

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий  
(наименование института полностью)

Кафедра «Прикладная математика и информатика»  
(наименование)

01.03.02 Прикладная математика и информатика

(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование  
(направленность (профиль)/специализация)

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)**

на тему «Разработка интеллектуальной системы распознавания лиц»

Обучающийся

А.С. Губарев

(Инициалы Фамилия)

(личная подпись)

Руководитель

М.А. Тренина

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант

канд. пед. наук Т.С. Якушева

(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

## Аннотация

Тема выпускной квалификационной работы: «Разработка интеллектуальной системы распознавания лиц».

Работа выполнена студентом Тольяттинского государственного университета, института математики, физики и информационных технологий, группы ПМИб-1802а, Губаревым Александром Станиславовичем.

Объект исследования: анализ систем распознавания лиц.

Цель работы: разработка системы распознавания лиц для идентификации людей на проходном пункте.

Для достижения цели работы необходимо решить следующие задачи:

- Изучить и сравнить методы обнаружения и распознавания лиц.
- Сравнить эффективность существующих систем распознавания.
- Определить основные инструменты и средства для разработки.
- Реализовать модуль распознавания лиц.

Отчет состоит из введения, трех разделов и заключения.

В первом разделе описываются алгоритмы обнаружения, распознавания и трансформации лиц.

Во втором разделе описаны выбранные алгоритмы для разработки программы, а также приводится описание ее работы.

В третьем разделе производится тестирование программы и анализ результатов ее работы.

Бакалаврская работа выполнена на 52 страницах, состоит из введения, трех разделов, заключения, списка литературы, состоящего из 26 литературных источников и 13 рисунков.

## **Abstract**

The title of the graduation work is: "Development of an artificial intelligent face recognition system".

The senior paper consists of an introduction, three parts, a conclusion, one scheme, list of references including foreign sources and 13 images.

The key issue of the thesis is the design of a face recognition system for identifying people at the checkpoint. We will touch upon the problem of the security of the company and its employees, as well as the problem of the convenience of using other technologies of the data control and management system.

The aim of the work is to model a face recognition system for identifying people at a checkpoint.

The graduation work may be divided into several logically connected parts which are the following: description of face detection and recognition algorithms, as well as methods for their transformation; analysis of existing face recognition systems; directional gradient histogram algorithm; data classification by support vector machine; construction of face key points using an ensemble of decision trees; extraction of key facial features using a convolutional neural network.

In conclusion, I would like to emphasize that this work is relevant for increasing the level of security and convenience in identifying people at the checkpoint. The algorithms used in the study are relevant both for a certain region and for all subjects of the Russian Federation.

## Содержание

Введение.....	5
1 Исследование предметной области .....	6
1.1 Постановка задачи .....	7
1.2 Алгоритмы распознавания лиц.....	8
1.3 Предварительная трансформация изображения .....	13
1.4 Анализ эффективности систем распознавания .....	14
2 Реализация системы распознавания лиц.....	18
2.1 Алгоритм HOG для обнаружения лиц .....	18
2.2 Классификация данных методом опорных векторов (SVM).....	23
2.3 Детектирование алгоритмом гистограммы направленных градиентов и классификация методом опорных векторов.....	28
2.4 Ансамбль решающих деревьев.....	28
2.5 Сверточная нейронная сеть (CNN) .....	31
2.6 Программная разработка модулей системы.....	34
2.6.1 Реализация модуля сохранения кодировок в файл (data.py). .....	35
2.6.2 Реализация модуля распознавания лиц (recognition.py) .....	37
3 Тестирование и анализ результатов .....	45
Заключение .....	49
Список используемой литературы и используемых источников.....	50

## Введение

На сегодняшний день многие предприятия используют системы контроля и управления доступом (СКУД), благодаря которым повышается уровень безопасности компании и сотрудников. С помощью такой системы разграничивается доступ к определенным объектам или территориям для конкретных лиц.

Ранее, для ограничения доступа использовались турникеты, к считывателям которых прикладывался один из идентификаторов: магнитная карта; брелок; отпечаток пальца. Однако, в связи с сильным ростом биометрических технологий, сегодня существуют более точные и удобные методы идентификации. Например, распознавание лиц по видеопотоку в режиме реального времени.

Распознавание лиц по видеопотоку гораздо более удобный способ идентификации, так как человеку надо лишь оказаться в поле зрения видеокамер, а все остальное сделает система. Также, благодаря распознаванию лиц решается проблема отсутствия идентификатора, ведь человек не сможет потерять или оставить дома свое лицо.

В связи с этим была поставлена цель данной работы – разработка интеллектуальной системы распознавания лиц.

Для достижения поставленной цели потребуется:

- Изучить и сравнить методы обнаружения и распознавания лиц.
- Сравнить эффективность существующих систем распознавания.
- Определить основные инструменты и средства для разработки.
- Реализовать модуль распознавания лиц.

## 1 Исследование предметной области

Распознавание лиц – процесс выполнения различных задач, в результате которых происходит идентификация человека по фото или видео.

В общем виде это выглядит следующим образом:

- Обнаружение. Система получает изображение с камеры, на котором с помощью специального алгоритма производится поиск лица.
- Трансформация. Далее найденное лицо подвергается трансформации (коррекция яркости, масштабирование и другие аффинные преобразования) с целью привести его к виду заданного шаблона.
- Идентификация. Затем система извлекает ключевые признаки обнаруженного лица и ищет максимальное совпадение этих признаков с известными ей в базе данных. Название последнего шага зависит от типа системы распознавания [3].

Верификация: сравнение образцов по схеме «1:1». Чтобы идентифицировать личность система сравнивает обнаруженное лицо с одним известным, и дает ответ на вопрос «Одинаковые ли эти лица?».

Идентификация: сравнение образцов по схеме «1:N». Чтобы идентифицировать личность система сравнивает обнаруженное лицо со всеми известными ей и дает ответ на вопрос «Известен ли этот человек системе?».

На рисунке 1 изображена блок-схема, описывающая общий алгоритм распознавания лиц.

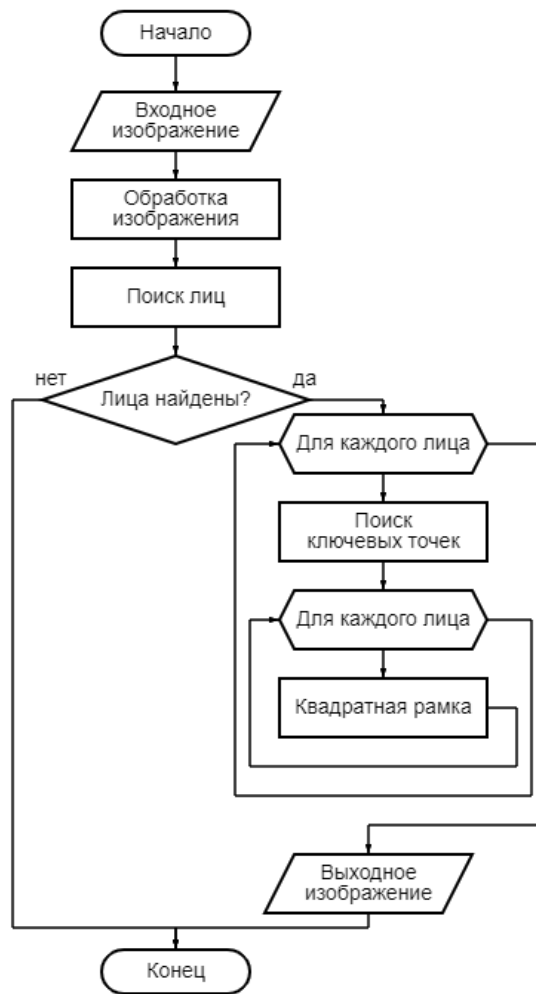


Рисунок 1 – Работа алгоритма распознавания лиц

## 1.1 Постановка задачи

Результатом данной работы должно являться создание системы распознавания лиц, способной идентифицировать лица сотрудников университета, попавшие в кадр видеокамер. Чтобы выполнить поставленную задачу требуется:

- изучить алгоритмы распознавания лиц;
- сравнить эффективность алгоритмов;
- определить основные инструментальные средства для разработки;
- спроектировать систему распознавания.

## 1.2 Алгоритмы распознавания лиц

Работа каждого алгоритма обнаружения лиц основана на загрузке и манипуляции над изображением. Изображение загружается в систему в виде данных, после чего алгоритм начинает его обработку и ведет поиск области, в которой может находиться лицо. На сегодняшний день было представлено множество подобных алгоритмов. Большинство из них – комбинация нескольких методов, которые работая в паре дают большую эффективность, однако их всех можно разделить на два вида: методы, основанные на знаниях и методы, основанные на обнаружении по внешним признакам.

Методы, основанные на знаниях, используют информацию о форме, цвете или яркости человеческого лица. С помощью этих особенностей создается набор специальных правил, которым должен удовлетворять фрагмент изображения, чтобы он считался лицом. Такой набор правил определяется знаниями человека, которыми он пользуется для определения, видит ли он сейчас лицо или нет.

Несколько основных правил: области глаз, носа и рта различаются по яркости относительно остального лица; глаза на лице расположены приблизительно на одном уровне друг от друга. Основываясь на этом, были разработаны алгоритмы, которые используют эти правила в процессе своей работы [19].

К этой же группе методов можно отнести метод сравнения с шаблонами. В нем существует некий шаблон, описывающий определенные свойства лица, с которым в процессе работы сравнивается изображение.

Методы, основанные на знаниях являются довольно известными и неплохо справляются с поставленными задачами, однако для достижения хороших результатов входные изображения должны быть высокого качества, не нести в себе шумов, и иметь несложный фон. При использовании данного метода камерами видеонаблюдения не стоит ожидать отличных результатов,



так как изменение ракурса, освещения и сложный задний фон ухудшают точность распознавания [20].

Методы обнаружения лиц по внешним признакам работают совершенно противоположно вышеупомянутым. Вместо подражанию человеческому мозгу они стараются выявить закономерность и свойство изображения лица, используя машинное обучение и математическую статистику. Отсюда следует, что этим методам не свойственны вышеупомянутые недостатки, из-за чего именно их чаще стали использовать в системах видеонаблюдения. Обнаружение лиц в данных методах выполняется поиском всех прямоугольных областей изображения для последующего присвоения одного из двух классов изображению: изображение содержащее лицо, или изображение, в котором лицо отсутствует.

Недостатком этого метода является высокая вычислительная сложность из-за большого объема работы. Для ускорения процесса и уменьшения количества вычислений авторы используют различные методы по сокращению операций рассматриваемых областей.

