Министерство науки и высшего образования Российской Федерации
Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа теоретической механики

Работа допущена к защите

Директор высшей школы

А. М. Кривцов

«___»_____20__ г.

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

# МОНИТОРИНГ ПРАВИЛЬНОСТИ ПОСАДКИ ЧЕЛОВЕКА НА РАБОЧЕМ МЕСТЕ МЕТОДАМИ МАШИННОГО ОБУЧЕНИЯ

По направлению подготовки

01.03.03 «Механика и математическое моделирование»

профиль

01.03.03_01 Механика и математическое моделирование сред с микроструктурой

Выполнил

студент гр. 3630103/70101 Д. В. Аливохин

Руководитель

Доцент, к.ф.-м.н. Е. А. Подольская

Консультант
ассистент Д. В. Цветков

Санкт-Петербург

2021

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**Институт прикладной математики и механики**

**Высшая школа теоретической механики**

УТВЕРЖДАЮ

Директор

Высшей школы теоретической механики

_____А. М. Кривцов

«__» _____ 20__ г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы**

студенту Аливохину Даниилу Вадимовичу, гр. 3630103/70101

1. Тема работы: Мониторинг правильности посадки человека на рабочем месте методами машинного обучения

2. Срок сдачи студентом законченной работы: 14.06.2021

3. Исходные данные по работе: Справочная литература, актуальный публикации по теме исследования

4. Содержание работы (перечень подлежащих разработке вопросов):

Формулировка задачи и определение критериев правильной посадки. Выбор модели для определения ключевых точек суставов тела человека. Исследование и сравнение алгоритмов машинного обучения для задачи классификации. Программная реализация на основе алгоритма с лучшей точностью для мониторинга видеопотока.

5. Перечень графического материала (с указанием обязательных чертежей): не предусмотрены

6. Консультанты по работе: ассистент Высшей школы теоретической механики Д. В. Цветков

7. Дата выдачи задания: 01.02.2021

Руководитель ВКР _____ Е. А. Подольская

Задание принял к исполнению: 01.02.2021

Студент _____ Д. В. Аливохин

## РЕФЕРАТ

На 38 с., 23 рисунка, 3 таблицы

КОМПЬЮТЕРНОЕ ЗРЕНИЕ, МАШИННОЕ ОБУЧЕНИЕ, ОЦЕНКА ПОЗЫ, ЗАДАЧА КЛАССИФИКАЦИИ

В рамках данной работы был рассмотрен способ мониторинга правильности посадки человека за рабочим местом методами машинного обучения. Работа состоит из 3 частей: изучение методов оценки позы, определяющих ключевые точки суставов тела человека, обзор, применение и анализ точности алгоритмов классификации и программная реализация мониторинга видео для оценки правильности посадки.

## THE ABSTRACT

38 pages, 23 pictures, 3 tables

COMPUTER VISION, MACHINE LEARNING, POSE ESTIMATION, CLASSIFICATION

In this paper were considered a way to monitor the correct posture of a person at the workplace using machine learning methods. This work consists of 3 parts: research of pose estimation methods, defining person’s body joint key points, review, application and accuracy analysis of classification algorithms and software implementation to assess the correct posture.

# СОДЕРЖАНИЕ

ВВЕДЕНИЕ ..... 5
Глава 1. Общие сведения ..... 8
1.1. Компьютерное зрение ..... 8
1.2. Задача оценки положения ..... 9
1.3. Машинное обучение ..... 10
1.4. Глубокое обучение ..... 11
1.5. Требования к посадке за рабочим местом ..... 14
Глава 2. Методы обнаружения ключевых точек ..... 15
2.1. OpenPose ..... 15
2.2. PoseNet ..... 18
2.3. Выбор модели ..... 19
Глава 3. Анализ алгоритмов классификации ..... 22
3.1. Логистическая регрессия ..... 22
3.2. Метод опорных векторов ..... 22
3.3. Случайный лес ..... 24
3.4. Градиентный бустинг ..... 25
3.5. Метрики ..... 26
3.6. Сравнение точности алгоритмов ..... 28
Глава 4. Анализ решения ..... 34
4.1. Оценка выбранного алгоритма на тестовых данных ..... 34
4.2. Реализация мониторинга видеопотока ..... 36
4.3. Анализ мониторинга видеопотока ..... 38
Заключение ..... 40
Список использованных источников ..... 42

# ВВЕДЕНИЕ

За последние десятилетия образ жизни людей достаточно сильно поменялся в контексте повседневных практик. В связи с развитием информационных технологий человеческая деятельность все больше компьютеризируется. Появляются новые профессии, предполагающие работу за компьютером, множество уже существовавших видов деятельности также автоматизируются. Соответственно, подобные тенденции приводят к тому, что в среднем человек все больше и больше времени проводит именно в сидячем положении. Особенно заметно это стало в последний год, когда из-за пандемии многие люди перешли на удаленную форму работы и стали ещё больше времени проводить в сидячем положении.

Однако при этом большинство людей уделяют достаточно мало внимания правильному положению за рабочим местом. На данную тему проводятся многочисленные исследования, отражающие вред для здоровья человека при неверной посадке. Так, это может влиять на позвоночник и осанку человека, на его зрение, на кровообращение, что приводит в итоге к различного рода заболеваниям, болям в спине и шее. В связи с увеличением числа проблем со здоровьем специалисты предлагают множество решений, включающих в себя уменьшение работы за столом, а также работу в положении стоя, ортопедические стулья и прочие товары, уменьшающие риск развития заболеваний. Тем не менее, зачастую изменить работу в привычном формате за столом становится трудной задачей и невозможно во всех случаях, а предлагаемые методы не исключают проблему полностью. Еще одно распространенное предлагаемое решение – просвещение людей о правильной посадке и регулярные напоминания о важности наблюдения за этим, однако изменить повседневную привычку остаётся тяжёлым процессом.

Соответственно, представляется возможность скорректировать эту практику другими методами. В данной работе предлагается рассмотреть, как методы и алгоритмы компьютерного зрения и машинного обучения могут помочь с описанной проблемой. При помощи технологий компьютерного зрения

возможно отслеживание положения человека за столом. Используя же алгоритмы машинного обучения, можно анализировать полученные данные, с их помощью контролировать посадку и сигнализировать, если человек сидит неправильно.

Использование данного метода представляется возможным в связи с развитием алгоритмов компьютерного зрения в последние годы, особенно используя алгоритмы на основе нейронных сетей. Данная область предоставляет возможность анализировать фото и видео, для чего используются технологии глубокого обучения. Так, появляется всё больше точных алгоритмов, позволяющих отслеживать и классифицировать объекты на основе изображений и видеоматериалов. Кроме того, данные алгоритмы позволяют применять биомеханическое моделирование. В этом случае можно с достаточно хорошей точностью получать компьютерное представление основных точек тела человека. Системы, обученные распознавать корректность положения, могут предоставлять данные человеку о необходимости изменить посадку. Такое решение даст возможность регулярного наблюдения без привлечения непосредственно специалистов. Конечно, существуют такие методы, как использование корректоров осанки, либо специальные ортопедические стулья. Преимуществом же использования методов компьютерного зрения является отсутствие необходимости каких-либо финансовых вложений, так как веб-камера есть практически у каждого. Привлечение же в дальнейшем для разработки профессиональных остеопатов позволит оценить точность алгоритма и выработать максимально верные критерии для оценивания.

Целью работы является исследование и оценка методов машинного обучения для мониторинга правильности посадки человека за рабочим местом и разработка алгоритма на основе наилучшей модели.

Таким образом, можно выделить следующие основные задачи, которые решаются в данной работе:

1. Сбор данных для анализа и аугментация изображений для увеличения разнообразия датасета.

2. Выбор модели для определения ключевых точек человеческого тела.
3. Исследование и сравнение алгоритмов машинного обучения для задачи классификации.
4. Реализация выбранного алгоритма.
5. Использование полученного алгоритма для мониторинга видеопотока.
6. Анализ получившегося решения и его точности.

# ГЛАВА 1. ОБЩИЕ СВЕДЕНИЯ

## *1.1. Компьютерное зрение*

Компьютерное зрение – область информатики, которая фокусируется на частичном воспроизведении сложной системы человеческого зрения для того, чтобы идентифицировать и обрабатывать объекты на изображениях и видео также, как это делает человек. Она включает в себя технологии и методы, которые позволяют компьютеру получать данные с изображений и видеозаписей и извлекать информацию из полученных данных.

