подготовки данных и обучения классификатора на базе сверточной нейронной сети;
— программное обеспечение для поиска региона ддя классификации (ROI) на изображении;
— программное обеспечение для классификации статических жестов и визуализации результата;
— программная документация.

Основными задачами исследования являлись:

— анализ методов классификации изображений, основанных на использовании искусственных нейронных сетей;
— выбор оптимального метода выделения информативной части на изображениях;
— выбор оптимальной архитектуры нейронной сети для распознавания статических жестов в видеопотоке;
— оптимизация параметров используемой нейронной сети;
— выбор методов и параметров аугментации (расширения) набора данных для обучения классификатора;
— реализация и исследование работоспособности и эффективности алгоритма распознавания статических жестов, основанного на использовании искусственной нейронной сети.

В диссертационной работе при решении поставленных задач использованы методы теории искусственных нейронных сетей, математического моделирования, теории вероятностей и математической статистики. Для разработки программных компонентов были использованы алгоритмы компьютерного зрения, а именно:

— морфологические преобразования изображения;
— поиск объекта (ROI — Region of interest) по цвету и контуру;
— изменение размеров (Resize) и выделение фрагментов изображений.

В качестве одного из ключевых компонентов программного модуля были

использованы алгоритмы машинного обучения, а именно сверточные нейронные сети. Также были разработаны вспомогательные программные компоненты для предварительной обработки данных, обучения нейронной сети и проверки точности классификатора (параметры — loss, accuracy).

В результате исследования был разработан и опубликован уникальный набор данных из 10 классов и более 2000 уникальных изображений, а также программный комплекс для обнаружения, отслеживания и классификации статических жестов русского языка жестов в видеопотоке с использованием методов компьютерного зрения и глубокого обучения (рисунок 35).

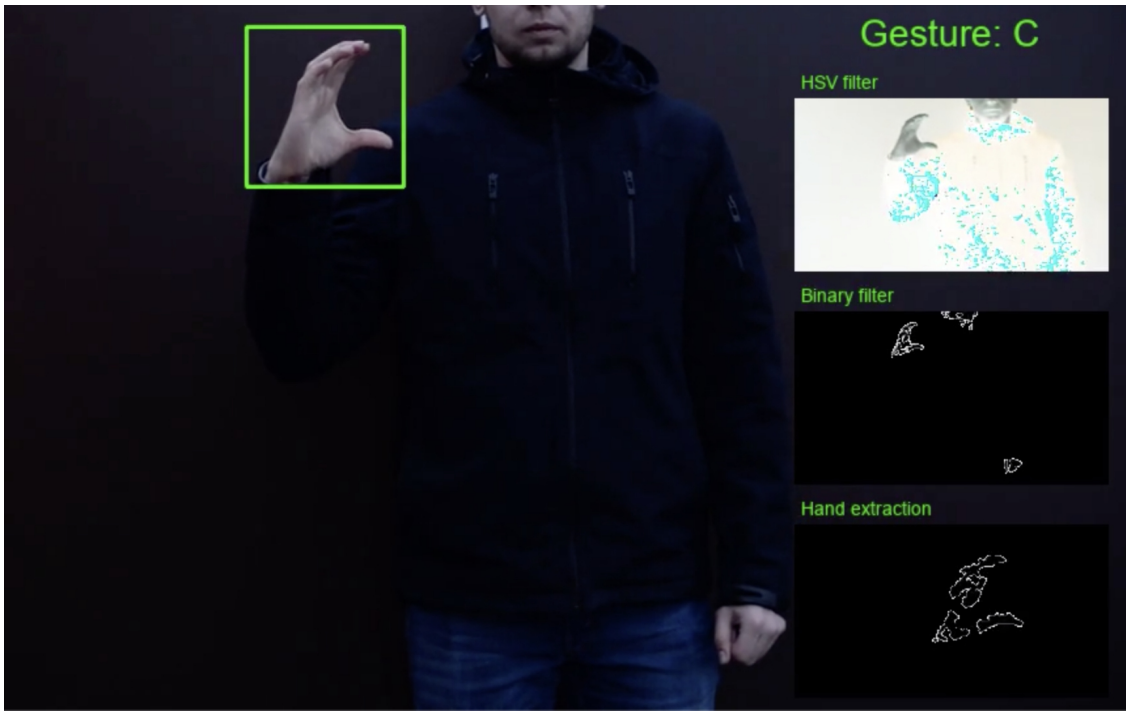

Рисунок 35 - Интерфейс системы жестового управления

Решение включает в себя модуль обнаружения рук, который использует цветовую маску, модуль отслеживания жестов, модуль классификации статических жестов в обнаруженной области изображения на основе сверточной нейронной сети, а также вспомогательный модуль предварительной обработки изображений и модуль расширения набора данных. Для разработки архитектуры нейронной сети была использована среда PyTorch.

Представленная версия усовершенствованного классификатора демонстрирует точность классификации в 93,6% по тестовому набору данных, которая превосходит результаты предыдущей версии — 91,38% и классификатор LeNet-5 — 87,08%.

Результаты точности являются достаточной основой для разработки промышленного прототипа интерфейса управления при помощи жестовых команд и дальнейших исследований в этом направлении, что являлось одной из основных целей проекта.

Другой важной целью является обеспечение возможности отдельным разработчикам и исследователям использовать разрабатываемые алгоритмы и набор данных в целях доработки и использования в собственных исследованиях. Для этого весь исходный код проекта, набор данных и обученные классификаторы были опубликованы на GitHub и уже являются основой для исследований научного и IT-сообществ (рисунок 36). Тексты научных статей опубликованы в открытом доступе на ресурсе ResearchGate.

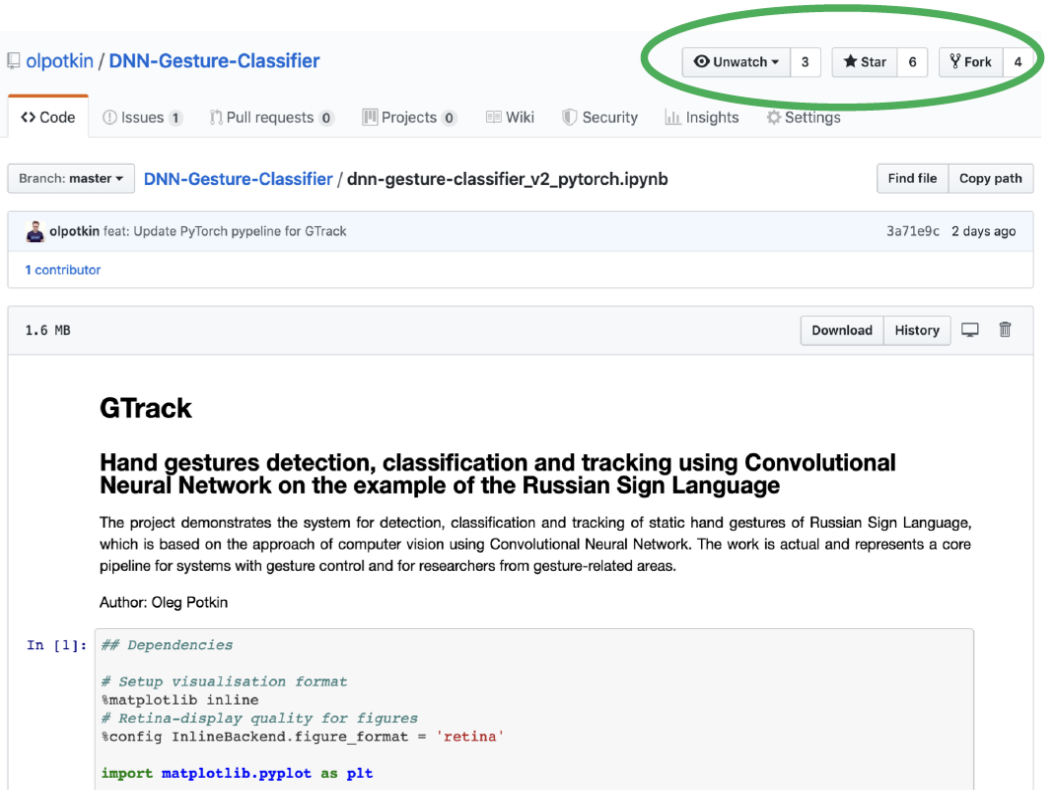

Рисунок 36 − Исходный код проекта на GitHub и статистика репозитория

Тема и материалы диссертации были представлена в трех научных публикациях, опубликованных в «CEUR-WS series»:

— статья «Static gestures classification using Convolutional Neural Networks on the example of the Russian Sign Language» для конференции AIST-2018;

— постер с графическими материалами для статьи «Static gestures classification using Convolutional Neural Networks on the example of the Russian Sign Language» для конкурса постеров на конференции AIST-2018;

— статья «Hand gestures detection, tracking and classification using Convolutional

Neural Network» для конференции AIST-2019;

В следующем поколении проекта будет применен подход семантической сегментации для задачи классификации с полносвязной сверточной сетью (Fully Convolutional Networks [74]). Предполагается иной способ маркировки набора данных и требуется больше данных. Для решения этой проблемы будут создаваться новые изображения с использованием синтетических данных (случайное преобразование перспективы, случайный шум, генерирующие состязательные сети — Generative Adversarial Nets [75] и другие).

СПИСОК ЛИТЕРАТУРЫ

[1] J. Krüger, T. Lien, A. Verl, Cooperation of human and machines in assembly lines, CIRP Annals-Manufacturing Technology, 2009, 628-646

[2] S.A. Green, M. Billinghurst, X. Chen, G. Chase, Human-robot collaboration: A literature view and augmented reality approach in design, International Journal of Advanced Robotic Systems, 2008, 1-18

[3] A. Bauer, D. Wollherr, M. Buss, Human–robot collaboration: a survey, International Journal of Humanoid Robotics, 2008, 47-66

[4] В. С. Стёпин. Б. В. Бирюков. Ф. И. Голдберг, Анализ, Гуманитарная энциклопедия: Концепты, 2002–2019, Центр гуманитарных технологий

[5] Е. А. Сидоренко, Формализация, Гуманитарная энциклопедия: Концепты [Электронный ресурс], 2002–2019, https://gtmarket.ru/concepts/6937

[6] Т. П. Давиденко, Краткий очерк по лингвистике РЖЯ, Современные аспекты жестового языка, 2006, 146-161

[7] Г. Л. Зайцева, Жестовая речь, Дактилология: учебник для студентов высших учебных заведений, 2000, 42

[8] О. О. Королькова, Проблемы классификаций жестов русского жестового языка, Научный диалог, 2016, 46-59

[9] А. Е. Харламенков, Сборник упражнений и текстов для перевода на жестовый язык, Методическое пособие, 2013, 17-18

[10] В.И. Флёри, Глухонемые, рассматриваемые в отношении к их состоянию и к способам образования, самым свойственным их природе, Типография А. Плюшара, 1835, ISBN

[11] J. Letessier, F. Bérard, Visual tracking of bare fingers for interactive surfaces, Proceedings of the 17th annual ACM symposium on User interface software and technology, 2004, 119-122

[12] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, Pattern Analysis and Machine Intelligence, 2002, 509-522

[13] B. Allen, B. Curless, Z. Popović, Articulated body deformation from range scan data, ACM Transactions on Graphics (TOG), 2002, 612-619

[14] H. Bay, T. Tuytelaars, L. Van Gool, Surf: Speeded up robust features, Computer vision–ECCV, 2006, 404-417

[15] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, Computer Vision (ICCV), 2011, 2564-2571

