МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных систем

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Допущен к защите

Заведующий кафедрой ПМИ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Разова

**Распознавание асан из йоги на изображении с помощью методов машинного обучения**

Курсовой проект по дисциплине  
«Машинное обучение»

Выполнил студент группы ПИм-1301-03-00     / Д.Ф. Бояринцев /

Руководитель к.ф.-м.н., доцент кафедры ПМИ     / В.А. Бызов /

Работа защищена с оценкой     \_\_\_\_.\_\_\_\_.2025 г.

Члены комиссии:     /     /

    /     /

Киров 2025

**Содержание**

Введение 4

1 Теоретические основы распознавания асан йоги 7

1.1 Классификация асан йоги и основы йоги, релевантные для автоматического распознавания 7

1.2 Описание популярных асан и их особенностей 9

1.3 Важность правильного выполнения асан в йоге 12

1.4 Методы распознавания изображений и поз 14

1.5 Традиционные методы обработки изображений в контексте распознавания асан йоги 15

1.6 Применение нейронных сетей и глубокого обучения для распознавания поз 17

1.7 Анализ существующих приложений и систем для распознавания асан йоги 19

1.8 Выводы по разделу 21

2 Разработка системы распознавания асан 23

2.1 Сбор и подготовка данных 23

2.1.1 Источники данных 23

2.1.2 Обзор моделей построения скелетных моделей 24

2.1.3 Создание датасетов и предобработка данных 27

2.2 Модели машинного обучения для многоклассовой классификации 30

2.3 Метрики качества в задачах многоклассовой классификации 35

2.4 Обучение моделей 37

Библиографический список 41

Заключение 42

Приложения 43

Приложение А. Листинг программы создания датасета на основе ключевых точек с использованием MediaPipe 43

Приложение Б. Листинг программы создания датасета на основе ключевых углов с использованием MediaPipe 46

Приложение В. Листинг программы создания датасета на основе ключевых точек с использованием YOLO 48

Приложение Г. Листинг программы создания датасета на основе ключевых углов с использованием YOLO 50

Введение

**Актуальность темы работы**

Йога приобретает все большую популярность как средство физического и психического оздоровления. Возрастает популярность интернет-курсов, не дающих возможности обратной связи о качестве выполнения асан. Однако неправильное выполнение асан может привести к травмам и снижению эффективности практики. В связи с этим возникает необходимость в автоматизированных системах, способных анализировать позы человека и предоставлять обратную связь в реальном времени. Машинное обучение и компьютерное зрение открывают новые возможности для решения этой задачи, что делает тему исследования актуальной как с научной, так и с практической точек зрения.

***1. Рост интереса к йоге и необходимость контроля выполнения асан***

Согласно исследованиям ВОЗ, йога способствует снижению стресса, улучшению гибкости и укреплению мышц. Однако новички часто допускают ошибки при выполнении асан из-за недостатка опыта или отсутствия квалифицированного инструктора. Автоматизированное распознавание поз на основе машинного обучения может стать доступной альтернативой персональным тренерам, особенно в условиях дистанционного обучения и домашних тренировок.

***2. Развитие технологий компьютерного зрения и deep learning***

Современные нейронные сети, такие как PoseNet, OpenPose и MediaPipe, позволяют с высокой точностью детектировать ключевые точки человеческого тела. Это создает основу для классификации асан йоги с использованием алгоритмов сверточных нейронных сетей (CNN), трансформеров или методов few-shot learning. Актуальность исследования обусловлена необходимостью адаптации этих методов для специфических поз йоги, которые часто требуют анализа не только положения конечностей, но и баланса, углов изгибов и симметрии.

***3. Применение в фитнес-технологиях и telemedicine***

Системы распознавания асан могут быть интегрированы в мобильные приложения для йоги (например, YogaAI, Down Dog), предоставляющие пользователям персонализированные рекомендации; умные тренажеры и AR/VR-системы, улучшающие иммерсивность тренировок; телемедицинские платформы, где контроль правильности выполнения упражнений важен для реабилитации пациентов.

***4. Научная новизна и перспективы развития***

Несмотря на существующие решения, остаются нерешенные проблемы, такие как ограниченные датасеты, что выражается в нехватке размеченных данных для редких асан; учет индивидуальных анатомических особенностей человека, таких как гибкость, пропорции тела; распознавание динамических переходов между асанами и др.

Таким образом, разработка систем распознавания асан йоги на основе машинного обучения отвечает запросам современного общества, сочетая здоровье сберегающие технологии, искусственный интеллект и цифровизацию спорта. Дальнейшие исследования в этой области могут способствовать созданию более точных и доступных инструментов для персонального фитнеса и профилактической медицины.

**Цель работы** состоит в разработке системы для автоматического распознавания асан йоги с использованием методов машинного обучения.

**Задачи работы:**

* анализ существующих методов распознавания поз;
* сбор и обработка данных (изображения с асанами);
* выбор и обучение моделей машинного обучения;
* оценка точности и эффективности разработанной системы.

**Объект и предмет исследования**

* ***Объект***: процесс распознавания асан йоги.
* ***Предмет***: методы машинного обучения для классификации изображений.

**Методы исследования**

* Анализ литературы.
* Эксперименты с моделями машинного обучения.
* Оценка результатов с использованием метрик точности.

**Содержание работы**

Работа состоит из введения, двух разделов,

1 Теоретические основы распознавания асан йоги

1.1 Классификация асан йоги и основы йоги, релевантные для автоматического распознавания

Асана (санскр. आसन – сидячая поза, место для сидения, поза) – название определённого неподвижного положения тела в йоге, принимаемого на несколько секунд или минут с целью телесного и душевного оздоровления.

Каждая асана – это комбинация положения суставов (углы сгиба/разгиба), распределения веса (опорные точки) и распределения мышечного напряжения (визуальные маркеры активации).

В процессе выполнения асан возникают кинематические цепи, характеризующие взаимосвязь частей тела. Например, в Уттанасане наклон зависит от гибкости подколенных сухожилий и позвоночника. Кроме этого, большое значение имеют дыхательные паттерны, поскольку фазы вдоха/выдоха коррелируют с динамическими переходами при смене поз и статическим удержанием поз.

Существуют различные классификации асан йоги. Некоторые из них оказываются более подходящими для автоматического распознавания. Оптимальной для машинного обучения является классификация по биомеханическим параметрам (см. табл. 1), поскольку она основана на количественно измеримых признаках.

Таблица 1 – Классификация асан по биомеханическим параметрам

| **Группа** | **Критерии классификации** | **Примеры асан** | **Методы распознавания** |
| --- | --- | --- | --- |
| ***Стоячие асаны*** | * Угол в коленях (90°-180°) * Положение таза относительно стоп | Вирабхадрасана I-II, Уткатасана | OpenPose + анализ углов суставов |
| ***Наклоны вперед*** | * Угол между торсом и ногами * Положение позвоночника (округлый/прямой) | Уттанасана, Пашчимоттанасана | CNN + LSTM для динамики |
| ***Прогибы назад*** | * Градус изгиба позвоночника * Распределение нагрузки на поясницу | Бхуджангасана, Урдхва Дханурасана | 3D реконструкция позы |
| ***Скручивания*** | * Угол ротации позвоночника * Положение плеч относительно таза | Ардха Матсиендрасана, Бхарадваджасана | Трансформеры для анализа асимметрии |
| ***Балансовые асаны*** | * Смещение центра тяжести * Опора на 1-2 конечности | Врикшасана, Бакасана | IMU-датчики + анализ стабильности |

Классификация асан по биомеханическим показателям позволяет формализовать асаны через числовые параметры (углы, расстояния), что критично для алгоритмов.

Второй, релевантной для автоматического распознавания, классификацией является классификация по сложности распознавания асан (см. табл. 2). Сложность распознавания рассматривается в аспекте разработки алгоритмов, то есть в процессе классификации учитываются решаемые вычислительные задачи.

