МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Технический директор
ООО «Программные системы»

_____ Е.В. Водяницкий

«___»_____ 2024 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка Android-приложения для распознавания жестового языка в режиме реального времени

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.04.04.2024.308-1678.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ А.Т. Латипова

Автор работы,
студент группы КЭ-229
_____ М.А. Свиридов

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистранта
студенту группы КЭ-229
Свиридову Михаилу Андреевичу,
обучающемуся по направлению
09.04.04 «Программная инженерия»
(магистерская программа «Искусственный интеллект и инженерия данных»)

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка Android-приложения для распознавания жестового языка в режиме реального времени.
2. **Срок сдачи студентом законченной работы:** 20.05.2024 г.
3. **Исходные данные к работе**

3.1. Гриф М.Г. Распознавание русского и индийского жестовых языков на основе машинного обучения. // Системы анализа и обработки данных, 2021. – № 3. – С. 53–74.

3.2. Miah A.S.M. Multistage Spatial Attention-Based Neural Network for Hand Gesture Recognition. // Computers, 2023. – Vol. 12. – no. 13. – P. 20–31.

3.3. Рюмин Д.А., Кагиров И.А., Аксенов А.А., Карпов А.А. Аналитический обзор моделей и методов автоматического распознавания жестов и жестовых языков. // Информационно-управляющие системы, 2021. – №. 6 (115). – С. 10–20.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести обзор литературы и существующих приложений по предметной области.

4.2. Подготовить набор данных, реализовать нейросетевую модель и провести ее обучение.

4.3. Реализовать Android-приложение по распознаванию жестового языка в режиме реального времени.
4.4. Провести тестирование приложения.
5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. А.Т. Латипова

Задание принял к исполнению М.А. Свиридов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

На сегодняшний день распознавание жестов и жестовых языков является одной из активно развивающихся областей компьютерного зрения. Результаты таких исследований могут находить различные применения, как в сурдопереводе, так и в программном обеспечении с жестовым интерфейсом.

По данным ВОЗ, более 5% населения мира – 466 миллионов человек, в том числе 34 миллиона детей – страдают от потери слуха в слышащем ухе, превышающей 35 децибел (дБ). По оценкам, к 2050 году более 700 миллионов человек, или каждый десятый, будут иметь такую потерю слуха [1].

Точное число таких людей в России неизвестно. По предварительным оценкам различных источников, более 9 миллионов людей имеют различные нарушения слуха.

Люди с такими нарушениями общаются с помощью жестовых языков. В них общение происходит не за счет артикуляции звуков, а за счет жестов, изменений положения тела и мимики для создания знаков. Функциональные и коммуникативные возможности в таком случае не уступают разговорным языкам. Язык жестов является основным языком глухого и слабослышащего населения, их основным средством общения, и он так же разнообразен, как и речь. Жестовые языки даже в рамках одной лингвистической группы обладают существенной вариативностью. Так, например, русскоговорящее население использует в своей жизни русский жестовый язык (РЖЯ) [2].

Поэтому разработка системы преобразования жестового языка в речь для улучшения взаимодействия людей с нарушениями слуха и без таких нарушений с учетом лингвистических особенностей жестов является актуальной социальной задачей для поддержания коммуникации между глухими и слышащими.

Постановка задачи

Целью выпускной квалификационной работы является разработка Android-приложения для распознавания жестового языка в режиме реального времени. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор литературы и существующих решений в области распознавания жестового языка;

2) подготовить набор данных, состоящий из различных видео фрагментов, для обучения нейросетевой модели для распознавания жестового языка в режиме реального времени;

3) реализовать нейросетевую модель на основе выбранной технологии и выполнить ее обучение на подготовленном наборе данных, привести результаты обучения с расчетом различных метрик;

4) выполнить проектирование и реализацию Android-приложения для распознавания жестового языка в режиме реального времени на основании составленных требований;

5) провести тестирование основных функций приложения, подтверждающих его работоспособность.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложения. Объем работы составляет 51 страницу, объем списка литературы – 26 источников.

Первая глава посвящена детальному анализу теоретических аспектов распознавания жестового языка. В рамках этой главы проведен обширный обзор существующих подходов и решений в данной области с использованием нейронных сетей и машинного обучения, выявлены и проанализированы их основные характеристики и особенности. Особое внимание уделено сравнительному анализу различных фреймворков, используемых для разработки Android-приложений, с целью выбора наиболее подходящего и перспективного варианта для дальнейшей работы.

Во второй главе описывается набор данных, который использовался для обучения нейросетевой модели по распознаванию жестового языка. Приведены примеры, входящих в него файлов и их ключевые особенности. А также описываются операции по предобработке, которые были применены к данным для получения наилучшего результата распознавания при обучении нейросетевой модели по распознаванию жестового языка в режиме реального времени.

Третья глава посвящена описанию технологий используемых в выбранной нейросетевой модели для распознавания видео фрагментов с жестовым языком в режиме реального времени. Разъяснены ключевые особенности в ее разработке и приведено описание гиперпараметров. Отображены результаты обучения на тестовой и обучающей выборках с подсчетом различных метрик, на основании которых можно судить об успешности работы модели.

В четвертой главе подробно описан процесс разработки Android-приложения для распознавания жестового языка в режиме реального времени. Рассмотрены шаги его проектирования, включая анализ как функциональных, так и нефункциональных требований к системе. Были составлены диаграмма вариантов использования системы и диаграмма деятельности для одного из этих вариантов. Подробно описана реализация разработанного приложения, основанная на выявленных требованиях. А также представлены результаты тестирования основных особенностей приложения, которые подтверждают корректность его работы.

В приложении приведен листинг кода обучения модели для распознавания жестового языка.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Описание предметной области

В жестовом языке, как и в любом другом, существует азбука, которая состоит из дактилей. Дактили – буквы алфавита. Различные положения пальцев обозначают знаки. Но, тем не менее, большинство жестов подобны не буквам, а словам, так как они выражают понятия (рисунок 1).

Рисунок 1 – Некоторые слова русского жестового языка

При разработке русской дактилологии стремились, чтобы дактилема (буква дактильного алфавита) была максимально похожа на свой оригинал. Во многом это удалось, но некоторые дактилемы все же весьма условные (рисунок 2).

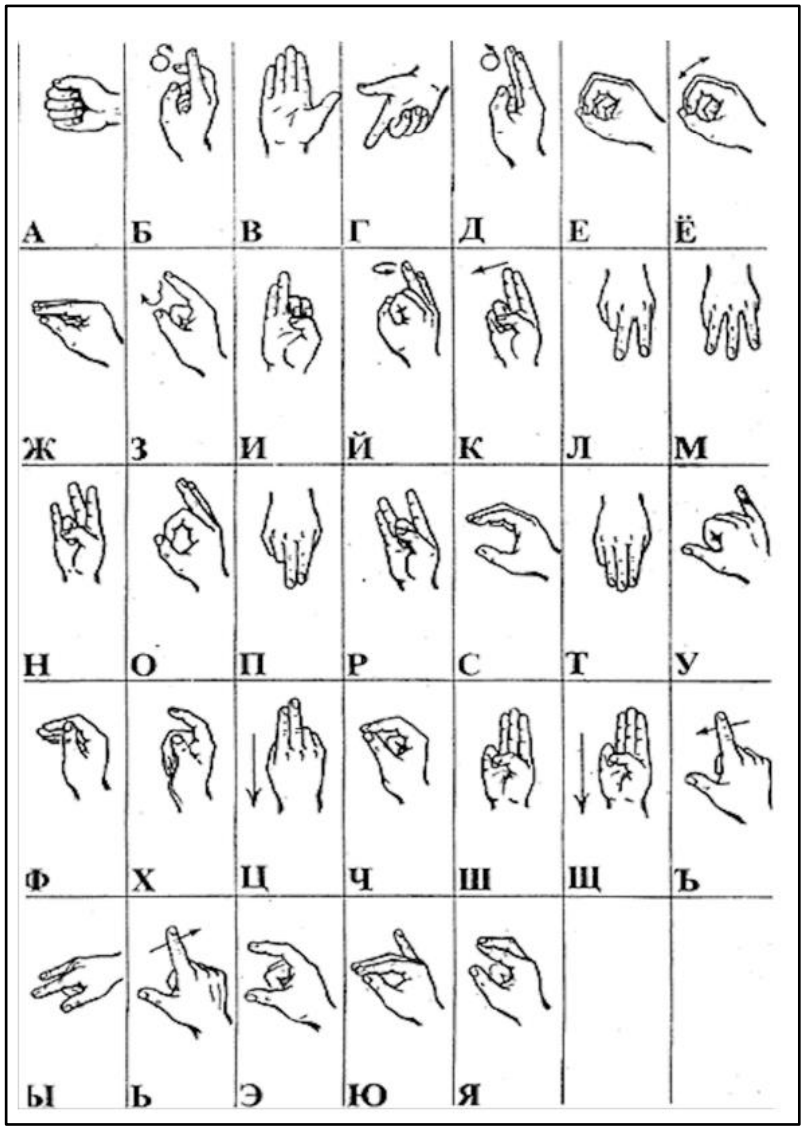

Рисунок 2 – Примеры дактилем РЖЯ

Как и в звуковых языках, в жестовых языках также присутствуют элементарные мельчайшие единицы, являющиеся аналогами морфем и фонем. Они не обладают собственным значением, но выполняют смыслоразличительную функцию. Поэтому при разработке системы для распознавания такого языка необходимо учитывать характерные для него особенности.

Распознавание языка жестов – сложная задача. Особенно трудным является распознавание динамических знаков, зависящих от движения. Русский жестовый язык имеет свои особенности, поэтому требует отдельного создания собственных алгоритмов распознавания. Кроме того, одной

из трудностей при разработке является сложность сбора достаточного количества данных для обучения модели.

На сегодняшний день существует два варианта распознавания жестов: на основе данных, получаемых от носимых на руках устройствах, и на основе компьютерного зрения с применением алгоритмов машинного обучения и нейронных сетей.

Первый вариант несет за собой значительный минус, так как требует ношение какого-либо устройства на руках. Второй способ предполагает выполнение распознавания на основании лишь полученного изображения с камеры. Поэтому в дальнейшем в работе будет рассматриваться только этот вариант.

Для решения подобной задачи можно выделить пять основных этапов [3].

Первым является сбор данных. Как правило, с данного этапа начинается любое исследование в области машинного обучения. На нем формируется набор аннотированных данных. Для задачи распознавания жестового языка широкое распространение получили корпусы, содержащие отдельные слова, числительные и буквы.

Далее идет сегментация. На данном этапе подразумевается выделение областей, содержащих артикуляторы диктора (в первую очередь руки). Существует несколько методов для проведения сегментации, но наилучшим по качеству является ручная сегментация [4].

Последующим этапом является выделение признаков. Он является одним из ключевых. На нем формируются входные данные и гипотеза предсказания, на которых будет обучаться модель.

Предпоследним идет обучение. На данном этапе происходит тренировка модели на входных признаках.

Заключительным этапом является оценка модели и получение итоговой гипотезы предсказания. На нем оцениваются результаты и вычисляет-

ся итоговая точность работы. После чего полученная модель используется для получения предсказаний в ходе распознавания.

Наиболее лучшими на данный момент считаются модели сверточной нейронной сети (CNN) и сети с долговременной краткосрочной памятью (LSTM), так как они обладают высоким потенциалом извлечения признаков. Также отличные результаты показывает использование трансформеров.