[16] NND, High-precision gesture control system, [Электронный ресурс], 2012, https://nnd.name/2012/10/vysokotochnaya-sistema-zhestovogo-upravleniya/

[17] 3dnews, CES 2016: новая медиасистема Volkswagen с жестовым управлением, [Электронный ресурс], 2016, https://3dnews.ru/926244

[18] DailyTechInfo, DICE - система жестового управления автомобилем от Mersedes-Benz, [Электронный ресурс], 2012, https://dailytechinfo.org/auto/3291-dice-sistema-zhestovogo-upravleniya-avtomobilem-ot-mersedes-benz.html

[19] Christopher Paul Urmson, Dmitri A. Dolgov, Philip Nemec, Driving pattern recognition and safety control, [Электронный ресурс], 2011, https://patents.google.com/patent/US8634980

[20] Sung-Ho Im, Dong-Sun Lim, Tae-Joon Park, Kee-Koo Kwon, Man-Seok Yang, Heung-Nam Kim, User interface apparatus using hand gesture recognition and

method thereof, [Электронный ресурс], 2004, https://patents.google.com/patent/US20060136846A1

[21] Navneet D., Garg, R., Gulshan, V., Mohan, A., System and method for gesture detection through local product map, [Электронный ресурс], 2013, https://patents.google.com/patent/US20140254864A1

[22] Дружков, П., Детектирование объектов на изображениях, Высокопроизводительные вычисления и алгоритмы компьютерного зрения, 2014, 3-4

[23] Чичварин, Н., Распознавание образов, Обнаружение и распознавание сигналов, 2016, 1-2

[24] Бовырин, А., Дружков, П., Ерухимов, В., Задача детектирования объектов на изображениях и методы её решения, Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP, 2015, 31-38

[25] Carbonell, J., Michalski, R., Mitchell, T., Machine Learning, AN OVERVIEW OF MACHINE LEARNING, 1983, 3-23

[26] Pang, Y., Yuan, Y., Li, X., Pan, J., Efficient HOG human detection, Signal Processing, 91(4), 2011, 773–781

[27] Choi, M., Torralba, A., Willsky, A., Exploiting Hierarchical Contex on a large database of object categories, IEEE Computer Vision and Pattern Recognition, 2010, 129-136

[28] Vidal-Naquet, M., Object Recognition with Informative Features and Linear Classification, The Weizmann Institute of Science, 2008, 1-6

[29] Wojek, C., Dorkó, G., Schulz, A., Schiele, B., Sliding-Windows for Rapid Object Class Localization: A Parallel Technique, Pattern Recognition, 2008, 71–81

[30] Elsen, V., Pol, E.-J. D., Viergever, M., Medical image matching-a review with classification, IEEE Engineering in Medicine and Biology Magazine, 12(1), 1993, 26–39

[31] Danielsson, P.-E., Euclidean distance mapping, Computer Graphics and Image Processing, 14(3), 1980, 227–248

[32] Bradley, A. P., The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition, 30(7), 1997, 1145–1159

[33] Заенцев, И., Нейронные сети: основные модели, Учебное пособие к курсу "Нейронные сети", 1999, 3-12

[34] Pao, Y., Adaptive pattern recognition and neural networks, Addison-Wesley, 1989, 1-12

[35] Круг, П., НЕЙРОННЫЕ СЕТИ И НЕЙРОКОМПЬЮТЕРЫ, Учебное пособие по курсу «Микропроцессоры», 2002, 47-50

[36] Pal, S. K., Mitra, S., Multilayer perceptron, fuzzy sets, and classification, IEEE Transactions on Neural Networks, 3(5), 1992, 683-697

[37] Hecht-Nielsen, R., Theory of the Backpropagation Neural Network, Based on “nonindent”, International Joint Conference on Neural Networks 1, 1992, 593–611

[38] Сирота, А., Цуриков, В., Модели и алгоритмы классификации многомерных данных на основе нейронных сетей с радиально-базисными функциями, Воронежский государственный университет, 2013, 1-7

[39] Костылев, И., Малинецкий, Г., Параметры порядка в нейронной сети Хопфилда , Ж. вычисл. матем. и матем. физ., 1994, 1733–1741

[40] Sutskever, I., Hinton, G., The Recurrent Temporal Restricted Boltzmann Machine,

NIPS, 2008, 1-8

[41] Lu, Jian John, Road crack condition performance modeling using recurrent Markov chains and artificial neural networks, Graduate Theses and Dissertations, 2004, 1310-1376

[42] Huang, G., Zhu, Q., Extreme learning machine: a new learning scheme of feedforward neural networks, 2004 IEEE International Joint Conference on Neural Networks, 2004, 985-991

[43] Белов, В., О перспективах искусственного интеллекта, М.: Дело, 2012, 82

[44] Круглов, В., Борисов, В., Искусственные нейронные сети, Теория и практика, 2002, 380

[45] Rosenblatt F., The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Phichological Review, 1958, 16-22

[46] Kröse B., Smagt P., An introduction to Neural Networks, University of Amsterdam, 1996, 178-185

[47] Smith L., An Introduction to Neural Networks, University of Stirling, 2001, 241-357

[48] Kohonen, T., Self-Organizing Maps, Springer-Verlag: third extended edition, 2001, 52-61

[49] Goh, A., Back-propagation neural networks for modeling complex systems, Artificial Intelligence in Engineering, 9(3), 1995, 143–151

[50] Krizhevsky, A., ImageNet Classification with Deep Convolutional Neural Networks, University of Toronto, 2014, 1-8

[51] Goodfellow, I., Bengio, Y., Deep Learning, The MIT Press, 2017, 281-293

[52] Visin, D., A guide to convolution arithmetic for deep learning, Dumoulin, Vincent, and Francesco Visin, 2016, 23

[53] Hawkins, D., The Problem of Overfitting, Journal of Chemical Information and Computer Sciences, 44(1), 2004, 1–12

[54] Krizhevsky, A., Salakhutdinov, R., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, University of Toronto, 2013, 1931-1958

[55] , An end-to-end open source machine learning platform, [Электронный ресурс], 2019, https://www.tensorflow.org/

[56] Yangqing, J., Deep learning framework by BAIR, https://caffe.berkeleyvision.org/, 2019, [Электронный ресурс]

[57] Docs, Theano: 1.0 release, [Электронный ресурс], 2019, http://deeplearning.net/software/theano/

[58] Docs, PyTorch documentation, [Электронный ресурс], 2019, https://pytorch.org/docs/stable/index.html

[59] Dyk, D., The Art of Data Augmentation, Journal of Computational and Graphical Statistics Vol. 10, No. 1, 2001, 4-31

[60] DeVries, T., DATASET AUGMENTATION IN FEATURE SPACE, School of Engineering University of Guelph Guelph, 2017, 12

[61] Bengio, Y., Better mixing via deep representations, In ICML (1), 2013, 552–560

[62] Box, G., Leonard, T., Scientific Inference, Data Analysis, and Robustness, Proceedings of a Conference Conducted by the Mathematics Research Center, the University of Wisconsin–Madison, 1981, 4-6

[63] McCord, M., & Chuah, M., Spam Detection on Twitter Using Traditional Classifiers, Autonomic and Trusted Computing, 2011, 175–186

[64] Doliotis, P., Stefan, A., McMurrough, C., Comparing gesture recognition accuracy using color and depth information, Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11, 2011, 212-227

[65] Nalepa, J., Grzejszczak, T., Kawulok, M., Wrist Localization in Color Images for Hand Gesture Recognition, Man-Machine Interactions 3, 2014, 79-86

[66] Habili, N., Lim, C., Moini, A., Segmentation of the Face and Hands in Sign Language Video Sequences Using Color and Motion Cues, IEEE Transactions on Circuits and Systems for Video Technology, 14(8), 2004, 1086-1097

[67] Shaik, K., Ganesan, P., Kalist, V., Sathish, B., Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space, Procedia Computer Science, 57, 2016, 41-48

[68] Forstner, W., Image Preprocessing for Feature Extraction in Digital Intensity, Color and Range Images, Lecture Notes in Earth Sciences, 2000, 165-189

[69] Wang, J., Perez, L., The Effectiveness of Data Augmentation in Image Classification using Deep Learning, Stanford University, 2017, -

[70] Brownlee, J., What is the Difi erence Between Test and Validation Datasets, Machine Learning Process, 2017, 1-12

[71] LeCun, Y., Jackel, L., Bottou, L., Learning algorithms for classiaшcation: a comparison on handwritten digit recognition, AT&T Bell Laboratories, 1995, -

[72] Potkin, O., Philippovich, A., Static Gestures Classification Using Convolutional Neural Networks on the Example of the Russian Sign Language, Supplementary Proceedings of the Seventh International Conference on Analysis of Images, Social Networks and Texts (AIST 2018), 2018, 229-234

[73] Krizhevsky, A., Sutskever, I., Hinton, G.E., Imagenet classification with deep convolutionalneural networks, NIPS, 2012, 1-9

[74] Long, J., Shelhamer, E., Darrell, T., Fully Convolutional Networks for Semantic Segmentation, The IEEE Conference on Computer Vision and Pattern Recognition, 2015, 3431-3440

[75] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Generative Adversarial Nets, Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014, -

ПРИЛОЖЕНИЕ А. ДОКЛАД ПО ТЕМЕ ВКР (РУССКАЯ ВЕРСИЯ)

Роботизированные системы стали ключевыми компонентами в различных отраслях промышленности. В последнее время концепция Human-Robot Collaboration (HRC) привлекла внимание исследователей. Человек обладает несравненными навыками решения проблем в значительной степени благодаря продвинутым сенсорно-двигательным способностям, но имеет ограничения по силе и точности. Но роботизированные системы имеют стойкость к усталости, высокую скорость, точность и производительность, но серьезные ограничения в гибкости.

Актуальность проведенных исследований обусловлена проблемой взаимодействия между людьми и машинами (роботами), над решением которой работают такие известные компании как Boston Dynamics, Kuka Robotics, Google и другие. Создание эффективных каналов взаимодействия, в том числе на базе жестовых команд, может освободить людей от тяжелых и потенциально опасных задач.

Целью работы является исследование и анализ существующих методов, алгоритмов и подходов к решению задачи человеко-машинного взаимодействия на основе жестовых команд, разработка архитектуры сверточной нейронной сети для классификации статических жестов, реализация алгоритмов предварительной обработки графических данных, создание набора данных для обучения и тестирования классификатора, а также сбор и анализ экспериментальных данных.

В рамках диссертации реализованы следующие модули системы:

— уникальный набор данных (изображения статических жестов) для обучения и тестирования алгоритмов машинного обучения;
— программное обеспечение для предварительной подготовки данных и обучения классификатора на базе сверточной нейронной сети;
— программное обеспечение для классификации статических жестов и визуализации результата.

Целью разрабатываемого программного комплекса является внедрение в мультимедийные системы транспортных средств (Infotainment Systems), а также обеспечение возможности отдельным разработчикам и исследователям использовать разрабатываемые алгоритмы и набор данных в целях доработки и использования в собственных исследованиях.

Основными задачами исследования являются:

— анализ методов классификации изображений, основанных на использовании искусственных нейронных сетей;

— выбор оптимального метода выделения информативной части на изображениях;
— выбор оптимальной архитектуры нейронной сети для распознавания статических жестов в видеопотоке;
— оптимизация параметров используемой нейронной сети;
— выбор методов и параметров аугментации (расширения) набора данных для обучения классификатора;
— реализация и исследование работоспособности и эффективности алгоритма распознавания статических жестов, основанного на использовании искусственной нейронной сети.

