|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **НПОУ «ЯКУТСКИЙ КОЛЛЕДЖ ИННОВАЦИОННЫХ ТЕХНОЛОГИЙ»** | | | | | | | |  | |
| (наименование образовательной организации) | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | **ОТДЕЛЕНИЕ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** | | | | | | | |  |
| (наименование структурного подразделения (кафедра / отделение)) | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| **ОТЧЕТ** | | | | | | | | | | |
| Разработка приложения для распознавания лиц с использованием OpenCV и нейронных сетей. | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | |  | Выполнил: | | | | | |
|  | | | |  | обучающиеся | | КИСП-23 | группы | | |
|  | | | |  | Еникеев Валерий,  Павлуцкий Айсен,  Андреев Айсен,  Корякин Игорь,  Борисов Кирилл. | | | | | |
|  | | | |  | (фамилия, имя, отчество (при наличии)) | | | | | |
|  | | | |  | \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | | | |
|  | | | |  | (личная подпись, дата) | | | | | |
|  | | | |  |  | | | | | |
|  | | | |  |  | | | | | |
| Проверил руководитель практики | | | |  |  | | | | | |
| от профильной организации: | | | |  | Проверил: | | | | | |
|  | | | |  | преподаватель | | | | | |
| (уч. степень, уч. звание, должность) | | | |  | (уч. степень, уч. звание, должность) | | | | | |
|  | | | |  | Федоров Дьулуур Андрианович | | | | | |
| (фамилия, имя, отчество (при наличии)) | | | |  | (фамилия, имя, отчество (при наличии)) | | | | | |
| Оценка | | |  |  | Оценка |  | | | | |
| \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | |  | \_\_\_\_\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | | | |
| (личная подпись, дата) | | | |  | (личная подпись, дата) | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| Рег. № \_\_\_\_\_\_\_ от «\_\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_\_ г. | | | | | | | | | | |
|  | | | | | | | | | | |
| Якутск, 2025 | | | | | | | | | | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc193536629)

[1. ТЕОРЕРИТЧЕСКАЯ ЧАСТЬ 4](#_Toc193536630)

[1.1 Терминология по проекту и глоссарий 4](#_Toc193536631)

[1.2 Распределение ролей и работы 7](#_Toc193536632)

[1.3 Стек технологий 8](#_Toc193536633)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 11](#_Toc193536634)

[2.1 Архитектура ПО 11](#_Toc193536635)

[2.2 Разработка проекта по ролям 12](#_Toc193536636)

[2.3 Контроль выполнения плана 12](#_Toc193536637)

[ЗАКЛЮЧНИЕ 14](#_Toc193536638)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc193536639)

[ПРИЛОЖЕНИЕ 18](#_Toc193536640)

# ВВЕДЕНИЕ

Актуальность исследования обусловлена возрастающей потребностью в надежных системах биометрической идентификации для обеспечения безопасности, контроля доступа и персонализации сервисов. Технологии распознавания лиц находят применение в различных сферах: от правоохранительной деятельности до банковских услуг и маркетинга.

Цель работы – разработка программного обеспечения для распознавания лиц с использованием методов компьютерного зрения и нейронных сетей.

Основные задачи исследования:

— Анализ современных алгоритмов распознавания лиц

— Проектирование архитектуры системы

— Реализация функциональных модулей

— Тестирование и оптимизация работы системы

— Разработка пользовательского интерфейса

Объект исследования – процесс автоматического распознавания лиц на цифровых изображениях.

Предмет исследования – методы и алгоритмы обработки изображений и машинного обучения для задач распознавания лиц.

Методы исследования включают анализ научной литературы, разработку программного обеспечения, экспериментальное тестирование и оценку результатов.

Практическая значимость работы заключается в создании эффективного решения для задач биометрической идентификации, которое может быть интегрировано в различные информационные системы.

# 1. ТЕОРЕРИТЧЕСКАЯ ЧАСТЬ

## 1.1 Терминология по проекту и глоссарий

Распознавание лиц — это процесс идентификации или верификации личности по изображению лица. Этот процесс включает несколько этапов, таких как обнаружение лица на изображении, извлечение уникальных признаков и сравнение этих признаков с базой данных. Эмбеддинг — это числовое представление изображения лица, полученное с помощью нейронной сети. Эмбеддинги используются для сравнения лиц и определения их схожести.