На основании исследований основным общим требованием в данном направлении является необходимость достижения точности перевода не ниже 90% [5].

Для оценки чаще всего используются метрики Precision (точность), Recall (полнота) и F-мера. Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а Recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. F-мера является гармоническим средним между ними.

Значительную сложность в достижении такого результата для системы по распознаванию РЖЯ представляет собой сбор корпусов данных, на которых можно было бы обучить модель, так как на текущий момент нет достаточного количества наборов, находящихся в свободном доступе.

1.2. Обзор существующих решений

Изначально для решения подобной задачи использовались алгоритмы машинного обучения, но с недавних пор для большинства разработок стали применяться методы глубокого обучения [6].

В работе [7] предлагается модель множественного параллельного потока: двумерная сверточная нейронная сеть для распознавания поз рук. Предлагаемая модель включает в себя несколько этапов и слоев для определения положения рук по картам изображений. В качестве наборов дан-

ных берутся три общедоступных эталонных набора Kaggle (10 классов), First Person (9 классов) и Dexter (7 классов), где количество данных для обучения составляет – 13375, 98842 и 19519 кадров. Средняя F-мера предлагаемого метода составляет 1,00, 1,00 и 0,92 при использовании набора данных о положении рук Kaggle, First Person и Dexter соответственно.

В статье [8] рассматриваются подходы к распознаванию жестовых языков глухих на примере русского жестового языка. Авторами был собран собственный корпус данных, который включает 35000 вариантов жестов (изображения и 10000 видеофайлов), построена модель рекуррентной сети (LSTM). Средняя точность правильного распознавания жестов проверялась на реальных видео, где добровольцы показывали предложения с помощью жестов РЖЯ, и она составила 0,95. Однако сам корпус данных не находится в открытом доступе.

Помимо решений для РЖЯ существует множество вариантов для других языков. Так, в статье [9] описывается решения для американского жестового языка. Основная идея в нем заключается в том, что рука рассматривается, как связный компонент на основе сегментации цвета и контура, а затем на полученной области выполняется определение жеста. Для сравнения жестов предлагается вариант использования расстояния Евклида между векторами. Полученные результаты достигают порядка 90%.

Еще один вариант для решения подобной задачи, но для американского жестового языка, рассматривается в статье [10]. В ней описан пример системы с использованием Microsoft Kinect для сбора изображений внешнего вида и глубины, а также платформу OpenNI+NITE для обнаружения и отслеживания рук. Формы рук, соответствующие буквам алфавита, характеризуются с помощью изображений внешнего вида и глубины и классифицируются с использованием случайных лесов. При таком подходе в исследовании была получена точность в размере 80%.

Помимо различных исследований, существует также рабочий программный продукт «СУРДОФОН 2», который представляет собой про-

граммно-аппаратный комплекс на базе клиент-серверной архитектуры, имеющий интегрированную систему управления, позволяющий распознавать русский жестовый язык в реальном времени и использоваться на различных устройствах: смартфонах, планшетах, ноутбуках. Его принцип заключается в том, что специальный коммуникатор распознает речь говорящего собеседника и переводит ее на русский жестовый язык при помощи компьютерного 3D-аватара.

Также существует система «Аватар» [11], которая также является автоматизированным переводчиком на РЖЯ. Она позволяет при помощи специализированного робота-помощника воспроизводить слова в жестовый язык (рисунок 3), а также предоставляет возможность перевода видео.

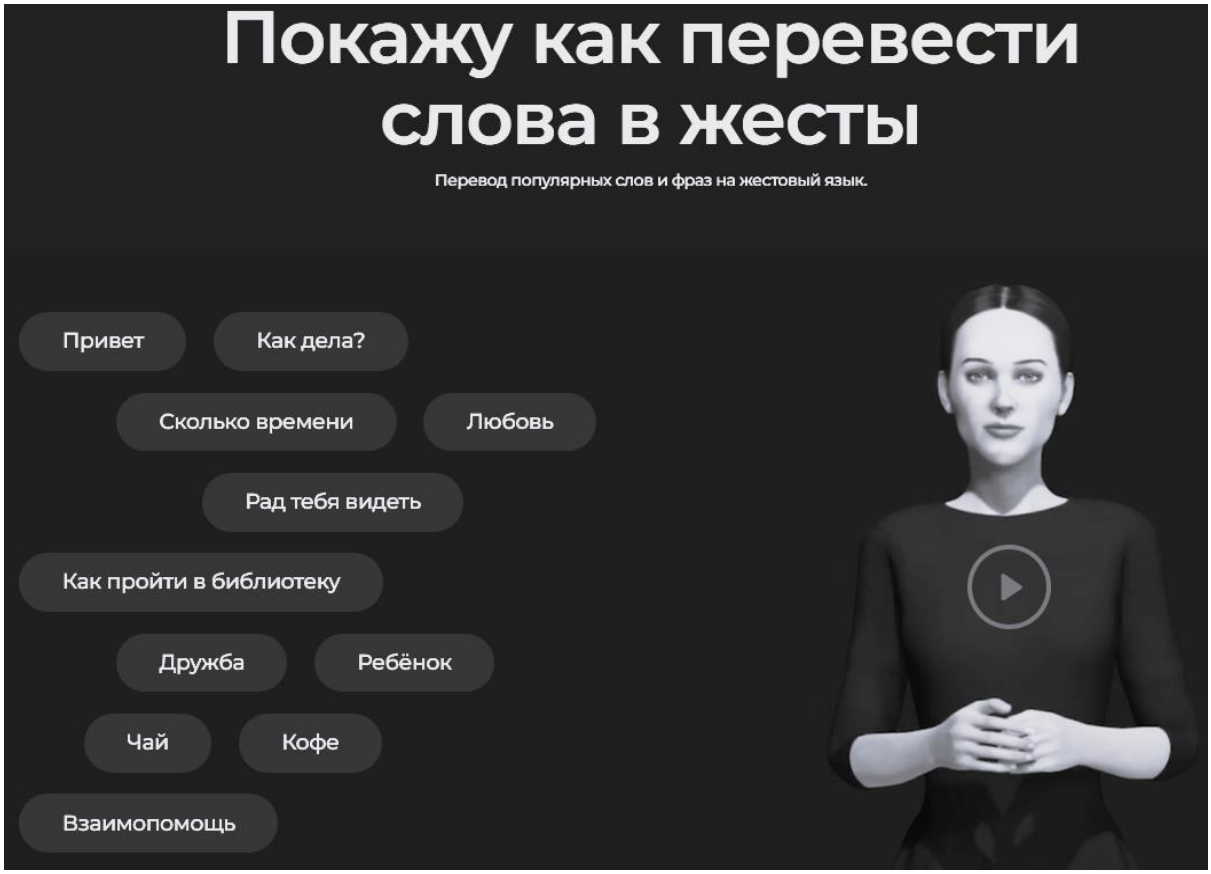

Рисунок 3 – Пример работы системы «Аватар»

Кроме того, распознавание жестов находит применение при осуществлении человеко-машинного взаимодействия [12–13].

1.3. Обзор фреймворков для разработки Android-приложения

Выбор подходящих инструментов для разработки является немаловажной частью разработки программного обеспечения. От этого будут зависеть дальнейшие перспективы развития приложения, а также его возможности к доработкам. В мобильной разработке по специфике создания продукта можно выделить два основных направления.

Одним из таких является использование нативных (от англ. native – родной) инструментов, в которых код создается под каждую платформу индивидуально на разработанном для нее языке и с использованием стандартных библиотек SDK (software development kit). Для Android основными в таком случае будут языки Java или Kotlin.

Плюсы такого подхода в том, что приложение создается со всеми имеющимися возможностями и максимально совместимо с устройствами, которые работают на соответствующих платформах. Но если необходимо выпустить приложение на другую операционную систему, то придется полностью переписывать его код и создавать соответствующий интерфейс.

Другим направлением является использование кроссплатформенных инструментов. В таком случае основная часть кода пишется единожды, а затем компилируется под отдельную платформу. Это значительно упрощает ситуацию, когда мобильное приложение необходимо выпустить на разных операционных системах, так как не придется переписывать его отдельно под каждый из вариантов.

Но в таком подходе есть и свои минусы. Поскольку код пишется один для всех, то фреймворки в некоторой степени не предоставляют полный функционал, а только ту часть, которая является общей для всех. В таком случае отдельные модули необходимо будет все равно писать на нативном языке, чтобы воспользоваться какими-то специфическими для платформы возможностями.

Для разработки кроссплатформенного приложения существует множество подходящих решений, которые на текущий момент продолжают

развиваться и широко используются по всему миру. Например, Kotlin Multiplatform Mobile (KMM) [14], Flutter [15], Apache Cordova [16] и .NET MAUI [17].

Последнее из перечисленных является платформой от компании Microsoft на основе .NET и использует языки C# и XAML для разработки приложений. Помимо использования общей части кода, данный фреймворк позволяет создать и использовать при необходимости модули специфичные для каждой из поддерживаемых операционных систем.

С помощью .NET MAUI можно разрабатывать приложения, которые могут работать на Android, iOS, macOS и Windows из одной общей базы кода. Поэтому, если необходимо выпустить разрабатываемый продукт на какую-либо другую платформу, в этом не возникнет сложностей. Также он обладает большим спектром плагинов, список которых постоянно пополняется различными пользовательскими решениями, способными упростить разработку. Именно поэтому данный фреймворк был выбран для реализации мобильного интерфейса.

Вывод по первой главе

В данной главе были проанализированы ключевые особенности жестовых языков и рассмотрены существующие подходы при построении систем для распознавания жестов. Из-за большого количества лингвистических особенностей для каждой языковой группы разработка такой системы должна производиться под каждый из языков. Поэтому в работе рассматривается русский жестовый язык. Также были приведены примеры существующих аналогичных решений и исследований в данной области для решения подобной задачи. Помимо этого, был проведен обзор существующих фреймворков для осуществления разработки Android-приложения, в результате которого был выбран .NET MAUI.

2. НАБОР ДАННЫХ

2.1. Описание набора данных

В работе использован набор данных видеоизображений русского жестового языка «Slovo» [18]. Он содержит 20400 RGB-видео для 1000 жестов, представляющих собой наиболее часто используемые пояснения и короткие фразы, алфавит и цифры от 194 спикеров. В каждом классе по 20 образцов. Все видео были собраны в помещении и различаются по сценам и условиям освещения. На рисунке 4 представлен пример фрагментов видео из обучающей выборки соответствующего жесту «хорошо».

Рисунок 4 – Фрагменты видео, соответствующего жесту «хорошо»

Набор данных разделен на обучающий и тестовый. Обучающий набор включает 15300 видеороликов, что составляет 75% от общей массы, а тестовый набор – 5100 видеороликов. Подобное разбиение является общепринятой практикой для моделей машинного обучения и нейронных сетей. Такая пропорция обеспечивает достаточное количество материала для обучения, чтобы удалось выявить закономерности в данных, и в то же время достаточное количество тестовых экземпляров для оценки работоспособности.

Общее время записи видео составляет приблизительно 9,2 часа. Около 35% видеороликов записаны в формате HD, а 65% видео – в разрешении FullHD. Средняя продолжительность видео с жестом – 50 кадров.

