МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»
(Самарский университет)

Институт информатики и кибернетики
Кафедра технической кибернетики

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Применение алгоритмов машинного обучения для задач захвата движения человека на видеоизображении

по программе бакалавриата по направлению подготовки
01.03.02 Прикладная математика и информатика,
профиль «Компьютерные науки»

Обучающийся _____ А.А. Сорока
(*подпись*)
Научный руководитель ВКР,
доцент, к.ф.-м.н. _____ Д.А. Савельев
(*подпись*)
Нормоконтролёр _____ С.В. Суханов
(*подпись*)

Самара 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики
Кафедра технической кибернетики

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.В. Куприянов
«____»_____ 20____ г.

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

обучающемуся группы 6409-010302D ***Сорока Александру Александровичу***

Тема ВКР: ***Применение алгоритмов машинного обучения для задач захвата движения человека на видеоизображении***

утверждена приказом по университету от « 24 » 04 2024 г. № 223-Т .

Исходные данные: *алгоритмы машинного обучения, ключевые точки, архитектуры свёрточных нейронных сетей, оптимизаторы и функции потерь, методы преобразования ключевых точек в трёхмерные координаты*.

Перечень вопросов, подлежащих разработке в ВКР:

- *формулировка задачи захвата движения и определение её цели, включая описание методов с маркерами и без маркеров;*
- *исследование алгоритмов машинного обучения и их сравнительный анализ для применения в задачах захвата движения;*
- *реализация алгоритма для определения двумерных ключевых точек на основе выбранных архитектур моделей, оптимизаторов и методов обучения;*

– *обзор и применение методов преобразования двумерных ключевых точек в трёхмерные координаты;*
– *проведение экспериментов и анализ полученных результатов для моделей с различными архитектурами, оценка точности и полноты моделей на основе ключевых метрик.*

Руководитель ВКР
доцент

_____ Д.А. Савельев
(*подпись*)
«_15_» ___02___ 2022 г.

Задание принял к исполнению

_____ А.А. Сорока
(*подпись*)
«_15_» ___02___ 2022 г.

РЕФЕРАТ

Выпускная квалификационная работа бакалавра: 56 с., 16 рисунков, 2 таблицы, 23 источников, 1 приложение

Презентация: 12 слайдов Microsoft PowerPoint.

НЕЙРОННЫЕ СЕТИ, БЕЗМАРКЕРНЫЙ ЗАХВАТ ДВИЖЕНИЯ, ОБРАБОТКА ИЗОБРАЖЕНИЙ, АНИМАЦИЯ, НЕЙРОСЕТЕВОЙ АНАЛИЗ ДАННЫХ

Работа посвящена исследованию применимости алгоритмов машинного обучения для решения задач захвата движения человека на видеоизображении.

Часть работы посвящена обзору алгоритмов машинного обучения для задачи захвата движения.

В другой части рассматривается программная реализация алгоритма определения двумерных ключевых точек.

Также, в работе рассматриваются методы преобразования двумерных ключевых точек в трехмерные, а также анализируются полученные результаты.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные исследования в области компьютерного зрения и машинного обучения актуальны благодаря широкому спектру применения этих технологий в различных индустриях, включая автоматизированное видеонаблюдение, интерактивные системы, спортивный анализ и реабилитацию после травм. Особенно значимыми становятся методы захвата и анализа движений человека, которые позволяют улучшить интерфейсы человеко-машинного взаимодействия и повысить точность биомеханических исследований.

Для анализа движений человека на видео часто используются алгоритмы машинного обучения, которые могут автоматически распознавать и классифицировать различные типы движений из видеоданных. Основным преимуществом этих алгоритмов является способность обучаться на больших объемах данных и адаптироваться к новым, ранее неизвестным условиям, что делает их идеально подходящими для задач компьютерного зрения.

В данной работе рассматривается задача разработки и апробации программного средства, основанного на методах машинного обучения, для захвата и анализа движения человека в видеопотоке.

Данная работа содержит пять разделов.

В первом разделе рассматривается постановка задачи, вводятся базовые определения, исследуется применение систем захвата движения в реальных задачах и ставится цель работы.

Во втором разделе рассматриваются алгоритмы машинного обучения для решения задачи захвата движения человека. Приводится обоснование выбора определенного алгоритма для реализации собственного решения.

В третьем разделе проводится описание программной реализации алгоритма определения двумерных ключевых точек. Подробно описывается модель сети, а также другие компоненты, использующиеся при обучении.

В четвертом разделе приведены методы преобразования двумерных ключевых точек в трехмерные.

И, наконец, в пятом разделе приведены результаты работы собственной модели определения ключевых точек. Также, провизуализирован результат совместной работы модели MiDaS с собственной моделью определения двумерных ключевых точек. Проведен анализ полученных результатов.

1 Постановка задачи и цель работы

1.1 Описание задачи захвата движения в общем смысле

Захват движения (Motion Capture, MoCap) — это технология, которая позволяет записывать движение объектов, в особенности человеческого тела, и применять эти данные для анимации моделей в 3D-пространстве.

При анимации видеоигр, фильмов и создании контента для виртуальной реальности, технологии захвата движения (рисунок 1) играют ключевую роль, добавляя реализм и интерактивность. В спортивных науках, анализ видеозаписей тренировок и соревнований позволяет тренерам и спортсменам улучшать технику и стратегии. В медицине, особенно в реабилитации, захват движений используется для оценки и коррекции походки пациентов, что критически важно для успешного восстановления.

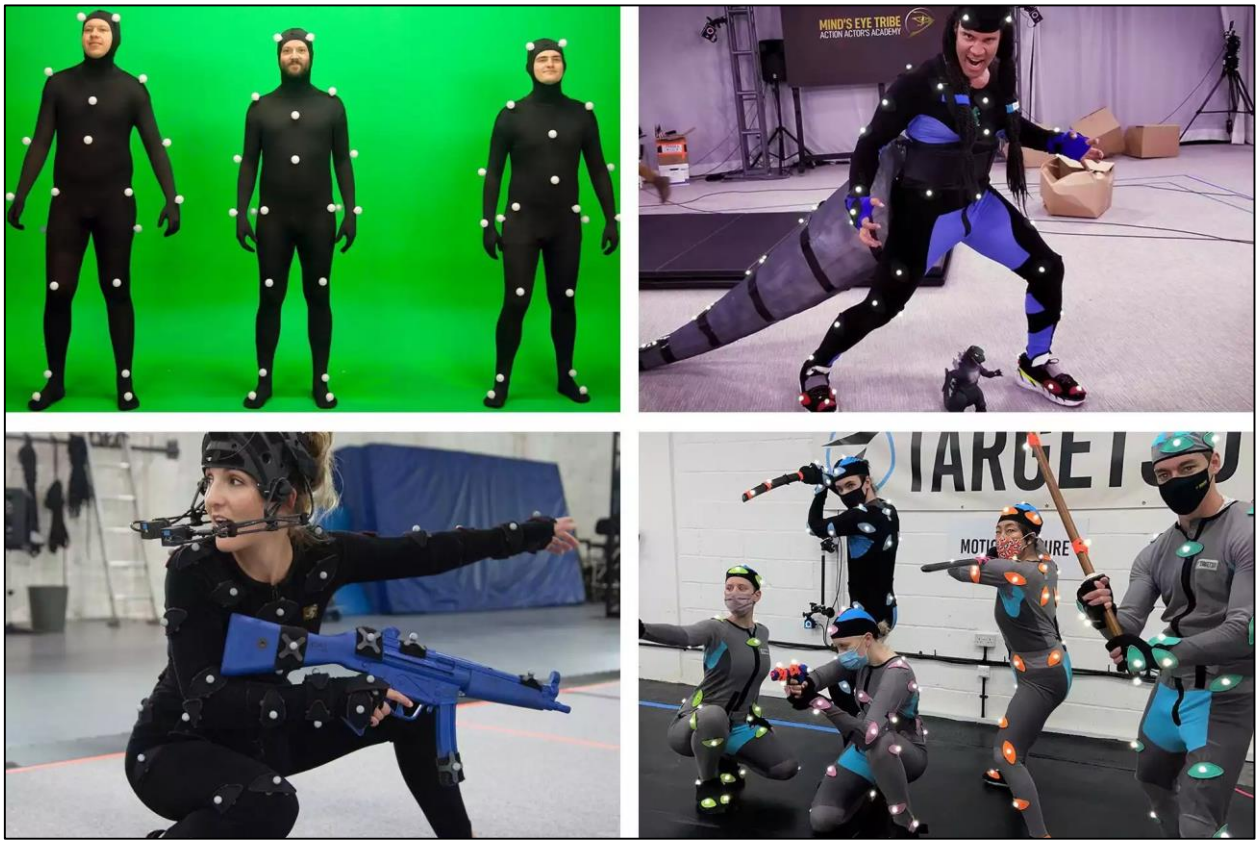

Рисунок 1 – Технология Motion Capture [1]

1.1.1 Захват движений с помощью специальных маркеров

Маркерный захват движения — это методика, при которой на теле актера или спортсмена закрепляются специальные маркеры, которые отслеживаются с помощью камер и других датчиков. Эти маркеры могут быть различных типов: отражающие, светящиеся или магнитные. В зависимости от системы захвата, маркеры отслеживаются камерами, которые расположены вокруг зоны захвата, и таким образом записывают точное положение маркеров в пространстве.

На первом этапе на тело испытуемого устанавливаются маркеры. Количество и расположение маркеров могут значительно варьироваться в зависимости от специфики задачи и системы захвата.

Прежде чем начать сессию захвата, необходимо калибровать систему. Это включает настройку камер или датчиков для оптимального захвата движения маркеров.

В процессе записи камеры считывают положение каждого маркера в пространстве. Данные с камер агрегируются и обрабатываются для создания трехмерной модели движения.

После записи сырые данные обрабатываются для построения анимационной модели. Это включает в себя удаление шумов, интерполяцию пропущенных данных и преобразование данных маркеров в координаты скелетной анимации.

Преимуществами захвата движения с помощью физических маркеров являются:

1) высокая точность и детализация захвата;

2) эффективность при сложных движениях, таких как акробатика или детальные мимические движения.

Но, такая система обладает и недостатками:

1) необходимость использования специального оборудования и пространства для захвата;

2) дискомфорт испытуемых из-за необходимости носить маркеры и ограничений, связанных с ними;

3) возможные ошибки из-за перекрытия маркеров или их потери в процессе движения.

1.1.2 Безмаркерный захват движения

Безмаркерный захват движения — это более современная и гибкая альтернатива, которая позволяет анализировать движения без прямого контакта с объектом захвата. Этот метод использует алгоритмы машинного зрения для анализа видеоизображений и выделения ключевых точек тела человека без физических маркеров.

В отличие от маркерных систем, безмаркерные системы требуют только видеозапись с одной или нескольких камер. С помощью алгоритмов компьютерного зрения происходит распознавание формы тела, определение положения суставов и ключевых точек. На основе полученных данных строится модель движения, которая может быть использована для анимации или анализа.

Преимущества:

1) удобство и доступность использования;

2) возможность анализа естественных движений в реальных условиях без искажений, вызванных наличием маркеров или специального снаряжения.

Недостатки:

1) меньшая точность и детализация по сравнению с маркерными системами;

2) зависимость от качества и условий съемки.

1.2 Задача захвата движения с точки зрения математики

Задача определения ключевых точек человеческого тела на видео может быть сформулирована как задача компьютерного зрения, в которой необходимо определить координаты предварительно определенных анатомических меток, таких как суставы, на изображении или в видеоряде. Математически это можно представить следующим образом.

Пусть I обозначает кадр из видеопотока, а $P = (p_1, p_2, \dots, p_n)$ – набор ключевых точек, которые необходимо определить. Каждая точка p_i описывается своими координатами на изображении (x_i, y_i). Задача алгоритма машинного обучения – максимизировать [2] вероятность правильного определения этих координат, основываясь на обучающем наборе данных, содержащем аннотированные изображения.

Процесс обучения модели заключается в минимизации функции потерь, которая оценивает разницу между предсказанными алгоритмом координатами и истинными координатами точек на обучающих данных. Одним из популярных выборов для этой функции потерь является сумма квадратов разностей между предсказанными и истинными значениями координат:

$$L = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2, \quad (1)$$

где $\hat{x}_i$ и $\hat{y}_i$ – предсказанные координаты точек;

x_i и y_i – истинные координаты.

1.3 Цель работы

Цель данной работы — исследовать применимость модели машинного обучения, способную с высокой точностью определять ключевые точки человеческого тела на изображениях и видео. Основные задачи включают улучшение точности захвата движения при различных условиях освещения, оптимизацию алгоритмов для работы в реальном времени на стандартном оборудовании и минимизацию нужды в ручной корректировке и аннотировании больших объемов данных.

Конечная цель исследования — не только создать эффективную техническую систему, но и продемонстрировать, как такие технологии могут быть интегрированы в реальные прикладные области, предоставляя значимую пользу в медицинских, спортивных и развлекательных приложениях.

2 Алгоритмы машинного обучения для захвата движения

2.1 Свёрточные нейронные сети

Convolutional Neural Networks (CNN) являются основным инструментом в современном компьютерном зрении и имеют значительное применение в анализе визуальных данных. В контексте захвата движения, CNN применяются для автоматического распознавания и отслеживания человеческих ключевых точек в последовательности видеокадров.

Принцип работы CNN заключается в автоматическом извлечении признаков из входных изображений посредством операций свёртки, что делает их идеальными для задач, требующих обработку больших и сложных визуальных данных без необходимости ручного задания характеристик. CNN состоят из нескольких типов слоёв (рисунок 2) [3]:

1) свёрточные слои (convolutional layers);
2) слои пулинга (pooling layers);
3) функции активации;
4) полносвязные слои (fully connected layers).