Таблица 2 – Классификация асан по сложности распознавания

| **Уровень сложности** | **Характеристики** | **Примеры асан** | **Проблемы ML** |
| --- | --- | --- | --- |
| **Простой** | * Минимум ключевых точек * Статичная поза | Тадасана, Сукхасана | Шум в данных, окклюзии рук |
| **Средний** | * Динамические элементы * Нестандартные углы | Адхо Мукха Шванасана, Уттхита Триконасана | Точность определения мелких суставов |
| **Сложный** | * Полные окклюзии * 3D-пространственная конфигурация | Саламба Ширшасана, Эка Пада Раджакапотасана | Необходимость мультимодальных данных |
| **Экстремаль-ный** | * Быстрые переходы * Неочевидные опорные точки | Аштавакрасана, Натараджасана | Требуется temporal моделирование (LSTM/3D-CNN) |

1.2 Описание популярных асан и их особенностей

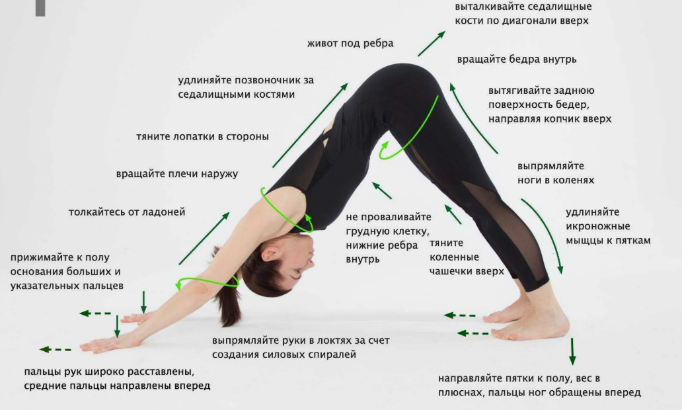
**1. Воин (Вирабхадрасана)**

Выполнение

1. Встаньте, широко расставив ноги.
2. Разверните правую стопу вправо на 90°, левую слегка внутрь.
3. Согните правое колено, чтобы бедро было параллельно полу.
4. Руки вытяните в стороны, взгляд направлен за правую ладонь.

Эффект: укрепляет ноги, раскрывает таз, улучшает баланс.

**2. Собака мордой вниз (Адхо Мукха Шванасана)**

Выполнение: из положения на четвереньках поднимите таз вверх, выпрямляя ноги и руки, спина прямая, копчик тянется назад и вверх, пятки стремятся к полу.

Эффект: растягивает заднюю поверхность ног, укрепляет руки и плечи, снимает напряжение в спине.

**3. Планка (Кумбхакасана или Пхалакасана)**

Выполнение: из положения на четвереньках вытяните ноги назад, опираясь на ладони (или предплечья) и носки. Тело образует прямую линию от головы до пяток, пресс и мышцы кора напряжены.

Эффект: укрепляет руки, кор, спину и ноги, улучшает выносливость.

**4. Дерево (Врикшасана)**

Выполнение:

1. Встаньте прямо, перенесите вес на левую ногу.
2. Правую стопу поставьте на внутреннюю часть левого бедра (или голени, но не на колено).
3. Руки сложите в намасте перед грудью или поднимите вверх.

Эффект: развивает баланс, концентрацию, укрепляет ноги и спину.

**5. Богиня (Уткхата Конасана или Девиасана)**

Выполнение:

1. Широко расставьте ноги, разверните стопы наружу.
2. Согните колени, опуская таз до уровня параллели бедер с полом.
3. Руки можно поднять вверх, согнуть в локт ях или развести в стороны.

Эффект: укрепляет ноги и таз, раскрывает бедра, улучшает осанку.

Эти асаны можно включать в разные последовательности для развития силы, гибкости и баланса.

1.3 Важность правильного выполнения асан в йоге

Правильное выполнение асан – основа безопасной и эффективной практики йоги. От точности положения тела зависит не только физический результат, но и воздействие на энергетические каналы (нади) и внутренние органы.

Правильное выполнение асан йоги обеспечивает профилактику травм, так как неверное положение суставов или перегрузка мышц может привести к растяжениям, болям в спине и другим повреждениям. Например, в собаке мордой вниз важно держать спину прямой, а не округлять её, чтобы не перегружать поясницу.

При правильном выполнении асан йоги достигается максимальная польза от асаны, так как каждая поза направлена на определённые группы мышц и внутренние процессы. Например, планка, выполненная с правильным выравниванием, укрепляет кор (совокупность мышц, отвечающих за стабилизацию позвоночника, таза и всего тела), а не просто напрягает руки.

В йоге дыхание (пранаяма) связано с движением. Если асана выполняется неправильно, дыхание сбивается, и эффект снижается. Например, в позе дерева баланс зависит от спокойного, ровного дыхания.

Правильное выполнение асан йоги требует осознанности и концентрации.Четкое следование технике развивает внимательность к телу, что усиливает медитативный эффект практики.

Распознавание асан йоги с помощью методов машинного обучения, развивающееся на стыке компьютерного зрения, бионики и wellness-технологий, позволяет значительно повысить эффективность занятий йогой.

Во-первых, оно позволяет получить персонализированную обратную связь для практикующих и провести коррекцию ошибок в реальном времени (Алгоритмы анализируют положение тела (например, угол сгиба колена в Воин II или прогиб в Собаке мордой вверх) и указывают на неточности выполнения). Кроме этого, возможна адаптация асан для разных уровней подготовленности, то есть система может предлагать упрощённые или усложненные варианты асан новичков в йоге или усложнённые – для продвинутых.

Во-вторых, возможность автоматического распознавания асан способствует повышению безопасности при выполнении практики и предотвращению травм, так как ML-модели, обученные на данных физиотерапевтов способны предупреждать о рисках (например, перегрузке коленей или поясницы). Кроме этого, контроль динамических переходов помогает избегать резких движений.

В-третьих, мобильные приложения с дополненной реальностью демонстрируют правильное выполнение асан, даже если нет тренера, что естественно повышает доступность йоги для новичков. А системы баллов за точность выполнения поз мотивируют к регулярной практике.

В-четвертых, рассматриваемая в работе тематика находит медицинские применения, так как в сочетании с данными датчиков машинное обучение помогает восстанавливаться после травм, подбирая асаны без нагрузки на повреждённые зоны.

В-пятых, автоматизация для инструкторов позволяет проводить качественный анализ групповых занятий, так как камеры могут отслеживать одновременное выполнение асан несколькими учениками, выделяя тех, кому нужна помощь.

1.4 Методы распознавания изображений и поз

Распознавание асан йоги с использованием методов машинного обучения и компьютерного зрения базируется на нескольких ключевых теоретических принципах.

***1. Принцип пространственно-временного анализа позы,*** основанный на методах компьютерного зрения и машинного обучения и включающий в себя детекцию ключевых точек тела, анализ их взаимного расположения, трекинг изменений позы во времени.

***2. Принцип эталонного сравнения,*** в соответствии с которым каждая асана описывается как набор эталонных параметров (углы в основных суставах, относительные расстояния между точками тела, пропорции и симметрия), а сравнение выполняется с использованием метрик подобия (евклидово расстояние, косинусная мера).

***3. Принцип иерархической классификации,*** заключается в многоуровневой системе распознавания: группы асан (стоячие, сидячие, перевернутые); конкретные асаны внутри групп; вариации выполнения.

***4. Принцип адаптивности к индивидуальным особенностям***, который требует от исследователя учета антропометрических параметров человека, например, таких как, длина конечностей, гибкость суставов, ограничения подвижности.

***5. Принцип обратной связи в реальном времени,*** который заключается в непрерывном анализе и коррекции выполнения поз йоги. Это может быть и визуальная индикация ошибок, и аудиоподсказки, и тактильная обратная связь.

***6. Принцип устойчивости к помехам,*** предполагающий качественную обработку частичных окклюзий (заслонения частей тела), изменений освещения, разных ракурсов съемки процесса выполнения асан.

Основанием для решения задачи автоматического распознавания асан йоги и правильности их выполнения являются методы компьютерного зрения (OpenPose, MediaPipe), глубокое обучение (CNN, LSTM, Transformers), теория обработки сигналов (для мультимодальных систем), принципы эргономики и биомеханики (для оценки правильности поз).

1.5 Традиционные методы обработки изображений в контексте распознавания асан йоги

Традиционные методы обработки изображений играют ключевую роль в задачах компьютерного зрения, включая распознавание поз человека, в том числе и асан йоги. Долгое время они являлись основным инструментом.

Эти методы основаны на математических и алгоритмических подходах, которые позволяют выделять значимые признаки из изображений без использования глубокого обучения, и имеют некоторые преимущества и некоторые ограничения по сравнению с современными методами машинного обучения.

Рассмотрим ключевые этапы традиционного подхода, их применение для распознавания асан, а также преимущества и ограничения по сравнению с современными методами глубокого обучения.

1. **Этап предварительной обработки изображений.** На этом этапе перед извлечением признаков изображения проводится нормализация для улучшения качества данных.

Основные методы включают:

1. *Фильтрацию шумов* с помощью Гауссова фильтра и медианного фильтра, вызванных плохим освещением или артефактами сжатия (Gonzalez & Woods, 2018).
2. *Бинаризацию* с использованием пороговых методов(например, метод Оцу) для сегментации тела от фона (Bradski & Kaehler, 2008).
3. *Коррекцию освещения* с помощью алгоритмов CLAHE (Contrast Limited Adaptive Histogram Equalization) для улучшения контрастности изображения, что особенно важно для работы в условиях переменного освещения (Zuiderveld, 1994).
4. **Этап извлечение признаков**

Для распознавания поз очень важны методы, позволяющие выделить ключевые точки и контуры тела. Эти методы разработаны достаточно давно.