Помимо видеороликов набор содержит файл annotation.csv, в котором хранятся все основные сведения об имеющихся видео: идентификатор файла, идентификатор спикера, разрешение, длительность в кадрах, текст,

показанный на записи, признак использования в обучающей выборке. В таблице 1 приведены примеры записей этого файла.

Таблица 1 – Пример записей файла annotation.csv

ИД видео	ИД спикера	Текст	Разрешение	Кадры	Обучение
73237629-5530-4aa6-b8ad-9c34f8f28d77	185bd3a81d9d618518d10abebf0d17a8	Креветка	1920 x 1080	40	Да
2d79de65-6e3a-4f39-891d-332d8b8afc3b	185bd3a81d9d618518d10abebf0d17a8	Корова	1920 x 1080	57	Да
db89392a-1aa2-425b-ae7f-cb3252994243	20f1589682a8e64769ae2ebc21b5fb6b	Грубый	720 x 1280	61	Нет
e78a79b9-499e-4244-aa85-f4a861944f07	db573f94204e56e0cf3fc2ea000e5bdc	Дом	1280 x 720	92	Да
f17a6060-6ced-4bd1-9886-8578cfbb864f	db573f94204e56e0cf3fc2ea000e5bdc	Привет	1280 x 720	112	Да

Данный набор данных распространяется под лицензией CC BY-SA 4.0, которая разрешает использование адаптированных материалов для любых целей, и доступен для скачивания в свободном доступе.

2.2. Предобработка

Поскольку все первоначальные данные представлены в различных разрешениях с различной цветовой гаммой, около 13% изображений имеют вертикальную ориентацию, 1% сделаны в квадратной съемке, в то время как остальные представлены в горизонтальной, и большинство фрагментов имеют различную цветовую гамму, то необходимо провести процесс предобработки кадров видеороликов для обеспечения их единообразия.

За выполнение предобработки данных отвечает функция resize, входными параметрами которой являются значение новой размерности и массив значений пикселей изображения, представляющий собой трехмер-

ную матрицу из ширины, высоты и цветовых каналов. Исходный код на языке Python представлен в листинге 1.

Листинг 1 – Исходный код функции resize на языке Python

```
def resize(im, new_shape=(224, 224)):
    # текущая размерность [height, width]
    shape = im.shape[:2]
    if isinstance(new_shape, int):
        new_shape = (new_shape, new_shape)
    # Коэффициент масштабирования (new / old)
    r = min(new_shape[0] / shape[0], new_shape[1] / shape[1])
    # Вычисляем отступы
    new_unpad = int(round(shape[1] * r)), int(round(shape[0] * r))
    dw, dh = new_shape[1] - new_unpad[0], new_shape[0] - new_unpad[1]
    dw /= 2
    dh /= 2
    # Изменяем размер
    if shape[::-1] != new_unpad:
        im = cv2.resize(im, new_unpad, interpolation=cv2.INTER_LINEAR)
    top, bottom = int(round(dh - 0.1)), int(round(dh + 0.1))
    left, right = int(round(dw - 0.1)), int(round(dw + 0.1))
    # Добавляем границу
    value = (114, 114, 114)
    bConst = cv2.BORDER_CONSTANT
    im = cv2.copyMakeBorder(im, top, bottom, left, right, bConst, value)
    #Стандартизация
    im = (im - mean) / std
    return im
```

Основным этапом этого процесса является изменение разрешения. Это не только помогает снизить вычислительную нагрузку при обработке видеоматериала, который в программном коде представляется, как матричное представление, но и создает единообразие данных. Поэтому каждый кадр приводится к разрешению 224 на 224 пикселей. Подобное разрешение было выбрано исходя из того, что изображение с такими размерами позволяет все еще сохранить детали и обеспечить приемлемое качество визуализации, но при этом является не настолько большим, чтобы увеличить вычислительную сложность. Также такой размер является наиболее универсальным и подходит для многих уже предобученных моделей с популярными архитектурами.

При изменении разрешения для сохранения качества данных была применена линейная интерполяция [19]. Суть этого метода заключается в получении усредненного значения пикселя между соседними значениями,

что позволяет уменьшить разрешение с минимальной потерей важных деталей.

Также для соблюдения единообразия всех видеофрагментов были добавлены границы, чтобы сделать размер всех кадров одинаково пропорциональным.

Финальным шагом после изменения разрешения является нормализация значений. Для этого была применена стандартизация. Ее основная суть заключается в том, что из полученных на предыдущем шаге значений вычитается среднее и затем выполняется деление на стандартное отклонение. Это позволяет добиться более стабильного и эффективного обучения нейронной сети и улучшить ее результаты, а также избежать переобучения.

Вывод по второй главе

В данной главе было представлено описание набора данных «Slovo» по русскому жестовому языку, который включает в себя 20400 видео для 1000 жестов, сделанных 194 спикерами. Данные разделены на обучающие (15300 видеороликов) и тестовые (5100 видеороликов) выборки.

Также был представлен исходный код функции на языке Python, отвечающей за процесс предобработки данных, приведено описание ключевых операций, включающих изменение разрешения до 224 на 224 пикселей, линейную интерполяцию для сохранения основных признаков при изменении размерности, стандартизацию для нормализации значений и добавление границы для соблюдения единого размера изображений. Все эти операции играют немаловажную роль в обеспечении качества данных и эффективного обучения модели, так как набор данных содержит записи в различных вариантах.

3. ПОСТРОЕНИЕ НЕЙРОСЕТЕВОЙ МОДЕЛИ

Общепринятый подход для решения задач компьютерного зрения – это представлять изображение, как значения пикселей в трехмерном массиве, состоящем из высоты, ширины и количества цветовых каналов. После чего применять к таким матричным представлениям операции свертки. Но такой подход не совсем подходит, когда надо рассматривать входную картинку, как целое.

3.1. Vision Transformer

Вдохоновляясь успехами использования моделей с трансформерами в области обработки естественного языка была предложена идея их использования для изображений. Такой вид архитектуры был назван Vision Transformer (ViT) [20].

Основная суть таких моделей заключается в том, чтобы представить изображение в виде его разбиения на небольшие участки. После чего эти части преобразовать в линейную последовательность и передать на вход в трансформер, который в дальнейшем применяет механизм внимания уже для отдельных фрагментов. Пример архитектуры Vision Transformer представлен на рисунке 5.

Входными данными для таких моделей является многомерный вектор, называемый тензором, состоящий из значений пикселей нескольких изображения, объединенных по каждому цветовому каналу. Таким образом, для модели, принимающей на вход последовательность из 16 изображений размером 224 на 224 пикселя с 3 цветовыми каналами, формируется вектор размерностью 3×16×224×224.

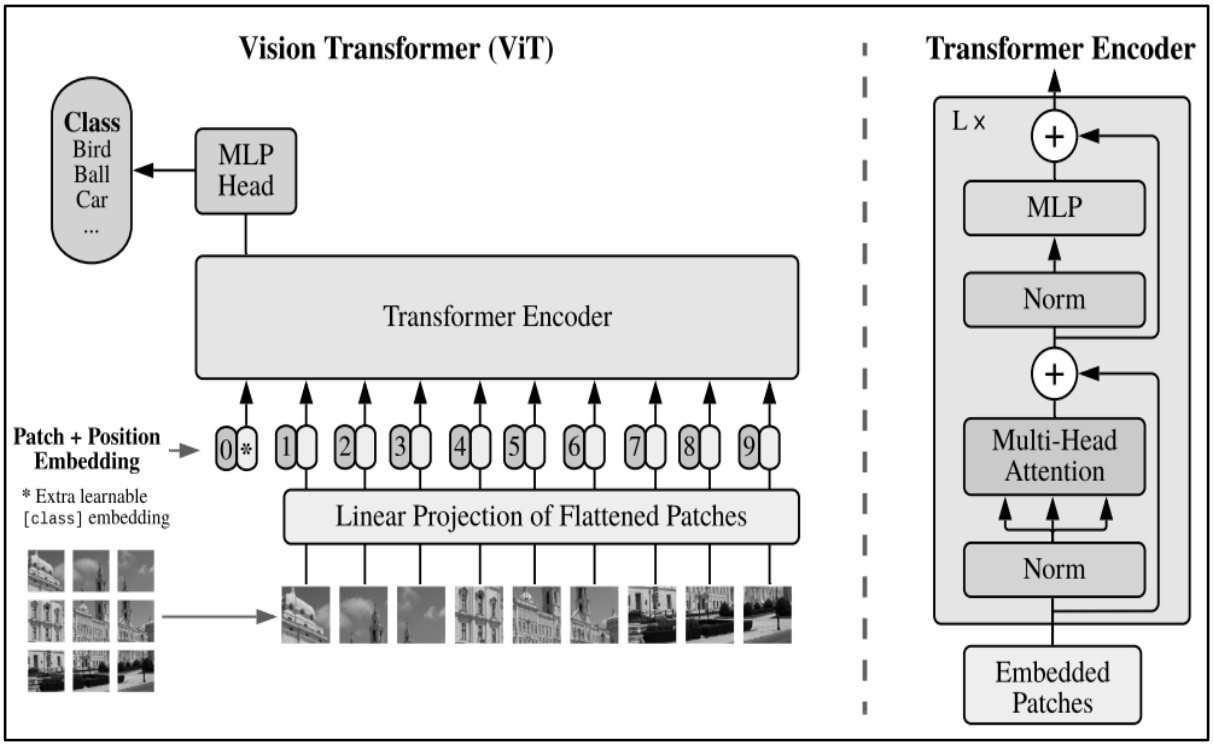

Рисунок 5 – Архитектура модели Vision Transformer (ViT)

Такая простая стратегия, применяемая в модели ViT, показывает отличные результаты как в обучении с нуля, так и в сочетании с работой на предварительно обученых больших наборах данных.

3.2. Multiscale Vision Transformer

Модель Multiscale Vision Transformer (MViT) [21] была специально разработана для задач распознавания видео. Такие трансформеры имеют несколько этапов масштабирования. Начиная с разрешения изображения и небольшого количества измерений этапы иерархически расширают объем этих измерений и уменьшают пространственное разрешение. Для достижения такого эффекта был представлен механизм внимания пулинга (рисунок 6).