Каскады Хаара — это метод обнаружения объектов на изображении, основанный на использовании Haar-признаков. Этот метод был предложен Виолой и Джонсом в 2001 году и до сих пор широко используется благодаря своей простоте и высокой скорости работы. Каскады Хаара работают путем анализа интенсивности пикселей в определенных областях изображения и сравнения их с заранее обученными шаблонами.

FaceNet — это модель нейронной сети, разработанная Google, которая преобразует изображение лица в 128-мерный вектор. Эта модель обучается с использованием тройной функции потерь (triplet loss), что позволяет достичь высокой точности даже при наличии небольших изменений в изображении.

OpenCV — это мощная библиотека для компьютерного зрения с открытым исходным кодом, которая предоставляет инструменты для обработки изображений и видео, детекции объектов, машинного обучения и дополненной реальности. Она поддерживает несколько языков программирования, включая Python, что делает её доступной для широкого круга разработчиков. Основные функции включают чтение и запись изображений, применение фильтров, обнаружение объектов с помощью предобученных моделей (например, Haar cascades для лиц), работу с видео в реальном времени и интеграцию с глубоким обучением через модуль DNN. OpenCV используется в таких областях, как безопасность (распознавание лиц), медицина (анализ снимков), робототехника (навигация) и автомобильная промышленность (автономные транспортные средства). Библиотека постоянно развивается, добавляя поддержку новых алгоритмов и аппаратных ускорителей.

Python — это высокоуровневый интерпретируемый язык программирования общего назначения, который стал одним из самых популярных в мире благодаря своей простоте и читаемости кода. Он поддерживает несколько парадигм программирования, включая объектно-ориентированное, функциональное и процедурное программирование. Python имеет обширную стандартную библиотеку и экосистему сторонних пакетов, что делает его универсальным инструментом для веб-разработки, анализа данных, научных вычислений, автоматизации и искусственного интеллекта. Язык отличается динамической типизацией и автоматическим управлением памятью, что ускоряет процесс разработки. Python используется такими компаниями, как Google, NASA, Instagram и Spotify, для решения разнообразных задач — от скриптинга до построения сложных распределённых систем.

Requests — это простая, но мощная HTTP-библиотека для Python, которая позволяет отправлять HTTP/1.1 запросы без необходимости ручного добавления параметров в URL или формирования заголовков. Она автоматически обрабатывает кодировки, сессии, куки, аутентификацию и другие низкоуровневые детали работы с HTTP. Библиотека поддерживает все основные методы HTTP (GET, POST, PUT, DELETE и другие), работу с JSON, загрузку файлов и обработку SSL-сертификатов. Requests значительно упрощает взаимодействие с веб-API и парсинг веб-страниц по сравнению со стандартными модулями Python. Она широко используется в веб-скрапинге, микросервисной архитектуре и автоматизации работы с веб-сервисами, являясь де-факто стандартом для HTTP-запросов в Python-экосистеме.

PyQt и Tkinter — это две основные библиотеки для создания графических интерфейсов (GUI) на Python. PyQt представляет собой набор Python-привязок к фреймворку Qt, написанному на C++, и предлагает богатый набор виджетов и инструментов для создания сложных кроссплатформенных приложений. Он поддерживает современные функции вроде стилей CSS, анимации и работы с мультимедиа. Tkinter, входящий в стандартную библиотеку Python, проще в освоении, но обладает более ограниченными возможностями и устаревшим внешним видом. Обе библиотеки используют событийно-ориентированную модель программирования, но PyQt предоставляет более гибкую систему сигналов и слотов. Выбор между ними зависит от требований проекта: для быстрого создания простых интерфейсов подойдёт Tkinter, тогда как для профессиональных приложений с современным дизайном лучше выбрать PyQt.

FastAPI и Django — это популярные веб-фреймворки для Python, но предназначенные для разных задач. FastAPI — это современный, быстрый (высокопроизводительный) фреймворк для построения API, который использует асинхронность и автоматически генерирует документацию (Swagger UI). Он идеально подходит для микросервисов и высоконагруженных систем. Django — это полноценный MVC-фреймворк "всё включено" с ORM, админ-панелью и встроенной системой аутентификации, предназначенный для создания сложных веб-приложений. Он включает множество готовых решений, но может быть избыточным для простых API. FastAPI лучше подходит для современных асинхронных приложений, требующих высокой производительности, тогда как Django остаётся лучшим выбором для традиционных веб-проектов с богатой бизнес-логикой и сложной структурой данных.