Некоторые актуальные и популярные методы, а также их достоинства и недостатки:

Метод Виолы-Джоса (Viola–Jones object detection). «Одним из самых известных и популярных алгоритмов в области компьютерного зрения является алгоритм Виолы-Джонса (Viola-Jones). Он был разработан Полом Виолой и Майклом Джонсом в 2001 году, и получил широкое применение в силу своей скорости и эффективности» [15]. Алгоритм использует модифицированные признаки Хаара для поиска лиц на изображении. Существует множество реализаций данного метода, например, в библиотеке компьютерного зрения OpenCV [24].

Преимущества:

- метод Виолы-Джонса имеет высокую скорость работы за счет применения каскадного классификатора;

- имеет хорошую точность распознавания повернутых лиц на угол до 30 градусов (при увеличении угла точность сильно падает).

Недостатки:

- длительное время обучения классификатора, поскольку требуется проанализировать большое количество изображений для его обучения.

Метод гибкого сравнения на графах (Elastic Graph Matching) [7][9]. Лица в методе представляются в виде графов со взвешенными ребрами и вершинами.

Процесс распознавания лиц происходит следующим образом – один из графов является эталонным и остается неизменным, в другом же начинается деформация (поочередное смещение вершин в определенном направлении на некоторое расстояние относительно их изначального местонахождения) с привязкой к антропометрическим точкам лица (линия глаз, ширина носа, расстояние между глазами и т.п.). Деформация графа продолжается до тех пор, пока между эталонным и деформированным графом не будут достигнуты минимальные значения различия признаков.

Преимущества:

- данный метод имеет высокую точность распознавания лиц и допускает их поворот до 15 градусов.

Недостатки:

- высокая сложность алгоритмов распознавания требует больших вычислительных мощностей;
- сложность введения новых эталонов в базу данных;
- линейная зависимость скорости работы системы от размеров базы данных, так как, метод относится к методам перебора.

Скрытые Марковские модели (СММ, НММ) [11]. Метод относится к статистическим методам распознавания лиц. СММ используют

статистические свойства сигналов и учитывают непосредственно их пространственные характеристики. Элементами модели являются: множество скрытых состояний, множество наблюдаемых состояний, матрица переходных вероятностей, начальная вероятность состояний. Каждому соответствует своя Марковская модель. Во время распознавания лица, проверяются все сгенерированные Марковские модели и происходит поиск наибольшей из наблюдаемых вероятности того, что последовательность наблюдений для объекта сгенерирована соответствующей моделью.

Недостатки:

- низкая скорость распознавания;
- плохая различающая способность;
- не оптимальный алгоритм обучения<sup>4</sup>
- нет возможности оптимизировать скорость перебора других моделей.

Метод главных компонент (РСА)[7][9]. Данный метод был представлен Карлом Пирсоном в 1901 году. Целью метода является уменьшение размерности данных, минимизировав потери информации. Этот метод использует в задачах распознавания лиц для представления лица в виде вектора малой размерности, и дальнейшего сравнения с эталонами из базы данных. Полученный на обучающей выборке набор собственных векторов применяется для кодирования других изображений лиц, которые можно представить взвешенной комбинацией собственных векторов. При использовании ограниченного количества собственных векторов можно получить сжатую аппроксимацию входному изображению лица, которую впоследствии можно хранить в БД, как вектор коэффициентов, который служит одновременно ключом поиска в БД.

Метод главных компонент часто встречается в приложениях, однако он имеет недостаток, в виде значительного падения эффективности при изменении освещенности или мимики лица. Это происходит из-за того, что

метод выбирает подпространство с целью максимальной аппроксимации входного набора данных, а не с целью выполнения дискриминации между классами лиц.

Метод опорных векторов (Support Vector Machines, SVM) – линейный алгоритм, который используется в задачах классификации и регрессии [5]. Целью данного метода является нахождение гиперплоскости в признаковом пространстве, которая разделяет два класса изображений: изображения с лицами и без лиц. Кроме того, следует выбрать оптимальную разделяющую гиперплоскость, то есть ту, до которой расстояние от каждого класса максимально.

Преимущества:

- устойчивость к переобучению;
- высокая скорость распознавания в сравнении с нейронными сетями;
- возможность оптимизировать чувствительность к шуму путем уменьшения точности работы.

Недостатки:

- существуют более точные алгоритмы распознавания лиц.

Сверточная нейронная сеть (CNN)[9]. С помощью машинного обучения данный алгоритм тренируется извлекать ключевые характеристики лица на основе готовых примеров. Таким образом нейросеть получает опыт, который в дальнейшем применяется для распознавания ранее неизвестного объекта. Нейронные сети отличаются своей высокой точностью к распознаванию и устойчивостью к искажениям шумам на изображениях, но считаются одними из самых сложных алгоритмов для реализации.

Преимущества:

- высокая точность распознавания;
- устойчивость к искажениям и шумам на изображении.

Недостатки:

- высокая сложность алгоритма требует больших вычислительных мощностей;
- внесение изменений тренировочную выборку требует полного переобучения нейронной сети.

Гистограмма направленных градиентов (HOG) [12]. Основная идея алгоритма заключается в том, что изображение может быть описано распределением градиентов интенсивности или направления краев. Построение этих дескрипторов происходит с помощью разбиения изображения на ячейки, и присвоения каждой ячейке гистограммы направлений градиентов для пикселей внутри ячейки, их комбинация и является дескриптором. Для повышения точности работы алгоритма изображения делают черно-белым, а локальные гистограммы нормализуют по контрасту относительно меры интенсивности, вычисляемой на большем фрагменте изображения.

Преимущества:

- нормализация по контрасту позволяет добиться большей инвариантности дескрипторов к освещению;
- Поддерживает инвариантность геометрических и фотометрических преобразований, кроме ориентации объекта.

### **1.3 Предварительная трансформация изображения**

Перед тем как начать распознавание лиц входные изображения с камер видеонаблюдения предварительно обрабатывают для повышения эффективности алгоритма.

Как только система находит лицо на изображении она делает его небольшой снимок, который подвергается нормализации: для начала на снимке проводится поиск и построение антропометрических точек лица. Затем он масштабируется, сдвигается, поворачивается таким образом, чтобы

лицо максимально соответствовало эталону. Так же существуют специальные фильтры, которые применяются для удаления различных видов шумов на изображении. [16]. Одни из наиболее часто используемых представлены ниже:

- нелинейные фильтры – фильтры, использующееся для удаления импульсного шума (отдельных точек с максимальной или минимальной яркостью). Данный фильтр ищет позицию импульса, после чего заменяет его на заранее заданное или случайное значение;
- фильтр Гаусса используется для размытия мелких деталей, которые не требуют отделения от фона;
- медианные фильтры – цифровой фильтр, целью которого является удаление шумов с изображения. Медианный фильтр особенно хорошо справляется с мелкими шумами.

#### **1.4 Анализ эффективности систем распознавания**

На сегодняшний день существует множество различных алгоритмов, каждый из которых имеет свою скорость и точность.

Алгоритмы распознавания делятся на две категории – двумерную (2D), в которой используется распознавание с применением технологии геометрии лица и трехмерную (3D), в которой используется распознавание с применением технологии геометрии черепа [14].

Двумерные системы работают с двумерными изображениями, такими как видеопоток с камер наблюдения. Они выполняют поиск лица на изображении, анализируя его текстуру и участки контрастности. Недостатками 2D систем являются нарушение освещения и поворот головы из-за чего эффективность распознавания падает.

Трехмерные системы менее зависимы от подобных изменений, т.к. распознавание человека идет по строению его черепа. Однако в связи с

высокой вычислительной сложностью и невозможностью обслуживания большого количества пользователей 3D системы пока не обрели широкого применения. Также, еще один недостаток – высокая стоимость оборудования для создания трехмерной системы в разы больше, чем двумерной.

Сегодня в системах видеонаблюдения используется несколько эффективных алгоритмов распознавания. Они основаны либо на характерных точках, либо на значениях пикселей. Ниже представлено краткое описание каждой из этих подгрупп.

Алгоритмы, основанные на значениях пикселей, используют для распознавания обнаруженных лиц цвет или яркость. Простейшими подобными алгоритмами являются:

Eigenfaces – алгоритм представленный в 1991 году Мэтью Терком и Алексом Пентландом получил широкую известность в качестве первого успешного метода распознавания лиц [25].

В процессе работы алгоритм представляет ключевые характеристики лица в виде двумерных изображений в градациях серого. Эти изображения сравниваются с известными лицами из базы данных. В результате сравнения вычисляется коэффициент различия, который определяет степень схожести.

Данный алгоритм эффективен при использовании в хорошо освещенных местах, а также при распознавании лица человека в анфас. Однако при изменении этих условий результаты работы алгоритма резко снижаются [18].

Fisherfaces – потомок Eigenfaces. Он стал более устойчивым к изменениям освещения и мимики лица, что дало существенное улучшение эффективности работы алгоритма в подобных условиях. В отличие от своего предшественника, основой Fisherfaces стал линейный дискриминант Фишера [7].

Действие алгоритма Fisherfaces основано на поиске проекции данных, при которой классы изображений лиц максимально разделимы. Именно благодаря этому отличию от Eigenfaces была решена проблема сильной

чувствительности к изменениям освещения. Недостатком является высокая вычислительная сложность по сравнению с другими алгоритмами.

Алгоритм локальных бинарных шаблонов (LBP) одновременно рассматривает девять пикселей изображения – точнее, матрицу  $3 \times 3$  – и уделяет особое внимание центральному пикселю [1]. LBP сравнивает интенсивность (яркость) центрального пикселя с соседом. Затем, каждому соседнему пикселю присваивается значение 0, если его яркость меньше центрально, иначе 1. В результате сравнения такой матрицы получается двоичное восьмиразрядное число. В дальнейшем полученные гистограммы конкатенируются и сравниваются друг с другом с использованием одного из методов машинного обучения. Алгоритм устойчив к изменению уровня освещения, но имеет недостаток в виде высокой вычислительной сложности.

Активная модель внешнего вида (ААМ) [9]. ААМ представляет собой статистическую модель, включающую два аспекта: построение модели и алгоритм подбора. ААМ изучает характеристики объектов, строя компактную статистическую модель. Статистическая модель, которая представляет изменения формы и текстуры объектов, получается путем применения анализа основных компонентов (РСА) к набору размеченных данных. В ААМ объект описывается набором ориентиров, которые указывают важные позиции на границе объекта. Ориентиры помечаются вручную на каждом объекте в обучающих данных. Перед использованием модель требует обучения на заранее размеченном множестве изображений. Из недостатков – данный алгоритм сильно чувствителен к мимике и углам поворота головы и уровню освещенности, а требования к обучающим данным гораздо выше в сравнении с другими алгоритмами.