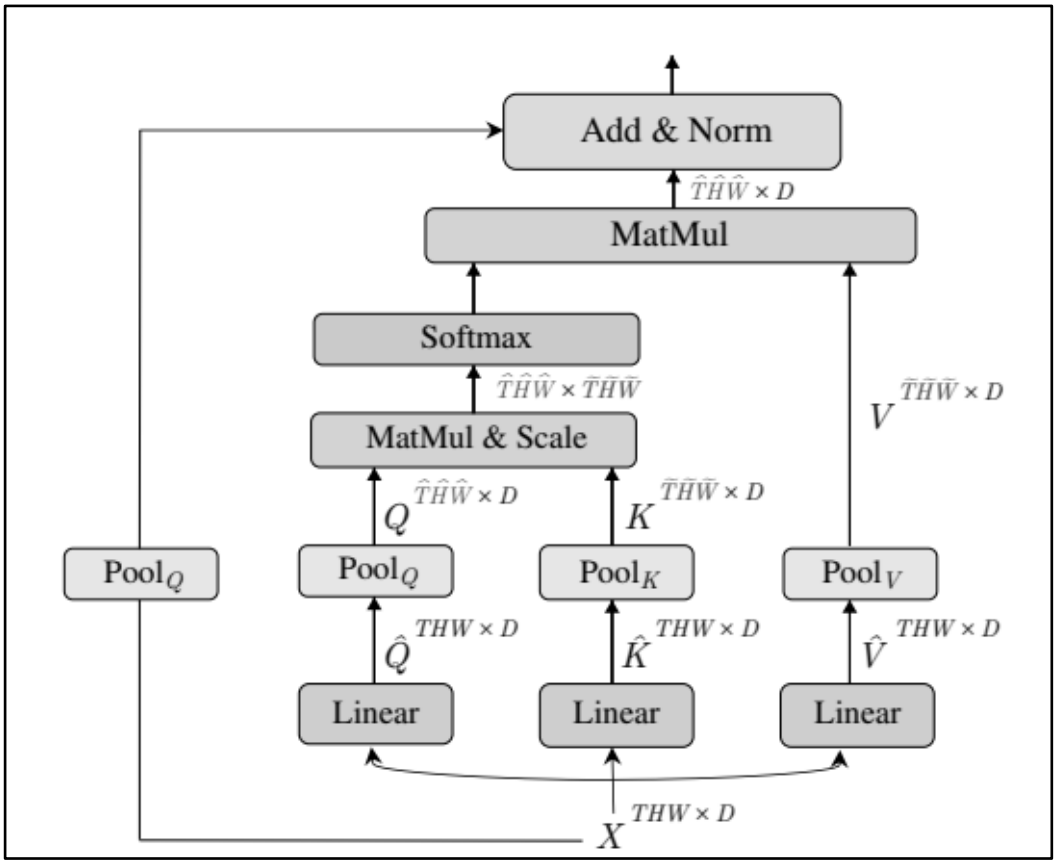

Рисунок 6 – Механизм внимания пулинга

Основная идея заключается в представлении входящего вектора-тензора X с количеством измерений D, как промежуточный тензор-запрос Q, ключ-тензор K и значение-тензор V, полученных применением линейной операции умножения на соответствующие векторы весов и применением к ним операции пулинга, вычисляемых по формулам 1–3:

$$Q = \rho(XW_q), \tag{1}$$

$$K = \rho(XW_k), \tag{2}$$

$$V = \rho(XW_v), \tag{3}$$

где W_q, W_k, W_v – векторы соответствующих весов, ρ – операция пулинга.

После чего вычисляются значения механизма внимания по формуле (4):

$$Z = Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{D}}\right)V. \tag{4}$$

В результате каждый шаг масштабирования содержит множественное количество блоков с трансформером. Это создает пирамидальную структуру, пример которой приведен на рисунке 7.

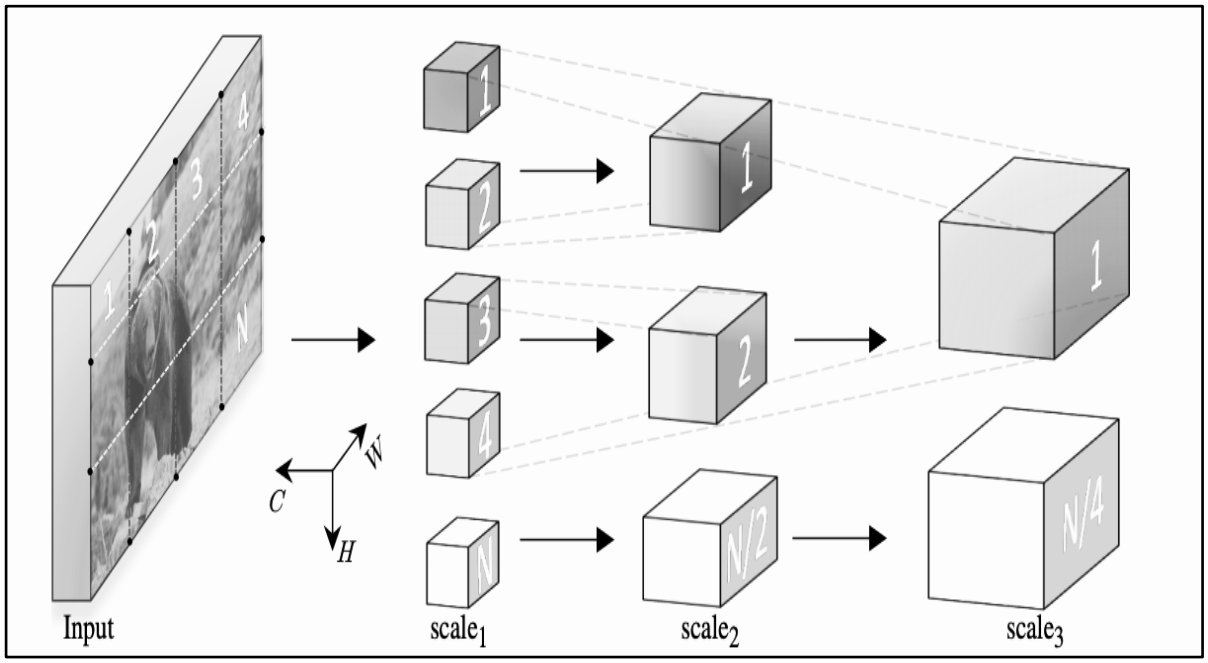

Рисунок 7 – Иерархическая структура MViT

Не смотря на то, что MViT показывает перспективы в своей способности моделировать взаимодействия между токенами, они сосредотачиваются на контенте, а не на структуре. Моделирование пространственно-временной структуры полностью основано на «абсолютном» позиционировании эмбеддингов. А именно, способ, которым MViT моделирует взаимодействие между двумя участками, будет изменяться в зависимости от их абсолютного положения на изображениях, даже если их относительное положение остается неизменным.

3.3. Multiscale Vision Transformer v2

Для решения этой проблемы в версии Multiscale Vision Transformer v2 (MViTv2) [22] были добавлены относительные позиционные эмбеддинги, которые зависят только от относительного расстояния между токенами, во время механизма внимания пулинга. И при этом со-

храняется относительная несложность и стоимость вычислений. Пример механизма указан на рисунке 8.

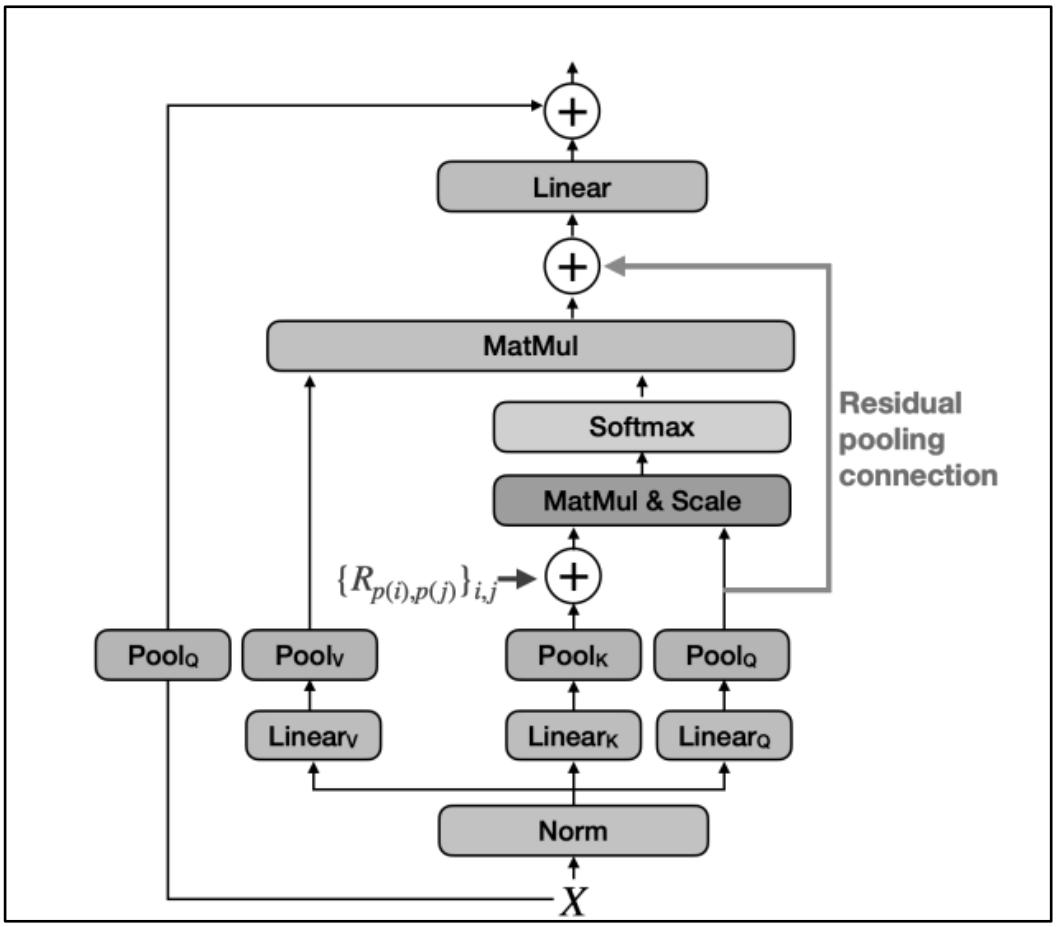

Рисунок 8 – Улучшенный механизм внимания пулинга

В таком случае механизм внимания будет вычисляться по формулам (5) и (6):

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T + E^{(rel)}}{\sqrt{d}}\right)V, \quad (5)$$

$$E_{i,j}^{(rel)} = Q_i R_{p(i),p(j)}, \quad (6)$$

где R – позиционный эмбеддинг.

На основании этого итоговый вектор Z будет рассчитываться по формуле (7):

$$Z = Attention(Q,K,V) + Q. \quad (7)$$

Подобное изменение позволило добиться успехов в повышении точности для задач распознавания видео. Результаты сравнения с другими мо-

делями, обученными на наборе данных Kinetic-400, представлены в таблице 2.

Таблица 2 – Сравнительные результаты обучения моделей на наборе данных Kinetic-400

Модель	Точность
SlowFast	79,8
MViTv1, 16×4	78,4
MViTv1, 32×3	80,2
MViTv2-S, 16×4	81,0
MViTv2-B, 32×3	82,9
Swin-L	84,9

Таким образом улучшенная архитектура MViTv2 является надежной общей основой для задач компьютерного зрения в области видео. Она демонстрирует современную производительность в различных тестах распознавания видео и может точно анализировать входной видеосигнал.

3.4. Обучение нейросетевой модели

Поскольку для задач распознавания видео недостаточно просто обрабатывать отдельные кадры, так как необходимо учитывать контекст в движении, то для обучения модели для распознавания жестового языка наилучшим вариантом будет использование предназначенного для этого различных наборов инструментов. Одним из таких является MMAction2 [23]. Он является набором инструментов с открытым исходным кодом, поддерживающим многочисленные модели понимания видео и предоставляющим готовые решения для различных вариаций задач распознавания действий.

Для достижения наилучших результатов обучения нейронной сети по распознаванию жестового языка в качестве стартовой контрольной точки использовались результаты предобучения на наборе данных Kinetic-400.

Тестирование гиперпараметров

Существует несколько гиперпараметров, которые могут повлиять на итоговое качество обучения модели. Для их более лучшей настройки были

проведены эксперименты по определению их вклада на итоговую точность.

Одним из таких гиперпараметров является шаг обучения (learning rate). Сильно малое значение приведет к увеличению времени обучения и может потребовать большего количества эпох для получения лучшей точности. Большое же значение ускоряет работу, но снижает качество. Результаты экспериментов и их влияние приведены в таблице 3.

