|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | ИУ «Информатика и системы управления» |

|  |  |
| --- | --- |
| КАФЕДРА | ИУ-1 «Системы автоматического управления» |

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

|  |
| --- |
|  |
| ***Система распознавания жестов*** |
| ***на видеопоследовательности*** |
| ***для реализации интерфейсов*** |
| ***человеко-машинного взаимодействия*** |
|  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ1-81Б |  | 28/05/2022 |  | Д.И. Юдаков |
|  | (Группа) |  | (Подпись, дата) |  | (И.О. Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель | 28/05/2022 |  | К.В. Парфентьев |
|  | (Подпись, дата) |  | (И.О. Фамилия) |

|  |  |  |  |
| --- | --- | --- | --- |
| Нормоконтролер | 28/05/2022 |  | Т.Ю. Цибизова |
|  | (Подпись, дата) |  | (И.О. Фамилия) |

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Заведующий кафедрой | | | | | | | ИУ-1 | | |
|  | | | | | | | (Индекс) | | |
|  |  | | | |  | К.А. Неусыпин | | | |
|  | | | | | | (И.О. Фамилия) | | | |
|  | « | 11 | » | февраля | | | 20 | 22 | г. |

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент группы | | | ИУ1-81Б | | | | |  | | | |
|  | | | | | | | | | | | |
| Юдаков Дмитрий Игоревич | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | |
|  | | | | | | | | | | | |
| Тема квалификационной работы | | | | | | | | Система распознавания жестов на | | | |
| видеопоследовательности для реализации интерфейсов человеко-машинного | | | | | | | | | | | |
| взаимодействия | | | | | | | | | | | |
|  | | | | | | | | | | | |
|  | | | | | | | | | | | |
| Источник тематики (НИР кафедры, заказ организаций и т.п.) | | | | | | | | | | | |
|  | | | | | | | | | | | |
|  | | | | | | | | | | | |
|  | | | | | | | | | | | |
|  | | | | | | | | | | | |
| Тема квалификационной работы утверждена распоряжением по факультету | | | | | | | | | | | |
|  | № |  | | от | « |  | » |  | 20 |  | г. |
|  | | | | | | | | | | | |
| ***Часть 1.*** | Сделать обзор существующих подходов в распознавании жестов. | | | | | | | | | | |
| Рассмотреть основные сенсорные технологии, использующиеся для сбора данных о жесте, | | | | | | | | | | | |
| выявить достоинства и недостатки каждого типа. | | | | | | | | | | | |
| Описать технологии для решения проблемы идентификации жестов, привести к каждой | | | | | | | | | | | |
| подходящий к ней датчик. | | | | | | | | | | | |
| Проанализировать методы отслеживания жестов, провести их сравнение на различных | | | | | | | | | | | |
| видеофрагментах. | | | | | | | | | | | |
| Рассмотреть основные методы классификации жестов, описать преимущества и недостатки | | | | | | | | | | | |
| каждого метода. | | | | | | | | | | | |
| Привести примеры и описать существующие системы распознавания жестов. | | | | | | | | | | | |
|  | | | | | | | | | | | |

|  |  |
| --- | --- |
| ***Часть 2.*** | Проанализировать нейросетевые технологии в детектировании и |
| идентификации кисти человека. | |
| Сделать обзор теории по искусственным нейронным сетям, функциям активации, видам | |
| искусственных нейронных сетей. | |
| Рассмотреть свёрточную нейронную сеть и её принципы работы. | |
| Рассмотреть рекуррентную нейронную сеть, её принципы работы и области применения. | |
| Изучить использование искусственных нейронных сетей в детектировании объектов. | |
| Подробно описать работу детекторов YOLO и SSD, сравнить их с остальными. | |
| Разобрать применение SSD в задаче детектирования кисти и её последующего описания. | |
| ***Часть 3.*** | Спроектировать систему распознавания жестов. |
| Реализовать интерфейс для сбора данных по статическим и динамическим жестам, | |
| собрать некоторое количество данных по нескольким классам. | |
| Провести обучение искусственной нейронной сети классификации статических жестов | |
| по собранному набору данных, проанализировать результаты. | |
| Провести обучение нейронной сети по большому набору данных жестов из открытых | |
| источников, проанализировать результаты. | |
| Провести обучение нейронной сети классификации динамических жестов, | |
| проанализировать результаты. | |
| Протестировать систему распознавания жестов на примере управления курсором мыши, | |
| выявить недостатки и предложить пути их исправления. | |

***Оформление квалификационной работы:***

|  |  |  |
| --- | --- | --- |
| Расчетно-пояснительная записка на | ***115*** | листах формата А4. |
| Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.) | | |
| Презентация на 20 слайдах | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | | |
|  | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | « | 11 | » | февраля | 20 | 22 | г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| В соответствии с учебным планом выпускную квалификационную работу выполнить в | | | | | | | |
| полном объеме в срок до | « | 28 | » | мая | 20 | 22 | г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Руководитель ВКР** |  | 11/02/2022 |  | К.В. Парфентьев |
|  |  | (Подпись, дата) |  | (И.О. Фамилия) |
|  |  |  |  |  |
| **Студент** |  | 11/02/2022 |  | Д.И. Юдаков |
|  |  | (Подпись, дата) |  | (И.О. Фамилия) |

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **ФАКУЛЬТЕТ** | ИУ | |  |  | | **КАФЕДРА** | ИУ-1 | |  |  | | **ГРУППА** | ИУ1-81Б | |  | УТВЕРЖДАЮ   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Заведующий кафедрой | | | | | | | ИУ-1 | | | |  | | | | | | | (Индекс) | | | |  |  | | | |  | К.А. Неусыпин | | | | |  | | | | | | (И.О. Фамилия) | | | | |  | « | 11 | » | февраля | | | 20 | 22 | г. | |

**КАЛЕНДАРНЫЙ ПЛАН**

**выполнения выпускной квалификационной работы**

|  |  |  |
| --- | --- | --- |
| студента: | Юдакова Дмитрия Игоревича | |
|  | (фамилия, имя, отчество) | |
| Тема квалификационной работы | | Система распознавания жестов на |
| видеопоследовательности для реализации интерфейсов человеко-машинного | | |
| взаимодействия | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ п/п** | **Наименование этапов выпускной квалификационной работы** | **Сроки выполнения этапов** | | **Отметка о выполнении** | |
| **план** | **факт** | **Должность** | **ФИО, подпись** |
|  | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | |  | | --- | | 11/02/2022 | | дата | | |  | | --- | | 11/02/2022 | | дата | | Руководитель ВКР |  |
|  | 1 часть | |  | | --- | | 26/03/2022 | | дата | | |  | | --- | | 26/03/2022 | | дата | | Руководитель ВКР |  |
|  | Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач | |  | | --- | | 26/03/2022 | | дата | | |  | | --- | | 26/03/2022 | | дата | | Заведующий кафедрой |  |
|  | 2 часть | |  | | --- | | 27/04/2022 | | дата | | |  | | --- | | 27/04/2022 | | дата | | Руководитель ВКР |  |
|  | 3 часть | |  | | --- | | 10/05/2022 | | дата | | |  | | --- | | 10/05/2022 | | дата | | Руководитель ВКР |  |
|  | 1-я редакция работы | |  | | --- | | 15/05/2022 | | дата | | |  | | --- | | 15/05/2022 | | дата | | Руководитель ВКР |  |
|  | Подготовка доклада и презентации | |  | | --- | | 27/05/2022 | | дата | | |  | | --- | | 27/05/2022 | | дата | | Руководитель ВКР |  |
|  | Заключение руководителя | |  | | --- | | 28/05/2022 | | дата | | |  | | --- | | 28/05/2022 | | дата | | Руководитель ВКР |  |
|  | Допуск работы к защите на ГЭК (нормоконтроль) | |  | | --- | | 28/05/2022 | | дата | | |  | | --- | | 28/05/2022 | | дата | | Нормоконтролер |  |
|  | Внешняя рецензия | |  | | --- | | 28/05/2022 | | дата | | |  | | --- | | 28/05/2022 | | дата | |  |  |
|  | Защита работы на ГЭК | |  | | --- | |  | | дата | | |  | | --- | |  | | дата | |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Студент:* | 11/02/2022 |  | *Руководитель работы:* | 11/02/2022 |
|  | (подпись, дата) |  |  | (подпись, дата) |

АННОТАЦИЯ

Расчётно-пояснительная записка на 115 с, 34 рис., 11 табл., 70 ист., 2 прил.

В работе представлена разработка метода распознавания жестов кисти для человеко-машинного взаимодействия

Рассмотрены существующие подходы распознавания жестов. Рассмотрены методы и технологии детектирования, отслеживания и классификации жестов по изображениям. Представлена реализация программы сбора данных для обучения классификации статических и динамических жестов на основе ключевых точек. Проведено обучение искусственных нейронных сетей классификации статических жестов на двух наборах данных, сравнение результатов работы сетей. Проведено обучение нейросети классификации динамических жестов, проанализированы результаты её работы. Представлена реализация системы распознавания жестов кисти, тестирование её в задаче управления курсором мыши, выявлены её недостатки и предложены способы их устранения.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 6](#_Toc105704907)

[ВВЕДЕНИЕ 9](#_Toc105704908)

[ПОСТАНОВКА ЗАДАЧИ 11](#_Toc105704909)

[1. ТЕХНОЛОГИИ ДЛЯ РАСПОЗНАВАНИЯ ЖЕСТОВ 12](#_Toc105704910)

[1.1. Распознавание жестов для человеко-машинного взаимодействия 13](#_Toc105704911)

[1.2. Сенсорные технологии 15](#_Toc105704912)

[1.2.1. Методы на основе получения изображений 15](#_Toc105704913)

[1.2.2. Методы, основанные не на изображениях 16](#_Toc105704914)

[1.2.3. Сравнение сенсорных технологий 17](#_Toc105704915)

[1.3. Идентификация жестов 18](#_Toc105704916)

[1.3.1. Визуальные характеристики 19](#_Toc105704917)

[1.3.2. Алгоритмы обучения 20](#_Toc105704918)

[1.3.3. Модель человека 21](#_Toc105704919)

[1.3.4. Выводы по подходам идентификации жестов 22](#_Toc105704920)

[1.4. Отслеживание жестов 23](#_Toc105704921)

[1.4.1. Отслеживание единичной гипотезы 23](#_Toc105704922)

[1.4.2. Расширенные методы отслеживания 24](#_Toc105704923)

[1.4.3. Сравнение различных подходов отслеживания жестов 25](#_Toc105704924)

[1.5. Классификация жестов 26](#_Toc105704925)

[1.6. Примеры систем распознавания жестов 30](#_Toc105704926)

[2. НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ В ДЕТЕКТИРОВАНИИ И ИДЕНТИФИКАЦИИ КИСТИ ЧЕЛОВЕКА 33](#_Toc105704927)

[2.1. Нейронные сети и их виды 34](#_Toc105704928)

[2.1.1. Свёрточная нейронная сеть 39](#_Toc105704929)

[2.1.2. Рекуррентные нейронные сети 40](#_Toc105704930)

[2.2. Использование искусственных нейронных сетей в детектировании 45](#_Toc105704931)

[2.2.1. YOLO 46](#_Toc105704932)

[2.2.2. SSD 47](#_Toc105704933)

[2.3. Применение SSD в задаче детектирования кисти. Идентификация кисти с помощью ключевых точек 59](#_Toc105704934)

[3. РАЗРАБОТКА ИНТЕРФЕЙСА ДЛЯ УПРАВЛЕНИЯ ЖЕСТАМИ 61](#_Toc105704935)

[3.1. Система распознавания жестов 61](#_Toc105704936)

[3.2. Требования к реализации 62](#_Toc105704937)

[3.3. Подготовка и обучение модели распознавания статических жестов 65](#_Toc105704938)

[3.4. Расширение модели распознавания статических жестов 70](#_Toc105704939)

[3.5. Подготовка и обучение модели распознавания динамических жестов 76](#_Toc105704940)

[3.6. Оценка работы интерфейса 81](#_Toc105704941)

[ЗАКЛЮЧЕНИЕ 84](#_Toc105704942)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 86](#_Toc105704943)

[ПРИЛОЖЕНИЕ А 92](#_Toc105704944)

[ПРИЛОЖЕНИЕ Б 105](#_Toc105704945)

СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

HRC (англ. Human-Robot Collaboration) – кооперация человека и робота.

HCI (англ. Human-Computer Interaction) – взаимодействие человека и машины.

ANN (англ. Artificial Neural Network) или ИНС – искусственная нейронная сеть.

MLP (англ. MultiLayer Perceptron) – многослойный перцептрон.

CNN (англ. Convolutional Neural Network) – свёрточная нейронная сеть.

RNN (англ. Recurrent Neural Network) – рекуррентная нейронная сеть.

LSTM (англ. Long Short-Term Memory) – долгая краткосрочная память.

R-CNN (англ. Region-based CNN) – свёрточная нейронная сеть, основанная на предложении регионов.

GUI (англ. Graphical User Interface) – графический интерфейс пользователя.

ВВЕДЕНИЕ

В последнее время концепция взаимодействия между людьми и роботизированными машинами вызвала множество научных интересов. Вместо того, чтобы машины заменили людей на рабочих местах, сотрудничество человека с ними – это направление, позволяющее сохранить рабочие места, организовав им возможность работать вместе. Кооперация людей и роботов может освободить людей от тяжёлых задач, если будут созданы эффективные каналы связи между ними.

Роботизированные системы уже стали ключевыми компонентами в различных отраслях промышленности. В последнее время концепция Human-Robot Collaboration (HRC) значительно подогрела исследовательские интересы. Литературные примеры предполагали, что работники-люди обладают несравненными навыками решения проблем и сенсорно-двигательными способностями, но имеют ограниченную силу и точность [23, 35]. Однако роботизированные системы имеют высокую стойкость к усталости, скорость, повторяемость и лучшую производительность, но они ограничены в гибкости. HRC может устанавливать каналы связи между людьми и роботами для повышения общей эффективности. В идеале связка людей и машин должна работать так же, как и человеческая совместная команда.

Компьютерные технологии машин уже позволяют обрабатывать и анализировать информацию подобно человеку: распознавать текст, изображения, анализировать звуки и мелодии, произносить осмысленные предложения, распознавать голосовые команды и реагировать на прикосновения пальцев. Но одной из самых приоритетных и наиболее трудных задач в области информационных технологий и интеллектуальных систем является задача распознавания жестов.

В данной работе идёт речь о детектировании кисти руки, извлечения её конфигурации с помощью ключевых точек и распознавание жеста для генерации в соответствии с ним сигнала выхода. В ходе работы исследуются различные методы и техники детектирования кисти человека и распознавания жеста, в частности – методы с использованием средств глубокого обучения. Важным требованием для реализации является минимизация времени задержки на обработку каждого кадра (скорость работы) и точность идентификации жеста.

**Актуальность** рассматриваемой тематики обусловлена возможностью применения предлагаемого подхода распознавания жестов для создания канала связи с роботизированными системами без тактильного контакта и голосовой идентификации команд, а также простотой с точки зрения конечного пользователя.

ПОСТАНОВКА ЗАДАЧИ

Поставленной **целью** является разработка и реализация системы детектирования кисти, идентификации и классификации жеста.

Для достижения цели необходимо решить ряд **задач**:

1. Сделать обзор существующих подходов распознавания жестов, провести их сравнение, выделить особенности.
2. Описать методы детектирования кисти человека и идентификации жеста.
3. Разработать и реализовать систему распознавания жестов.
4. Провести анализ разработанной системы, выявить недостатки и описать пути их устранения.

1. ТЕХНОЛОГИИ ДЛЯ РАСПОЗНАВАНИЯ ЖЕСТОВ

Традиционно методы разделения времени или пространственного разделения применяются в системах HRC, что снижает производительность как для рабочих, так и для роботов [35]. Чтобы создать эффективную команду HRC, в качестве примеров можно проанализировать группы сотрудничества человека с человеком. В совместной работе и сотрудничестве людей существует две теории: теория совместных намерений и теория обучения [11, 13, 14, 64]. Чтобы применить теории в команде HRC, существует три опытных подхода, которые приносят пользу команде HRC:

* Все члены команды в команде HRC должны иметь один и тот же план исполнения;
* Все члены команды в команде HRC должны знать контекст среды совместной работы;
* Команда HRC должна иметь структурированные способы общения.

В данной работе основное внимание уделяется именно третьему пункту, то есть структурированным способам коммуникации.

Жесты – один из способов общения. Мимика, жесты рук и позы тела являются эффективными каналами связи в сотрудничестве между людьми [8][23]. Жесты могут быть разделены на три типа [42]:

* Жесты тела (действия или движения всего тела);
* Жесты рук и кистей рук (позы рук, жесты рук);
* Головные и лицевые жесты (кивания или вращения головы, подмигивания, губы);

Распознавание жеста относится к математической интерпретации человеческих движений вычислительным устройством. Чтобы взаимодействовать с человеком, машины должны правильно понимать человеческие жесты и действовать в соответствии с жестом в достаточной степени точности. В среде HRC должен быть доступен естественный способ коммуникации между роботами и людьми.

1.1. Распознавание жестов для человеко-машинного взаимодействия

Чтобы распознать жесты в контексте человеко-машинного взаимодействия, полезно исследовать общую и упрощённую модель обработки информации. Как показано на рис. 1, Parasuraman представил [45] упрощённую модель обработки информации, состоящую из четырёх этапов.

|  |
| --- |
|  |
| Рис. 1. Четырёхступенчатая модель обработки информации. |

На основе общей модели (рис. 1), предлагается конкретная модель распознавания жестов в человеко-машинном взаимодействии. Как показано на рис. 2, существует пять основных частей, связанных с распознаванием жестов для HRC: сбор данных с датчиков, идентификация жестов, отслеживание жестов, классификация жестов и отображение жестов.

|  |
| --- |
|  |
| Рис. 2. Модель процесса распознавания жестов при взаимодействии человека и машины. |

Поясним каждый из этапов:

* Сбор данных с датчиков: жестовые данные фиксируются датчиками;
* Идентификация жеста: в каждом фрейме жест отображается из необработанных данных;
* Отслеживание жеста: локализованный жест отслеживается во время движения. Для статических жестов отслеживание жестов не требуется;
* Классификация жестов: отслеживание движения жестов классифицируется в соответствии с заранее определёнными типами жестов;
* Отображение жеста: результат распознавания жестов преобразуется в управляющие команды.

1.2. Сенсорные технологии

Процесс распознавания основан на сопоставлении некоторых признаков данных с существующими признаками. Распознавание жестов — это технология ввода данных жестами в реальном времени, реализуемая с помощью датчиков движения. Чтобы выполнить действие, не нужно использовать клавиши или касаться сенсорного экрана. Датчики движения устройства воспринимают и интерпретируют жесты пользователя как основной источник ввода данных. Перед процессом распознавания жестов рабочие данные должны быть собраны датчиками для дальнейшей их обработки. Существуют две основные категории сбора данных: основанные на изображениях и не основанные на изображениях.

Рассмотрим эти категории подробнее.

1.2.1. Методы на основе получения изображений

Технологии часто вдохновляются природой: человек использует глаза, что распознавать жесты, поэтому для машин разумно использовать камеры, что «видеть» жесты. Подходы, основанные на работе с изображениями, составляют четыре категории:

* *Маркер*. В маркерном подходе датчиком является оптическая камера. В большинстве решений на основе маркеров пользователям необходимо носить видимые маркеры.
* *Камера*. В начале 90-х годов исследователи начали анализировать жесты при помощь одиночной камеры [58, 59]. Недостатком однокамерного подхода является ограничение угла обзора, что влияет на надёжность системы [30]. Однако в недавних исследованиях применялся однокамерный подход к высокоскоростному распознаванию жестов [34]. Система использует датчик скорости и специально разработанный процессор визуальных вычислений для достижения высокоскоростного распознавания жестов.
* *Стереокамера*. Чтобы добиться надёжного распознавания жестов, исследователи предложили стереоподход к созданию 3D-зрения. Здесь определяются стереокамерных подходы как приложения, которые используют две оптические камеры для создания информации о глубине 3D. Многие стереоподходы к камерам соответствовали аналогичному процессу [17, 41]. Хотя системы стереокамер улучшили устойчивость во внешней среде, они по-прежнему имеют таким проблемы, как сложность вычислений и трудности с калибровкой [65].
* *Сенсор глубины*. В последнее время технологии глубокого зондирования быстро развиваются. Датчики глубины обладают рядом преимуществ по сравнению с традиционными стереокамерами. Например, можно предотвратить проблемы настройки калибровки и условий освещения [60]. Кроме того, выходной информацией датчика глубины является информация о глубине 3D, по сравнению с информацией о цвете, эта информация упрощает проблему идентификации жестов [42].

