

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Р.Е. АЛЕКСЕЕВА

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Синицына Марина Евгеньевна

(фамилия, имя, отчество)

Институт Институт радиоэлектроники и информационных
технологий

Кафедра Вычислительные системы и технологии

Группа 16-В-1

Дата защиты « 06 » июля 2020 г.

Индекс
010

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт Радиоэлектроники и информационных технологий

Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника
(код и наименование)

Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы и сети
(наименование)

Кафедра Вычислительных систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра
(бакалавра, магистра, специалиста)

Студента Синицына Марина Евгеньевна группы 16-В-1
(Ф.И.О.)

на тему Программная система идентификации человека по изображению лица
(наименование темы работы)



СТУДЕНТ:

Синицына М.Е.
(подпись) (фамилия, и., о.)

02.07.2020
(дата)



РУКОВОДИТЕЛЬ:

Гай В. Е.
(подпись) (фамилия, и., о.)

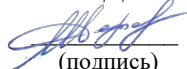
02.07.2020
(дата)

РЕЦЕНЗЕНТ:

(подпись) (фамилия, и., о.)

(дата)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ



Жевнерчук Д. В.
(подпись) (фамилия, и.о.)

02.07.2020
(дата)

КОНСУЛЬТАНТЫ:

1. По _____

(подпись) (фамилия, и., о.)

(дата)

2. По _____

(подпись) (фамилия, и., о.)

(дата)

3. По _____

(подпись) (фамилия, и., о.)

(дата)

ВКР защищена _____
(дата)

протокол № _____

с оценкой _____

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Кафедра Вычислительные системы и технологии



Утверждаю:
заведующий кафедрой
Д.В. Жевнерчук

15.03.2020

З А Д А Н И Е
на выполнение выпускной квалификационной работы

по направлению подготовки 09.03.01 Информатика и вычислительная техника

студенту: Сеницына Марина Евгеньевна группы 16-В-1

1. Тема ВКР Программная система идентификации человека по изображению лица

(утверждена приказом по вузу)

2. Срок сдачи студентом законченной работы 2.07.2020

3. Исходные данные к работе: Данные из видеопотока, база данных для обучения и тестирования

4. Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих разработке)

Требование к продукту, Обзор существующих систем, Анализ и подбор алгоритм решения задач,
Разработка структуры системы распознавания лиц , Тестирование системы

5. Перечень графических и других материалов, представляемых на защиту
Презентация

6. Консультанты по ВКР (с указанием относящихся к ним разделов)

Гай В. Е. Разделы: Разработка структуры системы распознавания лиц

Требование к продукту, Анализ требований к разрабатываемому продукту

Нормконтроль Гай В.Е.

7. Дата выдачи задания 13.03.2020

Код и содержание компетенции	Задание	Проектируемый результат	Отметка о выполнении
ПК-1 способностью разрабатывать модели компонентов информационных систем, включая модели баз данных и модели интерфейсов «человек - электронно-вычислительная машина»	Разработка интерфейса взаимодействия программного обеспечения и человека. Применяемый инструментарий: командная строка операционной системы.	Разработан ряд команд типа "запрос-ответ" с ключами ввода пользователем	Выполнено
ПК-2 способностью разрабатывать компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии программирования	Разработаны алгоритмы идентификации человека.	Разработан ряд алгоритмов для каждой подзадачи программного обеспечения	Выполнено
ПК-3 способностью обосновывать принимаемые проектные решения, осуществлять постановку и выполнять эксперименты по проверке их корректности и эффективности	Обосновать выбор алгоритма решения задач, обосновать выбор языка и	Обоснования раскрыты и приведены в дипломной работы	Выполнено

Руководитель



В.Е. Гай

(подпись)

Задание принял к исполнению

(дата)

Студент



М.Е. Синицына

(подпись)

**МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА»
(НГТУ)**

АННОТАЦИЯ

к выпускной квалификационной работе

по направлению подготовки 09.03.01 Информатика и вычислительная техника
(код и наименование)

Студента Синицына Марина Евгеньевна группы 16-B-1
(Ф.И.О.)

по теме Программная система идентификации человека по изображению лица

Выпускная квалификационная работа выполнена на 54 страницах, содержит 0 диаграмм, 0 таблиц, библиографический список из 7 источников, 1 приложения.

Актуальность: автоматизация процесса идентификации личности, позволяющим уменьшить трудоемкость решения задачи и увеличить скорость идентификации

Объект исследования: Аппаратное и программное обеспечение системы пропускных пунктов, обеспечения безопасности личных данных .

Предмет исследования: алгоритмы нахождения объекта, нахождение особых точек и принятия решения

Цель исследования: реализация системы идентификации личности

Задачи исследования: Разработка алгоритмов, тестирование системы.

