ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ / INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2025.152.19

ИНФОРМАЦИОННАЯ СИСТЕМА НА ОСНОВЕ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ПЕРЕВОДА НЕВЕРБАЛЬНОГО ЯЗЫКА В ЛИНГВИСТИЧЕСКУЮ МОДЕЛЬ

Научная статья

Бреус А.И.[1], Минязев Р.Ш.[2], Нуриев М.Г.[3,] *, Мангушева А.Р.[4], Обухова М.Ю.[5]
[3] ORCID : 0009-0003-0741-1734;
[1, 2, 3] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация
[4] Казанский национальный исследовательский технологический университет, Казань, Российская Федерация
[5] Казанский инновационный университет имени В.Г. Тимирясова (ИЭУП), Казань, Российская Федерация

* Корреспондирующий автор (mgnuriev[at]kai.ru)

Аннотация

Данная статья посвящена разработке информационной системы на основе искусственного интеллекта для перевода невербального языка, в частности русского жестового языка, в лингвистическую модель. Основной задачей исследования является создание системы, способной автоматически распознавать жесты и преобразовывать их в текстовые сообщения. Для этого использовались алгоритмы машинного обучения, компьютерного зрения и современные модели искусственного интеллекта. В процессе разработки была создана и протестирована система на языке Python с использованием инструментов MMAction2 и ONNX Runtime, что позволило добиться высокой точности распознавания жестов и обеспечения работы системы в реальном времени. Разработка направлена на решение проблемы ограниченного доступа слабослышащих к информации и средствам коммуникации, что делает данную работу актуальной для социальной интеграции данной категории населения.

Ключевые слова: жестовый язык, искусственный интеллект, распознавание жестов, компьютерное зрение, Python, MMAction2, ONNX Runtime.

AI-BASED INFORMATION SYSTEM FOR TRANSLATING NON-VERBAL LANGUAGE INTO A LINGUISTIC MODEL

Research article

Breus A.I.[1], Minyazev R.S.[2], Nuriev M.G.[3,] *, Mangusheva A.R.[4], Obukhova M.Y.[5]
[3] ORCID : 0009-0003-0741-1734;
[1, 2, 3] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation
[4] Kazan National Research Technological University, Kazan, Russian Federation
[5] Kazan Innovative University named after V.G.Timiryasov (KIU), Kazan, Russian Federation

* Corresponding author (mgnuriev[at]kai.ru)

Abstract

This article is dedicated to the development of an information system based on artificial intelligence to translate non-verbal language, in particular Russian sign language, into a linguistic model. The main objective of the research is to create a system capable of automatically recognising gestures and converting them into text messages. For this purpose, algorithms of machine learning, computer vision and modern models of artificial intelligence were used. During the development process, the system was created and tested in Python language using MMAction2 and ONNX Runtime tools, which allowed to achieve high accuracy of gesture recognition and ensure real-time operation of the system. The development is aimed at solving the problem of limited access of the hearing impaired to information and means of communication, which makes this work relevant for the social integration of this population.

Keywords: sign language, artificial intelligence, gesture recognition, computer vision, Python, MMAction2, ONNX Runtime.

Введение

Современное общество характеризуется разнообразием людей с различными физиологическими особенностями, которые, несмотря на различия, активно взаимодействуют в социальных и профессиональных сферах. Одной из таких групп являются слабослышащие люди, которые сталкиваются с рядом трудностей, связанных с недостаточным доступом к информации и социальной изоляцией. Согласно данным Всемирной организации здравоохранения, примерно 650 миллионов человек по всему миру страдают от различных форм потери слуха, что составляет около 5% от общего числа населения планеты [1]. В России, по официальным данным, на 2024 год число людей с нарушениями слуха и зрения превышает 19 миллионов, а по неофициальным подсчетам, количество слабослышащих может составлять более 2 миллионов. С учетом этих тенденций ожидается, что число людей с нарушениями слуха будет неуклонно расти, что подчеркивает необходимость разработки новых подходов к решению связанных с этим проблем.

Слабослышащие люди сталкиваются с рядом социальных и коммуникативных проблем, среди которых можно выделить ограниченный доступ к информации, трудности в понимании речи, а также проблемы в общении со слышащими людьми. Эти сложности зачастую становятся препятствием для их полноценной интеграции в общество. В частности, слабослышащие испытывают затруднения в получении образовательных, медицинских и иных услуг из-

за нехватки перевода на жестовый язык или субтитров в реальном времени. В условиях современных технологий некоторые из этих проблем могут быть решены с помощью автоматизированных систем перевода и распознавания жестового языка, что позволит существенно улучшить качество жизни данной категории населения.

Жестовые языки, включая русский жестовый язык (РЖЯ), являются основным средством общения для миллионов людей с нарушениями слуха. Эти языки основываются на жестах, мимике, позах тела и других невербальных компонентах. Однако, несмотря на важность жестового языка как средства коммуникации, его освоение и использование ограничены, поскольку большинство людей не владеют данным языком. Также существует дефицит профессиональных переводчиков жестового языка, что усугубляет проблему. Это подчеркивает необходимость разработки автоматизированных решений, которые помогут преодолеть языковой барьер между слышащими и слабослышащими людьми.

Актуальность темы заключается в необходимости создания технологий для автоматического перевода жестового языка в текст или другие языки в реальном времени. Современные достижения в области искусственного интеллекта (ИИ), машинного обучения и компьютерного зрения открывают новые возможности для решения этой задачи. Автоматический перевод жестового языка является перспективной технологией, которая может значительно улучшить доступность образования, здравоохранения, публичных услуг и других сфер для людей с нарушениями слуха. С развитием технологий в этой области можно ожидать улучшение качества жизни слабослышащих людей и их интеграции в общество.

Целью данной работы является разработка системы на основе искусственного интеллекта для перевода жестового языка в лингвистическую модель. Для достижения этой цели будут рассмотрены существующие методы и технологии распознавания жестового языка, а также разработана информационная система на основе Python с применением алгоритмов машинного обучения и компьютерного зрения, способная эффективно распознавать и переводить жесты РЖЯ в текст.

В рамках исследования поставлены следующие задачи: провести обзор существующих решений к распознаванию и переводу жестового языка, а также рассмотреть особенности РЖЯ; изучить современный инструментарий для распознавания жестов и перевода невербальных языков; оценить эффективность существующих решений и их применимость в контексте перевода жестового языка; разработать информационную систему на языке Python, использующую методы ИИ для распознавания и перевода жестов РЖЯ; оценить эффективность и применимость разработанной системы на основе экспериментальных данных.

Практическая ценность работы заключается в создании программного продукта, который будет способствовать улучшению доступности и качества жизни слабослышащих людей, позволяя им более эффективно взаимодействовать с обществом.

Существующие решения

Рассмотрим существующие решения, способные в режиме реального времени переводить жесты русского жестового языка, и приведем показания метрик для дальнейшего сравнения с разрабатываемой системой.

Команда Vision RnD сделала, не побоюсь этого слова, величайший шаг в развитии распознавания и перевода русского жестового языка – собрала крупнейший набор данных для обучения моделей искусственного интеллекта, направленных на распознавание отдельных жестов [2].

В общей сложности удалось собрать более 200 тысяч видеозаписей с жестами РЖЯ, разделенных на 3 тысячи уникальных классов. К сожалению, в открытом доступе есть лишь малая часть этих данных, но и этого достаточно для обучения моделей искусственного интеллекта, способного распознавать одну тысячу жестов. Именно этот датасет будет использован в качестве обучающей и тестовой выборки в проектируемой нами системе. Процесс сбора и обработки информации будет рассмотрен в следующей главе.

Последней разработкой команды из SberDevices является семейство моделей SignFlow, содержащее две модели – SignFlow-R для распознавания РЖЯ и SignFlow-A для распознавания американского жестового языка [3].

Модель SignFlow-R была встроена в GigaChat – русскоязычную мультимодальную нейросетевую модель, которая умеет отвечать на вопросы, вести диалог, писать код, рисовать картинки по запросу. Таким образом, данная система может распознавать и переводить жесты русского жестового языка и общаться с пользователем (рисунок 1). Стоит отметить, что данное решение производит обсчет данных на удаленных серверах и внутри сервиса GigaChat, т.е. распознавание жестов невозможно использовать отдельно для каких-либо других задач. Кроме того, разработчики не приводят метрики – вероятность правильного предсказания жеста.