1.2.2. Методы, основанные не на изображениях

В распознавании жестов в течение долгого времени доминировали датчики на основе изображений. Недавние разработки в микроэлектромеханических системах (МЭМС) и сенсорных технологиях значительно улучшили технологии распознавания жестов, основанные не на изображениях. Рассмотрим основные из них:

* *Перчатки*. Жестовые интерфейсы на основе перчаток также используются для распознавания жестов. Обычно методы на основе перчаток требуют проводного подключения акселерометров и гироскопов. Однако громоздкая перчатка с проводами может вызывать проблемы в человеко-машинном взаимодействии, и данные подходы имеют сложности в процедурах калибровки и настройки.
* *Браслет (носимая электроника)*. Другая бесконтактная технология использует сенсоры на браслетах. Сенсоры установлены на браслете или аналогичных носимых устройствах. Сенсоры на браслетах позволяют использовать беспроводные технологии и датчики электромиограммы, что позволяет избежать подключения кабелей. Сенсоры должны контактировать с запястьем, при этом руки и пальцы пользователя могут быть свободны.
* *Бесконтактные устройства*. В третьем типе технологий, не связанных с изображениями, используются датчики, не предназначенные для ношения. Бесконтактные датчики могут обнаруживать жесты без контакта с человеческим телом. Google представил Project Soli, системы радиолокационного контроля и распознавания жестов на радиочастотном спектре (радар) [22]. Устройство способно распознавать разные жесты рук на небольшом расстоянии. В течение многих лет MIT является ведущим новатором в области распознавания жестов. Технология электрического поля была впервые разработана в MIT [57]. Недавно Adib [2][3][4] из MIT представил систему WiTrack и RF-Capture, которая отслеживает движение пользователя по радиочастотным сигналам, отражённым от человеческого тела. Система способна захватывать человеческие жесты даже из другой комнаты через стену с точностью до 20 см. Таким образом, технологии, не пригодные для ношения, являются перспективными и быстрорастущими сенсорными технологиями для распознавания жестов.

1.2.3. Сравнение сенсорных технологий

Таблица 1 содержит сравнение различных сенсорных технологий, показаны преимущества и недостатки различных подходов. Понятно, что ни один датчик не подходит для всех приложений. На основе вышеуказанных методов можно выделить два вида сенсорных технологий:

* Сенсоры для использования внутри помещений: датчики глубины являются наиболее перспективными технологиями на основе изображений, они обладают преимуществами простоты калибровки, установки и скорости обработки данных.
* Бесконтактные сенсоры являются наиболее перспективной технологией среди подходов, основанных на не изображении. Они не требуют прямого контакта с пользователями.

|  |  |  |
| --- | --- | --- |
| Таблица 1. | | |
| Достоинства и недостатки различных сенсорных технологий. | | |
|  | Достоинства | Недостатки |
| Маркеры | Низкая вычислительная нагрузка | Маркеры на теле пользователя |
| Камера | Простота установки | Низкий уровень надёжности |
| Стереокамера | Надёжность | Сложность вычислений,  трудности с калибровкой |
| Камера глубины | Высокая частота кадров | Разрешение зависит от мощности света и отражения,  невозможно использовать на расстоянии более 2 метров. |
| Перчатка | Скорость отклика,  точность трекинга | Громоздкое устройство с  проводами |
| Браслет (носимая электроника) | Скорость отклика,  область действия | Браслет должен контактировать с человеческим телом |
| Бесконтактные устройства | Не требует контакта с телом | Низкое разрешение, технология недостаточно зрелая |

1.3. Идентификация жестов

Идентификация жеста – первый шаг в процессе распознавания жестов после получения необработанных данных, полученных с датчиков. Идентификация жеста означает обнаружение жестовой информации и сегментацию соответствующей жестовой информации из необработанных данных. Популярные технологии для решения проблемы идентификации жестов основаны на визуальных особенностях, алгоритмах обучения и человеческих моделях.

1.3.1. Визуальные характеристики

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

* *Цвет.* Цвет – это простая визуальная функция для идентификации жестов из фоновой информации. Однако на системы распознавания жестов на основе цветов сильно влияют освещение и тени в сложной среде человеко-машинного взаимодействия. Ещё одна распространённая проблема в обнаружении цвета кожи заключается в том, что цвет кожи человечка сильно различается среди человеческих рас. Из-за вышеперечисленных проблем, в современных подходах, цвет кожи рассматривается только как один из многих параметров при идентификации жестов.
* *Локальные признаки.* В распознавании жестов на основе изображения условия освещения сильно влияют на качество идентификации жестов. Поэтому многие исследователи используют метод локальных признаков, которые не чувствителен к условиям освещения. Локальный подход к объектам – это детализированный подход на основе текстур. Он раскладывает изображение на более мелкие области, которые не соответствуют частям тела. Одной из наиболее важных локальных функций является преобразование признаков инвариантных объектов (SIFT) [40]. Метод SIFT является вращательным, трансляционным, масштабируемым и частично осветляющим инвариантом. Существует несколько подобных методов локальных признаков, например, SURF и ORB, предложенные в более поздние годы [9, 50]. Как правильно, подходы к локальным особенностям также рассматриваются только 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.
* *Форма и контур.* Другой интуитивный и простой способ идентификации жестов – использование уникальной формы и контуры человеческого тела в среде человеко-машинного взаимодействия. Существенный вклад в определение формы и соответствия был внесён Belongie и другими в [10]. Они ввели метод дескриптора контекста формы. Они ввели метод дескриптора контекста формы. Дескриптор контекста формы используется для обнаружения похожих фигур на разных изображениях. Разработка датчика глубины даёт возможность точно измерять форму поверхности. 3D-модели, созданные на основе технологий, позволяют очень детально представлять форму человеческого тела [5].
* *Движение.* В конкретной среде человеко-машинного взаимодействия человек является единственным движущимся объектов в массиве необработанных данных. Поэтому движение является полезной функцией для обнаружения человеческих жестов. Оптических поток является ключевой технологией идентификации жестов на основе движения. Оптический поток не нуждается в вычитании фона, что является преимуществом по сравнению с подходами на основе формы и контура. Дарал и Турау [62] представили знаменитый метод гистограмм ориентированных градиентов (HOG). Дескрипторы HOG делят изображения кадры изображения на блоки, для каждого из которых вычисляется гистограмма. Среди подходов, основанных не на изображении, распознавание жестов на основе движения является популярным методом. Пороговые значения и фильтрация обычно применяются к необработанным данным с датчика для идентификации человеческих жестов.

1.3.2. Алгоритмы обучения

Недавняя тенденция идентификации жестов заключается в применении алгоритмов обучения, особенно для обнаружения статических жестов, которые могут быть представлены в одном кадре. Методы визуальной функции основаны на различных визуальных особенностях, в то время как алгоритмы обучения используют алгоритмы машинного обучения для определения жестов из необработанных данных датчика. Хотя некоторые алгоритмы основаны на методах визуальных признаков, удаление фона изображения не всегда необходимо для корректной работы алгоритмов. В системах распознавания жестов широко применяются алгоритмы обучения, такие как метод опорных векторов (SVM, Support-Vector Machine), искусственные нейронные сети (ANN, Artificial Neural Network) и случайные решения (RDF, Random Decision Forest).

1.3.3. Модель человека

В отличие от вышеупомянутых подходов, модельных подход использует явную модель человеческого тела для восстановления позы человеческого тела. Поэтому модельный подход также называется генеративным подходом. Поскольку идентификация жестов на основе модели человеческого тела даёт преимущество упрощения процесса классификации жестов, подход с использованием модели человека стал популярным решением при работе с датчиками глубины. Современные исследования идентификации жестов на основе модели человека можно разделить на два типа: идентификация модели рук и идентификация модели скелета тела.

* *Модель руки*. Существует три основных подхода к моделированию ручных моделей: модель формы, трёхмерная модель и модель ручного скелета. Подход, основанные на форме, соответствует предварительно построенной формы руки из наблюдений. Подход 3D-модель интерпретирует проблему обнаружения руки как проблему оптимизации, которая минимизирует различия между моделью 3D-руки и наблюдаемой рукой [44].
* *Модель телесного скелета*. Чтобы определить жест тела, подробная человеческая модель бесполезна. Однако, в задаче идентификации жестов тела, модель именно скелета тела является популярным подходом. Модель скелета тела представляет собой упрощённую модель человеческого тела, которая сохраняет информацию о суставах тела. Большая часть литературы по алгоритмом для обработки скелета тела основана на информации о глубине, собранной с датчиков глубины.

1.3.4. Выводы по подходам идентификации жестов

В таблице 2 представлены наиболее подходящие способы идентификации жестов для каждого популярного датчика.

|  |  |
| --- | --- |
| Таблица 2. | |
| Методы идентификации жестов для разных жестов. | |
| Датчик | Метод идентификации жестов |
| Системы с одной камерой | В системах с одной камерой могут быть реализованы методы визуальных признаков и алгоритмы обучения. Для обеспечения надежной работы следует применять алгоритмы обучения. Чтобы добиться более быстрой обработки изображений, следует применять метод визуальных признаков. |
| Датчик глубины | Поскольку метод человеческой модели использует и упрощает информацию из облака точек, метод человеческой модели является лучшим вариантом для систем с датчиками глубины. |
| Браслет (сенсоры на базе браслета) | Никакие визуальные методы не могут применяться для систем на базе браслетов. Обычно собранные данные требуют базовой фильтрации для идентификации жестов, а алгоритмы обучения будут реализованы на более поздней стадии классификации жестов. |
| Бесконтактные датчики | Бесконтактные датчики также принимают данные сигналов вместо изображений. Из-за того, что радиочастотные сигналы содержат шумы, современные системы фильтрации и обработки должны быть реализованы в бесконтактных системах. |

Из-за характера человеко-машинного взаимодействия работники-люди являются наиболее важными его членами. Несмотря на понимание жестов человеческого тела, подход моделирования человеческого тела также будет контролировать человеческие движения, что обеспечивает безопасную среду для совместной работы роботов и людей. Как упоминалось ранее, модель скелета тела упрощает телесную модель человека, а информация о суставах тела легко извлекается. Более того, скелетные подходы могут упростить последующий процесс классификации жестов. Поэтому в настоящее время модели с использованием скелетного подхода является подходящим решением для распознавания жестов в системах человеко-машинного взаимодействия.

1.4. Отслеживание жестов

При распознавании жестов понятие отслеживания используется по-разному в разной литературе. В данной работе понятие отслеживания определяется как процесс поиска временных соответствий между кадрами. В частности, производится фокусировка на проблеме отслеживания жестов, которая связывает идентифицированный жест в предыдущих кадрах с текущим фреймом. Для статических жестов, то есть жестов, которые могут быть представлены одним кадром, отслеживание жестов не требуется.

1.4.1. Отслеживание единичной гипотезы

Отслеживание единичной гипотезы относится к оценке наилучшего соответствия при согласовании с минимальной ошибкой. Поэтому при использовании единичной гипотезы отслеживания жест представлен только одной гипотезой. Большинство усовершенствованных алгоритмов отслеживания основаны на технологиях единичной гипотезы.

* *Средний сдвиг*. Отслеживание методом среднего сдвига – это базовая технология отслеживания. Метод выполняет сопоставление с гистограммами цвета RGB. Для каждого нового кадра средний сдвигающий трекер сравнивает расстояние Бхаттачария между гистограммами целевого окна нового кадра и кадрами старого кадра. Полное математическое объяснение этого подхода можно найти в [15].
* *Фильтр Калмана*. Фильтр Калмана (KF) представляет собой рекурсивный алгоритм реального времени, используемый для оптимальной оценки лежащих в основе состояний ряда шумовых и неточных результатов измерений, наблюдаемых с течением времени. Полный математический вывод KF представлен в [33]. В настоящее время KF активно развивается и применяется в различных областях, таких как аэрокосмическая промышленность, робототехника и экономика.
* *Расширенные фильтры Калмана*. KF предполагает, что вектор состояния является линейной моделью. Расширенный фильтр Калмана (EKF, Extended Kalman Filter) является функциональным алгоритмом отслеживания, даже если модель нелинейна [25]. Другим алгоритмом, который решает ту же проблему под другим углом, является сигма-точечный фильтр Калмана (UKF, Unscented Kalman Filter) [66]. UKF решает проблему, применяя детерминированный подход с взвешенной выборкой. Распределение состояний представлено с использованием минимального набора выбранных точек выборки.

1.4.2. Расширенные методы отслеживания

За последнее время появилось много продвинутых методов отслеживания. Некоторые из этих передовых методов использовали часть алгоритмов отслеживания, упомянутых выше. Другие методы улучшили отслеживаемость с помощью алгоритмов обнаружения или обучения.

* *Расширенное отслеживание модели*. Для долгосрочных проблем отслеживания многие алгоритмы отслеживания терпят неудачу, поскольку цель поддерживает фиксированные модели. Расширенное отслеживание модели сохраняет целевое поведение или внешний вид из последних кадров изображения, поэтому для целевой оценки зарезервировано больше целевой информации.
* *Отслеживание путём обнаружения*. Ещё один вид алгоритмов отслеживания строится на базе алгоритмов обучения идентификации жестов. Для этих алгоритмов отслеживания в кадрах изображений применяется классификатор или детектор, чтобы идентифицировать жест из справочной информации [36]. Одним из характерных подходов является трекер отслеживания, обучения и обнаружения. Этот подход объединяет результаты детектора объекта с устройством оптического отслеживания потока. Ещё одна типичная технология отслеживания по обнаружения – применение множественного обучения экземпляров [7]. Алгоритм обучения может повысить надёжность трекера и уменьшить количество параметров.

1.4.3. Сравнение различных подходов отслеживания жестов

Smeulders [56] представил результат теста различных алгоритмов отслеживания жестов. Итоговый результат является оценкой нормализованного значения F-меры. F-мера даёт представление о среднем охвате ограничивающего блока объекта и истинной ограничивающей рамки, поэтому чем выше значение F-меры, тем лучше качество отслеживания. На рис. 3 представлены результаты испытаний на разных видеофрагментах.

|  |
| --- |
|  |
| Рис. 3. Результат теста алгоритмов отслеживания на разных видеофрагментах. |

Kalman Appearance Tracker (трекер появления Калмана) и Mean Shift Tracker (трекер среднего сдвига) относятся к трекеру единичной гипотезы. Отслеживание с помощью Sampling Tracker (трекер отбора) и Incremental Visual Tracker (инкрементный визуальный трекер) относится к расширенной модели отслеживания. Трекеры отслеживания (Tracking, Learning and Detection Tracker) и трекер обнаружения нескольких экземпляров (Multiple Instance Learning Tracker) относятся к методам отслеживания с помощью обнаружения. Легко заметить, что трекеры одиночной гипотезы менее производительны, чем остальные. Таблица 3 содержит резюме различных подходов к отслеживанию жестов.

|  |  |
| --- | --- |
| Таблица 3. | |
| Описание основных подходов к отслеживанию. | |
| Подход | Описание |
| Единичная гипотеза | Быстрый и простой алгоритм. Подходит для отслеживания одного жеста в контролируемой среде. |
| Множественные  гипотезы | Возможность отслеживания нескольких целей одновременно. Подходит для отслеживания нескольких жестов в контролируемой среде. |
| Расширенная модель трекинга | Целевая история сохраняется и доступна для целевой оценки. Подходит для долговременного отслеживания жестов. |
| Отслеживание путём обнаружения | Алгоритм обучения повышает надежность и снижает уровень шума. Этот комбинированный подход имеет предпочтительную производительность в тестах. Подходит для отслеживания жестов в сложной среде. |

1.5. Классификация жестов

Классификация жестов – это последний и самый важный шаг в распознавании жестов. Большинство человеческих жестов – это динамические жесты. Один динамический жест всегда состоит из нескольких кадров. Чтобы классифицировать динамические жесты, классификация жестов должна выполняться после или вместе с отслеживанием жестов. Рассмотрим основные методы классификации жестов.

* *Метод K-ближайших соседей (K-Nearest Neighbours)*. Является фундаментальным и базовым алгоритмом классификации жестов, который классифицирует входные данные в соответствии с ближайшими примерами обучения [46].
* *Скрытая Марковская модель.* Является комбинацией ненаблюдаемой цепи Маркова и стохастического процесса.
* *Метод опорных векторов.* Метод опорных векторов является дискриминационным классификатором, определяемым разделительной гиперплоскостью [27][54]. Границы решения классификации определяются путём максимизации расстояния от границы. Оптимальная разделительная гиперплоскость максимизирует запас обучающих данных. Примеры обучения, наиболее близкие к оптимальной гиперплоскости, называются вспомогательными векторами. Общей проблемой для данного метода является то, что число опорных векторов растёт линейно с размером обучающего набора.
* *Метод ансамбля.* Основное предположение ансамблевого метода состоит в том, что ансамбли более точны, чем слабые отдельные классификаторы. Одним из известных методов ансамбля является бустинг [52][53]. Алгоритм бустинга начинается с нескольких слабых классификаторов. Слабые классификаторы применяются многократно. В обучающей итерации часть обучающих образцов используется в качестве входных данных. После итерации обучения создается новая граница классификации. После всех итераций алгоритм бустинга объединяет эти границы и сливается в одну конечную границу предсказания. Одним из известных ансамблевых методов является алгоритм Адабуст (Adaboost). Существенным преимуществом алгоритма Адабуст является то, что Адабуст не нуждается в большом количестве обучающих данных.
* *Динамическое деформирование временных рядов (Dynamic Time Warping, DTW).* Является оптимальным алгоритмом выравнивания для двух последовательностей. DTW генерирует кумулятивную матрицу расстояний, которая искажает последовательности нелинейным способом. Первоначально DTW использовался для распознавания речи, но в последнее время активно используется в распознавании жестов [6] [12].
* *Искусственные нейронные сети (ANN).* Являются семейством моделей обработки информации, основанных на биологических нейронных сетях. ANN состоит из множества взаимосвязанных обрабатывающих нейронов, которые работают параллельно. Каждый нейрон получает входные данные, обрабатывает их и даёт выходные данные. ANN может использоваться для оценки функций, зависящих от большого количества входных данных.
* *Глубокое обучение (Deep Learning).* Является быстро развивающейся отраслью машинного обучения. Глубокое обучение позволяет компьютеру моделировать данные с абстракциями высокого уровня, используя множественную нейронную сеть на уровне обработки. Более того, в отличие от традиционных алгоритмов обучения, глубокое обучение не требует ручной подготовки данных, что позволяет использовать преимущества экспоненциально увеличивающихся объёмов доступных данных и вычислительных мощностей.

В таблице 4 представлены преимущества и недостатки подходов к классификации жестов.

|  |  |  |
| --- | --- | --- |
| Таблица 4. | | |
| Преимущества и недостатки подходов к классификации жестов. | | |
| Подход | Преимущества | Недостатки |
| Метод K-ближайших  соседей | Простота | K-параметр следует выбирать осторожно |
| Скрытая  Марковская модель | Гибкость обучения и проверки, прозрачность модели | Необходимо отрегулировать множество свободных параметров |
| Метод опорных векторов | Могут применяться различные функции ядра | Число опорных векторов растёт линейно с размером обучающего набора |
| Метод  ансамбля | Не требуется большого количества данных для обучения | Легко переобучить, чувствительность к шумам и выбросам |
| Динамическое деформирование временных рядов | Надёжное нелинейное выравнивание между образцами | Сложность по времени и по объёму данных |
| Искусственные нейронные сети | Может обнаруживать сложные нелинейные зависимости между переменными | Принцип «чёрного ящика», не может использоваться при наличии небольшого набора данных для обучения |
| Глубокое  обучение | Не нуждается в хорошей подготовке признаков, превосходит другие методы машинного обучения | Требуется большое количество обучающих данных и значительные вычислительные мощности |

Одной из тенденций является подход глубокого обучения. Основными ограничениями глубокого обучения являются ограниченные вычислительные мощности. Однако экспоненциально возрастающая вычислительная мощность может легко решить эту проблему. Количество приложений классификации жестов, основанных на глубоком обучении, быстро растет. Другая тенденция заключается в объединении различных алгоритмов классификации. Каждый алгоритм классификации имеет свои преимущества и недостатки. Чтобы использовать это, различные классификаторы могут быть объединены для достижения лучшей производительности. Также было замечено, что важно координировать алгоритмы классификации жестов с помощью алгоритмов идентификации и трекинга жестов.