Активные модели формы (ASM) – это алгоритм, основанный на модели распределения антропометрических точек. Когда система просматривает изображение, она связывается с чертами лица, такими как нос, рот и т. д., как только обнаружит близость к какой-либо из этих черт. Координаты этих точек берутся как карта, и отсюда создается маска, которую можно изменить



вручную. Даже если система принимает решение о форме, ее может настроить пользователь. Тренируясь с большим количеством изображений, можно получить лучшую карту.

Вывод по первому разделу.

В первом разделе происходит исследование предметной области.

Производится постановка задач, показывается блок-схема большинства алгоритмов обнаружения лиц, позволяющая наглядно увидеть их работу. Происходит изучение и анализ существующих алгоритмов обнаружения и распознавания лиц, их сравнение и выявление преимуществ и недостатков. Также, рассматриваются различные популярные фильтры обработки изображений.

## **2 Реализация системы распознавания лиц**

Перед тем, как начать реализовывать систему распознавания лиц требуется выбрать алгоритмы, которые будут лежать в ее основе. Изучив то, как устроены алгоритмы, проанализировав их положительные и отрицательные качества, было принято решение использовать гистограмма направленных градиентов (HOG) обнаружения лица, ансамбль деревьев регрессии (случайный лес) для построения ключевых точек, сверточная нейронная сеть (CNN) для вычисления ключевых признаков и линейный метод опорных векторов (SVM) для классификации.

### **2.1 Алгоритм HOG для обнаружения лиц**

«Впервые гистограмма направленных градиентов была представлена в работе Навнит Далалом и Биллом Триггсом в июне 2005 г. Они применяли этот метод для распознавания пешеходов на статичных изображениях. В настоящее время этот метод широко используется не только для нахождения пешеходов, но и распознавания лиц, автомобилей и других объектов на видеопоследовательностях.» [13]

Алгоритм состоит из векторного пространства, которое вычисляет сходство с использованием евклидовых или косинусных расстояний, что хорошо подходит для методов машинного обучения. В его основе лежит предположение, что вид распределения градиентов интенсивности изображения позволяет достаточно точно определить наличие и форму присутствующих на нем объектов [26].

Первый шаг заключается во многих алгоритмах поиска особых точек используют нормализацию цвета и гамма-коррекцию, однако Далал и Триггс выяснили, что их применение принесет тот же результат, что и последующая нормализация. Поэтому первый шаг в алгоритме HOG заключается в расчете

значений градиентов. Для этого используются в применении одномерных производных масок как в вертикальном, так и в горизонтальном направлениях. Размеры используемых здесь масок – 1X3 и 3X1 (рисунок 2).

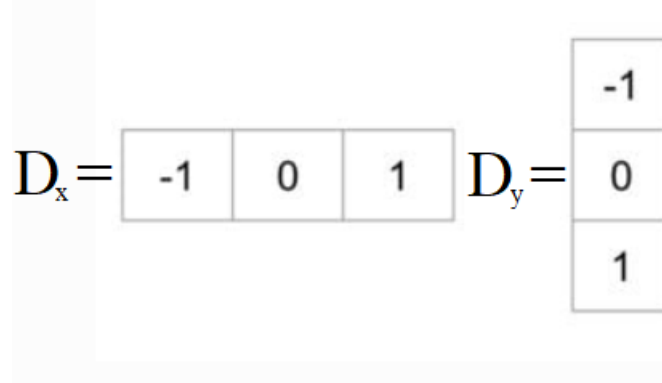


Рисунок 2 – Используемые маски

Для вычисления градиента в точке используются следующие формулы (1-4):

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y), \quad (1)$$

$$G_y(x, y) = H(x, y + 1) - H(x, y - 1), \quad (2)$$

$$G = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}, \quad (3)$$

$$\theta(x, y) = \tan^{-1} \left( \frac{G_y(x, y)}{G_x(x, y)} \right), \quad (4)$$

где:

- $G_x$  – горизонтальный градиент точки  $x, y$ .
- $G_y$  – вертикальный градиент точки  $x, y$ .
- $H$  – значение пикселя  $x, y$ .
- $G$  – размер градиента  $x, y$ .
- $\theta$  – направление градиента  $x, y$ .

Результаты обработки изображения алгоритмом с применением разных масок можно наблюдать на рисунках 3(а-г) .

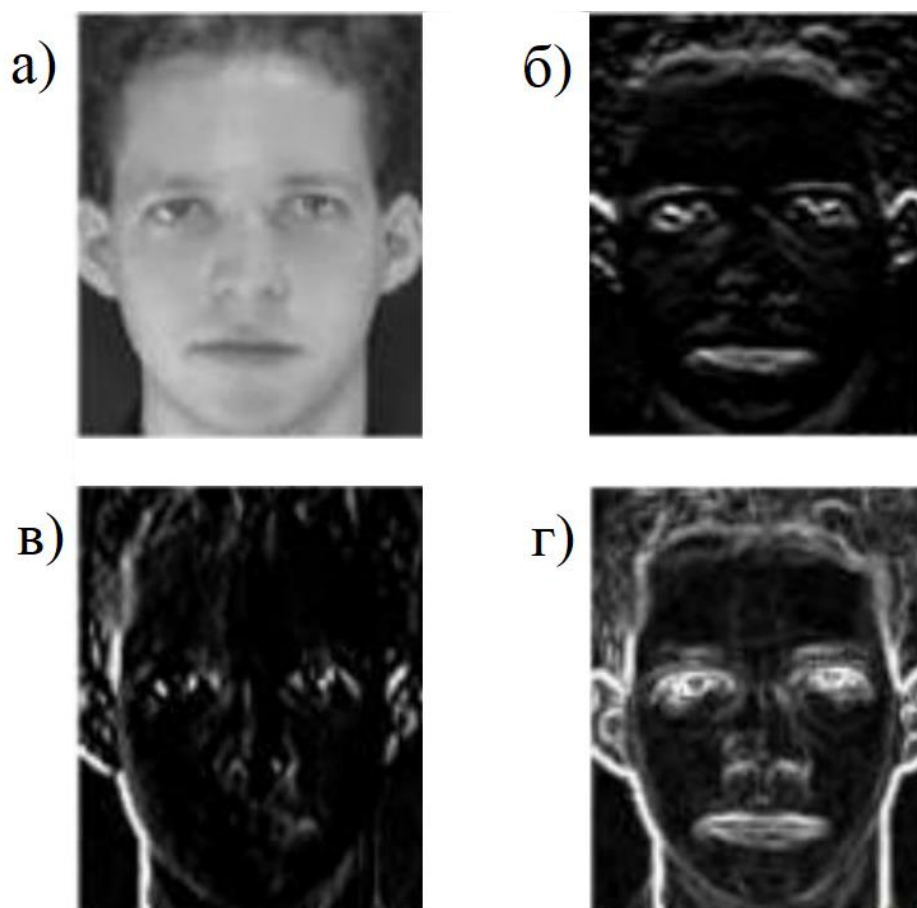


Рисунок 3а – Изображение до применения фильтров; 3б – результат применения горизонтальной маски; 3в – результат применения вертикальной маски; 3г – результат применения вертикальной и горизонтальной маски одновременно.

На следующем этапе происходит группировка направлений. Каждый пиксель, основываясь на значении и направлении градиента, отдает свой голос в один из каналов (интервалов) гистограммы. Сама гистограмма представляет собой вектор (или массив) чисел, в котором интервалы соответствуют углу градиента. Таким образом, если величина градиента равна 5, а его угол 20 градусов, тогда его значение будет записано в канал, отвечающий за этот угол. При вычислении «знакового» градиента – каналы

равномерно распределены от 0 до 360 градусов, в «беззнаковом» они распределяются от 0 до 180 градусов. Значение пикселя во время голосования может быть задано корнем или квадратом градиента, либо его абсолютным или урезанным значением [6]. Во время исследования эффективности алгоритма было установлено, что вычисление «беззнакового» градиента и использование 9 каналов гистограммы показывает гораздо более высокие результаты при распознавания людей.

На рисунке 4(а) показана операция группировки направлений. Образец изображения лица разбит на ячейки. Рисунок 4(б) представляет собой соответствующие гистограммы клеток.

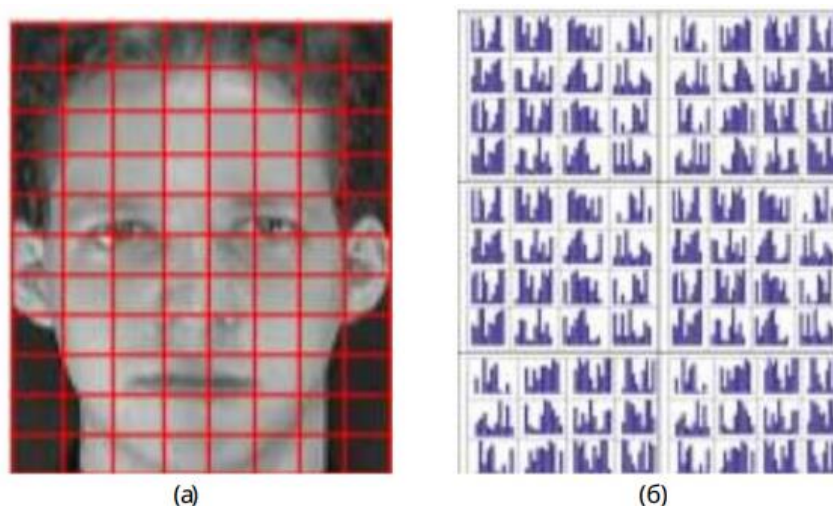


Рисунок 4 – Группировка направлений

Далее следует провести локальное нормирование градиентов для учета изменений контраста и освещения. Для этого требуется группировать ячейки в более крупные и пространственно-связанные блоки. В этом случае дескриптор гистограммы ориентированных градиентов является вектором компонент гистограмм ячеек, нормированных из всех областей блока. Эти блоки обычно перекрываются, что означает, что каждая ячейка вносит свой вклад в окончательные дескрипторы как минимум более одного раза.

Существует два типа геометрии блоков: прямоугольные (R-HOG) и круговые (C-HOG) блоки гистограммы направленных градиентов. Блоки R-HOG представляют собой прямоугольные или квадратные сетки, которые характеризуются тремя параметрами: ячейки на каждый блок, пиксели на каждую ячейку и каналы на каждую гистограмму. В эксперименте по обнаружению человеческого лица наиболее благоприятными параметрами оказались четыре ячейки 8X8 пикселей (16X16 пикселей в каждом блоке) с 9 каналами гистограммы.

Следующим шагом стоит нормализация блоков. Коэффициент нормализации можно получить одним из следующих способов [13]:

$$\text{L2-норма: } f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}, \quad (5)$$

$$\text{L2-hys: } f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}, \text{ если } v > 0,2; v = 0,2, \quad (6)$$

$$\text{L1-норма: } f = \frac{v}{\sqrt{\|v\|_1 + e}}, \quad (7)$$

$$\text{L1-квадрат: } f = \sqrt{\frac{v}{\|v\|_1 + e}}, \quad (8)$$

где:

- $v$  – не нормированный вектор, содержащий все гистограммы в данном блоке,
- $\|v\|_k$  –  $k$ -норма этого вектора для  $k = 1, 2$ ,
- $e$  – некоторая малая константа.