Для детекции краёв, то есть выделения границ тела и его частей применяются операторы Собеля, Кэнни и Лапласиан Гаусса (Canny, 1986).

Для устранения разрывов в контурах и улучшения качества сегментации разработаны, так называемые морфологические операции, а именно расширение и эрозия (Serra, 1982).

Метод HOG (Гистограммы ориентированных градиентов) кодирует ориентацию градиентов интенсивности пикселей, что позволяет описывать форму тела (Dalal & Triggs, 2005).

Дескрипторы, описывающие Устойчивые локальные признаки (SIFT, SURF), используются для обнаружения ключевых точек, устойчивых к изменениям масштаба и поворота (Lowe, 2004; Bay et al., 2008).

1. **Этап классификации поз**

После извлечения признаков применяются алгоритмы машинного обучения для классификации асан:

1. метод опорных векторов (SVM), который эффективен для разделения классов на основе признаков HOG или SIFT (Cortes & Vapnik, 1995);
2. алгоритм Random Forest, основанный на ансамбле решающих деревьев, используется для классификации поз с учетом множества признаков (Breiman, 2001);
3. простой, но эффективный метод метод k-ближайших соседей (k-NN) для сравнения новых данных с эталонными позами (Cover & Hart, 1967).

Несмотря на свою эффективность, традиционные методы имеют ряд ограничений:

* зависимость от качества изображения, на котором возможно имеется перекрытие частей тела, плохое освещение или сложный фон, что значительно снижает точность распознавания;
* необходимость ручной настройки параметров на каждом этапе обработки, что усложняет процесс;
* низкая устойчивость к вариациям, например, ракурса, масштаба, или наличию нестандартных поз (Forsyth & Ponce, 2011).

Традиционные методы остаются актуальными для задач с ограниченными вычислительными ресурсами или малыми наборами данных. Их точность и устойчивость уступают современным методам глубокого обучения, которые способны автоматически извлекать признаки и адаптироваться к различным условиям.

1.6 Применение нейронных сетей и глубокого обучения для распознавания поз

Глубокое обучение значительно активизировало область распознавания изображений и поз. Оно позволяет автоматически извлекать признаки и классифицировать позы с высокой точностью, что делает его идеальными для распознавания асан йоги.

Перечислим некоторые из методов глубокого обучения, применяемых для распознавания асан.

* 1. ***Свёрточные нейронные сети (CNN)*** – это основной инструментом для анализа изображений. Их архитектура включает:
* свёрточные слои для автоматического извлечения локальных признаков, таких как края и текстуры (LeCun et al., 1998);
* пулинговые слои для уменьшения размерностьи данных, сохраняя важные признаки;
* полносвязные слои для классификации поз на основе извлечённых признаков.
  1. Для точного распознавания поз используются методы, такие как:
* *OpenPose,* который обнаруживает ключевые точки тела, включая суставы, что позволяет точно определять позы (Cao et al., 2017).
* *PoseNet,* которая предсказывает позиции ключевых точек в реальном времени (Kendall et al., 2015).
  1. предобученные с помощью методов трансферного обучения модели (например, ResNet, VGG и EfficientNet) часто используются для распознавания асан (трансферное обучение позволяет адаптировать эти модели к конкретной задаче, даже при ограниченном объёме данных) (He et al., 2016).
  2. Для анализа видео применяются Рекуррентные нейронные сети (RNN), которые обрабатывают последовательности кадров, улучшая точность распознавания (Graves et al., 2013), двухпоточные архитектуры, комбинирующие CNN для анализа отдельных кадров и RNN для учёта временных зависимостей (Simonyan & Zisserman, 2014).

Глубокое обучение позволяет выполнять извлечение признаков без использования ручной настройки. Современные модели превосходят традиционные методы по точности и устойчивости, а также они могут быть адаптированы для работы с различными типами данных.

Таким образом, глубокое обучение предлагает мощные инструменты для распознавания асан йоги, обеспечивая высокую точность и устойчивость к изменениям условий. Однако для их эффективного применения требуются большие объёмы размеченных данных и значительные вычислительные ресурсы.

1.7 Анализ существующих приложений и систем для распознавания асан йоги

В последние годы появилось множество мобильных приложений и облачных систем, использующих методы компьютерного зрения и машинного обучения для распознавания асан йоги. Эти решения варьируются от простых трекеров поз до сложных платформ с обратной связью в реальном времени.

Рассмотрим некоторые популярные приложения, их функциональность, применяемые алгоритмы, а также преимущества и недостатки.

* 1. **YogaGuru** (2022) – Мобильное приложение с AR-подсказками, использующее PoseNet и MediaPipe для трекинга поз.

Основные функции: распознавание 50+ асан в реальном времени, AR-визуализация правильного положения тела, оценка точности выполнения (точность до 85%).

Методы: MediaPipe Pose для детекции ключевых точек (Lugaresi et al., 2021), LightGBM для классификации асан на основе углов суставов.

*Преимущества*:

* Низкие требования к железу (работает на mid-range смартфонах).
* Поддержка оффлайн-режима.

*Недостатки*:

* Плохая работа при сложных ракурсах (например, перевернутые асаны).
* Нет учета индивидуальных анатомических особенностей
  1. **Wimu Yoga** (2023) – профессиональная система с датчиками движения и камерами для студий йоги.

Основные функции: точный трекинг 3D-позы с использованием IMU-датчиков, анализ дыхания через микрофон, интеграция с VR для виртуальных тренировок.

Методы: гибридный подход: OpenPose + данные сенсоров (Zhang et al., 2022), LSTM для анализа временных последовательностей.

*Преимущества*:

* Высокая точность (93–95%).
* Поддержка динамических переходов между асанами.

*Недостатки*:

* Требуется дорогое оборудование (камеры глубины, датчики).
* Сложность настройки.
  1. **AI Yoga Coach** (2023) – облачное решение для фитнес-клубов с AI-тренером.

Основные функции: распознавание 100+ асан, включая редкие, генерация персонализированных программ, анализ риска травм на основе биомеханики.

Методы: Transformer-модели для обработки видео (Dosovitskiy et al., 2021), GNN (Graph Neural Networks) для анализа скелетных данных.

*Преимущества***:**

* Автоматическая адаптация под уровень пользователя.
* Интеграция с фитнес-браслетами.

*Недостатки***:**

* Задержка в 1–2 секунды из-за облачной обработки.
* Подписка стоит от $20/месяц.

Таблица 3 – Сравнительный анализ методов

| **Критерий** | **Традиционные (MediaPipe)** | **Гибридные (IMU + CV)** | **Глубокое обучение (Transformers)** |
| --- | --- | --- | --- |
| **Точность** | 80–85% | 90–95% | 88–93% |
| **Скорость** | Реалтайм | Реалтайм | Задержка 1–3 сек. |
| **Требования** | Смартфон | Датчики + камеры | Облачный сервер |
| **Стоимость** | Бесплатно | $$$ | $$ |
| **Гибкость** | Ограничена | Высокая | Максимальная |

При использовании этих приложений наблюдаются следующие проблемы.

1. Окклюзии – перекрытие частей тела (например, в скручиваниях) снижает точность Pure-CV решений (Chen et al., 2023).
2. Индивидуальные различия – антропометрия и гибкость пользователей требуют калибровки моделей.
3. Ресурсоемкость – 3D-трекинг в реальном времени потребляет до 5 Вт энергии (Lee et al., 2022).

Современные системы демонстрируют высокую точность в идеальных условиях, но сталкиваются с вызовами при работе с нестандартными позами, пользователями с ограниченной подвижностью, реальным освещением и фонами.

1.8 Выводы по разделу

В данном разделе приведены теоретические основы распознавания асан йоги, которые представляют собой сложные биомеханические структуры и могут быть формализованы через ключевые точки, отрезки, углы между ними и расстояния. Классификация асан йоги по биомеханическим параметрам наиболее подходит для автоматического распознавания, так как опирается на приведенные количественно измеряемые признаки.

Также в данном разделе значительное внимание уделяется традиционным методам распознавания позы человека на изображении, и рассмотрены некоторые аспекты применения нейронных сетей и глубокого обучения к решению поставленной задачи.

Сложность распознавания асан йоги варьируется от распознавания простых статических поз до экстремальных динамических позиций, требующих мультимодальных данных и временного моделирования.

В заключение в первом разделе приведена сравнительная характеристика существующих приложений и систем для распознавания асан йоги.

Для дальнейшей работы выбраны пять асан йоги, которые имеют четкие критерии правильного выполнения, и могут быть использованы для автоматической оценки.

В следующей главе будет описана кинематическая модель – это модель на основе скелета, которая представляет человеческое тело. На основе этой модели будет реализован SVM-классификатор для определения позы человека на изображении с использованием датасета YOGA.

2 Разработка системы распознавания асан

2.1 Сбор и подготовка данных

2.1.1 Источники данных

Существуют разные наборы данных с изображениями асан. Они отличаются числом изображений, числом разных асан, представленных в наборе, качеством изображений. Приведем характеристики основных наборов данных.