1.6. Примеры систем распознавания жестов

Одним из примеров распознавания жеста при помощи вычисления расстояние между точками является «Система распознавания жестов в реальном времени с цветными перчатками» (Real-Time Hand-Tracking with a Color Glove), которую создали Роберт Ванг (Robert Y. Wang) и Йован Попович (Jovan Popovic) из лаборатории компьютерных наук и искусственного интеллекта Массачусетского технологического института [67].

Распознавание жестов возможно при надевании цветных перчаток, по которым программа распознаёт жесты рук при помощи веб-камеры по наличию цветных пикселей на изображении. После обнаружения этих пикселей происходит вычисление расстояния между ними. Вычислив это расстояние, программа выдаёт информацию о распознанном жесте. Вид цветных перчаток изображён на рисунке 4.

|  |
| --- |
|  |
| Рис. 4. Перчатки, при помощи которых программа распознаёт жесты. |

Этот способ распознавания очень интересен в своей реализации, но он имеет огромный недостаток: необходимость надевания специальных перчаток для того, чтобы программа смогла выделить цветные пиксели. Без них программа неработоспособна.

Следующим примером является распознавание жестов при помощи сигналов с тензодатчиков. Представителем этой группы являются перчатки GRASP – говорящие перчатки для глухонемых (Glove-base Recognition of Auslan using Simple Processing) [29].

Их принцип работы следующий: к компьютеру подключены две перчатки при помощи кабелей. Программа обрабатывает сигналы тензодатчиков и, в зависимости от информации, полученной с этих датчиков, выдаёт информацию о распознанном жесте. Внешний вид устройства приведён на рисунке 5.

|  |
| --- |
|  |
| Рис. 5. Перчатка, при помощи которой программа распознаёт жесты. |

Данный способ реализации так же имеет недостаток в необходимости надевания специальной перчатки, без которой программа не сможет распознать жест.

Еще одним примером реализации распознавания жестов является программа [1], преобразующая жесты русской ручной азбуки в текстовый вид, работает непосредственно с веб-камерой, при помощи которой определяется положение руки в кадре. После этого строится «скелет» человеческой руки, по которому программа распознаёт жест и переводит его в букву русской азбуки. Вид работы программы приведён на рисунке

6.

|  |
| --- |
| page8image2919337056 |
| Рис. 6. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид. |

Эта программа является наиболее удобной в использовании, так как не требует наличия различных устройств, надетых на руку, но работает только с полутоновыми изображениями, то есть изображениями в оттенках серого. Такие изображения очень светочувствительны, что является большим недостатком этой программы.

2. НЕЙРОСЕТЕВЫЕ ТЕХНОЛОГИИ В ДЕТЕКТИРОВАНИИ И ИДЕНТИФИКАЦИИ КИСТИ ЧЕЛОВЕКА

Человеко-машинное взаимодействие (HCI – англ. Human-Computer Interaction) – это междисциплинарное научное направление, изучающее взаимодействие между людьми и машинами. Предметом HCI является изучение, планирование и разработка методов взаимодействия человека с машиной, где в роли машины может выступать персональный компьютер, компьютерная система больших масштабов, система управления процессами и т.д. [16]. Под взаимодействием понимается любая коммуникация между человеком и машиной. Одним из методов HCI, получившим широкое распространение в последние годы, является взаимодействие, основанное на жестах человека [51, 68].

Распознавание жестов на основе ключевых точек предоставляет не только гибкий способ классификации жеста, поскольку в этом случае жест определяется лишь последовательностью координат этих точек, но и дополнительными возможностями, таким как привязка расположения какого-либо объекта к расположению конкретной ключевой точки в кадре (относительном или абсолютном).

Использование одного из класса методов искусственного интеллекта – машинного обучения – для решения самых разнообразных задач приобретает всё более широкий спектр, особенно это характерно для тех задач, найти алгоритмическое решение которых либо не представляется возможным, либо оно неразумно и неэффективно. Машинное обучение характеризуется не прямым решением задачи, а обучением за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятности, теории графов и различные техники работы с данными в цифровой форме.

Подмножеством машинного обучения является глубокое обучение, являющееся совокупностью методов машинного обучения, основанных на обучении представлениям, а не специализированных алгоритмах под конкретные задачи, то есть на основе искусственных нейронных сетей (ИНС). Процесс обучения называется *глубоким*, так как структура ИНС состоит из входного, выходного и нескольких скрытых слоёв. Каждый слой содержит части, преобразующие входные данные в сведения, которые следующие слой может использовать для определённой задачи. Рассмотрим нейронные сети и их виды подробнее.

2.1. Нейронные сети и их виды

*Нейронная сеть* (или искусственная нейронная сеть, ИНС) – это математическая модель, а также её программная или аппаратная реализация, построенная по принципу организации и функционирования *биологических нейронных сетей* – сетей нервных клеток живого организма. Если же говорить проще, то нейронная сеть – это последовательность *нейронов*, соединённых между собой синапсами. На рис. 7а изображена схема простой нейросети, состоящей из трёх слоёв: зелёным обозначены входные нейроны, голубым – скрытые нейроны (скрытый слой) и жёлтым – выходной нейрон.

Математически, нейрон представляют как некоторую нелинейную функцию от единственного аргумента – линейной комбинации всех входных сигналов. Данную функцию называют *функцией активации* или *передаточной функцией*. На рис. 7б изображена схема нейрона, цифрой 1 обозначены нейроны, выходные сигналы которых поступают на вход данного, цифрой 2 – сумматор входных сигналов, цифрой 3 – вычислитель передаточной функции и цифрой 4 – нейроны, на входы которых подаётся выходной сигнал данного.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 7. Схема простой нейронной сети (а), искусственного нейрона (б). | |

Описать нейрон уравнением можно так:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | (1) |
| где: | | | |
|  | | | |
| и |  | соответственно сигналы на входах и веса входов | |
|  |  | индуцированное локальное поле, | |
|  |  | передаточная функция. | |

Возможные значения сигналов на входах нейрона обычно считают заданными в отрезке . Дополнительный вход и соответствующий ему вес используются для инициализации нейрона. Под инициализацией понимается смещение активационной функции нейрона по горизонтальной оси, то есть формирование порога чувствительности нейрона. Иногда к выходу нейрона специально добавляют некую случайную величину, называемую сдвигом.

Передаточная функция нейрона определяет зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах. В большинстве случаев она является монотонно возрастающей, а область значений лежит в или . Нейрон полностью характеризуется своей передаточной функцией. Рассмотрим основные типы передаточных функций.

*Линейная передаточная функция*. Задаётся формулой:

Иногда возможен сдвиг функции по обеим осям. Одним из преимуществ является дифференцируемость на всей числовой оси.

*Пороговая передаточная функция*. Иногда называют *функцией Хевисайда*. Представляет собой резкий скачок на некоторый уровень . Задаётся формулой:

Здесь – сдвиг функции активации относительно горизонтальной оси. Недостаток – не дифференцируема на всей оси абсцисс.

*Сигмоидальная передаточная функция*. Является самой часто используемой функцией. Существует несколько видов её математической записи. *Логистическая функция*:

где t – параметр функции, определяющий её крутизну. Важным достоинством этой функции является простота её производной:

Особенностью нейронов с такой передаточной характеристикой является их усиление сильных сигналов существенно меньше, чем слабых. Это предотвращает насыщение от больших сигналов.

Другой математической записью сигмоидальной функции является *гиперболический тангенс*:

Она отличается от логистической функции тем, что её область значений лежит в интервале

*Радиально-базисная функция передачи*. Она принимает в качестве аргумента расстояние между входным вектором и некоторым наперёд заданным центром активационной функции. Значение этой функции тем выше, чем ближе входной вектор к центру. В качестве такой функции можно взять *функцию Гаусса*:

|  |  |  |
| --- | --- | --- |
|  | | |
| где: | | |
|  |  | расстояние между центром и вектором входных сигналов , |
|  |  | определяет скорость спада функции при удалении вектора от центра и называется *шириной окна*, |
|  |  | определяет сдвиг активационной функции по оси абсцисс. |

В качестве расстояния между векторами могут быть использованы различные метрики, обычно используется евклидово расстояние.

Также в качестве передаточных функций могут использоваться: *экспонента*, *тригонометрический синус*, *модуль, квадратичная*.

Каждая нейронная сеть включает в себя первый входной слой нейронов. Этот слой не выполняет каких-либо преобразований и вычислений, его задача состоит в том, чтобы принимать и распределять входные сигналы по остальным нейронам. Таким образом, этот слой единственный, являющийся общим для всех типов нейросетей, а критерием для деления является уже дальнейшая структура:

* + - 1. *Однослойная структура нейронной сети*. Представляет собой структуру взаимодействия нейронов, в которой сигналы со входного слоя сразу направляются на выходной слой, который не только преобразует сигнал, но и сразу же выдаёт ответ. Как уже было сказано, первый входной слой только принимает и распределяет сигналы, а нужные вычисления происходят уже во втором слое. Входные нейроны являются объединёнными с основным слоем с помощью синапсов с разными весами, обеспечивающими качество связей.
      2. *Многослойная нейронная сеть*. Данная сеть помимо выходного и входного слоёв имеет ещё несколько скрытых промежуточных слоёв. Число этих слоёв зависит от степени сложности нейронной сети. Соответствующие решения обладают большими возможностями, если сравнивать с однослойными, ведь в процессе обработки данных каждый промежуточный слой – это промежуточный этап, на котором осуществляется обработка и распределение информации.

Кроме количества слоёв, нейронные сети можно классифицировать по направлению распределения информации по синапсам между нейронами:

* + - 1. *Нейросети прямого распространения (однонаправленные)*. В этой структуре сигнал перемещается строго по направлению от входного слоя к выходному. Движение сигнала в обратном направлении не осуществляется и в принципе невозможно. Сегодня разработки этого плана распространены широко и успешно решают задачи распознавания образов, прогнозирования и кластеризации.
      2. *Рекуррентные нейронные сети (с обратными связями)*. Для данных типов сетей характерно движение сигнала и в прямом, и в обратном направлении. В итоге результат выхода способен возвращаться на вход. Выход нейрона определяется весовыми характеристиками и входными сигналами и дополняется предыдущими выходами, снова вернувшимися на вход. Этим нейросетям присуща функция кратковременной памяти, на основании чего сигналы восстанавливаются и дополняются во время их обработки.
      3. *Радиально-базисные функции*.
      4. *Самоорганизующиеся карты*.

Основными критериями выбора типов среди огромного многообразия искусственных нейронных сетей являются способность эффективно работать с изображениями и с последовательностями. Сверточная нейронная сеть удовлетворяет первому критерию, а рекуррентная – второму. Рассмотрим их поподробнее.

2.1.1. Свёрточная нейронная сеть

Свёрточная нейронная сеть (CNN – Convolutional Neural Network) – специальная архитектура нейронных сетей, предложенная Яном Лекуном в 1988 году и нацеленная на эффективное распознавание образов. Как следует из названия, вместо обычных слоёв в ней используется операция *свёртки*, суть которой состоит в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки по элементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

Работа свёрточной нейронной сети обычно интерпретируется как переход от конкретных особенностей изображения к более абстрактным деталям, и далее к ещё более абстрактным деталям вплоть до выделения понятий высокого уровня. При этом сеть самонастраивается и вырабатывает сама необходимую иерархию абстрактных признаков (последовательности карт признаков), фильтруя маловажные детали и выделяя существенное.

Подобная интерпретация носит скорее метафорический или иллюстративный характер. Фактически «признаки», вырабатываемые сложной сетью, малопонятны и трудны для интерпретации настолько, что на практике суть этих признаков даже не пытаются понять, тем более «подправлять», а вместо этого для улучшения результатов распознавания меняют структуру и архитектуру сети. Так, игнорирование системой каких-то существенных явлений может говорить о том, что либо не хватает данных для обучения, либо структура сети обладает недостатками, и система не может выработать эффективных признаков для данных явлений.

Оператор свёртки с ядром , получая на вход изображение , на выходе формирует матрицу :

Для многоканального изображения (или просто трёхмерной матрицы) используется трёхмерное ядро свёртки :

Так же обычно в свёрточную нейронную сеть включаются оператор пулинга (MaxPool), для которого определяются два параметра: размер ядра () и сдвиг (:

Пример свёрточной сети [38] для распознавания цифр из набора данных MNIST изображён на рис. 8.

|  |
| --- |
|  |
| Рис. 8. Пример свёрточной сети. |

Вначале применяются разных ядер свёртки размерности , получая из матрицы трёхмерную матрицу . Затем используется оператор пулинга , который уменьшает матрицу в два раза, сводя её к размеру . Ещё разных ядер свёртки той же размерности и слой пулинга «сжимают» матрицу до размера , получая величин, которые подаются на выходной слой с нейронами.

2.1.2. Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN – англ. Recurrent Neural Network) – вид нейронных сетей, где связи между элементами образуют направленную последовательность. Благодаря этому появляется возможность обрабатывать серии событий во времени или последовательные пространственные цепочки. В отличие от многослойных перцептронов, рекуррентные сети могут использовать свою внутреннюю память для обработки последовательностей *произвольной длины*, поэтому сети RNN применимы в таких задачах, где нечто целостное разбито на части, например: распознавание рукописного текста, речи или жестов.

Любая рекуррентная нейронная сеть имеет форму цепочки повторяющихся модулей нейронной сети. В обычной RNN структура одного такого модуля очень проста, например, он может представлять собой один слой с функцией активации (гиперболический тангенс) (рис. 9).

|  |
| --- |
|  |
| Рис. 9. Структура обычной RNN. |

Одна из привлекательных идей RNN состоит в том, что потенциально они имеют возможность связывать предыдущую информацию с текущей задачей, так, например, знания о предыдущем кадре видео могут помочь в понимании текущего кадра. Однако предоставление такой возможности зависит от разрыва между актуальной информацией и точкой её применения. По мере роста этого расстояния, RNN теряют способность связывать информацию. Однако модификация RNN – LSTM – не имеет такой проблемы.

Долгая краткосрочная память (LSTM – англ. Long Short-Term Memory) – особая разновидность архитектуры рекуррентных нейронных сетей, способная к обучению долговременным зависимостям. Данный вид был разработан в 1997 году специально для решения проблемы долговременной зависимости, то есть они способны запоминать информацию на долгие периоды времени [28].

Структура LSTM по аналогии с RNN также напоминает цепочку, но её модули выглядят иначе (рис. 10): модули имеют четыре особым образом взаимодействующих между собой слоя вместо одного.

|  |
| --- |
|  |
| Рис. 10. Структура LSTM. |

Ключевой компонент LSTM – это состояние ячейки (cell state) – горизонтальная линия, проходящая по верхней части схемы (рис. 11).

|  |
| --- |
|  |
| Рис. 11. Состояние ячейки в LSTM. |

Состояние ячейки напоминает ленту на конвейере, проходя напрямую через всю цепочку, оно участвует лишь в нескольких линейных преобразованиях, то есть информация «течёт» по ней, не подвергаясь изменениям. Тем не менее, LSTM может удалять информацию из состояния ячейки, посредством регулирования структурами, называемыми фильтрами (gates). Фильтры позволяют пропускать информацию на основании некоторых условий и состоят из слоя сигмоидальной нейронной сети и операции поточечного умножения. Сигмоидальный слой имеет диапазон выходных значений, лежащий в отрезке , который обозначает, какую долю каждого блока информации следует пропустить дальше по сети. Ноль в данном случае означает «не пропускать ничего», единица – «пропустить всё». В LSTM три таких фильтра, позволяющих защищать и контролировать состояние ячейки.

Первый шаг в LSTM – определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый «слоем фильтра забывания» (forget gate layer) (рис. 12а). В зависимости от и этот слой на основе выхода сигмоидального слоя принимает решение о сохранении каждого числа из состояния ячейки . Следующий шаг – решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием «слой входного фильтра» (input layer gate) (рис. 12б) определяет, какие значения следует обновить. Затем -слой строит вектор новых значений-кандидатов , которые можно добавить в состояние ячейки. И наконец старое состояние ячейки заменяется на новое состояние , умножая его на и прибавляя (рис. 12в). Это новые значения-кандидаты, умноженные на – на сколько необходимо обновить каждое из значений состояния. Последним этапом работы LSTM сети является отбор информации, получаемой на выходе. Выходные данные будут основаны на применении сигмоидального слоя и текущем состоянии ячейки, пройденном через -слой (рис. 12г).

|  |  |
| --- | --- |
|  |  |
| а) | б) |
|  |  |
| в) | г) |
| Рис. 12. Фильтры в LSTM: «слой фильтра забывания» (a), «слой входного фильтра» (б), замена старого состояния на новое (в), получение информации на выход (г). | |

Таким образом, хорошая приспособленность к обучению на задачах классификации и относительная невосприимчивость к длительности временных разрывов даёт LSTM преимущество по отношению к альтернативным рекуррентным нейронным сетям.

2.2. Использование искусственных нейронных сетей в детектировании

Когда нейронные сети в компьютерном зрении не применялись так широко, для детектирования объектов на изображении использовался достаточно простой подход. Тренировался бинарный классификатор, который мог определить есть ли на картинке объект определённого типа или нет. После того как классификатор был натренирован, по изображению «проходило» скользящее окно (sliding window) и для каждого прямоугольника, обрамлённого этим окном, применялся этот бинарный классификатор. Поскольку на одном изображении объекты могут быть и маленькими, и большими, то необходимо было либо запускать окно разных размеров, либо из исходного изображения строить пирамиду разных масштабов и запускать окно по каждому слою пирамиды. Основная задача была найти такие *особенности (features)*, которые бы позволили натренировать хороший бинарный классификатор.

С внедрением свёрточных нейронных сетей произошёл скачок в поиске хороших особенностей. Например, уже в 2013 появилась статья [55], про сеть Overfeat, где подход со скользящим окном обобщался на свёрточную нейронную сеть, применяемую в качестве классификатора.

Затем появилась линейка R-CNN, Fast R-CNN, Faster R-CNN в статьях [20], [21], [49], и расширения этого подхода. Здесь, вместо скользящего окна, используется предварительный детектор для поиска объектов, а уже после этого нейросеть определяет класс объекта и уточняет его прямоугольник. Точность у данных детекторов довольно высока, но они всё ещё довольно медленные.

Наконец, третий подход появился в статье [48] – YOLO (You Only Look Once) и потом продолжился в [37] – SSD (Single Shot Detector). Для ускорения процесса детектирования в этих работах предлагается не искать объекты, пробегая по изображению скользящим окном или каким-то вариантом поискового алгоритма, а выбрать некоторое фиксированное количество прямоугольников и для каждого проверить наличие в нём объекта, и, если объект найден, то определить какого он класса и уточнить ограничивающую его рамку (*bounding box*). Например, в [48] на изображение накладывается решётка, в каждой клетке которой проверяется наличие центра какого-нибудь объекта. Рассмотрим данный подход подробнее.

2.2.1. YOLO

Авторы [48] предложили вместо того, чтобы искать объекты с последующей их проверкой классификатором, разбить изображение равномерной сеткой и для каждой ячейки определять есть ли в ней центр какого-то объекта, какого он класса и его точный прямоугольник. То есть для каждой ячейки определять:

1. Набор из пятёрок вида: , где относительные координаты центра объекта внутри ячейки, размеры объекта, уверенность сети, что объект есть и его прямоугольник задан правильно;
2. вероятностей (по количеству классов объектов), что объект с центром в данной ячейке принадлежит соответствующему классу.

Итак, получая на вход изображение, на выходе сеть должна выдавать тензор размера . В общем виде, схема сети представлена на рис. 13.

|  |
| --- |
|  |
| Рис. 13. Схема сети YOLO. |

В частности, экспериментально обосновав, авторы посоветовали:

1. Подавать на вход изображения размером ;
2. Разбивать их при помощи решётки с ;
3. Для каждой ячейки выдавать предсказанных прямоугольника;
4. Так как в качестве набора данных для обучения берётся Pascal VOC, в котором размечено 20 классов, то .

Таким образом, на выходе сети будет тензор . В качестве базовой сети для выделения особенностей авторы берут GoogLeNet с некоторыми упрощениями и в двух вариантах: свёрточных слоя или для особо быстрого детектирования.

2.2.2. SSD

YOLO показал неплохие результаты: скорость выросла, а качество упало не очень сильно. Поэтому авторы статьи [37] решили улучшить этот подход, повысив качество, при этом не снижая скорости.