В последние десятилетия произошел заметный рост вычислительных мощностей компьютеров, что в свою очередь привело к значительному скачку в области компьютерного зрения [1]. Кроме того, этому также способствовал прогресс в области нейронных сетей. До 2012 года в области обработки изображений не использовались нейросетевые подходы, так как они считались неспособными превзойти классические алгоритмы. Революция произошла в 2012 году, когда на соревнованиях по классификации изображений ILSVRC (ImageNet Large Scale Visual Recognition Competition) первое место занял алгоритм AlexNet [11], который в 5 предсказаниях получил ошибку более, чем в 1.5 раза меньше, чем у ближайшего конкурента. С этого момента начался бурный рост использования нейронных сетей в задачах компьютерного зрения. В настоящие же время они практически вытеснили другие алгоритмы.

Современные методы компьютерного зрения охватывают достаточно широкий круг задач. Существует достаточно большое количество вариантов применения данных алгоритмов, основными можно считать:

- Классификация изображений – фундаментальная задача, которая состоит в том, чтобы понять все изображение как единое целое, и на основе данного понимания присвоить ему определенную метку.
- Обнаружение объектов – находит конкретный объект на изображении. Расширенное обнаружение объектов позволяет выявить несколько объектов на одной картинке.

- Сегментация изображения – позволяет определить, какие именно пиксели принадлежат искомому объекту на изображении.
- Оценка положения – задача компьютерного зрения, которая позволяет определить позу человека или некого объекта путем обнаружения ключевых точек. Кроме того, данная задача может решать проблему определения положения и ориентации камеры относительно человека или объекта.
- Распознавание текстов – дает возможность определять текст на различных изображениях.
- Генерация объектов – заключается в том, чтобы по известному набору изображений создать новое, не совпадающее при этом ни с одним из тестовых.

Для данной работы наибольший интерес представляет задача оценки положения. Так как она позволяет определять ключевые суставные точки человеческого тела, которые могут достаточно хорошо описывать положение человека в пространстве. Благодаря чему, можно будет делать выводы относительно посадки за рабочим местом.

### *1.2. Задача оценки положения*

Оценка положения, а конкретно оценка позы человека – достаточно сложная задача компьютерного зрения, так как зависит от масштаба и расширения изображения, освещенности, различных вариантов одежды и окружения [19**Ошибка! Источник ссылки не найден.**]. Основная идея заключается в том, чтобы обнаруживать суставы человека на изображении или видеозаписи.

В последние годы данная задача получила огромное развитие. Кроме методов оценки 2D позы появились методы для 3D оценки. Также теперь алгоритмам необязательно, чтобы на изображении был только один человек, они достаточно хорошо справляются и с несколькими людьми. Например, в [7] алгоритмы оценки позы человека используются для создания приложения, которое позволяет определять на видеозаписи воров в магазинах. В [14] авторы

используют одну RGB-камеру для решения задачи 3D оценки позы. Предлагаемое в данной работе решение позволяет на основе обычного 2D изображения с камеры получить трёхмерную модель каждого человека в кадре.

Построение модели для оценки позы человека основано на использовании глубокого обучения, которое является совокупностью методов машинного обучения, основанных на имитации работы человеческого мозга.

### *1.3. Машинное обучение*

Машинное обучение – подраздел искусственного интеллекта, изучающий методы построения алгоритмов, которые не решают задачу напрямую, а обучаются, используя решения множества похожих задач. Основные задачи подразделяются на 2 типа [2]:

- Обучение с учителем – основано на выявлении закономерности в данных, когда для множества объектов (признаков) имеется множество возможных ответов (откликов).
- Обучение без учителя – область задач, связанных с обработкой данных, в которых требуется отыскать некоторые взаимосвязи, когда известно только описание множества объектов.

В случае обучения с учителем задается множество объектов $X$, множество допустимых ответов $Y$, и существует целевая функция $f: X \to Y$, ее значения $y_i = f(x_i)$ известны только на ограниченном множестве $x_1, \dots, x_l \subset X$. И $X^l = (x_i, y_i)^l{}_{i=1}$.

Обучение с учителем также разделяется на несколько типов:

- Классификация – множество возможных ответов конечно $Y = \{1, \dots, N\}$.
- Регрессия – возможные ответы являются множеством действительных чисел $Y = \mathbb{R}$.

Так как алгоритмы оценки положения будут использоваться только для того, чтобы получить ключевые точки, необходимо также обеспечить классификацию набора точек. В данном случае достаточно хорошим решением могут быть классические алгоритмы машинного обучения.

Учитывая, что у исходного набора данных будет конечное число ответов, в данной работе будет использоваться задача классификации. Каждому объекту в наборе данных будет присвоен один из двух классов (“0” – правильная посадка; “1” – неправильная посадка). Таким образом, алгоритм должен будет обнаружить взаимосвязь между точками, которая позволит разделить все множество признакового пространства на несколько классов и присваивать объектам верные метки.

### *1.4. Глубокое обучение*

Основным отличием глубокого обучения от классических алгоритмов машинного обучения является наличие в первых более одного скрытого слоя. Основой всего глубокого обучения является искусственная нейронная сеть – модель, имитирующая работу нейронов в человеческом мозге, которые являются ее частью, а в качестве связей между ними выступают веса [9]. Каждая нейронная сеть в общем случае состоит из трёх типов слоев:

- Входной слой – отвечает за прием входных данных.
- Скрытые слои – выполняют различные математические операции и преобразования над данными.
- Выходной слой – выдает получившейся результат.

Базовое представление данного типа алгоритмов – однослойная нейронная сеть, примером которой является перцептрон, который является бинарным классификатором [2]. Его принцип работы:

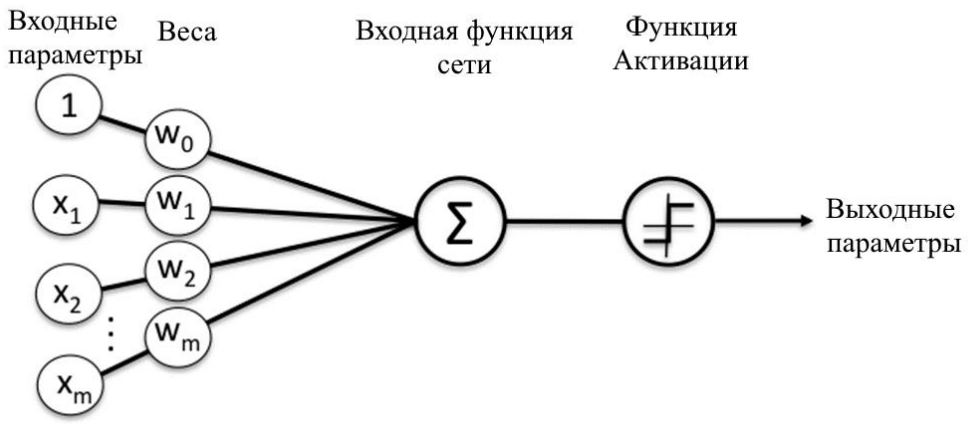



Рисунок 1. Архитектура перцептрона

- На вход подается набор данных $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$, где $x_i \in \mathbb{R}^n, y_i \in \{-1,1\}$.
- Инициализируются начальные веса $w_0 = 0$.
- Для каждого обучающего примера происходит обновление весов:
  - $y' = sign({w_t}^T x_i)$
  - Если $y_i \neq y'$:
    - $w_{t+1} = w_t + r(y_i x_i)$
- Использование функции активации
- Возвращается вектор с результатами

Как видно по рисунку 1 в данной модели только один скрытый слой. Для реализации алгоритмов глубокого обучения, как было сказано ранее, требуется модель сложнее, состоящая из двух и более скрытых слоев. Кроме того, необходимо наличие алгоритма обратного распространения ошибки, который позволяет минимизировать функцию потерь путем корректировки весов и смещений нейронной сети [10].

Для реализации задач классификации в области компьютерного зрения используется сверточная нейронная сеть.