1. *Yoga Poses Dataset* (<https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset/data>). Набор данных содержит 1551 файл с изображениями, которые разделены на подкаталоги train и test, с пятью подкаталогами в каждом каталоге, соответствующими пяти классам поз йоги (собака мордой вниз, богиня, планка, дерево, воин2). Изображения были извлечены из bing с использованием их API-функций, поэтому они не очень точны (на некоторых присутствуют водяные знаки, текст). Объем датасета 316,23 Mb.
2. *Yoga-82* [] (<https://sites.google.com/view/yoga-82/home>). Данный датасет предназначен для крупномасштабного распознавания асан йоги с 82 классами. Yoga-82 состоит из сложных поз, для которых точные аннотации могут быть невозможны. Чтобы решить эту проблему, авторы предоставили иерархические метки для поз йоги на основе конфигурации тела в позе. Набор данных содержит трехуровневую иерархию, включая положения тела (6 классов асан: стоячие, сидячие, балансирующие, перевернутые, откидывающиеся, колесо), вариации положений тела (20 классов) и фактические названия поз (82 класса). В датасете хранятся адреса изображений, метки классов трехуровневой иерархии, названия асан.
3. *Yoga Pose Image classification dataset* (<https://www.kaggle.com/datasets/shrutisaxena/yoga-pose-image-classification-dataset/data>). В данном датасете содержится 5991 файл с изображением асан йоги, разбитых на 107 классов. Объем датасета 1.11 Gb.

2.1.2 Обзор моделей построения скелетных моделей

Для построения классификатора на основе моделей машинного обучения была выбрана за основу скелетная модель определения позы – модель, основанная на извлечении ключевых точек, лежащих в основе построения графического скелета.

*Графический скелет* – набор координат, описывающих позу человека.

Каждый сустав – это отдельная координата, которая известна как ключевая точка или поза-ориентир. А связь между ключевыми точками известна как пара.

Существует несколько моделей оценки позы человека.

1. *DeepPose* это одна из первых моделей для оценки позы человека, основанная на глубоком обучении. Она была предложена исследователями из Google в 2014 году и стала важным шагом в переходе от традиционных методов (например, Deformable Part Models) к нейросетевым подходам. Использует сверточную нейронную сеть (CNN) для прямой регрессии координат суставов тела из изображения. Модель предсказывает положения ключевых точек в формате (*x*, *y*) как единую задачу регрессии. Модель обладает невысокой точностью.
2. *OpenPose* – один из самых популярных методов оценки позы человека для нескольких человек. Это многопользовательское обнаружение в реальном времени с открытым исходным кодом и высокоточными ключевыми точками.
3. *PoseNet* построен на Tensorflow.js. Это легкая архитектура, которая работает на мобильных устройствах и браузерах. Модель оптимизирована по скорости, но уступает по точности OpenPos.
4. *BlazePose* – это высокооптимизированная модель для 3D-оценки позы человека, разработанная Google в рамках фреймворка MediaPipe. Она предназначена для работы в реальном времени даже на мобильных устройствах и в браузерах. Модель сочетает высокую скорость с детализированным выводом 33 ключевых точек тела, включая 3D-координаты и углы поворота суставов.
5. *YOLO-Pose* (или YOLO + Pose Estimation) – это гибридная модель, объединяющая детекцию объектов (YOLO) и оценку позы человека (Pose Estimation) в единый фреймворк. В отличие от классических подходов (где сначала детектируют человека, а затем оценивают его позу), YOLO-Pose делает это за один проход нейросети, что ускоряет обработку и упрощает pipeline.
6. *HRNet* (High-Resolution Network) – это передовая архитектура свёрточной нейронной сети (CNN), специально разработанная для задач точной локализации ключевых точек, таких как оценка позы человека (Human Pose Estimation), сегментация объектов и другие задачи компьютерного зрения, требующие сохранения пространственных деталей. Представленная в 2019 году, HRNet стала золотым стандартом для 2D-оценки позы благодаря своей уникальной способности сохранять высокое разрешение на всех этапах обработки изображения.

Наиболее часто используемые системы при оценке поз – семейство моделей YOLO и фреймворк MediaPipe.

YOLO расшифровывается как «You Only Look Once» («Стоит только раз взглянуть»). YOLO – современный алгоритм глубокого обучения, который широко используется для обнаружения объектов. Он был разработан Джозефом Редмоном и Али Фархади в 2016 году [].

Модели семейства YOLO, предназначенные для оценки поз человека, способны детектировать 17 ключевых точек скелета, что соответствует разметке в наборе данных COCO (см. рис. 1).



Рисунок 1 – Ключевые точки моделей семейства YOLO

MediaPipe – это фреймворк с открытым исходным кодом от Google. Это очень легкое многоплатформенное фреймворк-решение для машинного обучения, которое может работать в режиме реального времени на CPU.

MediaPipe использует TensorFlow lite. Используя детектор, сначала находит человека (ROI) в кадре. Затем он использует кадрированный ROI в качестве входа и прогнозирует ориентиры/ключевые точки в ROI. Оценщик позы MediaPipe обнаруживает в общей сложности 33 ключевые точки (см. рис. 2).

|  |  |  |
| --- | --- | --- |
| точки тела | 0 – нос  1 – левый глаз внутренний угол  2 – левый глаз  3 – левый глаз внешний угол  4 – правый глаз внутренний угол  5 – правый глаз  6 – правый глаз внешний угол  7 – левое ухо  8 – правое ухо  9 – рот левый угол  10 – рот правый угол  11 – левое плечо  12 – правое плечо  13 – левый локоть  14 – правый локоть  15 – левое запястье  16 – правое запястье  17 – левый мизинец | 18 – правый мизинец  19 – левый указательный палец  20 – правый указательный палец  21 – левый большой палец  22 – правый большой палец  23 – левая тазовая часть бедра  24 – правая тазовая часть бедра  25 – левое колено  26 – правое колено  27 – левая лодыжка  28 – правая лодыжка  29 – левая пятка  30 – правая пятка  31 – указательный палец левой стопы  32 – указательный палец правой стопы |

Рисунок 2 – Ключевые точки позы человека, определяемые MediaPipe

Оценка позы MediaPipe – это единый 3D-оценщик позы. Он определяет координаты *x*, *y* и *z* для каждого ориентира. Ось *Z* – это глубина ориентира, который означает, насколько далеко или близко ориентиры находятся от камеры относительно других ориентиров.

2.1.3 Создание датасетов и предобработка данных

Для построения датасета для обучения и исследования моделей машинного обучения классификации асан был использован *Yoga Poses Dataset*, на основе изображений которого было построено пять пар датасетов:

1. dataset\_train\_mediapipe\_full.csv и dataset\_test\_mediapipe\_full.csv, содержащие имя файла, значения *x*, *y*, *z*, *visibility* для каждой из 33 ключевых точек, метку класса (асаны) (target);
2. dataset\_train\_mediapipe\_short.csv и dataset\_test\_mediapipe\_short.csv, содержащие имя файла, значения *x*, *y* для каждой из 33 ключевых точек, метку класса (асаны) (target);
3. dataset\_train\_mediapipe\_angle.csv и dataset\_test\_mediapipe\_angle.csv, содержащие имя файла, значения 8 углов (угол левого локтя, угол правого локтя, угол левой руки, угол правой руки, угол левого колена, угол правого колена, угол левой ноги, угол правой ноги), метку класса (асаны) (target).
4. dataset\_train\_yolo.csv и dataset\_test\_yolo.csv, содержащие имя файла, значения *x*, *y* для каждой из 17 ключевых точек, метку класса (асаны) (target);
5. dataset\_train\_yolo\_angle.csv и dataset\_test\_yolo\_angle.csv, содержащие имя файла, значения 8 углов (угол левого локтя, угол правого локтя, угол левой руки, угол правой руки, угол левого колена, угол правого колена, угол левой ноги, угол правой ноги), метку класса (асаны) (target).

Листинги программ по созданию датасетов содержаться, соответственно, в Приложениях А, Б, В, Г.

При построении обучающего и тестового набора данных не для всех изображений модель MediaPipe смогла обнаружить и построить скелетную модель, это связано в том числе и с тем, что MediaPipe предназначен для обнаружения одного человека в кадре, а на некоторых изображениях присутствуют по два человека.

В таблицах 4, 5 и на рисунках 3, 4 приведены характеристики исходного набора изображений и набора данных ключевых точек, построенного с помощью MediaPipe.