Было предложено, так же, как и в YOLO, не искать объекты на фотографии, а задать фиксированный набор прямоугольников и проверять наличие объекта в каждом из них. Но, поскольку в YOLO претендентов было слишком мало, и к тому же особенности брались с последнего свёрточного слоя сети, а значит мелкие объекты могли в такой ситуации потеряться, то было решено брать особенности с разных свёрточных слоёв, что привело к увеличению набора претендентов. Для того, чтобы заставлять сеть тренироваться, отдавая для каждого объекта ровно одно обнаружение, дополнительно после детектирования применяется NMS (Non-Maximum Suppression). На рисунке 14 изображено сравнение моделей SSD и YOLO. В SSD модель в конце базовой сети добавляются несколько слоёв, предсказывающих смещения к *анкерам* (default box) различных масштабов и соотношений сторон и связанные с ними вероятности. SSD с размером входного изображения значительно превосходит аналог YOLO по точности и по скорости.

|  |
| --- |
|  |
| Рис. 14. Сравнение моделей детекторов SSD и YOLO. |

Для работы SSD необходимо только входное изображение и рамки истинности для каждого объекта во время обучения (рис. 15а). С помощью свертки оценивается небольшой набор (например, ) полей по умолчанию с разными соотношениями сторон в каждом месте на нескольких картах признаков с разными масштабами (например, (рис. 15б) и (рис. 15в)). Для каждой рамки по умолчанию прогнозируется как смещение формы, так и достоверность для всех категорий объектов . Во время обучения сначала сопоставляются эти рамки по умолчанию с рамками истинности. Например, совпали две рамки по умолчанию с котом и одно с собакой на рис. 15а, которые рассматриваются как положительные, а остальные – как отрицательные. Потеря модели представляет собой взвешенную сумму между потерей локализации (Smooth L1 [21]) и потерей достоверности (например, Softmax).

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а) | б) | в) |
| Рис. 15. Пример работы SSD детектора. | | |

Как уже было сказано, подход SSD основан на свёрточной сети с прямой связью, которая создаёт набор ограничивающих рамок (прямоугольников) фиксированного размера и оценивает наличие экземпляров класса объектов в этих рамках с последующим подавлением немаксимумов для получения результатов окончательных обнаружений. Модель SSD изображена на рис. 16.

|  |
| --- |
|  |
| Рис. 16. Подробное описание модели SSD. |

Первые слои нейросети основаны на стандартной архитектуре VGG-16, используемой для высококачественной классификации изображений (*базовая сеть)*. Затем в сеть добавляется вспомогательная структура для создания обнаружений со следующими ключевыми особенностями:

1. *Мультимасштабные карты ключевых точек для детектирования*.

В конец усечённой базовой сети добавляются свёрточные слои ключевых точек. Эти слои постепенно уменьшаются в размере и позволяют прогнозировать обнаружение в нескольких масштабах. Свёрточная модель для прогнозирования обнаружений различается для каждого слоя объектов (Overfeat [55] и YOLO [48]), которые работают с картой признаков в одном масштабе;

1. *Свёрточные «предсказатели» для детектирования.*

Каждый добавленный слой признаков (или, возможно, существующий слой признаков из базовой сети) может создавать фиксированный набор прогнозов обнаружения с использованием набора свёрточных фильтров. Для слоя признаков размера с каналами базовым элементом для прогнозирования параметров потенциального обнаружения является небольшое ядро , которое даёт либо оценку для категории, либо смещение формы относительно координат анкера. В каждом из положений, где ядро применяется, оно выдает выходное значение. Выходные значения смещения прямоугольника измеряются относительно положения анкера относительно каждого местоположения карты признаков (архитектура YOLO использует промежуточный полносвязный слой вместо свёрточного фильтра для этого шага).

1. *Анкеры (*или *default boxes).*

Набор анкеров связывается с каждой ячейкой карты признаков для нескольких карт признаков в начале нейросети. Анкеры разбивают карту признаков свёрткой так, что положение каждой ячейки относительно соответствующей ячейки фиксировано. В каждой ячейке карты признаков прогнозируется смещение относительно форм анкеров в ячейке, а также оценки для каждого класса, которые указывают на присутствие экземпляра класса в каждой из этих рамок. В частности, для каждой рамки из в заданном месте вычисляются оценки класса и 4 смещения относительно исходной формы анкера. В результате получается фильтров, которые применяются вокруг каждого местоположения на карте признаков, которые дают выходов для карты признаков размера .

Ключевое отличие обучения SSD от обучения типичного детектора, который использует предложения регионов (то есть предлагается какой-либо регион изображения и классифицируется) заключается в эталонной информации, которая должна быть назначена конкретным выходам в фиксированном наборе выходов детектора. Некоторая вариация этого также требуется для обучения YOLO [48] и для этапа предложения региона Faster R-CNN [49] и Multibox [18]. Как только это назначение определено, функция потерь и обратное распространение применяются сквозным образом. Обучение также включает в себя выбор набора анкеров и масштабов по умолчанию для обнаружения, а также стратегии аугментации и получения сложных «неверных» тренировочных данных (hard negative mining).

*Стратегия сопоставления*. Во время обучения необходимо определять, какие анкеры соответствуют верному обнаружению и тренировать сеть соответствующим образом. Каждый верный прямоугольник, сопоставляется с каждым анкером, различающимся по местоположению, соотношению сторон и масштабу и с наилучшим коэффициентом Жаккара (как в MultiBox [18]). Затем, в отличие от MultiBox, в SSD сопоставляются анкеры c истинными прямоугольниками с коэффициентом Жаккара выше порогового значения (. Это упрощает задачу обучения, позволяя сети предсказывать высокие значения для нескольких перекрывающихся прямоугольников, а не требовать от неё выбора только одного с максимальным перекрытием.

*Цель обучения*. Цель обучения SSD получена из цели Multibox [18], но расширена до обработки нескольких категорий объектов. Пусть индикатор для сопоставления -того анкера с -м истинным прямоугольником категории . Из приведённой выше стратегии сопоставления следует, что можно иметь . Общая целевая функция потерь представляет собой взвешенную сумму потерь локализации (loc) и потерь достоверности (conf):

|  |  |  |
| --- | --- | --- |
|  | | |
| где: | | |
|  |  | число совпавших анкеров. Если , то потери |
|  |  | весовой коэффициент, устанавливающийся равным с помощью перекрёстной проверки (кросс-валидации или cross-validation). |
|  |  |  |

* Потери локализации – это потери Smooth L1 [19] между предсказанным прямоугольником () и истинным прямоугольником (). Подобно Faster R-CNN [49] происходит регрессия к смещениям для центра ограничивающего блока по умолчанию (), а также для его ширины () и высоты ():
* Потери достоверности – это потери Softmax по достоверности нескольких классов ():

|  |  |  |
| --- | --- | --- |
|  | | |
| где: | | |
|  |  | . |

*Выбор масштабов и соотношений сторон для анкеров*. Для обработки различных масштабов объектов некоторые методы [26, 55] предполагают обработку изображения разных размеров и последующее объединение результатов. Однако, используя карты признаков из нескольких разных слоёв в одной сети для прогнозирования, можно имитировать тот же эффект, а также использовать общие параметры для всех масштабов объектов. Предыдущие работы [24, 39] показали, что использование карт признаков из нижних слоёв может улучшить качество семантической сегментации, поскольку нижние слои захватывают более мелкие детали входных объектов. Точно так же в [32] показано, что добавление главного контекста, объединённого из карты признаков, может помочь сгладить результаты сегментации. Руководствуясь этими методами, в SSD для обнаружения используются как нижняя, так и верхняя карта признаков.

Известно, что карты признаков с разных уровней внутри сети имеют разные (эмпирические) размеры рецептивного поля [70]. К счастью, в структуре SSD анкеры не обязательно должны соответствовать фактическим рецептивным полям каждого слоя. Мозаика анкеров разрабатывается таким образом, чтобы определённые карты признаков научились реагировать на определённые масштабы объектов. Предположим, нужно использовать карт признаков для предсказания. Масштаб блоков по умолчанию для каждой карты признаков рассчитывается как:

где , означающие, что самый нижний слой имеет масштаб , а самый верхний слой и все слои меду ними расположены через равные промежутки. Для анкеров устанавливаются различные пропорции и обозначаются как . Можно вычислить ширину () и высоту для каждого анкера. Для соотношения сторон также добавляется анкер, масштаб которого и в результате получается анкеров для каждого местоположения карты признаков. Центр каждого анкера устанавливается в , где это размер -ого квадрата карты признаков, .

На практике можно также спроектировать распределение анкеров, чтобы наилучшим образом соответствовать конкретному набору данных. Вопрос о том, как спроектировать оптимальную мозаику, также остаётся открытым.

Комбинируя прогнозы для всех анкеров с разными масштабами и соотношениями сторон из всех местоположений на многих картах признаков, получается разнообразный набор прогнозов, охватывающий различные размеры и формы входных объектов. Например, на рис. 15 собака сопоставляется с блоком по умолчанию на карте признаков , но не с каким-либо блоком по умолчанию на карте признаков . Это связано с тем, что эти блоки имеют разные масштабы и не соответствуют блоку собаки, и поэтому во время тренировки считаются негативами.

*Получение сложных «неверных» тренировочных данных (hard negative mining).* После этапа сопоставления большинство анкеров являются неверными, особенно когда количество возможных анкеров велико. Это вносит значительный дисбаланс между положительными и отрицательными обучающими примерами. Вместо того, чтобы использовать все отрицательные примеры, они сортируются по их наибольшей потере достоверности для каждого анкера и выбираются так, чтобы соотношение между отрицательными и положительными примерами было не более чем . Было обнаружено [37], что это приводит к более быстрой оптимизации и более стабильному обучению.

*Аугментация данных.* Чтобы сделать модель более устойчивой к различным размерам и формам входных объектов, каждое обучающее изображение случайным образом изменяется по одному из следующих правил:

* + Используется исходное изображение без изменений;
  + Из исходного изображения вырезается фрагмент таким образом, чтобы минимальный коэффициент Жаккара с объектами составлял или ;
  + Из исходного изображения вырезается случайный фрагмент.

Размер каждого фрагмента выборки лежит в отрезке от исходного размера изображения, а соотношение сторон находится в диапазоне от до . Перекрывающаяся часть истинного прямоугольника сохраняется, если его центр находится в фрагменте выборки. После вышеупомянутого шага размер каждого фрагмента изменяется до фиксированного и фрагмент горизонтально отражается с вероятностью , в дополнение к применению некоторых фотометрических искажений, подобных описанным в [26].

Как уже было отмечено, существует два класса методов детектирования объектов на изображениях: один основан на скользящих окнах, а другой – на классификации предложений регионов. До появления свёрточных нейронных сетей уровень развития этих двух подходов – модели DPM (Deformable Part Model [19]) и выборочного поиска [63] – имели сопоставимую производительность. Однако после резкого улучшения, внесённого R-CNN (Region-Based Convolutional Neural Network) [21], который сочетает в себе выборочный поиск предложений регионов и пост-классификацию на основе свёрточной сети, методы обнаружения объектов на основе предложений регионов стали преобладающими. Сравнение подходов представлено в таблице 5.

|  |  |  |
| --- | --- | --- |
| Таблица 5. | | |
| Сравнение подходов на тесте Pascal VOC2007. | | |
| Метод | mAP, % | FPS |
| Faster R-CNN (VGG16) |  |  |
| Fast YOLO |  |  |
| YOLO (VGG16) |  |  |
| SSD300 |  |  |
| SSD512 |  |  |

SSD300 – единственный метод, способный работать в реальном времени и набравший более 70% mAP (mean Average Precision – средняя точность предсказания), являющийся компромиссом между скоростью и качеством работы.

Первоначальный подход R-CNN был улучшен различными способами. Первый набор подходов улучшает качество и скорость пост-классификации, поскольку требует классификации тысяч кадров изображений, что дорого и затратно по времени. SPPnet [26] значительно ускоряет исходный подход R-CNN за счёт ввода слоя объединения слоя объединения пространственных пирамид, который более устойчив к размеру и масштабу региона и позволяет слоям классификации переиспользовать признаки, вычисленные по картам признаков, сгенерированным для нескольких разрешений изображения. Fast R-CNN [21] расширяет возможности SPPnet, так что он может точно настраивать все уровни от начала до конца, минимизируя потери как для достоверности, так и для регрессии ограничивающей рамки, которая впервые была представлена в MultiBox [18] для изучения объектности.

Второй набор подходов улучшает качество генерации предложений, используя глубокие нейронные сети. В самых последних работах, таких как MultiBox [18], предложения региона выборочного поиска, которые основаны на низкоуровневых признаках изображения, заменены предложениями, сгенерированными непосредственно из отдельной глубокой нейронной сети. Это дополнительно повышает точность обнаружения, но приводит к несколько сложной настройке, требующей обучения двух нейронных сетей с зависимостью между ними. Faster R-CNN [49] заменяет предложения выборочного поиска предложениями, полученными из сети региональных предложений (RPN), и вводит метод интеграции RPN с Fast R-CNN путем чередования точной настройки общих свёрточных слоёв и слоёв прогнозирования для этих двух сетей. Таким образом, предложения регионов используются для объединения признаков среднего уровня, а последний этап классификации обходится дешевле. SSD очень похож на сеть региональных предложений (RPN) в Faster R-CNN, поскольку также используется фиксированный набор блоков (по умолчанию) для прогнозирования, аналогично полям привязки в RPN. Но вместо того, чтобы использовать их для объединения признаков и оценки другого классификатора, оценка для каждой категории объектов получается одновременно в каждом блоке. Таким образом, этот подход позволяет избежать осложнений, связанных с объединением RPN с Fast R-CNN, и его легче обучать, быстрее и проще интегрировать в другие задачи.

Другой набор методов, непосредственно связанных с подходом SSD, полностью пропускает шаг предложения и напрямую прогнозирует ограничивающие рамки и достоверность для нескольких категорий. OverFeat [55], глубокая версия метода скользящего окна, предсказывает ограничивающую рамку непосредственно из каждого местоположения самой верхней карты признаков после того, как известны достоверности базовых категорий объектов. YOLO [48] использует всю самую верхнюю карту признаков для прогнозирования как достоверности для нескольких категорий, так и ограничивающих рамок (которые являются общими для этих категорий). Метод SSD попадает в эту категорию, поскольку в нём нет шага предложения, но используются блоки по умолчанию. Однако этот подход является более гибким, чем существующие методы, поскольку существует возможность использовать блоки по умолчанию с разными соотношениями сторон для каждого местоположения признака из нескольких карт признаков в разных масштабах. Если будет использоваться только один блок по умолчанию для каждого местоположения из самой верхней карты признаков, то SSD будет иметь архитектуру, аналогичную OverFeat [55]; если используется вся самая верхняя карта признаков и добавляется полносвязный слой для прогнозов вместо свёрточных предсказателей и не будет явно учитываться несколько соотношений сторон, то приблизительно воспроизведётся архитектура YOLO [48].

Таким образом, резюмируя всё выше сказанное, можно отметить следующие особенности Single Shot детектора:

* Архитектура SSD позволяет детектировать объекты в реальном времени;
* Качество работы близко к Faster R-CNN;
* Детектирование происходит на разных масштабах, что позволяет локализовывать объекты разных размеров;
* Используется большое количество рамок по умолчанию, покрывающих входное изображение на разных масштабах;
* На этапе работы архитектура SSD300 осуществляет детектирование 7308 объектов, большая часть из которых впоследствии фильтруется;

Ключевой же особенностью является использование многомасштабных ограничительных рамок на выходах свёрточных слоёв, прикреплённых к нескольким картам признаков в верхней части сети. Экспериментально подтверждено в [37], что при соответствующих стратегиях обучения, большее количество тщательно выбранных анкеров приводит к повышению производительности.

Для решения задачи детектирования кисти на изображении Single Shot детектор подходит наилучшим образом, поскольку он имеет хорошее соотношение между скоростью работы и точностью. Этот же детектор также используется в библиотеке MediaPipe Hands, использующейся для поиска ключевых точек на руках. Рассмотрим её поподробнее.

2.3. Применение SSD в задаче детектирования кисти. Идентификация кисти с помощью ключевых точек

Одним из применений Single Shot детектора в задаче поиске кисти на изображении является подход, реализованный в фреймворке MediaPipe от Google – MediaPipe Hands [69]. Для определения начального положения рук была разработана модель SSD, оптимизированная для использования в реальном времени. Поскольку обнаружение рук чрезвычайно сложная задача из-за огромного количества их конфигураций, то вместо детектора рук обучался детектор ладоней, так как получать прямоугольники негнущихся объектов, таких как ладони и кулаки, значительно проще, нежели обнаружение рук вместе с пальцами. Кроме того, поскольку ладони являются более мелкими объектами, алгоритм подавления немаксимумов хорошо работает даже для случаев пересечения двух рук, таких как рукопожатия. Более того, ладони могут быть смоделированы с помощью квадратных анкеров, игнорируя другие соотношения сторон, уменьшая количество анкеров в 3-5 раз. Во-вторых, кодирующий-декодирующий экстрактор признаков используется для большей осведомленности о контексте сцены даже для небольших объектов. Наконец, во время обучения минимизируется функция кросс-энтропийных потерь с динамическим масштабированием (focal loss), чтобы поддерживать большое количество анкеров, возникающих из-за большой дисперсии масштаба. С помощью описанных выше методов достигается средняя точность обнаружения ладони . Использование обычной функции кросс-энтропийных потерь и без декодера даёт точность всего .

После обнаружения ладони на всём изображении последующая модель выполняет точную локализацию 21 ключевой точки (рис. 17а) внутри обнаруженных областей руки посредством регрессии (как для определения цифры, написанной рукой), то есть прямого прогнозирования координат. Модель обучается последовательному внутреннему представлению позиции рук и устойчива даже к частично видимым рукам и самопересечениям. Так же стоит отметить, что кроме 21 ключевой точки модель так же определяет вероятность присутствия кисти на кадре и классификацию типа кисти – левой или правой.

|  |  |  |
| --- | --- | --- |
|  | |  |
| а) | б) | |
| Рис. 17. Ключевые точки на руке, определяемые с помощью  MediaPipe Hands (а). Данные для обучения (б). Сверху: реальные изображения рук, аннотированные вручную. Снизу: сгенерированные синтетические изображения рук с аннотациями. | | |

Чтобы получить данные для обучения, вручную было промаркировано около изображений реального мира с координатой, как показано на рис. 17б. А также, чтобы лучше охватить возможные позиции рук и обеспечить дополнительное наблюдение за характером геометрии рук, была визуализирована высококачественная синтетическая модель руки на различных фонах, таким образом синтезируя набор из изображений. Для детектора ладони использовались лишь  000 изображений данного набора, поскольку этого достаточно для его обучения локализации кисти.

Таким образом, технология состоит из детектора ладоней, определяющего область изображения, на которой изображена рука и модели трекинга руки, предсказывающая ключевые точки на области, полученной детектором.

Основываясь на данной технологии, полученные ключевые точки можно использовать в дальнейших приложениях, например, в распознавании жестов.

3. РАЗРАБОТКА ИНТЕРФЕЙСА ДЛЯ УПРАВЛЕНИЯ ЖЕСТАМИ

В разработку интерфейса для управления жестами входит разработка GUI-приложения сбора данных для обучения нейросети, обучение искусственных нейронных сетей классификации статических и динамических жестов и анализ спроектированного и реализованного метода.

Прежде чем начать разрабатывать эргономичный интерфейс для управления путём распознавания, необходимо обобщить все ранее отмеченные выкладки и сформировать окончательный алгоритм работы человеко-машинного взаимодействия с помощью распознавания жестов.

3.1. Система распознавания жестов

Общая схема алгоритма работы интерфейса изображена на рис. 18.

|  |
| --- |
|  |
| Рис. 18. Общая схема алгоритма работы интерфейса. |

Изображение с кистью, считанное одиночной камерой, поступает на вход Single Shot детектора ладоней. Затем отделённое от всего остального кадра изображение кисти поступает на вход искусственной нейронной сети MediaPipe Hands, которая определяет положение ключевых точек с помощью регрессии. Массив ключевых точек сохраняется в буфер и поступает на вход искусственной нейронной сети классификации статических жестов, которая сопоставляет массив точек с жестом и возвращает результат в виде номера жеста. Если буфер точек, работая по принципу очереди, заполнился то, все данные в нём поступают на вход искусственной нейронной сети классификации динамических жестов, и если эта сеть определила, что массив сопоставляется с каким-либо динамическим жестом, то возвращает номер динамического жеста. На выходе системы формируется два отфильтрованных сигнала с номерами жестов. Фильтрация заключается в пропуске сигналов, не обозначающих какой-либо жест, а являющихся шумом. Рассмотрим подробнее реализацию метода.