Таблица 3 – Результаты экспериментов с разным шагом обучения и их влияние на итоговую точность модели

Шаг обучения	**Точность**
0,001	0,64
0,0001	0,81
0,00001	0,59
0,000001	0,55

Еще одним гиперпараметром, влияющим на модель, является значение процента отбрасывания нейронов во время обучения (dropout). Это один из методов регуляризации, который случайным образом зануляет значения нейронов на каждой итерации обучения. Более высокие показатели могут привести к высокой регуляризации, но в то же время модель может недообучиться. Наоборот низкие показатели могут привести к переобучению. Результаты экспериментов со значением dropout и их влияние на итоговую точность приведены в таблице 4.

Таблица 4 – Результаты экспериментов с разным значением dropout и их влияние на итоговую точность модели

Значение dropout	**Точность**
0	0,78
0,1	0,81
0,2	0,77
0,5	0,53
0,8	0,46

Также еще одним способом регуляризации является уменьшение весов (weight decay). Он изменяет значения весов на соответствующее значе-

ние при рассчете функции потерь. Чем больше значение, тем больше величина регуляризации. Результаты экспериментов для weight decay приведены в таблице 5.

Таблица 5 – Результаты экспериментов с разным значением dropout и их влияние на итоговую точность модели

Значение weight decay	**Точность**
0,01	0,77
0,001	0,79
0,0001	0,81
0,00001	0,80

Еще одним эффективным гиперпараметром является изменение шага обучения на различных эпохах (LR Scheduler Step Size). Данное значение корректирует скорость обучения на определенных эпохах с заданным шагом с целью более точного улучшения результата, когда уже не требуется изначальная величина. Результаты экспериментов приведены в таблице 6.

Таблица 6 – Результаты экспериментов с разным значением изменения шага обучения

LR Scheduler Step Size	**Точность**
1	0,65
5	0,74
10	0,81

Для выполнения корректировки шага используется значение гиперпараметра LR Scheduler gamma, задающее коэффициент, на который умножается скорость обучения. Результаты экспериментов приведены в таблице 7.

Таблица 7 – Результаты экспериментов с разным значением корректировки скорости обучения

Коэффициент умножения	**Точность**
0,1	0,81
0,5	0,64
0,9	0,58

Одним из решающих гиперпараметров является количество эпох. Чем больше значение, тем больше итераций обучения будет произведено. В случае малого количества итоговая точность может быть низкой, а в обратном случае вероятно переобучение. Результаты экспериментов с разными значениями эпох представлены в таблице 8.

Таблица 8 – Результаты экспериментов с разным количеством эпох обучения

Количество эпох	**Точность**
10	0,46
15	0,54
20	0,70
25	0,81
30	0,78

Как видно из экспериментов каждый гиперпараметр играет важную роль в итоговой точности. Поэтому их тщательный подбор необходим для достижения наилучшего результата. Но также необходимо помнить, что с различными значениями помимо точности может изменяться и время обучения модели.

Итоговый список с наилучшими значениями гиперпараметров и их эффектов на обучение модели представлен в таблице 9.

Таблица 9 – Итоговый список гиперпараметров

Гиперпараметр	**Значение**	**Эффект**
backbone.type	MViT	Задает тип используемой модели
backbone.drop_path_rate	0,2	Задает процент отбрасывания нейронов во время обучения с целью регуляризации
num_classes	1001	Задает количество классов, включая «пустое» значение.
optimizer.type	Adam	Задает тип оптимизатора.
optimizer.lr	0,0001	Задает начальный шаг оптимизатора для рассчета весов.
optimizer.weight_decay	0,0001	Задает корректировку значения весов в функции потерь с целью регуляризации.
vis_backends.type	TensorboardVisBackend	Задает тип визуализатора для просмотра визуального отображения обучения.

Окончание таблицы 9

Гиперпараметр	Значение	Эффект
max_epochs	25	Задает количество эпох для обучения модели.
scheduler.gamma	0,1	Задает коэффициент для изменения скорости обучения модели.
scheduler.step_size	10	Задает шаг, при котором изменяется скорость обучения.
mean	123,675; 116,28; 103,53	Задает средние значения, используемые для нормализации. Взяты значения в соответствии с используемой предобученной моделью, на основании которой начинается обучение.
std	58,395; 57,12; 57,375	Задает значения стандартного отклонения, используемые для нормализации. Взяты значения в соответствии с используемой предобученной моделью, на основании которой начинается обучение.

Результаты обучения

Методы классификации применяются в широком спектре научных и технических областей, поэтому крайне важно осуществлять правильную оценку эффективности выбранного алгоритма для решения поставленных задач. Несмотря на то, что в некоторых простейших случаях достаточно оценить способность классификатора точно идентифицировать различные классы, в других ситуациях это может быть недостаточным, и необходимо брать в расчет другие критерии.

Существуют множество способов оценки производительности алгоритмов [24]. В качестве самых распространненых выступают численные методы, которые строятся на использовании матрицы ошибок, которая содержит в себе количество корректно и некорректно классифицированных примеров для каждого класса. На рисунке 9 приводится пример такой матрицы для бинарной классификации, где один класс трактуется как положительный (P), а другой как отрицательный (F).

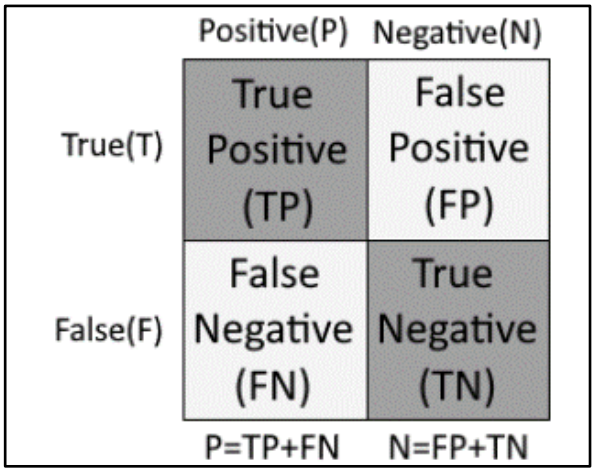

Рисунок 9 – Пример матрицы ошибок для бинарной классификации

Одним из эмпирических методов оценки точности классификации является значение корректно классифицированных примеров в процентах – правильность (accuracy). Она рассчитывается по формуле (8):

$$accuracy = \frac{TP + TN}{FP + FN + TP + TN}. \quad (8)$$

Также существуют метрики, которые позволяют оценить правильность классификации примеров для разных классов. Наиболее информативными среди них являются точность ($precision$), которая рассчитывается по формуле (9), и полнота ($recall$) с расчетом по формуле (10):

$$precision = \frac{TP}{TP + FP}, \quad (9)$$

$$recall = \frac{TP}{TP + FN}. \quad (10)$$

На основании изложенных метрик были рассчитаны соответствующие значения обучения модели для распознавания жестового языка. Для получения наилучших результатов были рассмотрены модели, использующие для классификации 16 кадров с интервалом 4 и 32 кадра с интервалом 3. Результаты обучения MViTv2 16x4 и MViTv2 32x3 представлены в таблицах 10 и 11 соответственно.

Таблица 10 – Результаты рассчетов метрик обучения модели MViTv2 16x4

Метрика	Значение
Accuracy	0,81
Precision	0,78
Recall	0,77

Таблица 11 – Результаты рассчетов метрик обучения модели MViTv2 32x3

Метрика	Значение
Accuracy	0,84
Precision	0,79
Recall	0,76

Полученные значения свидетельствуют о том, что уже на данном этапе модель показывает отличное качество распознавания, но в перспективе может быть доработана с помощью расширения обучаемого материала для достижения наилучшего результата.

Вывод по третьей главе

В данной главе были рассмотрены направления использования моделей нейронных сетей для распознавания видеоизображений на основе трансформеров. А именно – Vision Transformer и ее вариации в виде Multiscale Vision Transformer и Multiscale Vision Transformer v2.

Также были приведены основные гиперпараметры и эксперименты по их подбору для обучения модели с применением набора инструментов MMAction2.

Заключительным этапом является расчет метрик accuracy, precision и recall. В результате был получен средний результат в виде 83%, что является отличным показателем на данном этапе обучения и в перспективе может быть улучшен за счет расширения набора данных.

Исходный код для обучения модели представлен в листинге 1 приложения.

4. РАЗРАБОТКА ПРИЛОЖЕНИЯ

4.1. Проектирование

В ходе проектирования Android-приложения для распознавания жестового языка в режиме реального времени были сформулированы следующие функциональные требования:

1) приложение должно получать изображение с камер устройства в режиме реального времени;

2) приложение должно отображать на основном экране текстовое представление полученного от нейронной сети результата распознавания жеста;

3) приложение должно предоставлять пользователю возможность переключения камер устройства (фронтальная, задняя).

На их основании была составлена диаграмма вариантов использования (рисунок 10).

Рисунок 10 – Диаграмма вариантов использования

Также Android-приложение должно удовлетворять следующим нефункциональным требованиям:

1) приложение должно быть написано на языке C# с использованием XAML;

2) приложение должно быть реализовано с использованием платформы .NET MAUI;

3) приложение должно использовать предобученную нейронную сеть для получения результата распознавания жестов.

Исходя из изложенных требований, был разработан макет графического интерфейса Android-приложения для распознавания жестового языка в режиме реального времени (рисунок 11).

Рисунок 11 – Макет графического интерфейса Android-приложения

4.2. Реализация

На основании того, что для разработки была выбрана платформа .NET MAUI, то реализация должна осуществляться с использованием Vis-

ual Studio 2022 и целевой среды выполнения .NET 8.0. В качестве минимальной целевой версии Android была установлена версия 5.0.

Реализация функционала

Поскольку для обучения модели входящий видеоряд подвергался предобработке, то аналогичные действия необходимо выполнять и для кадров, получаемых от камеры устройства. Эту задачу выполняет функция Resize (листинг 2). В качестве входных параметров ей передается исходный кадр в виде объекта Mat, который представляет собой многомерную матрицу со значениями пикселей и количеством цветовых каналов, и его необходимый размер. На выходе возвращается аналогичный объект – результат после всех преобразований. Для выполнения операций используется библиотека Emgu.CV [25], являющаяся кроссплатформенной оболочкой .Net для библиотеки обработки изображений Open CV.

Листинг 2 – Программный код функции Resize на языке C#

```
public static float[][][] Resize(Mat im, Size newShape)
{
    Size shape = im.Size; // текущая размерность [height, width]

    CvInvoke.CvtColor(im, im, ColorConversion.Bgr2Rgb);
    im.ConvertTo(im, DepthType.Cv32F);

    // Коэффициент масштабирования (new / old)
    double r = Math.Min((double)newShape.Height / shape.Height, (dou-
ble)newShape.Width / shape.Width);
    // Вычисляем отступы
    Size newUnpad = new ((int)Math.Round(shape.Width * r) + 1,
(int)Math.Round(shape.Height * r));
    double dw = (newShape.Width - newUnpad.Width) / 2;
    double dh = (newShape.Height - newUnpad.Height) / 2;
    if (shape != newUnpad) // Изменяем размер
        CvInvoke.Resize(im, im, newUnpad, 0, 0, Inter.Linear);
    //Добавляем границу
    int top = (int)Math.Round(dh - 0.1);
    int bottom = (int)Math.Round(dh + 0.1);
    int left = (int)Math.Round(dw - 0.1);
    int right = (int)Math.Round(dw + 0.1);
    var value = new Emgu.CV.Structure.MCvScalar(114, 114, 114);
    CvInvoke.CopyMakeBorder(im, im, top, bottom, left, right, Border-
Type.Constant, value);
    //Стандартизация
    CvInvoke.Subtract(im, MEAN_MAT, im);
    CvInvoke.Divide(im, STD_MAT, im);

    return TransposeArray(im.GetData() as float[,,]);
}
```

Помимо преобразований, основной сложностью в реализации является применение обученной модели нейронной сети прямо в мобильном приложении. Для этой задачи оптимальным выбором стало использование набора инструментов Open Neural Network Exchange (ONNX) [26]. Он позволяет преобразовать исходную модель нейронной сети в собственный универсальный формат, который впоследствии можно использовать в различных системах на разных языках программирования без особых технических усложнений для задач как машинного, так и глубокого обучения. Структура модели в файле формата ONNX представлена на рисунке 12.