Одним из ключевых компонентов программного модуля были использованы алгоритмы машинного обучения, а именно сверточные нейронные сети. Также были разработаны вспомогательные программные компоненты для предварительной обработки данных, обучения нейронной сети и проверки точности классификатора. Проверка точности заключается в вычислении погрешности, точности и полноты.

Научная новизна заключается в способах оптимизации методов обучения нейронной сети, повышении качества и увеличении объема набора данных, увеличения точности классификации.

В качестве способов оптимизации применяются:

— создание набора синтетических данных для обучения;
— подбор параметров классификатора.

Для обучения, кросс-валидации (перекрестной проверки) и тестирования классификатора был разработан оригинальный набор данных. Набор данных включает 2071 изображение рук в определенной жестовой конфигурации. Каждая конфигурация руки является классом и соответствует определенному статическому жесту русского языка жестов. Всего представлено 10 классов.

В работе также описана архитектура нейронной сети, применяемая для задачи классификации жестовых команд. В качестве эталона были взяты две модели: архитектура LeNet-5 и классификатор статических жестов, который был представлен на конференции AIST в 2018 году. Новая усовершенствованная архитектура сети по сравнению с эталонными архитектурами включает в себя более глубокие свертки в сверточных слоях и значительно большее число элементов в полносвязных слоях.

Усовершенствованный классификатор демонстрирует точность классификации в 93,6% по тестовому набору данных, которая превосходит результаты предыдущей версии — 91,38% и классификатор LeNet-5 — 87,08%. Результаты точности являются

достаточной основой для разработки промышленного прототипа интерфейса управления при помощи жестовых команд и дальнейших исследований в этом направлении.

Практическая значимость заключается в разработке системы распознавания жестовых команд, которую можно использовать в качестве интерфейса для человеко-машинного взаимодействия, а также в обеспечении возможности тестирования алгоритмов классификации и применении пользовательских наборов данных и параметров нейронной сети (метод обучения, топология нейронной сети и др.) для решения смежных задач.

В следующем поколении проекта будет применен подход семантической сегментации для задачи классификации с полносвязной сверточной сетью (Fully Convolutional Networks). Предполагается иной способ маркировки набора данных и требуется больше данных. Для решения этой проблемы будут создаваться новые изображения с использованием синтетических данных (случайное преобразование перспективы, случайный шум, генерирующие состязательные сети — Generative Adversarial Nets и другие).

ПРИЛОЖЕНИЕ Б. ДОКЛАД ПО ТЕМЕ ВКР (АНГЛИЙСКАЯ ВЕРСИЯ)

Robotic systems have become key components in various industries. Recently, the Human-Robot Collaboration (HRC) concept has attracted the attention of many researchers. A human possesses incomparable problem-solving skills largely due to advanced sensory-motor abilities but has limitations in strength and accuracy. But robotic systems have resistance to fatigue, have high speed, accuracy, and performance, but limitations in flexibility are significant.

The relevance of the research is due to the problem of interaction between people and machines (robots), which are being solved by such well-known companies like Boston Dynamics, Kuka Robotics, Google and others. Creating effective channels of interaction, including on the basis of gestural commands, can free people from difficult and potentially dangerous tasks.

The aim of the work is to study and analyze existing methods, algorithms and approaches for solving the problem of Human-Robot Collaboration based on gesture commands, developing a convolutional neural network architecture for static gestures classification, implementing pre-processing algorithms for graphics data, creating a dataset for training and testing a classifier and the collection and analysis of experimental data.

In the scope of the thesis the following system modules were implemented:

— a unique dataset (images of static gestures) for training and testing machine learning algorithms;
— software for preliminary data preparation and classifier training based on a convolutional neural network;
— software for the classification of static gestures and visualization of the result.

The main purpose of the software pipeline is the introduction of multimedia vehicle systems (Infotainment Systems), as well as providing opportunities for individual developers and researchers to use the developed algorithms and data set for the purpose of improving and use in their own research.

The main objectives of the study are:

— analysis of image classification methods based on the artificial neural networks;
— the choice of the optimal method of selection of the informative part on the images
— selection of the optimal neural network architecture for recognizing static gestures in a video stream;
— tuning of the parameters of the neural network;
— selection of methods and parameters of augmentation of a dataset for classifier training;

— implementation and study of the efficiency and effectiveness of the algorithm for recognizing static gestures based on the artificial neural networks.

One of the key components of the program module is machine learning algorithms, namely convolutional neural networks. Auxiliary software components for preprocessing data, training the neural network and checking the accuracy of the classifier were also developed. The quality check consists of calculating the accuracy, and loss.

The scientific novelty is in the approaches for optimization of neural network training methods, improving the quality and increasing the volume of the dataset, improving the classification accuracy.

The following methods are used as optimization methods:

— creating a set of synthetic data for training;

— tuning of classifier parameters.

A unique dataset was developed for training, cross-validation, and testing of the classifier. The dataset includes 2071 images of hands in a certain gestural configuration. Each configuration of a hand is a class and corresponds to a certain static gesture of the Russian Sign Language. There are 10 classes in total.

The thesis also describes the neural network architecture used for the task of classification of gesture commands. Two models were taken as a benchmark: the LeNet-5 architecture and the Static Gesture Classifier, which was presented at the AIST conference in 2018. The new improved network architecture, in comparison to the benchmark architectures, includes deeper convolutions in convolutional layers and a larger number of elements in fully connected layers.

The improved classifier demonstrates the classification accuracy of 93.6% for the test set, which exceeds the results of the previous version — 91.38% and the LeNet-5 classifier — 87.08%. The results of accuracy are a sufficient basis for the development of an industrial prototype of the control interface based on the gesture commands and further research in this direction.

The practical significance is in the development of a system of recognition of gesture commands, which can be used as an interface for human-robot collaboration, as well as providing the ability to test classification algorithms and use custom data sets and neural network parameters (training method, neural network topology, etc.) to solve similar problems.

ПРИЛОЖЕНИЕ В. ГРАФИЧЕСКИЙ МАТЕРИАЛ РАБОТЫ (МАКЕТЫ ПЛАКАТОВ)