3.2. Требования к реализации

Прежде чем выдвинуть требования к реализации интерфейса человеко-машинного взаимодействия необходимо вывести основные требования для эффективной работы систем «человек — техника», следующих из *инженерной психологии*:

1. Наличие высокопроизводительной и надёжной техники;
2. Конструкция техники и организация производственного процесса должны позволять человеку реализовывать все технические возможности;
3. Человек должен быть способным по своим качествам реализовывать все эти возможности, добиваться высокой производительности труда и обеспечивать выполнение производственных операций.

Требования к разрабатываемому человеко-машинному интерфейсу относятся ко второму пункту, то есть должно быть достигнуто согласование человека и техники как элементов единой системы. Под согласованием понимается максимальное приспособление техники к человеку (по параметрам конструкции и технологического процесса), максимальное приспособление человека к технике (по параметрам профессиональной пригодности и профессиональной подготовленности) и рациональное распределение функций между человеком и автоматическими устройствами в системах «человек – техника».

Отсюда следует, что одним из требований к реализации интерфейса человеко-машинного взаимодействия на основе распознавания жестов является *эргономичность*, включающая в себя *гибкость* использования. Здесь под гибкостью понимается лёгкость и быстрота добавления новых жестов обычным пользователем, поскольку в зависимости от машины, сферы применения и количества действий количество жестов может варьироваться, а также изменяться их вид – могут добавляться новые или удаляться/переназначаться старые. Гибкость использования реализуется посредством реализации двух режимов работы: «*обучение*», включающее в себя добавление новых жестов, и «*применение*», идентифицирующее жест и посылающее связанный с этим жестов сигнал.

Таким образом в режиме «обучение» нужно учесть следующее:

* Выбор режима работы перед запуском программы;
* Переключение между записью статического и динамического жеста;
* Непрерывный мониторинг уже записанных жестов и количества записей по ним;
* Вывод текущего записанного жеста;

Для реализации алгоритма будет использоваться язык программирования Python [47]. Выбор обусловлен тем, что Python имеет большое количество развивающихся библиотек для работы с нейронными сетями и простой синтаксис, позволяющий быстро и качественно реализовывать идеи, в том числе с использованием Jupyter Notebook [31].

В листинге 1 приложения А представлены параметры запуска программы. Код программы находится в листингах приложения А.

Спроектированный вид интерфейса в режиме «обучение» изображён на рис. 19. В верхней части выводится количество записей по каждому из жестов, предоставляя возможность выравнивать количество данных по ним. При нажатии на

* клавишу ‘q’ или Escape программа завершает свою работу;
* цифру, то есть ‘0’-‘9’ – начинается запись или записывается жест по данной цифре, ‘c’ – переключаются режимы одиночной и непрерывной записи статического жеста. В первом режиме координаты ключевых точек записываются только при нажатии на соответствующую цифру. Последний же позволяет сохранять координаты ключевых непрерывно, а не только при нажатии на цифру. Данный подход позволяет намного быстрее собирать большое количество данных, непрерывно модифицируя и перемещая руку с жестом;
* клавишу ‘s’ – включается/выключается запись динамического жеста, которая продолжается до нажатия какой-либо другой клавиши. При последовательной записи динамических жестов удобно пользоваться только цифровыми клавишами, поскольку они одновременно останавливают предыдущую запись и начинают новую.

|  |
| --- |
|  |
| Рис. 19. Вид программы в режиме "обучение. |

Таким образом, собрав необходимое количество данных, необходимо на их основе обучить модель нейронной сети распознавать конкретных жест исходя из ключевых точек.

3.3. Подготовка и обучение модели распознавания статических жестов

Подготовка данных для обучения модели заключается в преобразовании координат к необходимому нормализованному виду, как, например, на рис. 20. Исходные значения координат поступают с привязкой к расположению ключевой точки на изображении, то есть в пикселях. Поскольку жест не всегда может находиться в конкретном данном расположении, то необходимо отвязать их от изображения. Для этого все координаты пересчитываются относительно ключевой точки, находящейся на запястье, и нормируются.

|  |
| --- |
|  |
| Рис. 20. Преобразование исходных координат. |

Затем, набор массивов координат, привязанных к конкретным жестам, подаётся на вход MLP со структурой, изображённой на рис. 21.

|  |
| --- |
|  |
| Рис. 21. Структура нейронной сети для распознавания статических жестов. |

Более подробно структуру можно записать следующим образом:

* Входной слой ()
* Дропаут
* Скрытый слой () с функцией активации ReLU:
* Дропаут
* Скрытый слой () с функцией активации ReLU
* Выходной слой () с функцией активации Softmax, где количество жестов.

*Исключение* или *дропаут* – метод регуляризации искусственных нейронных сетей, предназначен для уменьшения переобучения сети за счёт предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения, характеризует исключение определённого процента случайных нейронов на разных итерациях во время обучения нейронной сети. Такой приём значительно увеличивает скорость обучения, качество обучения на тренировочных данных, а также повышает качество предсказаний модели на новых тестовых данных.

В качестве функции оптимизации была выбрана функция Adam. поскольку это один из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Он сочетает в себе идеи среднеквадратичного распространения корня (RMSProp) и оптимизатора импульса. Вместо того чтобы адаптировать скорость обучения параметров на основе среднего первого момента (среднего значения), как в RMSProp, Adam также использует среднее значение вторых моментов градиентов. В частности, алгоритм вычисляет экспоненциальное скользящее среднее градиента и квадратичный градиент.

В качестве примера были выбраны 3 жеста с вытянутым одним, двумя и тремя пальцами соответственно и заранее сгенерированный набор из 3138 точек для этих жестов. Сгенерированный набор необходимо разбить на три выборки: обучающую, валидационную и тестовую. Обучающая выборка – это набор данных, который используется для разработки модели, проверочная выборка – это набор данных, который используется в процессе разработки модели для подбора оптимального набора гиперпараметров, а тестовая выборка – это набор данных, который не используется непосредственно в процессе обучения модели или для подбора гиперпараметров, однако позволяет протестировать модель и является контрольным.

Распределение объёмов выборок по каждому классу представлено в таблице 6.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица 6. | | | | |
| Распределение объёмов выборок по каждому классу | | | | |
| Класс | Размер во всём наборе | Размер выборки | | |
| Для обучения ( от всего) | | Тестовая  (% от всего) |
| Обучающая | Проверочная |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Для разработки и обучения искусственной нейронной сети была выбрана библиотека TensorFlow v2 [61]. По сравнению с похожей библиотекой PyTorch, TensorFlow показывает себя намного производительней, а значит тренировка модели займёт меньшее время.

Инициализация модели искусственной нейросети классификации статиеских жестов, её обучение, тестирование и упаковка представлены в листинге 10 приложения Б. Обучение для набора данных из трёх классов жестов проходило на 97 эпохах (рис. 22). Доля правильных ответов (accuracy) , потери (loss) . Основные метрики по каждому классу представлены в таблице 7.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 22. Изменение метрики accuracy (а) и loss (б) на тренировочных и проверочных (validation) данных во время обучения. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 7. | | | |
| Показатели метрик классификации статических жестов. | | | |
| Класс | Точность (precision) | Полнота (recall) | F-мера (f1-score) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

На матрице ошибок для задачи распознавания статических жестов (рис. 23) показано соотношение истинных значений с ответами нейросети.

|  |
| --- |
|  |
| Рис. 23. Матрица ошибок для задачи распознавания статических жестов. |

Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy – доля правильных ответов алгоритма:

Для оценки качества работы алгоритма на каждом из классов по отдельности вводятся метрики precision (точность) и recall (полнота):

Precision можно интерпретировать как долю объектов, верно определённых классификатором, а recall показывает, какую долю объектов класса из всех объектов этого класса нашёл алгоритм. Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. -мера (в общем случае ):

Переменная в данном случае определяет вес точности в метрике, и при это среднее гармоническое precision и recall. -мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

Проанализировав результаты, можно сделать вывод, что при тестировании нейросеть показала неплохие результаты, а значит её можно применять в задаче классификации статических жестов по ключевым точкам.

3.4. Расширение модели распознавания статических жестов

Расширение модели распознавания статических жестов заключается в обучении модели на большем наборе данных с большим количеством классов. Примером такого набора может быть набор HANDS статических жестов для взаимодействия человека и робота [43]. Этот набор был создан для исследования взаимодействия между человеком и роботом и состоит из пространственно и временно выровненных кадров RGB и глубины. Он содержит 13 видов статических жестов, выполненных как правой, так и левой рукой, таким образом, всего 26 уникальных классов.

Пять испытуемых (2 женщины и 3 мужчины) выполняли жесты, каждый из которых был на разном фоне и в разных условиях освещения. Для каждого жеста было собрано 150 кадров RGB и соответствующие им 150 кадров глубины, всего 2400 кадров RGB и 2400 кадров глубины на человека.

Данные были собраны с помощью камеры Kinect V2, откалиброванной для пространственного выравнивания данных RGB с данными о глубине. Временное выравнивание было выполнено в автономном режиме с использованием MATLAB, выравнивая кадры с максимальным временным расстоянием 66 мс. Примеры жестов изображены на рис. 24.

|  |
| --- |
|  |
| Рис. 24. Примеры фотографий каждого класса из набора данных HANDS. |

На каждом изображении были найдены ключевые точки и сопоставлены с классом жеста. Таким образом, общий набор составил записей по всем классам. Код подготовки данных для обучения нейросети из набора HANDS представлен в листинге 11 приложения Б.

Распределение объёмов выборок по каждому классу представлено в таблице 8.

| Таблица 8. | | | | |
| --- | --- | --- | --- | --- |
| Распределение объёмов выборок HANDS по каждому классу | | | | |
| Класс | Размер во всём наборе | Размер выборки | | |
| Для обучения ( от всего) | | Тестовая  (% от всего) |
| Обучающая | Проверочная |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Тренировка нейросети (рис. 25) на эпохах на этом наборе жестов c оптимизатором Adam, а затем валидация на тестовой выборке из него показала точность и потери , что недостаточно для корректной классификации жестов, поэтому было решено её модифицировать.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 25. Изменение метрики accuracy (а) и loss (б) на тренировочных и валидирующих данных во время обучения нейросети на большой выборке. | |

Улучшенная структура искусственной нейронной сети для классификации статических жестов содержит большее количество скрытых слоёв и нейронов в них и изображена на рис. 26. Все слои, кроме последнего, как и прежде имеют функцию активации ReLU.

|  |
| --- |
|  |
| Рис. 26. Улучшенная архитектура нейросети. |

Код инициализации улучшенной искусственной нейросети классификации статических жестов, её обучения, тестирования и упаковки представлен в листинге 12 приложения Б.

Обучение улучшенной нейросети проходило на эпохах (рис. 27). Проверив работу нейросети на тестовых данных, доля правильных ответов (accuracy) составила , а потери (loss) – . Основные метрики по каждому классу представлены в таблице 9.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 27. Изменение метрики accuracy (а) и loss (б) на тренировочных и валидирующих данных во время обучения улучшенной нейросети. | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Таблица 9. | | | | | | | | |
|  | | Показатели метрик расширенной модели классификации. | | | | | | | | |
| Класс | Точность (precision) | | Полнота (recall) | F-мера (f1-score) |  | Класс | Точность (precision) | Полнота (recall) | F-мера (f1-score) |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |
|  |  | |  |  |  |  |  |  |

Матрица ошибок для задачи распознавания жестов с помощью улучшенной нейросети представлена на рис. 28.

|  |
| --- |
|  |
| Рис. 28. Матрица ошибок для задачи распознавания жестов с помощью улучшенной нейросети. |

Проанализировав результаты, можно сделать вывод, что улучшенная нейросеть успешно обучена и её можно применять в задаче классификации большего количества жестов.

3.5. Подготовка и обучение модели распознавания динамических жестов

По аналогии с подготовкой данных для распознавания статических жестов для обучения модели распознавания динамических жестов с каждым массивом координат точек необходимо проделывать все те же действия, описанные ранее (рис. 29).

|  |
| --- |
|  |
| Рис. 29. Подготовка данных для обучения нейросети распознавания динамических жестов. |

Поскольку динамический жест является последовательностью координат различной длины, то каждую запись жеста необходимо привести к одному общему размеру. В данном случае можно дополнить каждую из записей незначащими нулями справа до максимального размера среди всех записей.

Собранный набор данных для обучения включает в себя 3 класса динамических жестов: «вращение указательным пальцем по часовой стрелке», «вращение указательным пальцем против часовой стрелки» и «какой-то другой жест».

Необходимость последнего обусловлена тем, что при работе система непрерывно собирает распознанные координаты в буфер и так же непрерывно распознаёт динамический жест. Поскольку в текущий момент времени динамического жеста может не быть, но при этом может быть статический, то необходимо отделять «полезные» динамические жесты от побочных. Всего было собрано записей, распределение между классами соответственно и записей. Графическое изображение последовательности координат указательного пальца (точка на рис. 17а) в записях изображено на рисунке 30а, а на рисунке 30б изображены те же координаты, приведённые к относительным запястью (точке ). После сбора и нормализации данных происходит фильтрация сильно длинных последовательностей, чтобы остальные наборы координат дополнялись нулями несущественно.

|  |  |
| --- | --- |
|  |  |
| *а)* | *б)* |
| Рис. 30. Графическое изображение примеров последовательности координат указательного пальца (а) относительно запястья (б)  ( «вращение по часовой стрелке», «против»).  Формат легенды: <номер записи>\_<класс>. | |

Затем, все эти наборы координат подаются на вход искусственной нейронной сети, проиллюстрированной на рис. 31. В структуре нейросети для классификации динамических жестов присутствует LSTM-слой с 50 выходами, слой дропаута 20%, скрытый слой с 100 нейронами и функцией активации ReLU и выходной слой с количеством нейронов, равным количеству динамических жестов (в текущем случае – 3), и функцией активации Softmax. При обучении будет так же использоваться функция оптимизации Adam.

|  |
| --- |
|  |
| Рис. 31. Структура нейронной сети для распознавания динамических жестов. |

Код подготовки данных, инициализации нейросети классификации динамических жестов, её обучения, тестирования и упаковки представлен в листинге 13 приложения Б.

Распределение объёмов выборок по каждому классу представлено в таблице 10.

| Таблица 10. | | | | |
| --- | --- | --- | --- | --- |
| Распределение объёмов выборок HANDS по каждому классу | | | | |
| Класс | Размер во всём наборе | Размер выборки | | |
| Для обучения ( от всего) | | Тестовая  (% от всего) |
| Обучающая | Проверочная |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Обучение сети проходило на эпохах (рис. 32). Доля правильных ответов (accuracy) составила , а потери (loss) – . Основные метрики по каждому классу представлены в таблице 11.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рис. 32. Инициализированный фон (а) и изображение кисти на нём (б). | |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 11. | | | |
| Показатели метрик классификации динамических жестов. | | | |
| Класс | Точность (precision) | Полнота (recall) | F-мера (f1-score) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Матрица ошибок для задачи распознавания динамических жестов изображена на рисунке 33.

|  |
| --- |
|  |
| Рис. 33. Матрица ошибок для задачи распознавания динамических жестов. |

Таким образом, можно сделать вывод, что нейросеть классификации динамических успешно обучена и её можно применять в задаче классификации динамических жестов.

3.6. Оценка работы интерфейса

В качестве практического применения интерфейса и тестирования распознавания жестов было выбрано управление курсором мыши, соотношения между жестами и действием которого распределено следующим образом:

* Жест 1 (показывается только указательный палец, остальные собраны в кулак) – перемещение курсора, где верхняя точка указательного пальца (точка на рис. 17а) отвечает за его координаты на экране;
* Жест 2 (по аналогии с 1 показывается только указательный и средний палец) – нажатие левой кнопки мыши;
* Жест 3 (по аналогии с 1 и 2 показывается указательный, средний и безымянный пальцы) – нажатие правой кнопки мыши;

Интерфейс программы изображён на рис. 34.

|  |
| --- |
|  |
| Рис. 34. Интерфейс программы при управлении курсором мыши. |

В верхней части отображается количество кадров в секунду (FPS, Frames Per Second) и номер распознанного жеста. В центральной части отображается область проекции экрана, то есть левый верхний угол данной области соответствует левому верхнему углу экрана и так далее. Перемещая указательный палец в данной области, перемещается курсор по экрану. Причина, по которой была выбрана именно область, а не всё изображение, состоит в том, что в крайних его частях плохо или вовсе не распознаётся жест, поскольку рука не полностью помещается в кадр.

При первом тестировании такой системы был найден недостаток такого подхода – из-за небольшой тряски руки курсор постоянно находится в движении и трудно корректно его позиционировать. В качестве решения может быть использован метод, заключающийся в игнорировании малых перемещений точки в пределах некоторого числа пикселей, в частности – пяти.

При последующих тестированиях алгоритм работал достаточно точно и быстро, в большинстве случаев верно определяя ключевые точки и классифицируя их, выдавая заданный жест. Неточности возникали при больших наклонах руки – это связано прежде всего с неудачными данными для обучения. При увеличении количества различных ситуаций и количества данных алгоритм работал еще точнее.

В качестве основного недостатка данной системы можно выделить неустойчивость при резких передвижениях руки в кадре. Это вызвано тем, что алгоритм предсказывает следующее положение руки в малой окрестности, а при резком перемещении срабатывает детектор, которому необходимо немного больше времени для работы распознавания.

ЗАКЛЮЧЕНИЕ

В результате данной работы были рассмотрены основные теоретические аспекты решения задачи распознавания жестов, подробно описаны существующие подходы и проведено их сравнение. Наиболее эффективным и прогрессивным на текущий момент среди существующих был выделен сенсор, работающий бесконтактно в радиочастотах. Но, поскольку данный тип недостаточно распространён и зрел, то в качестве датчика была выбрана одиночная камера, считывающая изображение жеста и являющаяся неплохой альтернативой.

Были сделан выводы и рекомендации по использованию методов идентификации жестов в соответствии с типами сенсоров. Также было отмечено, что в системах с одной камерой могут быть реализованы методы визуальных признаков для более быстрой обработки изображений или алгоритмы обучения для обеспечения надежной работы.

Среди подходов отслеживания жеста был выделен подход отслеживания путём обнаружения, показавший лучшие среди остальных результаты в тестировании.

Был сделан вывод, что классификация жестов с помощью методов глубокого обучения хоть и требует большое количество обучающих данных и значительные вычислительные мощности, но превосходит остальные методы по качеству работы.

Также в работе были подробно изучены теоретические аспекты актуального способа детектирования объектов на основе искусственных нейронных сетей – Single Shot детектора. В результате сравнения с аналогами был сделан вывод, что на текущий момент этот детектор наилучшим образом подходит для решения требуемой задачи с использованием средств глубокого обучения.

После этого были описаны особенности работы современной библиотеки MediaPipe Hands, использующей Single Shot детектор и регрессию для определения 21 ключевой точки на руке. На основе хороших результатов работы этой библиотеки, полученных в результате тестирования, ей были делегированы задачи детектирования и извлечения конфигурации руки.

Далее был разработан метод обучения и распознавания статических и динамических жестов для управления курсором мыши. Для этого:

* была разработана программа с графическим интерфейсом для быстрого сбора данных;
* были собраны два набора данных: первый – состоящий из 3000 записей ключевых точек для трёх динамических статических жестов, и второй – 549 записей для трёх классов динамических жестов;
* была спроектирована и обучена искусственная нейронная сеть, классифицирующая *статические* жесты со средней точностью по классам равной 96%;
* была спроектирована и обучена искусственная нейронная сеть с рекуррентным слоем LSTM, классифицирующая *динамические* жесты со средней точностью по классам равной 99%;
* была разработана программа управления курсором мыши для тестирования метода, в результате использования которой исправлены незначительные недостатки метода и выявлены особо крупные.