Методы исследования: Моделирование, эксперимент

Структура работы:

1. Требование к продукту
2. Анализ поставленной задачи
3. Разработка структуры системы распознавания лиц
4. Тестирование системы

Во введении приводится цель работы и особенности

В 1 разделе «Требование к продукту» определяются требования к разработке

Во 2 разделе «Анализ поставленной задачи» производится выбор средств для реализации функциональных требований, а также подбираются алгоритмы решения задачи

В 3 разделе «Разработка структуры системы распознавания лиц » описывается схема решения задачи

В 4 разделе «Тестирование системы » проводятся и фиксируются тесты разработанной программы

В заключении Приводятся основные выводы по работе

Выводы:

1. Выполнена система идентификации личности по биометрии лица
2. Рассмотрены алгоритмы решения всех подзадач программного обеспечения

Рекомендации:

1. В дальнейшем можно развить проект и добавить в него технологию от мошенничества







/ Синицына

подпись студента /расшифровка подписи

«12» июня 2020 г.

Оглавление

Введение	3
1. Требование к продукту.....	4
1.2 Назначение разработки и область применения	4
1.3 Технические требования	4
2. Анализ требований к разрабатываемому продукту	5
2.1 Выбор операционной системы	5
2.2 Выбор языка программирования	7
2.3 Выбор среды разработки	9
2.4 Обзор существующих систем распознавания лиц	11
2.5 Выбор подходов к решению задачи распознавания лиц	14
3 Разработка структуры системы распознавания лиц	19
3.1 Разработка общей структурной схемы.	19
3.2 Разработка алгоритма предварительной обработки данных	21
3.3 Разработка алгоритма нахождения лица	21
3.4 Алгоритм выявления признаков	24
3.5 Алгоритм принятия решений	25
4 Тестирование системы	26
4.1 Описание набора данных	26
4.2 Описание метода тестирования	27
Заключение	30
Литературный список	31
Приложение	32

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020						
Изм.	Лист	№ докум.	Подпись	Дата	Программная система идентификации человека по изображению лица			Лит.	Лист	Листов	
Провер.	Гай В. Е.		02.07								
Разраб.	Синицына М.Е.		02.07						2	54	
Н. контр.	Гай В.Е.		02.07	16-В-1							
Утв.	Жевнерчук Д.В.		02.07								

Введение

Распознавание лиц - одна из новых тенденций, широко распространённых в данный момент во многих приборах и индустриях. Востребованность автоматической системы обнаружения и отслеживания лиц возросла, поскольку она необходима для видеонаблюдения и новых пользовательских интерфейсов. Система распознавания лица в качестве биометрической технологии менее точна, чем распознавание радужки и отпечатков пальцев, они повсеместно применяются в связи с их неинвазивным и бесконтактным процессом.

На данный момент технология распознавания лиц внедрилась уже довольно глубоко в нашу жизнь. Различные новые технологии позволяют улучшать алгоритм и увеличивать скорость его работы, что позволяет повысить степень безопасности в различных отраслях. Таких как банковская безопасность, доступ к личным данным клиентов, пропускные пункты в аэропортах и даже хранение личных данных на различных переносных устройствах.

А так же идентификация подозрительных или разыскиваемых лиц, при проведении массовых мероприятий и обеспечения безопасности общественно значимых событий (голосования, митинги и т.д.)

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

1 Требование к продукту

1.1 Назначение разработки и область применения

Разрабатываемая система распознавания и идентификации личности человека создана для входа или разрешения к использованию ресурсов ограниченному числу людей. Система, приобретая широкое распространение, может использоваться во многих сферах, таких как безопасность, путешествия, развлечения. Так, в сфере безопасности - на пропускных пунктах или при доступе к личным банковским счетам, в индустрии развлечений систему можно использовать для сравнения степени схожести черт лица со знаменитостями, а в сфере путешествий - в аэропортах, где прохождение через регистрацию с помощью идентификации лица человека увеличивает скорость движения очереди.

1.2 Технические требования

Требования, представляемые сконструированной системой к ЭВМ:

1. операционная система с графическим интерфейсом;
2. требование к аппаратному обеспечению формируется операционной системой;
3. дисплей, камера.

Требование к опциям разрабатываемой системы идентификации человека по изображению лица:

1. Система должна обеспечить возможность пользователю добавлять в базу данных информацию о новом абоненте системы;
2. Система должна сделать предварительную обработку видеоизображения, поступающего на вход;
3. Система должна сформировать комплекс признаков индивидуальных черт каждого зарегистрированного пользователя;
4. Система должна сформировать комплекс признаков индивидуальных черт каждого человека, при попытке использования им системы.

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

2 Анализ требований к разрабатываемому продукту

2.1 Выбор операционной системы

В настоящее время самыми востребованными операционными системами являются Windows, Linux, Android, MAC OS. Рассмотрим каждую из них.

MAC OS создан из дистрибутива BSD UNIX. Mac OS – это закрытая операционная система (производитель – компания Apple). Она известная и распространённая, но из-за того что она сконфигурирована специально под аппаратную часть Mac, функционал и приложения операционной системы узко направлены и не универсальны.

Microsoft Windows представляет собой ряд проприетарных семейств графических операционных систем, которые разрабатываются и распространяются на коммерческой основе компанией Microsoft. Операционная система Windows предлагает широкий выбор программного обеспечения различного предназначения и поддержку для различных задач администрирования и конструирования собственных приложений. Microsoft Windows может устанавливаться на разные аппаратные платформы для решения широкого круга задач. Причем данная операционная система проста в понимании и доступна для освоения неподготовленными пользователями.