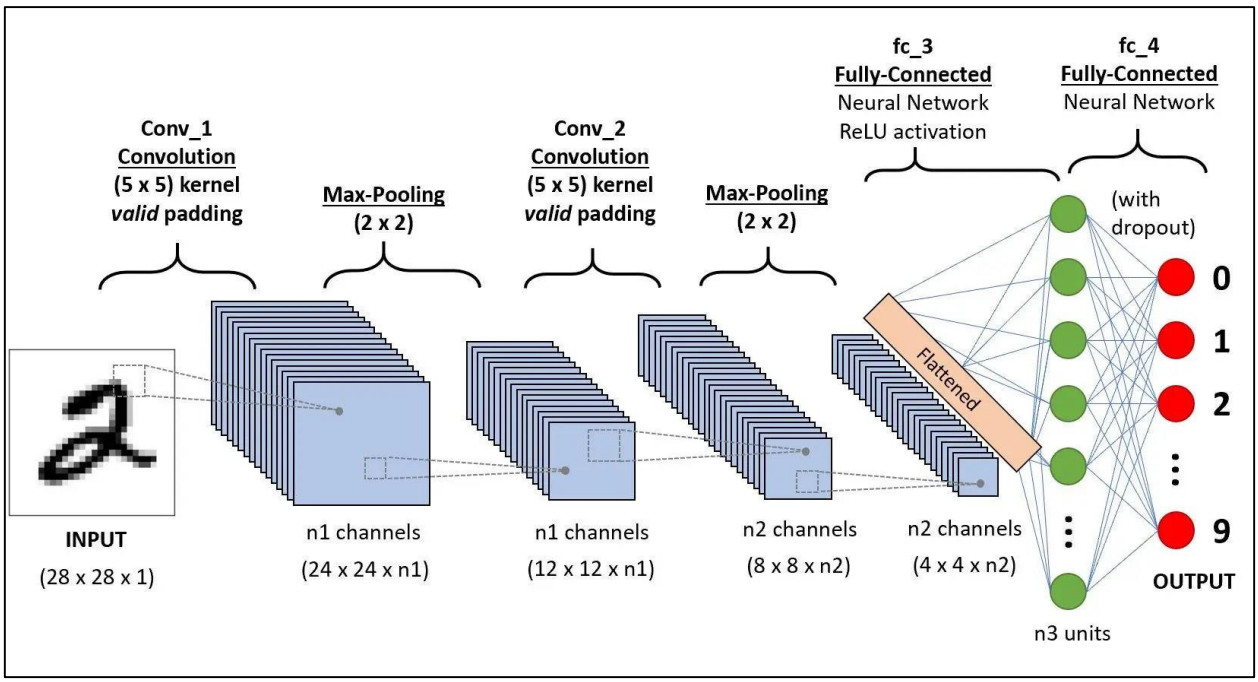

Рисунок 2 – Пример архитектуры свёрточной нейронной сети [3]

2.1.1 Свёрточные слои

Свёрточные слои являются фундаментальной составляющей большинства архитектур глубокого обучения. Эти слои используют математическую операцию свертки для выделения важных признаков из входных данных.

Свёрточный слой использует набор фильтров или ядер, которые перемещаются по всему входному изображению, применяя математическую операцию свертки. Эта операция позволяет извлекать изображения из входных данных, создавая карты признаков, которые представляют собой агрегированную информацию о наличии определённых характеристик в различных регионах изображения [4].

В качестве входных данных выступают многоканальные матрицы данных, где каждый канал соответствует определенному цветовому каналу (например, RGB). Свертка применяется отдельно к каждому каналу, а результаты суммируются для получения итоговой карты признаков для каждого фильтра.

Математическая формулировка свёртки [5]: пусть I – входное изображение с размерами $H \times W \times D$, где D – количество каналов (например, для RGB $D = 3$). Фильтр свертки K имеет размеры $F \times F \times D$. Свертка вычисляется, как:

$$O(i,j) = \sum_{c=1}^{D} \sum_{m=1}^{F} \sum_{n=1}^{F} I(i+m-1, j+n-1, c) \cdot K(m,n,c), \qquad (2)$$

где $i \in \{1,2,\dots,H-F+1\}$;

$j \in \{1,2,\dots,W-F+1\}$;

$I(i+m-1, j+n-1, c)$ – значение пикселя входного изображения I с координатами $(i+m-1, j+n-1)$ в канале c;

$K(m,n,c)$ – значение ядра фильтра K с координатами (m,n) в канале c;

$O(i,j)$ – значения пикселя в результирующем изображении.

2.1.2 Слои пулинга

Слои пулинга, или слои субдискретизации, представляют собой ключевой компонент сверточных нейронных сетей, используемых для уменьшения

размерности пространственных данных. Эти слои следуют непосредственно за сверточными слоями и играют важную роль в обеспечении инвариантности сети к масштабированию и другим искажениям изображения.

Суть пулинга заключается в применении операции уменьшения размерности к отдельным сегментам карт признаков, полученных после сверточных слоев. Это достигается путем применения агрегирующей функции, такой как максимум или среднее, к каждому такому сегменту.

Самый распространенный тип пулинга — максимальный пулинг [6]. В его рамках из каждого рассматриваемого подмножества входных данных выбирается максимальное значение. Например, если применять максимальный пулинг с размером окна 2x2 и шагом 2 к матрице признаков, каждое неперекрывающееся подокно 2x2 в этой матрице будет уменьшено до одного значения, равного максимальному из четырех.

Математически операция максимального пулинга для матрицы признаков A с размером окна $f \times f$ и шагом s можем быть описана следующим образом:

1) разделить матрицу A на неперекрывающиеся подматрицы размером $f \times f$;

2) для каждой подматрицы A_{sub} найти максимальное значение:

$$A'_{i,j} = \max(A_{sub}), \tag{3}$$

где i и j – индексы в результирующей матрице признаков A', которая будет иметь уменьшенные пространственные размеры.

Представим, что у нас есть матрица признаков следующего вида:

$$A = \begin{bmatrix} 1 & 3 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}. \tag{4}$$

Применение операции максимального пулинга с размером окна 2×2 и шагом 2 приведет к следующей матрице:

$$A' = \begin{bmatrix} 6 & 8 \\ 14 & 16 \end{bmatrix}. \tag{5}$$

Слои пулинга помогают уменьшить количество параметров и вычислений в сети, что способствует борьбе с переобучением и ускоряет обучение. Кроме того, благодаря уменьшению размерности, нейросеть становится менее чувствительной к местоположению объектов во входном изображении, что улучшает её способность к обобщению [4].

2.1.3 Функции активации

Функции активации в нейронных сетях — это математические уравнения, определяющие выходной сигнал нейрона по сумме входных сигналов. Эти функции необходимы для введения нелинейности в модель. Это позволяет нейронной сети обучаться и выполнять более сложные задачи, чем просто линейная регрессия или классификация. Без нелинейности она все равно оставалась бы линейной моделью, даже при достаточно сложной архитектуре, а это значительно снижает её способность к обучению и аппроксимации функций.

Функция активации применяется к каждому нейрону в сети и определяет активность нейрона при данном входе. Рассмотрим этот процесс на примере функций активации Rectified Linear Unit (ReLU).

Формула для ReLU:

$$f(x) = \max(0, x). \tag{6}$$

Эта функция активации принимает один вход x и выдает x, если x положительное, и 0, если x отрицательное.

ReLU используется очень часто, так как она проста в вычислении и помогает уменьшить вероятность исчезающего градиента, что часто встречается при использовании таких функций активации, как сигмоид или гиперболический тангенс. Важно отметить, что ReLU активирует нейроны только тогда, когда на входе есть активация. Это делает нейронные сети разреженными, увеличивая тем самым эффективность и уменьшая вычислительные затраты.

2.1.4 Полносвязные слои

Полносвязные слои (dense layers) – это основные строительные блоки в архитектуре нейронных сетей. В этих слоях каждый нейрон предыдущего слоя соединён с каждым нейроном следующего слоя, что создаёт полную связность

между слоями. Это позволяет сети интегрировать информацию, полученную на предыдущих этапах, для выполнения конкретных задач, таких как классификация или регрессия.

В контексте свёрточных нейронных сетей (CNN), полносвязные слои обычно располагаются в конце архитектуры после последовательности свёрточных и пулинг слоев. Основная функция этих слоев в CNN — синтезировать данные, извлечённые из предыдущих слоев, в предсказания, которые могут быть использованы для классификации или других типов вывода.

Каждый нейрон в полносвязном слое получает входы от всех активаций предыдущего слоя, умножает их на соответствующие веса, добавляет смещение (bias) и пропускает через функцию активации для получения выходного значения. Это можно математически представить следующим образом [7]:

$$y = f(Wx + b), \tag{7}$$

где x – вектор входных активаций из предыдущего слоя;

W – матрица весов;

b – вектор смещений;

f – функция активации;

y – вектор выходных активаций.

Полносвязные слои широко используются во многих областях машинного обучения и глубокого обучения.

В классификации изображений после извлечения признаков через свёрточные и пулинг слои полносвязные слои используются для классификации изображений на основе этих признаков.

В задачах регрессии, где требуется предсказать непрерывные значения, такие как цены на дома или температура, полносвязные слои могут обрабатывать признаки для предсказания этих значений.

В сложных задачах, где имеется множество переменных, например, в распознавании речи или машинном переводе, полносвязные слои помогают усилить признаки, интегрируя и абстрагируя информацию на высоком уровне.

2.2 Рекуррентные нейронные сети

Рекуррентные нейронные сети (рисунок 3) представляют собой класс нейросетевых моделей, специализированных на обработке последовательных данных, благодаря своей уникальной способности передавать информацию через временные шаги. Это достигается за счёт внутреннего состояния (или «памяти»), которое позволяет сети удерживать информацию о предыдущих данных в последовательности, делая RNN особенно подходящими для задач, где контекст имеет решающее значение, например, в анализе временных рядов, обработке естественного языка, синтезе речи и других.

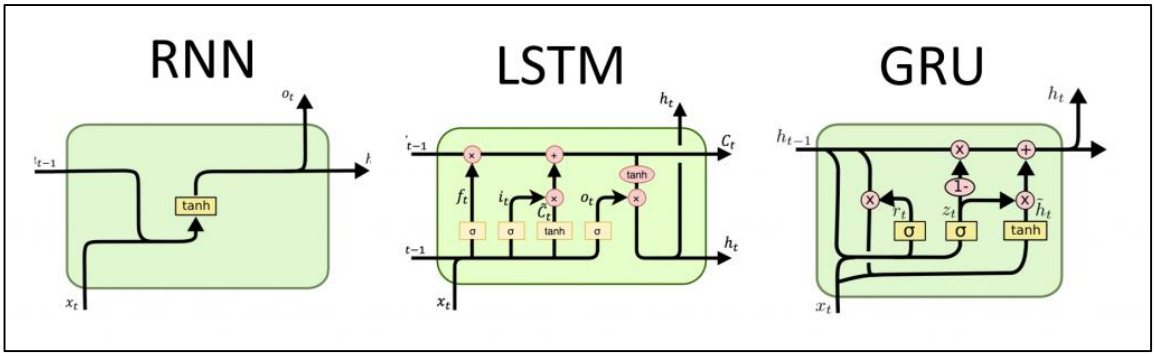

Рисунок 3 – Примерная архитектура рекуррентных нейронных сетей [8]

Одним из расширений RNN являются сети с долгой краткосрочной памятью (LSTM). RNN и их вариации, включая LSTM и GRU, находят применение в областях, где требуется анализ последовательностей данных. Они могут анализировать исторические данные временных рядов, такие как акции или погодные данные, для предсказания будущих значений.

В задачах NLP, таких как машинный перевод или распознавание речи, RNN могут улавливать семантические и синтаксические зависимости в тексте. Используя последовательности нот и аккордов, RNN способны генерировать музыкальные произведения. Рекуррентные нейронные сети используются для предсказания действий и событий в видеопотоках. Это может быть применено в системах видеонаблюдения или интерактивных мультимедийных системах.

2.3 Сравнение CNN и RNN/LSTM для захвата движения

Эффективность применения CNN и RNN/LSTM в этой задаче варьируется в зависимости от конкретных требований и условий применения.

Свёрточные нейронные сети подходят для анализа визуальной информации благодаря своей способности эффективно обрабатывать пространственные отношения и текстуры на изображениях. В контексте захвата движения, CNN могут быть использованы для точного определения ключевых точек и сегментов тела человека в различных позах и условиях освещения. Благодаря своим свёрточным слоям, CNN способны автоматически выделять важные признаки из видеоданных, что значительно упрощает процесс обучения и уменьшает необходимость в ручном извлечении признаков и их предварительной обработке.

Однако, основным недостатком CNN является их неспособность учитывать временные зависимости в данных. В контексте захвата движения, где важно анализировать последовательность движений и каждое последующее состояние может зависеть от предыдущих, этот недостаток может существенно ограничивать применимость CNN.

В отличие от CNN, рекуррентные нейронные сети (RNN) и их усовершенствованный вариант — LSTM (Long Short-Term Memory) — разработаны специально для работы с последовательными данными. Эти сети способны обрабатывать информацию о предыдущих состояниях, что делает их идеальными для задач, где необходимо учитывать контекст, таких как анализ сложных движений и жестов. LSTM улучшает возможности базовых RNN за счет механизмов забывания и сохранения информации, это в свою очередь позволяет эффективно управлять потоком информации.

RNN и особенно LSTM требуют значительных вычислительных ресурсов для обучения, что может стать проблемой при работе с большими объемами данных. Кроме того, настройка и оптимизация этих сетей может быть сложной из-за большого количества параметров и настроек. В следствие этого увеличивается сложность моделей и время, необходимое для их тонкой настройки.

При выборе между CNN и RNN/LSTM для задач захвата движения важно учитывать конкретные требования к задаче. В случае, когда важно пространственное распознавание и высокая точность в определении позы на отдельных кадрах, CNN становится безальтернативным выбором. Это связано с тем, что RNN не предназначены для обработки изображений и не обладают способностью эффективно выявлять пространственные особенности на изображениях.

Однако, если требуется анализ сложных движений, которые зависят от времени и имеют последовательную структуру, тогда использование RNN или LSTM будет более подходящим вариантом. LSTM, например, может быть использован для уточнения позиции ключевых точек, которые не всегда видны на изображении, учитывая контекст временных зависимостей.