Таблица 4 – Статистика построения с помощью MediaPipe скелетных моделей изображений асан обучающего набора изображений

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип асаны** | **Всего изображений** | **Распознанных изображений** | **Процент нераспознанных изображений** |
| downdog | 223 | 202 | 9% |
| goddess | 180 | 160 | 11% |
| plank | 266 | 226 | 15% |
| tree | 159 | 140 | 12% |
| warrior2 | 252 | 235 | 7% |
| **Всего** | **1080** | **963** | **11%** |

Таблица 5 – Статистика построения с помощью MediaPipe скелетных моделей изображений асан тестового набора изображений

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип асаны** | **Всего изображений** | **Распознанных изображений** | **Процент нераспознанных изображений** |
| downdog | 97 | 91 | 6% |
| goddess | 80 | 76 | 5% |
| plank | 115 | 102 | 11% |
| tree | 69 | 62 | 10% |
| warrior2 | 109 | 104 | 5% |
| **Всего** | **470** | **435** | **7%** |

Рисунок 3 – Диаграмма распределения по классам результатов построения с помощью MediaPipe скелетных моделей изображений асан обучающего набора изображений

Рисунок 4 – Диаграмма распределения по классам результатов построения с помощью MediaPipe скелетных моделей изображений асан тестового набора изображений

2.2 Модели машинного обучения для многоклассовой классификации

Многоклассовая классификация – одна из ключевых задач машинного обучения, требующая предсказания принадлежности объекта к одному из трёх и более классов. Для её решения применяются различные модели, каждая из которых имеет свои преимущества и ограничения.

**Метод опорных векторов (SVM)** с адаптацией для многоклассовой классификации использует стратегии «один против одного» (One-vs-One) или «один против всех» (One-vs-All). SVM эффективен в задачах с линейно разделимыми классами, а с ядерными функциями (например, RBF) может работать и с нелинейными данными [1]. Однако его производительность снижается на больших выборках.

**Деревья решений и ансамбли (Random Forest, Gradient Boosting)** часто применяются благодаря интерпретируемости и устойчивости к переобучению. Random Forest агрегирует предсказания множества деревьев, уменьшая дисперсию, а Gradient Boosting (XGBoost, LightGBM) минимизирует ошибку последовательно, что улучшает точность [2]. Эти методы хорошо работают с разнородными данными, но требуют тонкой настройки гиперпараметров.

**Нейронные сети**, особенно глубокие (DNN), способны автоматически извлекать сложные признаки, что делает их мощным инструментом для многоклассовой классификации. Архитектуры на основе свёрточных (CNN) или трансформеров (например, BERT для текста) показывают высокие результаты, но требуют больших вычислительных ресурсов и данных для обучения [3].

**Логистическая регрессия с расширением (Softmax)** – простой, но эффективный метод для линейно разделимых классов. Softmax-регрессия обобщает бинарную классификацию на множественные классы, минимизируя кросс-энтропию [4]. Её преимущество – скорость и интерпретируемость коэффициентов.

**Переход от бинарной к многоклассовой классификации**

Деревья и ансамбли изначально поддерживают многоклассовость, но и большинство бинарных методов можно адаптировать для многокласовой классификации с помощью:

* cтратегии «Один против всех» (OvA) – обучение отдельного классификатора для каждого класса;
* cтратегии «Один против одного» (OvO) – обучение парных классификаторов и голосование;
* gрямого многоклассового обобщения (например, Softmax вместо сигмоиды).

Современные библиотеки (например, TensorFlow 2.15, PyTorch 2.0) автоматизируют процесс перехода от бинарной к многоклассовой классификации (2023–2024).

* Стратегия OvO применяется в SVM по умолчанию (оптимизирована для работы с разреженными данными).
* Встроенная многоклассовая поддержка в ансамблевых методах (например, OneVsRestClassifier в scikit-learn).
* Нейросети используют категориальную кросс-энтропию с Softmax, а для несбалансированных данных – Focal Loss [9].

Многоклассовая классификация требует выбора модели, способной эффективно разделять объекты между тремя и более классами. Выбор модели зависит от специфики данных, объёма выборки и требований к интерпретируемости. Например, для структурированных данных часто используют ансамбли, а для изображений или текста – нейронные сети, SVM чаще использует OvO (меньше перекосов в несбалансированных данных), логистическая регрессия и нейросети применяют Softmax.

Рассмотрим ключевые методы, их преимущества, недостатки и способы адаптации от бинарной к многоклассовой классификации (см. табл. 6).

Таблица 6 – Сравнительная характеристика моделей многоклассовой классификации

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Модель** | **Особенности** | **Достоинства** | **Недостатки** | **Особенности современной реализации** |
| **Метод опорных  векторов (SVM)** | * Изначально разработан для бинарной классификации, но расширяется на многоклассовый случай с помощью стратегий: * «Один против всех» (One-vs-All, OvA) – строится N классификаторов, каждый отделяет один класс от остальных. * «Один против одного» (One-vs-One, OvO) – создаётся N\*(N-1)/2 классификаторов для всех пар классов, итоговый класс определяется голосованием. * Может использовать ядерные функции (RBF, полиномиальные) для нелинейного разделения. | * Эффективен на данных малой и средней размерности. * Устойчив к переобучению при правильном выборе ядра и регуляризации. | * Медленная работа на больших выборках. * Требует ручного подбора ядра и параметров. | 2023  по-прежнему использует стратегии OvO/OvA, но с оптимизациями для работы с большими данными через методы приближённого решения [1].  Появились гибридные подходы, сочетающие SVM с глубоким обучением (Deep SVM) для автоматического подбора ядер [2]  + Эффективен для данных с чёткими границами классов (например, медицинская диагностика [3]).  - Уступает ансамблям и нейросетям на сложных нелинейных данных. |
| **Деревья решений и ансамбли (Random Forest, Gradient Boosting)** | * Деревья решений автоматически обрабатывают многоклассовые задачи без дополнительных модификаций. * Random Forest использует бэггинг (усреднение предсказаний множества деревьев), снижая переобучение. * Gradient Boosting (XGBoost, LightGBM, CatBoost) последовательно улучшает предсказания, минимизируя ошибку. | * Работают с разнородными данными (числовые, категориальные). * Устойчивы к шумам и выбросам. * Интерпретируемость (особенно у простых деревьев). | * Склонность к переобучению при недостаточной регуляризации (особенно у Gradient Boosting). * Большие ансамбли требуют значительных вычислительных ресурсов. | 2023–2024  LightGBM и XGBoost (версии 3.0+) поддерживают GPU-ускорение и методы обработки категориальных признаков без предварительного кодирования [4].  В CatBoost (2024) улучшена работа с несбалансированными классами через адаптивные веса [5].  + Лучшая интерпретируемость с помощью SHAP-значений (интегрировано в библиотеки).  - Высокие требования к памяти для больших ансамблей. |
| **Логистическая  регрессия с Softmax** | * Softmax-регрессия – обобщение логистической регрессии на множественные классы.   Использует функцию активации Softmax для преобразования выходов в вероятности классов. | * Простота и высокая скорость обучения. * Интерпретируемость (значимость признаков через веса). | * Предполагает линейную разделимость классов. * Плохо работает с нелинейными зависимостями без feature engineering. | 2023  Реализации в scikit-learn 1.4+ используют ускоренные алгоритмы (например, L-BFGS) для больших данных [8].  + Поддержка L1/L2-регуляризации в реальном времени.  - По-прежнему слабая эффективность на нелинейных данных. |
| **Нейронные сети (DNN, CNN,  Transformer)** | * Автоматически извлекают сложные признаки за счёт глубины архитектуры. * Для многоклассовой классификации используют: * Softmax на выходном слое, * Функцию потерь – категориальную кросс-энтропию. | * Высокая точность на сложных данных (изображения, текст, временные ряды). * Гибкость архитектур (например, свёрточные сети для изображений). | * Требуют больших объёмов данных для обучения. * Высокие вычислительные затраты. * Сложность интерпретации (чёрный ящик). | 2022–2024  Трансформеры (например, TabTransformer) адаптированы для табличных данных, показывая точность выше, чем у градиентного бустинга [6].  В CNN для изображений применяются многоголовые механизмы внимания (Multi-Head Attention) даже в традиционных архитектурах [7].  + Автоматическое выделение признаков в Vision Transformer (ViT) и BERT-подобных моделях.  - Требуют дообучения на специфичных датасетах. |

2.3 Метрики качества в задачах многоклассовой классификации

В многоклассовой классификации используются различные метрики для оценки качества моделей. Их выбор зависит от специфики задачи (сбалансированность данных, важность каждого класса, допустимость ошибок).

Рассмотрим ключевые метрики и их интерпретацию.

### *1. Точность (Accuracy)*

Точность показывает долю правильно классифицированных образцов ко всем образцам. Формула для расчета точности выглядит следующим образом:

,

где:

* *TP* (True Positive) – это верно предсказанные положительные примеры,
* *TN* (True Negative) – верно предсказанные отрицательные примеры,
* *FP* (False Positive) – неверно предсказанные положительные примеры,
* *FN* (False Negative) – неверно предсказанные отрицательные примеры.