Linux - операционная система, состоящая в семействе Unix. Особенностью данной системы является ее базирование на общем для всего семейства операционных систем ядре Linux. Данная операционная система свободно распространяемая (в большинстве случаев, например, по лицензии GNU GPL) и в ней используется открытое программное обеспечение. Однако, не все операционные системы Linux просты в понимании и управлении для рядовых пользователей.

Android - это мобильная операционная система, основанная на модифицированной версии ядра Linux и другого программного обеспечения с открытым исходным кодом, предназначенная в основном для мобильных устройств с сенсорным экраном, таких как смартфоны и планшеты.

Создание программного обеспечения для каждой операционной системы занимает достаточное количество материальных и временных ресурсов. Поэтому, при создании программного обеспечения в рамках дипломной работы учтено требование об универсальности приложения, которое способно запускаться и функционировать на любой программной

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист 5
Изм.	Лист	№ докум.	Подпись	Дата		

основе. С этой целью для создания программного обеспечения необходимо использовать соответствующий кроссплатформенный язык программирования.

Вместе с тем, основной выбор целесообразно сделать на операционной системе Microsoft Windows, как на наиболее распространенной системе для полноценных персональных компьютеров. Создаваемое в рамках данной дипломной работы программное обеспечение планируется к внедрению именно на наиболее распространенных аппаратных платформах, на которых устанавливается операционная система из семейства Microsoft Windows.

2.2 Выбор языка программирования.

Наиболее распространенными языками программирования для проектирования и создания систем идентификации человека по изображению лица являются C++ и Python. Рассмотрим преимущества и недостатки каждого из них для решения данной задачи.

C++ – компилируемый, строго типизированный язык программирования общего назначения. Он поддерживает многие модели программирования: процедурную, функциональную, обобщённую. Особенное внимание уделяется поддержке объектно-ориентированного программирования. Однако, стоит учитывать, что при создании приложений в C++ необходимо ограничиваться той программной платформой, на которой оно будет запускаться и функционировать. Для полноценного распространения созданного программного решения на все самые распространённые платформы необходимо создавать для каждой из них свою версию программы. Данный подход не универсален при необходимости внесения доработок и изменений в ранее созданные версии программы, что может привести к «путанице» и увеличению количества ошибок в компилируемом коде. Вместе с тем, при разработке на C++ программных решений для работы с нейронными сетями и распознаванием образов, даже с учетом созданных сторонними разработчиками библиотек, от программиста требуются дополнительные усилия и действия при описании структур программы (например, детальная настройка используемых классов с помощью механизмов инкапсуляции, наследования, полиморфизма).

Python – высокоуровневый, объектно-ориентированный язык программирования общего назначения. В тоже время, Python является кроссплатформенным, что позволяет запускать созданные на нем решения практически на всех известных программных платформах. Кроме того, данный язык программирования имеет многочисленные реализации и расширения, предназначенные для решения разных задач (CPython и JPython – тесная интеграция с языками C и Java соответственно, IronPython – интеграция со средой Microsoft .Net Framework, NumPy и SciPy – математические вычисления различной сложности и др.). Помимо этого, в языке Python для работы с нейронными сетями создано большое количество библиотек и модулей, которые существенно упрощают решение задач данного класса.

С учетом вышеизложенного для создания программного обеспечения в рамках дипломной работы целесообразно выбрать язык программирования

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист 7
Изм.	Лист	№ докум.	Подпись	Дата		

Python. Данное решение обусловлено меньшими трудозатратами на разработку и поддержку создаваемого программного решения, а также возможностью его использования на различных аппаратных и программных платформах, поддерживающих запуск приложений, созданных на Python.

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

2.3 Выбор среды разработки

Для удобства создания программного обеспечения в рамках данной дипломной работы целесообразно использовать интегрированную среду разработки (IDE). Среда IDE представляет собой программное приложение, которое предоставляет комплексные возможности для пользователей в сфере разработки программного обеспечения.

В настоящее время имеется множество IDE. Часть из них сильно развиты в своем функционале и повсеместно распространены, часть узконаправлены и используются под решение конкретных задач. В большинстве случаев каждая из них предназначена для определенного языка программирования. Однако имеются IDE, которые поддерживают сразу несколько языков программирования. Ниже рассмотрим несколько сред разработки, среди которых будет осуществлен выбор в вопросе ее использования при разработке программного обеспечения в рамках данной дипломной работы.

IDLE Python (Shell) – стандартная интегрированная среда разработки, устанавливаемая вместе с пакетом языка программирования Python. Она предназначена для работы только с языком Python. В IDLE входит минимальный набор инструментов, предназначенный только для начинающих пользователей. Кроме того, в данной среде разработки затруднительно написание программного решения, состоящего из нескольких, связанных друг с другом, модулей.

Eclipse – свободно распространяемая интегрированная среда разработки модульных кроссплатформенных приложений. Свою известность она приобрела благодаря функции разработки расширений. Eclipse написан на Java, поэтому выступает платформо-независимым продуктом. Однако, использование данной среды разработки для создания решений на языке программирования Python, требует дополнительных усилий в вопросе скачивания и установки необходимых модулей и расширений. Кроме того, Eclipse может вызывать затруднения при работе с интерфейсом данной среды разработки.