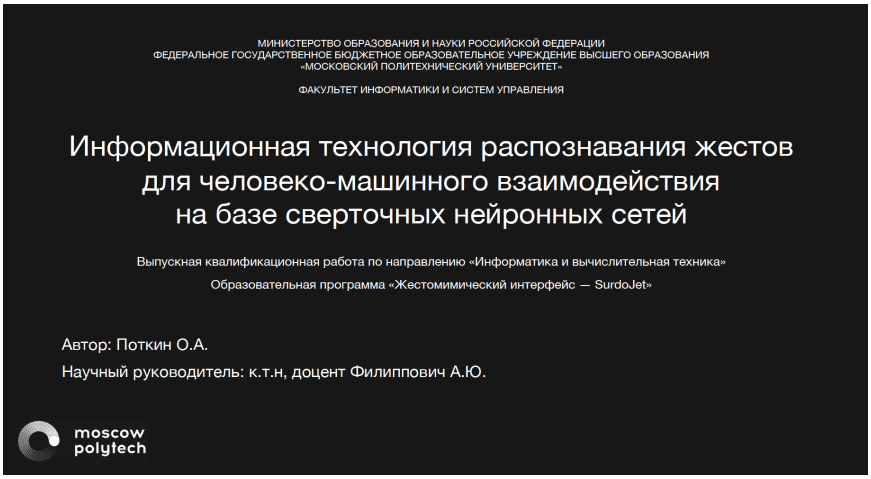

Рисунок 37 − Титульный лист презентации

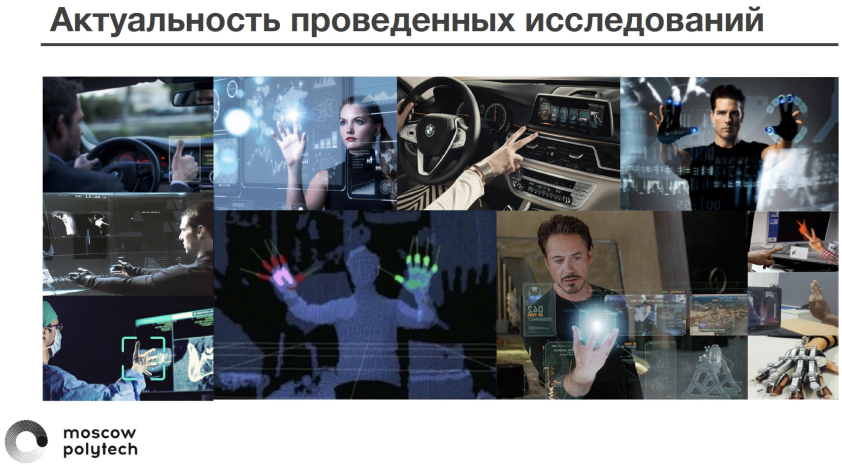

Рисунок 38 − Актуальность темы диссертации

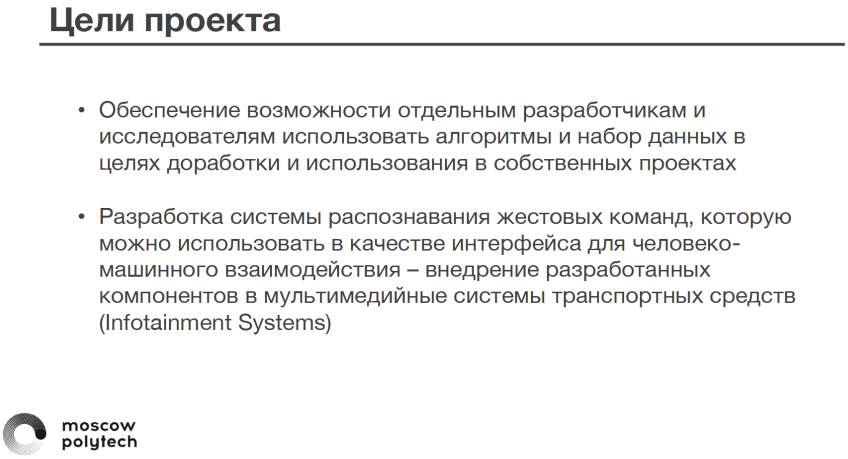

Рисунок 39 − Цели проекта

Рисунок 40 − Обзор аналогов

Рисунок 41 - Схема работы системы

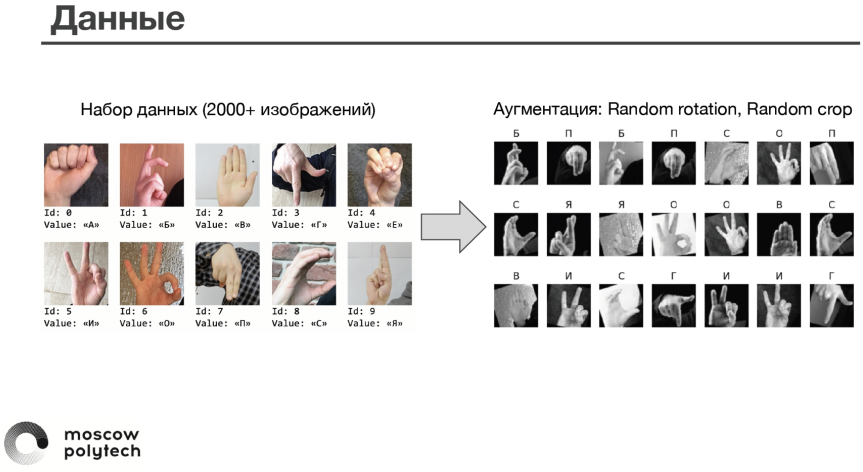

Рисунок 42 − Данные

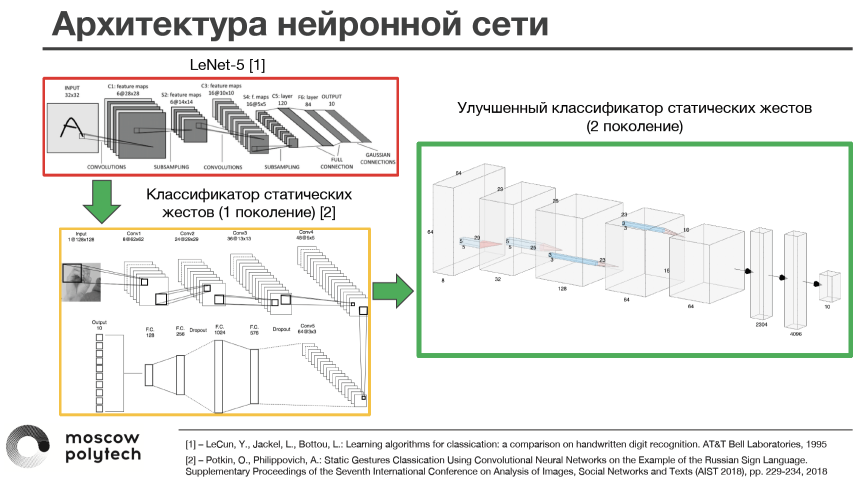

Рисунок 43 – Архитектура нейронной сети

Сравнительный анализ классификаторов

Архитектура классификатора	Потери при обучении (Training Loss)	Точность при обучении (Training accuracy)	Потери на тестовых данных (Test Loss)	Точность на Тестовых данных (Test accuracy)
LeNet-5	0.0681	0.9985	0.5134	0.8708
Классификатор статических жестов (1 поколение)	0.1411	0.9369	0.2385	0.9138
Улучшенный классификатор статических жестов (2 поколение)	0.1310	0.9433	0.2119	0.9360

moscow polytech

Рисунок 44 – Сравнительный анализ классификаторов (часть 1)

Сравнительный анализ классификаторов

Признаки недообучения (Underfitting)

Улучшенный показатель точности

Архитектура классификатора	Потери при обучении (Training Loss)	Точность при обучении (Training accuracy)	Потери на тестовых данных (Test Loss)	Точность на Тестовых данных (Test accuracy)
LeNet-5	0.0681	0.9985	0.5134	0.8708
Классификатор статических жестов (1 поколение)	0.1411	0.9369	0.2385	0.9138
Улучшенный классификатор статических жестов (2 поколение)	0.1310	0.9433	0.2119	0.9360

Точность, приемлемая для разработки прототипа

moscow polytech

Рисунок 45 – Сравнительный анализ классификаторов (часть 2)

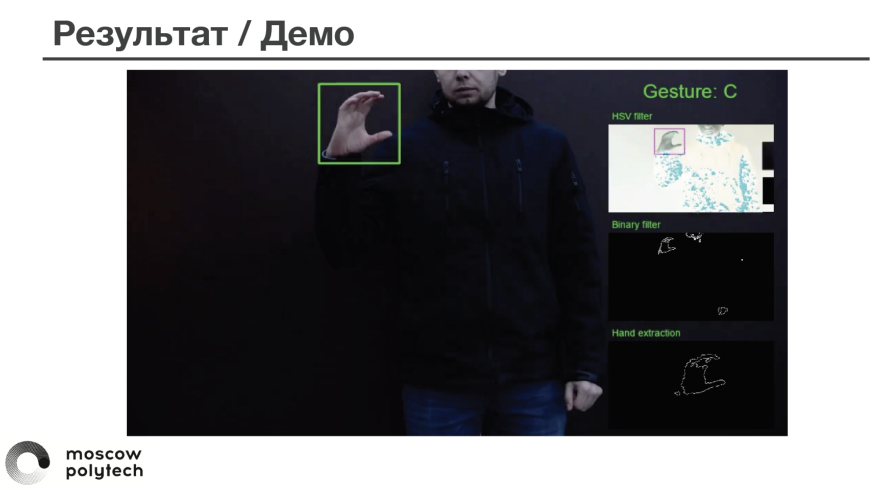

Рисунок 46 – Результат / демо

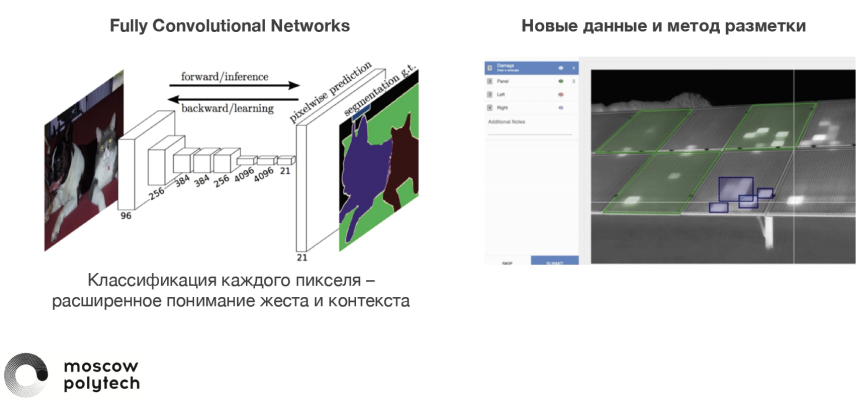

Рисунок 47 – Развитие проекта

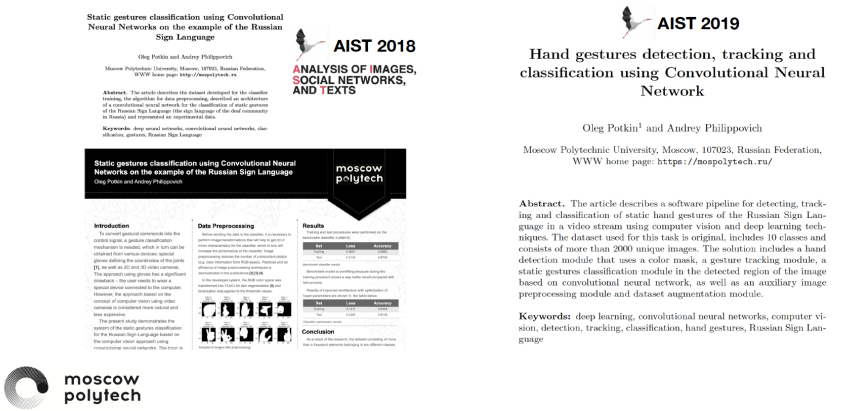

Рисунок 48 – Публикации (часть 1)

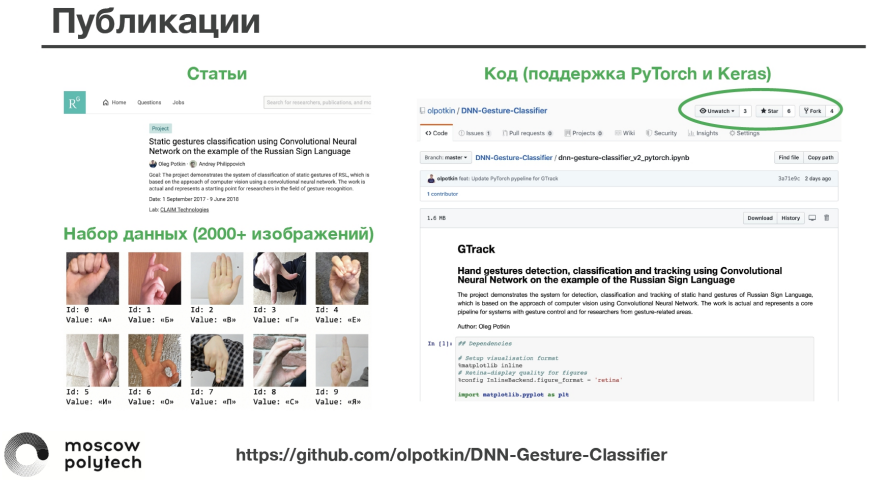

Рисунок 49 – Публикации (часть 2)