В результатах исследования Далал и Триггс обнаружили, что схема L2-норма, L2-hys и L1-квадрат (5,6,8) имеют примерно равную эффективность, а L1-норма (7) обеспечивает менее надежную работу. Но в итоге все методы

показали значительное улучшение по сравнению с ненормализованными данными.

## 2.2 Классификация данных методом опорных векторов (SVM)

Метод опорных векторов используется для решения задач классификации и регрессии. Он представляет каждый объект данных как вектор в пространстве, каждый из которых принадлежит одному из двух классов. Его цель – построить гиперплоскость, которая разделяющую эти два класса. Однако в результате может быть построено множество плоскостей, поэтому требуется выбрать такую, расстояние до которой от близлежащей точки каждого класса будет максимальным. Такая разделяющая гиперплоскость называется оптимальной [4].

Дана обучающая выборка  $D$ , а набор данных состоит из  $n$  объектов:

$$D = (x_1, y_1), \dots, (x_n, y_n); \quad (9)$$

где:

- $y_i$  принимает значение 1 или  $-1$ , в зависимости от того, к какому классу относится точка  $x_i$ .
- Каждый  $x_i$  это  $p$ -мерный вещественный вектор, нормализованный значениями  $[0,1]$  или  $[-1,1]$ .

Ненормализованная точка с наибольшим отклонением может существенно повлиять на работу классификатора, поэтому производится их нормирование. Это можно рассматривать как выборку с заранее заданным классом каждого элемента, к которому он относится. В результате классификатор должен научиться самостоятельно правильно распределять элементы по классам.

Для этого требуется построить разделяющую гиперплоскость, имеющую следующий вид:

$$w \times x - b = 0, \quad (10)$$

где вектор  $w$  – перпендикуляр к разделяющей плоскости,  $b$  – вспомогательный параметр. Параметр  $\frac{b}{\|w\|}$  равен по модулю расстоянию гиперплоскости от начала координат. В случае, когда  $b = 0$  – гиперплоскость проходит сквозь начало координат, что ограничивает решение.

В случае, когда данные являются линейно разделимыми требуется построить две гиперплоскости, которые разделят точки на два класса. Далее следует максимизировать расстояние между этими плоскостями и найти оптимальную разделяющую гиперплоскость, которая будет располагаться непосредственно на пол пути между ними. Разделяющие гиперплоскости могут быть описаны уравнениями (11-12):

$$w \times x - b = 1, \quad (11)$$

$$w \times x - b = -1. \quad (12)$$

Таким образом все элементы на этой границе или выше, принадлежат классу 1 (11), а все элементы на этой границе или ниже, принадлежат к классу 2 (12).

Геометрическое расстояние между этими плоскостями равно  $\frac{2}{\|w\|}$ , а значит чтобы максимизировать расстояние, нужно минимизировать  $\|w\|$ . Чтобы исключить попадание точек на поля, добавляется ограничение (13).

Для каждого  $i$

$$\begin{cases} w \times x_i - b \geq 1, & c_i = 1, \\ w \times x_i - b \leq -1, & c_i = -1. \end{cases} \quad (13)$$



Эти ограничения гласят, что каждая точка данных должна лежать на правильной стороне поля.

Это можно записать в виде (14):

$$c_i(w \times x_i - b) \geq 1, \quad 1 \leq i \leq n. \quad (14)$$

Если поставленные выше условия (14) выполняются – задача построения оптимальной разделяющей сводится к минимизации  $\|w\|$ , и имеет вид (15):

$$\begin{cases} \|w\|^2 \rightarrow \min, \\ c_i(w \times x_i - b) \geq 1, \quad 0 \leq i \leq n. \end{cases} \quad (15)$$

По теореме Куна – Таккера данная задача эквивалентна двойственной задаче поиска седловой точки функции Лагранжа (16)

$$\begin{cases} L(w, b; \lambda) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i (c_i((w \times x_i) - b) - 1) \rightarrow \min_{w,b} \max_{\lambda}, \\ \lambda_i \geq 0, \quad 1 \leq i \leq n, \end{cases} \quad (16)$$

где  $\lambda = (\lambda_1, \dots, \lambda_n)$  – вектор двойственных переменных.

Далее следует свести эту задачу к эквивалентной задаче квадратичного программирования, содержащую только двойственные переменные (17):

$$\begin{cases} -L(\lambda) = -\sum_{i=1}^n \lambda_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j c_i c_j (x_i \times x_j) \rightarrow \min_{\lambda}, \\ \lambda_i \geq 0, \quad 1 \leq i \leq n, \\ \sum_{i=1}^n \lambda_i c_i = 0. \end{cases} \quad (17)$$

Допустим, задача решена. В таком случае, можно найти  $w$  и  $b$  по следующим формулам (18,19):

$$w = \sum_{i=1}^n \lambda_i c_i x_i, \quad (18)$$

$$b = w \times x - c_i, \quad \lambda > 0. \quad (19)$$

В итоге алгоритм классификации может быть записан в виде (20):

$$a(x) = \text{sign}(\sum_{i=1}^n \lambda_i c_i x_i \times x - b). \quad (20)$$

При этом суммирование идёт не по всей выборке, а только по опорным векторам, для которых  $\lambda_i \neq 0$

В случае, когда данные линейно неразделимы, алгоритму разрешается допускать ошибки в процессе обучения. Введём набор дополнительных переменных  $\xi_i \geq 0$ , характеризующих величину ошибки на объектах  $x_i \geq 0, 1 \leq i \leq n$ . Возьмём за отправную точку (16), смягчим ограничения неравенства, а так же введём в минимизируемый функционал штраф за суммарную ошибку (21):

$$\begin{cases} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \rightarrow \min_{w,b,\xi_i}, \\ c_i (w \times x_i - b) \geq 1 - \xi_i, \quad 1 \leq i \leq n, \\ \xi_i \geq 0, \quad 1 \leq i \leq n. \end{cases} \quad (21)$$

Коэффициент  $C$  – параметр настройки метода, который позволяет регулировать отношение между максимизацией ширины разделяющей полосы и минимизацией суммарной ошибки.

Аналогично, по теореме Куна-Таккера сводим задачу к поиску седловой точки функции Лагранжа (22):

$$\left\{ \begin{array}{l} L(w, b, \xi; \lambda, \eta) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \lambda_i (c_i((w \times x_i) - b) - 1) - \\ \quad - \sum_{i=1}^n \xi_i (\lambda_i + \eta_i - C) \rightarrow \min_{w, b, \xi} \max_{\lambda, \eta} \\ \quad \xi_i \geq 0, \lambda_i \geq 0, \eta_i \geq 0, \quad 1 \leq i \leq n, \\ \quad \left\{ \begin{array}{l} \lambda_i = 0, \\ c_i(w \times x_i - b) = 1 - \xi_i, \end{array} \right. \quad 1 \leq i \leq n, \\ \quad \left\{ \begin{array}{l} \eta_i = 0, \\ \xi_i = 0, \end{array} \right. \quad 1 \leq i \leq n. \end{array} \right. \quad (22)$$

По аналогии сведём эту задачу к эквивалентной (23):

$$\left\{ \begin{array}{l} -L(\lambda) = -\sum_{i=1}^n \lambda_i + \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j c_i c_j (x_i \times x_j) \rightarrow \min_{\lambda}, \\ \quad 0 \leq \lambda_i \leq C, \quad 1 \leq i \leq n, \\ \quad \sum_{i=1}^n \lambda_i c_i = 0. \end{array} \right. \quad (23)$$

При использовании метода опорных векторов решают именно эту задачу, а не (17), так как гарантировать линейную разделимость точек на два класса в общем случае не представляется возможным. Этот вариант алгоритма называют алгоритмом с мягким зазором (soft-margin SVM), тогда как в линейно разделимом случае говорят о жёстком зазоре (hard-margin SVM).

Для алгоритма классификации сохраняется формула (20), с той лишь разницей, что теперь ненулевыми  $\lambda_i$  обладают не только опорные объекты, но и объекты-нарушители, которые являются шумовыми выбросами. В таком случае появится недостаток в виде построенного на них решающего правила.

Константу  $C$  обычно выбирают по критерию скользящего контроля. Это трудоёмкий способ, так как задачу приходится решать заново при каждом значении  $C$ .

Существуют случаи, когда только объекты-выбросы мешают правильной классификации линейной выборки. Чтобы решить подобную проблему применяется фильтрация выбросов. Сначала задача решается при некотором  $C$ , и из выборки удаляется небольшая доля объектов, имеющих

наибольшую величину ошибки  $\xi_i$ . После этого задача решается заново по усечённой выборке. Возможно, придётся проделать несколько таких итераций, пока оставшиеся объекты не окажутся линейно разделимыми.

### **2.3 Детектирование алгоритмом гистограммы направленных градиентов и классификация методом опорных векторов**

Обнаружение лиц на изображении методом гистограммы направленных градиентов (HOG) с помощью машин опорных векторов (SVM) основаны на принципе скользящего окна, который можно полностью описать следующим образом. Сначала рассчитывается гистограмма направленного градиента для всех изображений объектов, подлежащих обнаружению в обучающей выборке. Затем обучается классификатор SVM на основе этих данных. В процессе обнаружения используется набор окон, размер которых фиксирован. Каждое такое окно обходит изображение и вычисляет HOG для заданной области, а обученный SVM-классификатор решает, относить ли найденный объект к искомому. В конце этого процесса получается набор окон с возможными объектами, исключаются лишние окна путем слияния окон с наибольшими пересечениями и удаления остальных [23].

### **2.4 Ансамбль решающих деревьев**

«Дерево решений – интуитивно понятная базовая единица алгоритма случайный лес. Его можно рассматривать как серию вопросов да/нет о входных данных. В конечном итоге вопросы приводят к предсказанию определённого класса (или величины в случае регрессии). Это интерпретируемая модель, так как решения принимаются так же, как и человеком: мы задаём вопросы о доступных данных до тех пор, пока не приходим к определённому решению (в идеальном мире)» [17].

Вычисление ключевых точек лица происходит при помощи алгоритма, основанного на ансамбле деревьев регрессии. Для его обучения необходимы следующие данные:

- обучающий набор помеченных вручную ключевых точек на изображении, т.е. конкретный набор (x,y)-координаты областей, окружающих каждую часть лица (рот, глаз, бровь и т.д.);
- заданные вероятности расстояния между парами входных пикселей.