Visual Studio – интегрированная среда разработки от компании Microsoft. Она применяется для создания компьютерных программ, различных веб-технологий и мобильных приложений. Visual Studio распространяется в открытом доступе для личного пользования, а также на коммерческой основе (расширенные версии, предназначенные для профессиональных разработчиков). Данная среда разработки приобрела широкую известность

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

благодаря удобному интерфейсу и большому количеству поддерживаемых технологий. Для создания приложений на языке программирования Python необходима установка дополнительного расширения. Опыт его использования показывает, что оно не в полной мере может интегрироваться с модулями и библиотеками сторонних разработчиков. Указанный факт негативно сказывается на выборе данной среды разработки для создания программного продукта для распознавания образов на языке Python.

Anaconda – дистрибутив для создания программ на языках программирования Python и R, содержит комплект известных свободных библиотек, решающих задачи науки о данных и машинного обучения. Особенность Anaconda в том, что дистрибутив включает в себя все необходимые пакеты и расширения, что требует от пользователя однократной загрузки данной среды программирования (с необходимой в дальнейшем локальной установкой дополнений). Кроме того, Anaconda работает с менеджером разрешения зависимостей conda, который основывается на графическом интерфейсе Anaconda Navigator, что позволяет отказаться от консольных менеджеров установки пакетов (например, pip).

С учетом описанных характеристик рассмотренных сред разработки целесообразно выбрать среду разработки Anaconda, как наиболее подходящую для поставленной задачи.

2.4 Обзор существующих систем распознавания лиц

На данный момент биометрическое распознавание лиц актуально и постоянно развивается. Постоянно появляется новое программное обеспечение, все больше внедряясь в различные сферы жизнедеятельности, тем самым создавая конкуренцию.

Рассмотрим наиболее известные сервисы распознавания лиц.

В качестве примера компании, успешно использующую технологии нейронных сетей, можно привести «ZKTeco». Она является лидером в области распознавания лиц. Ее филиалы распространены по всему миру. Компания производит не только программные продукты, но и сопровождает их специальным оборудованием. Таким образом, она поставляет уже готовый продукт для решения задач распознавания, контроля и безопасности. Один из ведущих ее проектов - «Visible Light», система распознавания лиц. Они расширили свой алгоритм для улучшения качества распознавания и внедрили некоторые механизмы против обманных действий мошенников.

Этапы технологии, по которой функционирует данное программное обеспечение, представлены на рисунке 1.

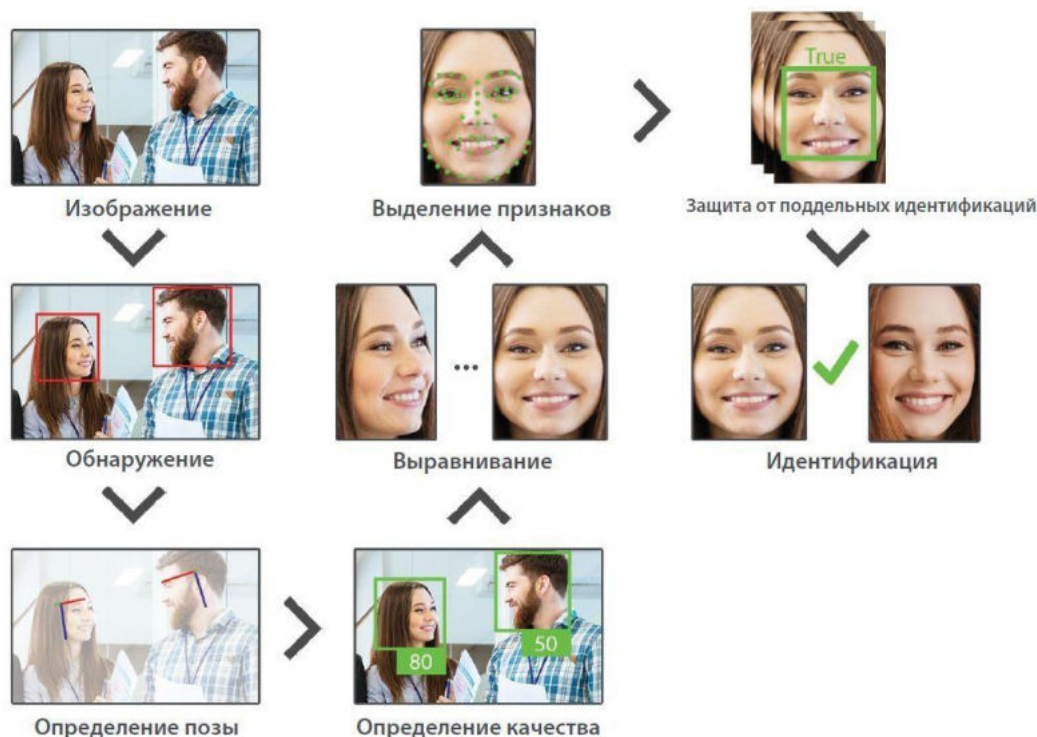


Рис.1 – Этапы алгоритма распознавания лица компании «ZKTeco»

Достоинства данного программного обеспечения:

- Система защиты от ложных изображений.
- Высокая точность определения личности.
- Минимальные затраты времени.