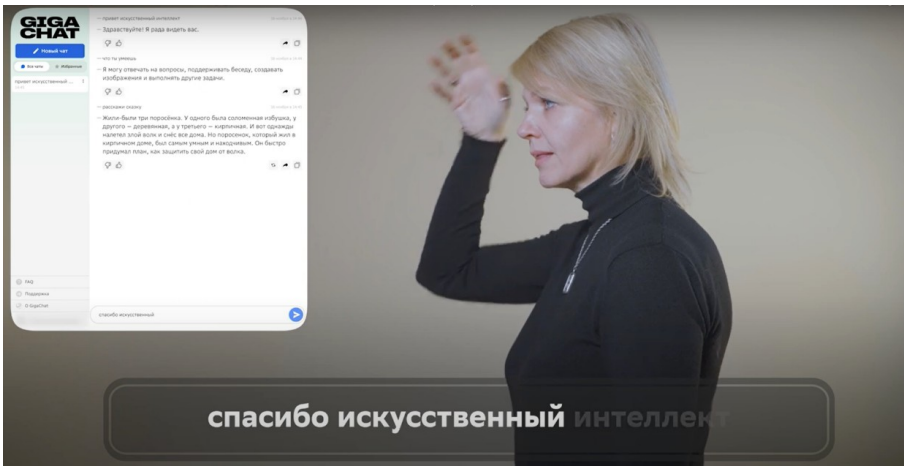

Рисунок 1 - Распознавание жестов РЖЯ с помощью GigaChat
DOI: https://doi.org/10.60797/IRJ.2025.152.19.1

В качестве более старой разработки авторы предлагают обученные модели искусственного интеллекта на архитектурах «ResNed3D-50» и «Swin-large» [4]. В данном случае мы можем оценить их метрики – точности прогнозируемых жестов в зависимости от окна сэмплирования и параметра децимации (рисунок 2). Данные модели требуют больших вычислительных мощностей, по этой причине для распознавания жестов в режиме реального времени необходим графический ускоритель NVIDIA с технологией CUDA.

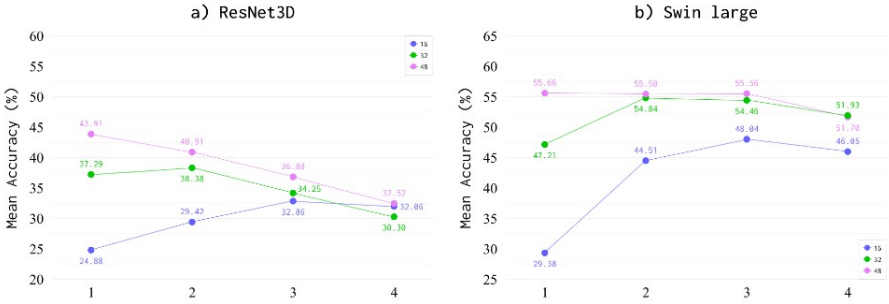

Рисунок 2 - Метрики для «ResNet3D» (*a*) и «Swin-large» (*b*)
DOI: https://doi.org/10.60797/IRJ.2025.152.19.2

Эти модели обучены на датасете «Slovo», который мы будем использовать для обучения собственной модели. По этой причине данные значения подойдут для сравнительной оценки качества распознавания и перевода нашей модели искусственного интеллекта.

Команда разработки из SberAI также представила свой вариант обученной модели искусственного интеллекта [5].

Отличительная особенность данного решения – использование более «легкой» ML-модели – «S3D», которая не требует столь больших вычислительных мощностей, что требуют «ResNet3D» и «Swin-large». По этой причине инференс модели в режиме реального времени не требует графического ускорителя – достаточно производительности центрального процессора (CPU).

По заявлению разработчиков, четырёхъядерного процессора достаточно для того, чтобы совершать 2-2,5 прогноза в секунду (рисунок 3). Этого более чем достаточно для задачи распознавания жестов в режиме реального времени.

Обучение модели производилось на собственной обучающей выборке, составляющей более чем 180 тысяч видеозаписей жестов, 1598 из которых – уникальные.

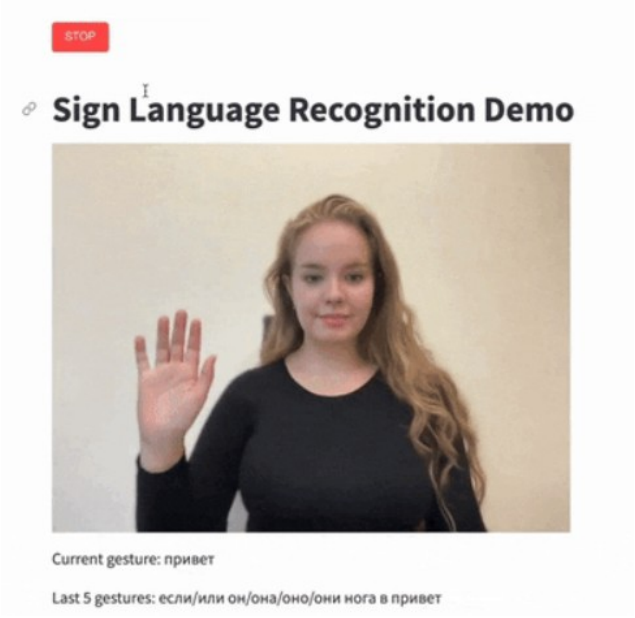

Рисунок 3 - Распознавание жестов РЖЯ с помощью решения команды разработчиков из SberAI
DOI: https://doi.org/10.60797/IRJ.2025.152.19.3

Помимо этого, разработчики выложили в открытый доступ модель, обученную на датасете «Slovo». В зависимости от сэмплирования входных данных, им удалось достичь точности предсказания от 44,22% до 55,86% (таблица 1), что превосходит наилучший результат, полученный при обучении моделей «ResNet3D» и «Swin-large» командой из SberDevices. Данные показатели мы также возьмем для сравнительной оценки качества распознавания и перевода нашей модели искусственного интеллекта.

Таблица 1 - Типы распознавания жестовых языков

DOI: https://doi.org/10.60797/IRJ.2025.152.19.4

Наименование модели	Кол-во входных кадров	Точность предсказания, %
S3D	32	44,22
S3D	48	52,28
S3D	64	55,86

Проектирование системы

Выбор модели искусственного интеллекта играет критическую роль в успешной разработке систем с использованием машинного обучения. Правильный выбор модели может существенно повлиять на результаты проекта по машинному обучению, включая точность прогнозов, скорость обучения, устойчивость к переобучению и масштабируемость системы.

Используемый нами инструментарий MMAction2 [6] предлагает большой выбор моделей, предназначенных для распознавания действий. Для сравнения были выбраны модели, которые хорошо работают с разрешением 224 на 224 точки (именно в это разрешение будет преобразовываться входной поток данных), а также показывают хорошие результаты с обработкой 16 и 32 кадров по 2 и 3 каналам данных. Их сравнение приведено в таблице 2.

Таблица 2 - Сравнение моделей искусственного интеллекта для распознавания действий

DOI: https://doi.org/10.60797/IRJ.2025.152.19.5

Наименование модели	Стратегия сэмплирования кадров	Разрешение	Скелет	top1/acc	top5/acc	FLOPS
MViTv2	16x4x1	224x224	MViTv2-S	81,1	94,7	64G
MViTv2	32x3x1	224x224	MViTv2-B	82,6	95,8	225G
C2D	16x4x1	224x224	ResNet50	74,97	91,91	39G
I3D	32x2x1	224x224	ResNet50	73,47	91,27	43.5G
VideoSwin	32x2x1	224x224	Swin-T	78,90	93,77	88G
VideoSwin	32x2x1	224x224	Swin-S	80,54	94,46	166G

Наименование модели	Стратегия сэмплирования кадров	Разрешение	Скелет	top1/acc	top5/acc	FLOPS
VideoSwin	32x2x1	224x224	Swin-B	80,57	94,49	282G

Исходя из представленных данных, для наших целей лучше всего подойдет модель искусственного интеллекта «MViTv2» с архитектурой «small», поскольку она находится на втором месте по точности предсказания класса и не сильно отстает от первого места, и при этом требует в разы меньше вычислительных мощностей, по сравнению с моделью «MViTv2» с архитектурой «base» [7].

Высокое качество обучающей и тестовой выборки критически важно для успешного обучения и получения высоких метрик модели искусственного интеллекта. По этой причине было принято решение не пытаться собрать свою выборку, а использовать уже готовую.

На данный момент в открытом доступе доступен лишь один датасет жестов русского жестового языка, собранный командой Vision RnD из SberDevices. Скачать его можно с репозитория GitHub «Slovo» [8].

Согласно данным команды SberDevices, им удалось собрать более 200 тысяч видео, которые разделены на 3 тысячи уникальных классов (т.е. 3 тысячи жестов). К сожалению, для скачивания доступны не все данные, а лишь 20400 видеозаписей, из которых 20000 означают какой-либо жест, а 400 – отсутствие жеста. Всего эти видеоданные содержат информацию о 1000 жестов, таким образом на каждый жест приходится 20 видео [5].

Датасет разбит на 2 выборки – обучающую и тренировочную. В тренировочную выборку входит 15300 видеозаписей, в обучающую – 5100.

Для сбора и разметки жестов русского жестового языка были использованы две компании: Yande.Toloka, которая записывала жесты на видео, и ABC Elementary, которая проводила сбор, валидацию и разметку жестов. Данный подход позволил повысить качество конечных данных, поскольку использование двух компаний создает две непересекающиеся аудитории носителей русского жестового языка.

Таким образом, полный цикл сбора обучающей и тестовой выборки выглядит следующим образом:

1) выявление группы людей, владеющих РЖЯ;

2) получение необработанных видеоданных;

3) проверка качества полученных видеоданных, удаление дубликатов и видео с плохим качеством (низкая частота кадров и/или разрешение видео);

4) выявление участка видео, в котором показывается жест;

5) финальная подготовка данных.