Эти данные являются частью 68-точечного набора данных iBUG 300-W, на котором происходило обучение модуля face\_recognition. Существует множество разновидностей детекторов ключевых точек лиц, к примеру модель из 194 точек, обученная на наборе тренировочных данных HELEN. Однако 68 ключевых характеристик будет вполне достаточно для построения точек лиц с высокой точностью.

С помощью перечисленных обучающих данных ансамбль деревьев регрессии обучается находить расположение ключевых точек, которые строятся по интенсивностям пикселей.

Реализация выбранного метода детектирования характеристик рассчитана на поиск 68 ключевых точек, представленных на рисунке 5.

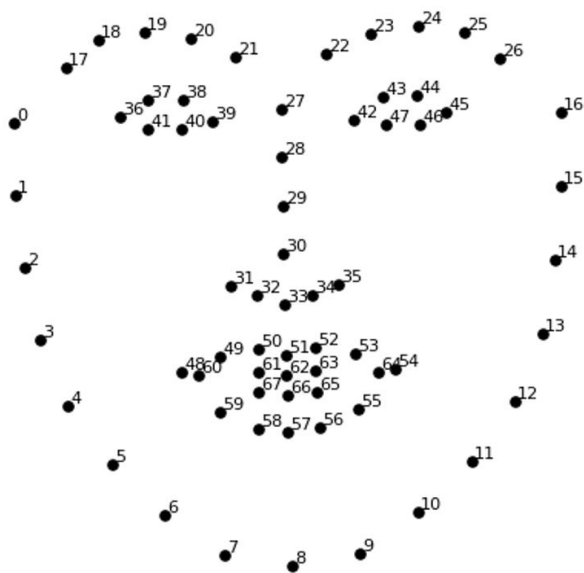


Рисунок 5 – Шаблон 68 ключевых точек лица

Благодаря построенным на изображении точкам лицо можно трансформировать для улучшения точности распознавания. Для этого будут использоваться основные преобразования изображения, которые сохраняют параллельные линии (аффинные преобразования). С их помощью изображение трансформируется таким образом, чтобы глаза, рот и нос располагались максимально по центру. Результат преобразований показан на рисунке 6.



Рисунок 6 – Преобразование лица

Как показано на рисунке 6 алгоритм получает на вход изображение, на котором строит ключевые точки. Затем программа пытается трансформировать изображение так, чтобы оно стало максимально схожим с шаблоном, при этом не изменив ключевые характеристики лица.

## **2.5 Сверточная нейронная сеть (CNN)**

В сверточной нейронной сети реализовано несколько слоев, которые преобразуют входные данные в вектор принадлежности к классам. Основные слои, которые используются в сверточной сети – это слои свертки, слой линейной ректификации, слой субдискретизации и полносвязный слой. У каждого из них своя функция и задача, но в итоге получается сложно структурированная сеть с наилучшими результатами классификации объектов на изображении [2]. Подробнее о каждом слое.

Сверточный слой является главным слоем сверточной нейронной сети. «Его основное назначение – выделить признаки на входном изображении и сформировать карту признаков. Карта признаков – это всего лишь очередной тензор (массив матриц), в котором каждый канал отвечает за какой-нибудь выделенный признак.»[10]

Чтобы слой мог выделять признаки используются фильтры, которые также являются набором тензоров. Эти тензоры имеют одинаковый размер, а их количество определяет глубину выходного 3D массива. При этом, глубина самих тензоров совпадает с количеством каналов входного изображения.

Весовые коэффициенты ядра свёртки (небольшой матрицы) неизвестны и устанавливаются в процессе обучения. Благодаря тому, что фильтр, который проходится по изображению, не изменяется во время самого прохождения, получается, что число обучаемых параметров во много раз меньше, чем у полносвязной сети.

Свёртка – процесс вычисления нового значения пикселя с учетом его значений его соседей. Ниже представлен алгоритм.

«Фильтр накладывается на левую верхнюю часть изображения и производится покомпонентное умножение значений фильтра и значений изображения, после чего фильтр перемещается дальше по изображению до тех пор, пока аналогичным образом не будут обработаны все его участки.

Затем числа полученных матриц суммируются в единую матрицу – результат применения фильтра. После этого к каждому значению матрицы добавляется одинаковое число – значение смещения данного фильтра. Полученная матрица составляет один канал выходной карты признаков.

После того, как будут получены каналы для каждого из фильтров, матрицы объединяются в единый тензор, благодаря чему на выходе снова получается изображение, с другим числом каналов и, возможно, другим размером.» [10]

Слой активации представляет из себя некоторую функцию, которая применяется к каждому числу входного изображения. Традиционно были использованы функции Sigmoid и Tanh (рисунок 7).

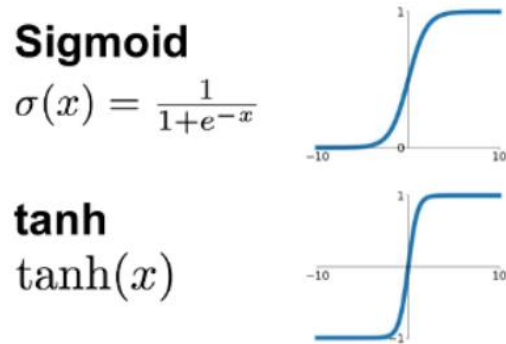


Рисунок 7 – Функции активации

Однако в 2000х годах была предложена и исследована новая функция активации – ReLU, которая позволила существенно ускорить процесс обучения (рисунок 8).



**ReLU**  
 $\max(0, x)$

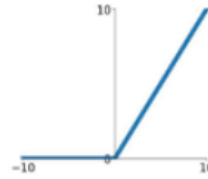


Рисунок 8 – Функция активации ReLU

Следующий слой – слой подвыборки (пулинга). С помощью него происходит уменьшение признакового пространства, при этом сохранив важную информацию. Существует несколько версий слоя подвыборки, однако наиболее часто использующийся – макспулинг. На рисунке 9 представлено несколько популярных видов слоев пулинга.

### Карта признаков



Рисунок 9 – Виды пулинга

Слой подвыборки имеет только один гиперпараметр – шаг пулинга. Этот шаг – число раз, в которое следует сократить пространственные размерности. Популярный слой макспулинга уменьшает входной тензор в два раза.

После нескольких проходов свертки изображений и сжатия с помощью пулинга система перестраивается от конкретной пиксельной сетки высокого разрешения к более абстрактным картам признаков, как правило, в каждом последующем слое количество каналов увеличивается, а размерность изображения в каждом канале уменьшается. В итоге остается большой набор каналов, хранящих небольшое количество данных (даже один параметр), которые интерпретируются как самые абстрактные понятия, выявленные из исходного изображения.

Эти данные объединяются и передаются в обычную полносвязную нейронную сеть, которая также может состоять из нескольких слоев. В то же время полносвязные слои уже теряют пиксельную пространственную структуру и имеют относительно небольшую размерность (относительно количества пикселей в исходном изображении).

«Как и полносвязная нейронная сеть, свёрточная сеть обучается с помощью алгоритма обратного распространения ошибки. Сначала выполняется прямое распространение от первого слоя к последнему, после чего вычисляется ошибка на выходном слое и распространяется обратно. При этом на каждом слое вычисляются градиенты обучаемых параметров, которые в конце обратного распространения используются для обновления весов с помощью градиентного спуска.»[10]

## **2.6 Программная разработка модулей системы**

Для разработки интеллектуальной системы распознавания лиц было принято решение использовать язык программирования Python [8]. Причиной выбора данного языка стало наличие большого количества библиотек и модулей, простой и понятный синтаксис, а также легкая читаемость кода.

В качестве основной библиотеки распознавания лиц выступает модуль `face_recognition` [21], который разработал [Adam Geitgey] на основе

библиотеки `dlib`. Причиной выбора является его несложная реализация, использование натренированной нейронной сети глубокого обучения, благодаря которой производится вычисление ключевых точек, а также использование линейного метода опорных векторов и гистограммы направленных градиентов для обнаружения лиц. Кроме модуля `face_recognition` будут использоваться следующие библиотеки:

Модуль `Pickle` – мощный алгоритм сериализации и десериализации объектов. С помощью его функций будет создана база данных с именами и «уникальными признаками лиц» людей.

Модуль `Os`, который, в основном, будет использоваться для извлечения названий изображений.

Библиотека `OpenCV` [22]. Данная библиотека содержит в себе множество алгоритмов компьютерного зрения и обработки изображений. Именно благодаря ей будет производиться получение видеопотока с камеры, трансформация изображения, а также выделение лиц во время процесса распознавания.

Библиотека `numpy`. Это библиотека языка `Python` добавляет поддержку многомерных массивов и матриц, а также позволяет производить вычисления над ними, благодаря высокоуровневым математическим функциям.

Запуск программы будет осуществляться в интегрированной среде разработки `PyCharm`, представляющую собой интеллектуальный редактор кода. Данная среда разработки предоставляет функцию навигации кода, его автодополнения и инспекции в режиме реального времени.

### **2.6.1 Реализация модуля сохранения кодировок в файл (`data.py`).**

В начале программы требуется подключить используемые в ней библиотеки:

- `face_recognition` для загрузки изображений и поиска лица и его ключевых признаков;

- pickle для создания базы данных;
- os для отделения названия изображения от его формата.

Далее создается три переменных:

- строковая переменная path, хранящая в себе название каталога, в котором будут храниться фотографии людей;
- переменная myList, в которую при помощи функции os.listdir(path) мы возвращаем список, содержащий имена файлов в каталоге, заданном путем path;
- переменная encodeList, которая представляет собой словарь данных и состоит из групп значений и пар ключей. В качестве ключей будут храниться имена людей, а кодировки их лиц будут переданы в группу значений.

После объявления переменных требуется поочередно загрузить все фотографии из каталога, получить и сохранить их названия и кодировки лиц в словарь данных. Для выполнения этой цели был создан цикл for cl in myList. Внутри цикла хранится переменная curImg, в которую при помощи функции load\_image\_file(f' {path}/{cl} ') библиотеки face\_recognition загружается текущее изображение.

Следующим шагом мы передаем имя файла и кодировку лица в словарь данных. Для выполнения данной операции мы присваиваем переменной encodeList результат работы функции «face\_encodings(curImg)[0]». Данная функция принимает изображение и возвращает 128-мерный массив, хранящий кодировки лица. Словарь хранит ключ и группу значений, поэтому массив кодировок лица будет храниться в паре с названием изображения, с которого были получены эти кодировки. Название каждого изображения должно соответствовать имени человека, лицо которого находится в этом изображении. Чтобы имена не имели в себе формата изображения (.jpg, .png и т.д.) была использована функция splitext модуля

os.path. В качестве входного параметра в нее подается название изображения (cl), а возвращаемое значение – двойной кортеж (root, ext). Если cl начинается с точки и содержит не более одной точки, возвращаемое значение ext будет пустым. В случае изображения, название которого не содержит точек, функция вернет его название, отбросив расширение.