MySQL и PostgreSQL — это две наиболее популярные реляционные системы управления базами данных с открытым исходным кодом. MySQL известен своей скоростью и простотой, широко используется в веб-приложениях (например, в составе LAMP-стека) и идеально подходит для операций чтения. PostgreSQL — это более продвинутая СУБД с поддержкой сложных запросов, пользовательских типов данных, оконных функций и полноценной ACID-совместимости. Она лучше подходит для аналитических задач и сложных транзакционных систем. Обе системы поддерживают JSON, но PostgreSQL предлагает более богатый функционал для работы с полуструктурированными данными. Выбор между ними зависит от требований проекта: для простых высоконагруженных веб-приложений часто выбирают MySQL, тогда как для сложных систем с жёсткими требованиями к целостности данных предпочтение отдают PostgreSQL.

Сверточные нейронные сети (CNN) — это специальный класс нейронных сетей, разработанный для эффективной обработки данных с сеточной структурой, таких как изображения. Они автоматически выявляют иерархию признаков — от простых границ и текстур на начальных слоях до сложных объектов на глубоких слоях. Ключевые компоненты CNN включают свёрточные слои (применяющие фильтры к входным данным), слои подвыборки (уменьшающие размерность представления) и полносвязные слои (для классификации). Современные архитектуры, такие как ResNet, EfficientNet и Vision Transformer, достигают сверхчеловеческой точности в задачах компьютерного зрения. CNN нашли применение в медицинской диагностике (анализ рентгеновских снимков), автономных транспортных средствах (распознавание дорожных сцен), системах безопасности (идентификация лиц) и даже в искусстве (стилизация изображений). Развитие CNN продолжается, сочетаясь с новыми подходами вроде attention-механизмов и обучением с малым количеством данных.

Модели нейронных сетей:

face\_recognition— для получения эмбеддингов лиц.

ResNet — модель с остаточными связями, позволяет строить очень глубокие сети без исчезновения градиента.

EfficientNet — архитектура, оптимизированная по размеру и точности.

Vision Transformer (ViT) — использует механизмы внимания для обработки изображений, конкурирующие с CNN.

Языки программирования:

Python — основной язык реализации проекта.

C++ — используется в компонентах OpenCV для ускоренной обработки.

Дополнительные библиотеки и инструменты:

NumPy — для работы с массивами и числовыми вычислениями.

TensorFlow / Keras — фреймворки для обучения и использования нейросетей.

## 1.2 Распределение ролей и работы

Разработка модулей, связанных с OpenCV, была поручена Еникееву Валерию и Корякину Игорю:

— Разработка модуля захвата видео с камер с помощью библиотеки OpenCV.

— Реализация методов предобработки изображений: коррекция освещенности, устранение шумов, нормализация.

— Обнаружение лиц на кадрах с использованием каскадных классификаторов или современных методов (например, модели на базе глубокого обучения).

— Извлечение признаков лиц для последующего распознавания.

— Оптимизация скорости обработки для работы в реальном времени.

— Тестирование системы на различных условиях освещения и фона, настройка параметров.

— Интеграция с остальными компонентами системы и обеспечение стабильной работы.

Основное приложение было разработано Павлуцким Айсеном и Борисовым Кириллом, Андреевым Айсеном:

— Проектирование и реализация графического интерфейса пользователя (GUI).

— Создание окон, меню, кнопок, индикаторов и области отображения видео.

— Обеспечение настройки системы (выбор камеры, порогов обнаружения, режимов работы).

— Реализация отображения результатов распознавания лиц в реальном времени.

— Реализация функций логирования событий и истории распознанных лиц.

— Обеспечение отзывчивости интерфейса и его адаптивности под разные устройства.

* Интеграция интерфейса с модулями обработки видео и распознавания.

## 1.3 Стек технологий

Обработка изображений и видеопотоков:

Для предобработки изображений, работы с видеопотоками и обнаружения лиц использовалась библиотека OpenCV. Она обеспечивала высокую скорость обработки и позволяла реализовать функции захвата и отображения видео, а также различные операции по улучшению качества изображений.

Распознавание лиц и извлечение признаков:

Для идентификации лиц и сравнения изображений применялись специализированные библиотеки и алгоритмы. Использовались инструменты для выделения лиц на изображениях, а затем — алгоритмы для извлечения признаков лицевых структур. Это позволяло точно и быстро находить совпадения между лицами в базе данных и текущими изображениями.