Недостатки «Visible Light»:

- Дорогое аппаратное и программное обеспечение.
- Базы данных занимают большой объем памяти.

Пример работы данного программного обеспечения представлен на рисунке 2.



Рис.2 – Работа алгоритма компании «ZKTeco»

Примером другой такой компании с подобным направлением деятельности является группа ЦРТ - Российская научно – исследовательская компания. Она специализируется на аудио идентификации, видео идентификации и на распознавании лиц. Созданная система уже применяется

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист 12
Изм.	Лист	№ докум.	Подпись	Дата		

во многих проектах и отраслях, например, такие как транспортные объекты, стадионы и ледяные арены, «умный город».

Достоинство программных продуктов ЦРТ:

- Совместимость со многими системами наблюдения.
- Быстрая обработка информации.
- Высокая результативность.

Недостатки:

- Дорогое аппаратное и программное обеспечение.
- Базы данных занимают большой объем памяти.

Пример работы алгоритма распознавания лиц в программном обеспечении компании группы ЦРТ представлен на рисунке 3.

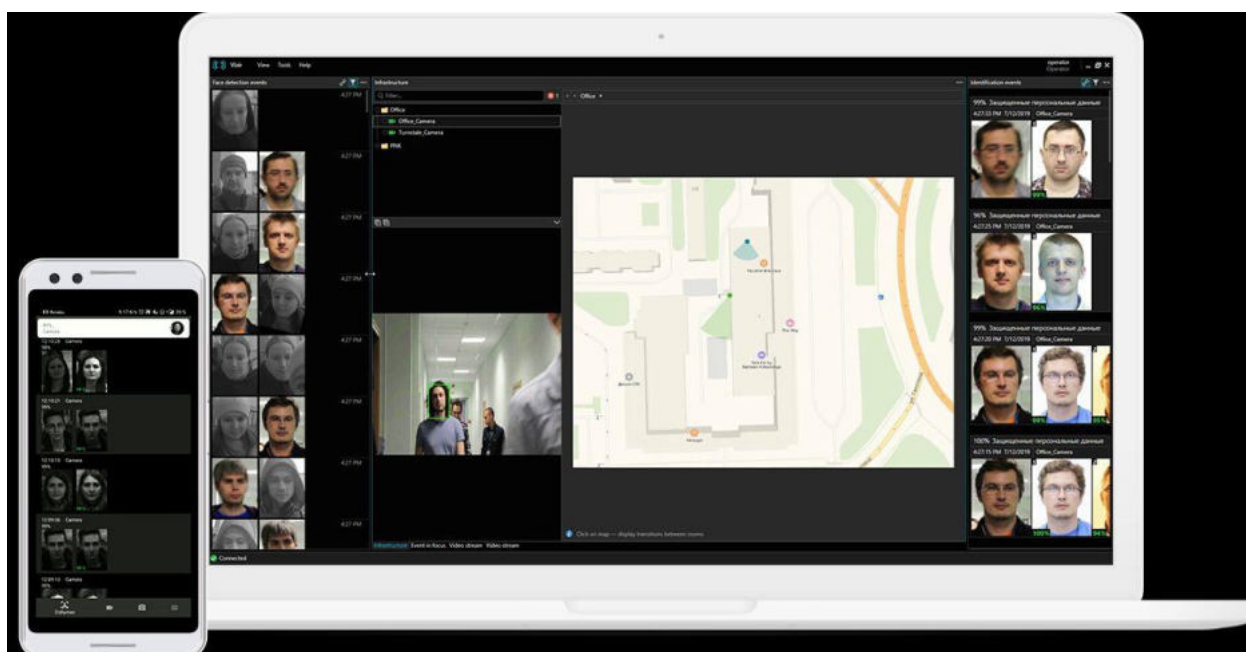


Рис. 3 – Пример работы распознавания лиц программного обеспечения группы ЦРТ

2.5 Выбор подходов к решению задачи распознавания лиц

Алгоритм распознавания лиц можно разделить на 3 части:

1. Предварительная обработка изображения - нахождение на нем лица;
2. Выявление признаков описания лица;
3. Принятие решения о соответствии выявленных признаков ранее известным признакам (кластеризация).

Для каждой части алгоритма необходимо определить подход к его решению.

Так, определить лицо в кадре можно следующими способами:

- применить алгоритм Viola-Jones;
- использовать нейронную сеть.

В основе метода Viola-Jones заложена технология скользящего окна (рисунок 4). Суть алгоритма состоит в том, что рамка меньшего размера, чем исходное изображение, передвигается по изображению с некоторым шагом. На каждом шаге используется каскад классификаторов Хаара для вычисления расположения лица в рассматриваемой рамке. В случае отсутствия лица в рамке, она смещается в следующий сектор с учётом шага и алгоритм повторяется. Если лицо найдено, то применяются признаки Хаара.