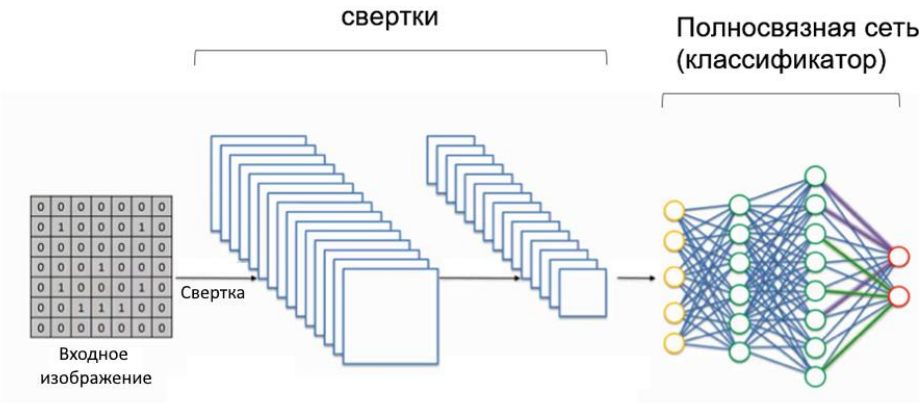



Рисунок 2. Архитектура сверточной нейронной сети.

На вход данной сети подается изображение, которое представляется массивом пикселей. В случае цветного RGB изображения – это трехмерный массив, где первые две размерности отвечают за длину и ширину изображения, а третья размерность – значение каналов RGB. Каждому элементу соответствует значение от 0 до 255, которое показывает интенсивность пикселя. Первым скрытым слоем будет сверточный слой. Свертка – линейное преобразование (перемножение матрицы изображения и ядра путем прохождения через длину и ширину), которое отвечает за извлечение признака из изображения.

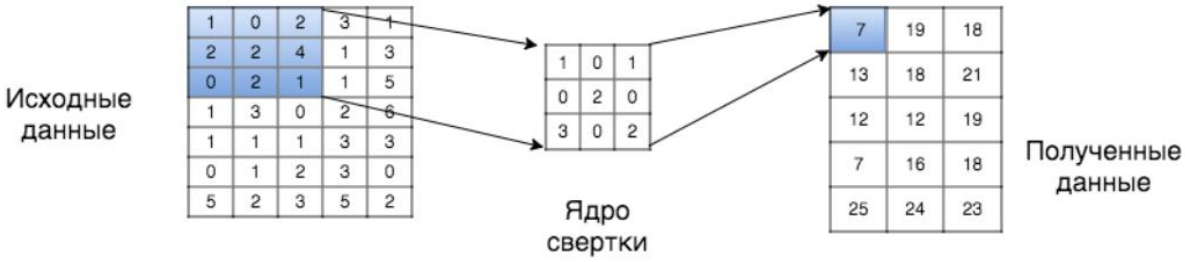



Рисунок 3. Операция свертки

Кроме того, в сверточном слое также встречается операция пулинга – уменьшение размерности изображения. Комбинации данных операций и составляют из себя сверточный слой.

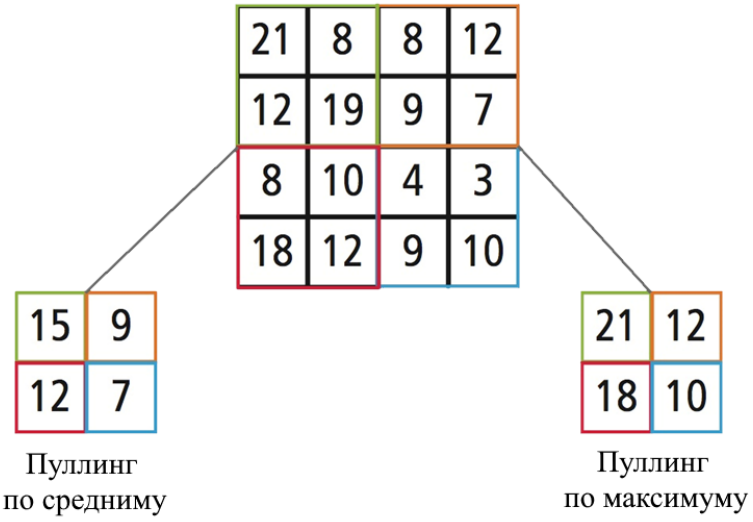



Рисунок 4. Операция пулинга

Следующей частью данного типа нейронных сетей является полносвязный слой, простейшим примером которого можно назвать упоминавшийся ранее перцептрон. На выходе получается вектор с результатами.

### *1.5. Требования к посадке за рабочим местом*

Так как набор данных собирался самостоятельно, то необходимо выделить требования и признаки, которые позволяли бы определять правильную посадку на фотографиях для дальнейшей верной разметки. Таким образом, были выделены основные критерии [3]:

- Спина: ровная, позвоночник расположен вертикально. Если нужно немного наклониться, то делают это всем корпусом, сохраняя позвоночник ровным.
- Плечи: расположены на прямой горизонтальной линии.
- Расстояние от груди до парты: равно раскрытой ладони.
- Руки: лежат на столе, согнутые в локтях.
- Ноги: расположены рядом, согнуты в коленях, стопы полностью соприкасаются с полом, что позволяет равномерно распределить нагрузку на ягодицы.

Соответственно, учитывая данные признаки, выбиралась необходимая метка класса при сборе набора данных.

# ГЛАВА 2. МЕТОДЫ ОБНАРУЖЕНИЯ КЛЮЧЕВЫХ ТОЧЕК

## *2.1. OpenPose*

OpenPose – это библиотека с открытым кодом для обнаружения ключевых точек, которая произвела революцию в задачах обнаружения позы. Она была реализована в Carnegie Mellon University лабораторией перцептивных вычислений [6]. В данной модели используется архитектура, основанная на нескольких сверточных нейронных сетях [22], которая позволяет обнаруживать ключевые точки человеческого тела. Поэтому эта библиотека широко используется в различных приложениях и работах, связанных с оценкой позы. Например, в работе [12] этот фреймворк используется для определения позы во время йоги. В данной статье [21] описывается решение задачи определения падения человека, где OpenPose используется для нахождения ключевых точек и в последствии фиксации момента падения. [18] предлагает использование этой библиотеки для обнаружения вооруженных людей в случае захвата заложников.

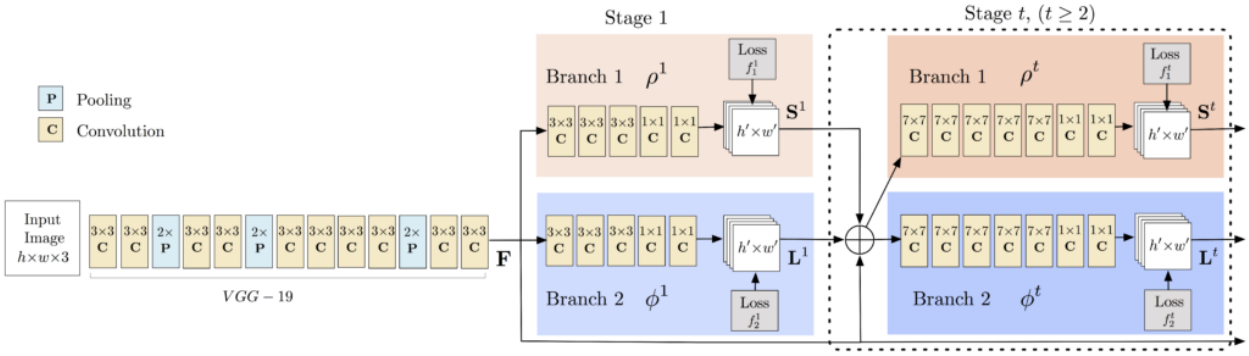



Рисунок 5. Архитектура нейронной сети, используемой в OpenPose

Используемая в библиотеке OpenPose модель принимает на вход цветное изображение, которое позволяет определить матрицу размером $h \times w \times 3$, где h – высота изображения в пикселях, w – ширина изображения и 3 – количество RGB каналов, с которой происходят дальнейшие вычисления.

Обнаружение точек состоит из трех основных этапов:

- Первые 10 слоев – часть нейронной сети VGG-19 [20], которая используется для извлечения признаков из поданного на вход изображения.

- После полученные признаки подаются на вход сверточной нейронной сети, которая содержит в себе две ветви, работающие параллельно.
    - Первая ветвь отвечает за предсказание карт достоверности (confidence maps) того, где находится определенная часть тела (локоть, колено, бедро и т.д.) (Рисунок 6).