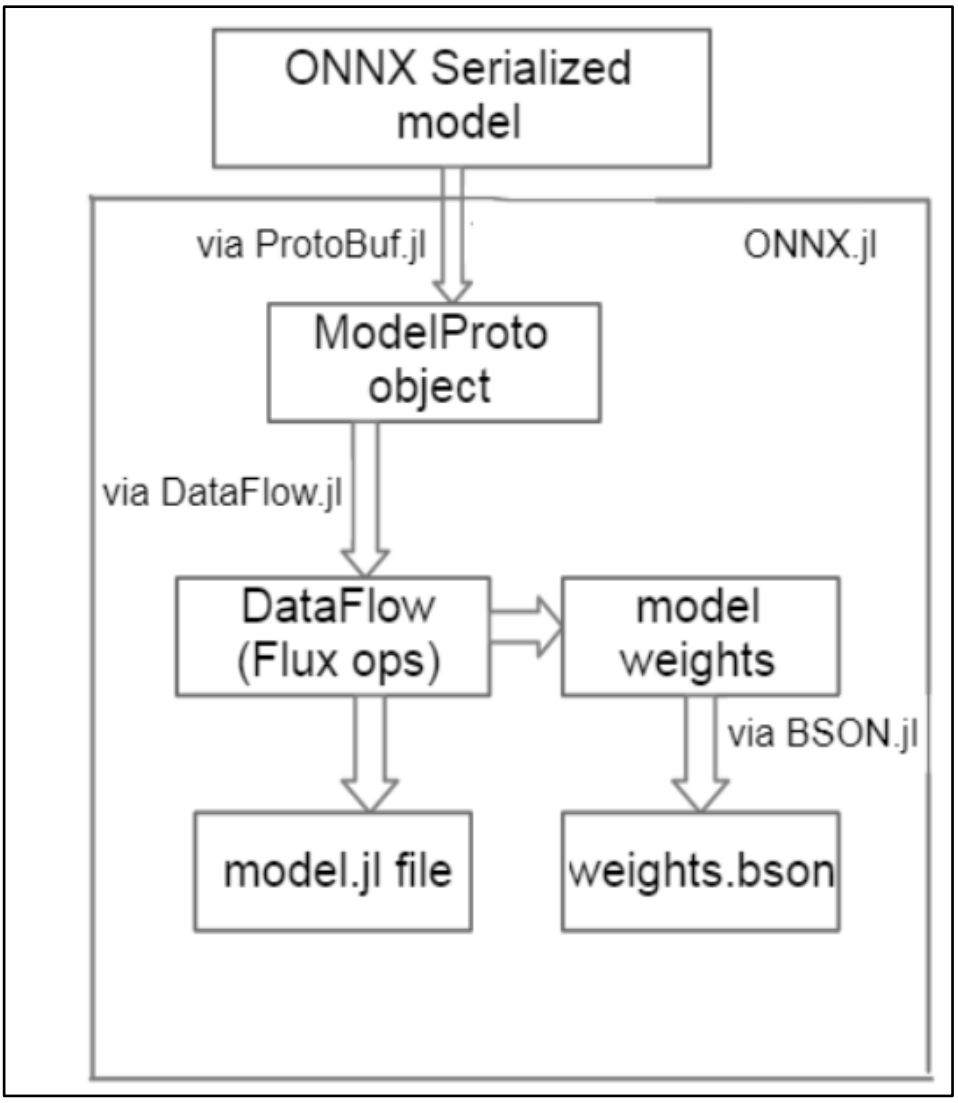

Рисунок 12 – Структура модели сериализованного ONNX-файла

Модель в формате ONNX создается на основе уже обученной модели PyTorch и тензора из входных данных. Для экспорта обученной модели используется функция exportToOnnx. Соответствующий код создания файла в данном формате на языке Python представлен в листинге 3.

Листинг 3 – Код создания модели в формате ONNX на языке Python

```
!pip install onnxruntime
!pip install onnxscript
!pip install onnx
import torch
income_model_path = '../my_model'
output_model_path = '../my_model.onnx'
def exportToOnnx(income_model_path, output_model_path):
  torch_model = torch.load(income_model_path)
  device = torch.device("cuda")
  torch_model = torch_model.to(device)
  input_tensor = input_tensor.to(device)
  torch.onnx.export(torch_model, input_tensor, output_model_path)
```

На текущий момент многие популярные языки программирования умеют работать с ONNX и предоставляют набор инструментов для работы с ним. На языке C# таким является пакет «Microsoft.ML.OnnxRuntime», предоставляемый и поддерживаемый компанией Microsoft. Поскольку разработка Android-приложения также осуществлялась на данном языке, то использование такого формата без труда интегрируется в разработанную программу.

Основной функциональностью приложения является распознавание жестового языка в режиме реального времени с интерпретацией полученного ответа в виде текстового представления. Распознавание и сбор кадров должен осуществляться в фоновом режиме. Для иллюстрации данной работы при использовании модели MViTv2 с количеством кадров 16 и интервалом их сбора 4 была разработана диаграмма деятельности (рисунок 14).

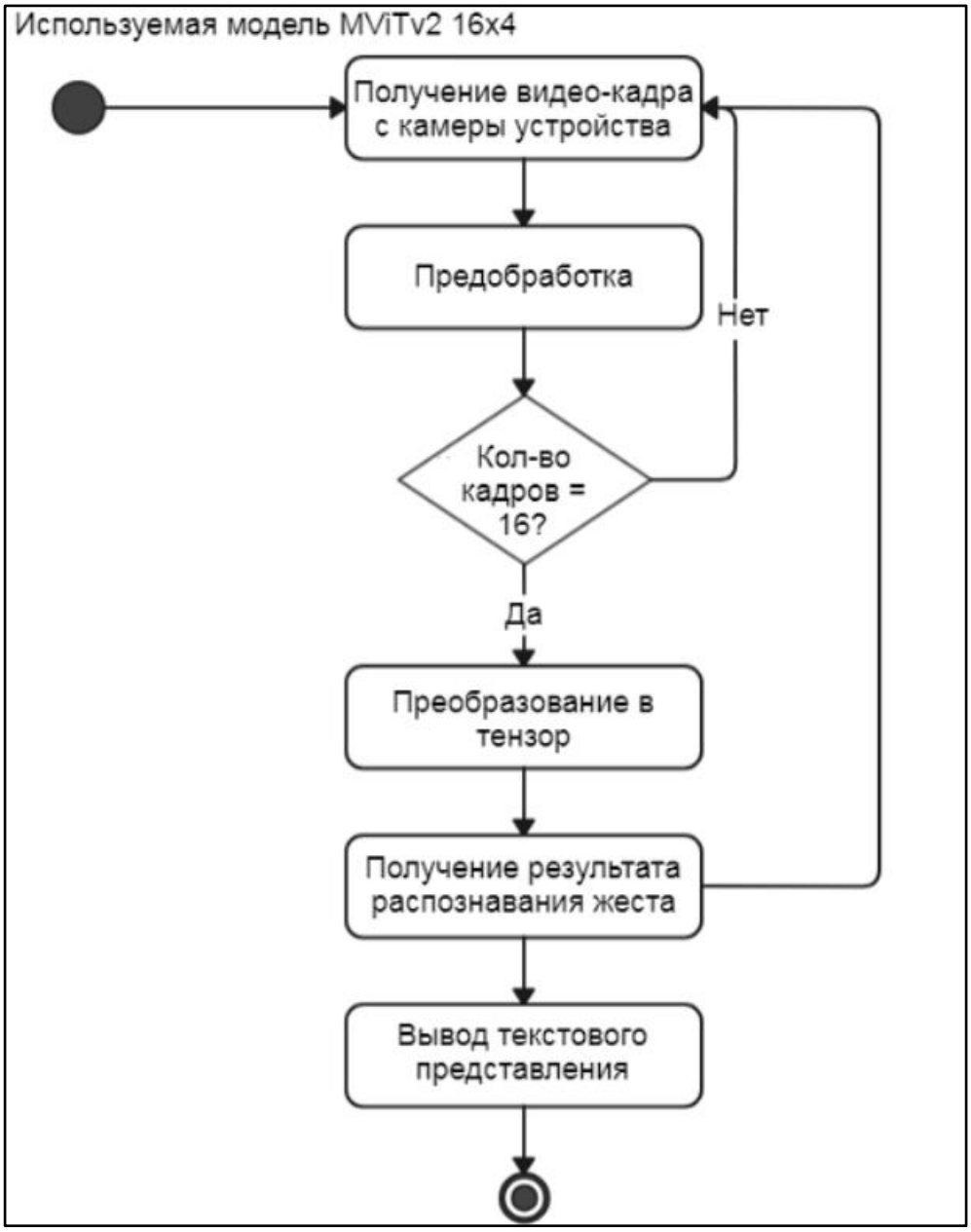

Рисунок 14 – Диаграмма деятельности для распознавания жестового языка

Для выполнения классификации полученного с камеры видеоряда необходимо сформировать тензор из массива предобработанных кадров в соответствии с размерностью входных данных модели. Затем передать его в модель в качестве входного параметра и получить текстовое представление классификации, если полученное вероятностное значение распознавания превышает пороговое. Соответствующий код формирования тензора и получения итоговых результатов распознавания жеста на языке C# представлен в листинге 4.

Листинг 4 – Код создания тензора и получения результата распознавания

```
List<string> predictionList = [];
try
{
    //Формируем тензор подающийся на вход в модель
    Dictionary<int, float[][]> tensors = [];
    for (int ch = 0; ch < CHANNEL_COUNT; ch++)
        tensors[ch] = tensorsList.SelectMany(s => s[ch]).ToArray();
    var data = tensors.SelectMany(s => s.Value).SelectMany(s =>
s).ToArray();
    var shape = InputShape.Select(s => (long)s).ToArray();
    long shapeCount = 1;
    for (int i = 0; i < shape.Length; i++)
        shapeCount *= shape[i];
    if (data.Length == (int)shapeCount)
    {
        var inputOrt = OrtValue.CreateTensorValueFromMemory(data, shape);
        var inputs = new Dictionary<string, OrtValue>
        {
            { InputName, inputOrt }
        };
        //Получаем результат распознавания от модели
        using var outputs = Session.Run(new RunOptions(), inputs, Ses-
sion.OutputNames)[0];
        //Находим индекс и макс. значение из полученного тензора
        var (value, indx) = out-
puts.GetTensorDataAsSpan<float>().ToArray().Select((v, i) => (v, i)).Max();
        //Текстовое представление полученного класса
        var gloss = Classes.Values[indx];
        //Если полученная вероятность больше пороговой, то выводим резуль-
тат
        if (value > THRESHOLD && gloss != EMPTY_STRING && (prediction-
List.Count == 0 || gloss != predictionList.Last()))
            predictionList.Add(gloss);
    }
    tensorsList = [];
}
catch(Exception e)
{
    throw new Exception($"Ошибка при распознавании жеста {e}");
}

return predictionList;
```

Реализация графического интерфейса

Графический интерфейс был реализован в виде двухстраничного приложения, состоящего из основной страницы с отображением полученного от камеры превью-изображения в режиме реального времени и страницы настроек, где пользователю предоставляется выбор используемой модели. При запуске приложения открывается загрузочный экран (рисунок 15).