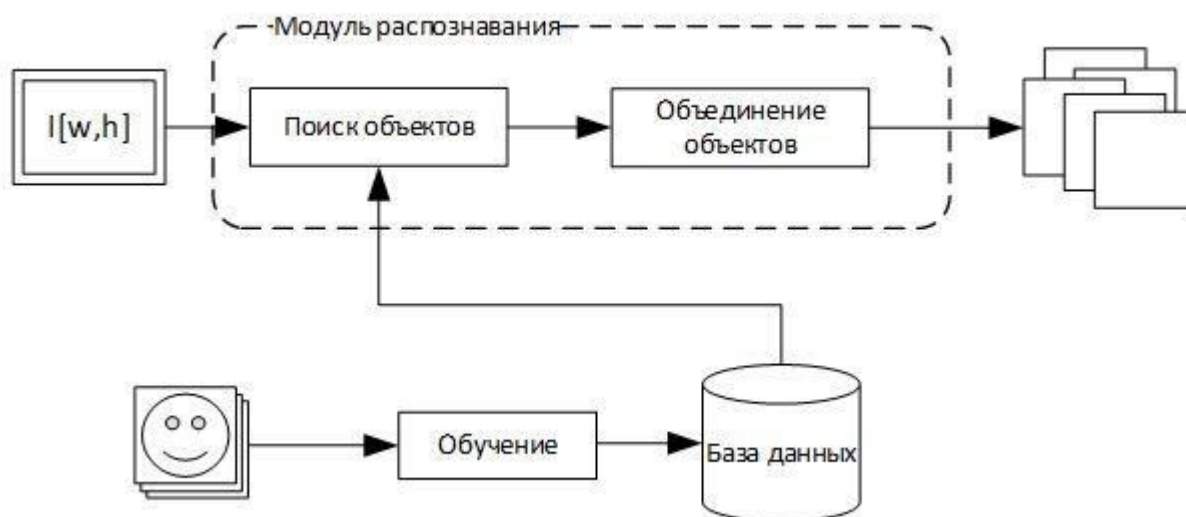


Рисунок 4 - Схема работы алгоритма Viola-Jones

Для распознавания лица в алгоритме используются признаки Хаара (рисунок 5). Они состоят из ограниченных прямоугольных областей, разбитых на набор разнообразных прямоугольников меньших по размеру.

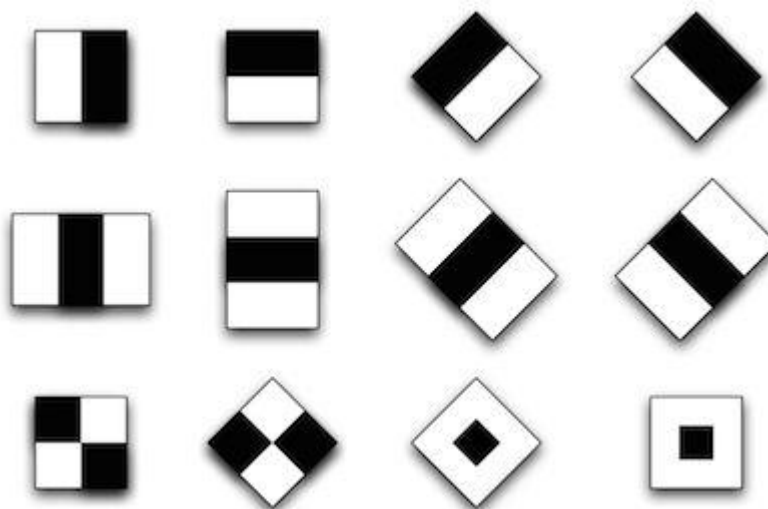


Рисунок 5 - Пример признаков Хаара

Стоит отметить, что в алгоритме Viola-Jones не используются прямоугольные области с поворотом. Для вычисления значений конкретного признака на изображении, необходимо сложить значение яркости пикселей в каждой из областей, затем определить разность значений яркости между областями. В данном случае применяется алгоритм обучения AdaBoost. Он формирует классификатор, представляющий собой линейную комбинацию взвешенных классификаторов. Взвешенный классификатор, в свою очередь, есть пороговая функция, которая базируется на следующем выражении:

$$h(x) = \text{sgn}(\sum_{j=1}^M a_j h_j(x)) ,$$

где

$$h_j(x) = \begin{cases} -s_j & \text{if } f_j < 0 \\ s_j & \end{cases}$$

При обучении алгоритма определяется значение (f_i), полярность (s_j) и коэффициенты (a_j).

К преимуществам данного алгоритма можно отнести большую скорость вычисления операций и простоту его реализации. В качестве недостатков стоит отметить, что данный алгоритм далеко не всегда дает точное и правильное решение [5].

В качестве альтернативы стоит рассмотреть использование нейронных сетей, которые в программном обеспечении представляют собой математическую модель, схожую с моделью человеческих нейронных клеток, слои которой связаны друг с другом и выполняют параллельные вычисления. Для распознавания объектов наиболее эффективно применяются сверточные нейронные сети (рисунок 6). Их особенность в том, что преобразование изображения осуществляется с помощью специальных ядер свертки. Ядра свертки представляют собой матрицы обычно размерности 3x3 или 5x5 (чаще всего нечетной величины). Каждый раз при проходе изображения через итерации и новые слои, выделяются новые признаки или дополняют уже известные. К тому же, у сверточных нейронных сетей имеется способность к обучению (например, обучение с учителем, без учителя, с подкреплением).