По окончании работы цикла заполняется словарь encodeList, который при помощи библиотеки pickle будет сохранен в виде файла «dataset\_faces.dat». Для этого используется функция open с атрибутами «wb» (w - запись, b – бинарный режим). Она открывает, либо создает файл, если он не существует, «dataset\_faces.dat», в который будет сохранен словарь encodeList при помощи функции dump библиотеки pickle. dump принимает в себя переменную encodeList и файл «dataset\_faces.dat», после чего записывает сериализованный объект в файл.

### **2.6.2 Реализация модуля распознавания лиц (recognition.py)**

В начале программного кода подключаются все требуемые библиотеки:

- face\_recognition для поиска лиц на изображении и извлечения его ключевых признаков;
- pickle для загрузки базы данных в программный код из файла;
- numpy для поиска максимального сходства между лицом из базы данных с лицом из видеопотока;
- cv2 для захвата видеопотока, трансформации изображений и выделения лица человека в кадре.

После объявления библиотек объявляются следующие переменные:

- METHOD – строковая переменная, хранящая в себе названия алгоритма, который будет использоваться для обнаружения лиц на изображении;

- `font` – переменная хранящая в себе шрифт, которым будет написано имя человека в процессе распознавания;
- `frame_skip`, хранящая в себе «*n*» количество пропускаемых кадров. Так как процесс распознавания лиц требует высокой вычислительной мощности было принято решение пропускать несколько кадров и распознавать лица каждый «*n*» кадр;
- в `imgresize` хранится число от 0 до 1.0. Переменная отвечает за то, насколько будет сжато изображение из видеопотока;
- `scale` используется для вычисления коэффициента масштабирования рамки лица, так как поиск лица идет на сжатом изображении, а выделение на оригинальном;
- в FPS следует записать количество кадров в секунду, которое передает видеокамера;
- в переменную `framesPerSecond` будет присвоено вычисленное значение количества кадров видеопотока в секунду, выдаваемых в результате работы программы;
- `frame_counter` хранящая в себе 0. Эта переменная так же участвует в вычислении количества кадров в секунду.

После объявления переменных требуется загрузить базу данных в программный код. Для этого используется функция `open` с атрибутами «`rb`» (`r` – чтение, `b` – бинарный режим) в которую передается название файла базы данных «`dataset_faces.dat`». Далее переменной `all_face_encodings` присваивается результат работы функции `load` библиотеки `pickle`. Эта функция читает упакованное представление объекта из открытого файла и возвращает указанную в нем восстановленную иерархию объектов.

Далее, когда база данных загружена в программный код, требуется объявить переменные `classNames` и `encodeListKnown`. В первую переменную передается список имен людей из базы данных при помощи функции `keys()`.

Во вторую возвращается результат работы функции `values()`, которым является двумерный массив кодировок лиц.

Следующим шагом требуется указать программе с какой камеры считывать видеопоток. Для этого в переменной `cap` создается объект `VideoCapture(j)` библиотеки `openCV`, где `j` – номер подключенной камеры. Так же, при помощи функции `set` настраиваются параметры камеры, а именно: количество пикселей в ширину и высоту.

Теперь все подготовительные действия выполнены. Следующей строкой объявляется цикл, который будет распознавать лица на протяжении всей трансляции видеопотока и может быть остановлен кнопкой `esc`.

В начале цикла переменная `frame_counter` проверяется на равенство 0. Если оператор `if` возвращает `true`, создается переменная `time_to_skipping` (при последующих итерациях цикла она обновляется), в которую будет записываться количество «тиков» с момента ее объявления, иначе оператор не выполняет каких-либо операций.

Чтобы покадрово передавать изображение с камер видеонаблюдения будет использоваться функция `read()`. Первое возвращаемое значение типа `Boolean` (`True` либо `false`) в зависимости от того правильно ли считан кадр, будет записываться в переменную `success`. Второе значение возвращает само изображение, которое записывается в переменную `img`.

Следующим шагом изображение трансформируется с помощью алгоритма `resize` библиотеки `opencv`. `Resize` имеет следующие входные данные: `src` – изображение для трансформации; `dst` – выходное изображение, имеющее размеры следующего параметра (`dsize`), когда он не равен нулю; `dsize` – размер выходного изображения (вводится в качестве кортежа); `fx` – коэффициент масштабирования по горизонтальной оси; `fy` – коэффициент масштабирования по вертикальной оси; `interpolation` – метод интерполяции. В данном случае в алгоритм передается: изображение (`img`); Манипулировать размером изображения будут коэффициенты масштабирования, так что значения передаваемого кортежа равны (0,0); В `dst` передается «None», так

как `dsize` равен нулю использовать этот параметр не является возможным; `fx` и `fy` принимают значение переменной «`imgresize`». Таким образом входное изображение было сжато на 50% по горизонтали и вертикали. Результат работы функции записывается в переменную `imgS`.

Порядок цифровых каналов в библиотеке `openCV` – `BGR`, однако алгоритмы распознавания лиц работают в `RGB`, поэтому следующим этапом требуется его изменить. Простейшее решение – использовать функцию `cvtColor`, в которую передается уменьшенное изображение `imgS` и параметр трансформации `cv2.COLOR_BGR2RGB`. Возвращаемое значение присваивается в `imgS`.

Следующим элементом стоит условный оператор `if`, который проверяет нужно ли производить распознавание лиц на этом кадре или нет. Когда его возвращаемое значение истинно, программа начинает поиск лиц и их ключевых признаков на текущем изображении. Для этого используются функции из библиотеки `face_recognition`.

В функцию `face_locations` передаются следующие входные данные: Первый аргумент – обработанное изображение `imgS`. Вторым аргументом передается строковая переменная `METHOD`, в которой хранится название метода обнаружения лиц «`hog`». Результат работы алгоритма записывается в переменную `FaceFrame`.

Другим алгоритмом является `face_encodings`, в который передается изображение `imgS`, а также недавно полученные координаты местоположения лица `FaceFrame`. Вычисленные ключевые признаки лица записываются в переменную `encodeFrame`.

Наступил этап сравнения лиц. В следующем цикле программа сравнивает найденные характеристики лица с известными ей и решает существует ли данное лицо в базе данных или нет.

Переменные `encodeFace` и `faceLoc` получают значения `encodeFrame` и `FaceFrame` соответственно, после для каждого `encodeFace` и `faceLoc` происходят следующие события:



Сначала проверяются базы данных на наличие найденных на изображении лиц. На вход в обе функции подается двумерный массив известных лиц `encodeListKnown` и недавно полученная кодировка лиц из видеопотока `encodeFace`. Функция `compare_faces` вернет список Boolean значений (True/False), который будет записан в переменную `matches`. Она сравнивает кодировки лиц из базы данных с лицом на изображении и решает, существует оно в базе данных или нет. Допустимое отклонение (`tolerance`) равно 0.6, то есть если ключевые характеристики лица более чем на 60% схожи с известными – функция возвращает True. Однако в таких условиях существует вероятность найти одновременно два или более лица, подходящие по критерию сравнения. Чтобы этого не допустить был добавлен алгоритм `face_distance`. Он производит вычисления и возвращает в переменную `faceDis` евклидово расстояние между сравниваемым и каждым известным системе лицом. В таком случае чем меньше расстояние – тем больше сходство с лицом. Чтобы найти минимальное расстояние между характеристиками используется функция `argmin`, в которую передается массив `faceDis`. На выходе получается индекс минимального элемента, который сохраняется в переменную `matchIndex`.

На следующем этапе требуется передать найденные координаты лиц `faceLoc` в координаты `y1, x2, y2, x1`. Так как координаты были получены на сжатом изображении требуется масштабировать каждую координату на заранее вычисленный коэффициент масштабирования `scale`. Далее с помощью условного оператора `if` проверяется существует ли в базе данных лицо с данным индексом. Если лицо существует:

- объявляется переменная `name`, в которую присваивается имя из массива `classNames` с индексом `matchIndex`;
- далее для построения квадрата, обводящего лицо на изображении, используется функция `cv2.rectangle`, в которую передаются следующие аргументы: изображение, на котором требуется построить

прямоугольник; координаты для построения; цвет прямоугольника в формате BGR; толщина линий;

- следующей строкой строится прямоугольник с помощью той же функции `cv2.rectangle` с такими координатами, чтобы он располагался непосредственно под квадратом, который обводит лицо. Добавлен аргумент `cv2.FILLED`, который заполняет область прямоугольника сплошным цветом.

В итоге, если данные `matches` удовлетворяют ограничению оператора `if` программа построит зеленый квадрат вокруг лица и прямоугольник того же цвета непосредственно под ним.

В случае, когда лица с таким индексом не существует:

- переменной `name` присваивается значение «Unknown»;
- первая и вторая функция `cv2.rectangle` передаются те же входные параметры за исключением элемента `color`.

Для выделения неизвестного лица будет использоваться красный цвет прямоугольников.

Теперь, когда лицо выделено, а в переменной `name` хранится информация о имени, требуется вывести текст и расположить его в области прямоугольника, залитого цветом. Вывод текста происходит с помощью функции `cv2.putText` со следующими аргументами: изображение на которое будет выведен текст; строка с именем человека; координаты на которых должен располагаться текст; шрифт текста; коэффициент, на который умножается базовый размер шрифта; цвет шрифта в формате BGR; толщина шрифта.

Следующей строкой в переменную `frame_counter` прибавляется 1. Это нужно для того, чтобы проверить требуется ли распознать текущий кадр или нет, а также подсчитать количество циклов программы. Однако в случае длительной работы программы значения `frame_counter` могут превышать

десятки тысяч, что повлияет на ее эффективность. Чтобы избежать этой проблемы существует следующий условный оператор. Если `frame_counter` равен 30, тогда выполняются следующие действия:

- Вычислить секунд, потраченных на обработку 30 циклов. Для этого требуется:
  1. Получить значения функции `cv2.getTickCount`, которая вернет текущий тик работы программы.
  2. Вычесть из текущего тика переменную `time_to_skipping`, в которой хранится тик, записанный 30 кадров назад.
  3. Разделить полученную разность на частоту тиков, вычисленную с помощью функции `cv2.getTickFrequency`
- Вычислить и записать в `framesPerSecond` количество кадров в секунду видеопотока, которое смогла обработать программа. Это рассчитывается делением числа кадров в секунду выдаваемого камерой (FPS) на количество секунд, потраченных на обработку того же числа циклов программы (`time_to_skipping`).
- Обнулить `frame_counter` для начала нового отсчета.

Вывод количества кадров в секунду происходит с помощью функции `cv2.putText` с аргументами: изображение, на которое будет выведен текст; строка с количеством кадров в секунду; координаты на которых должен располагаться текст; шрифт текста; коэффициент, на который умножается базовый размер шрифта; цвет шрифта в формате BGR; толщина шрифта.