Рисунок 15 – Загрузочный экран Android-приложения

В случае первого запуска приложение попросит предоставить разрешение на использование камеры устройства (рисунок 16).

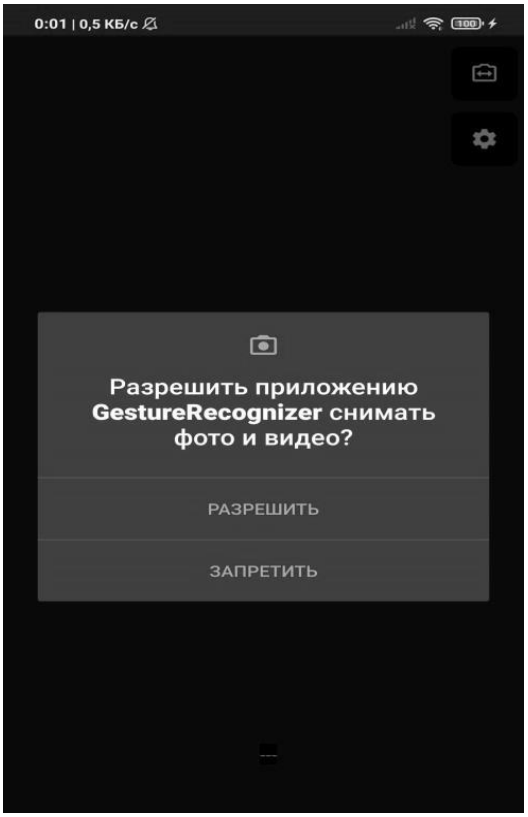

Рисунок 16 – Предоставление разрешения на использование камеры

После запуска на основной странице помимо превью-изображения отображается текстовое описание распознанного жеста в области внизу экрана и вмещает в себя 5 последних распознанных значений. При наличии на устройстве задней и фронтальной камер в интерфейсе реализована кпопка в правом верхнем углу, которая по нажатию на нее осуществляет переключение между ними.

Процесс распознавания жестов начинается сразу после загрузки всего графического интерфейса приложения и не требует от пользователя никакого ручного запуска и дополнительных настроек. Пример визульного оформления основной страницы Android-приложения для распознавания жестового языка в режиме реального времени приведен на рисунке 17.

Рисунок 17 – Основная страница Android-приложения

Также на основной странице реализована кнопка для открытия страницы настроек, содержащей список доступных моделей с возможностью переключения между ними (рисунок 18).

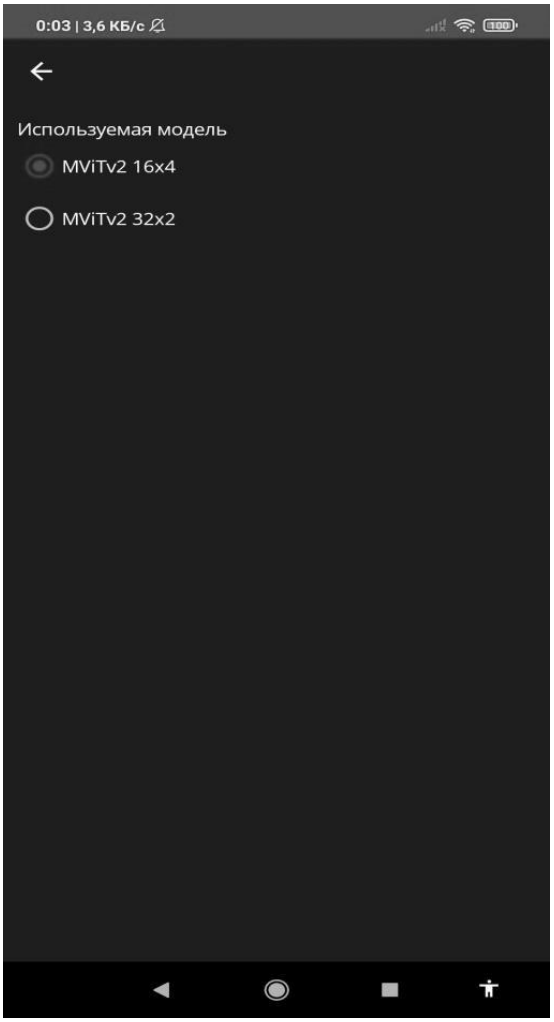

Рисунок 18 – Страница настроек Android-приложения

В случае возникновения ошибки во время работы предусмотрено появление всплывающего сообщения с соответствующим описанием проблемы. Соответствующее окно представлено на рисунке 19.

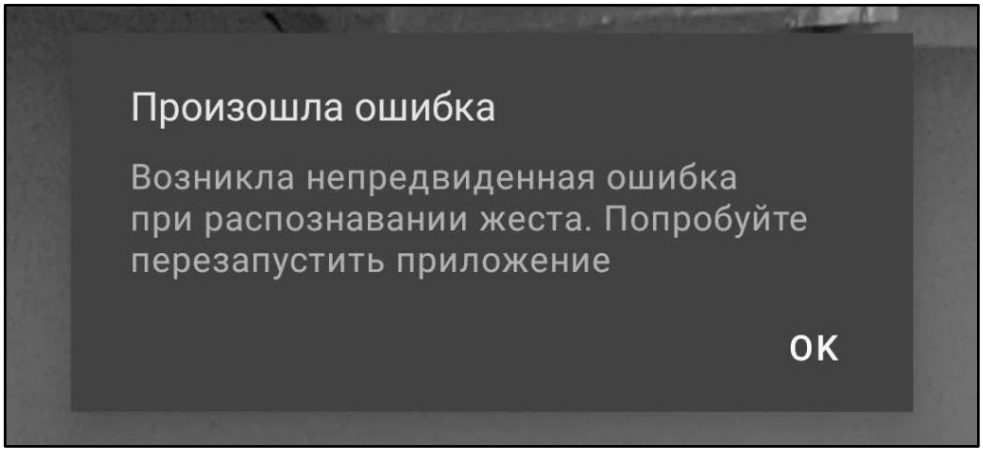

Рисунок 19 – Окно с сообщением об ошибке

4.3. Тестирование

Для проверки работоспособности функционала и графического интерфейса были сформулированы ключевые тесты. Результаты их прохождения отображены в таблице 12.

Таблица 12 – Результаты тестирования Android-приложения для распознавания жестового языка

№	Наименование теста	Шаги воспроизведения	Ожидаемый результат	Итоговый результат
1	Появление вопроса о предоставлении разрешения к камере устройства.	Первый запуск приложения.	При первом запуске появляется всплывающее окно с просьбой о предоставлении разрешений к камере устройства.	Совпадает с ожидаемым.
2	Отображение превью-изображения с камеры устройства в режиме реального времени.	Запуск приложения.	На экране устройства отображается полученное с камеры превью-изображение без задержек в режиме реального времени.	Совпадает с ожидаемым.
3	Отображение кнопки переключения камеры.	Запуск приложения на устройстве с задней и фронтальной камерами.	Кнопка переключения камеры отображается в правом верхнем углу.	Совпадает с ожидаемым.
4	Отображение области для текстового представления распознанного жеста.	Запуск приложения.	В нижнем углу экрана отображается текст «…», соответствующий пустой строке.	Совпадает с ожидаемым.
5	Распознавание жеста русского жестового языка.	Демонстрация одного из поддерживаемых жестов русского жестового языка после запуска приложения и сбора кадров.	В области для отображения текстового представления распознанного жеста отображается соответствующий текст.	Совпадает с ожидаемым.
6	Переключение камеры устройства.	Нажатие на соответствующую кнопку графического интерфейса.	Превью-изображение меняется в соответствии с используемой после переключения камерой.	Совпадает с ожидаемым.
7	Открытие страницы настроек.	Нажатие на соответствующую кнопку графического интерфейса.	Открывается страница настроек, содержащая список моделей для переключения.	Совпадает с ожидаемым.

Окончание таблицы 12

№	Наименование теста	Шаги воспроизведения	Ожидаемый результат	Итоговый результат
8	Переключение используемой модели.	Выбор нового значения на форме настроек и закрытие соответствующей формы.	После закрытия приложение продолжает распознавать жесты в режиме реального времени.	Совпадает с ожидаемым.
9	Сохранение выбранного варианта модели.	Открытие формы настроек после изменения используемой модели.	После открытия на форме настроек выделенной моделью является ранее установленный вариант.	Совпадает с ожидаемым.

В результате выполнения тестирования не было обнаружено ошибок в работоспособности как графического интерфейса, так и функционала. Android-приложение демонстрирует приемлемую скорость распознавания жестового языка в режиме реального времени.

Вывод по четвертой главе

В данной главе были описаны все функциональные и нефункциональные требования, а также ключевые особенности создания Android-приложения для распознавания жестового языка в режиме реального времени. Помимо этого были разработаны диаграмма вариантов использования и диаграмма деятельности. На их основании был спроектирован графический интерфейс, состоящий из двух страниц: основной и настроек. Также был реализован весь необходимый функционал для распознавания жестов в режиме реального времени, переключения камеры устройства и изменению используемой модели.

Для оценки работоспособности были сформулированы и приведены ключевые тесты, по результатам которых приложение в полном объеме удовлетворяет необходимым требованиям и показывает приемлемую скорость работы.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы было разработано Android-приложение, которое в режиме реального времени позволяет распознавать жестовый язык.

В ходе реализации был произведен обзор литературы по предметной области, рассмотрены существующие решения для распознавания жестов в РЖЯ, а также проанализированы существующие фреймворки для разработки мобильных приложений, из которых был выбран .NET MAUI.

Для реализации нейросетевой модели был использован набор данных Slovo, содержащий 20040 видеозаписей для 1000 жестов, и приведено описание различных операций по их предобработке, включая изменение разрешения, интерполяцию, добавление границ и нормализацию.

Для достижения наилучших результатов в качестве архитектуры были рассмотрены модели с использованием трансформеров и выбрана версия с наилучшими заявленными результатами – Multiscale Vision Transformer v2. Помимо этого был выбран набор инструментов для обучения видео MMAction2 и приведены различные эксперименты по подбору гиперпараметров.

На основании этих исследований был спроектирован графический интерфейс мобильного приложения для платформы Android и реализован необходимый функционал, который в режиме реального времени позволяет осуществлять распознавание русского жестового языка. По результатам тестирования оно безошибочно выполняет необходимые действия и в дальнейшем может быть модифицировано с целью улучшения качества распознавания, скорости обработки данных и переноса на другие операционные системы.