Рисунок 50 – Контакты

ПРИЛОЖЕНИЕ Г. ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ

```
## GTrack
##
## @brief Hand gestures detection, classification and tracking using Convolutional Neural Network
## on the example of the Russian Sign Language
##
## The project demonstrates the system for detection, classification and tracking of static hand
## gestures of Russian Sign Language, which is based on the approach of computer vision using
## Convolutional Neural Network. The work is actual and represents a core pipeline for systems with
## gesture control and for researchers from gesture-related areas.
##
## @author Oleg Potkin
## @source https://github.com/olpotkin/DNN-Gesture-Classifier

########################################
## Dependencies
########################################

# Setup visualisation format
%matplotlib inline
# Retina-display quality for figures
%config InlineBackend.figure_format = 'retina'

import matplotlib.pyplot as plt
import os                # operating system functions
import numpy as np  # matrix operations

# Computer vision functions
import cv2
(major_ver, minor_ver, subminor_ver) = (cv2.__version__).split('.')
print('OpenCV Version: {}.{}.{}'.format(major_ver, minor_ver, subminor_ver))

# Deep Learning components
import torch, torchvision
from torchvision import datasets, transforms, models
from torch.utils.data.sampler import SubsetRandomSampler

# Check if CUDA is available
train_on_gpu = torch.cuda.is_available()
if not train_on_gpu:
    print('CUDA is not available.  Training on CPU ...')
else:
    print('CUDA is available!  Training on GPU …')

########################################
## Part 1: Data
## 1.1 Load data
########################################

# Directory with dataset
data_dir = 'gesture_set'

# Number of subprocesses to use for data loading
num_workers = 0
# How many samples per batch to load
batch_size = 10
# Percentage of training set to use as validation
```

```
valid_size = 0.25
# Learning rate for CNN
learning_rate = 0.01

# Define transforms for the training data and testing data
transform = transforms.Compose([
    transforms.RandomRotation(7),
    transforms.Resize(64),
    transforms.Grayscale(num_output_channels=1),
    transforms.ToTensor()])

# Pass transforms in here, then run the next cell to see how the transforms look
train_data = datasets.ImageFolder(data_dir + '/train', transform=transform)
test_data = datasets.ImageFolder(data_dir + '/test', transform=transform)

# Obtain training indices that will be used for validation
num_train = len(train_data)
print("Training set size = {0}".format(train_data))
indices = list(range(num_train))
np.random.shuffle(indices)
split = int(np.floor(valid_size * num_train))
train_idx, valid_idx = indices[split:], indices[:split]

# Define samplers for obtaining training and validation batches
train_sampler = SubsetRandomSampler(train_idx)
valid_sampler = SubsetRandomSampler(valid_idx)

# Prepare data loaders (combine dataset and sampler)
train_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
    sampler=train_sampler, num_workers=num_workers)
valid_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size,
    sampler=valid_sampler, num_workers=num_workers)
test_loader = torch.utils.data.DataLoader(test_data, batch_size=batch_size,
    num_workers=num_workers)

# Specify the image classes
classes = ['А', 'Б', 'В', 'Г', 'Е', 'И', 'О', 'П', 'С', 'Я']

fig = plt.figure(figsize=(8,8));
columns = 7;
rows = 5;
for i in range(1, columns*rows+1):
    img_xy = np.random.randint(len(train_data));
    img = train_data[img_xy][0][0,:,:]
    fig.add_subplot(rows, columns, i)
    plt.title(classes[train_data[img_xy][1]])
    plt.axis('off')
    plt.imshow(img, cmap='gray')
plt.show()

# Obtain one batch of training images
dataiter = iter(train_loader)
images, labels = dataiter.next()
# convert images to numpy for display
images = images.numpy()
# Plot the images in the batch, along with the corresponding labels
fig = plt.figure(figsize=(10, 4))
# display 10 images
```

```
for idx in np.arange(10):
    ax = fig.add_subplot(2, 10/2, idx+1, xticks=[], yticks=[])
    plt.imshow(np.transpose(images[idx],
(1,2,0)).reshape(images[idx].shape[1],images[idx].shape[2]),cmap='gray')
    ax.set_title(classes[labels[idx]])

########################################
## Part 2: CNN Architecture
########################################

# Working use-case
import torch.nn as nn
import torch.nn.functional as F

# define the CNN architecture
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        # convolutional layer (sees 32x32x3 image tensor)
        self.conv1 = nn.Conv2d(1, 8, kernel_size=5, padding=1)
        # convolutional layer (sees 16x16x16 tensor)
        self.conv2 = nn.Conv2d(8, 16, kernel_size=3, padding=1)
        self.conv3 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
        # convolutional layer (sees 8x8x32 tensor)
        self.conv4 = nn.Conv2d(32, 64, kernel_size=2, padding=1)

        # max pooling layer
        self.pool = nn.MaxPool2d(2, 2)
        # linear layer (64 * 4 * 4 -> 500)
        self.fc1 = nn.Linear(64 * 4 * 4, 1200)
        # linear layer (500 -> 200)
        self.fc2 = nn.Linear(1200, 500)
        # linear layer (500 -> 10)
        self.fc3 = nn.Linear(500, 10)
        # dropout layer (p=0.2)
        self.dropout = nn.Dropout(0.2)

        ## The 'forward' function is called on the Neural Network for a set of inputs,
        ## and it passes that input through the different layers that have been defined.
    def forward(self, x):
        # add sequence of convolutional and max pooling layers
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        # flatten image input
        #print(x.shape)
        x = x.view(-1, 64 * 4 * 4)
        # add dropout layer
        x = self.dropout(x)
        # add 1st hidden layer, with relu activation function
        x = F.relu(self.fc1(x))
        # add dropout layer
        x = self.dropout(x)
        # add 2nd hidden layer, with relu activation function
        x = self.fc2(x)
        x = self.dropout(x)
        x = self.fc3(x)
```

```
        return x
# create a complete CNN
model = Net()
print(model)

# move tensors to GPU if CUDA is available
if train_on_gpu:
    model.cuda()

import torch.optim as optim

# specify loss function (categorical cross-entropy)
criterion = nn.CrossEntropyLoss()
# specify optimizer
optimizer = optim.SGD(model.parameters(), lr=learning_rate)

# number of epochs to train the model
n_epochs = 70
valid_loss_min = np.Inf    # track change in validation loss

for epoch in range(1, n_epochs+1):
    # keep track of training and validation loss
    train_loss = 0.0
    valid_loss = 0.0

    # train the model
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # move tensors to GPU if CUDA is available
        if train_on_gpu:
            data, target = data.cuda(), target.cuda()
        # clear the gradients of all optimized variables
        optimizer.zero_grad()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the batch loss
        loss = criterion(output, target)
        # backward pass: compute gradient of the loss with respect to model parameters
        loss.backward()
        # perform a single optimization step (parameter update)
        optimizer.step()
        # update training loss
        train_loss += loss.item()*data.size(0)

    # validate the model
    model.eval()
    for batch_idx, (data, target) in enumerate(valid_loader):
        # move tensors to GPU if CUDA is available
        if train_on_gpu:
            data, target = data.cuda(), target.cuda()
        # forward pass: compute predicted outputs by passing inputs to the model
        output = model(data)
        # calculate the batch loss
        loss = criterion(output, target)
        # update average validation loss
        valid_loss += loss.item()*data.size(0)
    # calculate average losses
    train_loss = train_loss/len(train_loader.dataset)
```

```
    valid_loss = valid_loss/len(valid_loader.dataset)
    # print training/validation statistics
    print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(
        epoch, train_loss, valid_loss))

    # save model if validation loss has decreased
    if valid_loss <= valid_loss_min:
        print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.format(
        valid_loss_min,
        valid_loss))
        torch.save(model.state_dict(), 'model_augmented.pt')
        valid_loss_min = valid_loss

## Load the Model with the Lowest Validation Loss
model.load_state_dict(torch.load('model_augmented.pt'))

########################################
## Part 3: Evaluation
########################################

from torch.autograd import Variable

model.eval()
correct = 0
total = 0
for data, target in test_loader:
    images = Variable(data.float())
    outputs = model(data)
    _, predicted = torch.max(outputs.data, 1)
    total += target.size(0)
    correct += (predicted == target).sum()
print('Test Accuracy of the model on the XXX test images: %.4f %%' % (100 * correct / total))
print (correct)

# track test loss
test_loss = 0.0
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))

model.eval()
# iterate over test data
test_var = 0;
for data, target in test_loader:
    # move tensors to GPU if CUDA is available
    if train_on_gpu:
        data, target = data.cuda(), target.cuda()
    # forward pass: compute predicted outputs by passing inputs to the model
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    # update test loss
    test_loss += loss.item()*data.size(0)
    # convert output probabilities to predicted class
    _, pred = torch.max(output, 1)
    # compare predictions to true label
    correct_tensor = pred.eq(target.data.view_as(pred))
    correct = np.squeeze(correct_tensor.numpy()) if not train_on_gpu else
np.squeeze(correct_tensor.cpu().numpy())
```

```
    # calculate test accuracy for each object class
    for i in range(batch_size):
        label = target.data[i]
        class_correct[label] += correct[i].item()
        class_total[label] += 1

# average test loss
test_loss = test_loss/len(test_loader.dataset)
print('Test Loss: {:.6f}\n'.format(test_loss))

for i in range(10):
    if class_total[i] > 0:
        print('Test Accuracy of %5s: %2d%% (%2d/%2d)' % (
            classes[i], 100 * class_correct[i] / class_total[i],
            np.sum(class_correct[i]), np.sum(class_total[i])))
    else:
        print('Test Accuracy of %5s: N/A (no training examples)' % (classes[i]))

print('\nTest Accuracy (Overall): %2d%% (%2d/%2d)' % (
    100. * np.sum(class_correct) / np.sum(class_total),
    np.sum(class_correct), np.sum(class_total)))
dataiter = iter(test_loader)

for i in range (16):
    # obtain one batch of test images
    images, labels = dataiter.next()
    images.numpy()

    # move model inputs to cuda, if GPU available
    if train_on_gpu:
        images = images.cuda()

    # get sample outputs
    output = model(images)
    # convert output probabilities to predicted class
    _, preds_tensor = torch.max(output, 1)
    preds = np.squeeze(preds_tensor.numpy()) if not train_on_gpu else
np.squeeze(preds_tensor.cpu().numpy())

    # plot the images in the batch, along with predicted and true labels
    fig = plt.figure(figsize=(16, 1))
    for idx in np.arange(10):
        ax = fig.add_subplot(1, 16, idx+1, xticks=[], yticks=[])
        plt.imshow(np.transpose(images[idx],
(1,2,0)).reshape(images[idx].shape[1],images[idx].shape[2]), cmap='gray')
        ax.set_title("{} ({})".format(classes[preds[idx]], classes[labels[idx]]),
                     color=("green" if preds[idx]==labels[idx].item() else "red"))

########################################
## Part 4: Predictor
########################################

from PIL import Image

def check_class(image):
    model.eval()

    image = Image.fromarray(image)
```

```
    image_pil = image

    transform = transforms.Compose([
        transforms.Resize(64),
        transforms.Grayscale(num_output_channels=1),
        transforms.ToTensor()])

    img = transform(image)
    img.unsqueeze_(0)

    # Predict classes using images from the test set
    output = model(img)
    _, prediction = torch.max(output, 1)
    preds = np.squeeze(prediction.cpu().numpy())

    return classes[preds]

########################################
## Part 5: Bounding box builder
########################################

def check_bounding_box(x, y, w, h):
    x = x - 10
    if (y < 10):
        y = 1
    else:
        y = y - 10
    if w > h:
        h = w
    else:
        w = h
        x = x - 10
    return x, y, w, h

########################################
## Part 6: Input Image Filtering
########################################

def hand_filter(input_image):
    # Make a copy of input image
    image_proc   = input_image   # For processing / filtering
    image_result = input_image   # For result

    # Blur the image
    image_blur = cv2.blur(image_proc, (3,3))

    # Convert the image from RGB to HSV color space
    image_hsv = cv2.cvtColor(image_blur, cv2.COLOR_BGR2HSV)

    # Define the upper and lower boundaries of the HSV pixel
    # intensities to be considered 'skin'
    skin_lower_bound  = np.array([120, 24, 80],   dtype = "uint8")
    skin_upper_bound  = np.array([200, 255, 255], dtype = "uint8")
    skin_mask_level_1 = cv2.inRange(image_hsv, skin_lower_bound, skin_upper_bound)

    # Apply a series of erosions and dilations to the mask
    # using an elliptical kernel
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
```

```
    skin_mask_level_2 = cv2.erode(skin_mask_level_1, kernel, iterations = 2)
    skin_mask_level_3 = cv2.dilate(skin_mask_level_2, kernel, iterations = 2)

    # Filter part of an image
    for i in range(400, 900):
        skin_mask_level_3[:,i] = 0
        # Detect and draw a hand rectangle (bounding box)
    active_px = np.argwhere(skin_mask_level_3 != 0)
    active_px = active_px[:, [1,0]]
    x, y, w, h = cv2.boundingRect(active_px)     # Detect bounding box
    x, y, w, h = check_bounding_box(x, y, w, h) # Resize bounding box

    # Save detected area
    roi = image_result[y:y+h+20, x:x+w+20]

    ## This line could be use to collect a dataset from the video stream
    # cv2.imwrite("ro{0}i{1}.jpg".format(h, w), roi)

    # Draw Bounding Box
    cv2.rectangle(image_result, (x, y), (x+w+20,y+h+20), (0,255,0), 2)

    return image_result, roi, skin_mask_level_1, skin_mask_level_3, image_hsv

########################################
## Part 7: Image pipeline
########################################

from PIL import Image, ImageDraw, ImageFont

def image_pipeline(image):
    # Step 1: preprocessing
    image, roi, skin_mask_init, skin_mask_final, image_hsv = hand_filter(image)
    # Step 2: classification
    class_name = check_class(roi)

    ### Add thumbnail 1 (HSV filter)
    thumb_width = 240
    thumb_height = 130
    x_offset1 = image.shape[1] - thumb_width - 20
    y_offset1 = 80
    hsv_crop = image_hsv[0:405, 100:100+645]
    hsv_bgr  = cv2.cvtColor(hsv_crop, cv2.COLOR_HSV2BGR)
    hsv_bgr_thumb = 255*cv2.resize(hsv_bgr, (thumb_width, thumb_height),
interpolation=cv2.INTER_AREA)
    image.setflags(write=1)     # Needed to modify image structure
    image[y_offset1:y_offset1 + hsv_bgr_thumb.shape[0], x_offset1:x_offset1 +
hsv_bgr_thumb.shape[1]] = hsv_bgr_thumb
    ### Add thumbnail 2 (Binary filter)
    x_offset2 = image.shape[1] - thumb_width - 20
    y_offset2 = 80 + thumb_height + 30
    resized_bin = 255*cv2.resize(skin_mask_init, (thumb_width, thumb_height),
interpolation=cv2.INTER_AREA)
    bin_thumb   = cv2.merge([resized_bin, resized_bin, resized_bin])
    image[y_offset2:y_offset2 + bin_thumb.shape[0], x_offset2:x_offset2 + bin_thumb.shape[1]] =
bin_thumb
    ### Add thumbnail 3 (Hand extraction)
    x_offset3 = image.shape[1] - thumb_width - 20
    y_offset3 = 80 + (thumb_height + 30)*2
```

```
    hand_crop = skin_mask_final[0:200, 160:160+320]
    resized_hand = 255*cv2.resize(hand_crop, (thumb_width, thumb_height),
interpolation=cv2.INTER_AREA)
    hand_thumb = cv2.merge([resized_hand, resized_hand, resized_hand])
    image[y_offset3:y_offset3 + hand_thumb.shape[0], x_offset3:x_offset3 + hand_thumb.shape[1]] =
hand_thumb
    ###
    im_pil = Image.fromarray(image)
    draw = ImageDraw.Draw(im_pil)
    font = ImageFont.truetype("arial.ttf", 28, encoding="unic")
    draw.text((x_offset1 + 50, 15), "Gesture: {}".format(class_name), (0,255,0), font=font)
    font = ImageFont.truetype("arial.ttf", 14, encoding="unic")
    draw.text((x_offset1 + 5, y_offset1-20), "HSV filter"     , (0,255,0), font=font)
    draw.text((x_offset2 + 5, y_offset2-20), "Binary filter"  , (0,255,0), font=font)
    draw.text((x_offset3 + 5, y_offset3-20), "Hand extraction", (0,255,0), font=font)

    image = np.array(im_pil)
    return image

########################################
## Part 8: Pipeline with a video
########################################

from moviepy.editor import VideoFileClip
from IPython.display import HTML

output_video = 'project_video_processed-02.mp4'
clip1 = VideoFileClip("test1.mp4")
video_clip = clip1.fl_image(image_pipeline)
%time video_clip.write_videofile(output_video, audio=False)
```