Точность полезна, когда классы сбалансированы (количество объектов в каждом классе примерно одинаково), но может вводить в заблуждение, если классы сильно не сбалансированы.

### *2. Полнота (Recall)*

Полнота показывает долю положительных случаев, которые были правильно классифицированы моделью. Формула:

**,**

где:

* *TP* (True Positive) – это верно предсказанные положительные примеры,
* *FN* (False Negative) – неверно предсказанные отрицательные примеры.

Высокая полнота важна при дисбалансе классов, если нужно минимизировать пропуски положительных случаев, если важны ложные срабатывания (*FP*) или пропуски (*FN*).

### *3. Precision*

где:

* *TP* (True Positive) – это верно предсказанные положительные примеры,
* *FP* (False Positive) – неверно предсказанные положительные примеры.

### *4. F-мера (F1 Score)*

F1-Score – это гармоническое среднее между точностью и полнотой. Оно полезно, когда важны оба показателя:

Для многоклассовой классификации используется Macro/Micro-averaged F1.

Различия:

Macro-F1: Усреднение F1 по всем классам (все классы равнозначны).

Micro-F1: Учёт общего числа *TP*, *FP*, *FN* (учитывает размеры классов).

### *5. Матрица ошибок (Confusion Matrix)*

Матрица ошибок помогает визуально оценить работу модели, показывая, сколько образцов было правильно и неправильно классифицировано для каждого класса. Она содержит четыре элемента:

|  |  |  |
| --- | --- | --- |
|  | 0 | 1 |
| 0 | TN | FN |
| 1 | FP | TP |

* *TP* (True Positive) – это верно предсказанные положительные примеры,
* *TN* (True Negative) – верно предсказанные отрицательные примеры,
* *FP* (False Positive) – неверно предсказанные положительные примеры,
* *FN* (False Negative) – неверно предсказанные отрицательные примеры.

Выбор метрик зависит от задачи, например, для общей оценки применяют *Accuracy*, *Macro-F1*; дисбаланс оценивается с помощью метрик *Precision/Recall* по классам.

2.4 Обучение моделей

Для сравнительного анализа были выбраны следующие модели классификаторов:

* логистическая регрессия;
* метод опорных векторов;
* K-ближайших соседей;
* дерево решений;
* случайный лес;
* AdaBoost;
* градиентный бустинг.

Для обучения обучающие наборы каждого варианта датасета были разбиты на обучающую и валидационную выборки в соотношении 80% и 20%.

Размеры выборок для MediaPipe:

- Обучающая: 770 примеров

- Валидационная: 193 примеров

Размеры выборок для YOLO:

- Обучающая: 864 примеров

- Валидационная: 216 примеров

Результаты обучения для каждого датасета приведены в таблицах 7, 8, 9, 10, 11 соответственно.

Таблица 7 – Результаты обучения классификаторов на датасете, построенном на ключевых точках, полученных с помощью MediaPipe

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Train Accuracy** | **Validation Accuracy** |
| Метод опорных векторов | 0.898701 | 0.818653 |
| Градиентный бустинг | **1.000000** | 0.818653 |
| Случайный лес | **1.000000** | **0.844560** |
| K-ближайших соседей | 0.854545 | 0.808290 |
| Логистическая регрессия | 0.794805 | 0.813472 |
| Дерево решений | **1.000000** | 0.678756 |
| AdaBoost | 0.577922 | 0.51813 |

Таблица 8 – Результаты обучения классификаторов на сокращенном датасете, построенном на ключевых точках, полученных с помощью MediaPipe

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Train Accuracy** | **Validation Accuracy** |
| Метод опорных векторов | 0.881818 | **0.839378** |
| Градиентный бустинг | **1.000000** | **0.839378** |
| Случайный лес | **1.000000** | 0.834197 |
| K-ближайших соседей | 0.863636 | 0.829016 |
| Логистическая регрессия | 0.788312 | 0.797927 |
| Дерево решений | **1.000000** | 0.715026 |
| AdaBoost | 0.531169 | 0.487047 |

Таблица 9 – Результаты обучения классификаторов на датасете, построенном на углах скелетной модели, полученной с помощью MediaPipe

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Train Accuracy** | **Validation Accuracy** |
| Метод опорных векторов | 0.810390 | 0.704663 |
| Градиентный бустинг | 0.994805 | 0.683938 |
| Случайный лес | **1.000000** | **0.725389** |
| K-ближайших соседей | 0.774026 | 0.704663 |
| Логистическая регрессия | 0.445455 | 0.388601 |
| Дерево решений | 0.676623 | 0.637306 |
| AdaBoost | 0.445455 | 0.580311 |

Таблица 10 – Результаты обучения классификаторов на датасете, построенном на точках скелетной модели, полученной с помощью YOLO

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Train Accuracy** | **Validation Accuracy** |
| Метод опорных векторов | 0.871528 | 0.791667 |
| Градиентный бустинг | **0.997685** | 0.833333 |
| Случайный лес | **0.997685** | **0.851852** |
| K-ближайших соседей | 0.868056 | 0.777778 |
| Логистическая регрессия | 0.780093 | 0.712963 |
| Дерево решений | **0.997685** | 0.754630 |
| AdaBoost | 0.660880 | 0.592593 |

Таблица 11 – Результаты обучения классификаторов на датасете, построенном на углах скелетной модели, полученной с помощью YOLO

|  |  |  |
| --- | --- | --- |
| **Классификатор** | **Train Accuracy** | **Validation Accuracy** |
| Метод опорных векторов | 0.791667 | 0.712963 |
| Градиентный бустинг | 0.990741 | 0.796296 |
| Случайный лес | **0.997685** | **0.824074** |
| K-ближайших соседей | 0.826389 | 0.726852 |
| Логистическая регрессия | 0.590278 | 0.555556 |
| Дерево решений | **0.997685** | 0.740741 |
| AdaBoost | 0.664352 | 0.634259 |

Таким образом, наилучшие результаты на валидационных данных показал классификатор Случайный лес на датасете, построенном на точках скелетной модели, полученной с помощью YOLO.

Библиографический список

1. Breiman, L. (2001). Random Forests. Machine Learning.
2. Cao, Z., et al. (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. CVPR.
3. Chen, T. (2023). Occlusion-Robust Yoga Pose Estimation. CVPR Workshops.
4. Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. CVPR.
5. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing. Pearson.
6. He, K., et al. (2016). Deep Residual Learning for Image Recognition. CVPR.
7. LeCun, Y., et al. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE.
8. Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. IJCV.
9. Lugaresi, C. et al. (2021). MediaPipe Pose: Real-time Human Pose Estimation. arXiv.
10. Zhang, Y. et al. (2022). Hybrid Sensor Fusion for 3D Yoga Pose Tracking. IEEE Sensors.
11. Manisha Verma, Sudhakar Kumawat, Yuta Nakashima, Shanmuganathan Raman. Yoga-82: A New Dataset for Fine-grained Classification of Human Poses //

Заключение

Приложения

Приложение А. Листинг программы создания датасета на основе ключевых точек с использованием MediaPipe

import mediapipe as mp

import cv2

import time

import numpy as np

import pandas as pd

import os

import matplotlib.pyplot as plt

import time

*# Импорт модулей mediapipe, необходимых для извлечения ключевых точек и их прорисовки*

mpPose **=** mp**.**solutions**.**pose

pose **=** mpPose**.**Pose()

mpDraw **=** mp**.**solutions**.**drawing\_utils *# Модуль для отрисовки поз*

points **=** mpPose**.**PoseLandmark *# Ключевые точки позы*

path\_train **=** "DATASET/TRAIN/" *# Путь к датасету*

path\_test **=** "DATASET/TEST/" *# Путь к датасету*

print('numpy',np.\_\_version\_\_)

print('cv2', cv2.\_\_version\_\_)

print('pandas', pd.\_\_version\_\_)

print('mediapipe', mp.\_\_version\_\_)

Key\_Points\_MediaPipe = ['Nose', 'RightEyeInnerCorner', 'RightEye', 'RightEyeOuterCorner',

'LeftEyeInnerCorner', 'LeftEye', 'LeftEyeOuterCorner',

'RightEar', 'LeftEar', 'MouthRightCorner', 'MouthLeftCorner', 'RightShoulder', 'LeftShoulder',

'RightElbow', 'LeftElbow', 'RightWrist', 'LeftWrist', 'RightLittleFinger', 'LeftLittleFinger',

'RightIndexFinger', 'LeftIndexFinger', 'RightThumb', 'LeftThumb',

'RightPelvicThigh', 'LeftPelvicThigh', 'RightKnee', 'LeftKnee', 'RightAnkle', 'LeftAnkle',

'RightHeel', 'LeftHeel', 'TheIndexFingerOfTheRightFoot', 'TheIndexFingerOfTheLeftFoot'

]