Схематически цикл сбора данных показан на рисунке 4.

Далее мы подробно рассмотрим все эти шаги.

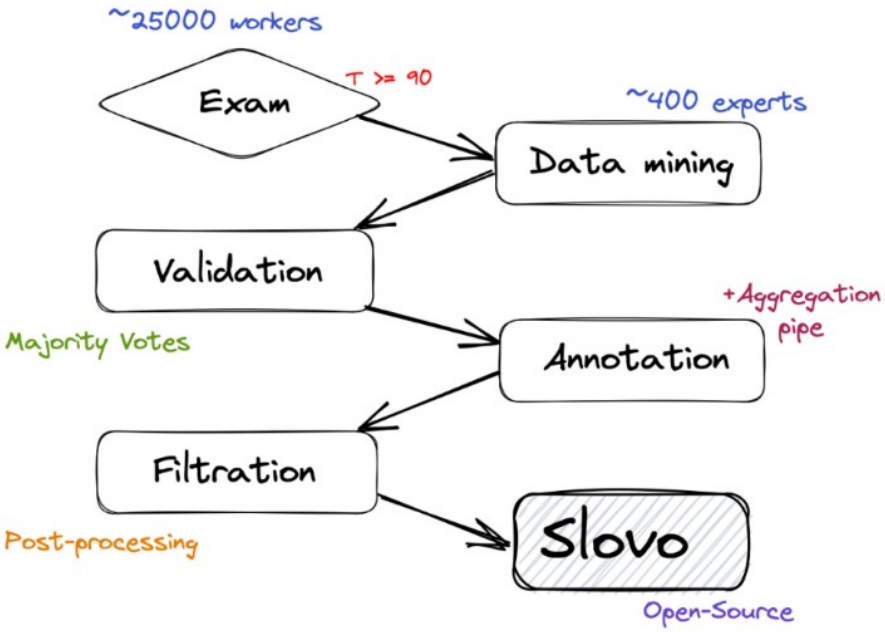

Рисунок 4 - Полный цикл сбора датасета

DOI: https://doi.org/10.60797/IRJ.2025.152.19.6

Экзамен – достаточно очевидная стадия сбора качественных данных. Данный этап позволяет отобрать из большой группы пользователей русского жестового языка людей, обладающих высоким уровнем владения языка.

Для сбора данных была создана система, где пользователям предоставлялись 10-15 видеозаписей, на которых было необходимо классифицировать показанный жест (рисунок 5). Интерес к этому проявило более 40 тысяч человек, из

которых полностью прошли экзамен 25 тысяч пользователей. Высокую оценку по результатам экзамена (более 90% правильных ответов) получили около 400 человек. Распределение результатов экзамена представлено на рисунке 6.

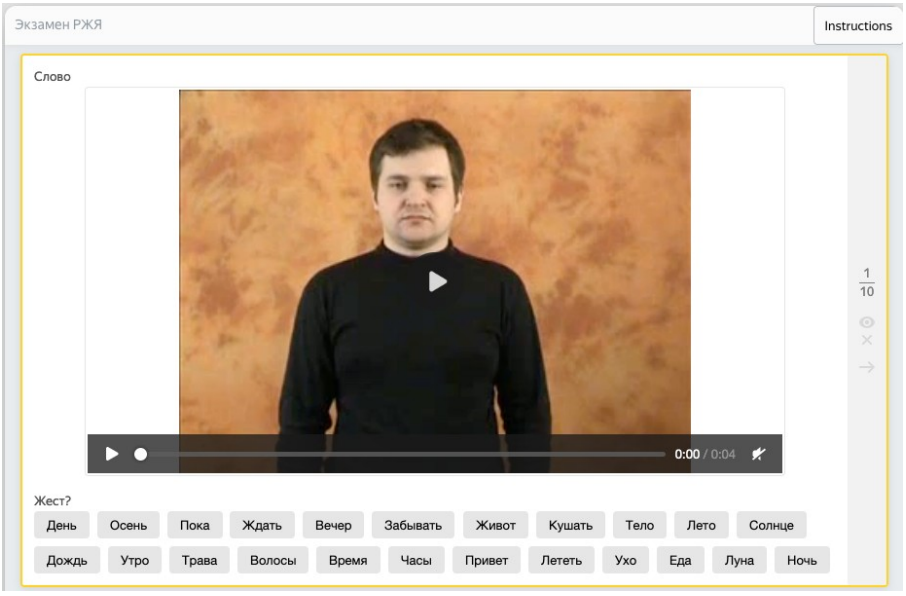

Рисунок 5 - Интерфейс экзаменационной системы
DOI: https://doi.org/10.60797/IRJ.2025.152.19.7

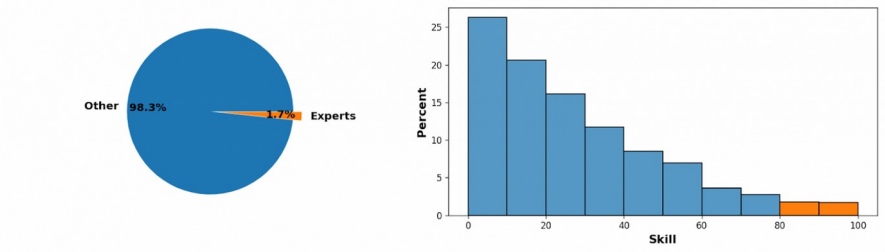

Рисунок 6 - Распределение результатов экзамена
DOI: https://doi.org/10.60797/IRJ.2025.152.19.8

Было принято решение привлечь к разметке видеозаписей тех экзаменуемых, что получили высокую оценку, а к записи видеоданных тех, кто получил хорошую оценку (от 80% до 90% правильных ответов).

В конечном итоге в данном этапе приняло участие около двухсот человек.

На этом этапе сбора данных экзаменуемые, получившие хорошую оценку по результатам экзамена, записывают видео с тем или иным жестом. Сбор данных также интуитивно понятен и прост – пользователю дается слово и пример жеста, который это слово означает. От пользователя требуется записать жест по аналогии и загрузить полученные данные на площадку по сбору данных.

Валидация – представляет собой проверку и отсеивание некачественных данных. Видеозаписи, полученные от пользователей, показываются отдельной группе людей, которые оценивают качество записи. Критерии оценки:

- камера стоит устойчиво, не трясется и не дрожит;
- жест выполнен верно;
- в процессе выполнения жеста руки не выходят за границы кадра.

Для повышения качества проверки происходил контроль за ответами пользователей, недобросовестных участников блокировали или временно отстраняли от выполнения задания. Помимо этого, видеоролики с низким разрешением (менее 320 точек по наименьшей стороне), низкой частотой развертки (менее 15 кадров в секунду) и слишком малой длительностью (менее секунды) удалялись и не подвергались проверке.

На этапе разметки пользователи определяли место начало и конца показываемого жеста. Для сравнения также давался шаблон правильного выполнения жеста. Записанные видео обрезаются по границам усредненной области, полученной от разных пользователей, и из этих данных формируется финальный датасет.

Заключительным этапом формирования выборки является фильтрация. Данный этап состоит из следующих этапов:

- убираются видео, неподходящие по параметрам (высота, ширина, кол-во кадров в секунду, длина);
- убираются сложные и составные жесты;
- слишком медленные видео ускоряются, а слишком быстрый – замедляются;

- данные разбиваются на обучающую и тестовую выборку.

Процесс сбора данных для обучения модели искусственного интеллекта завершен. Датасет в своем конечном виде представляет собой 2 папки «train» и «test», а также аннотационный файл, содержащий информацию о каждой видеозаписи. Первые строки аннотационного файла «annotations.csv» имеют вид, представленный в таблице 3.

Таблица 3 - Содержание аннотационного файла «annotations.csv»

DOI: https://doi.org/10.60797/IRJ.2025.152.19.9

	attachment_id	text	user_id	height	width	length	train	begin	end
0	44e8d2a0-7e01-450b-90b0-beb7400d2c1e	Ё	185bd3a81d9d618518d10abebf0d17a8	640	360	156.0	True	36	112
1	df5b08f0-41d1-4572-889c-8b893e71069b	А	185bd3a81d9d618518d10abebf0d17a8	640	360	150.0	True	36	76
2	17f53df4-c467-4aff-9f48-20687b63d49a	Р	185bd3a81d9d618518d10abebf0d17a8	640	360	133.0	True	40	97

В таблице 3: attachment_id – название видеофайла; user_id – уникальный идентификатор пользователя, записавшего видео; width – ширина видео; height – высота видео; length – длина видео; text – обозначение жеста на русском жестовом языке; train – флаг обучающей или тестовой выборки; begin – начало жеста на оригинальном видео; end – конец жеста на оригинальном видео.

Поскольку обученную модель искусственного интеллекта будет необходимо выполнять в режиме реального времени, то немаловажным является время, затраченное на обработку входных параметров моделью искусственного интеллекта и получения результата на выходе. Помимо этого, важно, чтобы обученная модель могла выполняться на множестве различных платформ. Для этих целей идеально подходит ONNX Runtime [9].