ПРИЛОЖЕНИЕ Д. РЕФЕРАТИВНЫЙ ОБЗОР ОСНОВНОЙ ЛИТЕРАТУРЫ

При написании диссертации были использованы материалы из 10 книг, 54 научных статьи и 11 электронных источников (интернет-ресурсы).

Среди книг наиболее значимыми являются:

— «Deep Learning» (автор I., Goodfellow) [51]. Книга содержит математические и концептуальные основы линейной алгебры, теории вероятностей и теории информации, численных расчетов и машинного обучения в том объеме, который необходим для понимания материала. Описываются приемы глубокого обучения, применяемые на практике, в том числе глубокие сети прямого распространения, регуляризация, алгоритмы оптимизации, сверточные сети, моделирование последовательностей и др. Рассматриваются такие приложения, как обработка естественных языков, распознавание речи, компьютерное зрение, онлайновые рекомендательные системы, биоинформатика и видеоигры.

— «Сборник упражнений и текстов для перевода на жестовый язык, методическое пособие» (автор А. Е. Харламенков) [9]. В книге содержится множество теоретических и практических материалов, необходимых для решения задачи формализации жестовых данных.

— «Детектирование объектов на изображениях, Высокопроизводительные вычисления и алгоритмы компьютерного зрения» (автор П., Дружков) [24]. Книга содержит множество примеров и широкоиспользуемых практик для решения задачи детектирования объектов на изображении.

— «Жестовая речь, Дактилология: учебник для студентов высших учебных заведений» (автор Г. Л. Зайцева) [7] содержит обзор дактильного алфавита, лексика и грамматика русского жестового языка; также приводится краткий словарь жестов.

Проблемы, концепты и подходы к решению задачи человеко-машинного взаимодействия (HRC) описаны в работах «Human–robot collaboration: a survey, International Journal of Humanoid Robotics» (авторы A. Bauer, D. Wollherr) [3] и «Human robot collaboration: A literature view and augmented reality approach in design» (авторы S. Green, M. Billinghurst) [2], «О перспективах искусственного интеллекта» (автор Белов, В.) [43].

Описание процессов и требований, а также результаты взаимодействия людей и роботов на производственных линиях описаны в статье «Cooperation of human and machines in assembly lines» (авторы J. Krüger, T. Lien) [1].

Классические методы машинного обучения для решения задач классификации (в том числе классификации объектов на изображении) описаны в работах «An overview of machine learning» (авторы Carbonell, J., Michalski, R.) [25], «Efficient HOG human detection» (авторы Pang, Y., Yuan, Y.) [26], «Object Recognition with Informative Features and Linear Classification» (автор Vidal-Naquet, M.) [28], «Medical image matching-a review with classification» (автор Elsen, V.) [30], «Spam Detection on Twitter Using Traditional Classifiers» (автор McCord, M.) [63].

Отправной точкой при освоении технологий, методов и приемов использования искусственных нейронных сетей для решения прикладных задач, а также теоретических основ стали работы «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain» (автор Rosenblatt F.) [45], «An introduction to Neural Networks» (авторы Krose B., Smagt P.) [46], «An Introduction to Neural Networks» (автор Smith L.) [47].

Наиболее значимыми для исследования и анализа работы нейронных сетей, построения моделей и классификаторов на базе нейронных сетей были статьи «Shape matching and object recognition using shape contexts, Pattern Analysis and Machine Intelligence» (автор S. Belongie) [12], «Multilayer perceptron, fuzzy sets, and classification» (автор Pal, S.) [36], «Искусственные нейронные сети, Теория и практика» (авторы Круглов, В., Борисов, В.) [44], «Back-propagation neural networks for modeling complex systems» (автор Goh, A.) [49], «ImageNet Classification with Deep Convolutional Neural Networks» (автор Krizhevsky, A.) [50], «Learning algorithms for classiaшcation: a comparison on handwritten digit recognition» (автор LeCun, Y.) [71].

Для освоения техники проектирования, разработки, оптимизации, тестирования и практического применения сверточных нейронных сетей при работе с изображениями использовались работы «A guide to convolution arithmetic for deep learning» (автор Visin, D.) [52], «Imagenet classification with deep convolutionalneural networks» (авторы Krizhevsky, A., Sutskever, I.) [73].

Наглядный пример интерфейса системы трекинга пальцев рук приведен в публикации «Visual tracking of bare fingers for interactive surfaces» (автор J. Letessier) [11]. А в работах «Comparing gesture recognition accuracy using color and depth information» (авторы Doliotis, P., Stefan, A.) [64] и «Wrist Localization in Color Images for Hand Gesture Recognition» (авторы Nalepa, J., Grzejszczak, T.) [65] приведены примеры трекинга ручных жестов с использованием цветовых фильтров.

В работе «System and method for gesture detection through local product map (авторы Navneet D., Garg, R.) [21], подробно описана техника обнаружения жестов на

изображениях при помощи элементов локальной карты продукта (LPM).

Научные работы «The Art of Data Augmentation» (автор Dyk, D.,) [59] и «Dataset augmentation in feature space» (автор DeVries, T.) [60], «What is the Difference Between Test and Validation Datasets» (автор Brownlee, J.) [70] описывают нюансы и приводят примеры подготовки и искусственного расширения набора данных для обучения нейронных сетей. А в работе «Surf: Speeded up robust features» (автор H. Bay) [14] описан подход с оптимальным выбором качественных признаков, увеличивающий производительность системы.

Статьи «The use of the area under the ROC curve in the evaluation of machine learning algorithms» (автор Bradley, A. P.) [32], «Dropout: A Simple Way to Prevent Neural Networks from Overfitting» (авторы Krizhevsky, A., Salakhutdinov, R.) [54] содержат информацию об оптимизации методов обучения и оценке качества результатов обучения классификаторов на базе нейронных сетей.

Среди интернет-ресурсов были использованы материалы из базы Google Patents для поиска и анализа существующих технологий для решения задачи управления на основе жестовых команд. Также был использован ресурс Research Gate для поиска открытых научных статей.

ПРИЛОЖЕНИЕ Е. СТАТЬЯ ДЛЯ КОНФЕРЕНЦИИ AIST 2018

Static gestures classification using Convolutional Neural Networks on the example of the Russian Sign Language

Oleg Potkin[1] and Andrey Philippovich

Moscow Polytechnic University, Moscow, 107023, Russian Federation,
WWW home page: http://mospolytech.ru

Abstract. The article describes the dataset developed for the classifier training, the algorithm for data preprocessing, described an architecture of a convolutional neural network for the classification of static gestures of the Russian Sign Language (the sign language of the deaf community in Russia) and represented an experimental data.

Keywords: deep neural networks, convolutional neural networks, classification, gestures, Russian Sign Language

1 Introduction

Human-computer interaction interfaces are diverse in their scope and implementation: systems with console input-output, controllers with gesture control, brain-computer interfaces [1] etc. Systems with the data input based on the custom gestures recognition have gained wide popularity since 2010 after the release of the contactless game controller "Kinect" from Microsoft. These devices increase their market share to this day and become a part of everyday life of different user categories. So, for example, the Volkswagen car manufacturer introduced the multimedia system Golf R Touch Gesture Control for the car multimedia system management using gestures [2], researchers from Cybernet Systems Corporation have developed personal computer software that allows to use hands gestures instead of the usual input devices [3] and Russian scientists have demonstrated a technology of protection against spam bots based on the CAPTCHA mechanism [4].

To convert gestural commands into the control signal, a gesture classification mechanism is needed, which in turn can be obtained from various devices: special gloves defining the coordinates of the joints [5], as well as 2D and 3D video cameras. The approach using gloves has a significant drawback – the user needs to wear a special device connected to the computer. However, the approach based on the concept of computer vision using video cameras is considered more natural and less expensive.

The present study demonstrates the system of the static gestures classification for the Russian Sign Language based on the computer vision approach using convolutional neural networks. The topic is relevant and represents a starting point for researchers in the field of gesture recognition.

2

2 Dataset description for classifier training

For training, cross-validation and testing of the classifier, a set of data was developed[1], containing 1042 images of hands in a certain gesture configuration – the class. In total, 10 classes are represented in the data set, each of which corresponds to a certain static gesture of the Russian Sign Language.

Fig. 1 shows sample images from each of the data set classes.

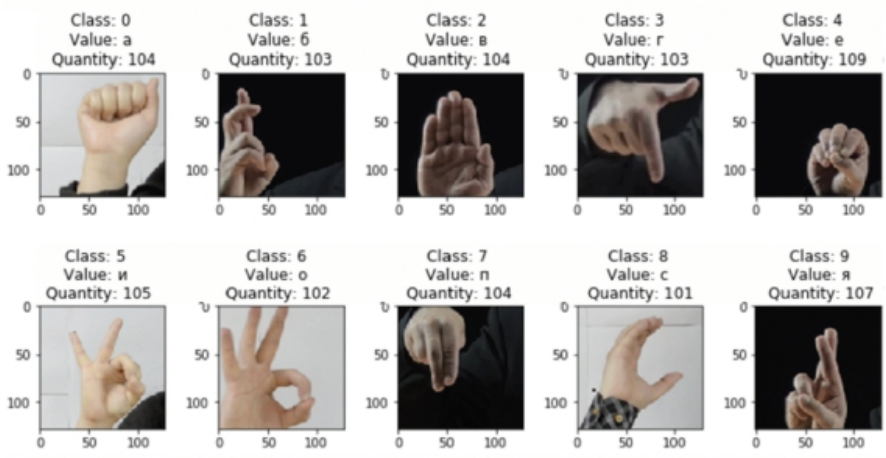

Fig. 1. Samples of images from the dataset. Parameters: the number of the class, the correspondence of the gesture to the letter of the Russian alphabet, the number of images in the class

Images were made with different light conditions and on a small range of object distances from the camera (0.4 – 0.7 meters).