База данных:

Все данные о распознанных лицах, включая изображения, их признаковые векторы и метаданные, хранились в легковесной базе данных SQLite. Такой подход обеспечивал быстрый доступ к информации, а также простоту в настройке и использовании.

Пользовательский интерфейс:

Для создания удобного и интуитивно понятного интерфейса использовалась библиотека Tkinter. Она позволяла реализовать окна, кнопки и другие элементы управления для взаимодействия пользователя с системой, а также отображения результатов распознавания.

Обработка изображений и массивов данных:

Для работы с изображениями и массивами данных применялась библиотека Pillow. Она обеспечивала загрузку, изменение размеров, сохранение и преобразование изображений. Для числовых вычислений и обработки признаковых векторов использовалась библиотека NumPy, что позволяло эффективно работать с матрицами и выполнять необходимые математические операции.

Язык программирования и интеграция компонентов:

Основной язык разработки — Python, что обеспечивало быструю интеграцию всех компонентов системы, удобство в написании кода и возможность расширения функционала. Также использовались дополнительные модули для работы с файлами, логированием событий и обработки ошибок, что повышало стабильность и удобство эксплуатации приложения.

# 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1 Архитектура ПО

Архитектура приложения для распознавания лиц состоит из нескольких взаимосвязанных модулей, каждый из которых выполняет определенную функцию. Модуль захвата изображений отвечает за получение изображений с камеры или загрузку из файла. Этот модуль использует библиотеку OpenCV для обработки видеопотоков и изображений.

Модуль обнаружения лиц использует модель CNN для обнаружения лиц на изображении. CNN обеспечивает высокую точность даже при изменении угла поворота лица или освещения. Обнаруженные лица выделяются в виде прямоугольных областей (ROI), которые передаются на следующий этап.

Модуль извлечения признаков использует модель face\_recognition для преобразования изображения лица в эмбеддинг. Эмбеддинг представляет собой числовой вектор, который кодирует уникальные характеристики лица. Полученные эмбеддинги сохраняются в базе данных для последующего сравнения.

Модуль сравнения и распознавания сравнивает эмбеддинги с использованием метрик расстояния, таких как косинусное расстояние или евклидово расстояние. Если расстояние между эмбеддингами меньше заданного порога, лица считаются одинаковыми. Результаты сравнения передаются на следующий этап.

Модуль вывода результатов отвечает за отображение результатов распознавания на экране. Например, на изображении могут быть отмечены обнаруженные лица, а также выведена информация о распознанных личностях. В случае успешного распознавания результаты могут быть сохранены в базе данных или переданы в другие системы для дальнейшей обработки.

## 2.2 Разработка проекта по ролям

Разработка модулей, связанных с OpenCV, была выполнена Еникеевым Валерием и Корякиным Игорем. Используя библиотеку OpenCV (cv2), реализована функциональность для обнаружения доступных камер и захвата видеопотока с выбранного устройства. В методе detect\_cameras() осуществляется проверка подключённых камер, а в методе start\_video() — запуск видеовхода для последующей обработки. Для обнаружения лиц в кадре применяется библиотека face\_recognition, которая использует алгоритмы на основе Dlib для определения лицевых координат (face\_locations) и генерации их признаков (face\_encodings). Обнаруженные лица выделяются рамками и им присваиваются имена в случае узнавания.

Основное приложение было разработано Павлуцким Айсеном и Борисовым Кириллом, Андреевым Айсеном. Для сравнения лиц используются векторы признаков, полученные с помощью face\_recognition.face\_encodings. Сравнение осуществляется через функцию face\_recognition.compare\_faces, которая сравнивает текущие эмбеддинги с заранее загруженными и сохранёнными в базе данных известными лицами. Используется SQLite3 для хранения информации о распознанных лицах: имя, время распознавания и путь к сохранённому изображению. Реализованы методы init\_db() для инициализации базы и save\_face\_to\_db() для внесения новых записей. Для взаимодействия с пользователем используется библиотека Tkinter. Создаётся окно с возможностью выбора камеры, стартом и остановкой видеопотока, отображением текущего кадра и результатов распознавания.

2.3 Контроль выполнения плана

В процессе разработки приложения были выполнены все поставленные задачи. Пользовательский интерфейс был успешно разработан и интегрирован с основным приложением. Модуль захвата изображений и обнаружения лиц был реализован с использованием OpenCV и CNN. Модуль извлечения признаков и сравнения эмбеддингов был разработан с использованием face\_recognition. Все модули были интегрированы в единое приложение, которое успешно обнаруживает и распознает лица на изображениях и видеопотоках.