ONNX Runtime – это кроссплатформенный движок для выполнения моделей искусственного, которые представлены в формате Open Neural Network Exchange (ONNX). Он предоставляет высокую производительность и оптимизирован для различных аппаратных платформ. ONNX Runtime позволяет эффективно выполнять модели искусственного интеллекта в различных сценариях, включая развертывание на мобильных устройствах, встраивание в приложения и облачные вычисления.

ONNX Runtime предлагает следующие преимущества:

- высокая производительность: он использует оптимизации для ускорения выполнения моделей искусственного интеллекта на различных аппаратных платформах, что обеспечивает быструю и эффективную работу моделей;
- кроссплатформенность: ONNX Runtime поддерживает различные операционные системы и аппаратные платформы, включая Windows, Linux, macOS, а также различные типы устройств, такие как CPU, GPU и специализированные ускорители моделей искусственного интеллекта TPU (Tensor Processing Units);
- гибкость: он предоставляет возможность выполнения моделей искусственного интеллекта, созданных с использованием различных фреймворков, таких как TensorFlow, PyTorch, Keras и др., что делает его удобным инструментом для интеграции различных моделей;
- оптимизации для ресурсов: ONNX Runtime может оптимизировать использование ресурсов, таких как память и процессорное время, что особенно важно для мобильных устройств и встроенных систем;
- простота использования: он предоставляет простой API для выполнения моделей, что делает его доступным для широкого круга разработчиков и специалистов по искусственному интеллекту.

Реализация проекта и тестирование разработанной системы

Для развертывания и тестирования системы была использована конфигурация персонального компьютера, приведенного в таблице 4. Данные характеристики не являются обязательными или оптимальными для использования системы распознавания и перевода жестов русского жестового языка, единственным необходимым критерием является наличие видеоадаптера компании NVIDIA, поддерживающего технологию NVIDIA CUDA [10]. В противном случае обработка данных будет осуществляться посредством мощностей центрального процессора (CPU), что может отрицательно сказаться на производительности развернутой системы.

Таблица 4 - Технические сведения о системе, на которой производилось развертывание и тестирование системы распознавания и перевода жестов РЖЯ

DOI: https://doi.org/10.60797/IRJ.2025.152.19.10

Параметр	Значение
Процессор	AMD Ryzen 5 5600x (6 core/12 threads)
Оперативная память	32 ГБ
Дисковое пространство	1024 ГБ NVME SSD
Видеоадаптер	NVIDIA GeForce RTX 3070 Ti
Средство видеосвязи (веб-камера)	Papalook PA552 PRO
Операционная система	Windows 10 Pro 22H2

Для реализации системы для перевода невербального языка в лингвистическую модель необходимо установить используемые в исполняемых файлах библиотеки и модули, а также инструменты.

Поскольку система разрабатывается на языке программирования Python [11], первоочередно нужно установить Python версии 3.9. В нижней части страницы выбираем и скачиванием установочный файл, соответствующий используемой операционной системе (в нашем случае «Windows x86-64 executable installer»).

После установки необходимо проверить установленную версию Python. Для этого вводим в командную строку cmd следующую команду:

py --version

Если мы получаем сообщение о том, что «py» не является внутренней или внешней командой, исполняемой программой или пакетным файлом», то необходимо вручную указать путь к исполняемому файлу python.exe. Для этого нажимаем комбинацию клавиш «Win + R», вводим «sysdm.cpl» и жмем «ОК» (рисунок 7).

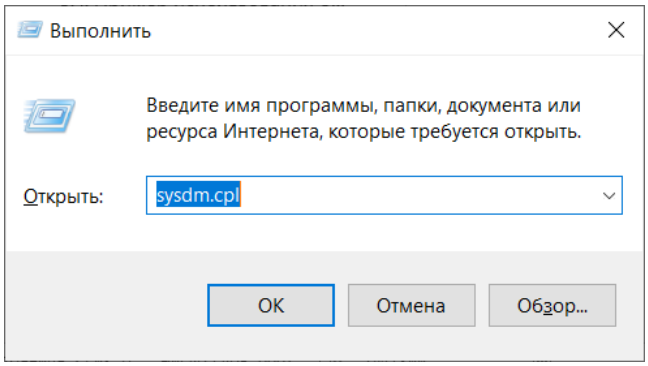

Рисунок 7 - Вызов утилиты «Выполнить» в ОС Windows 10

DOI: https://doi.org/10.60797/IRJ.2025.152.19.11

В открывшемся окне переходим во вкладку «Дополнительно» и открываем «Переменные среды». Далее выбираем строку с переменной «Path», нажимаем «изменить» (рисунок 8). В открывшемся окне создаем переменную среды и указываем путь установки Python 3.9. После этого снова повторить проверку установленной версии Python через командную строку cmd.

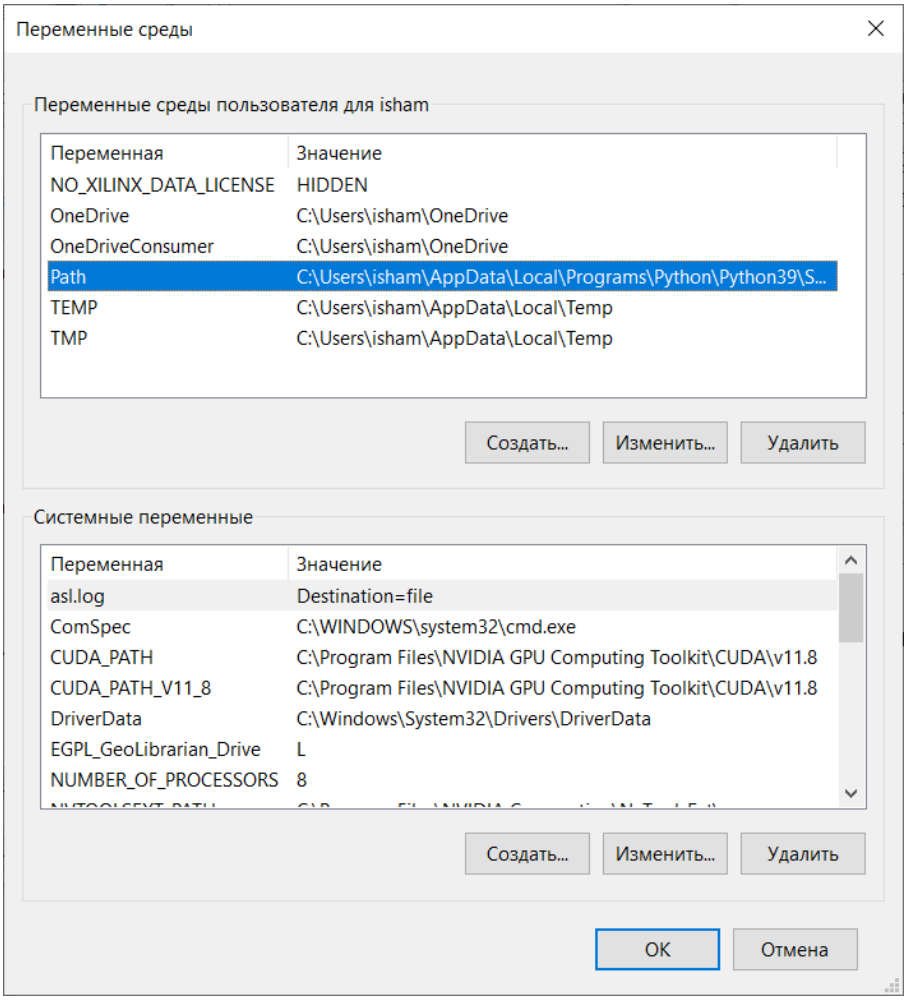

Рисунок 8 - Окно переменных среды Windows
DOI: https://doi.org/10.60797/IRJ.2025.152.19.12

Далее устанавливаем OpenCV для Python. В командной строке cmd вводим следующую команду и дожидаемся конца установки:

pip install opencv-python

По аналогии с предыдущим шагом, устанавливаем следующие библиотеки:

- pandas 2.2.2;
- tqdm 4.66.4;
- matplotlib 3.9.0.

Команды для установки вышеперечисленных библиотек приведены в листинге 1.

Листинг 1. Команды для установки библиотек

```
::загрузка и установка библиотеки pandas 2.2.2
pip install pandas
::загрузка и установка библиотеки tqdm 4.66.4
pip install tqdm
::загрузка и установка библиотеки matplotlib 3.9.0
pip install matplotlib
```

Следующим шагом будет установка CUDA Toolkit. Для того, чтобы выбрать подходящую версию программного обеспечения CUDA Toolkit необходимо проверить поддержку CUDA установленного графического ускорителя. В нашем случае используется NVIDIA RTX 3070 Ti, который поддерживает как 11, так и 12 версию CUDA Toolkit. В процессе обучения и тестирования модели искусственного интеллекта было замечено несколько критических ошибок при использовании версии 12, которые приводили к остановке обучения и потере всего прогресса. При использовании версии 11 критических ошибок не возникало, по этой причине рекомендуется использовать CUDA Toolkit 11.8.

Для установки CUDAToolkit11.8 переходим на официальный сайт компании NVIDIA, в разделе загрузок выбираем используемую операционную систему, разрядность, версию и вариант установки (рисунок 9).