Table 1 represents the image parameters from the dataset.

Table 1. Parameters of images from dataset

Parameter	Value
Format	.PNG
Width (px)	128
Height (px)	128
Color space	RGB
Color Depth	3

[1] Dataset and the source code of the project are available on GitHub: https://github.com/olpotkin/DNN-Gesture-Classifier

Before sending the data to the classifier, it is necessary to perform image transformations that will help to get rid of minor characteristics for the classifier, which in turn will increase the performance of the classifier. This process is called preprocessing. Image preprocessing reduces the number of unimportant details (e.g. color information from RGB space). Practices and an efficiency of image preprocessing techniques is demonstrated in the publications [6] [7] [8].

In the developed system, the RGB color space was transformed into YCbCr for skin segmentation [9] and binarization was applied to the threshold values (Fig. 2).

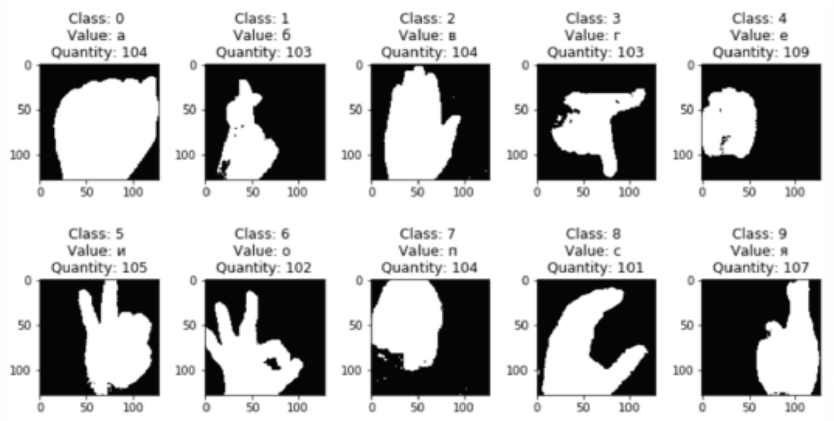

Fig. 2. Samples of images after preprocessing. Parameters: the number of the class, the correspondence of the gesture to the letter of the Russian alphabet, the number of images in the class

The final stage of processing is normalization. The pixel values are in the range from 1 to 255, but for correct operation of the neural network, it is necessary to input values from 0 to 1. To do this, performed a conversion using the formula 1:

$$N = \frac{P}{P_{max}} \tag{1}$$

where N – the pixel value after normalization, P – the value of the normalized pixel, P_{max} – the maximum value of the range.

In order to minimize the overfitting possibility of the classifier [10], the dataset is divided into 3 parts:

- Training set contains 80% of the dataset (666 images).
- Test set, contains 20% of the data set (209 images). Using for evaluation of the model. These samples never used during training and cross-validation processes.
- Cross-validation set, contains 20% of the training sample (167 images).

3 Neural network architecture and the classification results

In this section described CNN architecture on the abstract level. As a benchmark model for an experiment LeNet-5 CNN architecture was taken. The main disadvantage of LeNet-5 is overfitting in some cases and no built-in mechanism to avoid this [11]. So, benchmark architecture was improved by adding dropout layers [13]. Improved LeNet-5 architecture is shown on Fig. 3.

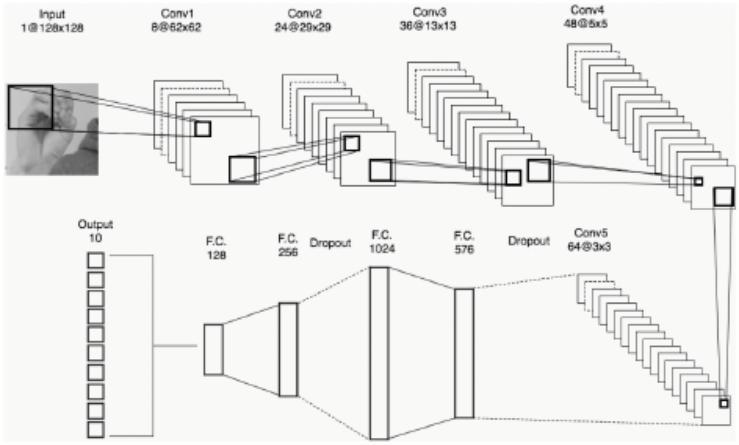

Fig. 3. The architecture of a convolutional neural network for the gestures classification of the Russian Sign Language

The neural network consists of the following layers:

- Input layer (Input). The size of the layer corresponds to the image size after the preprocessing: $1 \times 128 \times 128$.
- Convolution layer (Conv1). Output size: $8 \times 62 \times 62$. Convolution window size: 5×5. Activation function – ReLU (Rectified Linear Unit). ReLU can be described by the equation $f(x) = max(0, x)$ and implements a simple threshold transition at zero. Compared to the sigmoid activation function, ReLU increases learning speed and classifier performance [12].
- Convolution layer (Conv2). Output size: $24 \times 29 \times 29$. Convolution window size: 5×5. Activation function – ReLU.
- Convolution layer (Conv3). Output size: $36 \times 13 \times 13$. Convolution window size: 5×5. Activation function – ReLU.
- Convolution layer (Conv4). Output size: $48 \times 5 \times 5$. Convolution window size: 5×5. Activation function – ReLU.
- Convolution layer (Conv5). Output size: $64 \times 3 \times 3$. Convolution window size: 3×3. Activation function – ReLU.
- Dropout layer 1, $p = 0.25$. The layer is constructed in such a way that each neuron can fall out of this layer with probability p, therefore, other neurons

remain in the layer with probability $q = 1–p$. The dropped neurons are not included in the classifier training, that is, at each new epoch, the neural network is partially changed. This approach effectively solves the overfitting problem of the classifier [13].
– Fully connected layer 1. Size: 576.
– Fully connected layer 2. Size: 1024.
– Dropout layer 2, $p = 0.25$.
– Fully connected layer 3. Size: 256.
– Fully connected layer 4. Size: 128.
– Output layer. Size: 10. Activation function – softmax.

The neural network is developed using the Keras framework based on TensorFlow with the following hyper-parameter values after the optimization procedures: the learning rate is 0.0001, the batch size is 128, the number of epochs is 24, the metrics – accuracy, the optimizer type – Adam.

Training and test procedures were performed on the benchmark classifier (LeNet-5) with results shown in Table 2.

Table 2. Benchmark classifier results

Set	Loss	Accuracy
Training	0.0681	0.9985
Test	0.5134	0.8708

Benchmark model is overfitting because during the training process it shows a way better result compared with test process.

Results of improved architecture with optimization of hyper-parameters shown in Table 3.

Table 3. Classifier optimization results

Set	Loss	Accuracy
Training	0.1411	0.9369
Test	0.2385	0.9138

4 Conclusion

As a result of the research, the dataset consisting of more than a thousand elements belonging to ten different classes of the Russian Sign Language was

developed and published. Algorithms and procedures for preliminary data processing in Python 3.5 were developed. The classifier based on the convolutional neural network using the Keras and TensorFlow frameworks was designed and developed.

The classifier demonstrates an accuracy of classification in 91.38% on the test dataset that is better than result of benchmark classifier – 87.08%. The represented results of an accuracy are sufficient to be a strong basis for further research in this direction.

References

1. Potkin, O., Ivanov, V.: Neurointerface Neurosky: interaction with the hardware computing platform Arduino. 64th Open Student Scientific and Technical Conference, Moscow: Moscow State University of Mechanical Engineering, pp. 151-153 (2014)
2. Volkswagen: Gesture Control: How to make a lot happen with a small gesture. [http://www.volkswagen.co.uk/technology/comfort-and-convenience/gesture-control]
3. Charles J. Cohen, Glenn Beach, Gene Foulk: A Basic Hand Gesture Control System for PC Applications. Cybemet Systems Corporation (2001)
4. Shumilov, A., Philippovich, A.: Gesture-based animated CAPTCHA. Information and Computer Security, Vol. 24 Iss 3 pp. 242 - 254 (2015)
5. Dipietro, L., Angelo M. Sabatini, Dario, P.: urvey of Glove-Based Systems and their applications. IEEE Transactions on systems, Man and Cybernetics, Vol. 38, No. 4, pp 461-482 (2008)
6. Vicen, R., Garcia-Gonzalez, A.: Traffic Sign Classification by Image Preprocessing and Neural Networks. Conference: Proceedings of the 9th international work conference on Artificial neural networks (1970)
7. A.A. Vedenov, M.V. Zurin, E.B. Levchenko: Optical image preprocessing for neural network classifier system. In book: Parallel Problem Solving from Nature (2006)
8. Kussul, E., Baidyk, T., Kussul, M.: Neural network system for face recognition. Conference: Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium onVolume: 5 (2004)
9. Son Lam Phung, Bouzerdoum, A., Chai, D.: A novel skin color model in YCbCr color space and its application to human face detection. Visual Information Processing Research Group Edith Cowan University, Westem Australia, IEEE KIP, pp 289-292 (2002)
10. Brownlee, J.: What is the Difference Between Test and Validation Datasets? Machine Learning Process, [https://machinelearningmastery.com/difference-test-validation-datasets] (2017)
11. Ahmed El-Sawy, Mohamed Loey: CNN for Handwritten Arabic Digits Recognition Based on LeNet-5. In book: Proceedings of the International Conference on Advanced Intelligent Systems and Informatics (2016)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NIPS pp. 1–9 (2012)
13. Geoffrey E. Hinton, Nitish Srivastava, Krizhevsky A., Sutskever, I., Ruslan R. Salakhutdinov: Improving neural networks by preventing co-adaptation of feature detectors. Department of Computer Science, University of Toronto, pp. 1-18 (2012)

ПРИЛОЖЕНИЕ Ж. СТАТЬЯ ДЛЯ КОНФЕРЕНЦИИ AIST 2019

Hand gestures detection, tracking and classification using Convolutional Neural Network

Oleg Potkin[1] and Andrey Philippovich

Moscow Polytechnic University, Moscow, 107023, Russian Federation,
WWW home page: https://mospolytech.ru/

Abstract. The article describes a software pipeline for detecting, tracking and classification of static hand gestures of the Russian Sign Language in a video stream using computer vision and deep learning techniques. The dataset used for this task is original, includes 10 classes and consists of more than 2000 unique images. The solution includes a hand detection module that uses a color mask, a gesture tracking module, a static gestures classification module in the detected region of the image based on convolutional neural network, as well as an auxiliary image preprocessing module and dataset augmentation module.

Keywords: deep learning, convolutional neural networks, computer vision, detection, tracking, classification, hand gestures, Russian Sign Language

1 Introduction

Robotic systems have become key components in various industries. Recently, the concept of Human-Robot Collaboration has attracted the attention of a wide range of researchers. Many examples suggest that a human possesses incomparable problem-solving skills, largely due to advanced sensory-motor abilities, but has limited strength and accuracy [1]. However, robotic systems have resistance to fatigue, a higher speed, accuracy, and performance, but significant limitations in flexibility. Well defined and implemented Human-Robot Collaboration concept can free a human from heavy tasks through an intuitive and reliable interface.

Gestures are one of the ways to exchange information, communicate. The information underlying facial expressions and gestures are at the basis of an efficient communication channel of human interaction [2].

Hand gesture recognition refers to the mathematical interpretation of human palm movements by a computing device. In order to interact with humans, robotic systems must correctly understand human gestures and execute appropriate commands with a sufficient degree of accuracy.