По итогу работы были выявлены незначительные недостатки разработанного метода, заключающиеся в необходимости изменения архитектуры нейронной сети при увеличении классов жестов и в ожидаемой нестабильности работы из-за использования одиночной камеры. Последний недостаток легко устраняется использованием изображений с нескольких камер.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Нагапетян В.Э., Хачумов В.М. Автоматическое преобразование жестов русской ручной азбуки в текстовый вид // Искусственный интеллект и принятие решений. 2013. с. 59-66.
2. Adib F. et al. 3D tracking via body radio reflections // 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14). – 2014. – С. 317-329.
3. Adib F. et al. Capturing the human figure through a wall // ACM Transactions on Graphics (TOG). – 2015. – Т. 34. – №. 6. – С. 1-13.
4. Adib F., Katabi D. See through walls with WiFi! // Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. – 2013. – С. 75-86.
5. Allen B., Curless B., Popović Z. Articulated body deformation from range scan data // ACM Transactions on Graphics (TOG). – 2002. – Т. 21. – №. 3. – С. 612-619.
6. Arici T. et al. Robust gesture recognition using feature pre-processing and weighted dynamic time warping // Multimedia Tools and Applications. – 2014. – Т. 72. – №. 3. – С. 3045-3062.
7. Babenko B., Yang M. H., Belongie S. Visual tracking with online multiple instance learning // 2009 IEEE Conference on computer vision and Pattern Recognition. – IEEE, 2009. – С. 983-990.
8. Bauer A., Wollherr D., Buss M. Human–robot collaboration: a survey // International Journal of Humanoid Robotics. – 2008. – Т. 5. – №. 01. – С. 47-66.
9. Bay H., Tuytelaars T., Gool L. V. Surf: Speeded up robust features // European conference on computer vision. – Springer, Berlin, Heidelberg, 2006. – С. 404-417.
10. Belongie S., Malik J., Puzicha J. Shape matching and object recognition using shape contexts // IEEE transactions on pattern analysis and machine intelligence. – 2002. – Т. 24. – №. 4. – С. 509-522.
11. Breazeal C. et al. Humanoid robots as cooperative partners for people // Int. Journal of Humanoid Robots. – 2004. – Т. 1. – №. 2. – С. 1-34.
12. Celebi S. et al. Gesture recognition using skeleton data with weighted dynamic time warping // VISAPP (1). – 2013. – С. 620-625.
13. Cohen P. R., Levesque H. J. Persistence, intention, and commitment // Reasoning about actions and plans. – 1990. – С. 297-340.
14. Cohen P. R., Levesque H. J. Teamwork // Nous. – 1991. – Т. 25. – №. 4. – С. 487-512.
15. Comaniciu D., Ramesh V., Meer P. Real-time tracking of non-rigid objects using mean shift, in: Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, IEEE, 2000, С. 142–149.
16. Dix A., Finlay J., Abowd G.D., Beale R. Human-Computer Interaction. - Third Edition, Pearson Education Limited: 2004. – С. 857
17. Elmezain M. et al. A hidden markov model-based continuous gesture recognition system for hand motion trajectory // 2008 19th international conference on pattern recognition. – IEEE, 2008. – С. 1-4.
18. Erhan D. et al. Scalable object detection using deep neural networks // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2014. – С. 2147-2154.
19. Felzenszwalb P., McAllester D., Ramanan D. A discriminatively trained, multiscale, deformable part model // 2008 IEEE conference on computer vision and pattern recognition. – IEEE, 2008. – С. 1-8.
20. Girshick R. et al. Rich feature hierarchies for accurate object detection and semantic segmentation // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2014. – С. 580-587.
21. Girshick R. Fast R-CNN. 2015. ICCV.
22. Google, Project Soli, in, Google, 2015. URL: <https://www.google.com/atap/project-soli/> (дата обращения: 23.04.2022)
23. Green S. A. et al. Human-robot collaboration: A literature review and augmented reality approach in design // International journal of advanced robotic systems. – 2008. – Т. 5. – №. 1. – С. 1.
24. Hariharan B. et al. Hypercolumns for object segmentation and fine-grained localization // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – С. 447-456.
25. Haykin S. Kalman filtering and neural networks. – John Wiley & Sons, 2004. – Т. 47.
26. He K. et al. Spatial pyramid pooling in deep convolutional networks for visual recognition // IEEE transactions on pattern analysis and machine intelligence. – 2015. – Т. 37. – №. 9. – С. 1904-1916.
27. Hearst M. A. et al. Support vector machines // IEEE Intelligent Systems and their applications. – 1998. – Т. 13. – №. 4. – С. 18-28.
28. Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780.
29. Howard A. G. Some improvements on deep convolutional neural network based image classification. 2013.
30. Howe N., Leventon M., Freeman W. Bayesian reconstruction of 3d human motion from single-camera video // Advances in neural information processing systems. – 1999. – Т. 12.
31. Jupyter. URL: <https://jupyter.org/> (дата обращения 10.05.2022)
32. Kadous M. W. Auslan sign recognition using computers and gloves // Deaf Studies Research Symposium. – 1998.
33. Kalman R. E. A new approach to linear filtering and prediction problems, Journal of Fluids Engineering, 82, 1960, С. 35–45.
34. Katsuki Y., Yamakawa Y., Ishikawa M. High-speed human/robot hand interaction system // Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction extended abstracts. – 2015. – С. 117-118.
35. Krüger J., Lien T. K., Verl A. Cooperation of human and machines in assembly lines // CIRP annals. – 2009. – Т. 58. – №. 2. – С. 628-646.
36. Kwon J., Lee K. M., Park F. C. Visual tracking via geometric particle filtering on the affine group with optimal importance functions, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, С. 991–998.
37. Liu W. et al. Ssd: Single shot multibox detector // European conference on computer vision. – Springer, Cham, 2016. – С. 21-37.
38. Liu W., Rabinovich A., Berg A.C. ParseNet: Looking wider to see better. 2016. ICLR.
39. Long J., Shelhamer E., Darrell T. Fully convolutional networks for semantic segmentation //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – С. 3431-3440.
40. Lowe D. G. Object recognition from local scale-invariant features // Proceedings of the seventh IEEE international conference on computer vision. – IEEE, 1999. – Т. 2. – С. 1150-1157.
41. Matsumoto Y., Zelinsky A. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement // Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580). – IEEE, 2000. – С. 499-504.
42. Mitra S., Acharya T. Gesture recognition: A survey // IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). – 2007. – Т. 37. – №. 3. – С. 311-324.
43. Nuzzi C. et al. HANDS: an RGB-D dataset of static hand-gestures for human-robot interaction // Data in Brief. – 2021. – Т. 35. – С. 106791.
44. Oikonomidis I., Kyriazis N., Argyros A. A. Efficient model-based 3D tracking of hand articulations using Kinect // BmVC. – 2011. – Т. 1. – №. 2. – С. 3.
45. Parasuraman R., Sheridan T. B., Wickens C. D. A model for types and levels of human interaction with automation // IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans. – 2000. – Т. 30. – №. 3. – С. 286-297.
46. Peterson L. E. K-nearest neighbor // Scholarpedia. – 2009. – Т. 4. – №. 2. – С. 1883.
47. Python.org. URL: <https://www.python.org>/ (дата обращения 10.05.2022)
48. Redmon J. et al. You only look once: Unified, real-time object detection //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 779-788.
49. Ren, S., He, K., Girshick, R., Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. NIPS. 2015.
50. Rublee E. et al. ORB: An efficient alternative to SIFT or SURF // 2011 International conference on computer vision. – IEEE, 2011. – С. 2564-2571.
51. Sanna A. et al. A Kinect-based natural interface for quadrotor control // Entertainment Computing. – 2013. – Т. 4. – №. 3. – С. 179-186.
52. Schapire R. E. The boosting approach to machine learning: An overview // Nonlinear estimation and classification. – 2003. – С. 149-171.
53. Schapire R. E., Freund Y. A decision-theoretic generalization of on-line learning and an application to boosting // Journal of computer and system sciences. – 1997. – Т. 55. – №. 1. – С. 119-139.
54. Schölkopf B., Smola A. Support Vector Machines, Encyclopedia of Biostatistics, 1998.
55. Sermanet P. et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2014. ICLR.
56. Smeulders A. W. M. et al. Visual tracking: An experimental survey // IEEE transactions on pattern analysis and machine intelligence. – 2013. – Т. 36. – №. 7. – С. 1442-1468.
57. Smith J. et al. Electric field sensing for graphical interfaces // IEEE Computer Graphics and Applications. – 1998. – Т. 18. – №. 3. – С. 54-60.
58. Starner T. E. Visual Recognition of American Sign Language Using Hidden Markov Models. – Massachusetts Inst Of Tech Cambridge Dept Of Brain And Cognitive Sciences, 1995.
59. Starner T., Weaver J., Pentland A. Real-time american sign language recognition using desk and wearable computer based video // IEEE Transactions on pattern analysis and machine intelligence. – 1998. – Т. 20. – №. 12. – С. 1371-1375.
60. Suarez J., Murphy R. R. Hand gesture recognition with depth images: A review //2012 IEEE RO-MAN: the 21st IEEE international symposium on robot and human interactive communication. – IEEE, 2012. – С. 411-417.
61. TensorFlow. URL: <https://tensorflow.org/> (дата обращения 14.05.2022)
62. Thurau C., Hlavác V. Pose primitive based human action recognition in videos or still images // 2008 IEEE conference on computer vision and pattern recognition. – IEEE, 2008. – С. 1-8.
63. Uijlings J. R. R. et al. Selective search for object recognition // International journal of computer vision. – 2013. – Т. 104. – №. 2. – С. 154-171.
64. Vygotsky L. S., Cole M. Mind in society: Development of higher psychological processes. – Harvard university press, 1978.
65. Wachs J. P. et al. Vision-based hand-gesture applications // Communications of the ACM. – 2011. – Т. 54. – №. 2. – С. 60-71.
66. Wan E., Van Der Merwe R. The unscented Kalman filter for nonlinear estimation, in: Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000, IEEE, 2000, С. 153–158.
67. Wang R. Y., Popović J. Real-time hand-tracking with a color glove // ACM transactions on graphics (TOG). – 2009. – Т. 28. – №. 3. – С. 1-8.
68. Wang J., Gao W. A fast sign word recognition method for Chinese sign language // International Conference on Multimodal Interfaces. – Springer, Berlin, Heidelberg, 2000. – С. 599-606.
69. Zhang F. et al. MediaPipe Hands: On-device Real-time Hand Tracking. 2020. CVPR.
70. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A. Object detectors emerge in deep scene cnns. 2015. ICLR.

ПРИЛОЖЕНИЕ А

Код GUI приложения

Листинг 1: Параметры запуска программы

1. >> python app.py --help
2. usage: app.py [-h] [--learning\_mode] [--device DEVICE] [--width WIDTH] [--height HEIGHT]
3. [--use\_static\_image\_mode] [--min\_detection\_confidence MIN\_DETECTION\_CONFIDENCE]
4. [--min\_tracking\_confidence MIN\_TRACKING\_CONFIDENCE] [--static\_mode]
5. [--image\_for\_static\_mode IMAGE\_FOR\_STATIC\_MODE]
6. optional arguments:
7. -h, --help Вывод этого сообщения
8. --learning\_mode, -lm Режим обучения
9. --device DEVICE. Номер устройства
10. --width WIDTH Ширина изображения с камеры
11. --height HEIGHT Высота изображения с камеры
12. --use\_static\_image\_mode Использовать режим со статическим изображением
13. --min\_detection\_confidence MIN\_DETECTION\_CONFIDENCE
14. MIN\_DETECTION\_CONFIDENCE = [0 ... 1]
15. Минимальная уверенность в обнаружении
16. --min\_tracking\_confidence MIN\_TRACKING\_CONFIDENCE
17. MIN\_TRACKING\_CONFIDENCE = [0 ... 1]
18. Минимальная уверенность в отслеживании

Листинг 2: Запуск приложения main.py

1. import argparse
2. import logging
3. import cv2
4. import mediapipe as mp
5. import numpy
6. from classes.AppRunInterface import AppRunInterface
7. from classes.GestureClassification import GestureClassification
8. from classes.GesturesTraining import GesturesTraining
9. from classes.utils.DiplomaVision import DiplomaVision
10. from classes.utils.FakeCamera import FakeCamera
11. from utils.utils import detect\_hand\_landmarks, write\_image\_to\_file
12. def get\_args():
13. parser = argparse.ArgumentParser()
14. parser.add\_argument('--learning\_mode', '-lm', action='store\_true', default=False,
15. help='Mode to learn new gestures')
16. parser.add\_argument('--device', type=int, default=0)
17. parser.add\_argument('--width', help='screen capture width', type=int, default=960)
18. parser.add\_argument('--height', help='screen capture height', type=int, default=540)
19. parser.add\_argument('--use\_static\_image\_mode', action='store\_true', default=False)
20. parser.add\_argument('--min\_detection\_confidence',
21. help='MIN\_DETECTION\_CONFIDENCE = [0 ... 1]',
22. type=float,
23. default=0.7)
24. parser.add\_argument('--min\_tracking\_confidence',
25. help='MIN\_TRACKING\_CONFIDENCE = [0 ... 1]',
26. type=float,
27. default=0.5)
28. return parser.parse\_args()
29. class App:
30. def \_\_init\_\_(self):
31. logging.basicConfig(level=logging.DEBUG)
32. self.logger = logging.getLogger()
33. self.args = get\_args()
34. if self.args.static\_mode:
35. self.args.use\_static\_image\_mode = True
36. self.hands = self.get\_hands()
37. self.camera = FakeCamera(self.args.image\_for\_static\_mode)
38. self.run\_class: AppRunInterface = DiplomaVision(self.hands, self.args.image\_for\_static\_mode)
39. return
40. self.hands = self.get\_hands()
41. self.camera = self.get\_camera()
42. if self.args.learning\_mode:
43. self.logger.info("Learning mode is activated!")
44. self.run\_class: AppRunInterface = GesturesTraining(self.hands,
45. self.camera)
46. else:
47. self.logger.info("Classification mode is activated!")
48. self.run\_class: AppRunInterface = GestureClassification(self.hands,
49. self.camera)
50. def get\_hands(self):
51. self.hands = mp.solutions.hands.Hands(
52. static\_image\_mode=self.args.use\_static\_image\_mode,
53. max\_num\_hands=1,
54. min\_tracking\_confidence=self.args.min\_tracking\_confidence,
55. min\_detection\_confidence=self.args.min\_detection\_confidence
56. )
57. return self.hands
58. def get\_camera(self):
59. self.camera = cv2.VideoCapture(self.args.device)
60. self.camera.set(cv2.CAP\_PROP\_FRAME\_WIDTH, self.args.width)
61. self.camera.set(cv2.CAP\_PROP\_FRAME\_HEIGHT, self.args.height)
62. return self.camera
63. def start(self):
64. cv2.namedWindow('Hands Landmarks Detection', cv2.WINDOW\_NORMAL)
65. while self.camera.isOpened():
66. ok, first\_frame = self.camera.read()
67. if not ok:
68. self.logger.error('not opened')
69. continue
70. frame, hand\_landmarks = self.process\_data\_from\_camera(first\_frame.copy(), detect\_landmarks=True)
71. frame = self.run\_class(frame, hand\_landmarks)
72. cv2.imshow('Hands Landmarks Detection', frame)
73. key = cv2.waitKeyEx(1)
74. self.run\_class.parse\_keyboard(key)
75. if key == 27 or key == ord('q'):
76. self.logger.info('STOP application')
77. break
78. elif key == ord('b'):
79. write\_image\_to\_file(first\_frame, dir='saved\_images')
80. self.camera.release()
81. cv2.destroyAllWindows()
82. def process\_data\_from\_camera(self, frame: numpy.ndarray, detect\_landmarks: bool = False) -> (numpy.ndarray,):
83. frame = cv2.flip(frame, 1)
84. hand\_landmarks = None
85. if detect\_landmarks:
86. frame, results = detect\_hand\_landmarks(frame, self.hands)
87. if results and results.multi\_hand\_landmarks:
88. hand\_landmarks = next(iter(results.multi\_hand\_landmarks))
89. return frame, hand\_landmarks
90. if \_\_name\_\_ == '\_\_main\_\_':
91. App().start()

Листинг 3: Интерфейс классов классификации или сбора данных

1. class AppRunInterface:
2. def \_\_call\_\_(self, frame, hand\_landmarks):
3. pass
4. def parse\_keyboard(self, key):
5. pass

Листинг 4: Классификация жестов

1. import logging
2. import autopy
3. import cv2
4. import mediapipe as mp
5. import numpy
6. from classes.AppRunInterface import AppRunInterface
7. from classes.DynamicBuffer import DynamicBuffer
8. from classifier.GestureClassifier import GestureClassifier
9. from utils.CVFpsCalc import CvFpsCalc
10. from utils.utils import landmarks\_to\_plain\_list, relativize
11. SHAKE\_CURSOR\_RANGE = 5
12. FPS\_COLOR = (229, 43, 80)
13. GESTURE\_INFO\_COLOR = (80, 43, 229)
14. def get\_finger\_coords(hand\_landmarks, shape):
15. finger\_tip\_coords = None
16. if hand\_landmarks:
17. image\_height, image\_width, \_ = shape
18. finger\_tip\_coords = (
19. int(hand\_landmarks.landmark[mp.solutions.hands.HandLandmark.INDEX\_FINGER\_TIP].x \* image\_width),
20. int(hand\_landmarks.landmark[mp.solutions.hands.HandLandmark.INDEX\_FINGER\_TIP].y \* image\_height)
21. )
22. return finger\_tip\_coords
23. def distance(point1, point2):
24. return ((point1[0] - point2[0]) \*\* 2 + (point1[0] - point2[0]) \*\* 2) \*\* 0.5
25. def pre\_process\_landmark(landmarks):
26. landmarks\_list = []
27. base\_x = landmarks.landmark[mp.solutions.hands.HandLandmark.WRIST].x
28. base\_y = landmarks.landmark[mp.solutions.hands.HandLandmark.WRIST].y
29. for i, landmark in enumerate(landmarks.landmark):
30. landmarks\_list.append(landmark.x - base\_x)
31. landmarks\_list.append(landmark.y - base\_y)
32. return landmarks\_list
33. class GestureClassification(AppRunInterface):
34. def \_\_init\_\_(self,
35. hands,
36. camera: cv2.VideoCapture,
37. scale\_factor: float = 0.5):
38. self.prev\_screen\_point = None
39. self.finger\_coords = None
40. self.prev\_click = None
41. logging.basicConfig(level=logging.DEBUG)
42. self.logger = logging.getLogger()
43. self.hands = hands
44. self.camera = camera
45. self.scale\_factor = scale\_factor
46. self.cvFpsCalc = CvFpsCalc(buffer\_len=10)
47. self.screen\_width = camera.get(cv2.CAP\_PROP\_FRAME\_WIDTH)
48. self.screen\_height = camera.get(cv2.CAP\_PROP\_FRAME\_HEIGHT)
49. self.shift\_x = int(self.screen\_width \* (1 - scale\_factor) // 2)
50. self.shift\_y = int(self.screen\_height \* (1 - scale\_factor) // 2)
51. self.pad\_width = int(self.screen\_width \* self.scale\_factor)
52. self.pad\_height = int(self.screen\_height \* self.scale\_factor)
53. self.gestures\_classifier = GestureClassifier()
54. self.dynamic\_gestures\_classifier = GestureClassifier(model\_path='data/gestures\_sequence\_classifier.tflite')
55. self.buffer = DynamicBuffer(buffer\_size=self.dynamic\_gestures\_classifier.get\_input\_shape()[1])
56. def \_\_call\_\_(self, frame, hand\_landmarks):
57. fps = self.cvFpsCalc.get()
58. frame = self.draw\_pad(frame)
59. self.buffer.save(landmarks\_to\_plain\_list(hand\_landmarks))
60. self.hand\_landmarks = hand\_landmarks
61. cv2.putText(frame, 'FPS: {}'.format(fps),
62. (10, 20), cv2.FONT\_HERSHEY\_PLAIN, 1.25, FPS\_COLOR, 2)
63. self.finger\_coords = get\_finger\_coords(hand\_landmarks, frame.shape)
64. if self.finger\_coords:
65. frame = cv2.circle(frame, center=self.finger\_coords,
66. radius=15, color=(0, 255, 255), thickness=-1)
67. self.place\_mouse()
68. if hand\_landmarks:
69. landmarks\_to\_classify = pre\_process\_landmark(hand\_landmarks)
70. if self.buffer.is\_full():
71. dynamic\_gesture\_num, ver = self.dynamic\_gestures\_classifier(relativize(self.buffer.get()))
72. if dynamic\_gesture\_num != 2:
73. self.logger.info(f'Dynamic gesture found! Gesture\_num: {dynamic\_gesture\_num} {ver}')
74. gesture\_num, \_ = self.gestures\_classifier(landmarks\_to\_classify)
75. gesture\_num += 1
76. cv2.putText(frame, f'GESTURE\_NUM = {gesture\_num}',
77. (10, 45), cv2.FONT\_HERSHEY\_PLAIN, 1.25, GESTURE\_INFO\_COLOR, 2)
78. self.action(gesture\_num)
79. else:
80. if self.prev\_click == 2:
81. autopy.mouse.toggle(button=autopy.mouse.Button.LEFT, down=False)
82. elif self.prev\_click == 3:
83. autopy.mouse.toggle(button=autopy.mouse.Button.RIGHT, down=False)
84. self.prev\_click = None
85. return frame
86. def action(self, gesture\_num):
87. if gesture\_num != self.prev\_click:
88. if self.prev\_click == 2:
89. autopy.mouse.toggle(button=autopy.mouse.Button.LEFT, down=False)
90. self.logger.debug('Left button toggle up')
91. elif self.prev\_click == 3:
92. autopy.mouse.toggle(button=autopy.mouse.Button.RIGHT, down=False)
93. self.logger.debug('Right button toggle up')
94. if gesture\_num == 2:
95. self.prev\_click = 2
96. autopy.mouse.toggle(button=autopy.mouse.Button.LEFT, down=True)
97. self.logger.debug('Left button toggle down')
98. elif gesture\_num == 3:
99. self.prev\_click = 3
100. autopy.mouse.toggle(button=autopy.mouse.Button.RIGHT, down=True)
101. self.logger.debug('Right button toggle down')
102. self.prev\_click = gesture\_num
103. def parse\_keyboard(self, key):
104. if key == ord(','):
105. self.scale\_factor -= 0.05
106. self.pad\_width = int(self.screen\_width \* self.scale\_factor)
107. self.pad\_height = int(self.screen\_height \* self.scale\_factor)
108. self.logger.info(f'scale\_factor decreased to {self.scale\_factor}')
109. elif key == ord('.'):
110. self.scale\_factor += 0.05
111. self.pad\_width = int(self.screen\_width \* self.scale\_factor)
112. self.pad\_height = int(self.screen\_height \* self.scale\_factor)
113. self.logger.info(f'scale\_factor increased to {self.scale\_factor}')
114. elif key == ord('j'):
115. self.shift\_x -= 10
116. self.logger.info(f'shift\_x decreased to {self.shift\_x}')
117. elif key == ord(';'):
118. self.shift\_x += 10
119. self.logger.info(f'shift\_x increased to {self.shift\_x}')
120. elif key == ord('k'):
121. self.shift\_y -= 10
122. self.logger.info(f'shift\_y decreased to {self.shift\_y}')
123. elif key == ord('l'):
124. self.shift\_y += 10
125. self.logger.info(f'shift\_y increased to {self.shift\_y}')
126. def draw\_pad(self, frame: numpy.ndarray):
127. image\_height, image\_width, \_ = frame.shape
128. return cv2.rectangle(frame, (self.shift\_x, self.shift\_y),
129. (self.shift\_x + self.pad\_width, self.shift\_y + self.pad\_height),
130. color=(255, 0, 255))
131. def place\_mouse(self):
132. pad\_coords = (self.finger\_coords[0] - self.shift\_x, self.finger\_coords[1] - self.shift\_y)
133. screen\_point = self.convert\_pad\_point\_to\_screen\_point(pad\_coords)
134. if self.prev\_screen\_point and distance(self.prev\_screen\_point, screen\_point) > SHAKE\_CURSOR\_RANGE:
135. try:
136. autopy.mouse.move(\*screen\_point)
137. except ValueError:
138. pass
139. self.prev\_screen\_point = screen\_point
140. def convert\_pad\_point\_to\_screen\_point(self, pad\_point: tuple) -> tuple:
141. scale\_x = self.screen\_width / self.pad\_width
142. scale\_y = self.screen\_height / self.pad\_height
143. return pad\_point[0] \* scale\_x, pad\_point[1] \* scale\_y