Функция `cv2.imshow` выводит окно с изображением и имеет два аргумента – название окна и изображение, которое следует вывести.

Чтобы программа не работала бесконечно следует добавить условие при котором она будет остановлена. С этой целью объявляется переменная `k` в которой хранится функция `cv2.waitKey`, которая ожидает нажатия какой-либо клавиши и возвращает ее значение. Для завершения работы программы была выбрана клавиша «Esc». Оператор `if` проверяет, является ли нажатая

клавиша `искемой`. Если условие выполняется – цикл программы останавливается оператором `break`.

Как только бесконечный цикл остановлен следует освободить камеру функцией `release`, а также закрыть окно отображения видеопотока с помощью `cv2.destroyAllWindows`.

На этом завершается работа модуля распознавания лиц.

Вывод по второму разделу.

Во втором разделе производится выбор языка алгоритмов обнаружения и распознавания лиц, метода построения ключевых точек, и классификатора. Так же, был выбран язык программирования и среда разработки, описаны используемые библиотеки и приведена программная реализация двух модулей системы распознавания лиц:

- первый – модуль сохранения известных кодировок лиц в файл. Данный модуль сокращает время запуска системы распознавания, так как именно в нем будет происходить вычисление кодировок лиц из базы данных;
- второй – модуль распознавания лиц, с помощью которого происходит считывание информации с видеопотока, трансформация изображения, а также обнаружение и распознавание лиц в кадре.

### 3 Тестирование и анализ результатов

Результатом тестирования системы должно стать точное соответствие поставленной задаче. Проекту необходимо правильно создавать базу данных кодировок лиц, распознавать известные лица, а также обнаруживать неизвестные. Опробуем весь реализованный функционал разработанных модулей.

Сначала требуется создать базу данных кодировок лиц. Для этого в папку с названием «Resource» требуется поместить фотографии лиц людей, которых система должна будет распознавать. На рисунке 10 показана папка «Resource» и ее содержимое.

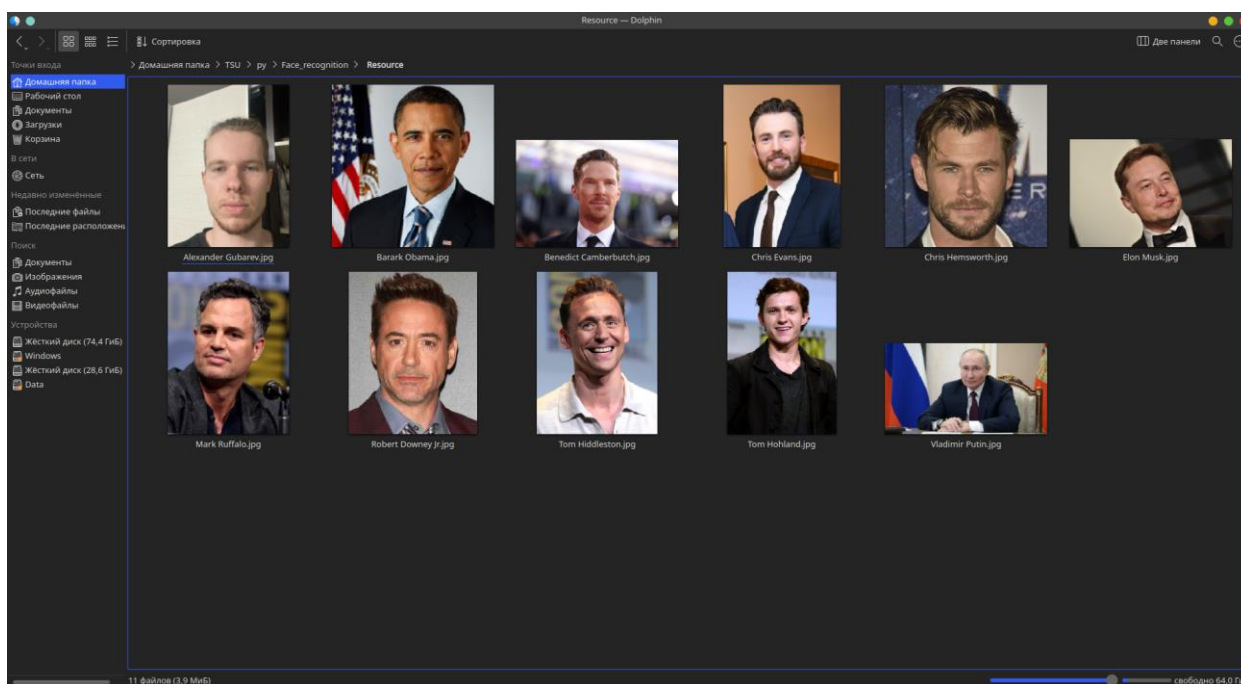


Рисунок 10 – Содержимое каталога "Resource"

Следующим шагом следует запустить программу data.py. С использованием программы PyCharm выполняется запуск программы. В результате был создан файл «dataset\_faces.dat», в котором будут храниться кодировки лиц людей из каталога «Resource». В результате работы

программы, в консоли должна появиться надпись об успешном создании файла, а процесс завершиться с кодом 0. Запуск программы data.py с помощью среды разработки PyCharm показан на рисунке (11)

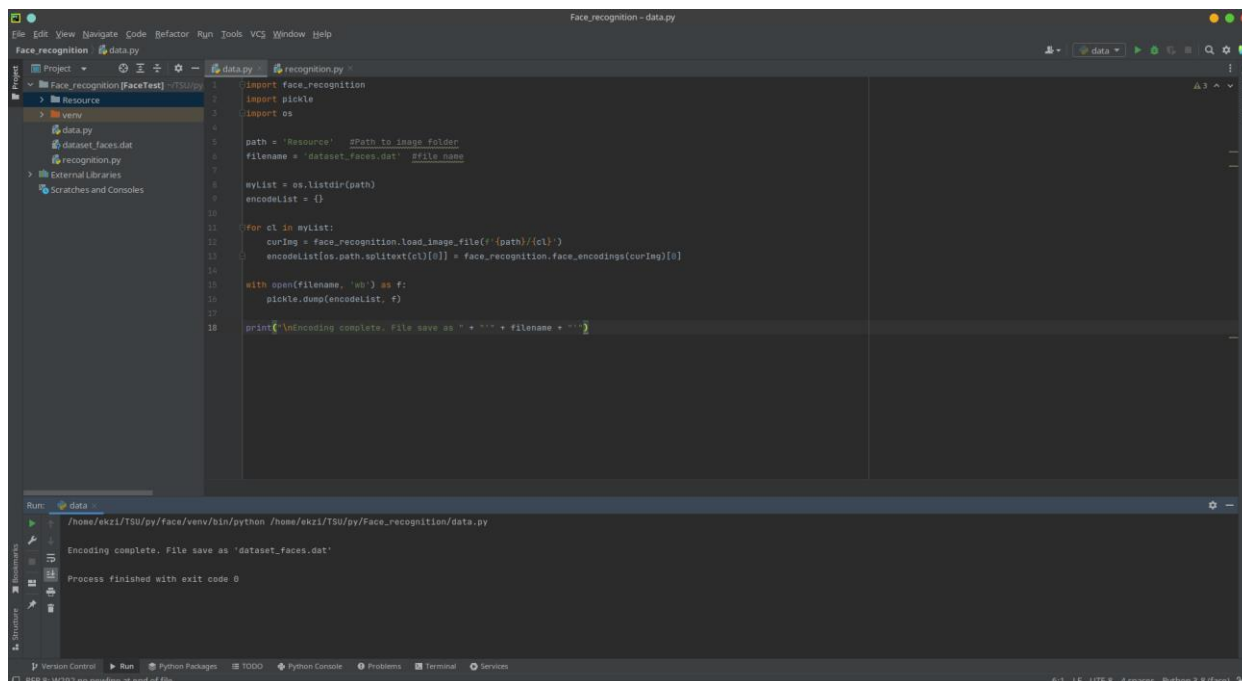


Рисунок 11 – Результат работы программы

Надпись об успешном создании файла появилась, а программа завершила свою работу с кодом 0. Теперь можно удостовериться в существовании файла «dataset\_faces.dat», открыв каталог.

Теперь, когда кодировки сохранены в отдельный файл, можно приступить к распознаванию лиц. Для этого будет использоваться программа recognition.py. С помощью среды разработки PyCharm выполняется запуск программы recognition.py. Результат запуска программы показан на рисунке 12.

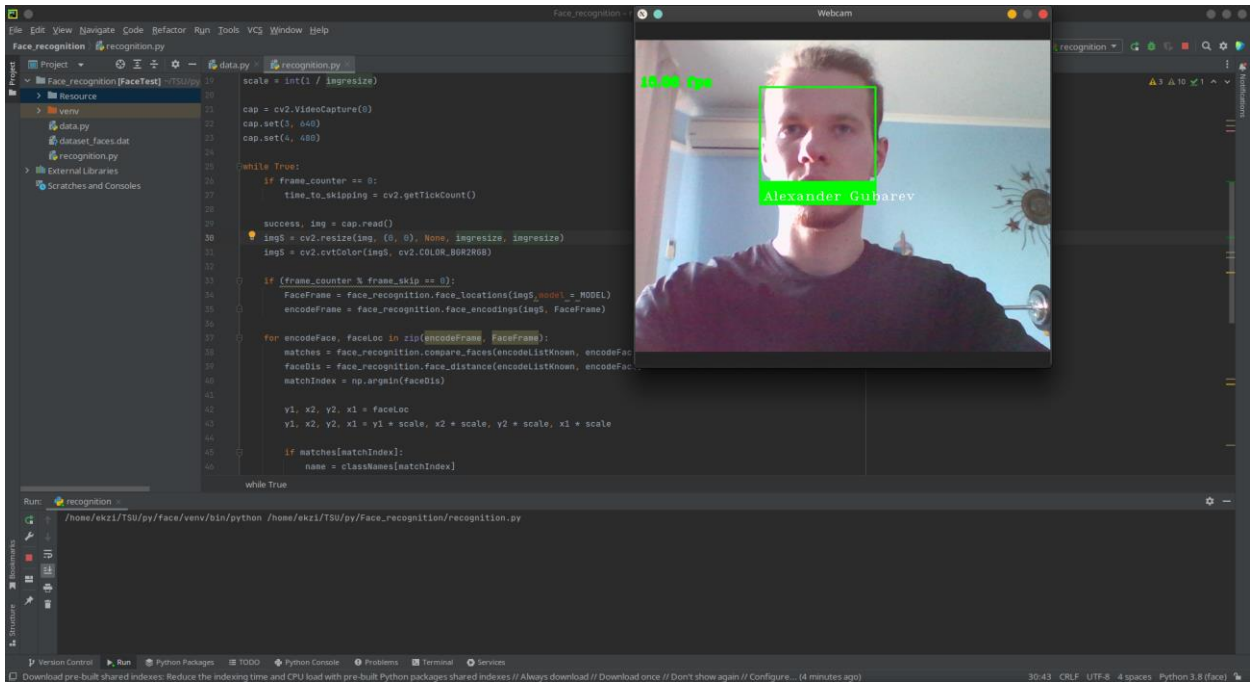


Рисунок 12 – Результат запуска программы "recognition"

Как можно заметить на рисунке захват видеопотока с камеры прошел успешно, изображение было обработано, а лицо найдено и распознано. Также, видно, как счетчик кадров в секунду вычисляет и показывает их количество.