В идеальном сценарии комбинирование CNN для пространственного анализа и LSTM для обработки временной последовательности движений может привести к наилучшим результатам. Такой подход позволяет максимально использовать преимущества обеих технологий, обеспечивая точное распознавание поз и анализ сложных движений с учетом временной динамики.

3 Программная реализация алгоритма определения двумерных ключевых точек

3.1 Подготовка датасета

Датасеты для обучения и оценки моделей компьютерного зрения играют критически важную роль в разработке и улучшении алгоритмов, способных анализировать и интерпретировать визуальную информацию. Ключевым аспектом этих датасетов является их способность предоставлять разнообразные данные, которые отражают множество сценариев использования в реальном мире.

Современные датасеты включают изображения с различными уровнями сложности. Это может включать в себя разнообразные фоны, условия освещения и конфигурации объектов. Например, уличные сцены содержат множество перекрывающихся объектов, разные типы движения и разнообразные погодные условия. Датасеты типа ImageNet или COCO включают изображения, которые захватывают эти сложности, помогая моделям учиться на данных, максимально приближенных к реальности.

Для распознавания и локализации объектов важно, чтобы датасеты включали широкий спектр объектов различных форм, размеров и категорий. Например, датасет Pascal VOC включает объекты, такие как велосипеды, машины, люди и животные, каждый из которых имеет уникальные атрибуты и формы. Это важно для обучения моделей распознавать и точно классифицировать объекты в различных контекстах.

Хорошо аннотированные датасеты предоставляют точные и подробные метки, которые необходимы для обучения и оценки моделей. Это включает не только метки классов, но и ограничивающие рамки, аннотации ключевых точек и маски сегментации. Например, датасет COCO [9] предоставляет аннотации для сегментации на уровне пикселей и ключевые точки для анализа человеческой позы, что позволяет использовать его для задач, требующих детального понимания визуального контента.

Разнообразие в условиях съемки, таких как освещение, ракурс и качество изображения, также критически важно. Модели становятся устойчивыми к изменениям во входных данных, которые неизбежны в реальных приложениях. Датасеты, такие как ImageNet [10], включают изображения с различным разрешением и качеством, что помогает улучшить устойчивость и надежность обученных моделей.

Датасеты должны отражать реальные сценарии применения, чтобы обеспечить практическую значимость разработанных моделей. Например, датасеты для автономного вождения, такие как KITTI, предоставляют данные, собранные с датчиков на автомобилях, что помогает в разработке систем, способных работать в реальных дорожных условиях.

В рамках данной работы был использован датасет COCO (Common Objects in Context), который является одним из стандартов в индустрии для задач детекции, сегментации и распознавания ключевых точек. COCO содержит более 330 тысяч изображений с более чем 200 тысячами меток, включающих ограничивающие рамки, сегментационные маски и ключевые точки.

Кроме того, используется аугментации данных и дополнительные техники предварительной обработки для улучшения обучения и обобщающей способности модели. Это включает в себя масштабирование, повороты и изменение цветовых настроек изображений, что позволяет модели лучше адаптироваться к разнообразным условиям и улучшить ее способность к обобщению на новые данные.

3.2 Общий подход и архитектура модели

3.2.1 Backbone сети

Backbone сеть является фундаментальным компонентом, который отвечает за первичное извлечение признаков из входных изображений. В контексте глубокого обучения, backbone — это обычно предварительно обученная сверточная нейронная сеть, которая преобразует исходные изображения в сложный набор признаков или карт признаков. Эти карты признаков служат

основой для всех последующих этапов анализа и обработки, таких как детекция объектов, классификация и определение ключевых точек.

Распространенные архитектуры, используемые в качестве backbone в задачах компьютерного зрения, включают [11]:

– ResNet (Residual Networks): Эта сеть использует так называемые «остаточные блоки», которые помогают обучать очень глубокие нейронные сети. Основное математическое выражение для слоя в ResNet выглядит следующим образом [11]:

$$x_{l+1} = x_l + F(x_l, W_l), \quad (8)$$

где x_l – входной вектор признаков на уровне l;

F – остаточная функция;

W_l – веса.

– VGG (Visual Geometry Group): Простая и мощная архитектура, основанная на повторении блоков, состоящих из сверточных слоев с маленьким размером фильтра (3x3), за которыми следуют слои пулинга. В VGG все слои используют одинаковый шаг и дополнение нулями, что позволяет сохранять пространственные размеры через слои [12].

Feature Pyramid Network (FPN) является дополнением к стандартному backbone и предназначена для улучшения детекции объектов различного размера. FPN создает иерархию признаков на разных уровнях разрешения, что позволяет модели лучше адаптироваться к объектам разных масштабов. В FPN каждый уровень пирамиды создается путем слияния информации из двух источников: нижнего уровня с высоким разрешением и верхнего уровня с более глубокой семантической информацией. Это достигается с помощью операций свертки и апсемплинга.

Таблица 1 - Эффективность оценки ключевых точек на валидационном наборе COCO [11]

Backbone	AP	AP_{50}	AP_{75}	AP_M	AP_L
ResNet-50	70,4	88,6	78,3	67,1	77,2
Res2Net-50	71,5	89,0	79,3	68,2	78,4
ResNet-101	71,4	89,3	79,3	68,1	78,1
Res2Net-101	72,2	89,4	79,8	68,9	79,2
Res2Net-vlb-50	72,2	89,5	79,7	58,5	79,4
Res2Net-vlb-101	73,0	89,5	80,3	69,5	80,0

Таблица 1 представляет сравнение различных архитектур backbone на валидационном наборе данных COCO для оценки эффективности определения ключевых точек. Оценки включают общую точность (AP), точность при IoU пороге 0,5 (AP_{50}) и 0,75 (AP_{75}), а также точность для средних (AP_M) и больших (AP_L) объектов.

ResNet-50 показывает средние результаты с общей точностью 70,4, что ниже, чем у более сложных моделей. Однако, данная модель уже интегрирована в среду PyTorch, что делает её простой в использовании для начальных этапов разработки и экспериментов, несмотря на то, что она не обеспечивает максимально возможной точности.

Res2Net-50 и Res2Net-101 обеспечивают лучшую общую точность, чем их аналоги ResNet, со значениями 71,5 и 72,2 соответственно. Это указывает на то, что более сложные модификации ResNet архитектуры могут лучше справляться с задачами определения ключевых точек благодаря улучшенному извлечению признаков и более глубокой обработке информации.

Res2Net-vlb-50 и Res2Net-vlb-101 показывают наивысшие результаты во всех категориях, с AP достигающим 73,0 для Res2Net-vlb-101. Это подчеркивает преимущества использования очень глубоких сетей с расширенной архитектурой для сложных задач определения ключевых точек.

3.2.2 RoI Pooling

RoI (Region of Interest) Pooling — это техника, используемая в сверточных нейронных сетях для извлечения фиксированного размера признаков из произвольно размерных регионов. Эта операция критически важна в задачах,

где необходимо обработать локальные области изображения, такие как детекция объектов и сегментация. RoI Pooling был популяризирован архитектурами, такими как Fast R-CNN, для улучшения производительности моделей в этих задачах [13].

На рисунке 4 можно посмотреть на результат операции определения RoI.

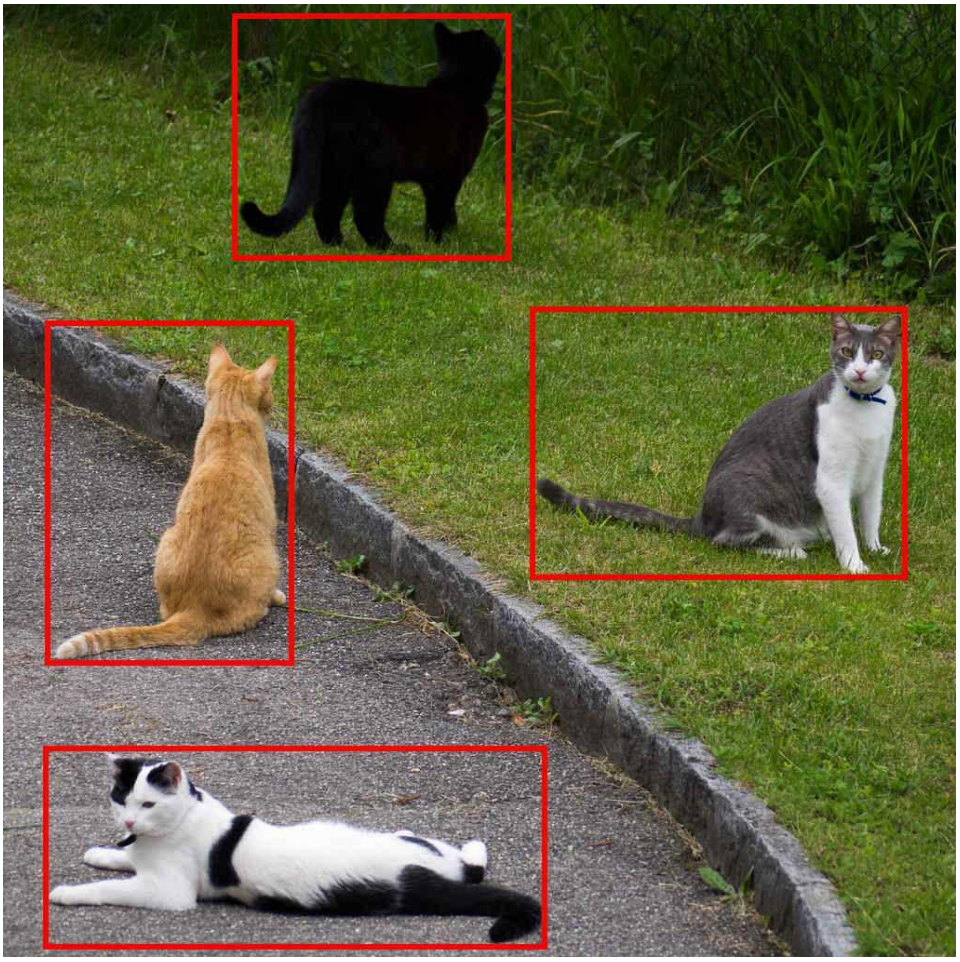

Рисунок 4 – Визуализация результатов работы определения RoI [14]

Рассмотрим входную карту признаков F размером $H \times W \times D$, где H, W и D обозначают высоту, ширину и глубину карты признаков соответственно. Пусть дано N регионов интереса, каждый из которых задан четырьмя координатами: (x_1, y_1, x_2, y_2), где (x_1, y_1) и (x_2, y_2) обозначают верхний левый и нижний правый углы региона на карте признаков.

Для каждого региона R_n RoI Pooling операция выполняется следующим образом [15]:

1) регион R_n преобразуется к фиксированному размеру $H' \times W'$, который задается заранее. Это делается для стандартизации выходных данных, позволяя использовать их на следующих слоях нейронной сети;

2) преобразованный регион делится на $H' \times W'$ ячеек равного размера;

3) в каждой ячейке выполняется операция пулинга для получения одного значения признака. Классификационные и регрессионные слои, блок ключевых точек

После извлечения признаков с помощью RoI Pooling, следующий шаг заключается в использовании классификационных и регрессионных слоев для определения класса объектов и точного позиционирования их ограничивающих рамок. Классификационный слой применяет функцию softmax для оценки вероятности принадлежности каждого предложенного региона к одному из возможных классов объектов:

$$p(c|r) = \frac{e^{s_c}}{\sum_{c'=1}^{C} e^{s_{c'}}}, \quad (10)$$

где $p(c|r)$ – вероятность класса c для региона r;

s_c – сырые оценки классификации для класса c;

а C – общее количество классов.

Регрессионный слой затем адаптирует ограничивающие рамки, предложенные RPN, для того чтобы они как можно точнее обрамляли детектируемый объект. Это достигается путем корректировки четырех параметров рамки: центра x, y, ширины w и высоты h.

После классификации объектов и уточнения их рамок, выполняется задача определения ключевых точек. Каждая детектированная рамка объекта снабжается набором признаков, из которых затем предсказываются ключевые точки. Это достигается с помощью регрессии для каждой точки внутри рамки.

3.3 Оптимизатор Stochastic Gradient Descent (SGD)

Оптимизатор Stochastic Gradient Descent (SGD) — это один из основных методов оптимизации, используемых в обучении нейронных сетей и других вычислительных задачах, связанных с минимизацией функций. SGD модифицирует традиционный метод градиентного спуска путём обновления параметров модели на основе оценки градиента, полученной из случайно выбранного подмножества всего набора данных.

Формула для обновления весов в SGD:

$$w_{t+1} = w_t - \eta \nabla L(w_t), \tag{11}$$

где w_t – параметры модели на шаге t;

η – скорость обучения, положительный коэффициент, определяющий размер шага в пространстве параметров;

$\nabla L(w_t)$ – градиент функции потерь L по параметрам w на шаге t, полученный из текущего батча данных.

В классическом градиентном спуске (Gradient Descent, GD) для обновления параметров модели используется весь набор данных для вычисления градиента функции потерь. Это требует значительных вычислительных ресурсов и времени, особенно при работе с большими данными. В отличие от GD, SGD обновляет параметры, используя только один обучающий пример за раз. Это делает SGD значительно быстрее, особенно в контекстах, где данные имеют высокую избыточность [17].

SGD идеально подходит для задач, где данные поступают последовательно, например, в системах рекомендаций или для персонализированных приложений, где модель должна адаптироваться к новым пользователям или меняющимся предпочтениям пользователя в реальном времени. Это обеспечивается возможностью модели обновляться постоянно, по мере поступления каждого нового данных, без необходимости повторного обучения с нуля.

Помимо достоинств, SGD обладает и рядом недостатков. Проблемы сходимости — один из его основных недостатков. Из-за того, что градиенты вычисляются исходя из одного образца, результаты могут сильно колебаться

между итерациями. Это может привести к тому, что процесс обучения будет менее стабильным и потребует большего количества итераций для достижения сходимости, особенно если выбранный размер шага обучения не оптимален.