Листинг 5: Тренировка жестов (генерация данных)

1. import logging
2. from enum import Enum
3. from typing import Dict, List
4. import cv2
5. import numpy as np
6. from classes.AppRunInterface import AppRunInterface
7. from utils.utils import landmarks\_to\_plain\_list
8. DIGIT\_KEYS = (ord('1'), ord('2'), ord('3'),
9. ord('4'), ord('5'), ord('6'),
10. ord('7'), ord('8'), ord('9'),
11. ord('0'))
12. ALL\_GESTURES\_INFO\_COLOR = (229, 43, 80)
13. CURRENT\_GESTURE\_INFO\_COLOR = (80, 43, 229)
14. HELP\_INFO\_COLOR = (74, 148, 68)
15. def hex2rgb(hex\_str: str) -> tuple:
16. """
17. Converts HEX format to RGB
18. :param hex\_str:
19. :return: tuple(r, g, b)
20. """
21. hex\_str = hex\_str.lstrip('#')
22. return tuple(int(hex\_str[i:i + 2], 16) for i in (0, 2, 4))
23. def save\_gesture\_to\_csv(fname, gesture\_num, hand\_landmarks):
24. with open(fname, 'a+') as f:
25. coords\_str = ",".join(map(str, landmarks\_to\_plain\_list(hand\_landmarks)))
26. if coords\_str:
27. f.write(f"{gesture\_num},{coords\_str}\n")
28. def save\_gesture\_sequence\_to\_csv(fname, gesture\_num, gesture\_sequence):
29. with open(fname, 'a+') as f:
30. coords\_str = ";".join(map(str, gesture\_sequence))
31. if coords\_str:
32. f.write(f"{gesture\_num} | {coords\_str}\n")
33. def load\_gestures\_from\_csv(where\_to\_load: dict, fname, delim=','):
34. try:
35. with open(fname, 'r') as f:
36. for line in f.readlines():
37. gesture\_num = line.split(delim, 1)[0]
38. if gesture\_num.isdigit():
39. gesture\_num = int(gesture\_num)
40. if gesture\_num in where\_to\_load.keys():
41. where\_to\_load[gesture\_num] += 1
42. else:
43. where\_to\_load[gesture\_num] = 1
44. except FileNotFoundError:
45. logging.getLogger().warning(f'{fname} not found, skip loading gestures.')
46. class GesturesTraining(AppRunInterface):
47. class Mode(Enum):
48. CONTINUOUS = 1,
49. SINGLE = 2,
50. SEQUENTIAL = 3
51. def \_\_init\_\_(self,
52. hands,
53. camera,
54. filename: str = 'data/gestures.csv',
55. filename\_seq: str = 'data/gestures\_seq.csv',
56. mode: Mode = Mode.SINGLE):
57. logging.basicConfig(level=logging.DEBUG)
58. self.logger = logging.getLogger()
59. self.hands = hands
60. self.camera = camera
61. self.mode = mode
62. self.gesture\_to\_save = None
63. self.filename = filename
64. self.filename\_seq = filename\_seq
65. self.hand\_landmarks = None
66. self.saved\_gestures\_dict: Dict = None
67. self.gesture\_sequence: List[List[float]] = []
68. self.load\_gestures()
69. def \_\_call\_\_(self, frame, hand\_landmarks):
70. self.hand\_landmarks = hand\_landmarks
71. self.put\_text(frame)
72. if self.mode is not GesturesTraining.Mode.SINGLE:
73. self.save\_gesture()
74. return frame
75. def put\_text(self, frame: np.ndarray):
76. frame\_height = frame.shape[0]
77. frame\_width = frame.shape[1]
78. cv2.rectangle(frame, (0, 0), (frame\_width, 50), (200, 200, 200), thickness=-1)
79. gestures\_str = self.create\_gestures\_string()
80. if len(gestures\_str) == 0:
81. cv2.putText(frame, 'No data available', (10, 20),
82. cv2.FONT\_HERSHEY\_PLAIN, 1.25, ALL\_GESTURES\_INFO\_COLOR, 2)
83. else:
84. cv2.putText(frame, f'{gestures\_str}', (10, 20),
85. cv2.FONT\_HERSHEY\_PLAIN, 1.25, ALL\_GESTURES\_INFO\_COLOR, 2)
86. cv2.putText(frame, f"Gesture = '{self.gesture\_to\_save}' {self.mode}",
87. (10, 45), cv2.FONT\_HERSHEY\_PLAIN, 1.25, CURRENT\_GESTURE\_INFO\_COLOR, 2)
88. cv2.rectangle(frame, (0, frame\_height - 70), (frame\_width, frame\_height), (200, 200, 200), thickness=-1)
89. cv2.putText(frame, "To enable sequential gesture saving press 's', and then any digit key for gesture.",
90. (10, frame\_height - 50),
91. cv2.FONT\_HERSHEY\_PLAIN, 1.25, HELP\_INFO\_COLOR, 2)
92. cv2.putText(frame, "To save current gesture press any digit key.",
93. (10, frame\_height - 30),
94. cv2.FONT\_HERSHEY\_PLAIN, 1.25, HELP\_INFO\_COLOR, 2)
95. cv2.putText(frame, "To continuous gesture saving press 'c', and then any digit key for gesture (for fast "
96. "generating static gestures)",
97. (10, frame\_height - 10),
98. cv2.FONT\_HERSHEY\_PLAIN, 1.25, HELP\_INFO\_COLOR, 2)
99. def parse\_keyboard(self, key):
100. if key in DIGIT\_KEYS:
101. if self.mode is GesturesTraining.Mode.SEQUENTIAL and self.gesture\_to\_save is not None:
102. self.action\_when\_change\_sequential\_mode()
103. self.gesture\_to\_save = key - ord('0')
104. if self.mode is GesturesTraining.Mode.CONTINUOUS:
105. self.logger.info(f"Begin to save '{self.gesture\_to\_save}' gesture. Mode = {self.mode}")
106. elif self.mode is GesturesTraining.Mode.SEQUENTIAL:
107. self.logger.info(f"Begin to save '{self.gesture\_to\_save}' gesture. Mode = {self.mode}")
108. else:
109. self.save\_gesture()
110. self.logger.info(f"Saved '{self.gesture\_to\_save}' gesture")
111. elif key == ord('c'):
112. self.gesture\_to\_save = None
113. self.action\_when\_change\_sequential\_mode()
114. self.mode = GesturesTraining.Mode.CONTINUOUS \
115. if self.mode is not GesturesTraining.Mode.CONTINUOUS else GesturesTraining.Mode.SINGLE
116. off\_on = 'on' if self.mode is GesturesTraining.Mode.CONTINUOUS else 'off'
117. self.logger.info(f'Turned {off\_on} flag for continuous gesture save')
118. elif key == ord('s'):
119. self.gesture\_to\_save = None
120. self.action\_when\_change\_sequential\_mode()
121. self.mode = GesturesTraining.Mode.SEQUENTIAL \
122. if self.mode is not GesturesTraining.Mode.SEQUENTIAL else GesturesTraining.Mode.SINGLE
123. off\_on = 'on' if self.mode is GesturesTraining.Mode.SEQUENTIAL else 'off'
124. self.logger.info(f'Turned {off\_on} flag for sequential gesture save')
125. def save\_gesture(self):
126. if self.gesture\_to\_save is not None and self.hand\_landmarks:
127. if self.mode is GesturesTraining.Mode.CONTINUOUS:
128. save\_gesture\_to\_csv(self.filename,
129. self.gesture\_to\_save,
130. self.hand\_landmarks)
131. if self.gesture\_to\_save not in self.saved\_gestures\_dict.keys():
132. self.saved\_gestures\_dict[self.gesture\_to\_save] = 1
133. else:
134. self.saved\_gestures\_dict[self.gesture\_to\_save] += 1
135. elif self.mode is GesturesTraining.Mode.SEQUENTIAL:
136. self.gesture\_sequence.append(landmarks\_to\_plain\_list(self.hand\_landmarks))
137. def load\_gestures(self):
138. self.saved\_gestures\_dict = {}
139. load\_gestures\_from\_csv(self.saved\_gestures\_dict, self.filename)
140. load\_gestures\_from\_csv(self.saved\_gestures\_dict, self.filename\_seq, delim=';')
141. self.logger.info('Gestures dict loaded')
142. self.logger.debug(f'{self.saved\_gestures\_dict}')
143. def create\_gestures\_string(self):
144. s = ""
145. for k, v in self.saved\_gestures\_dict.items():
146. s += f"['{k}': {v}] "
147. return s
148. def action\_when\_change\_sequential\_mode(self):
149. """
150. Перед сменой режима и номера жеста сохраняем сохранённую последовательность,
151. если она вообще есть
152. """
153. if self.mode is GesturesTraining.Mode.SEQUENTIAL and self.gesture\_to\_save:
154. gesture\_num = str(self.gesture\_to\_save) + 's'
155. save\_gesture\_sequence\_to\_csv(self.filename\_seq,
156. gesture\_num,
157. self.gesture\_sequence)
158. if gesture\_num not in self.saved\_gestures\_dict.keys():
159. self.saved\_gestures\_dict[gesture\_num] = 1
160. else:
161. self.saved\_gestures\_dict[gesture\_num] += 1
162. self.gesture\_sequence = []

Листинг 6: Динамический буфер последовательности точек

1. import logging
2. class DynamicBuffer:
3. def \_\_init\_\_(self, buf\_size=0):
4. self.\_\_buffer\_size = buf\_size
5. self.\_\_current\_size = 0
6. self.\_\_buffer = []
7. logging.getLogger().debug(f'Created buffer with size={buf\_size}')
8. def save(self, data):
9. if data:
10. self.\_\_buffer.append(data)
11. self.\_\_current\_size += 1
12. if self.\_\_current\_size > self.\_\_buffer\_size:
13. self.\_\_buffer.pop(0)
14. self.\_\_current\_size = self.\_\_buffer\_size
15. def is\_full(self):
16. return self.\_\_current\_size == self.\_\_buffer\_size
17. def clear(self):
18. self.\_\_buffer = []
19. self.\_\_current\_size = 0
20. def get(self):
21. return self.\_\_buffer

Листинг 7: Классификатор жестов

1. import numpy as np
2. import tensorflow as tf
3. class GestureClassifier(object):
4. def \_\_init\_\_(self,
5. model\_path='data/gestures\_classifier\_upgraded.tflite',
6. num\_threads=1):
7. self.interpreter = tf.lite.Interpreter(model\_path=model\_path,
8. num\_threads=num\_threads)
9. self.interpreter.allocate\_tensors()
10. self.input\_details = self.interpreter.get\_input\_details()
11. self.output\_details = self.interpreter.get\_output\_details()
12. def \_\_call\_\_(self, landmark\_list):
13. input\_details\_tensor\_index = self.input\_details[0]['index']
14. self.interpreter.set\_tensor(
15. input\_details\_tensor\_index,
16. np.array([landmark\_list], dtype=np.float32)
17. )
18. self.interpreter.invoke()
19. output\_details\_tensor\_index = self.output\_details[0]['index']
20. result = self.interpreter.get\_tensor(output\_details\_tensor\_index)
21. result\_index = np.argmax(np.squeeze(result))
22. return result\_index, np.squeeze(result)
23. def get\_input\_shape(self):
24. return self.input\_details[0]['shape']

Листинг 8: Счётчик кадров в секунду

1. from collections import deque
2. import cv2
3. class CvFpsCalc(object):
4. def \_\_init\_\_(self, buffer\_len=1):
5. self.\_start\_tick = cv2.getTickCount()
6. self.\_freq = 1000.0 / cv2.getTickFrequency()
7. self.\_difftimes = deque(maxlen=buffer\_len)
8. def get(self):
9. current\_tick = cv2.getTickCount()
10. different\_time = (current\_tick - self.\_start\_tick) \* self.\_freq
11. self.\_start\_tick = current\_tick
12. self.\_difftimes.append(different\_time)
13. fps = 1000.0 / (sum(self.\_difftimes) / len(self.\_difftimes))
14. fps\_rounded = round(fps, 2)
15. return fps\_rounded

Листинг 9: Утилиты

1. import os.path
2. import time
3. from typing import NamedTuple
4. import cv2
5. import mediapipe as mp
6. import numpy as np
7. mp\_drawing = mp.solutions.drawing\_utils
8. mp\_hands = mp.solutions.hands
9. def detect\_hand\_landmarks(image: np.array, hands: mp.solutions.hands.Hands) -> (np.array, NamedTuple):
10. """
11. Detect and draw hand landmarks in image
12. :rtype: tuple
13. :param image: Frame
14. :param hands: object of mediapipe.solutions.hands.Hands
15. :return: tuple of (output\_image, results), where \*output\_image\* is
16. a copy of image with landmarks and \*results\* -- result of hands.process(image)
17. (image converted to RGB)
18. """
19. mp\_drawing\_styles = mp.solutions.drawing\_styles
20. image\_height, image\_width, \_ = image.shape
21. output\_image = image.copy()
22. imgRGB = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
23. results = hands.process(imgRGB)
24. if results.multi\_hand\_landmarks:
25. for hand\_landmarks in results.multi\_hand\_landmarks:
26. mp\_drawing.draw\_landmarks(image=output\_image,
27. landmark\_list=hand\_landmarks,
28. connections=mp\_hands.HAND\_CONNECTIONS,
29. landmark\_drawing\_spec=mp\_drawing\_styles.get\_default\_hand\_landmarks\_style(),
30. connection\_drawing\_spec=mp\_drawing\_styles.get\_default\_hand\_connections\_style())
31. return output\_image, results
32. def write\_image\_to\_file(image, filename=None, dir=None):
33. """
34. Writes image to file, if directory is not created, creates directory
35. :param filename: if None, uses image{time.strftime("%Y-%m-%d\_%H:%M:%S")}.jpg'
36. :param dir:
37. :return: None
38. """
39. if filename is None:
40. filename = f'image{time.strftime("%Y-%m-%d\_%H:%M:%S")}.jpg'
41. curr\_dir = os.path.dirname(\_\_package\_\_)
42. if dir is not None:
43. dir = os.path.join(curr\_dir, dir)
44. if not os.path.exists(dir):
45. os.mkdir(dir)
46. filename = os.path.join(dir, filename)
47. cv2.imwrite(filename, image)
48. def convert\_pad\_point\_to\_screen\_point(pad\_point: tuple, pad\_width: int, pad\_height: int,
49. screen\_width: int, screen\_height: int) -> tuple:
50. scale\_x = screen\_width / pad\_width
51. scale\_y = screen\_height / pad\_height
52. return pad\_point[0] \* scale\_x, pad\_point[1] \* scale\_y
53. def landmarks\_to\_plain\_list(hand\_landmarks):
54. coordinates = []
55. if hand\_landmarks:
56. for i in range(0, 21):
57. landmark = hand\_landmarks.landmark[i]
58. coordinates.extend([landmark.x, landmark.y])
59. return coordinates
60. def relativize(arr: np.array, to: int = 0, rng: int = 2):
61. """
62. Relativize every {rng} elems of array by subtracting it from arr[to:range]
63. """
64. if type(arr) is not np.ndarray:
65. arr = np.array(arr, dtype=np.float)
66. for i in range(rng):
67. arr[to + i::rng] -= arr[to + i]
68. return arr

ПРИЛОЖЕНИЕ Б

Модели обучения нейросетей

Листинг 10: Инициализация модели нейросети классификации статических жестов, её обучение, тестирование и упаковка