*# Создание пустого датафрейма для последующего сохраниения ключевых точек*

def init\_data(Key\_Points):

data = []

data.append("FileName")

i = 0

for p in points:

x = Key\_Points[i]

data.append(x + "\_x")

data.append(x + "\_y")

data.append(x + "\_z")

data.append(x + "\_vis")

i += 1

data.append("target")

*#data.append("name\_file")*

data = pd.DataFrame(columns = data)

return data

*# Создание датасета каркасных моделей (набора ключевых точек и метки - названия позы)*

def create\_df\_pos(path, name\_file):

data = init\_data(Key\_Points\_MediaPipe)

count = 0

target = []

count\_files = {}

for dr in os.listdir(path): *# Перебор папок с видами поз*

count\_file = 0

for image in os.listdir(path+"/"+dr): *# Перебор файлов в каждой папке*

temp = []

img = cv2.imread(path+"/"+dr + "/" + image)

count\_file+=1

*# Копирование и конвертация изображения в RGB*

imageWidth, imageHeight = img.shape[:2]

imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) *# Преобразование BGR модели OpenCV в RGB модель, с которой работает MediaPipe*

*# Обнаружение позы*

results = pose.process(imgRGB) *# Построение скелетной модели*

if results.pose\_landmarks: *# Если модель построена*

landmarks = results.pose\_landmarks.landmark *# Извлечение ключевых точек (ориентиров)*

temp = temp + [path + dr + "/" + image]

for i,j in zip(points,landmarks):

temp = temp + [j.x, j.y, j.z, j.visibility] *# Добавление ключевых точек*

temp = temp + [dr] *# Добавление метки*

data.loc[count] = temp *# Запись в дата фрейм*

count +=1

else:

print(path+"/"+dr + "/" + image)

count\_files[dr] = count\_file

data.to\_csv(name\_file) *# Запись датафрейма ключевых точек в файл*

return count\_files, data

*# Построение диаграммы распределения скелетных моделей по классам*

def paint\_bar(data, s):

df = data['target'].value\_counts()

label = list(df.index)

counts = list(df.values)

*# Построение диаграммы распределения*

plt.bar(label, counts)

plt.title('Распределение классов поз в ' + s + ' наборе')

*# Построение датафрейма скелетных моделей обучающего набора*

time\_start = time.time()

count\_files, data\_tr = create\_df\_pos(path\_train, 'dataset\_train\_mediapipe\_full.csv')

time\_end = time.time()

print('time: ',time\_end-time\_start)

print(count\_files) # Перечень нераспознанных изображений

data\_tr.head()

*# Вывод характеристик датафрейма скелетных моделей обучающего набора*

print(data\_tr.info())

print(data\_tr['target'].value\_counts())

paint\_bar(data\_tr, 'TRAIN')

*# Отображение гистограммы*

plt.show()

*# Построение датафрейма скелетных моделей тестового набора*

time\_start = time.time()

count\_files, data\_te = create\_df\_pos(path\_test, 'dataset\_test\_mediapipe\_full.csv')

time\_end = time.time()

print('time: ',time\_end-time\_start)

print(count\_files)

data\_te.head()

*# Вывод характеристик датафрейма скелетных моделей тестового набора*

print(data\_te.info())

print(data\_te['target'].value\_counts())

paint\_bar(data\_te, 'TEST')

*# Отображение гистограммы*

plt.show()

Приложение Б. Листинг программы создания датасета на основе ключевых углов с использованием MediaPipe

*# Создание наборов данных*

from sklearn.svm import SVC

from sklearn.linear\_model import RidgeClassifierCV

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

import mediapipe as mp

import cv2

import time

import numpy as np

import pandas as pd

import os

import math

*# Загрузка обучающего набора*

data\_train = pd.read\_csv("dataset\_train\_mediapipe\_short.csv")

*# Загрузка тестового набора*

data\_test = pd.read\_csv("dataset\_test\_mediapipe\_short.csv")

*# Вычисление угла между двумя прямыми*

def angle(x0, y0, x1, y1, x2, y2):

calc\_angle = math.degrees(math.atan2(y2 - y0, x2 - x0) - math.atan2(y1 - y0, x1 - x0))

if calc\_angle < 0:

calc\_angle += 360

return calc\_angle

Angle\_mediaPipe = ['RightElbowAngle', 'LeftElbowAngle', 'RightHandAngle', 'LeftHandAngle', 'RightKneeAngle', 'LeftKneeAngle',

'RightLegAngle', 'LeftLegAngle',

]

Key\_Points\_MediaPipe = ['Nose', 'RightEyeInnerCorner', 'RightEye', 'RightEyeOuterCorner',

'LeftEyeInnerCorner', 'LeftEye', 'LeftEyeOuterCorner',

'RightEar', 'LeftEar', 'MouthRightCorner', 'MouthLeftCorner', 'RightShoulder', 'LeftShoulder',

'RightElbow', 'LeftElbow', 'RightWrist', 'LeftWrist', 'RightLittleFinger', 'LeftLittleFinger',

'RightIndexFinger', 'LeftIndexFinger', 'RightThumb', 'LeftThumb',

'RightPelvicThigh', 'LeftPelvicThigh', 'RightKnee', 'LeftKnee', 'RightAnkle', 'LeftAnkle',

'RightHeel', 'LeftHeel', 'TheIndexFingerOfTheRightFoot', 'TheIndexFingerOfTheLeftFoot'

]

*# Создание пустого датафрейма*

def init\_data(angle\_name):

data = []

data.append("FileName")

for p in angle\_name:

data.append(p)

data.append("target")

data = pd.DataFrame(columns = data)

return data

*# Создание датасета каркасных моделей (набора углов и метки - названия позы)*

def create\_df\_angle\_pos(angle\_name, df, name\_file):

data = init\_data(angle\_name)

target = []

for i in range(len(df)):

temp = []

temp += [df.loc[i]['FileName']]

temp += [angle(df.loc[i]['RightElbow\_x'], df.loc[i]['RightElbow\_y'],

df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'],

df.loc[i]['RightWrist\_x'], df.loc[i]['RightWrist\_y'])]

temp += [angle(df.loc[i]['LeftElbow\_x'], df.loc[i]['LeftElbow\_y'],

df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'],

df.loc[i]['LeftWrist\_x'], df.loc[i]['LeftWrist\_y'])]

temp += [angle(df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'],

df.loc[i]['RightElbow\_x'], df.loc[i]['RightElbow\_y'],

df.loc[i]['RightPelvicThigh\_x'], df.loc[i]['RightPelvicThigh\_y'])]

temp += [angle(df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'],

df.loc[i]['LeftElbow\_x'], df.loc[i]['LeftElbow\_y'],

df.loc[i]['LeftPelvicThigh\_x'], df.loc[i]['LeftPelvicThigh\_y'])]

temp += [angle(df.loc[i]['RightKnee\_x'], df.loc[i]['RightKnee\_y'],

df.loc[i]['RightPelvicThigh\_x'], df.loc[i]['RightPelvicThigh\_y'],

df.loc[i]['RightAnkle\_x'], df.loc[i]['RightAnkle\_y'])]

temp += [angle(df.loc[i]['LeftKnee\_x'], df.loc[i]['LeftKnee\_y'],

df.loc[i]['LeftPelvicThigh\_x'], df.loc[i]['LeftPelvicThigh\_y'],

df.loc[i]['LeftAnkle\_x'], df.loc[i]['LeftAnkle\_y'])]

temp += [angle(df.loc[i]['RightPelvicThigh\_x'], df.loc[i]['RightPelvicThigh\_y'],

df.loc[i]['RightKnee\_x'], df.loc[i]['RightKnee\_y'],

df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'])]

temp += [angle(df.loc[i]['LeftPelvicThigh\_x'], df.loc[i]['LeftPelvicThigh\_y'],

df.loc[i]['LeftKnee\_x'], df.loc[i]['LeftKnee\_y'],

df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'])]

temp += [df.loc[i]['target']]

data.loc[i] = temp *# Запись в датафрейм*

data.to\_csv(name\_file) *# Запись датафрейма углов в файл*

return data

*# Построение датафрейма угловых скелетных моделей обучающего набора*

data\_tr\_angle = create\_df\_angle\_pos(Angle\_mediaPipe, data\_train, 'dataset\_train\_mediapipe\_angle.csv')

print(data\_tr\_angle.info())

data\_tr\_angle.head()

*# Построение датафрейма угловых скелетных моделей тестового набора*

data\_te\_angle = create\_df\_angle\_pos(Angle\_mediaPipe, data\_test, 'dataset\_test\_mediapipe\_angle.csv')

print(data\_te\_angle.info())

data\_te\_angle.head()

Приложение В. Листинг программы создания датасета на основе ключевых точек с использованием YOLO