Рисунок 6. Карта достоверности для левого плеча

    - Вторая ветвь предсказывает набор 2-мерных векторных полей, которые показывают связь ключевых точек (Рисунок 7).

Рисунок 7. Связь между шеей и левым плечом

- Дополнительные этапы нужны для того, чтобы улучшать полученные предсказания на основе прогнозов предыдущих стадий.
- На выходе получается 4-мерная матрица, которая содержит:
  - ID изображения (если на вход было подано более одного изображения).
  - Индексы карт достоверности ключевых точек и карт связей частей тела.
  - Высоты полученной на выходе карты.
  - Ширину полученной на выходе карты.

Создатели библиотеки предлагают две модели – первая обучена на Multi-Person Dataset (MPII), впервые представленном в данной работе [5], вторая на наборе данных COCO [13]. Первая предлагает 15 точек на выходе, вторая – 18 точек.

Рисунок 8. Точки, получаемые для двух моделей

Индексы точек для COCO: нос – 0, шея – 1, правое плечо – 2, правый локоть – 3, правое запястье – 4, левое плечо – 5, левый локоть – 6, левое запястье – 7, правое бедро – 8, правое колено – 9, правая лодыжка – 10, левое бедро – 11, левое колено – 12, левая лодыжка – 13, правый глаз – 14, левый глаз – 15, правое ухо – 16, левое ухо – 17, задний фон – 18.

Индексы точек для MPII: голова– 0, шея – 1, правое плечо – 2, правый локоть – 3, правый кулак – 4, левое плечо – 5, левый локоть – 6, левое запястье – 7, правое бедро – 8, правое колено – 9, правая лодыжка – 10, левое бедро – 11, левое колено – 12, левая лодыжка – 13, грудь – 14, задний фон – 15.

### *2.2. PoseNet*

Еще одна модель для определения ключевых точек – PoseNet. Данная модель была разработана в Google [17]. Она также основана на архитектуре сверточной нейронной сети и предоставляет те же возможности, что и OpenPose. В качестве примера использования данной модели можно привести работу по реализации системы по обнаружение пешехода в реальном времени для умного города [4].

На вход модели подается цветное RGB изображение и также, как для OpenPose в дальнейшем она работает с трехмерной матрицей. Реализация PoseNet основана на двухэтапной архитектуре, которая состоит из сверточной нейронной сети и алгоритма декодирования. СНС генерирует тепловые карты, которые предсказывают положение ключевых точек, а также создает векторы, которые позволяют в дальнейшем связать точки между собой в случае, если на изображении присутствует более одного человека [16]. Можно сказать, что, в общем, принцип работы OpenPose и PoseNet похож.

Рисунок 9 Точки, определяемые PoseNet

Так как модель PoseNet обучена на наборе данных COCO, то нумерация ключевых точек совпадает с ранее указанной.

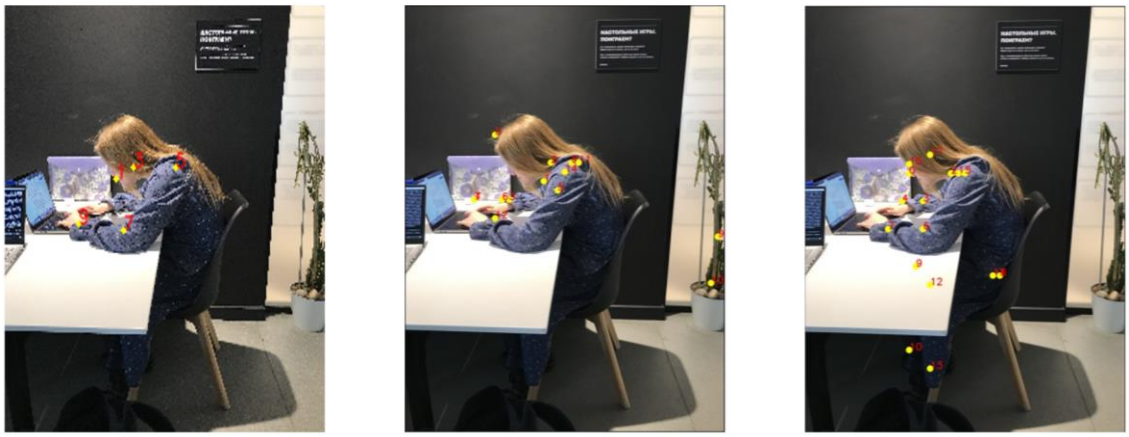
*2.3. Выбор модели*

На основе исследования работы представленных алгоритмов при определении ключевых точек на фотографии человека, сидящего в профиль, было принято решение об использовании модели OpenPose, обученной на датасете COCO. Данный алгоритм показал лучший результат при определении ключевых точек у сидящего человека. В то время как алгоритмы PoseNet и OpenPose (обученный на MPII) показали неудовлетворительные результаты,

особенно при условиях плохой освещенности, либо при отсутствие человеческого лица на фотографии, либо если шея человека закрыта волосами.

Рисунок 10 Пример сложного изображения

На рисунке 10 представлен пример сложного для алгоритмов изображения, лицо и шея закрыты волосами. Покажем, как работает каждый алгоритм на этом изображении.



PoseNet OpenPose (MPII) OpenPose (COCO)

Рисунок 11 Примеры использования алгоритмов на сложном изображении

На рисунке 11 видно, что алгоритм OpenPose на наборе данных MPII показывает плохой результат. Точки проставлены в случайном порядке. Модель PoseNet работает несколько лучше, точки определены верно, но не все, самое критичное – нехватка точек бедер. Что также позволяет отбросить идею использования данного алгоритма. OpenPose, обученная на наборе данных COCO показывает наилучший результат из трех, большинство точек найдены и проставлены верно.

# ГЛАВА 3. АНАЛИЗ АЛГОРИТМОВ КЛАССИФИКАЦИИ

## *3.1. Логистическая регрессия*

Логистическая регрессия является обобщением линейной регрессии для задачи классификации [2]. Будем рассматривать случай, когда $Y = \{-1,1\}$ – имеется два класса. И есть обучающая выборка $X^l$. В этом случае нам необходимо построить линейный алгоритм классификации $a: X \to Y$, где

$$a(x,w) = sign\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right) = sign\langle x,w\rangle,$$

Где $w = (w_1, \dots, w_n)$ – вектор весов, $w_0$ – порог принятия решения, $w_j$ – вес j-го признака. Также необходимо ввести нулевой признак $f_0(x) = -1$

Основная задача при обучении логистической регрессии заключается в том, чтобы получить вектор весов $w$ по обучающей выборке $X^l$. Для этого необходимо минимизировать эмпирический риск при помощи функции потерь:

$$Q(w) = \sum_{i=1}^{m} ln\left(1 + e^{(-y_i\langle x,w\rangle)}\right) \to min_w$$

После нахождения весов w, можно кроме самой классификации $a(x) = sign\langle x,w\rangle$ также находить и оценивать апостериорные вероятности принадлежности к классам:

$$P\{y|x\} = \sigma(y\langle x,w\rangle), y \in Y,$$

Где $\sigma(z) = \frac{1}{1+e^{-z}}$.

## *3.2. Метод опорных векторов*

Метод опорных векторов можно считать улучшением модели перцептрона. Основная идея данного алгоритма заключается в нахождении оптимальной разделяющей гиперплоскости с максимальным зазором [2]. Зазор – расстояние между разделяющей гиперплоскостью и ближайшими объектами обучающей выборки, которые принадлежат разным классам (Рисунок 12). Данные объекты называются опорным векторами.

Рассмотри задачу бинарной классификации. Необходимо построить линейный пороговый классификатор:

$$a(x) = sign\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = sign(\langle w, x\rangle - w_0),$$

уравнение $\langle w, x\rangle = w_0$ описывает гиперплоскость.

Требуется найти оптимальную гиперплоскость, учитывая, что выборка линейно разделима и функционал ошибок:

$$Q(w, w_0) = \sum_{i=1}^{l} [y_i(\langle w, x_i\rangle - w_0) \leq 0].$$

Видим, что $a(x)$ не меняется, если одновременно умножить $w$ и $w_0$ на одну положительную константу. Для выбора константы будем руководствоваться условием, что:

$$\min_{i=1,\dots,l} y_i(\langle w, x_i\rangle - w_0) = 1.$$

Для описания полосы, которая разделяет классы, используется множество точек $\{x: -1 \leq \langle w, x\rangle - w_0 \leq 1\}$.