# ЗАКЛЮЧНИЕ

Разработка приложения для распознавания лиц с использованием OpenCV и нейронных сетей представляет собой сложный, но крайне важный процесс, который объединяет в себе достижения компьютерного зрения, машинного обучения и программной инженерии. В ходе работы над проектом были решены ключевые задачи, связанные с обнаружением, извлечением признаков и распознаванием лиц, а также интеграцией всех компонентов в единое приложение. Финальная версия проекта успешно соответствует ожиданиям, поставленным на начальном этапе разработки, и демонстрирует высокую точность и производительность в различных условиях.

Одним из ключевых достижений проекта стало успешное внедрение современных технологий, таких как OpenCV для обработки изображений и видеопотоков, а также нейронных сетей, включая face\_recognition и CNN, для извлечения признаков и обнаружения лиц. Эти технологии позволили достичь высокой точности распознавания даже в сложных условиях, таких как изменение освещения, угла поворота лица или наличие аксессуаров. Кроме того, использование предобученных моделей значительно ускорило процесс разработки и позволило сосредоточиться на оптимизации и интеграции компонентов.

Важным аспектом проекта стало распределение задач между членами команды, что позволило эффективно организовать процесс разработки. Разработка модулей, связанных с OpenCV, выполнена Еникеевым Валерием и Корякиным Игорем, позволила реализовать функциональность захвата изображений и обнаружения лиц с высокой точностью. Основное приложение, разработанное Павлуцким Айсеном, Борисовым Кириллом и Андреевым Айсеном, объединило все модули в единую систему, способную эффективно распознавать лица в реальном времени.

В процессе разработки были изучены и применены современные методы и подходы, такие как использование эмбеддингов для сравнения лиц, применение метрик расстояния для определения схожести, а также интеграция нейронных сетей в приложение. Эти методы позволили не только достичь высокой точности распознавания, но и обеспечить возможность масштабирования системы для работы с большими объемами данных.

Одним из важных уроков, извлеченных в ходе работы над проектом, стало понимание важности тщательного тестирования и оптимизации. В процессе разработки было проведено множество тестов для проверки точности и производительности приложения в различных условиях. Это позволило выявить и устранить потенциальные проблемы, такие как задержки в обработке изображений или ошибки в распознавании лиц при изменении условий освещения.

Кроме того, проект позволил команде освоить новые технологии и инструменты, такие как TensorFlow, Keras и face\_recognition, которые широко используются в области машинного обучения и разработки программного обеспечения. Эти навыки будут полезны для будущих проектов, связанных с искусственным интеллектом и компьютерным зрением.

В перспективе возможно дальнейшее развитие приложения, включая добавление новых функций, таких как работа с большими базами данных лиц, интеграция с облачными сервисами для хранения и обработки данных, а также использование более сложных моделей нейронных сетей для повышения точности распознавания. Также можно рассмотреть возможность адаптации приложения для работы на мобильных устройствах и IoT-устройствах, что расширит область его применения.

В заключение можно сказать, что разработка приложения для распознавания лиц стала важным шагом в изучении и применении современных технологий искусственного интеллекта и компьютерного зрения. Проект не только достиг поставленных целей, но и предоставил ценный опыт, который будет полезен для будущих исследований и разработок в этой области.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

OpenCV Documentation. https://docs.opencv.org/

TensorFlow Documentation. https://www.tensorflow.org/api\_docs

Keras Documentation. https://keras.io/

MTCNN: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. <https://arxiv.org/abs/1604.02878>

face\_recognition:https://face-recognition.readthedocs.io/en/latest/readme.html

NumPy: <https://numpy.org/doc/>

Python: <https://docs.python.org/3/>

SSD:[https://huggingface.co/spaces/liangtian/birthdayCrown/blob/3db8f1c391e44bd9075b1c2854634f76c2ff46d0/res10\_300x300\_ssd\_iter\_140000.caffemodel?](https://huggingface.co/spaces/liangtian/birthdayCrown/blob/3db8f1c391e44bd9075b1c2854634f76c2ff46d0/res10_300x300_ssd_iter_140000.caffemodel?utm_source=chatgpt.com)

Face Detection: <https://github.com/Isfhan/face-detection-python>?

# ПРИЛОЖЕНИЕ