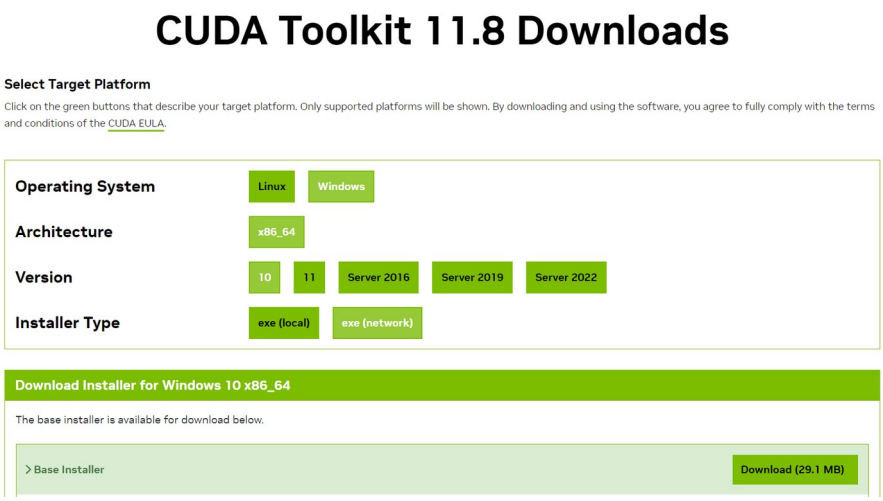

Рисунок 9 - Выбор платформы установки NVIDIA CUDA

DOI: https://doi.org/10.60797/IRJ.2025.152.19.13

Далее устанавливаем PyTorch с поддержкой технологии CUDA. Поддержка технологии CUDA в нашем случае играет ключевую роль, поскольку, как было сказано ранее, при отсутствии графического адаптера всю обработку данных будет выполнять центральный процессор (CPU), мощностей которого недостаточно для наших целей.

Ранее мы установили CUDA версии 11.8, поэтому устанавливаем PyTorch с поддержкой CUDA версии 11.8, введя следующую команду в командную строку cmd:

pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

С помощью команд через командную строку устанавливаем следующие библиотеки (листинг 2):

- openmim 0.3.9;
- mmengine 0.10.4;
- mmcv 2.1.0.

Листинг 2. Команды для установки библиотек

::загрузка и установка библиотеки openmim 0.3.9

pip install -U openmim

::загрузка и установка библиотеки mmengine 0.10.4

mim install mmengine

::загрузка и установка библиотеки matplotlib mmcv 2.1.0

mim install mmcv==2.1.0

С установкой библиотек для работы с инструментами мы закончили, переходим к установке непосредственно самих инструментов. Ключевым в разработке системы является MMAction2 – инструмент с открытым исходным кодом для распознавания информации в видеоданных.

Для установки MMAction2 необходимо через командную строку перейти по пути, где в дальнейшем будут храниться файлы инструмента, используя команду «cd».

Для загрузки и установки необходимо ввести следующие команды:

git clone https://github.com/open-mmlab/mmaction2.git

cd mmaction2

pip install -v -e

Помимо MMAction2, используемого для обучения модели, нам понадобится MMDeploy – инструмент для развертывания моделей на различных платформах и устройствах.

Для загрузки и установки необходимо ввести следующие команды:

git clone -b main https://github.com/open-mmlab/mmdeploy.git

pip install mmdeploy

pip install mmdeploy-runtime-gpu

Последним шагом будет установка библиотек, необходимых для запуска демонстрационного приложения, также написанного на языке Python. Для этого необходимо установить следующие библиотеки:

- onnxruntime с поддержкой графических ускорителей;
- loguru;
- omegaconf.

Команды для установки библиотек приведены в листинге 3.

Листинг 3. Команды для установки библиотек

::загрузка и установка библиотеки onnxruntime с поддержкой gpu

pip install onnxruntime-gpu

::загрузка и установка библиотеки loguru

mim install loguru

::загрузка и установка библиотеки omegaconf

mim install omegaconf

Для обучения модели искусственного интеллекта необходимо создать 2 аннотационных файла, в которых будут указаны путь до данных, используемых в обучении и тестировании, а также соответствующие значения на выходе модели. Для этого создадим файл «create_anns.py», код которого рассмотрим далее.

Как уже было сказано ранее, код для разрабатываемой системы на основе искусственного интеллекта для перевода невербального языка в лингвистическую модель написан на языке программирования Python, поэтому код во всех дальнейших листингах приведен на языке программирования Python.

В первую очередь импортируем необходимые библиотеки (листинг 4).

Листинг 4. Код импортирования библиотек

```
import os
import cv2
import pandas as pd
from tqdm import tqdm
from glob import glob
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.dpi'] = 300
plt.rcParams['savefig.dpi'] = 300
plt.rcParams.update({'font.size': 10})
```

Затем указываем директории расположения данных для обучения и тестирования, аннотации ко всем данным (листинг 5).

Листинг 5. Директории файлов

```
DATA_DIR = os.path.join(os.getcwd(), 'dataset')
TRAIN_DIR = os.path.join(DATA_DIR, 'train')
TEST_DIR = os.path.join(DATA_DIR, 'test')
ANNOTATIONS_DIR = os.path.join(DATA_DIR, 'annotations')
```

Получаем данные для обучения и для тестирования из директорий, записываем их в массив, а также сортируем (листинг 6).

Листинг 6. Заполнение массивов данных

```
ann = pd.read_csv(os.path.join(DATA_DIR, 'annotations.csv'), sep='\t')
ann.head()

train_files = sorted(glob(os.path.join(TRAIN_DIR, '*')))
test_files = sorted(glob(os.path.join(TEST_DIR, '*')))
```

Для проверки созданных массивов просмотрим первые файлы и визуализируем в виде покадровой развертки с шагом в 4 кадра (листинг 7). Так мы можем удостовериться, что для обучения модели будут использоваться правильные данные.

Листинг 7. Визуализация данных

```
train_sample = train_files[0]
test_sample = test_files[0]

def visualize_frames(video_path: str, title: str, num_frames=5):
    fig, axes_list = plt.subplots(nrows=1, ncols=5, figsize=(10, 3))
    vidcap = cv2.VideoCapture(video_path)

    STEP = 4
    for i in range(num_frames):
        vidcap.set(cv2.CAP_PROP_POS_FRAMES, (i * STEP) - 1)
        success, frame = vidcap.read()
        if success:
            axes_list[i].imshow(frame[:, :, ::-1])
            axes_list[i].axis('off')
    plt.suptitle(title,
                 x=0.05, y=1.0,
                 horizontalalignment='left',
                 fontweight='semibold',
                 fontsize='large')
    plt.show()


visualize_frames(train_sample, 'Training sample')
visualize_frames(test_sample, 'Test sample')
```

Заключительным этапом будет обработка полученных на предыдущих этапах массивов данных для обучения и тестирования модели искусственного интеллекта и дальнейшее формирование аннотационных файлов для обучения и для тестирования (листинг 8).

Листинг 8. Обработка массивов и формирование аннотационных файлов

```
NUM_CLASSES = len(ann['text'].unique())
classes = {label: label_id for label, label_id in zip(ann['text'].unique(), range(NUM_CLASSES))}

ann_train = []
ann_test = []
print(type(ann))
for file in tqdm(train_files + test_files):
    video_id = file.split('\\')[-1][:-4]
    label = ann[ann['attachment_id'] == video_id]['text'].to_string(index=False)
    class_id = classes[label]
    line = file + ' ' + str(class_id) + '\n'
    if ann[ann['attachment_id'] == video_id]['train'].bool():
        ann_train.append(line)
    else:
        ann_test.append(line)

with open('ann_train.txt', 'w') as train_file, open('ann_test.txt', 'w') as test_file:
    train_file.writelines(ann_train)
    test_file.writelines(ann_test)
```

После выполнения Python-скрипта «create_anns.py» в текущей директории мы получим файлы «ann_train.txt» и «ann_test.txt», в которых записаны пути до видеоданных, и числовое значение показанного жеста.

На этом подготовка к обучению модели искусственного интеллекта заканчивается, далее приступаем непосредственно к обучению модели.

Обучение и тестирование модели искусственного интеллекта осуществляется с помощью рассмотренного ранее инструмента MMAction2. Для того чтобы приступить к обучению, необходимо создать конфигурационный файл на языке программирования Python. Для этого создадим файл «model_config.py», код которого рассмотрим далее.

В качестве модели искусственного интеллекта была выбрана модель «Improved Multiscale Vision Transformers for Classification and Detection» (MViTv2) с вариантом архитектуры «small». Для повышения значения метрик обученной модели было принято решение использовать предобученную модель (скелет) на обучающей выборке Kinetics400 с репозитория инструментария MMAction2. Для этого запишем в конфигурационный файл параметры, представленные в листинге 9.