Необходима максимальная ширина полосы:

$$\langle (x_+ - x_-), \frac{w}{\|w\|}\rangle = \frac{\langle w, x_+\rangle - \langle w, x_-\rangle}{\|w\|} = \frac{(w_0 + 1) - (w_0 - 1)}{\|w\|} = \frac{2}{\|w\|}.$$

Получаем следующие условия (задачу квадратичного программирования), по которым необходимо найти $w$ и $w_0$:

$$\begin{cases} \langle w, w\rangle \to min; \\ y_i(\langle w, x_i\rangle - w_0) \geq 1, i = 1, \dots, l. \end{cases}$$

Данные условия справедливы в случае линейной разделимости классов. Но такие случаи редки. Поэтому требуется обобщить условия для случая, когда выборка линейно неразделима.

Условия для линейно неразделимой выборки:

$$\begin{cases} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l} \xi_i \to min\, 1_{w, w_0, \xi} \\ y_i(\langle w, x_i\rangle - w_0) \geq 1 - \xi_i, i = 1, \dots, l \\ \xi_i \geq 0, i = 1, \dots, l, \end{cases}$$

где $\xi_i$ – характеризуют величину ошибки на $x_i$, константа $C$ – управляющий параметр метода.

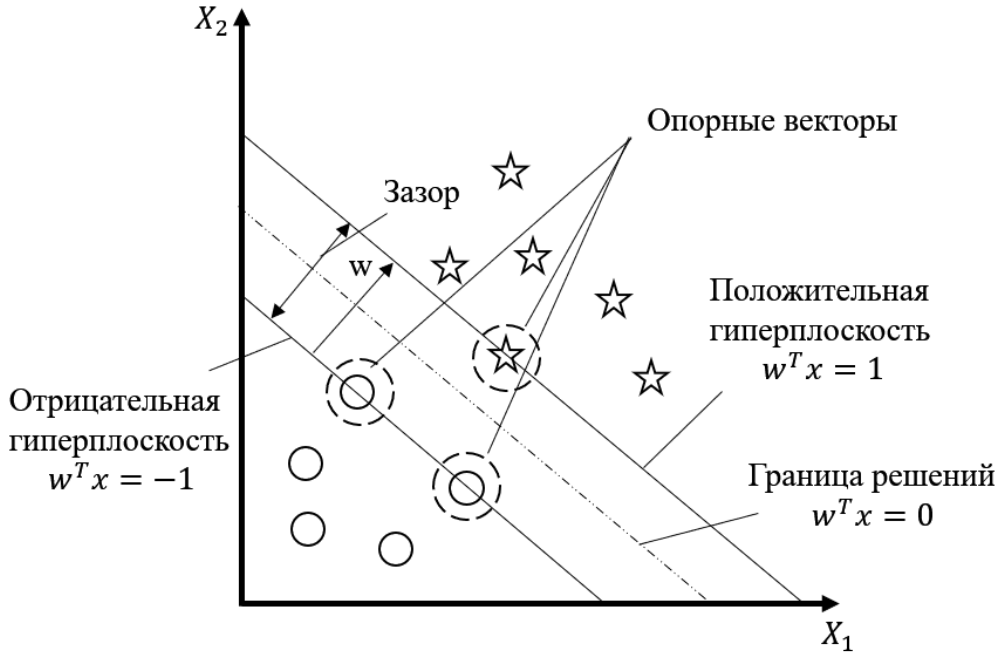



Рисунок 12. Метод опорных векторов

### *3.3. Случайный лес*

Случайный лес – алгоритм машинного обучение, который представляет из себя ансамбль решающих деревьев. Основная идея метода состоит в том, чтобы взять ответы множества решающих деревьев и, для задачи классификации, получить ответ с помощью голосования по большинству [2].

Решающее дерево – k-ичное дерево с решающими правилами в не листовых вершинах (узлах) и некоторым заключением о целевой функции в листовых вершинах (прогнозом). Обычно в машинном обучении используются двоичные деревья. Пример можно увидеть на рисунке 13.

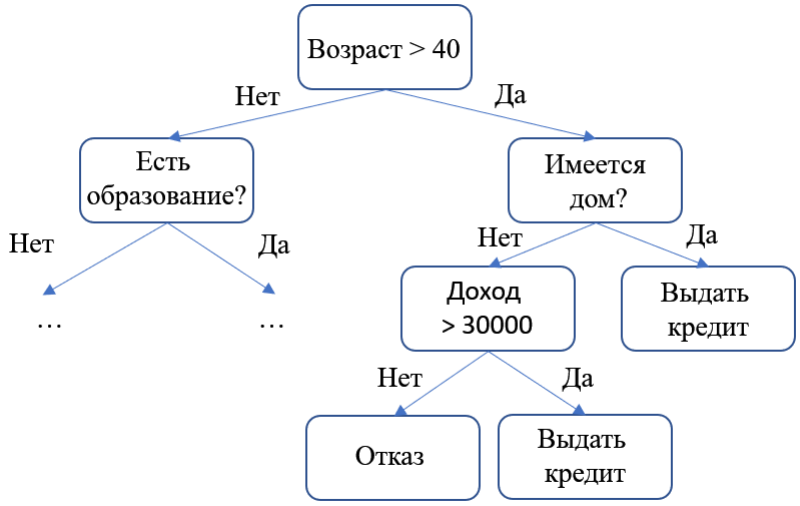



Рисунок 13. Пример решающего дерева

Чтобы построить случайный лес из N решающих деревьев, необходимо:

1. Построить с помощью бутстрапа N случайных подвыборок $\tilde{X}_n$, $n = 1, \dots, N$.
2. Каждая получившаяся подвыборка $\tilde{X}_n$ используется как обучающая выборка для построения соответствующего решающего дерева $b_n(x)$.
3. Построенные деревья объединяются в композицию:

$$a(x) = sign \frac{1}{N} \sum_{n=1}^{N} b_n(x).$$

### *3.4. Градиентный бустинг*

Градиентный бустинг также, как и случайный лес, представляет собой композицию алгоритмов. Обычно в качестве базового алгоритма выбираются деревья решений, но возможно использование и других алгоритмов.

Главное отличие бустинга от случайного леса в том, что первый обучает каждый следующий алгоритм на основе ошибки предыдущего алгоритма.

Чтобы построить градиентный бустинг требуется:

1. Инициализировать композицию $a_0(x) = b_0(x)$ – построить простой алгоритм $b_0(x)$.
2. Шаг итерации:
   - Вычислить вектор сдвига

$$s = -\nabla F = \begin{pmatrix} -L'_z\,(y_1\,, a_{n-1}(x_1)) \\ \dots \\ -L'_z\,(y_l\,, a_{n-1}(x_l)) \end{pmatrix}.$$

- Построить алгоритм

$$b_n(x) = argmin_b \frac{1}{l}\sum_{i=1}^{l}(b(x_i) - s_i)^2\,,$$

необходимо подобрать параметры так, чтобы значения алгоритма на обучающей выборке были как можно ближе к вычисленному вектору сдвига.

3. Остановить итерационный процесс при выполнении критерия останова.

В данной работе использовались реализация градиентного бустинга – CatBoost [8].

### *3.5. Метрики*

Основными метриками в задачах классификации являются accuracy, precision и recall. Accuracy – доля правильных ответов алгоритма, иногда она может не отображать реальную ситуацию, так как часто попадаются задачи с несбалансированными классами. В данном случае она не различает разные виды ошибок. Precision и recall являются метриками, которые оценивают качество работы алгоритмов по отдельности на каждом из классов.

Точность (precision) отвечает за то, какая часть объектов, которым алгоритм классификации присвоил определенную метку, реально принадлежит данном классу.

Полнота (recall) дает понять, какую часть объектов указанного класса из всех нашел алгоритм.

Далее на рисунке 13 представлена графическая интерпретация данных метрик.