1. from datetime import datetime
2. import numpy as np
3. import tensorflow as tf
4. from sklearn.model\_selection import train\_test\_split
5. RANDOM\_SEED = 42
6. dataset = 'gestures.csv'
7. model\_save\_path = 'gestures\_classifier.hdf5'
8. NUM\_CLASSES = 3
9. x\_dataset = np.loadtxt(dataset, delimiter=',', dtype='float32', usecols=list(range(1, (21 \* 2) + 1)), skiprows=1)
10. for x\_row in x\_dataset:
11. first\_x = x\_row[0]
12. first\_y = x\_row[1]
13. x\_row[0::2] -= first\_x
14. x\_row[1::2] -= first\_y
15. y\_dataset = np.loadtxt(dataset, delimiter=',', dtype='int32', usecols=0, skiprows=1)
16. y\_dataset -= 1
17. x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_dataset, y\_dataset, train\_size=0.75, random\_state=RANDOM\_SEED)
18. model = tf.keras.models.Sequential([
19. tf.keras.layers.Input((21 \* 2,)),
20. tf.keras.layers.Dropout(0.2),
21. tf.keras.layers.Dense(20, activation='relu'),
22. tf.keras.layers.Dropout(0.4),
23. tf.keras.layers.Dense(10, activation='relu'),
24. tf.keras.layers.Dense(NUM\_CLASSES, activation='softmax')
25. ])
26. cp\_callback = tf.keras.callbacks.ModelCheckpoint(
27. model\_save\_path, verbose=1, save\_weights\_only=False
28. )
29. es\_callback = tf.keras.callbacks.EarlyStopping(patience=20, verbose=1)
30. model.compile(
31. optimizer='adam',
32. loss='sparse\_categorical\_crossentropy',
33. metrics=['accuracy']
34. )
35. begin = datetime.now()
36. model.fit(
37. x\_train,
38. y\_train,
39. epochs=1000,
40. batch\_size=128,
41. validation\_data=(x\_test, y\_test),
42. callbacks=[cp\_callback, es\_callback]
43. )
44. end = datetime.now()
45. print(f'Затрачено времени на обучение: {end-begin}')
46. val\_loss, val\_acc = model.evaluate(x\_test, y\_test, batch\_size=64)
47. import pandas as pd
48. import seaborn as sns
49. import matplotlib.pyplot as plt
50. from sklearn.metrics import confusion\_matrix, classification\_report
51. def print\_confusion\_matrix(y\_true, y\_pred, report=True):
52. labels = sorted(list(set(y\_true)))
53. cmx\_data = confusion\_matrix(y\_true, y\_pred, labels=labels)
54. df\_cmx = pd.DataFrame(cmx\_data, index=labels, columns=labels)
55. fig, ax = plt.subplots(figsize=(7, 6))
56. sns.heatmap(df\_cmx, annot=True, fmt='g', square=False)
57. ax.set\_ylim(len(set(y\_true)), 0)
58. plt.show()
59. if report:
60. print('Classification Report')
61. print(classification\_report(y\_test, y\_pred))
62. Y\_pred = model.predict(x\_test)
63. y\_pred = np.argmax(Y\_pred, axis=1)
64. print\_confusion\_matrix(y\_test, y\_pred)
65. model.save(model\_save\_path, include\_optimizer=False)
66. tflite\_save\_path = 'gestures\_classifier.tflite'
67. converter = tf.lite.TFLiteConverter.from\_keras\_model(model)
68. converter.optimizations = [tf.lite.Optimize.DEFAULT]
69. tflite\_quantized\_model = converter.convert()
70. open(tflite\_save\_path, 'wb').write(tflite\_quantized\_model)

Листинг 11: Преобразование набора изображений HANDS в набор координат и его упаковка

1. import cv2
2. import csv
3. import matplotlib.pyplot as plt
4. from pathlib import Path
5. import mediapipe as mp
6. hands = mp.solutions.hands.Hands(
7. static\_image\_mode=True,
8. max\_num\_hands=1,
9. min\_tracking\_confidence=0.5,
10. min\_detection\_confidence=0.7
11. )
12. def parse\_arr(str\_arr: str):
13. str\_arr = str\_arr[1:-1]
14. nums = list(map(int, map(float, str\_arr.split())))
15. return nums
16. def detect\_hand\_landmarks(image, hands):
17. """
18. Detect and draw hand landmarks in image
19. :rtype: tuple
20. :param image: Frame
21. :param hands: object of mediapipe.solutions.hands.Hands
22. :return: tuple of (output\_image, results), where \*output\_image\* is
23. a copy of image with landmarks and \*results\* -- result of hands.process(image)
24. (image converted to RGB)
25. """
26. mp\_hands = mp.solutions.hands
27. mp\_drawing = mp.solutions.drawing\_utils
28. mp\_drawing\_styles = mp.solutions.drawing\_styles
29. output\_image = image.copy()
30. imgRGB = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)
31. results = hands.process(imgRGB)
32. if results.multi\_hand\_landmarks:
33. for hand\_landmarks in results.multi\_hand\_landmarks:
34. mp\_drawing.draw\_landmarks(image=output\_image,
35. landmark\_list=hand\_landmarks,
36. connections=mp\_hands.HAND\_CONNECTIONS,
37. landmark\_drawing\_spec=mp\_drawing.DrawingSpec(color=(245,117,66), thickness=-1, circle\_radius=1)
38. )
39. return output\_image, results
40. def landmarks\_to\_list(landmarks):
41. landmarks\_list = []
42. for landmark in landmarks.landmark:
43. landmarks\_list.append(landmark.x)
44. landmarks\_list.append(landmark.y)
45. return landmarks\_list
46. def dump(fname: str, obj):
47. import pickle
48. with open(fname, "wb") as outfile:
49. pickle.dump(obj, outfile)
50. landmarks\_dict = {}
51. # Преобразование имён файлов к UNIX-подобным
52. files = {
53. 'Subject1/Subject1.txt': lambda s: s.replace('..\\Subject2\\Color\\', 'Subject1/Subject1/'),
54. 'Subject2/Subject2.txt': lambda s: s.replace('..\\Subject3\\Color\\', 'Subject2/Subject2/'),
55. 'Subject3/Subject3.txt': lambda s: s.replace('..\\Subject4\\Color\\', 'Subject3/Subject3/'),
56. 'Subject4/Subject4.txt': lambda s: s.replace('..\\Subject5\\Color\\', 'Subject4/Subject4/'),
57. 'Subject5/Subject5.txt':lambda s: s.replace('..\\Subject6\\Color\\', 'Subject5/Subject5/')
58. }
59. for fname, func in files.items():
60. with open(fname, newline='') as csvfile:
61. reader = csv.DictReader(csvfile)
62. for row in reader:
63. path = Path(func(row['rgb'])).resolve().absolute().as\_posix()
64. print(path)
65. img = cv2.imread(path)
66. if img is not None:
67. for k, v in row.items():
68. arr = parse\_arr(v)
69. if all(arr) != 0:
70. hand = img[arr[1]:arr[1]+arr[3], arr[0]:arr[0]+arr[2]]
71. out, results = detect\_hand\_landmarks(hand, hands)
72. if results and results.multi\_hand\_landmarks:
73. hand\_landmarks = next(iter(results.multi\_hand\_landmarks))
74. lmarks = landmarks\_to\_list(hand\_landmarks)
75. out = {'path': path, 'bbox': [arr[1], arr[1]+arr[3], arr[0], arr[0]+arr[2]], 'landmarks': lmarks, 'hand': hand}
76. if k not in landmarks\_dict.keys():
77. landmarks\_dict[k] = [out]
78. else:
79. landmarks\_dict[k].append(out)
80. # Оценка количества координат по каждому классу
81. for key in landmarks\_dict.keys():
82. print(f"{key}: {len(landmarks\_dict[key])}")
83. dump('dataset\_dump.pickle', landmarks\_dict)

Листинг 12: Инициализация улучшенной модели нейросети классификации статических жестов, её обучение, тестирование и упаковка

1. from datetime import datetime
2. import numpy as np
3. import tensorflow as tf
4. from sklearn.model\_selection import train\_test\_split
5. import tensorflow\_datasets as tfds
6. import matplotlib.pyplot as plt
7. RANDOM\_SEED = 42
8. model\_save\_path = 'gestures\_classifier\_upgraded.hdf5'
9. PATH\_TO\_DUMP = 'dataset\_dump.pickle'
10. with open(PATH\_TO\_DUMP, "rb") as infile:
11. import pickle
12. landmarks\_dump = pickle.load(infile)
13. y\_classes = list(landmarks\_dump.keys())
14. NUM\_CLASSES = len(y\_classes)
15. x, y\_strs = [], []
16. for cls in y\_classes:
17. for elem in landmarks\_dump[cls]:
18. x.append(elem['landmarks'])
19. y\_strs.append(cls)
20. class\_label = tfds.features.ClassLabel(names=y\_classes)
21. y = list(map(lambda x: class\_label.str2int(x), y\_strs))
22. x = np.array(x)
23. y = np.array(y)
24. for x\_row in x:
25. first\_x = x\_row[0]
26. first\_y = x\_row[1]
27. x\_row[0::2] -= first\_x
28. x\_row[1::2] -= first\_y
29. x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, train\_size=0.75)
30. model = tf.keras.models.Sequential([
31. tf.keras.layers.Input((21 \* 2,)),
32. tf.keras.layers.Dense(60, activation='relu'),
33. tf.keras.layers.Dense(20, activation='relu'),
34. tf.keras.layers.Dropout(0.2),
35. tf.keras.layers.Dense(40, activation='relu'),
36. tf.keras.layers.Dropout(0.4),
37. tf.keras.layers.Dense(60, activation='relu'),
38. tf.keras.layers.Dropout(0.2),
39. tf.keras.layers.Dense(40, activation='relu'),
40. tf.keras.layers.Dense(20, activation='relu'),
41. tf.keras.layers.Dense(NUM\_CLASSES, activation='softmax')
42. ])
43. cp\_callback = tf.keras.callbacks.ModelCheckpoint(
44. model\_save\_path, verbose=1, save\_weights\_only=False
45. )
46. es\_callback = tf.keras.callbacks.EarlyStopping(patience=20, verbose=3)
47. model.compile(
48. optimizer='adam',
49. loss='sparse\_categorical\_crossentropy',
50. metrics=['accuracy']
51. )
52. begin = datetime.now()
53. history = model.fit(
54. x\_train,
55. y\_train,
56. epochs=1000,
57. batch\_size=64,
58. validation\_split=0.1,
59. validation\_steps=3,
60. callbacks=[cp\_callback, es\_callback]
61. )
62. end = datetime.now()
63. print(f'Затрачено времени на обучение: {end-begin}')
64. def plot\_history(history):
65. plt.plot(history.history['accuracy'])
66. plt.plot(history.history['val\_accuracy'])
67. plt.title('Изменение accuracy')
68. plt.ylabel('accuracy')
69. plt.xlabel('Эпоха')
70. plt.legend(['Тренировка', 'Валидация'], loc='best')
71. plt.show()
72. plt.plot(history.history['loss'])
73. plt.plot(history.history['val\_loss'])
74. plt.title('Изменение loss')
75. plt.ylabel('loss')
76. plt.xlabel('Эпоха')
77. plt.legend(['Тренировка', 'Валидация'], loc='best')
78. plt.show()
79. plot\_history(history)
80. val\_loss, val\_acc = model.evaluate(x\_test, y\_test, batch\_size=1)
81. import pandas as pd
82. import seaborn as sns
83. import matplotlib.pyplot as plt
84. from sklearn.metrics import confusion\_matrix, classification\_report
85. def print\_confusion\_matrix(y\_true, y\_pred, report=True):
86. labels = sorted(list(set(y\_true)))
87. cmx\_data = confusion\_matrix(y\_true, y\_pred, labels=labels)
88. df\_cmx = pd.DataFrame(cmx\_data, index=labels, columns=labels)
89. fig, ax = plt.subplots(figsize=(7, 6))
90. sns.heatmap(df\_cmx, annot=True, fmt='g', square=False)
91. ax.set\_ylim(len(set(y\_true)), 0)
92. plt.show()
93. if report:
94. print('Classification Report')
95. print(classification\_report(y\_test, y\_pred))
96. Y\_pred = model.predict(x\_test)
97. y\_pred = np.argmax(Y\_pred, axis=1)
98. print\_confusion\_matrix(y\_test, y\_pred)
99. model.save(model\_save\_path, include\_optimizer=False)
100. tflite\_save\_path = 'gestures\_classifier\_upgraded.tflite'
101. converter = tf.lite.TFLiteConverter.from\_keras\_model(model)
102. converter.optimizations = [tf.lite.Optimize.DEFAULT]
103. tflite\_quantized\_model = converter.convert()
104. open(tflite\_save\_path, 'wb').write(tflite\_quantized\_model)

Листинг 13: Инициализация модели нейросети классификации динамических жестов, её обучение, тестирование и упаковка

1. from datetime import datetime
2. import matplotlib.pyplot as plt
3. import numpy as np
4. import tensorflow as tf
5. from keras.preprocessing import sequence
6. from sklearn.model\_selection import train\_test\_split
7. np.random.seed(42)
8. dataset = gestures\_seq.csv'
9. model\_save\_path = 'gesture\_sequence\_classifier.hdf5'
10. LIMIT\_LENGTH = 30
11. def plot\_history(history):
12. plt.plot(history.history['accuracy'])
13. plt.plot(history.history['val\_accuracy'])
14. plt.title('model accuracy')
15. plt.ylabel('accuracy')
16. plt.xlabel('epoch')
17. plt.legend(['train', 'test'], loc='upper left')
18. plt.show()
19. plt.plot(history.history['loss'])
20. plt.plot(history.history['val\_loss'])
21. plt.title('model loss')
22. plt.ylabel('loss')
23. plt.xlabel('epoch')
24. plt.legend(['train', 'test'], loc='upper left')
25. plt.show()
26. import pandas as pd
27. import seaborn as sns
28. import matplotlib.pyplot as plt
29. from sklearn.metrics import confusion\_matrix, classification\_report
30. def print\_confusion\_matrix(y\_true, y\_pred, report=True):
31. labels = sorted(list(set(y\_true)))
32. cmx\_data = confusion\_matrix(y\_true, y\_pred, labels=labels)
33. df\_cmx = pd.DataFrame(cmx\_data, index=labels, columns=labels)
34. fig, ax = plt.subplots(figsize=(7, 6))
35. sns.heatmap(df\_cmx, annot=True, fmt='g', square=False)
36. ax.set\_ylim(len(set(y\_true)), 0)
37. plt.show()
38. if report:
39. print('Classification Report')
40. print(classification\_report(Y\_test, y\_pred))
41. def plot\_coords(x, labels, point\_num: int, num\_from, delta=45):
42. to = num\_from + delta
43. ranges = [range(a, min((a + 5), len(labels))) for a in range(num\_from, to, 5)]
44. subplots\_nums = [331, 332, 333, 334, 335, 336, 337, 338, 339]
45. tips\_x = []
46. tips\_y = []
47. for a in x:
48. finger\_tip\_coords\_x = list(map(lambda x: x[point\_num \* 2], a))
49. finger\_tip\_coords\_y = list(map(lambda x: x[point\_num \* 2 + 1], a))
50. tips\_x.append(finger\_tip\_coords\_x)
51. tips\_y.append(finger\_tip\_coords\_y)
52. plt.figure(figsize=(12, 12))
53. for rng, subplot\_num in zip(ranges, subplots\_nums):
54. plt.subplot(subplot\_num)
55. for i in rng:
56. if len(tips\_x) > i:
57. plt.plot(tips\_x[i], tips\_y[i])
58. plt.legend([f'{i}\_{labels[i]}' for i in rng], loc='best')
59. plt.show()
60. def get\_length\_str(x):
61. lengths\_dict = {}
62. lengths\_list = list(map(lambda x: x.shape[0], x))
63. for length in lengths\_list:
64. if length in lengths\_dict.keys():
65. lengths\_dict[length] += 1
66. else:
67. lengths\_dict[length] = 1
68. max\_len = max(list(map(lambda x: x.shape[0], x)))
69. lengths\_list = [(k, v) for k, v in lengths\_dict.items()]
70. lengths\_list.sort(key=lambda x: x[0], reverse=True)
71. lengths\_str\_list = [f'{i[0]}: {i[1]}' for i in lengths\_list]
72. str\_out = 'Length: count\n' + "\n".join(lengths\_str\_list)
73. return f'Max\_length: {max\_len}\n{str\_out}'
74. def check\_count\_classes(Y\_test, Y\_train):
75. unique\_train, counts\_train = np.unique(Y\_train, return\_counts=True)
76. unique\_test, counts\_test = np.unique(Y\_test, return\_counts=True)
77. print('Train:', dict(zip(unique\_train, counts\_train)))
78. print('Test:', dict(zip(unique\_test, counts\_test)))
79. def load\_data(fname: str, normalize: bool=True, sep: str = ';'):
80. x = []
81. labels = []
82. with open(fname) as f:
83. lines = f.readlines()
84. for line in lines:
85. spl = line.split(sep)
86. labels.append(int(spl[0][:-1]))
87. filtered\_list = list(filter(lambda x: x and len(x.strip()) != 0, spl[1:]))
88. coords\_seq = np.array(
89. list(
90. map(
91. lambda x: np.array(x),
92. map(eval, filtered\_list)
93. )
94. )
95. )
96. if normalize:
97. for coords in coords\_seq:
98. first\_x = coords[0]
99. first\_y = coords[1]
100. coords[0::2] -= first\_x
101. coords[1::2] -= first\_y
102. x.append(coords\_seq)
103. x = np.array(x, dtype='object')
104. labels = np.array(labels, dtype='int32')
105. return x, labels
106. def plot\_length(x):
107. lengths\_dict = {}
108. lengths\_list = list(map(lambda x: x.shape[0], x))
109. for length in lengths\_list:
110. if length in lengths\_dict.keys():
111. lengths\_dict[length] += 1
112. else:
113. lengths\_dict[length] = 1
114. max\_len = max(list(map(lambda x: x.shape[0], x)))
115. xs = []
116. ys = []
117. for k, v in lengths\_dict.items():
118. xs.append(k)
119. ys.append(v)
120. plt.bar(xs, ys)
122. # загрузим данные
123. x, labels = load\_data(dataset)
124. # нормализуем подписи
125. NUM\_CLASSES = len(set(labels))
126. min\_label = min(labels)
127. labels\_norm = labels.copy()
128. labels\_norm -= min\_label
129. labels\_norm[labels\_norm == 3] = 2
130. # графически отобразим координаты
131. FROM = 0 #@param ["0", "45", "90", "135", "180", "225", "270", "315", "360"] {type:"raw"}
132. point\_num = 4 #@param {type:"slider", min:0, max:21, step:1}
133. plot\_coords(x, labels\_norm, point\_num, FROM)
134. # посмотрим на длины последовательностей
135. plot\_length(x)
136. # удалим последовательности длиннее LIMIT\_LENGTH
137. new\_arr\_x = []
138. new\_arr\_labels\_norm = []
139. indexes = []
140. for i, x1 in enumerate(x):
141. if x1.shape[0] <= LIMIT\_LENGTH:
142. new\_arr\_x.append(x1)
143. new\_arr\_labels.append(labels\_norm[i])
144. else:
145. indexes.append(i + 1)
146. x = np.array(new\_arr\_x, dtype=object)
147. labels\_norm = np.array(new\_arr\_labels\_norm, dtype='int32')
148. plot\_length(x)
149. # доводим все последовательности до LIMIT\_LENGTH нулями
150. max\_len = max(list(map(lambda x: x.shape[0], x)))
151. padded\_x = sequence.pad\_sequences(x, maxlen=max\_len, dtype='float', padding='post')
152. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(padded\_x, labels\_norm, test\_size=0.25, random\_state=42, shuffle=True)
153. # оценим где сколько классов
154. check\_count\_classes(Y\_test, Y\_train)
155. model\_conv = tf.keras.models.Sequential([
156. tf.keras.layers.Input(X\_train[0].shape),
157. tf.keras.layers.LSTM(50),
158. tf.keras.layers.Dropout(0.2),
159. tf.keras.layers.Dense(100, activation='relu'),
160. tf.keras.layers.Dense(NUM\_CLASSES, activation='softmax')
161. ])
162. model\_conv.compile(
163. optimizer='adam',
164. loss='sparse\_categorical\_crossentropy',
165. metrics=['accuracy']
166. )
167. begin = datetime.now()
168. history\_conv = model\_conv.fit(
169. X\_train,
170. Y\_train,
171. epochs=500,
172. batch\_size=32,
173. validation\_data=(X\_test, Y\_test),
174. callbacks=[cp\_callback, es\_callback]
175. )
176. end = datetime.now()
177. plot\_history(history\_conv)
178. print(f'Затрачено времени на обучение: {end-begin}')
179. val\_loss, val\_acc = model\_conv.evaluate(X\_test, Y\_test, batch\_size=128)
180. Y\_pred = model\_conv.predict(X\_test)
181. y\_pred = np.argmax(Y\_pred, axis=1)
182. print\_confusion\_matrix(Y\_test, y\_pred)
183. tf.keras.utils.plot\_model(model\_conv,
184. show\_shapes=True,
185. show\_layer\_names=True,
186. rankdir='TB', # 'TB" or 'LR'
187. dpi=100)
188. tflite\_save\_path = 'gestures\_sequence\_classifier.tflite'
189. converter = tf.lite.TFLiteConverter.from\_keras\_model(model\_conv)
190. converter.optimizations = [tf.lite.Optimize.DEFAULT]
191. converter.target\_spec.supported\_ops = [tf.lite.OpsSet.TFLITE\_BUILTINS, tf.lite.OpsSet.SELECT\_TF\_OPS]
192. converter.\_experimental\_lower\_tensor\_list\_ops = False
193. tflite\_quantized\_model = converter.convert()
194. open(tflite\_save\_path, 'wb').write(tflite\_quantized\_model)