Следующим этапом в тестировании системы является проверка поиска неизвестных лиц, а также возможность одновременного обнаружения нескольких лиц. Попросим человека, лицо которого отсутствует в базе данных, посмотреть в камеру, при этом известный системе человек должен оставаться в кадре. Результат данной проверки показан на рисунке 13.

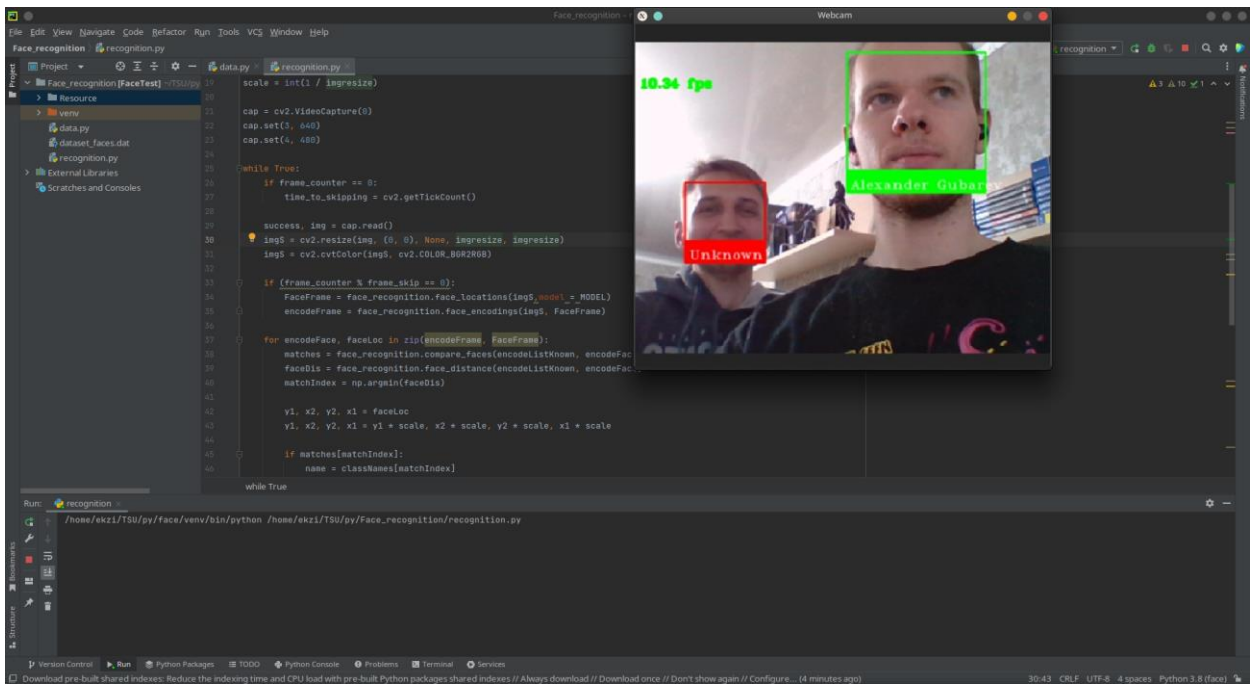


Рисунок 13 – Результат проверки системы распознавать несколько лиц

После проверки становится понятно, что программа успешно проводит поиск нескольких лиц, а также обнаруживает и выделяет неизвестные ей лица на изображении.

Вывод по третьему разделу.

В третьем разделе проводится тестирование и анализ результатов.

Из результатов тестирования можно сделать вывод: программа успешно прошла тестирование, справилась с поставленными перед ней задачами. Это означает, что разработанная система верно выполняет свои функции и готова к использованию для распознавания лиц.



## Заключение

В данной пояснительной записке к выпускной квалификационной работе были рассмотрены основные аспекты разработки интеллектуальной системы распознавания лиц. Реализация производилась на тестовой упрощенной базе данных, состоящей из 20 изображений лиц разных людей.

В ходе исследования предметной области были указаны материалы, позволяющие ознакомиться с теоретическими данными и реализацией алгоритмов обнаружения лиц с точки зрения математического моделирования, а также приложена блок-схема, способствующая пониманию того, как работает программная часть модуля.

При описании программной реализации были рассмотрены и применены выбранные алгоритмы обнаружения и распознавания лиц, реализация модулей программы на скриптовом языке Python, позволяющих создавать файл, хранящий в себе кодировки и имена лиц, и распознавать лица с видеопотока в режиме реального времени.

Во время описания тестирования модулей и анализа результатов приводились примеры входных данных и их взаимодействия, чтобы показать правильность работы всех алгоритмов. В ходе тестирования не было встречено ошибок, а результаты, полученные при взаимодействии с программой, оказались верными. Из всей информации, полученной ранее следует, что программный продукт работает правильно.

Подытожив все указанные данные, следует отметить, что для разработки данного программного продукта и модулей, для решения схожих задач, требуется знание и понимание алгоритмов обнаружения и распознавания лиц, а также языка программирования Python. В результате выполнения выпускной квалификационной работы были разработаны модули системы распознавания лиц.

## Список используемой литературы и используемых источников

1. Алгоритм извлечения изображения изображений LBP. 2018. URL: <https://russianblogs.com/article/12861795224/>
2. Гузий Е.А., Федоренков В.В. Сверточная нейронная сеть для разработки системы распознавания и классификации изображений. 2017. 9 с. URL: <https://docplayer.com/58000991-Svertochnaya-neyronnaya-set-dlya-razrabotki-sistemy-raspoznvaniya-i-klassifikacii-izobrazheniy.html> (дата обращения 11 апреля 2022)
3. Истишев В. Как работает распознавание лиц? Разбор. 2021 URL: <https://habr.com/ru/company/droider/blog/568764/>
4. Клоков А. SVM. Объяснение с нуля и реализация на python. Подробный разбор метода опорных векторов. 2020. URL: <https://habr.com/ru/company/ods/blog/484148/>
5. Корешкова Т. Компьютерное зрение: технологии, компании, тренды. 2021. URL: <https://rdc.grfc.ru/2021/04/analytics-computer-vision/>
6. Костерин В.В. Урок 2. Гистограмма направленных градиентов с использованием openCV. 2021. URL: <https://waksoft.susu.ru/2021/11/01/histogram-of-oriented-gradients/>
7. Левчук С.А., Якименко А.А. Сборник научных трудов НГТУ. 2018. с. 40–58. URL: [https://journals.nstu.ru/digital-tech-security/catalogue/contents/view\\_article?id=19341](https://journals.nstu.ru/digital-tech-security/catalogue/contents/view_article?id=19341)
8. Мусин Д. Самоучитель Python. Выпуск 0.2. 2017. URL: <https://pythonworld.ru/uploads/pythonworldru.pdf> (дата обращения 12 апреля 2022)
9. Паршин С.Е. Исследование параметров алгоритмов распознавания лиц. 2019. 16 с. URL: [https://journals.nstu.ru/digital-tech-security/catalogue/contents/view\\_article?id=20363](https://journals.nstu.ru/digital-tech-security/catalogue/contents/view_article?id=20363)

10. Перминов А. Свёрточная нейронная сеть с нуля. Часть 0. Введение. 2019 URL: <https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-zero-introduction>
11. Сармьенто К., Саваж Х. (2020). Сравнение двух методов классификации объектов с использованием скрытых марковских моделей и сверточных нейронных сетей. 2020. 33 с. URL: <http://ia.spcras.ru/index.php/sp/article/view/14007/14825>
12. Современные тенденции технических наук. 2018. 38 с. URL: <https://moluch.ru/conf/tech/archive/346/>
13. Солдатов А.Н., Миньков С.Л. Сборник материалов инноватика-2017. с. 392-395 URL: [https://storage.tusur.ru/files/67171/Innovatika\\_2017.pdf](https://storage.tusur.ru/files/67171/Innovatika_2017.pdf)
14. Титов А. Технология распознавания лиц от А до Я. 2017. URL: <https://securityrussia.com/blog/face-recognition.html>
15. Тымчук А. И. Метод Виолы-Джонса для распознавания объектов на изображении. URL: <http://www.nauteh-journal.ru/files/7a7d96c9-acd8-4936-819f-5607386faf51>
16. Фильтр в обработке изображений (ядро свертки). URL: <https://russianblogs.com/article/77801672445/>
17. Штукатуров С. Реализация и разбор алгоритма «случайный лес» на Python. 14 июня 2019. URL: <https://tproger.ru/translations/python-random-forest-implementation/>
18. Acar N. Eigenfaces: Recovering Humans from Ghosts. 2018. URL: <https://towardsdatascience.com/eigenfaces-recovering-humans-from-ghosts-17606c328184>
19. Dwivedi D. Face detection for begginers. 2018. URL: <https://towardsdatascience.com/face-detection-for-beginners-e58e8f21aad9>
20. Dwivedi D. Face recognition for beginers. 2018. URL: <https://towardsdatascience.com/face-recognition-for-beginners-a7a9bd5eb5c2>
21. Geitgey A. Face recognition. 2017 URL: <https://face-recognition.readthedocs.io/en/latest/readme.html>

22. Learn OpenCV in 3 Hours. URL: <https://www.computervision.zone/courses/learn-opencv-in-3-hours/>
23. Mithi. Vegicle detection with HOG and linear SVM. 2017. URL: <https://medium.com/@mithi/vehicles-tracking-with-hog-and-linear-svm-c9f27eaf521a>
24. Rovai M. Real-Time Face Recognition: An End-To-End Project. 2018. URL: <https://towardsdatascience.com/real-time-face-recognition-an-end-to-end-project-b738bb0f7348>
25. Rosebrock A. OpenCV EigenFaces for face recognition. 2021. URL: <https://pyimagesearch.com/2021/05/10/opencv-eigenfaces-for-face-recognition/>
26. Sanjeevkumar A., Suvarna N. Human Identification using Histogram of Oriented Gradients (HOG) and Non-Maximum Suppression (NMS) for ATM Video Surveillance. 2021. 10 c.