Листинг 9. Конфигурация модели искусственного интеллекта

```
# Model settings
model = dict(
    type='Recognizer3D',
    backbone=dict(
        type='MViT',
        arch='small',
        drop_path_rate=0.2,
        init_cfg=dict(
            type='Pretrained',
            checkpoint=
                                    'https://download.openmmlab.com/mmaction/v1.0/recognition/mvit/converted/mvit-small-
p244_16x4x1_kinetics400-rgb_20221021-9ebaaeed.pth',
            prefix='backbone.')),
    data_preprocessor=dict(
        type='ActionDataPreprocessor',
        mean=[123.675, 116.28, 103.53],
        std=[58.395, 57.12, 57.375],
        format_shape='NCTHW'),
    cls_head=dict(
        type='MViTHead',
        in_channels=768,
        num_classes=1001,
        label_smooth_eps=0.1,
        average_clips='prob'))
```

Для того, чтобы контролировать процесс обучения модели, будем использовать журнал событий, куда будут записываться промежуточные результаты обучения модели, такие как потери, время итерации, скорость обучения, полученные метрики и другие. Для этого добавим в конфигурационный файл код, представленный в листинге 10.

В данном случае мы будем записывать следующую информацию во время обучения модели:

- номер эпохи обучения («Epoch»);

- номер итерации («Iter»);
- скорость обучения («lr»);
- среднее время вывода последней итерации («time»);
- среднее время загрузки данных последней итерации («data_time»);
- расчетное время прибытия для завершения обучения («eta»);
- усредненный результат потерь по модели за последнюю итерацию («loss»).

Помимо этого, мы будем сохранять информацию о каждых трех завершенных эпохах и будем сохранять контрольные точки, с которых можно будет продолжить обучение в случае возникновения критической ошибки.

Листинг 10. Конфигурация журнала событий

```
# Logging settings
default_scope = 'mmaction'
default_hooks = dict(
    runtime_info=dict(type='RuntimeInfoHook'),
    timer=dict(type='IterTimerHook'),
    logger=dict(type='LoggerHook', interval=100, ignore_last=False),
    param_scheduler=dict(type='ParamSchedulerHook'),
    checkpoint=dict(
        type='CheckpointHook', interval=3, save_best='auto', max_keep_ckpts=5),
    sampler_seed=dict(type='DistSamplerSeedHook'),
    sync_buffers=dict(type='SyncBuffersHook'))
env_cfg = dict(
    cudnn_benchmark=False,
    mp_cfg=dict(mp_start_method='fork', opencv_num_threads=0),
    dist_cfg=dict(backend='nccl'))
log_processor = dict(type='LogProcessor', window_size=20, by_epoch=True)
vis_backends = [dict(type='LocalVisBackend')]
visualizer = dict(
    type='ActionVisualizer', vis_backends=[dict(type='LocalVisBackend')])
log_level = 'INFO'
load_from = None
resume = True
```

Следующим шагом будет конфигурирование входных данных для обучения модели искусственного интеллекта. Для этого укажем пути до аннотационных файлов, а также директории файлов для обучения и тестирования модели (листинг 11).

Листинг 11. Конфигурация путей

```
# Specify dataset paths
dataset_type = 'VideoDataset'
data_root = './dataset/train/'
data_root_val = './dataset/test/'
ann_file_train = './ann_train.txt'
ann_file_val = './ann_test.txt'
ann_file_test = './ann_test.txt'
```

Поскольку для обучения модели входные видеоданные преобразуются в так называемые «Sample Frames» (выборка отдельных кадров видеоизображения), необходимо сконфигурировать обработку и преобразование видеопотока в отдельные кадры, преобразованные в разрешение 224 на 224 точки.

В нашем случае мы имеем следующую конфигурацию:

- окно сэмплирования входных данных – 32;
- интервал между соседними кадрами выборки – 2;
- количество выбранных кадров между интервалами – 1;
- разрешение кадра (ширина и высота) – 224 на 224.

В качестве дополнительного параметра зададим горизонтальное отражение изображения, поскольку жесты можно показывать обеими руками. Итоговую конфигурацию обработки входных данных можно увидеть в листинге 12. Аналогичным путем сконфигурирована обработка данных для тестирования модели (листинг 13).

Листинг 12. Конфигурация обработки данных для обучения

```
train_pipeline = [
    dict(type='DecordInit', io_backend='disk'),
    dict(
        type='SampleFrames',
        clip_len=32,
        frame_interval=2,
        num_clips=1,
        out_of_bound_opt='repeat_last'),
    dict(type='DecordDecode'),
    dict(type='Resize', scale=(224, 224), keep_ratio=False),
    dict(type='Flip', flip_ratio=0.5, direction='horizontal'),
```

```
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='PackActionInputs')
```

Листинг 13. Конфигурация обработки данных для тестирования

```
train_pipeline = [
    dict(type='DecordInit', io_backend='disk'),
    dict(
        type='SampleFrames',
        clip_len=32,
        frame_interval=2,
        num_clips=1,
        out_of_bound_opt='repeat_last'),
    dict(type='DecordDecode'),
    dict(type='Resize', scale=(224, 224), keep_ratio=False),
    dict(type='Flip', flip_ratio=0.5, direction='horizontal'),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='PackActionInputs')
```

Последний шаг – конфигурация параметров обучения. Для обучения модели мы задаем количество эпох обучения равное 15, и задаем в качестве вида сходимости градиентный клиппинг (параметр «тип оптимизатора» = «AdamW»). Конфигурация параметров обучения представлена в листинге 14.

Листинг 14. Конфигурация параметров обучения модели

```
# Training settigns
val_evaluator = dict(type='AccMetric')
test_evaluator = val_evaluator
train_cfg = dict(
    type='EpochBasedTrainLoop', max_epochs=1, val_begin=15, val_interval=3)
val_cfg = dict(type='ValLoop')
test_cfg = dict(type='TestLoop')
base_lr = 0.0016
optim_wrapper = dict(
    optimizer=dict(
        type='AdamW', lr=0.0016, betas=(0.9, 0.999), weight_decay=0.05),
    paramwise_cfg=dict(norm_decay_mult=0.0, bias_decay_mult=0.0))
param_scheduler = [
    dict(
        type='LinearLR',
        start_factor=0.1,
        by_epoch=True,
        begin=0,
        end=15,
        convert_to_iter_based=True)
]
auto_scale_lr = dict(enable=False, base_batch_size=64)
dist_params = dict(backend='nccl')
launcher = 'pytorch'
work_dir = 'work_dirs/mvit-slovo'
randomness = dict(seed=None, diff_rank_seed=False, deterministic=False)
```

После создания конфигурационного файла можно приступать к обучению модели искусственного интеллекта.

Тестирование модели искусственного интеллекта осуществляется с помощью рассмотренного ранее инструмента MMAction2. Для тестирования модели используется тот же самый конфигурационный файл, что и для обучения. После того, как модель искусственного интеллекта была обучена, необходимо

Для этого через командную строку необходимо перейти в директорию, в котором располагаются установленный ранее инструмент MMAction2 и конфигурационный файл для обучения модели. После этого необходимо вызвать Python-скрипт «test.py» из директории «./mmaction2/tools/» и в качестве первого аргумента передать конфигурационный файл «model_config.py», а в качестве второго аргумента передать файл обученной модели с расширением «PTH», расположенный в директории «./work_dirs/mvit-slovo/»:

py ./mmaction2/tools/test.py model_config.py best_acc_top1_epoch3.pth

После выполнения данного скрипта начнется процесс тестирования модели искусственного интеллекта, в командной строке можно будет увидеть прогресс обучения (рисунок 10).

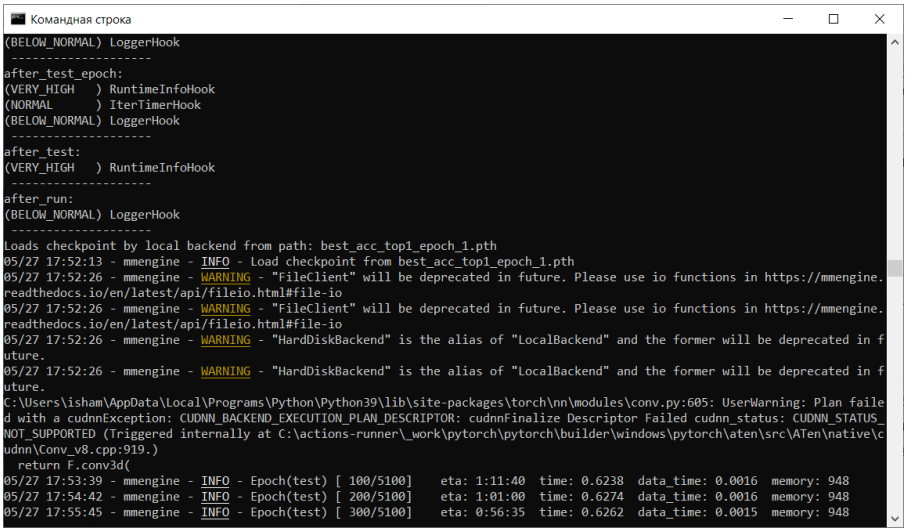

Рисунок 10 - Процесс тестирования модели ИИ
DOI: https://doi.org/10.60797/IRJ.2025.152.19.14

Процесс тестирования модели составляет менее продолжительный период времени, чем обучение, и также варьируется в зависимости от вычислительной мощности графического процессора.