Interfaces based on the custom gestures recognition are various on its application, implementation techniques, and readiness for the market. So, for example,

system DICE (Dynamic and Intuitive Control Experience) from Mercedes-Benz provides a gesture-based user interface for the multimedia system of a car [3]. Google LLC patented the system for driving pattern recognition and safety control that offers to track the driver's hands movements and transform certain gestures into commands to control onboard electronics [4]. Russian researches also demonstrated a concept of protection against spam bots based on the gestural CAPTCHA mechanism [5].

Moreover, technological giants such as Apple, Kuka Robotics, BMW, Facebook, Netflix, and others are actively developing the direction of promising human-machine interaction interfaces, where gesture interaction is one of the most popular direction. And the task of precise and confident recognition of gesture commands is in the high priority.

The present study demonstrates a system for detecting, tracking, and classification of static hand gestures of the Russian Sign Language in a video stream using computer vision and deep learning techniques. The topic is relevant and represents a basis for a scalable and robust control system based on the gesture commands, which can be used as an interface for human-machine interaction.

2 Hand detection using color mask

Human hands and body have unique visual features. In the task of recognizing gestures based on images, gestures consist of fragments of hands and/or body images. Therefore, the usage of such visual signs in the identification of gestures is quite reasonable.

Color is a simple visual function for identifying gestures from background information. However, color-based gesture recognition systems are strongly influenced by lighting and shadows [6]. Another common problem in body parts detecting based on skin color is that the skin color is very different among human races. (Fig 1).

Fig. 1. Skin color palette

Another way to identify gestures is to use the unique shape and contour of the human body. A significant contribution to the formalization of this method was made by Serge Belongie and his colleagues [7].

However, the use of color masks for such tasks is justified in view of the relative simplicity of the method itself, especially at the prototyping stage [8] [9] [10].

Finding the ROI (region of interest) - hand region in this case, divided into the following steps (Fig. 2):

- initial blur the image (Gaussian blur was applied with kernel size = 3) in order to avoid noises;

- convert the image from RGB to HSV color space [11];
- define the upper and lower boundaries of the HSV pixel intensities to be considered as a skin;
- in order to remove binary noises, apply a series of erosions and dilations to the color mask using an elliptical kernel (OpenCV library features);
- detect and draw a hand's ROI. In order to simplify the solution, ROI, in this case, is the area with the highest number of neighbor white pixels;
- ROI is ready for the classification task.

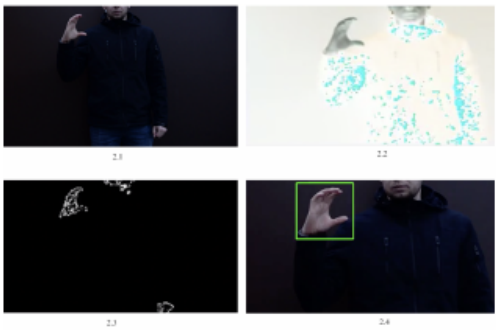

Fig. 2. ROI detection using skin color mask: 2.1 - original, 2.2 - HSV filter, 2.3 - color mask, 2.4 - ROI

3 Dataset and data preprocessing

In order to train, cross-validate and test of the classifier, original dataset was developed[1]. Dataset includes 2071 images of hands in a certain gesture configuration. Each hand configuration is a class itself and corresponds to a certain static gesture of the Russian Sign Language. In total, 10 classes are represented (Fig 3). Images were made with different people, light conditions and backgrounds.

Before sending the data to the classifier, it is necessary to perform image preprocessing - transformations that will help to get rid of minor characteristics for the classifier, which in turn will increase the performance of the classifier. Image preprocessing reduces the number of unimportant details (e.g. color information from RGB space) [12].

Table 1 represents the image parameters from the dataset before and after preprocessing.

In case of small datasets more synthetic data is needed. In order to do so, data augmentation techniques are used. It is common knowledge that the more data a deep learning algorithm has access to, the more effective it can be. Even

[1] Source code and dataset (gestureset) are available on the GitHub: https://github.com/olpotkin/DNN-Gesture-Classifier

4

Fig. 3. Random samples from the dataset

Table 1. Parameters of images from dataset: before and after preprocessing

Parameter	Original image	Preprocessed image
Format	.PNG	.PNG
Width (px)	128	64
Height (px)	128	64
Color space	RGB	Grayscale
Color Depth	3	1

when the data is of lower quality, algorithms can actually perform better, as long as useful data can be extracted by the model from the original data set [13].

Following operation were applied in order to get more synthetic images (Fig. 4):

- random rotation (all directions, +/- 10 degrees);
- random resized crop.

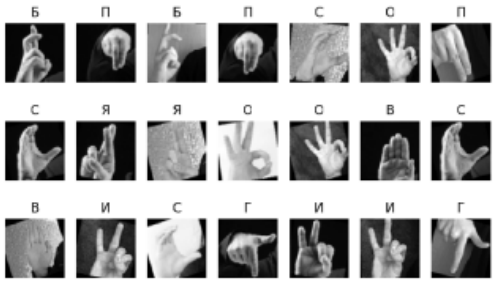

Fig. 4. Random samples from the dataset

In order to minimize overfitting of the classifier, the dataset was divided into 3 parts:

- training set contains about 80% of the dataset (1671 images);

– test set, contains about 20% of the data set (400 images) and used for evaluation of the model (these samples never used during training and cross-validation processes);
– cross-validation set, contains about 20% of the training set (300 images).

4 Neural network architecture for the classification task and performance review

In this section described CNN architecture applied for classification task on the abstract level. As a benchmark were taken two models: LeNet-5 CNN architecture [14] and static gesture classifier from [15].

The main disadvantage of LeNet-5 is overfitting in some cases and no built-in mechanism to avoid this. So, benchmark architecture was improved by adding dropout layers. Improved LeNet-5 architecture for gesture classification task was represented in [15]. The main disadvantage of this model is a comparably low performance on the test set (91.38%).

New improved architecture is more complex, includes more convolutions and units in dense layers (Fig. 5).

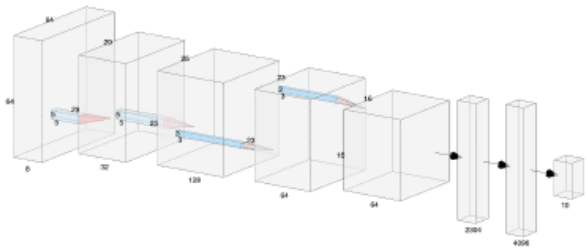

Fig. 5. Improved CNN architecture for the gestures classification task

Training and test procedures were performed on the benchmark classifiers. Results of benchmark classifiers and improved classifier are shown in Table 2.

Table 2. Classification results: benchmarks vs current implementation

CNN	Training Loss	Training Accuracy	Test Loss	Test Accuracy
LeNet-5	0.0681	0.9985	0.5134	0.8708
Static Gesture Classifier from [15]	0.1411	0.9369	0.2385	0.9138
Imroved Classifier	0.1310	0.9433	0.2119	0.9360

The neural network is developed using the PyTorch framework with the following hyper-parameters after the optimization: learning rate is 0.001, batch size is 20, the number of epochs is 24, the metrics – accuracy, the optimizer type – Adam.

5 Conclusion/Future Work

As a result of the research, a unique dataset with 10 classes and more than 2000 unique images was developed and published as well as software pipeline for detecting, tracking and classification of static hand gestures of the Russian Sign Language in a video stream using computer vision and deep learning techniques (Fig. 6).

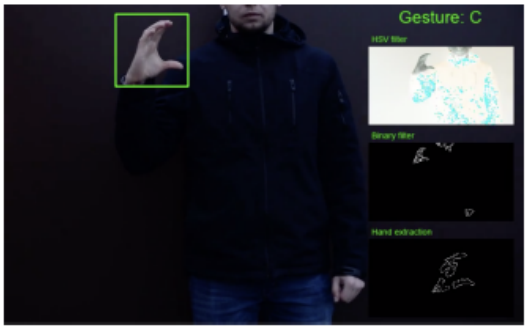

Fig. 6. Gesture control Interface

The solution includes a hand detection module that uses a color mask, a gesture tracking module, a static gestures classification module in the detected region of the image based on convolutional neural network, as well as an auxiliary image preprocessing module and dataset augmentation module. For neural network design development PyTorch framework was used.

The classifier demonstrates an accuracy of classification in 93.6% on the test dataset that is better than the result of the previous version – 91.38% and LeNet-5 classifier - 87.08 %. The represented results of accuracy are sufficient to be a strong basis for the initial industrial prototype of gesture control interface and further research in this direction.

In the next generation of the project, the semantic segmentation approach for classification task will be used with Fully Convolutional Network [16]. It assumes another way for dataset labeling and requires more data. In order to solve this problem new images will be created using new synthetic data techniques (random perspective transformation, random noise, Generative Adversarial Nets [17], etc.)

References

1. Krüger, J., Lien, T., Verl, A.: Cooperation of human and machines in assembly lines. CIRP Annals-Manufacturing Technology, pp. 628-646, 2009
2. Bauer, A., Wollherr, D., Buss, M.: Human–robot collaboration: a survey. International Journal of Humanoid Robotics, pp. 47-66, 2008
3. DailyTechInfo: DICE - gesture-based control HMI from Mersedes-Benz. [https://dailytechinfo.org/auto/3291-dice-sistema-zhestovogo-upravleniya-avtomobilem-ot-mersedes-benz.html], 2012
4. Urmson, C., Dolgov, D., Nemec, P.: Driving pattern recognition and safety control. [https://patents.google.com/patent/US8634980], 2011
5. Shumilov, A., Philippovich, A.: Gesture-based animated CAPTCHA. Information and Computer Security, Vol. 24 Iss 3, pp. 242-254, 2015
6. Letessier, J., Bérard, F.: Visual tracking of bare fingers for interactive surfaces. Proceedings of the 17th annual ACM symposium on User interface software and technology, pp. 119-122, 1970
7. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. Pattern Analysis and Machine Intelligence, pp. 509-522, 2002
8. Doliotis, P., Stefan, A., McMurrough, C., Eckhard, D., Athitsos, V.: Comparing gesture recognition accuracy using color and depth information. Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '11, 2011
9. Nalepa, J., Grzejszczak, T., Kawulok, M.: Wrist Localization in Color Images for Hand Gesture Recognition. Man-Machine Interactions 3, pp 79-86, 2014
10. Habili, N., Lim, C., Moini, A.: Segmentation of the Face and Hands in Sign Language Video Sequences Using Color and Motion Cues. IEEE Transactions on Circuits and Systems for Video Technology, 14(8), pp. 1086–1097, 2004
11. Shaik, K., Ganesan, P., Kalist, V., Sathish, B.: Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space. Procedia Computer Science, 57, pp. 41–48, 2016
12. Förstner, W.: Image Preprocessing for Feature Extraction in Digital Intensity, Color and Range Images. Lecture Notes in Earth Sciences, pp. 165–189, 2000
13. Wang, J., Perez, L.: The Effectiveness of Data Augmentation in Image Classification using Deep Learning. Stanford University, 2017
14. LeCun, Y., Jackel, L., Bottou, L.: Learning algorithms for classification: a comparison on handwritten digit recognition. AT&T Bell Laboratories, 1995
15. Potkin, O., Philippovich, A.: Static Gestures Classification Using Convolutional Neural Networks on the Example of the Russian Sign Language. Supplementary Proceedings of the Seventh International Conference on Analysis of Images, Social Networks and Texts (AIST 2018), pp. 229-234, 2018
16. Long, J., Shelhamer, E., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431-3440, 2015
17. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B. : Generative Adversarial Nets. Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014