Обучение с учителем представляет собой сравнение входных данных с имеющимися уже выявленными чертами в специальной, заранее подготовленной подборке.

При обучении без учителя нейронная сеть собирает выходные данные только на основе входных данных, проходящих через архитектуру сети.

Обучение с подкреплением основано на взаимодействии с некоторой средой, в которой на принятие решений влияют сигналы подкрепления.

К достоинствам сверточных нейронных сетей относятся высокая точность обработки изображений, а также большой выбор уже созданных инструментов для работы с ними. В качестве недостатков стоит выделить требовательность к размеру места памяти, в которой идет обработка изображений, что требует наличия дорогого аппаратного обеспечения.



Рисунок 6 – Пример работы сверточной нейронной сети

С учетом вышеизложенного, для предварительной обработки изображений целесообразно использовать сверточные нейронные сети, так как

они дают наиболее точный результат при решении данной задачи, что соответствует требованиям, предъявляемым к создаваемому программному обеспечению в рамках данной дипломной работы.

Для выявления признаков описания, второй части решения поставленной задачи, наиболее распространено использование алгоритма SIFT (Scale-invariant feature).

Алгоритм SIFT является алгоритмом нахождения признаков. Признаки, на которых основывается этот алгоритм, локальны и базируются на выделении в изображении объекта конкретных особых точек. Их нахождение не зависит от изменений масштаба и вращения изображения. Так же, они устойчивы к изменению освещения и шуму. Алгоритм выявления точек состоит из 3 шагов:

- определение особых точек с помощью фильтров Гаусса, которое заключается в сворачивании изображения с фильтрами в различных масштабах. Уже обработанные изображения группируются по октаве;
- вычисление разности последовательных размытых изображений;
- выбор минимумов и максимумов ключевых точек.

Разность Гаусса определяется по формуле:

$$D(x, y, \sigma) = L(x, y, k_i \sigma) - L(x, y, k_j \sigma)$$

где L является свёрткой исходного изображения I с размытием по Гауссу G в масштабе k, то есть,

$$L(x, y, k \sigma) = G(x, y, k \sigma) * I(x, y)$$

Обнаружение точек с помощью вычисления минимума и максимума даёт слишком обширный результат. Множество кандидатов в опорные точки путают алгоритм, так как многие из них неустойчивые. В данном случае проводится детальная подгонка к соседям.

Интерполяция выполняется благодаря квадратичному разложению Тейлора. Она задаётся уравнением:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

Для выявления ключевых точек при низком контрасте, как правило, используется ряд Тейлора второго порядка со смещением. Если значение текущей точки оказалось меньше значения ряда Тейлора, то этот кандидат в опорные точки не учитывается. Если значение больше, то кандидат в

ключевые точки остаётся с сохранением места положения. Результат работы алгоритма SIFT представлен на рисунке 7[3].

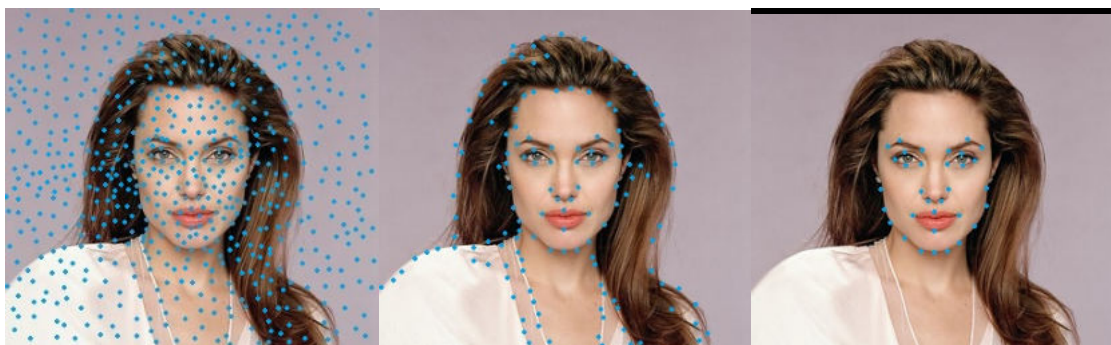


Рисунок 7 – Результат выполнения алгоритма SIFT

Третью часть проекта можно реализовать многими способами. Рассмотрим некоторые из них:

- с помощью нейронных сетей;
- с помощью метода ближайших соседей.

Нейросеть можно использовать для разных подзадач, благодаря ее всевозможным конфигурациям и методам вычисления обрабатываемых данных. Многое из этого было описано выше.

Метод ближайших соседей может справиться с двумя задачами: классификацией объектов и регрессией. В свою очередь задача классификации может включать в себя задачу кластеризации. Для применения кластеризации можно использовать алгоритм обучения, как с учителем, так и без него.