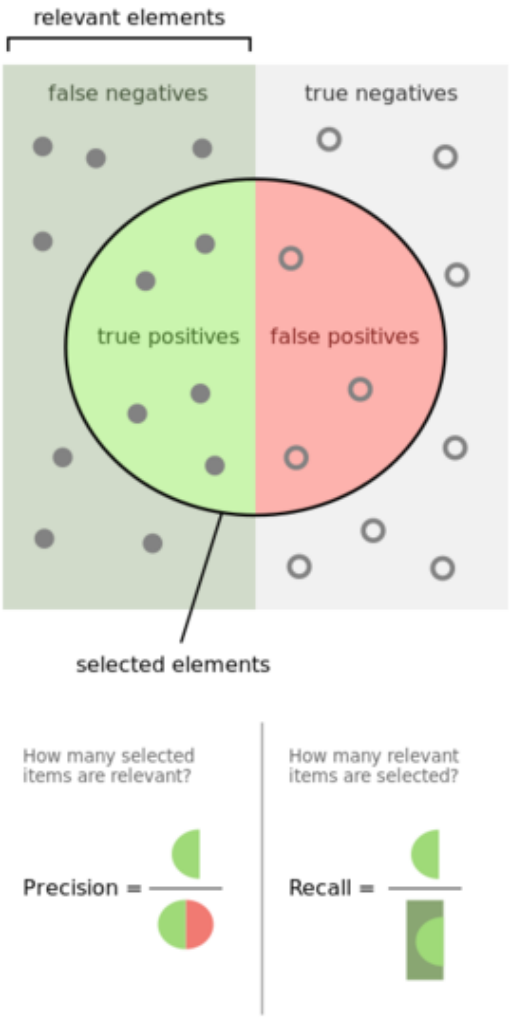



Рисунок 14. Точность и полнота

Иногда недостаточно использования только одной метрики и требуется получить их обобщение. В таком случае можно использовать агрегированный критерий, который называется F-мера:

$$F_\beta = (1+\beta^2)\frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall},$$

где $\beta$ – отвечает за весь precision в данном критерии. В данной работе будет использоваться F1-мера, где $\beta = 1$. Значения данного критерия принадлежат промежутку от 0 до 1, чем больше precision и recall, тем ближе F-мера к единице. Данные метрики подробнее описаны в [2].

Кроме того, также будет использоваться матрица ошибок (confusuion matrix). Данная матрица достаточно полно показывает точность модели. Есть четыре важных понятия, которые необходимы для понимания матрицы ошибок:

- True Positive: Предсказанное и актуальное значения равны 1.
- True Negative: Предсказанное и актуальное значения равны 0.
- False Positive: Предсказанное значение равно 1, а актуальное – 0.
- False Negative: Предсказанное значение равно 0, а актуальное – 1.

Сама матрица ошибок представляется в данном виде:

*Таблица 1*

**Матрица ошибок**

| | y = 1 | y = 0 |
|---|---|---|
| $\hat{y} = 1$ | True Positive (TP) | False Positive (FP) |
| $\hat{y} = 0$ | False Negative (FN) | True Negative (TN) |

В данной матрице: $y$ – актуальное значения, $\hat{y}$ – предсказанное классификатором значение.

На основе матрицы ошибок и ее значений можно ввести формулы для точности и полноты:

- Точность (precision) = $\frac{TP}{TP+FP}$.
- Полнота (recall) = $\frac{TP}{TP+FN}$.

Основной метрикой для оценки качества алгоритмов будет считаться взвешенное среднее F1-меры. Такой выбор сделан, так как недостаточно в данной задаче использовать только одну из двух метрик, требуется отслеживать увеличении обеих. Взвешенное же среднее используется, так как классы в наборе данных являются несбалансированными. Соответственно, таким выбором обеспечивается равноценный учет полноты, точности и классов.

### *3.6. Сравнение точности алгоритмов*

После получения координат ключевых точек из них был сформирован набор данных в виде двухмерной матрицы, где строками являются объекты, одна строка – координаты ключевых точек с одной фотографии, а столбцы – признаки, координаты x и y каждой ключевой точки. Данный датасет для дальнейшего обучения алгоритмов классификации необходимо подготовить. В процессе

подготовки данных были использованы два метода и удалены некоторые столбцы с координатами как не оказывающие влияния на качество методов.

Изначально незначительными признаками были выбраны точки лодыжек, ушей и глаз и удалены из набора данных.

Так как в некоторых столбцах датасета было около 26.5% пропущенных значений, то был выбран метод K-ближайших соседей для заполнения пропусков в данных. Данный алгоритм позволяет заполнять пропущенные значения, обнаруживая ближайших соседей с помощью матрицы евклидовых расстояний и используя их. Подробнее о применении метода K-ближайших соседей для заполнения пропущенных значений написано в [23].

Кроме того, была проведена нормализация данных [2]. В качестве метода для трансформации использовалась Z-нормализация:

$$z = \frac{x - \mu}{\sigma},$$

где μ – среднее выборки, σ – стандартное отклонение.

Оценка точности алгоритмов проводилась в два этапа. На первом этапе весь набор данных был разделен на обучающую, 70% данных, и валидационную, 30%, выборки. Проверка качества проводилась на валидационной части. После этого алгоритм, показавший лучшую точность, проверялся на автономной тестовой выборке. Это необходимо для того, чтобы уточнить качество метода и исключить возможность переобучения.

Таким образом, результаты, полученные на первом этапе представлены далее:

*Таблица 2*

**Результаты работы алгоритмов**

| | Точность | Полнота | F1-мера |
|---|---|---|---|
| Логистическая регрессия | | | |
| 0 | 0.74 | 0.59 | 0.66 |
| 1 | 0.76 | 0.87 | 0.81 |
| Взвешенное среднее | 0.76 | 0.76 | 0.75 |
| Случайный лес | | | |
| 0 | 0.71 | 0.75 | 0.73 |
| 1 | 0.83 | 0.80 | 0.82 |
| Взвешенное среднее | 0.78 | 0.78 | 0.78 |
| Градиентный бустинг | | | |
| 0 | 0.85 | 0.83 | 0.84 |
| 1 | 0.89 | 0.91 | 0.90 |
| Взвешенное среднее | 0.88 | 0.88 | 0.88 |
| Метод опорных векторов | | | |
| 0 | 0.76 | 0.88 | 0.81 |
| 1 | 0.91 | 0.82 | 0.86 |
| Взвешенное среднее | 0.85 | 0.84 | 0.84 |

Также на рисунках 15–18 можно увидеть матрицы ошибок для каждого алгоритма.