Риск застревания в локальных минимумах или седловых точках особенно актуален в контекстах, где функция потерь не является выпуклой. В таких случаях SGD может «застрять», не найдя глобальный минимум ошибки, который оптимален для всех данных. Это связано с тем, что локальные шумы и несовершенства данных могут ввести модель в заблуждение, указывая на неправильное направление обновлений.

3.4 Расписание скорости обучения с использованием MultiStepLR

MultiStepLR — это метод адаптации скорости обучения в процессе тренировки нейронной сети, реализуемый в библиотеке *PyTorch* через модуль *torch.optim.lr_scheduler*. Он дает возможность уменьшать скорость обучения в предопределённые моменты, что может привести к более эффективному и стабильному обучению моделей глубокого обучения.

Скорость обучения — один из гиперпараметров при обучении нейронных сетей. Она определяет величину шага, с которым обновляются веса модели в направлении антиградиента функции потерь. Слишком большая скорость обучения может привести к тому, что обучение будет «перепрыгивать» через минимумы функции потерь. Слишком маленькая скорость замедлит процесс обучения и может привести к застреванию в локальных минимумах.

MultiStepLR изменяет скорость обучения на заданный коэффициент γ в заданные моменты времени, обычно они совпадают с эпохами. Когда текущая эпоха обучения достигает одной из указанных эпох в списке, скорость обучения умножается на коэффициент γ.

Преимуществом использования MultiStepLR является возможность настройки контрольных точек для изменения скорости обучения. Это позволяет адаптировать процесс обучения под конкретные этапы развития модели. Постепенное уменьшение скорости обучения может также улучшить сходи-

мость, позволяя модели более стабильно и глубоко аппроксимировать оптимальные значения весов, что снижает риск осцилляций вокруг минимума функции потерь.

Однако использование MultiStepLR не лишено недостатков. Одной из основных проблем является необходимость ручной настройки моментов изменения скорости и коэффициента уменьшения, что может потребовать значительных усилий и времени на эксперименты, особенно в начале обучения. Эффективность конкретного расписания скорости обучения может также значительно варьироваться в зависимости от специфики задачи и используемых данных, что ограничивает универсальность данного подхода. Кроме того, существует риск застревания в локальных минимумах или на седловых точках, если скорость обучения будет снижена слишком рано или если начальная скорость обучения была установлена слишком низкой, что может препятствовать достижению глобального минимума функции потерь [18].

3.5 Функция потерь и процесс обучения

Функция потерь, используемая в данной модели, включает в себя несколько компонентов, предназначенных для различных аспектов задачи:

Smooth L1 Loss – это вариант функции потерь, который обычно используется для задач регрессии, таких как определение ограничивающих рамок в задачах детекции объектов. Эта функция потерь менее чувствительна к выбросам по сравнению с традиционной L2 потерей, так как имеет способность уменьшать влияние больших ошибок на процесс обучения. Формула Smooth L1 Loss представлена как [19]:

$$l_n = \begin{cases} \frac{0{,}5(x_n - y_n)^2}{\beta}, \text{если } |x_n - y_n| < \beta \\ |x_n - y_n| - 0{,}5\beta, \text{иначе} \end{cases}, \quad (12)$$

где x – разница между предсказанным значением и истинным значением.

Cross-Entropy Loss часто применяется в задачах классификации. Она измеряет различие между двумя вероятностными распределениями: предсказанным и истинным. Для двухклассовой классификации функция потерь вычисляется как:

$$L_c(p, y) = -(y \log(p) + (1 - y) \log(1 - p)), \quad (13)$$

где p – предсказанная вероятность принадлежности объекта к классу;

y – истинная метка класса (0 или 1).

Процесс обучения модели разделяется на несколько ключевых этапов:

1) прямое распространение: на этом этапе входные данные подаются в модель, проходят последовательно через все слои сети, в результате чего модель генерирует предсказание. Каждый слой сети преобразует входные данные согласно своим обученным параметрам и передает результат следующему слою;

2) вычисление потерь: после получения предсказаний на выходе модели рассчитывается значение функции потерь. Это значение представляет собой оценку того, насколько предсказания модели отличаются от истинных значений;

3) обратное распространение: с помощью этого метода производится расчет градиентов функции потерь по всем параметрам модели — от выходных слоев к входным. Этот процесс помогает определить вклад каждого параметра в ошибку и определить, как необходимо изменить параметры, чтобы минимизировать функцию потерь;

4) оптимизация: на основе вычисленных градиентов и с использованием метода SGD параметры модели обновляются таким образом, чтобы функция потерь уменьшалась. Оптимизация является итеративным процессом, и каждое обновление параметров приближает модель к оптимальному решению задачи.

3.6 Мониторинг, валидация и регуляризация

Одним из основных этапов в процессе обучения является валидация модели. Валидация (рисунок 5) позволяет оценить, как модель будет работать в

реальных условиях, используя тестовый набор данных, который не участвовал в обучении. Этот процесс включает в себя периодическую оценку модели на этом наборе данных, чтобы мониторить такие метрики, как точность классификации, точность локализации объектов (в случае задач детекции или сегментации), а также другие метрики, специфичные для конкретной задачи, такие как площадь под ROC-кривой (AUC) или среднее значение точности пересечения по объединению (IoU). Значения IoU после первой эпохи можно увидеть на рисунке 6.

```
Test:  [2200/5000]  eta: 0:04:01  model_time: 0.0635 (0.0615)  evaluator_time: 0.0050 (0.0057)  time: 0.0731  data: 0.0010  max mem: 7163
Test:  [2300/5000]  eta: 0:03:50  model_time: 0.0475 (0.0615)  evaluator_time: 0.0040 (0.0057)  time: 0.0606  data: 0.0009  max mem: 7163
Test:  [2400/5000]  eta: 0:03:39  model_time: 0.0515 (0.0612)  evaluator_time: 0.0040 (0.0056)  time: 0.0635  data: 0.0009  max mem: 7163
Test:  [2500/5000]  eta: 0:03:29  model_time: 0.0490 (0.0610)  evaluator_time: 0.0040 (0.0057)  time: 0.0596  data: 0.0009  max mem: 7163
Test:  [2600/5000]  eta: 0:03:18  model_time: 0.0500 (0.0608)  evaluator_time: 0.0030 (0.0056)  time: 0.0597  data: 0.0011  max mem: 7163
Test:  [2700/5000]  eta: 0:03:08  model_time: 0.0480 (0.0605)  evaluator_time: 0.0040 (0.0056)  time: 0.0578  data: 0.0007  max mem: 7163
Test:  [2800/5000]  eta: 0:02:59  model_time: 0.0655 (0.0607)  evaluator_time: 0.0040 (0.0056)  time: 0.0907  data: 0.0010  max mem: 7163
Test:  [2900/5000]  eta: 0:02:50  model_time: 0.0540 (0.0606)  evaluator_time: 0.0040 (0.0056)  time: 0.0638  data: 0.0009  max mem: 7163
Test:  [3000/5000]  eta: 0:02:41  model_time: 0.0545 (0.0606)  evaluator_time: 0.0040 (0.0056)  time: 0.0674  data: 0.0009  max mem: 7163
Test:  [3100/5000]  eta: 0:02:32  model_time: 0.0480 (0.0605)  evaluator_time: 0.0040 (0.0056)  time: 0.0634  data: 0.0010  max mem: 7163
Test:  [3200/5000]  eta: 0:02:23  model_time: 0.0582 (0.0606)  evaluator_time: 0.0040 (0.0056)  time: 0.0661  data: 0.0009  max mem: 7163
Test:  [3300/5000]  eta: 0:02:15  model_time: 0.0520 (0.0608)  evaluator_time: 0.0040 (0.0056)  time: 0.0838  data: 0.0009  max mem: 7163
Test:  [3400/5000]  eta: 0:02:07  model_time: 0.0455 (0.0607)  evaluator_time: 0.0040 (0.0056)  time: 0.0552  data: 0.0012  max mem: 7163
Test:  [3500/5000]  eta: 0:01:58  model_time: 0.0440 (0.0606)  evaluator_time: 0.0040 (0.0056)  time: 0.0555  data: 0.0010  max mem: 7163
Test:  [3600/5000]  eta: 0:01:49  model_time: 0.0520 (0.0604)  evaluator_time: 0.0040 (0.0056)  time: 0.0699  data: 0.0009  max mem: 7163
Test:  [3700/5000]  eta: 0:01:41  model_time: 0.0505 (0.0604)  evaluator_time: 0.0040 (0.0056)  time: 0.0668  data: 0.0011  max mem: 7163
Test:  [3800/5000]  eta: 0:01:33  model_time: 0.0540 (0.0604)  evaluator_time: 0.0050 (0.0056)  time: 0.0795  data: 0.0011  max mem: 7163
Test:  [3900/5000]  eta: 0:01:25  model_time: 0.0510 (0.0603)  evaluator_time: 0.0040 (0.0056)  time: 0.0572  data: 0.0010  max mem: 7163
Test:  [4000/5000]  eta: 0:01:17  model_time: 0.0520 (0.0602)  evaluator_time: 0.0040 (0.0055)  time: 0.0659  data: 0.0010  max mem: 7163
Test:  [4100/5000]  eta: 0:01:09  model_time: 0.0565 (0.0601)  evaluator_time: 0.0040 (0.0055)  time: 0.0626  data: 0.0009  max mem: 7163
Test:  [4200/5000]  eta: 0:01:01  model_time: 0.0630 (0.0603)  evaluator_time: 0.0040 (0.0055)  time: 0.0881  data: 0.0011  max mem: 7163
Test:  [4300/5000]  eta: 0:00:53  model_time: 0.0540 (0.0604)  evaluator_time: 0.0040 (0.0055)  time: 0.0596  data: 0.0010  max mem: 7163
Test:  [4400/5000]  eta: 0:00:46  model_time: 0.0535 (0.0606)  evaluator_time: 0.0040 (0.0055)  time: 0.0703  data: 0.0010  max mem: 7163
Test:  [4500/5000]  eta: 0:00:38  model_time: 0.0630 (0.0608)  evaluator_time: 0.0040 (0.0055)  time: 0.0983  data: 0.0011  max mem: 7163
Test:  [4600/5000]  eta: 0:00:30  model_time: 0.0535 (0.0608)  evaluator_time: 0.0040 (0.0055)  time: 0.0638  data: 0.0012  max mem: 7163
Test:  [4700/5000]  eta: 0:00:22  model_time: 0.0535 (0.0608)  evaluator_time: 0.0040 (0.0055)  time: 0.0675  data: 0.0010  max mem: 7163
Test:  [4800/5000]  eta: 0:00:15  model_time: 0.0520 (0.0607)  evaluator_time: 0.0040 (0.0055)  time: 0.0670  data: 0.0007  max mem: 7163
Test:  [4900/5000]  eta: 0:00:07  model_time: 0.0555 (0.0607)  evaluator_time: 0.0050 (0.0055)  time: 0.0704  data: 0.0011  max mem: 7163
Test:  [4999/5000]  eta: 0:00:00  model_time: 0.0491 (0.0607)  evaluator_time: 0.0040 (0.0055)  time: 0.0600  data: 0.0007  max mem: 7163
```

Рисунок 5 – Валидация модели

```
IoU metric: keypoints
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.491
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.753
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.528
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.462
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.559
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.591
 Average Recall     (AR) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.842
 Average Recall     (AR) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.633
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.543
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.660
```

Рисунок 6 – IoU метрики

Проведение регулярных проверок в процессе обучения помогает выявить такие проблемы, как переобучение, когда модель хорошо работает на тренировочных данных, но плохо на тестовых. Это также позволяет разработчикам принять меры для корректировки параметров обучения, выбора модели или самого процесса обучения в реальном времени.

Для предотвращения переобучения часто используются методы регуляризации. Одним из самых популярных методов является L2 регуляризация, известная также как Weight Decay. Этот метод добавляет к функции потерь член, пропорциональный квадрату нормы вектора весов модели:

$$L = L_0 + \lambda \|w\|^2, \tag{14}$$

где L_0 – исходная функция потерь;

w – вектор весов модели;

λ – коэффициент регуляризации, который контролирует степень влияния штрафа на итоговую функцию потерь.

Этот штраф за большие веса помогает предотвратить переобучение, делая модель менее чувствительной к небольшим колебаниям во входных данных, тем самым способствуя лучшей обобщающей способности.

Также широко используется регуляризация Dropout, при которой случайным образом исключается часть нейронов в процессе обучения, что помогает уменьшить зависимость модели от конкретных атрибутов входных данных и тем самым увеличить её способность к обобщению.

4 Методы преобразования двумерных ключевых точек в трехмерные

4.1 Трёхмерная реконструкция по нескольким изображениям

Мультивидовая стереоскопия (рисунок 7) является одним из наиболее эффективных подходов для создания детализированных трехмерных моделей из двухмерных изображений. Процесс включает в себя анализ нескольких снимков объекта, сделанных с разных ракурсов, что позволяет воссоздать объемные структуры с высокой точностью [20].