По окончании тестирования мы получаем метрики, которые характеризуют качество получившейся модели. Для дальнейшего сравнения за основу мы возьмем параметр «acc/mean1», что означает точность – долю правильных ответов алгоритма. В нашем случае по результатам тестирования мы получили значение в 0,6016, что соответствует 60,16% правильных ответов.

Для дальнейшего использования обученной модели искусственного интеллекта необходимо конвертировать ее из расширения «PTH» в расширение «ONNX», поскольку далее используемая нами библиотека ONNX Runtime поддерживает исключительно этот формат.

Для конвертации модели используется инструментарий MMDeploy. Необходимый нам Python-скрипт «deploy.py» расположен в директории «./mmdeploy/tools/». Для его работы необходимо передать в качестве аргументов следующие обязательные параметры:

- deploy_cfg (конфигурационный файл, описывающий во что происходит конвертирование);
- model_cfg (конфигурационный файл, описывающий модель искусственного интеллекта, которая была обучена ранее);
- checkpoint (непосредственно сама обученная модель с расширением «PTH»);
- img (файл, используемый для конвертации модели – представляет собой видеоданные, которые использовались для обучения или для тестирования обученной модели).

Таким образом, для конвертации модели необходимо ввести следующую команду в командную строку cmd:

py ./mmdeploy/tools/deploy.py ./mmdeploy/configs/mmaction/video-recognition/video-recognition_onnxruntime_static.py model_config.py best_acc_top1_epoch3.pth ./dataset/test/0000f6de-ba06-43fc-9cc7-3a93f4350b04.mp4

После окончания процесса конвертации в директории, из которой был вызван скрипт, появится модель искусственного интеллекта с наименованием «end2end.onnx», которая в дальнейшем будет использована в тестовом приложении. Для удобства переименуем ее в «mvit32-2.onnx».

Для оценки качества обученной модели искусственного интеллекта сравним точность прогнозируемых жестов с другими моделями, которые были рассмотрены в первой главе и также способны распознавать и переводить жесты русского жестового языка.

Для удобства сравнения поместим метрики обученных сетей в сводную таблицу и сделаем выводы (таблица 5). Обученная нами модель расположена вверху таблицы.

Таблица 5 - Сравнение точности прогнозируемых жестов

DOI: https://doi.org/10.60797/IRJ.2025.152.19.15

Наименование модели искусственного интеллекта	Точность прогнозируемых жестов (accuracy), %	Используемые вычислительные мощности
MViTv2-small-32-2 (обученная нами модель)	60,16	GPU (NVIDIA CUDA)
Swin-large-16-3	48,04	GPU (NVIDIA CUDA)
Swin-large-32-2	54,84	GPU (NVIDIA CUDA)
Swin-large-48-1	55,56	GPU (NVIDIA CUDA)
ResNet-i3d-16-3	32,86	GPU (NVIDIA CUDA)

Наименование модели искусственного интеллекта	Точность прогнозируемых жестов (accuracy), %	Используемые вычислительные мощности
ResNet-i3d-32-2	38,38	GPU (NVIDIA CUDA)
ResNet-i3d-48-1	43,91	GPU (NVIDIA CUDA)
S3D-32	44,22	CPU
S3D-48	52,28	CPU
S3D-64	55,86	CPU

На основании данных, представленных в таблице, можно сделать вывод, что обученная нами модель «MViTv2» с архитектурой «small» показывает наилучшую точность прогнозируемых жестов среди всех представленных моделей.

Но, несмотря на это, она имеет некоторые недостатки. Например, для хорошей производительности в режиме реального времени необходимо использовать графический ускоритель с технологией NVIDIA CUDA [12], [13]. В противном случае скорость обработки данных заметно увеличивается, что делает невозможным распознавание и перевод жестов с помощью обученной модели искусственного интеллекта в режиме реального времени.

Для демонстрации обученной модели искусственного интеллекта [14], [15], способного распознавать и переводить жесты, было разработано демонстрационное приложение на языке Python.

Для запуска приложения необходимо запустить Python-скрипт из директории проекта, а также в качестве аргумента передать конфигурационный файл «config.yaml»:

py demo.py –p config.yaml

Следует отметить, что для работы приложения необходимо указать путь расположения обученной нами модели искусственного интеллекта с расширением «ONNX», а также параметр интервала кадров, который был выбран при обучении модели (в нашем случае равен двум). Поскольку ранее название модели было изменено на «mvit32-2.onnx», и при этом модель расположена в одной директории с исполняемым скриптом, то достаточно указать следующие параметры:

model_path: mvit32-2.onnx

frame_interval: 2

Результаты работы приложения можно увидеть на рисунке 11, где основную часть окна занимает входной поток видеоданных, поступающих с веб-камеры, а в нижней части отображается перевод распознанных жестов. В общей сложности обученная модель может распознать 1000 жестов русского жестового языка, а также имеет отдельный класс – «отсутствие жеста».

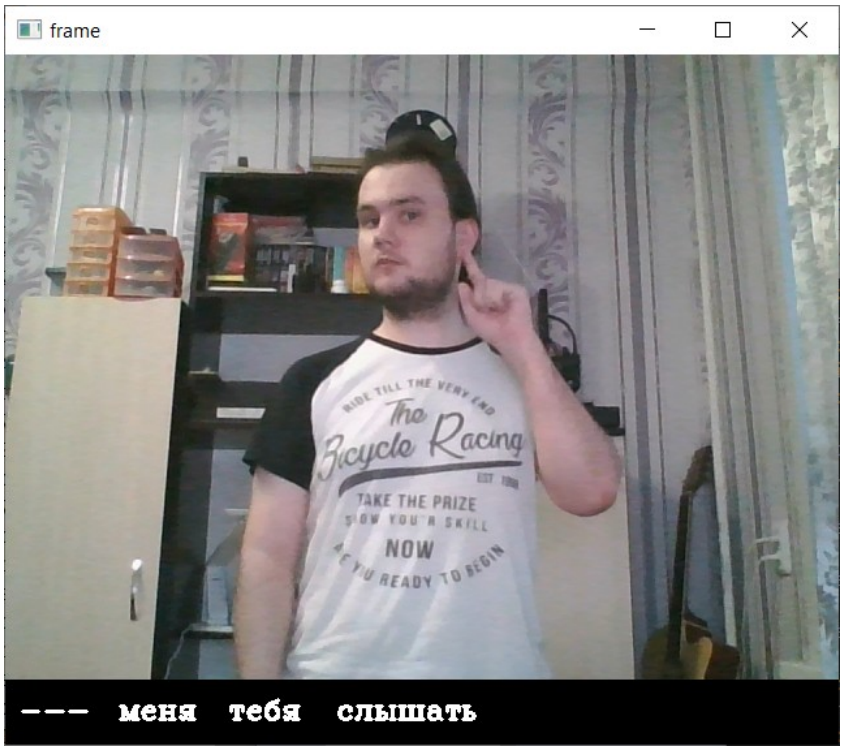

Рисунок 11 - Распознавание жеста «слышать»

DOI: https://doi.org/10.60797/IRJ.2025.152.19.16

Заключение

Подводя итоги проделанной работы, можно считать поставленную цель достигнутой: разработана информационная система на языке Python с использованием искусственного интеллекта [16], [17], позволяющая распознавать и переводить жесты русского жестового языка.

Результаты данного исследования имеют практическую значимость и вносят вклад в область создания программного обеспечения для распознавания жестовых языков с помощью современных моделей искусственного интеллекта. Разработанная система может распознавать и переводить жесты русского жестового языка в реальном времени, что позволяет улучшить качество коммуникации с людьми, способом коммуникации которых является использование жестового языка.

Помимо этого, результаты проделанной работы могут быть использованы для дальнейшего совершенствования распознавания русского жестового языка, используемого слабослышащими людьми, улучшения качества перевода невербального языка в вербальную модель.

Конфликт интересов

Не указан.

Рецензия

Бикмуллина И.И., доцент кафедры АСОИУ, КНИТУ-КАИ, Казань, Российская Федерация

DOI: https://doi.org/10.60797/IRJ.2025.152.19.17

Conflict of Interest

None declared.

Review

Bikmullina I.I., associate of professor, KNRTU-KAI, Kazan, Russian Federation

DOI: https://doi.org/10.60797/IRJ.2025.152.19.17

Список литературы / References

1. Храмушина О.В. Речевая и жестовая подготовка студентов с нарушениями слуха в условиях инклюзивного образования / О.В. Храмушина // Редакционная коллегия. — 2015. — С. 244.

2. Das A. Development of a Real Time Vision-Based Hand Gesture Recognition System for Human-Computer Interaction / A. Das, K. Maitra, S. Roy [et al.] // 2023 IEEE 3rd Applied Signal Processing Conference (ASPCON). — 2023. — P. 294–299. — DOI: 10.1109/ASPCON59071.2023.10396583.

3. Maitrey A. A Framework for Sign Language to Speech Conversion Using Hand Gesture Recognition Method / A. Maitrey, V. Tyagi, K. Singhal [et al.] // 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN). — Ghaziabad, 2023. — P. 333–338. — DOI: 10.1109/CICTN57981.2023.10140730.