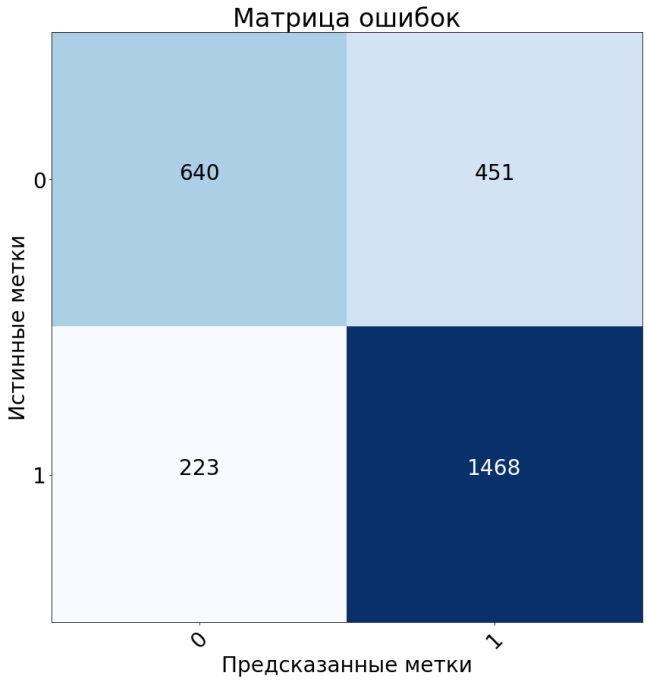



Рисунок 15. Матрица ошибок для логистической регрессии

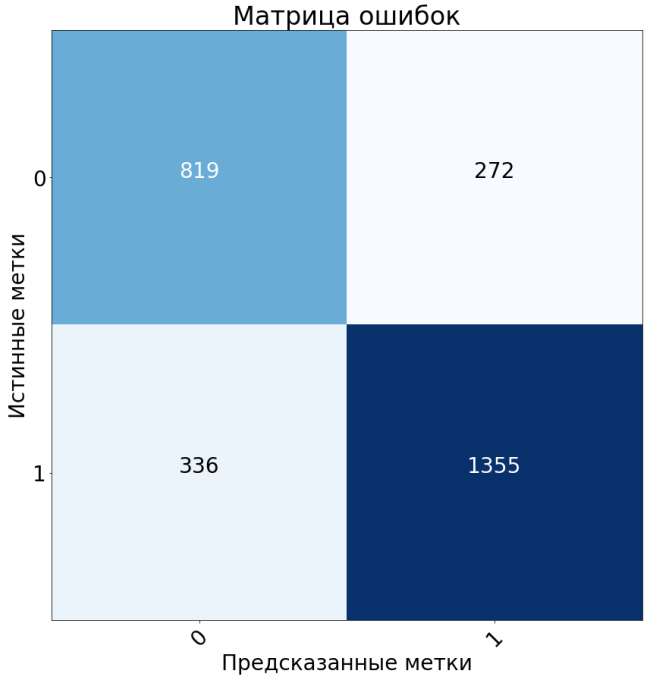



Рисунок 16. Матрица ошибок для случайного леса

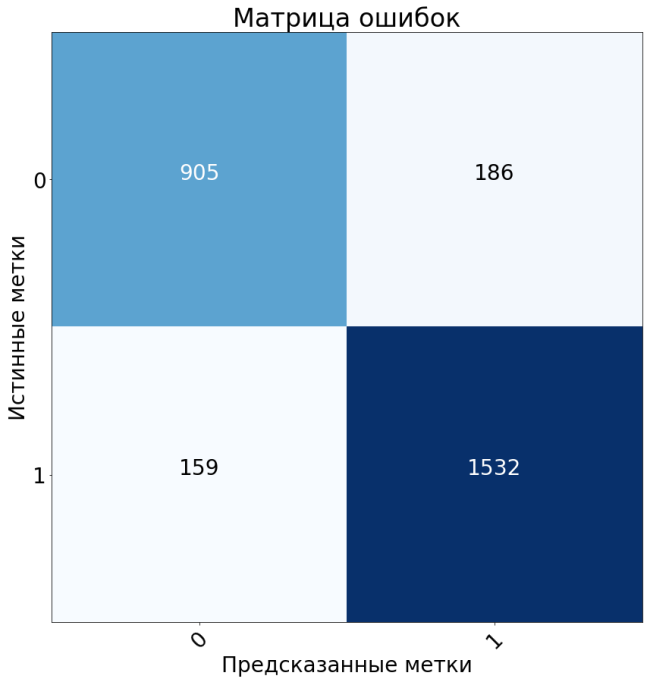



Рисунок 17. Матрица ошибок для градиентного бустинга

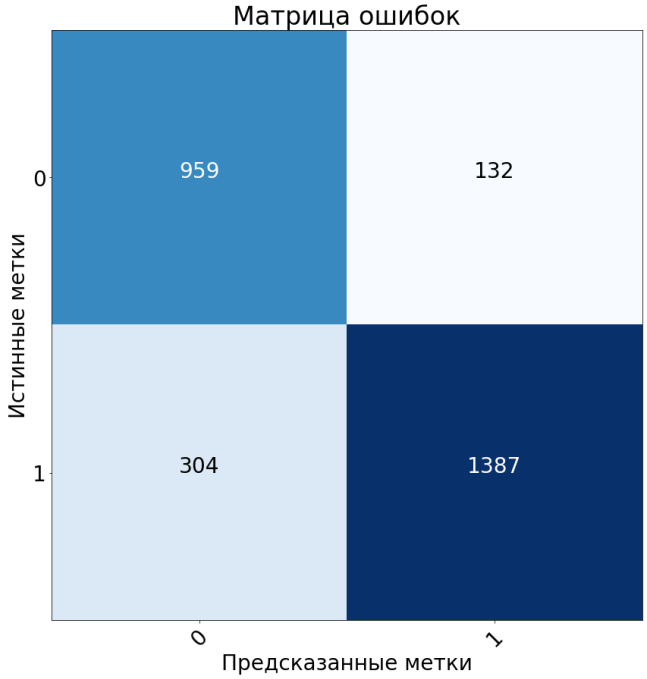



Рисунок 18. Матрица ошибок для метода опорных векторов

На основание полученных результатов можно сделать вывод, что алгоритмы логистической регрессии и случайного леса не обеспечивают необходимую точность для решения задачи по сравнению с градиентным бустингом и методом опорных векторов. Отличия в качестве достаточно сильные

и варьируются от 6 до 13 %. Также видно, что метод опорных векторов уступает градиентному бустингу по взвешенному среднему F1-меры на 4%. Кроме того, по матрице ошибок видно, что бустинг лучше каждого алгоритма определяет неправильную посадку. Таким образом, для дальнейшей реализации выбран алгоритм градиентного бустинга.

# ГЛАВА 4. АНАЛИЗ РЕШЕНИЯ

## *4.1. Оценка выбранного алгоритма на тестовых данных*

Для лучшего понимания качества выбранного алгоритма градиентного бустинга была проведена оценка точности на автономной тестовой выборке. Набор данных для проверки состоял 183 изображений, где были 92 фотографии с правильной посадкой и 91 с неправильной. Полученные на тесте результаты представлены далее в таблице 6 и на рисунке 19.

*Таблица 3*

**Результаты градиентного бустинга на тестовой выборке**

| | Точность | Полнота | F1-мера |
|---|---|---|---|
| 0 | 0.81 | 0.95 | 0.87 |
| 1 | 0.93 | 0.77 | 0.84 |
| Взвешенное среднее | 0.87 | 0.86 | 0.86 |

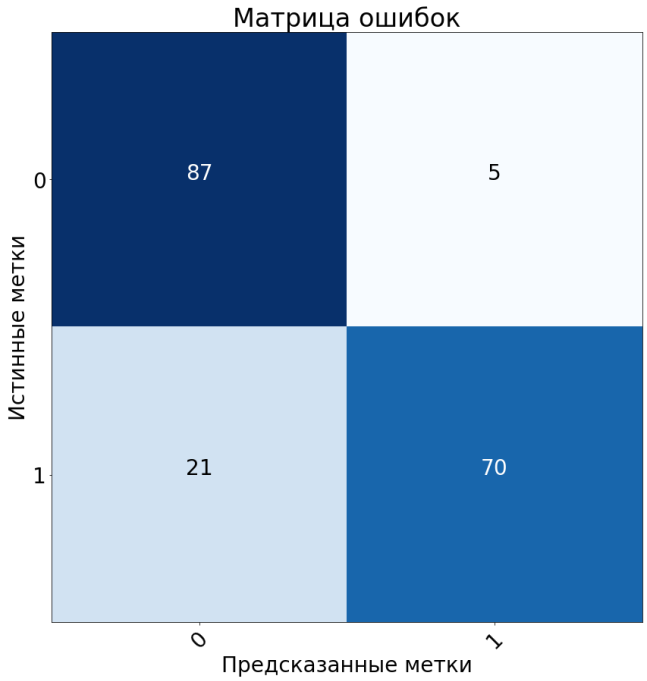



Рисунок 19. Матрица ошибок для градиентного бустинга на тестовой выборке

На основании полученных результатов видно, что произошло небольшое снижение точности на 2%. Также по матрице ошибок можно сделать вывод, что на данном тестовом наборе алгоритм хуже определяет неправильную посадку. Правильно найдено только 77% из всех фотографий с неправильной посадкой. Есть предположение, что это связано с присутствием в тестовом наборе фотографий со сложным углом наклона.

### *4.2. Реализация мониторинга видеопотока*

Для программной реализации мониторинга видеопотока использовались следующие языки программирования и пакеты:

- Python 3.8
- OpenCV 4.5.2
- Sklearn
- Joblib
- Numpy
- Argparse

Были реализованы два варианта работы программы:

- Анализ загруженного видео
- Анализ видео с вебкамеры

Обеспечено считывание одного кадра в секунду для анализа посадки. Такой временной промежуток позволяет получать хорошую производительность – в среднем на анализ одного кадра уходит 0.06 с. Кроме того, в таких условиях алгоритм анализа правильности посадки также имеет возможность быстро реагировать на изменение положения тела человека.