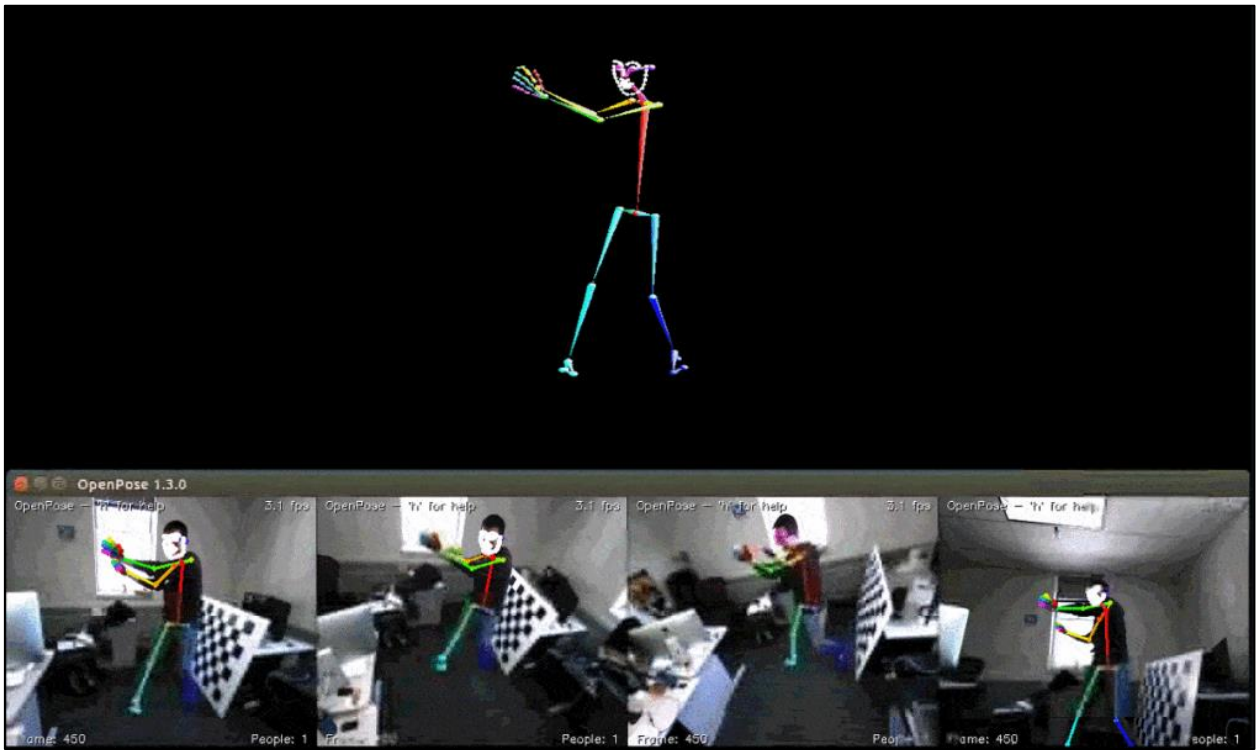

Рисунок 7 – Реконструкция 3D скелета из нескольких изображений [21]

На начальном этапе происходит сбор данных. Данный этап является критически важным этапом, поскольку качество и количество изображений напрямую влияют на точность и детализацию трехмерной модели. Изображения должны быть сняты так, чтобы между ними было достаточное перекрытие, что позволит алгоритмам сопоставления точек находить соответствия с высокой точностью. Обычно используются специализированные камеры, расположенные под разными углами относительно объекта.

Затем, алгоритмы компьютерного зрения, такие как SIFT (Scale-Invariant Feature Transform) или SURF (Speeded Up Robust Features), используются для обнаружения уникальных особенностей на изображениях. Эти особенности должны быть инвариантны к масштабированию, вращению и изменениям освещения, чтобы обеспечить их надежное сопоставление между разными снимками.

Финальным этапом является триангуляция. Триангуляция - это процесс определения положения точки в трехмерном пространстве по её проекциям на несколько изображений. Для каждой сопоставленной точки строится система уравнений, основанная на параметрах проекции каждой камеры и координатах точек на изображениях. Решение этой системы позволяет найти координаты точки в глобальной системе координат. Математически этот процесс можно описать через минимизацию ошибки воспроизведения:

$$\min_X \sum_{i=1}^{n} \|x_i - P_i X\|^2, \tag{15}$$

где X – искомая точка в трехмерном пространсве;

x_i – проекция точки X на изображение i;

P_i – матрица проекции камеры i.

Мультивидовая стереоскопия широко используется в киноиндустрии для создания реалистичных CGI (computer-generated imagery) сцен, в археологии для реконструкции артефактов, в робототехнике для создания точных карт окружающей среды, и в медицине для создания трехмерных моделей анатомических структур.

4.2 Одиночное изображение с использованием глубины

Использование одиночного изображения с картой глубины актуально, когда доступ к мультивидовым настройкам ограничен или невозможен. Этот метод позволяет эффективно восстанавливать пространственную структуру объектов и сцен из одного изображения, используя информацию о глубине каждой точки.

Первым этапом является получение карты глубины. Основная идея данного подхода состоит в том, что глубина пикселя на изображении известна. Но вот возможностей ее получения несколько:

– Стереокамеры: используют две камеры (или больше) для захвата двух видов одной и той же сцены под немного разными углами. Глубина рассчитывается на основе различий между этими двумя видами, опираясь на принципы триангуляции;

– LiDAR: системы, основанные на LiDAR излучают лазерные лучи и измеряют время, необходимое им для возвращения после отражения от объектов. Это время преобразуется в расстояние, создавая высокоточные карты глубины;

– современные методы глубокого обучения могут предсказывать карту глубины (рисунок 8) непосредственно из одиночного изображения на основе обучения модели на больших наборах данных с аннотированной глубиной.

Следующим этапом является преобразование 2D ключевых точек в 3D. Используя карту глубины, преобразование 2D пиксельных координат изображения в 3D пространственные координаты осуществляется через процесс, известный как проекция обратного вида (back-projection). Этот процесс учитывает внутреннюю калибровку камеры, включая фокусное расстояние и центр проекции, для перевода пиксельных координат в реальные координаты. Математически это описывается следующими уравнениями:

$$\begin{cases} x = (u - c_x) \cdot \frac{d}{f} \\ y = (v - c_y) \cdot \frac{d}{f}, \\ z = d \end{cases} \tag{16}$$

где u, v – координаты пикселя на изображении;

c_x, c_y – координаты центра проекции;

f – фокусное расстояние камеры;

d – значение глубины для пикселя.

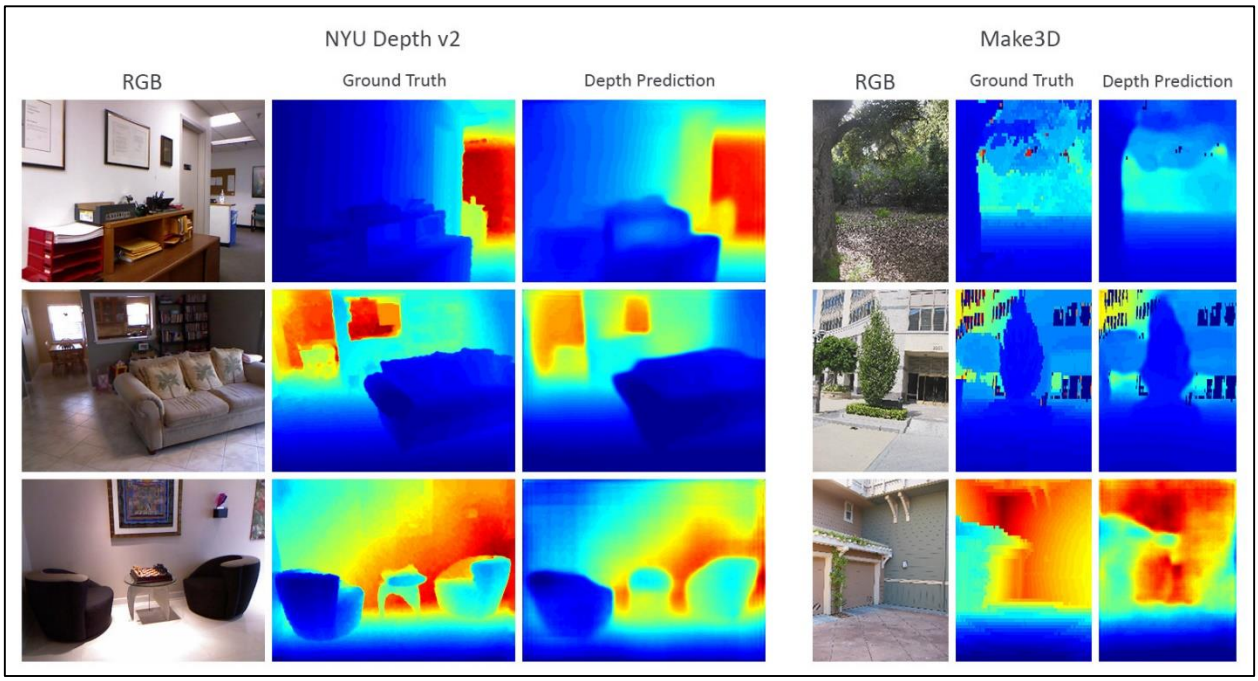

Рисунок 8 – Карта глубины, предсказанная с помощью нейросетей [22]

4.3 Использование учебных данных с аннотацией глубины

В данном методе предполагается, что данные для обучения модели предварительно дополнительно размечены значениями глубины. Он имеет множество преимуществ, таких как: отсутствие необходимости в дополнительных устройствах, высокая мобильность и возможность работы в реальном времени. Но при этом обладает и недостатками: качество напрямую зависит от качество данных для обучения.

4.4 Собственная реализация

4.4.1 Описание модели MiDaS

Для реализации собственного решения была выбрана модель определения глубины MiDaS.

MiDaS (Monocular Depth Estimation via Deep Learning) — это нейросетевая модель, разработанная для оценки глубины сцены из одиночных изображений. Модель была обучена на множестве данных, содержащих изображения с разнообразными сценами, что позволяет ей эффективно определять глубину в различных условиях освещения и композиций.

Принцип ее работы заключается в том, что она использует предобученную на датасете ImageNet сеть для извлечения признаков из входного изображения. Затем, на основе признаков модель предсказывает карту глубины, где каждому пикселю изображения соответствует значение глубины. Это достигается за счет использования слоев, специально адаптированных для работы с регрессией глубины.

4.4.2 Преобразование 2D ключевых точек в 3D координаты

Для трансформации 2D ключевых точек в их 3D координаты используется карта глубины, сгенерированная моделью MiDaS. Каждой ключевой точке, определённой на изображении, соответствует пиксель на карте глубины. Из этого пикселя извлекается значение глубины, которое представляет собой расстояние от камеры до объекта в данной точке.

На практике, значения глубины могут содержать шумы или неточности из-за ограничений самой модели глубины или из-за визуальных аномалий в исходном изображении. Эти неточности могут привести к ошибкам в определении истинного 3D положения точек. Чтобы уменьшить эти ошибки, применяется процесс коррекции [23]:

1) вычисляется евклидово расстояние между соседними ключевыми точками в проекции на плоскость изображения и в пространстве предсказанных глубин;

2) если расстояние между точками превышает заданный порог, предполагается, что между точками произошла ошибка в оценке глубины;

3) координаты точек корректируются, чтобы минимизировать визуальные искажения, такие как неестественные изгибы или разрывы скелета.

Для понимания общей логики работы всей системы была создана диаграмма последовательности (рисунок 9).

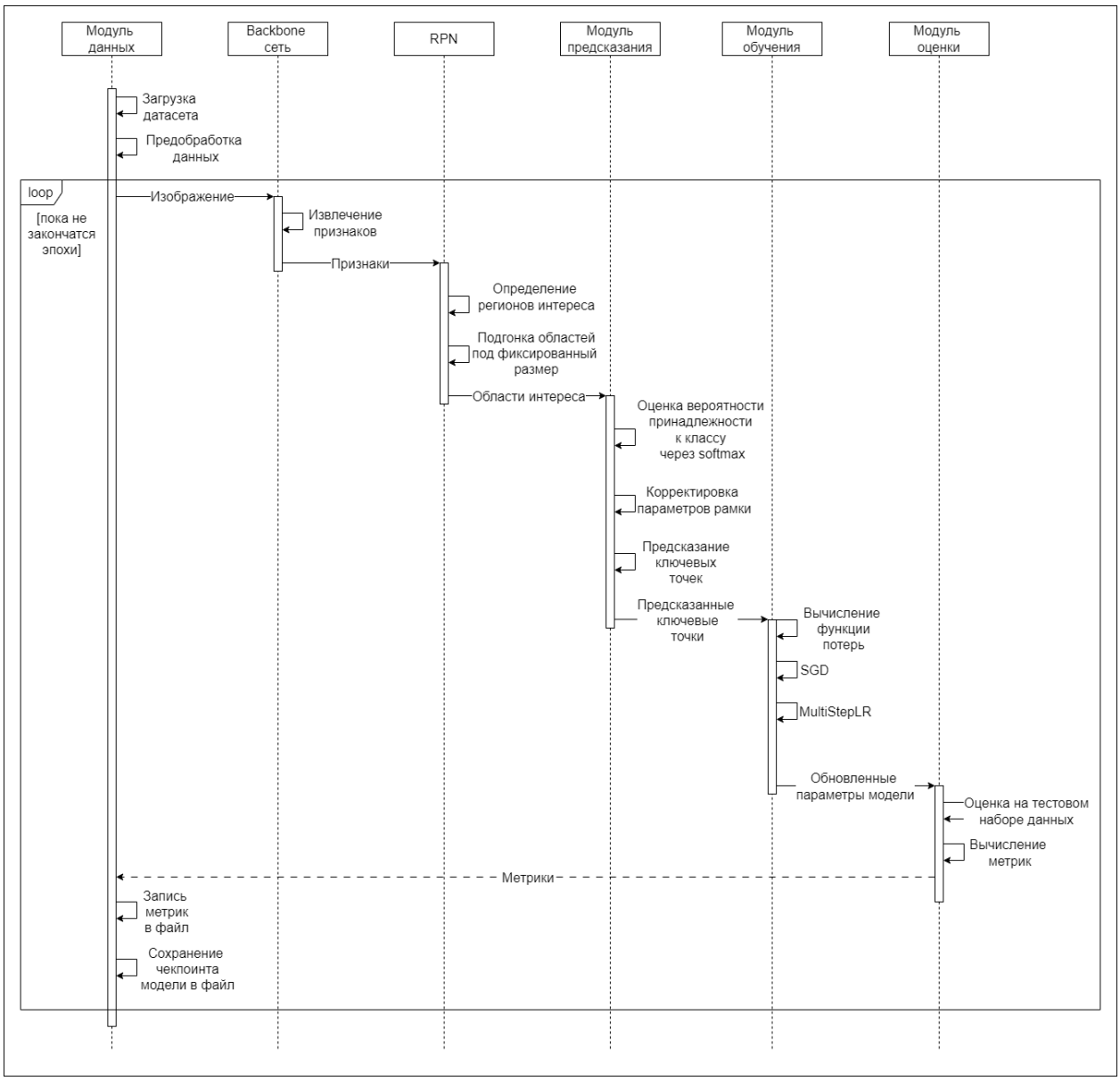

Рисунок 9 – Диаграмма последовательности работы системы определения трехмерных ключевых точек

5 Анализ полученных результатов

5.1 Описание эксперимента

В рамках экспериментальной проверки эффективности разработанной модели для распознавания ключевых точек человеческого тела на изображении был реализован комплексный подход, включающий загрузку и обработку изображений, применение предварительно обученной модели и визуализацию результатов. Для достижения максимальной точности и наглядности результатов эксперименты проводились с использованием специализированных библиотек и алгоритмов компьютерного зрения.