ЛИТЕРАТУРА

1. Глобальный веб-сайт Всемирной организации здравоохранения. [Электронный ресурс] URL: https://www.who.int/ru/news-room/fact-sheets/detail/deafness-and-hearing-loss (дата обращения: 26.02.2024 г.).

2. Буркова С.И., Киммельман В.И., Филимонова Е.В., Кюсева М.В., Варинова О.А., Заварицкий Д.А., Приставко К.В., Кадыргулова Р.С. Введение в лингвистику жестовых языков. Русский жестовый язык. // НГТУ, 2021. – 356 с.

3. Рюмин Д.А., Кагиров И.А., Аксенов А.А., Карпов А.А. Аналитический обзор моделей и методов автоматического распознавания жестов и жестовых языков. // Информационно-управляющие системы, 2021. – №. 6 (115). – С. 10–20.

4. Starmans M., Voort van der S., Castillo Tovar J., Veenland J., Klein S., Niessen W. Radiomics: Data Mining Using Quantitative Medical Image Features. // Handbook of Medical Image Computing and Computer Assisted Intervention, 2020. – 429–456 pp.

5. Rajalakshmi E., Elakkiya R., Subramaniyaswamy V., Prikhodko A.L., Grif M., Bakaev M., Kotecha K., Gabralla L.A., Abraham A. Multi-Semantic Discriminative Feature Learning for Sign Gesture Recognition Using Hybrid Deep Neural Architecture. // IEEE, 2023. – Vol. 11. – 2226–2238 pp.

6. Miah A.S.M., Hasan M.A., Shin J., Okuyama Y., Tomioka Y. Multi-stage Spatial Attention-Based Neural Network for Hand Gesture Recognition. // Computers, 2023. – Vol. 12. – no. 13. – 20–31 pp.

7. Noreen I., Hamid M., Akram U., Malik S., Saleem M. Hand Pose Recognition Using Parallel Multi Stream CNN. // Sensors, 2021. – Vol. 21. – no. 24. – 69–84 pp.

8. Гриф М. Г., Элаккия Р., Приходько А.Л., Бакаев М.А., Раджалакшми Е. Распознавание русского и индийского жестовых языков на основе машинного обучения. // Системы анализа и обработки данных, 2021. – № 3. – С. 53–74.

9. Pansare J.R., Gawande S.H., Ingle M. Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background. // Journal of Signal and Information Processing, 2012. – Vol 3. – no 3. – 364–367 pp.

10. Pugeault N., Bowden R. Spelling It Out: Real-Time ASL Fingerspelling Recognition. // Proceedings of the 1st IEEE Workshop on Consumer Depth Cameras for Computer Vision, 2011. – 1114–1119 pp.

11. Веб-сайт проекта «Аватар». [Электронный ресурс] URL: https://adaptis.pro/ (дата обращения: 26.02.2024 г.).

12. Fronteddua G., Porcuab S., Florisab A., Atzoriab L. A dynamic hand gesture recognition dataset for human-computer interfaces. // Computer Networks, 2022. – Vol 5. – 1–4 pp.

13. Elakkiyaa R., Selvamanib K., Kanimozhic S., Velumadhavac R., Kannand A. Intelligent System for Human Computer Interface Using Hand Gesture Recognition. // Procedia Engineering, 2012. – Vol. 38. – 3180–3191 pp.

14. Kotlin Multiplatform. [Электронный ресурс] URL: https://developer.android.com/kotlin/multiplatform (дата обращения: 13.05.2024 г.).

15. Windmill E. Flutter in Action. // Manning Publications, 2020. – P. 368.

16. Apache Cordova. [Электронный ресурс] URL: https://cordova.apache.org (дата обращения: 13.05.2024 г.).

17. .NET Multi-platform App UI. [Электронный ресурс] URL: https://dotnet.microsoft.com/en-us/apps/maui (дата обращения: 13.05.2024 г.).

18. Kapitanov A., Kvanchiani K., Nagaev A., Petrova E. Slovo: Russian Sign Language Dataset. // International Conference on Computer Vision Systems, 2023. – 63–73 pp.

19. МакХью Ш. Интерполяция цифрового изображения. Учебник цифровой фотографии. [Электронный ресурс] URL:

https://www.cambridgeincolour.com/ru/tutorials-ru/image-interpolation.htm (дата обращения: 02.04.2024 г.).

20. Dosovitskiy A., Beyer L., Kolesnikov A., Weissenborn D., Zhai X., Unterthiner T., Dehghani M., Minderer M., Heigold G., Gelly S., Uszkoreit J., Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. [Электронный ресурс] // arXiv.org. 2020. Дата обновления: 03.06.2021 г. URL: https://arxiv.org/abs/2010.11929 (дата обращения: 02.04.2024 г.).

21. Fan H., Xiong B., Mangalam K., Li Y., Yan Z., Malik J., Feichtenhofer C. Multiscale vision transformers. // Proceedings of the IEEE/CVF international conference on computer vision, 2021. – 6824–6835 pp.

22. Li Y., Wu C., Fan H., Mangalam K., Xiong B., Malik J., Feichtenhofer C. Mvitv2: Improved multiscale vision transformers for classification and detection. // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022. – 4804–4814 pp.

23. Open-source toolbox for video understanding MMAction2. [Электронный ресурс] URL: https://github.com/open-mmlab/mmaction2 (дата обращения: 17.03.2024 г.).

24. Михайличенко А.А. Аналитический обзор методов оценки качества алгоритмов классификации в задачах машинного обучения. // Вестник Адыгейского государственного университета. Серия 4: Естественно-математические и технические науки, 2022. – №. 4 (311). – С. 52–59.

25. Официальный веб-сайт платформы EmguCV. [Электронный ресурс] URL: https://www.emgu.com/wiki/index.php/Main_Page (дата обращения: 18.03.2024 г.).

26. Shridhar A., Tomson P., Innes M. Interoperating Deep Learning models with ONNX. Jl. // Proceedings of the JuliaCon Conferences, – 2020. – Т.1. – №. 1. – P. 59.

ПРИЛОЖЕНИЕ. Обучение модели распознавания жестового языка

Листинг 1 – Код обучения модели для распознавания жестового языка

```
#Установка необходимых пакетов
!pip install torch==1.12.0 torchvision --extra-index-url
https://download.pytorch.org/whl/cu113
!pip install -U openmim
!mim install mmengine
!mim install 'mmcv >= 2.0.0, <2.2.0'
! git clone https://github.com/open-mmlab/mmaction2.git
%cd mmaction2
! pip install -v -e .
!pip install timm

#Импортирование необходимых библиотек
import os
import cv2
import pandas as pd
from tqdm import tqdm
from glob import glob
import matplotlib.pyplot as plt
import torch
import warnings
warnings.filterwarnings('ignore')

#Установка путей до рабочей директории и набора данных
DATA_DIR = '/content/slovo'
TRAIN_DIR = os.path.join(DATA_DIR, 'train')
TEST_DIR = os.path.join(DATA_DIR, 'test')
ANNOTATIONS_DIR = os.path.join(DATA_DIR, 'annotations')
ann = pd.read_csv(os.path.join(DATA_DIR, 'annotations.csv'), sep='\t')
train_files = sorted(glob(os.path.join(TRAIN_DIR, '*')))
test_files = sorted(glob(os.path.join(TEST_DIR, '*')))
NUM_CLASSES = len(ann['text'].unique()) # Including "no-action" class
classes = {label: label_id for label, label_id in zip(ann['text'].unique(),
range(NUM_CLASSES))}
ann_train = []
ann_test = []

#Формирование аннотированных файлов для обучения модели
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)
%%writefile mvit-slovo.py

# Файл-конфиг для обучения модели
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')
        )

# Logging settings
default_scope = 'mmaction'
default_hooks = dict(
    runtime_info=dict(type='RuntimeInfoHook'),
    timer=dict(type='IterTimerHook'),
    logger=dict(type='LoggerHook', interval=600, ignore_last=False),
    param_scheduler=dict(type='ParamSchedulerHook'),
    checkpoint=dict(
        type='CheckpointHook', interval=1, 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='TensorboardVisBackend'),
dict(type='LocalVisBackend')]
visualizer = dict(
    type='ActionVisualizer',
    vis_backends=vis_backends,
    name='visualizer',
    save_dir='/content/visualization_dir'
    )
log_level = 'INFO'
load_from = None
resume = False

# Specify dataset paths
dataset_type = 'VideoDataset'
data_root = '/content/slovo/train'
data_root_val = '/content/slovo/test'
ann_file_train = '/content/mmaction2/ann_train.txt'
ann_file_val = '/content/mmaction2/ann_test.txt'
ann_file_test = '/content/mmaction2/ann_test.txt'

train_pipeline = [
    dict(type='DecordInit', io_backend='disk'),
    dict(
        type='SampleFrames',
```

```
        clip_len=16,
        frame_interval=4,
        num_clips=1,
        out_of_bound_opt='repeat_last'),
    dict(type='DecordDecode'),
    dict(type='Resize', scale=(224, 224)),
    dict(type='Flip', flip_ratio=0.5, direction='horizontal'),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='PackActionInputs')
]
val_pipeline = [
    dict(type='DecordInit', io_backend='disk'),
    dict(
        type='SampleFrames',
        clip_len=16,
        frame_interval=4,
        num_clips=1,
        test_mode=True,
        out_of_bound_opt='repeat_last'),
    dict(type='DecordDecode'),
    dict(type='Resize', scale=(224, 224)),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='PackActionInputs')
]
test_pipeline = [
    dict(type='DecordInit', io_backend='disk'),
    dict(
        type='SampleFrames',
        clip_len=16,
        frame_interval=4,
        num_clips=2,
        test_mode=True,
        out_of_bound_opt='repeat_last'),
    dict(type='DecordDecode'),
    dict(type='Resize', scale=(224, 224)),
    dict(type='FormatShape', input_format='NCTHW'),
    dict(type='PackActionInputs')
]

train_dataloader = dict(
    batch_size=2,
    num_workers=2,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=True),
    dataset=dict(
        type='VideoDataset',
        ann_file=ann_file_train,
        data_prefix=dict(video=data_root),
        pipeline=train_pipeline))
val_dataloader = dict(
    batch_size=2,
    num_workers=2,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=dict(
        type='VideoDataset',
        ann_file=ann_file_val,
        data_prefix=dict(video=data_root_val),
        pipeline=val_pipeline,
        test_mode=True))
    batch_size=1,
```

```
    num_workers=2,
    persistent_workers=True,
    sampler=dict(type='DefaultSampler', shuffle=False),
    dataset=dict(
        type='VideoDataset',
        ann_file=ann_file_test,
        data_prefix=dict(video=data_root_val),
        pipeline=test_pipeline,
        test_mode=True))

# Training settigns
val_evaluator = dict(type='AccMetric')
test_evaluator = dict(type='AccMetric')
train_cfg = dict(
    type='EpochBasedTrainLoop', max_epochs=25, val_begin=1, val_interval=1)
val_cfg = dict(type='ValLoop')
test_cfg = dict(type='TestLoop')
optim_wrapper = dict(
    optimizer=dict(
        type='Adam', lr=0.0001, weight_decay=0.0001),
    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=25,
        convert_to_iter_based=True) ,
      dict(
         type='MultiStepLR',
         by_epoch=True,
         begin=0,
         end=30,
         milestones=[8, 16, 24],
        gamma=0.1)
]
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)

#Запуск обучения
! python tools/train.py ./mvit-slovo.py
```