В качестве оповещения в левом верхнем углу видеозаписи выводится одна из меток: “Good” – в случае правильной посадки, “Bad” – в случае неправильной (Рисунок 20, 21).

Рисунок 20. Вывод метки для неправильной посадки

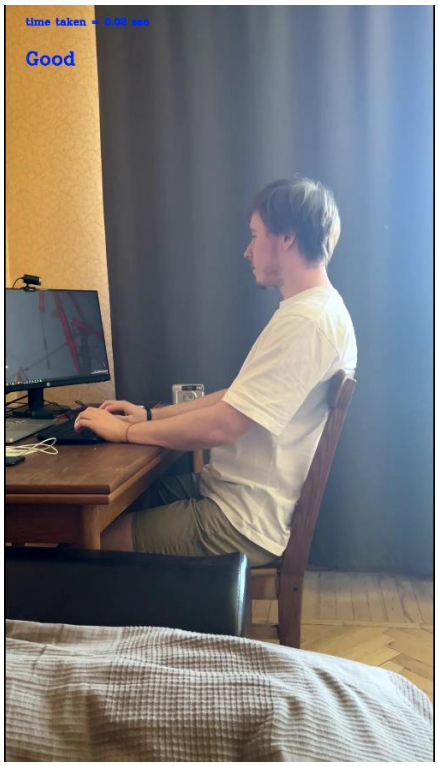



Рисунок 21. Вывод метки для правильной посадки

### *4.3. Анализ мониторинга видеопотока*

Полученная программная реализация была протестирована на 2 видеозаписях. Модель хорошо справилась с задачей оценки позы человека за рабочем местом на загруженных видео. В основном алгоритм выдает верные результаты (Рисунок 22).

Рисунок 22. Пример правильной работы алгоритма

Но так как точность алгоритма не 100%, программа может выдавать и ошибочные результаты (Рисунок 23).

Рисунок 23. Пример неправильной работы алгоритма

Также было установлено, что на анализ 1908 кадров программная реализация тратит 119 секунд или 0.06 с. на один кадр. Такие результаты были получены на ноутбуке с дискретной видеокартой с 4 Гб видеопамяти. Что позволяет без задержек обрабатывать 16 кадров в секунду. Данный результат позволяет использовать алгоритм для мониторинга видеопотока на ноутбуках и персональных компьютерах. Добиться улучшения производительности можно, используя либо различные методы оптимизации, либо более производительные вычислительные машины.

# ЗАКЛЮЧЕНИЕ

В ходе данной работы была реализована программа для мониторинга правильности посадки человека на рабочем месте. Она позволяет отслеживать в режиме реального времени правильное положение, что позволит избежать осложнений, связанных с нарушением осанки. Кроме того, данный метод является более доступным для контроля правильной посадки, чем корсеты и специальные анатомические стулья, так как он финансово дешевле. Особенно в случаях отсутствия серьезных нарушений осанки, когда не требуется серьезного медицинского вмешательства.

Был проведен анализ трех методов для обнаружения ключевых точек человека. Выбрана оптимальная для данной задачи модель, которая справляется с детекцией суставных точек лучше других в случае съемки в профиль и при сложных условиях. Также были собраны фотографии людей в профиль, сидящих за рабочим местом.

После выбора модели для обнаружения ключевых точек, с ее помощью на основе собранной подборки фотографий был получен набор данных для анализа. Были выбраны и протестированы 4 алгоритма классификации. По результатам тестов был сделан выбор в пользу модели градиентного бустинга. Данный алгоритм показал наилучшую точность.

Также выбранный алгоритм был проверен на отдельной тестовой выборке, где его качество подтвердилось, и было исключено предположение о переобучении.

На следующим этапе была осуществлена программная реализация с использованием обученного алгоритма градиентного бустинга. Которая позволила обрабатывать непрерывный видеопоток: как загруженное в программу видео, так и трансляцию с вебкамеры.

В итоге была получена программа, которая позволяет определять правильность посадки на рабочем месте в режиме реального времени, используя только камеру, подключенную к ноутбуку.

Таким образом, можно считать, что цель выпускной квалификационной работы выполнена и задачи решены полностью.

В дальнейшем возможны такие направления развития работы, как увеличение набора данных для повышения точности, дообучение модели, определяющей точки, на фотографиях сидящих людей, оптимизация программы для повышения производительности на слабых устройствах и реализация веб-приложения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Горячкин Б.С., Китов М.А. Компьютерное зрение // E-Scio. – 2020. – №9 (48).
2. Рашка С., Мирджалили В. Python и машинное обучение. Машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow. – СПб.: ООО «Диалектика», 2019.
3. Рубцова И.В., Кубышкина Т.В., Лукшина И.В. Осанка. Средства и методы оценки и коррекции: учебно-методическое пособие для вузов. – Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2008. – 22 с.
4. Ali Muthanna M.S., Lyachek Y.T., Obadi Musaeed A. M., Ahmed Hazzaa Esmail Y. and Adam A. B. M. Smart System of a Real-Time Pedestrian Detection for Smart City // IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). – 2020. – P. 45-50.
5. Andriluka M., Pishchulin L., Planck M., Schiele B. 2d human pose estimation: New benchmark and state of the art analysis // Proceedings of the IEEE Conference on computer Vision and Pattern Recognition. – 2014. – P. 3686-3693.
6. Cao Z., Hidalgo G., Simon T., Wei S.E, Sheikhet Y. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields // IEEE transactions on pattern analysis and machine intelligence. – 2019. – Vol. 43. – №.1. – P. 172-186.
7. Dar P. ‘AI Guardman’—A machine learning application that uses pose estimation to detect shoplifters // Anal. Vidhya. – 2018.
8. Dorogush A. V., Ershov V., Gulin A. CatBoost: gradient boosting with categorical features support // ArXiv.org. – 2018.

9. Goodfellow I., Bengio Y., Courville A. Deep learning: textbook. – Cambridge: MIT press, 2016. – Vol. 1. – №. 2.

10. Hecht-Nielsen R. Theory of the backpropagation neural network // Neural networks for perception. – Academic Press, 1992. – P. 65-93.

11. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. – 2012. – Vol. 25. – P. 1097-1105.

12. Lai A., Reddy B. and Vlijmen B., Yog.ai: deep learning for yoga // Stanford University, CA. CS230: Deep Learning. – 2019. URL: http://cs230.stanford.edu/projects_winter_2019/reports/15813480.pdf (дата обращения: 12.05.2021)

13. Lin T. Y. et al. Microsoft COCO: Common objects in context // European conference on computer vision. – Springer, Cham. –2014. – P. 740-755.

14. Mehta D. et al. Xnect: Real-time multi-person 3d human pose estimation with a single rgb camera // ArXiv.org. – 2019. URL: https://arxiv.org/pdf/1907.00837v1.pdf (дата обращения: 12.05.2021)

15. Natekin A., Knoll A. Gradient boosting machines, a tutorial // Frontiers in neurorobotics. – 2013. – Vol. 7. – P. 21.

16. Papandreou G. et al. Towards accurate multi-person pose estimation in the wild // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2017. – P. 4903-4911.

17. Papandreou G., Zhu T, Chen L.C., Gidaris S., Tompson J., Murphy K. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model // Proceedings of the European Conference on Computer Vision (ECCV). – 2018. – P. 269-286.

18. Prathima T., Varma VSVS, Prathyush K., Govardhan A., Detection of Armed Assailants in Hostage Situations-A Machine Learning based approach // Third

International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). –2021. – P. 1-8.

19. Sigal L. Human pose estimation. Encyclopedia of Computer Vision: encyclopedia. – Springer, 2011.
20. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition // ArXiv.org. – 2014. URL: https://arxiv.org/abs/1409.1556 (дата обращения: 12.05.2021)
21. Wang B. H. et al. Fall Detection Based on Dual-Channel Feature Integration // IEEE Access. – 2020. – Vol. 8. – P. 103443-103453.
22. Wei S.E., Ramakrishna V., Kanade T., Sheikh Y. Convolutional pose machines // Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. – 2016. – P. 4724-4732.
23. Zhang S. Nearest neighbor selection for iteratively kNN imputation //Journal of Systems and Software. – 2012. – Vol. 85. – №11. – P. 2541-2552.