Ядро эксперимента составляют PyTorch и его модуль для работы с изображениями torchvision, которые предоставляют инструменты для глубокого обучения и обработки изображений. Это включает в себя загрузку предварительно обученных моделей, операции с тензорами и их обработку.

Для распознавания ключевых точек использовалась обученная ранее собственная модель, адаптированная для задачи с использованием нескольких архтиектур в качестве основы. Эта модель была обучена на отдельных данных и загружена из сохраненного состояния (checkpoint).

Исходные изображения загружались в формате RGB, после чего преобразовывались в тензоры с помощью функций библиотеки torchvision. Это преобразование включало нормализацию данных и добавление размерности батча для соответствия входным требованиям модели.

Для визуализации результатов распознавания использовались предопределенные соединения между ключевыми точками (соединения рук, туловища, ног и головы), что позволило наглядно демонстрировать положение и ориентацию человеческого тела на изображении.

Для отрисовки ключевых точек и соединений на изображении использовалась библиотека OpenCV, а результаты визуализировались с помощью Matplotlib.

Для улучшения визуализации трехмерного пространственного распределения ключевых точек человеческого тела, к эксперименту было добавлено

использование технологии визуализации глубины. Это включало применение модели оценки глубины MiDaS. Модель создает карту глубины, которая используется для корректировки и уточнения положения ключевых точек в трехмерном пространстве.

Ключевые точки теперь могли быть обогащены данными о глубине, полученными из карты глубины. Это позволило проводить визуализацию не только в двух измерениях, но и в трехмерной перспективе.

Визуализация трехмерных скелетов проводилась с использованием Matplotlib и его трехмерного модуля *mpl_toolkits.mplot3d*.

В рамках исследования были обучены модели с тремя различными архитектурами backbone: ResNet18, ResNet50 и ResNet101. Основное отличие этих моделей заключается в количестве слоев и сложности архитектуры, что, в свою очередь, влияет на их способность извлекать и обрабатывать признаки из изображений. Значения AP и AR были перенесены в таблицу 2.

Таблица 2 – Результаты, полученные после обучения моделей на полном датасете COCO с различными архитектурами

Backbone	AP_{50}	AP_{75}	AR_{50}	AR_{75}
ResNet18	82,2	65,1	88,7	72,4
ResNet50	85,5	69,7	90,5	76,0
ResNet101	84,2	67,0	89,6	74,0

ResNet18 показала наименьшую точность среди всех трех моделей, что ожидаемо, учитывая её простую архитектуру и небольшое количество слоев. Несмотря на это, модель все же смогла достичь относительно высокого значения AP_{50} и AR_{50}. Меньшая глубина сети приводит к более ограниченной способности извлекать сложные признаки из изображений, что отражается на значениях AP_{75} и AR_{75}, которые существенно ниже по сравнению с более глубокими моделями. Модель обеспечивает приемлемую точность и полноту, но её возможности ограничены в сложных сценариях.

ResNet50 продемонстрировала наилучшие результаты среди всех моделей. Значительное улучшение точности по сравнению с ResNet18 объясняется большей глубиной сети и, как следствие, более эффективным извлечением

признаков. Это подтверждается высокими значениями как AP_{50} и AR_{50}, так и AP_{75} и AR_{75}. Улучшение точности и полноты при увеличении глубины сети показывает, что модель способна лучше адаптироваться к различным вариациям в данных. Более глубокая сеть может распознавать более сложные и детализированные признаки, что приводит к повышению эффективности при разных уровнях IoU.

ResNet101, несмотря на наибольшую глубину среди всех рассмотренных моделей, показала результаты несколько хуже, чем ResNet50. Это может быть связано с несколькими факторами, включая возможное переобучение или сложности в оптимизации такой глубокой сети. Хотя ResNet101 теоретически должна обеспечивать лучшее извлечение признаков за счет своей глубины, на практике, увеличение числа слоев не всегда приводит к улучшению результатов, если модель становится слишком сложной и трудной для обучения. Более высокая вычислительная сложность и требования к ресурсам также могли сыграть роль в этом. Несмотря на это, ResNet101 все же показала хорошие результаты и превзошла ResNet18 по всем метрикам.

Ввиду того, что архитектура ResNet50 продемонстрировала наилучшие результаты среди всех трех исследуемых моделей, целесообразно провести более детальный анализ ее производительности и рассмотреть дополнительные метрики, полученные в процессе обучения.

5.2 Результаты работы модели определения двумерных ключевых точек

На рисунке 10 можно наблюдать результат работы модели компьютерного зрения, которая выполняет задачу определения позы человека. На изображении изображены четыре человека, каждому из которых модель пыталась присвоить ключевые точки тела и соединить их линиями для визуализации структуры скелета.

Хоть и оценка такого рода результатов является субъективной, кажется, что модель достаточно точно определила ключевые точки всех людей на данном изображении, что говорит об успешности эксперимента.

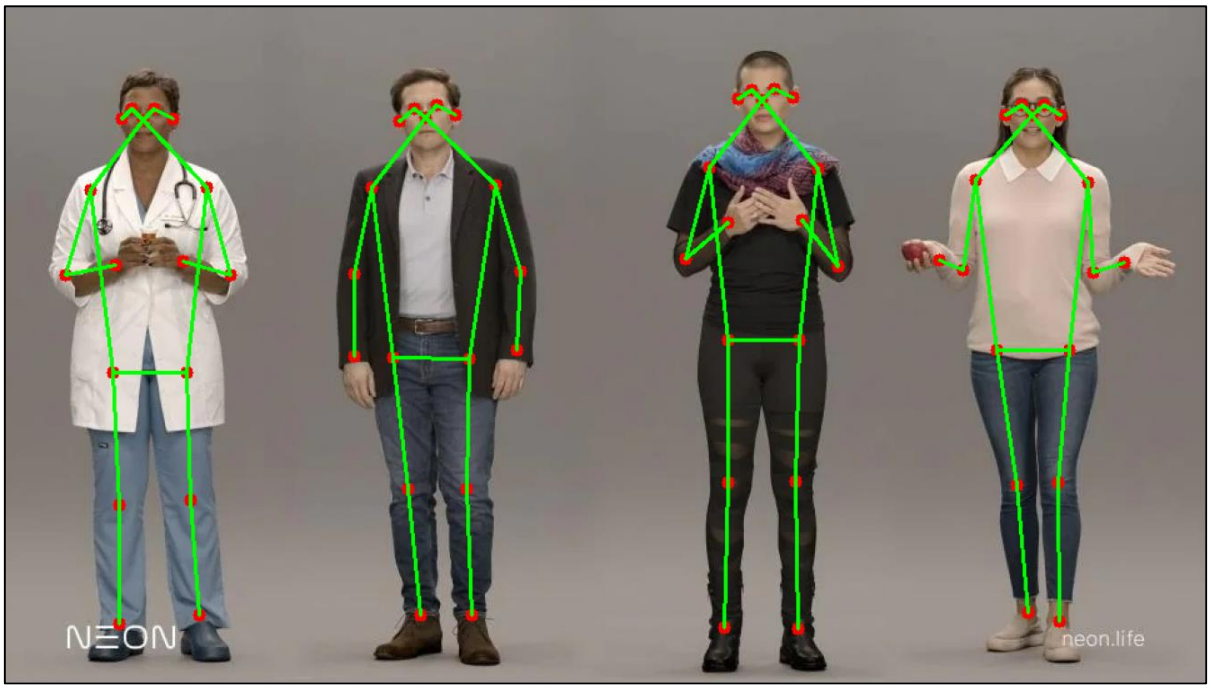

Рисунок 10 – Результат работы собственной модели для определения двумерных ключевых точек

На рисунке 11 изображено изменение скорости обучения (learning rate) по эпохам в процессе обучения нейронной сети. График показывает, как параметр скорости обучения изменяется с течением времени от начала до конца обучения модели.

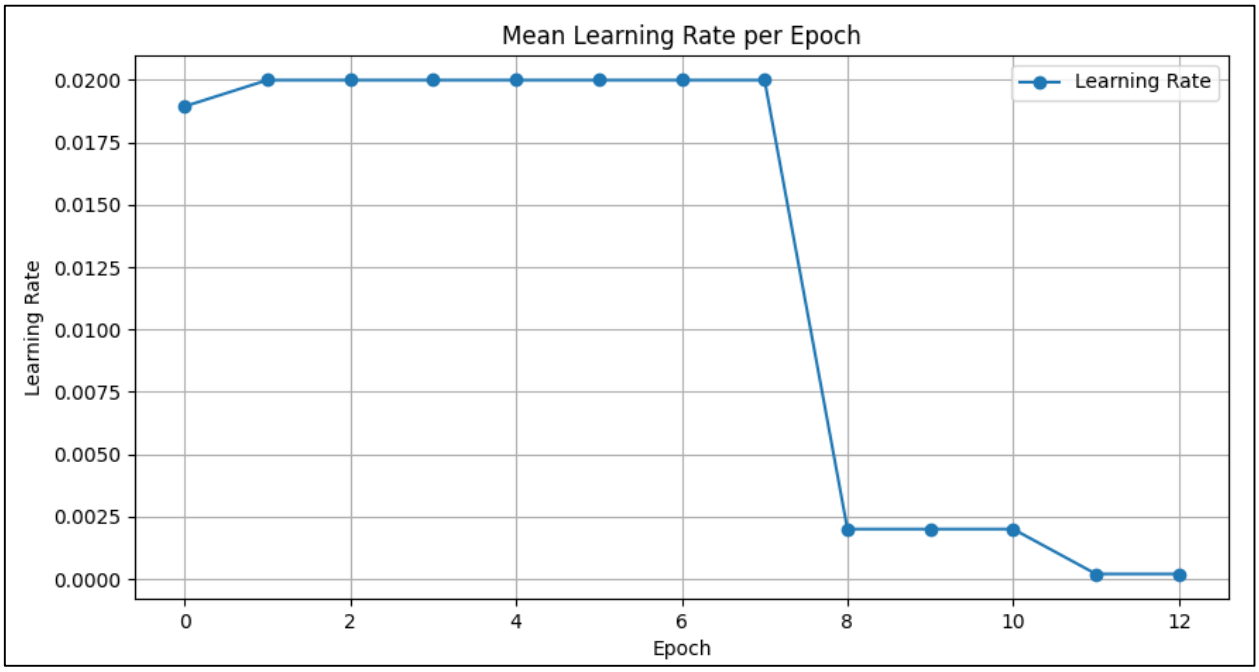

Рисунок 11 – График значений скорости обучения

На начальном этапе обучения, от эпохи 0 до эпохи 4, скорость обучения остается практически неизменной. Это стандартная практика, когда начальные параметры скорости обучения устанавливаются на уровне, который способствует стабильному, но не слишком быстрому улучшению производительности модели.

Наблюдается значительное падение скорости обучения после 5-й эпохи. Это сделано для того, чтобы помочь модели более тонко настраивать свои параметры и избегать переобучения. Особенно это актуально, если начальные эпохи показали достаточное снижение функции потерь и улучшение других метрик точности.

После резкого падения скорость обучения продолжает снижаться и остается на низком уровне в последних эпохах. Это указывает на то, что обучение приближается к завершению, и модель достигает состояния, в котором дальнейшие изменения параметров происходят очень осторожно, чтобы не нарушить уже достигнутую точность.

Снижение скорости обучения во время тренировки модели является общепринятой техникой, которая помогает модели постепенно сходиться к оптимальному решению, уменьшая риск переобучения и позволяя более точно адаптировать веса сети. Этот метод часто используется в сочетании с другими техниками регуляризации и оптимизации процесса обучения.

На рисунке 12 изображено изменение средней потери по ключевым точкам (mean keypoint loss) модели машинного обучения по эпохам обучения. График показывает, как модель оптимизировала свою способность точно определять ключевые точки на объектах в данных.

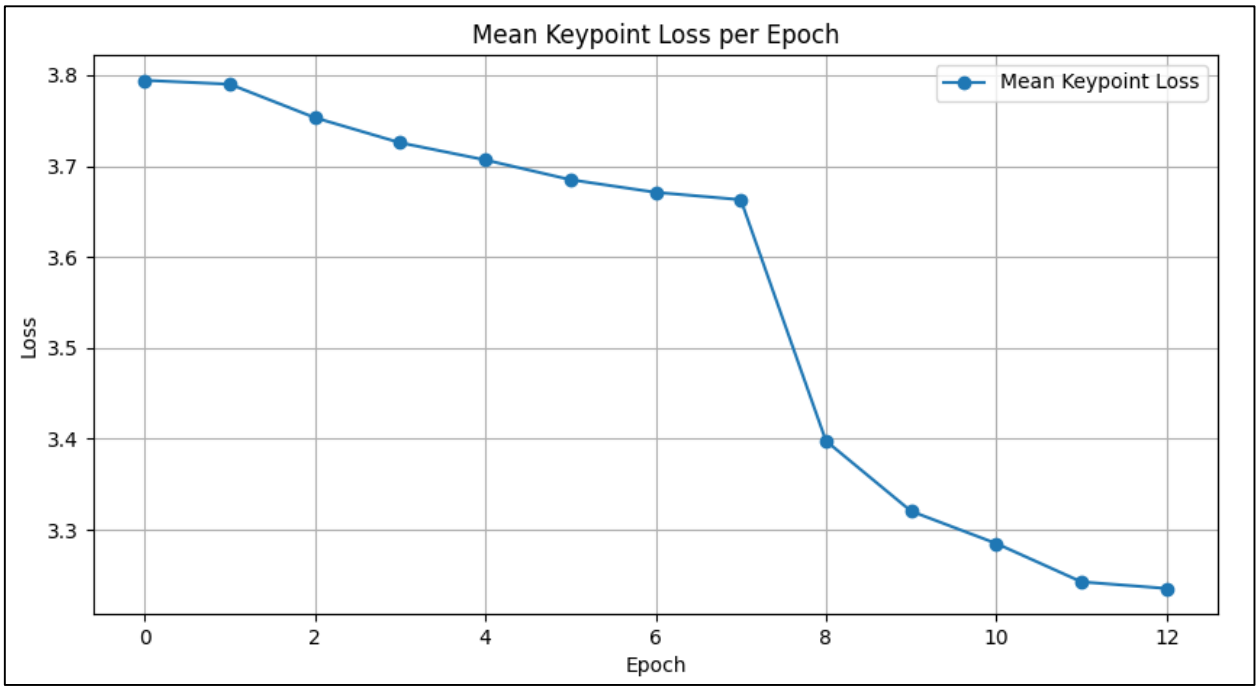

Рисунок 12 – Изменение средней потери по ключевым точкам по эпохам обучения

На начальном этапе обучения средняя потеря ключевых точек постепенно уменьшается. Это указывает на то, что модель начинает адаптироваться к задаче и улучшает свои предсказания благодаря уменьшению разницы между предсказанными и фактическими позициями ключевых точек.