%matplotlib inline

import mediapipe as mp

import cv2

import time

import numpy as np

import pandas as pd

import os

import matplotlib.pyplot as plt

from ultralytics import YOLO

from PIL import Image

*# Загрузка модели*

model = YOLO('yolo11n-pose.pt') # load an official model

points = 17

path\_train = "DATASET/TRAIN/" # Путь к датасету

path\_test = "DATASET/TEST/" # Путь к датасету

Key\_Points\_YOLO = ['Nose', 'LeftEye', 'RightEye', 'LeftEar', 'RightEar',

'LeftShoulder', 'RightShoulder', 'LeftElbow', 'RightElbow',

'LeftWrist', 'RightWrist', 'LeftHip', 'RightHip',

'LeftKnee', 'RightKnee', 'LeftAnkle', 'RightAnkle'

]

*# Создание пустого датафрейма для последующего сохраниения ключевых точек*

def init\_data(Key\_Points):

data = []

data.append("FileName")

i = 0

for p in range(points):

x = Key\_Points[i]

data.append(x + "\_x")

data.append(x + "\_y")

i += 1

data.append("target")

data = pd.DataFrame(columns = data)

return data

*# Создание датасета каркасных моделей (набора ключевых точек и метки - названия позы)*

def create\_df\_pos(path, name\_file):

data = init\_data(Key\_Points\_YOLO)

count = 0

target = []

for dr in os.listdir(path): *# Перебор папок с видами поз*

for image in os.listdir(path + "/" +dr): *# Перебор файлов в каждой папке*

temp = []

img = cv2.imread(path + "/" + dr + "/" + image)

*# Копирование и конвертация изображения в RGB*

imageWidth, imageHeight = img.shape[:2]

imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) *# Преобразование BGR модели OpenCV в RGB модель, с которой работает YOLO*

*# Обнаружение позы*

results = model(imgRGB) *# Построение скелетной модели*

*# Извлечение результатов*

for result in results[0]:

xy = result.keypoints.xy *# x and y coordinates*

xyn = result.keypoints.xyn *# normalized*

kpts = result.keypoints.data *# x, y, visibility (if available)*

keypoints = np.array(xy[0])

normal\_keypoints = np.array(xyn[0])

temp = temp + [path + dr + "/" + image]

for i in range(len(normal\_keypoints)):

temp = temp + [normal\_keypoints[i][0], normal\_keypoints[i][1]] *# Добавление ключевых точек*

temp = temp + [dr] *# Добавление метки*

data.loc[count] = temp *# Запись в датафрейм*

count +=1

data.to\_csv(name\_file) *# Запись датафрейма ключевых точек в файл*

return data

*# Построение диаграммы распределения скелетных моделей по классам*

def paint\_bar(data, s):

df = data['target'].value\_counts()

label = list(df.index)

counts = list(df.values)

*# Построение диаграммы распределения*

plt.bar(label, counts)

plt.title('Распределение классов поз в ' + s + ' наборе')

*# Построение датафрейма скелетных моделей обучающего набора*

time\_start = time.time()

data\_tr = create\_df\_pos(path\_train, 'dataset\_train\_yolo.csv')

time\_end = time.time()

print('time: ', time\_end-time\_start)

data\_tr.head()

*# Вывод характеристик датафрейма скелетных моделей обучающего набора*

print(data\_tr.info())

print(data\_tr['target'].value\_counts())

paint\_bar(data\_tr, 'TRAIN')

*# Отображение гистограммы*

plt.show()

*# Построение датафрейма скелетных моделей тестового набора*

time\_start = time.time()

data\_te = create\_df\_pos(path\_test, 'dataset\_test\_yolo.csv')

time\_end = time.time()

print('time: ', time\_end-time\_start)

data\_te.head()

*# Вывод характеристик датафрейма скелетных моделей тестового набора*

print(data\_te.info())

#print(data\_te.describe())

print(data\_te['target'].value\_counts())

paint\_bar(data\_te, 'TEST')

*# Отображение гистограммы*

plt.show()

Приложение Г. Листинг программы создания датасета на основе ключевых углов с использованием YOLO

*# Создание наборов данных*

from sklearn.svm import SVC

from sklearn.linear\_model import RidgeClassifierCV

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.metrics import confusion\_matrix, ConfusionMatrixDisplay

from sklearn.metrics import accuracy\_score, classification\_report

import matplotlib.pyplot as plt

import mediapipe as mp

import cv2

import time

import numpy as np

import pandas as pd

import os

import math

*# Загрузка обучающего набора*

data\_train = pd.read\_csv("dataset\_train\_yolo.csv")

*# Загрузка тестового набора*

data\_test = pd.read\_csv("dataset\_test\_yolo.csv")

*# Вычисление угла между двумя прямыми*

def angle(x0, y0, x1, y1, x2, y2):

calc\_angle = math.degrees(math.atan2(y2 - y0, x2 - x0) - math.atan2(y1 - y0, x1 - x0))

if calc\_angle < 0:

calc\_angle += 360

return calc\_angle

Angle\_YOLO = ['RightElbowAngle', 'LeftElbowAngle', 'RightHandAngle', 'LeftHandAngle', 'RightKneeAngle', 'LeftKneeAngle',

'RightLegAngle', 'LeftLegAngle'

]

Key\_Points\_YOLO = ['Nose', 'LeftEye', 'RightEye', 'LeftEar', 'RightEar',

'LeftShoulder', 'RightShoulder', 'LeftElbow', 'RightElbow',

'LeftWrist', 'RightWrist', 'LeftHip', 'RightHip',

'LeftKnee', 'RightKnee', 'LeftAnkle', 'RightAnkle'

]

*# Создание пустого датафрейма*

def init\_data(angle\_name):

data = []

data.append("FileName")

for p in angle\_name:

data.append(p)

data.append("target")

data = pd.DataFrame(columns = data)

return data

*# Создание датасета каркасных моделей (набора углов и метки - названия позы)*

def create\_df\_angle\_pos(angle\_name, df, name\_file):

data = init\_data(angle\_name)

target = []

for i in range(len(df)):

temp = []

temp += [df.loc[i]['FileName']]

temp += [angle(df.loc[i]['RightElbow\_x'], df.loc[i]['RightElbow\_y'],

df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'],

df.loc[i]['RightWrist\_x'], df.loc[i]['RightWrist\_y'])]

temp += [angle(df.loc[i]['LeftElbow\_x'], df.loc[i]['LeftElbow\_y'],

df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'],

df.loc[i]['LeftWrist\_x'], df.loc[i]['LeftWrist\_y'])]

temp += [angle(df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'],

df.loc[i]['RightElbow\_x'], df.loc[i]['RightElbow\_y'],

df.loc[i]['RightHip\_x'], df.loc[i]['RightHip\_y'])]

temp += [angle(df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'],

df.loc[i]['LeftElbow\_x'], df.loc[i]['LeftElbow\_y'],

df.loc[i]['LeftHip\_x'], df.loc[i]['LeftHip\_y'])]

temp += [angle(df.loc[i]['RightKnee\_x'], df.loc[i]['RightKnee\_y'],

df.loc[i]['RightHip\_x'], df.loc[i]['RightHip\_y'],

df.loc[i]['RightAnkle\_x'], df.loc[i]['RightAnkle\_y'])]

temp += [angle(df.loc[i]['LeftKnee\_x'], df.loc[i]['LeftKnee\_y'],

df.loc[i]['LeftHip\_x'], df.loc[i]['LeftHip\_y'],

df.loc[i]['LeftAnkle\_x'], df.loc[i]['LeftAnkle\_y'])]

temp += [angle(df.loc[i]['RightHip\_x'], df.loc[i]['RightHip\_y'],

df.loc[i]['RightKnee\_x'], df.loc[i]['RightKnee\_y'],

df.loc[i]['RightShoulder\_x'], df.loc[i]['RightShoulder\_y'])]

temp += [angle(df.loc[i]['LeftHip\_x'], df.loc[i]['LeftHip\_y'],

df.loc[i]['LeftKnee\_x'], df.loc[i]['LeftKnee\_y'],

df.loc[i]['LeftShoulder\_x'], df.loc[i]['LeftShoulder\_y'])]

temp += [df.loc[i]['target']]

data.loc[i] = temp *# Запись в датафрейм*

data.to\_csv(name\_file) *# Запись датафрейма углов в файл*

return data

*# Построение датафрейма угловых скелетных моделей обучающего набора*

time\_start = time.time()

data\_tr\_angle = create\_df\_angle\_pos(Angle\_YOLO, data\_train, 'dataset\_train\_yolo\_angle.csv')

time\_end = time.time()

print('time: ', time\_end-time\_start)

print(data\_tr\_angle.info())

data\_tr\_angle.head()

*# Построение датафрейма угловых скелетных моделей тестового набора*

time\_start = time.time()

data\_te\_angle = create\_df\_angle\_pos(Angle\_YOLO, data\_test, 'dataset\_test\_yolo\_angle.csv')

time\_end = time.time()

print('time: ', time\_end-time\_start)

print(data\_te\_angle.info())

data\_te\_angle.head()