4. Surekha P. Hand Gesture Recognition and voice, text conversion using / P. Surekha, N. Vitta, P. Duggirala [et al.] // 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS). — Coimbatore, 2022. — P. 167–171. — DOI: 10.1109/ICAIS53314.2022.9743064.

5. Nikolaev K. Development of a Neural Network Model for Recognizing Russian-Language Generated Texts / K. Nikolaev // 2024 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). — IEEE, 2024. — P. 396–400.

6. Chen Y. Action Detection in Badminton Courts Using AVA Dataset and MMAction2 Architecture with Slow Fast Model / Y. Chen // Highlights in Science, Engineering and Technology. — 2024. — № 85. — P. 783–789.

7. Li Y. MViTv2: Improved Multiscale Vision Transformers for Classification and Detection / Y. Li [et al.] // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — New Orleans, 2022. — P. 4794–4804. — DOI: 10.1109/CVPR52688.2022.00476.

8. Kapitanov A. Slovo: Russian Sign Language Dataset / A. Kapitanov, K. Karina, A. Nagaev // International Conference on Computer Vision Systems — Cham: Springer Nature Switzerland, 2023. — P. 63–73.

9. Kim J.H. ONNX-based Runtime Performance Analysis: YOLO and ResNet / J.H. Kim, D.E. Lee, S.B. Choi // The Journal of Bigdata. — 2024. — № 9 (1). — P. 89–100.

10. Bakhoda A. Analyzing CUDA workloads using a detailed GPU simulator / A. Bakhoda, G.L. Yuan, W.W. Fung [et al.] // 2009 IEEE international symposium on performance analysis of systems and software. — 2009. — P. 163–174.

11. De Smedt T. Pattern for python / T. De Smedt, W. Daelemans // The Journal of Machine Learning Research. — 2012. — № 13 (1). — P. 2063–2067.

12. Гибадуллин Р.Ф. Ускорение AES шифрования на аппаратно-программной платформе NVIDIA CUDA / Р.Ф. Гибадуллин, А.С. Яковлев, А.А. Новиков [и др.] // Вестник Технологического университета. — 2017. — Т. 20. — № 12. — С. 97–103.

13. Гибадуллин Р.Ф. Ускорение обработки SQL-запросов к базам данных на GPU посредством аппаратно-программной платформы NVIDIA CUDA / Р.Ф. Гибадуллин, А.Г. Савельев, М.Ю. Перухин // Вестник Технологического университета. — 2016. — Т. 19. — № 20. — С. 110–116.

14. Гаевская З.А. Оптимизация процессов повышения энергоэффективности зданий на основе энергомоделирования и машинного обучения / З.А. Гаевская, Х.М. Вафаева // Инновационный потенциал развития общества: взгляд молодых ученых : сборник научных статей 2-й Всероссийской научной конференции перспективных разработок : в 5 т., Курск, 01 декабря 2021 года. — Курск: Юго-Западный государственный университет, 2021. — Т. 4. — С. 213–218.

15. Kumar M. Investigation of cyber attacks using post-installation app detection method / M. Kumar, K. Muppavaram, K. Gotlur [et al.] // Cogent Engineering. — 2024. — № 11. — DOI: 10.1080/23311916.2024.2411859.

16. Borovkov A. Synergistic Integration of Digital Twins and Neural Networks for Advancing Optimization in the Construction Industry: A Comprehensive Review / A. Borovkov, K.M. Vafaeva, N. Vatin // Construction Materials and Products. — 2024. — № 7. — DOI: 10.58224/2618-7183-2024-7-4-7.

17. Mouli K. An analysis on classification models for customer churn prediction / K. Mouli, Dr.Ch. Raghavendran, V. Bharadwaj // Cogent Engineering. — 2024. — № 11. — DOI: 10.1080/23311916.2024.2378877.

Список литературы на английском языке / References in English

1. Hramushina O.V. Rechevaja i zhestovaja podgotovka studentov s narushenijami sluha v uslovijah inkljuzivnogo obrazovanija [Speech and gesture training of students with hearing impairment in the conditions of inclusive education] / O.V. Hramushina // Redakcionnaja kollegija [Editorial board]. — 2015. — P. 244. [in Russian]

2. Das A. Development of a Real Time Vision-Based Hand Gesture Recognition System for Human-Computer Interaction / A. Das, K. Maitra, S. Roy [et al.] // 2023 IEEE 3rd Applied Signal Processing Conference (ASPCON). — 2023. — P. 294–299. — DOI: 10.1109/ASPCON59071.2023.10396583.

3. Maitrey A. A Framework for Sign Language to Speech Conversion Using Hand Gesture Recognition Method / A. Maitrey, V. Tyagi, K. Singhal [et al.] // 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN). — Ghaziabad, 2023. — P. 333–338. — DOI: 10.1109/CICTN57981.2023.10140730.

4. Surekha P. Hand Gesture Recognition and voice, text conversion using / P. Surekha, N. Vitta, P. Duggirala [et al.] // 2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS). — Coimbatore, 2022. — P. 167–171. — DOI: 10.1109/ICAIS53314.2022.9743064.

5. Nikolaev K. Development of a Neural Network Model for Recognizing Russian-Language Generated Texts / K. Nikolaev // 2024 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). — IEEE, 2024. — P. 396–400.

6. Chen Y. Action Detection in Badminton Courts Using AVA Dataset and MMAction2 Architecture with Slow Fast Model / Y. Chen // Highlights in Science, Engineering and Technology. — 2024. — № 85. — P. 783–789.

7. Li Y. MViTv2: Improved Multiscale Vision Transformers for Classification and Detection / Y. Li [et al.] // 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). — New Orleans, 2022. — P. 4794–4804. — DOI: 10.1109/CVPR52688.2022.00476.

8. Kapitanov A. Slovo: Russian Sign Language Dataset / A. Kapitanov, K. Karina, A. Nagaev // International Conference on Computer Vision Systems — Cham: Springer Nature Switzerland, 2023. — P. 63–73.

9. Kim J.H. ONNX-based Runtime Performance Analysis: YOLO and ResNet / J.H. Kim, D.E. Lee, S.B. Choi // The Journal of Bigdata. — 2024. — № 9 (1). — P. 89–100.

10. Bakhoda A. Analyzing CUDA workloads using a detailed GPU simulator / A. Bakhoda, G.L. Yuan, W.W. Fung [et al.] // 2009 IEEE international symposium on performance analysis of systems and software. — 2009. — P. 163–174.

11. De Smedt T. Pattern for python / T. De Smedt, W. Daelemans // The Journal of Machine Learning Research. — 2012. — № 13 (1). — P. 2063–2067.

12. Gibadullin R.F. Uskorenie AES shifrovanija na apparatno-programmnoj platforme NVIDIA CUDA [Acceleration of AES encryption on NVIDIA CUDA hardware platform] / R.F. Gibadullin, A.S. Jakovlev, A.A. Novikov [et al.] // Vestnik Tehnologicheskogo universiteta [Bulletin of the Technological University of Russia]. — 2017. — Vol. 20. — № 12. — P. 97–103. [in Russian]

13. Gibadullin R.F. Uskorenie obrabotki SQL-zaprosov k bazam dannyh na GPU posredstvom apparatno-programmnoj platformy NVIDIA CUDA [Acceleration of SQL-queries processing to databases on GPU by means of NVIDIA CUDA hardware-software platform] / R.F. Gibadullin, A.G. Savel'ev, M.Ju. Peruhin // Vestnik Tehnologicheskogo universiteta [Bulletin of the Technological University of Russia]. — 2016. — Vol. 19. — № 20. — P. 110–116. [in Russian]

14. Gaevskaja Z.A. Optimizacija processov povyshenija jenergojeffektivnosti zdanij na osnove jenergomodelirovanija i mashinnogo obuchenija [Optimization of energy efficiency improvement processes in buildings based on energy modelling and machine learning] / Z.A. Gaevskaja, H.M. Vafaeva // Innovacionnyj potencial razvitija obshhestva: vzgljad molodyh uchenyh : sbornik nauchnyh statej 2-j Vserossijskoj nauchnoj konferencii perspektivnyh razrabotok : v 5 t., Kursk, 01 dekabrja 2021 goda [Innovative potential of society development: young scientists' view : collection of scientific articles of the 2nd All-Russian Scientific Conference of promising developments : in 5 vol., Kursk, 01 December 2021]. — Kursk: South-West State University, 2021. — Vol. 4. — P. 213–218. [in Russian]

15. Kumar M. Investigation of cyber attacks using post-installation app detection method / M. Kumar, K. Muppavaram, K. Gotlur [et al.] // Cogent Engineering. — 2024. — № 11. — DOI: 10.1080/23311916.2024.2411859.

16. Borovkov A. Synergistic Integration of Digital Twins and Neural Networks for Advancing Optimization in the Construction Industry: A Comprehensive Review / A. Borovkov, K.M. Vafaeva, N. Vatin // Construction Materials and Products. — 2024. — № 7. — DOI: 10.58224/2618-7183-2024-7-4-7.

17. Mouli K. An analysis on classification models for customer churn prediction / K. Mouli, Dr.Ch. Raghavendran, V. Bharadwaj // Cogent Engineering. — 2024. — № 11. — DOI: 10.1080/23311916.2024.2378877.