На эпохах 4-8 видно стабильное, но более медленное снижение потерь, что может указывать на постепенное улучшение модели и возможное приближение к пределу своих возможностей на данном наборе данных.

На 8 эпохе можно наблюдать резкое снижение потерь. Это может быть следствием существенного изменения скорости обучения (рисунок 12).

На последних эпохах потери продолжают уменьшаться, хотя и более плавно. Это указывает на то, что модель продолжает оптимизировать свои веса для улучшения производительности, но большинство значительных улучшений уже достигнуто.

Снижение потерь по ключевым точкам на протяжении эпох свидетельствует о эффективности процесса обучения модели. Резкое уменьшение потерь может указывать на успешное применение методик оптимизации обуче-

ния, таких как изменение гиперпараметров или использование адаптивных методов обучения. Стабилизация потерь на более поздних стадиях обучения указывает на то, что модель достигла своего предела точности на доступных данных.

На рисунке 13 изображены изменения средней точности (Average Precision, AP) и средней полноты (Average Recall, AR) при значении $IoU = 0{,}50$ для ключевых точек по мере обучения модели.

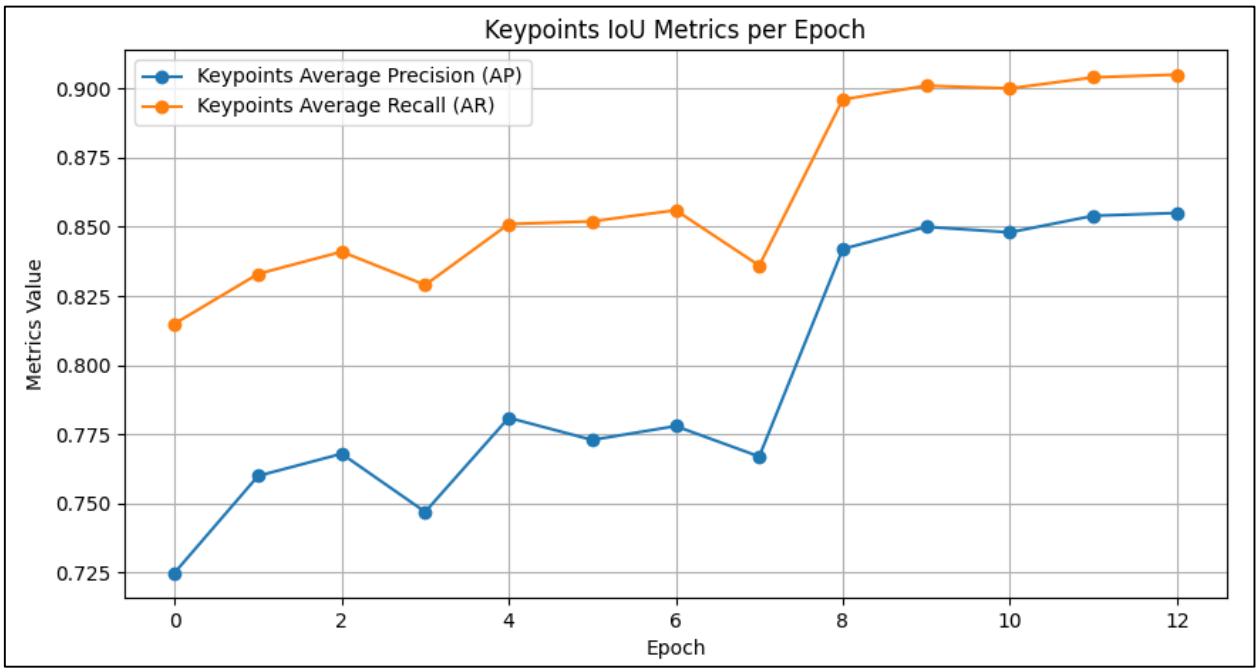

Рисунок 13 – Изменения средней точности и средней полноты для ключевых точек по мере обучения модели

Average Precision (AP) относится к средней точности предсказаний модели. Это мера, которая учитывает и точность, и полноту, где точность — это доля правильно идентифицированных положительных результатов среди всех идентифицированных положительных результатов, а полнота показывает, какая доля реальных положительных результатов была обнаружена моделью.

Average Recall (AR) измеряет способность модели обнаруживать все релевантные случаи в данных. Это доля правильно идентифицированных положительных результатов среди всех реальных положительных случаев.

Начальное значение AP начинается относительно низко, что может указывать на недостаточную способность модели точно локализовать ключевые

точки на ранних этапах обучения. AR начинается выше, чем AP, и демонстрирует более плавные изменения, что характерно для метрики полноты, так как она менее чувствительна к точным локациям и более фокусирована на количестве обнаруженных правильных точек.

Виден рост AP с 6-й по 8-ю эпоху, что коррелирует с резким падением потерь ключевых точек, показанным на предыдущем графике. Это улучшение может быть результатом оптимизаций или корректировки параметров обучения, таких как скорость обучения. Значительное увеличение AR с 6-й по 8-ю эпоху также совпадает с улучшением точности, подтверждая, что модель становится лучше не только в точности, но и в способности обнаруживать больше релевантных ключевых точек.

Стабилизация AP после 8-й эпохи свидетельствует о достижении моделью некоторого порога эффективности. Постоянство AR на высоком уровне после 8-й эпохи показывает, что модель эффективно обнаруживает большинство ключевых точек.

Эффективность модели в задачах распознавания ключевых точек значительно улучшилась после определенных эпох, что подтверждается увеличением метрик точности и полноты. Управление параметрами обучения, такими как скорость обучения, играет важную роль в достижении оптимальной производительности модели.

5.3 Результаты получения трёхмерных ключевых точек

На рисунке 14 можно увидеть карту глубины, которая получена с помощью MiDaS для дальнейшего использования при построении трехмерного скелета.

Рисунок 14 – Карта глубины, полученная с помощью MiDaS

Карта глубины представляет собой визуализацию, где более темные области соответствуют большей глубине (далее от камеры), а светлые области ближе. Эта карта, полученная из модели глубокого обучения, кажется довольно размытой и нечеткой, что указывает на некоторую неточность в определении точного расстояния до различных частей тела человека. Нечеткость может быть связана с ограничениями самой модели глубокого обучения, такими как недостаточное количество данных для обучения или сложность интерпретации данных сцен с однородными или сложными текстурами.

На рисунке 15 показан трехмерный скелет, созданный на основе карты глубины и определения ключевых точек. Скелет визуализирован с помощью линий, соединяющих ключевые точки, определенные алгоритмом.

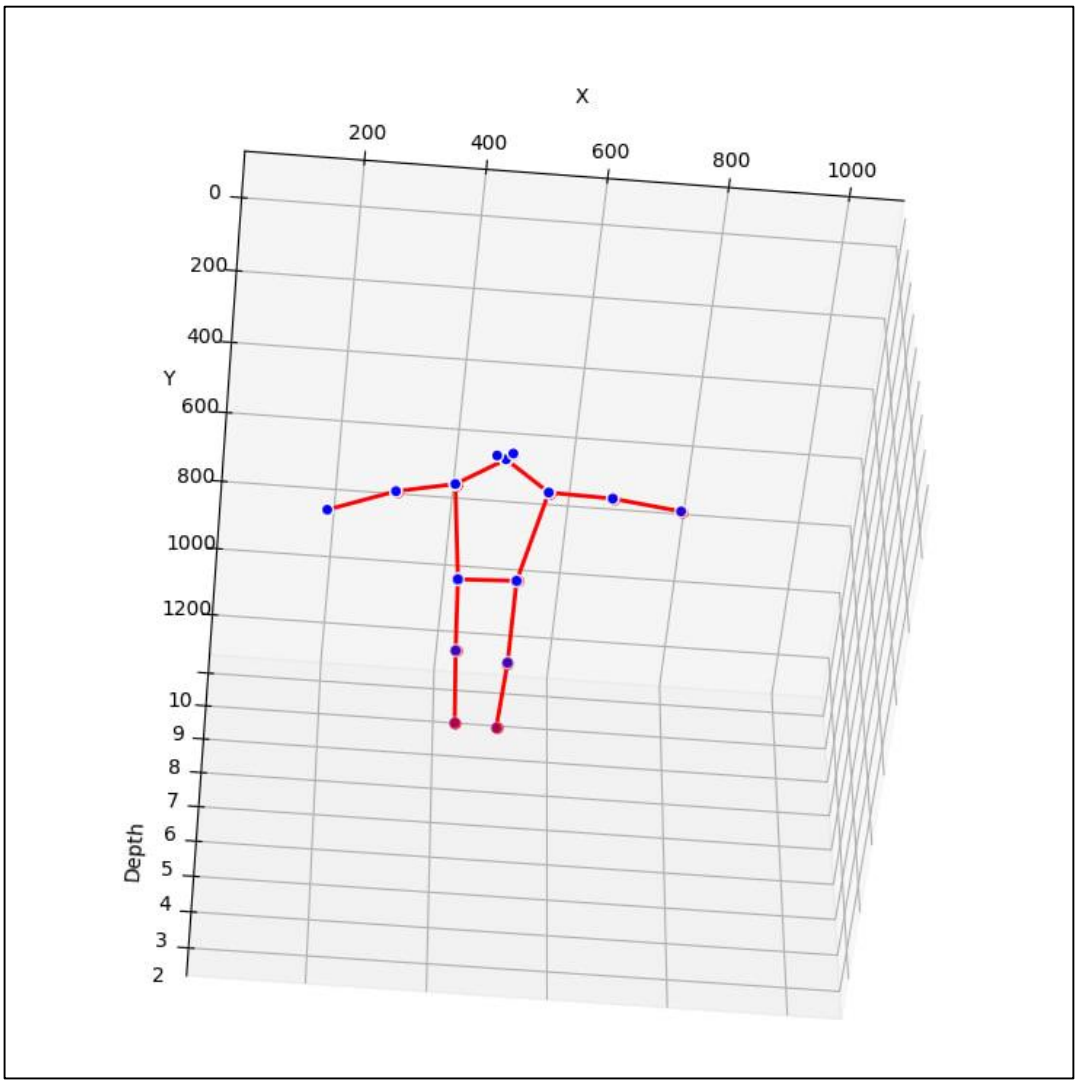

Рисунок 15 – Результат совместной работы собственной модели определения двумерных ключевых точек с моделью MiDaS

Можно заметить, что общая картина трехмерного скелета похожа на правдивую позу из входного изображения. Но, как уже упоминалось, карта глубины демонстрирует значительную нечеткость, что может привести к ошибкам в определении точного положения ключевых точек в пространстве. Это может оказать влияние на точность всей системы построения 3D модели скелета. Ошибки в данных о глубине и определении ключевых точек могут усугубляться при их слиянии. Например, если ключевая точка неправильно

помечена на изображении или если глубина этой точки неверно рассчитана, это приведет к созданию неправильной 3D модели.

Эксперимент подчеркивает значимость точности в каждом из аспектов системы, особенно в генерации карты глубины и определении ключевых точек. Модель для определения 2D ключевых точек демонстрирует хорошие результаты, однако задача генерации карты глубины требует дополнительной работы. Важно отметить, что использование модели глубины MiDaS в данной работе служило лишь в качестве инструмента, а основное внимание было уделено другим аспектам исследования. Существующие проблемы с точностью карты глубины подчеркивают потребность в дальнейших исследованиях и уточнении этой части системы, чтобы достичь более высокой точности в создании трехмерных моделей на основе полученных данных.

ЗАКЛЮЧЕНИЕ

В рамках данной работы было проведено исследование применимости алгоритмов машинного обучения для захвата и анализа движений человека, с особым акцентом на использование сверточных нейронных сетей в контексте обработки изображений и видео.

Основной фокус работы был направлен на реализацию и апробацию модели, адаптированной для задач захвата движения, где ключевую роль играет архитектура ResNet50 в качестве backbone. Модель позволила не только точно локализовать ключевые точки человеческого тела на двумерных изображениях, но и расширить анализ до трехмерного пространства с помощью интеграции алгоритмов оценки глубины изображения.

Для оценки эффективности были проведены эксперименты с тремя различными архитектурами: ResNet18, ResNet50 и ResNet101. Результаты показали, что модель с архитектурой ResNet50 продемонстрировала наилучшие результаты, достигая высоких показателей точности и надежности локализации ключевых точек. Эти результаты указывают на то, что архитектура ResNet-50 обеспечивает оптимальный баланс между точностью и вычислительной эффективностью, что делает её предпочтительной для использования в реальных приложениях.

Среди основных вызовов, с которыми столкнулись в ходе исследования, следует отметить сложности, связанные с точностью локализации ключевых точек в условиях низкой контрастности и при частичном перекрытии объектов. Решение данных проблем было частично найдено за счет внедрения корректировки ключевых точек на основе анализа глубины, что позволило значительно повысить точность распознавания движений.

Перспективы развития данной области весьма обширны. Возможность использования алгоритмов для захвата движения без специальных маркеров открывает новые направления в реабилитационной медицине, спортивных науках и развлекательной индустрии. Например, точный захват движений поз-

волит создавать более сложные и интерактивные системы виртуальной реальности, а также способствовать более глубокому анализу техники спортсменов для предотвращения травм.

Дальнейшие исследования могут быть направлены на улучшение алгоритмов обработки изображений с низким разрешением, а также на разработку новых методов обучения моделей, способных адаптироваться к различным условиям освещения и фона. Также важным аспектом остается минимизация времени обработки данных в реальном времени для создания систем, способных функционировать в динамичных, непредсказуемых условиях.

Целесообразным является также использование рекуррентных нейронных сетей (RNN), включая LSTM и GRU, для анализа временных последовательностей движения. Эти технологии могут значительно улучшить способность системы распознавать и интерпретировать сложные последовательности движений, что особенно важно в задачах, где необходимо учитывать динамику и предыдущее состояние для предсказания будущих поз.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Common Problems in Motion Capture // Axis : [сайт]. – 2024. – URL: https://axisxr.gg/common-problems-in-motion-capture/ (дата обращения: 20.04.2022)

2 Improving classification performance of softmax loss function based on scalable batch-normalization / Q. Zhu, Z. He, T. Zhang, W. Cui. – DOI: doi.org/10.3390/app10082950 // Applied Sciences. – 2020. – Vol. 10, Number 8. – P. 2950. – URL: https://www.mdpi.com/2076-3417/10/8/2950

3 Романов, А. А. Сверточные нейронные сети / А.А. Романов Научные исследования: ключевые проблемы III тысячелетия : сборник научных трудов по материалам XXI Международной научно-практической конференции, Москва, 09–10 января 2018 года. – Москва : Проблемы науки, 2018. – С. 5-9. – URL: https://scientificresearch.ru/images/PDF/2018/21/svertochnye.pdf

4 Convolutional neural networks: an overview and application in radiology / R. Yamashita, M. Mishio, K.G. Richard, K. Togashi. – DOI: 10.1007/s13244-018-0639-9 // Insights into imaging. – 2018. – Vol. 9. – P. 611-629. – URL: https://link.springer.com/content/pdf/10.1007/s13244-018-0639-9.pdf

5 Гончаров, Е. И. Многомерно-матричное определение операции свертки / Е. И. Гончаров. – DOI: 10.25559/SITITO.17.202103.541-549 // Современные информационные технологии и ИТ-образование. – 2021. – Т. 17, №. 3. – С. 541-549. – URL: https://cyberleninka.ru/article/n/mnogomerno-matrichnoe-opredelenie-operatsii-svertki

6 Wu, H. Max-pooling dropout for regularization of convolutional neural networks / H. Wu, X. Gu. – DOI: 10.1007/978-3-319-26532-2_6 // Neural Information Processing : 22nd International Conference, November 9-12. – Istanbul : Springer International Publishing, 2015. – P. 46-54. – URL: https://arxiv.org/pdf/1512.01400

7 Frequency bias in neural networks for input of non-uniform density / R. Basri, M. Galun, A. Geifman [et al.]. – DOI: 10.5555/3524938.3525002 // International Conference on Machine Learning. – 2020. – P. 685-694. – URL: https://proceedings.mlr.press/v119/basri20a.html

8 Deep ran: A scalable data-driven platform to detect anomalies in live cellular network using recurrent convolutional neural network / R. Tanhatalab, H. Yousefi, H. Hosseini [et al.]. – DOI: 10.48550/arXiv.1911.04472 // 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2020. – P. 269-274. – URL: https://arxiv.org/pdf/1911.04472

9 Microsoft coco: Common objects in context / T. Lin, M. Maire, S. Belongie [et al.]. – DOI: 10.1007/978-3-319-10602-1_48 // Computer Vision–ECCV 2014 : 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13. – Springer International Publishing. – 2014. – P. 740-755.

10 Imagenet: A large-scale hierarchical image database / J. Deng, W. Dong, R. Socher [et al.]. – DOI: 10.1109/CVPR.2009.5206848 // 2009 IEEE conference on computer vision and pattern recognition. – 2009. – P. 248-255. – URL: https://projet.liris.cnrs.fr/imagine/pub/proceedings/CVPR-2009/data/papers/0103.pdf

11 Res2net: A new multi-scale backbone architecture / S. Gao, M. Cheng, K. Zhao [et al.]. – DOI: 10.1109/TPAMI.2019.2938758 // IEEE transactions on pattern analysis and machine intelligence. – 2019. – Vol. 43, Number 2. – P. 652-662. – URL: https://arxiv.org/pdf/1904.01169

12 Going deeper in spiking neural networks: VGG and residual architectures / A. Sengupta, Y. Ye, R. Wang [et al.]. – DOI: 10.3389/fnins.2019.00095 // Frontiers in neuroscience. – 2019. – Vol. 13. – P. 95. – URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2019.00095/full

13 Nam, S. Improvement in object detection using multi-scale RoI pooling and feature pyramid network / S. Nam, D, Lee. – DOI: 10.5626/JCSE.2022.16.1.14 // Journal of Computing Science and Engineering. – 2022. – Vol. 16, Number 1. –

P. 14-24. – URL: https://www.dbpia.co.kr/Journal/articleDetail?nodeId=NODE11053665

14 Understanding Region of Interest (RoI Pooling) // Erdem : [сайт]. – 2020. – URL: https://erdem.pl/2020/02/understanding-region-of-interest-ro-i-pooling

15 Roi pooled correlation filters for visual tracking / Y. Sun, C. Sun. D. Wang [et al.]. – DOI: 10.1109/CVPR.2019.00593 // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2019. – P. 5783-5791. – URL: https://openaccess.thecvf.com/content_CVPR_2019/html/Sun_ROI_Pooled_Correlation_Filters_for_Visual_Tracking_CVPR_2019_paper.html

16 A high-speed and low-complexity architecture for softmax function in deep learning / M. Wang, S. Lu, D. Zhu [et al.]. – DOI: 10.1109/APCCAS.2018.8605654 // 2018 IEEE asia pacific conference on circuits and systems (APCCAS). – 2018. – P. 223-226. – URL: https://www.researchgate.net/profile/Meiqi-Wang-5/publication/330585942_A_High-Speed_and_Low-Complexity_Architecture_for_Softmax_Function_in_Deep_Learning/links/5ef70ca145851550507538bb/A-High-Speed-and-Low-Complexity-Architecture-for-Softmax-Function-in-Deep-Learning.pdf

17 Amir, I. SGD generalizes better than GD (and regularization doesn’t help) / I. Amir, T. Koren, R. Livni. – DOI: 10.48550/arXiv.2102.01117 // Conference on Learning Theory. – 2021. – P. 63-92. – URL: https://proceedings.mlr.press/v134/amir21a/amir21a.pdf

18 Takase, T. Effective neural network training with adaptive learning rate based on training loss / T. Takase, S. Oyama, S. Kurihara. – DOI: 10.1016/j.neunet.2018.01.016 // Neural Networks. – 2018. – Vol. 101. – P. 68-78. – URL: https://www.sciencedirect.com/science/article/abs/pii/S0893608018300303

19 Wei, L. Oriented Object Detection in Aerial Images Based on the Scaled Smooth L1 Loss Function / L. Wei, C. Zheng, Y. Hu. – DOI: 10.3390/rs15051350 // Remote Sensing. – 2023. – Vol. 15, Number 5. – P. 1350. – URL: https://www.mdpi.com/2072-4292/15/5/1350

20 Stereoscopy and the human visual system / M. Banks, J. Read, R. Allison, S. Watt. – DOI: 10.5594/j18173 // SMPTE motion imaging journal. – 2012. – Vol. 121, Number 4. – P. 24-43. – URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3490636/

21 OpenPose Advanced Doc - 3-D Reconstruction Module and Demo // GitHub : [сайт]. – 2022. – URL: https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/advanced/3d_reconstruction_module.md

22 Monocular depth estimation // Keras : [сайт]. – 2021. – URL: https://keras.io/examples/vision/depth_estimation/

23 2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud / M. Feng, S. Hu, M. Ang, G. Lee. – DOI: 10.1109/ICRA.2019.8794415 // 2019 International Conference on Robotics and Automation (ICRA). – 2019. – P. 4790-4796.

ПРИЛОЖЕНИЕ А

Фрагменты кода программы

А.1 Функция для обучения модели

```
def train(backbone_model='resnet50'):
    output_directory = '..'
    log file path = os.path.join(output directory, f'{backbone model} train-
ing.log')

    with redirect_stdout_to_file(log_file_path):
        print(f"Active device: {device}")

        dataset_key = 'coco_kp'
        params = {
            'batch_size': 5,
            'epochs': 13,
            'num_workers': max(1, os.cpu_count() - 2 if os.cpu_count() is not
None else 1),
            'learning_rate': 0.02,
            'momentum': 0.9,
            'weight_decay': 1e-4,
            'lr_steps': [8, 11],
            'lr_gamma': 0.1,
            'print_frequency': 20,
            'resume_path': '',
            'aspect_ratio_group_factor': 0,
            'test_only': False,
            'pretrained': True
        }

        print(f"Setting number of workers to {params['num_workers']}")

        print("Loading datasets")
        train_dataset, num_classes = get_dataset(dataset_key, "train_libs",
get_transform(train=True), subset_size=None)
        val_dataset, _ = get_dataset(dataset_key, "val", get_trans-
form(train=False), subset_size=None)

        print("Creating data loaders")
        train_sampler = torch.utils.data.RandomSampler(train_dataset)
        val_sampler = torch.utils.data.SequentialSampler(val_dataset)

        if params['aspect_ratio_group_factor'] >= 0:
            aspect_ratio_groups = create_aspect_ratio_groups(train_dataset,
k=params['aspect_ratio_group_factor'])
            train_batch_sampler = GroupedBatchSampler(train_sampler, as-
pect_ratio_groups, params['batch_size'])
        else:
            train_batch_sampler = torch.utils.data.BatchSampler(train_sam-
pler, params['batch_size'], drop_last=True)

        train_loader = torch.utils.data.DataLoader(train_dataset, batch_sam-
pler=train_batch_sampler,
                                                   num_work-
ers=params['num_workers'],
                                                   collate_fn=utils.col-
late_fn)
        val_loader = torch.utils.data.DataLoader(val_dataset, batch_size=1,
sampler=val_sampler,
```

```
                                                num_work-
ers=params['num_workers'], collate_fn=utils.collate_fn)

        print("Initializing model")
        backbone = resnet_fpn_backbone(backbone_model, pre-
trained=params['pretrained'])
        detection_model = KeypointRCNN(backbone, num_classes=num_classes)
        detection_model.to(device)

        model_single_device = detection_model

        trainable_params = [p for p in detection_model.parameters() if p.re-
quires_grad]
        optimizer = torch.optim.SGD(trainable_params, lr=params['learn-
ing_rate'], momentum=params['momentum'],
                                    weight decay=params['weight decay'])
        lr_scheduler = torch.optim.lr_scheduler.MultiStepLR(optimizer, mile-
stones=params['lr_steps'],

gamma=params['lr_gamma'])

        if params['resume_path']:
            checkpoint = torch.load(params['resume_path'], map_location=de-
vice)
            model_single_device.load_state_dict(checkpoint['model'])
            optimizer.load_state_dict(checkpoint['optimizer'])
            lr_scheduler.load_state_dict(checkpoint['lr_scheduler'])

        if params['test_only']:
            evaluate(detection_model, val_loader, device=device)
            return

        print("Starting training")
        start_time = time.time()

        for epoch in range(params['epochs']):
            train_one_epoch(detection_model, optimizer, train_loader, device,
epoch, params['print_frequency'])
            lr_scheduler.step()
            if output_directory:
                model_save_path = os.path.join(output_directory, f'{back-
bone_model}_epoch_{epoch}.pth')
                utils.save_on_master({'model': model_single_de-
vice.state_dict(), 'optimizer': optimizer.state_dict(),
                                      'lr_scheduler': lr_sched-
uler.state_dict()}, model_save_path)

            evaluate(detection_model, val_loader, device=device)

        elapsed_time = time.time() - start_time
        formatted_time = str(datetime.timedelta(seconds=int(elapsed_time)))
        print('Total training time: {}'.format(formatted_time))
```

A.2 Функция обучения одной эпохи

```
def train_one_epoch(model, opt, loader, device, epoch_num, print_interval):
    model.train()
    metrics = utils.MetricLogger(delimiter="  ")
    metrics.add_meter('learning_rate', utils.SmoothedValue(window_size=1,
fmt='{value:.6f}'))
    header = 'Epoch: [{}]'.format(epoch_num)

    lr_scheduler = None
    if epoch_num == 0:
        warmup_factor = 1. / 1000
```

```
        warmup_iterations = min(1000, len(loader) - 1)

        lr_scheduler = utils.warmup_lr_scheduler(opt, warmup_iterations,
warmup_factor)

    for imgs, tgts in metrics.log_every(loader, print_interval, header):
        imgs = [img.to(device) for img in imgs]
        tgts = [{k: v.to(device) for k, v in t.items()} for t in tgts]

        loss_dict = model(imgs, tgts)

        total_loss = sum(loss for loss in loss_dict.values())

        reduced_loss_dict = utils.reduce_dict(loss_dict)
        reduced_total_loss = sum(loss for loss in reduced_loss_dict.values())

        loss_value = reduced_total_loss.item()

        if not math.isfinite(loss_value):
            print("Loss is {}, stopping training".format(loss_value))
            print(reduced_loss_dict)
            sys.exit(1)

        opt.zero_grad()
        total_loss.backward()
        opt.step()

        if lr_scheduler is not None:
            lr_scheduler.step()

        metrics.update(loss=reduced_total_loss, **reduced_loss_dict)
        metrics.update(learning_rate=opt.param_groups[0]["lr"])
```