Алгоритм сформирован так, что при работе с ним выявляется минимизация суммарных квадратичных отклонений точек кластеров от центра этих кластеров.

$$V = \sum_{i=1}^k \sum_{x_j \in S_i} (x_j - \mu_i)^2$$

где k — число кластеров, S_i — полученные кластеры, $i = 1, 2, \dots, k$ и μ_i — центры масс векторов $x_j \in S_i$.

Преимущества данного алгоритма заключаются в скорости выполнения операций и легкости его использования. К недостаткам относится недостаточная точность вычислений.

Исходя из вышеизложенного, в данной работе целесообразней применить метод ближайших соседей.

3. Разработка структуры системы распознавания лиц

3.1 Разработка общей структурной схемы.

Разрабатывая систему распознавания лиц необходимо учитывать несколько задач. К каждой из этих задач подбирается свой алгоритм решения и, в итоге, это собирается в единую систему. Выбрав к каждой задаче алгоритм можно построить общую структурную схему системы, которая состоит из 3 пунктов:

1. нахождение лица в кадре;
2. выявление системой признаков;
3. принятие решения.

Для реализации первой части системы применяется сверточная нейронная сеть, для второй - алгоритм SIFT, для третьей - «метод ближайших соседей». Исходя из этого, строится обобщенная схема системы (рисунок 8).

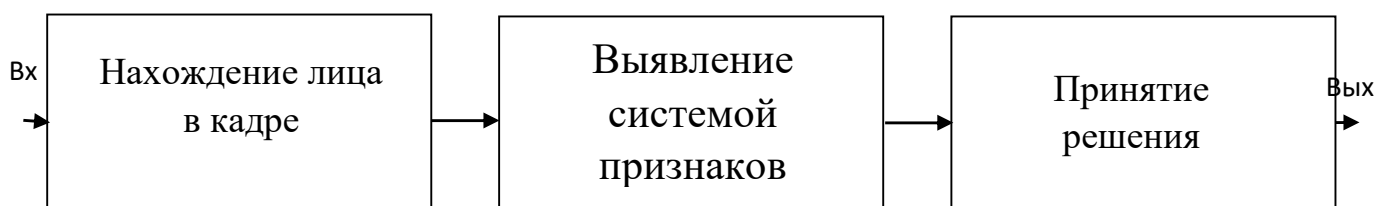


Рисунок 8 – Обобщенная структура схемы

На вход структуры подаем предобработанное изображение. Для упрощения восприятия данных алгоритмом, оно уже обработано от шумов и преобразовано в палитру серого. На выходе после всех преобразований в системе выводиться то же изображение, но с пометкой о разрешении доступа.

В сформированной системе база данных представляет собой отдельный файл, в котором заложена вся информация о каждой личности с правом доступа к ресурсам, которые защищает эта система. Информация о каждой фотографии имеет несколько атрибутов: имя файла изображения, количество блоков в изображении, список признаков изображения.

Общая структура базы данных представлена на рисунке 9.

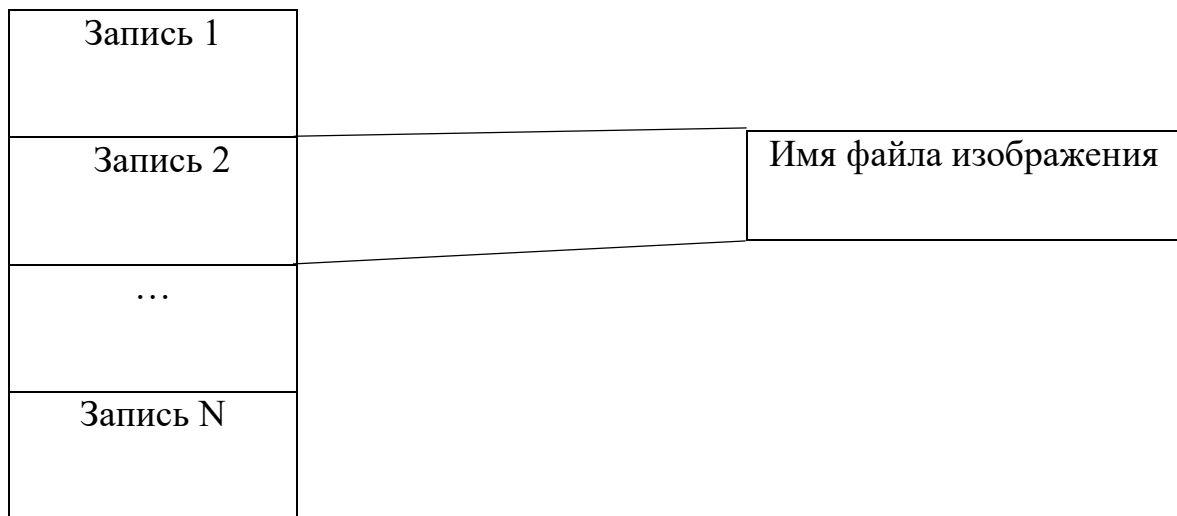


Рисунок 9 – Структура базы данных

Развернутая структура создаваемого программного обеспечения для распознавания лиц представлена на рисунке 10.

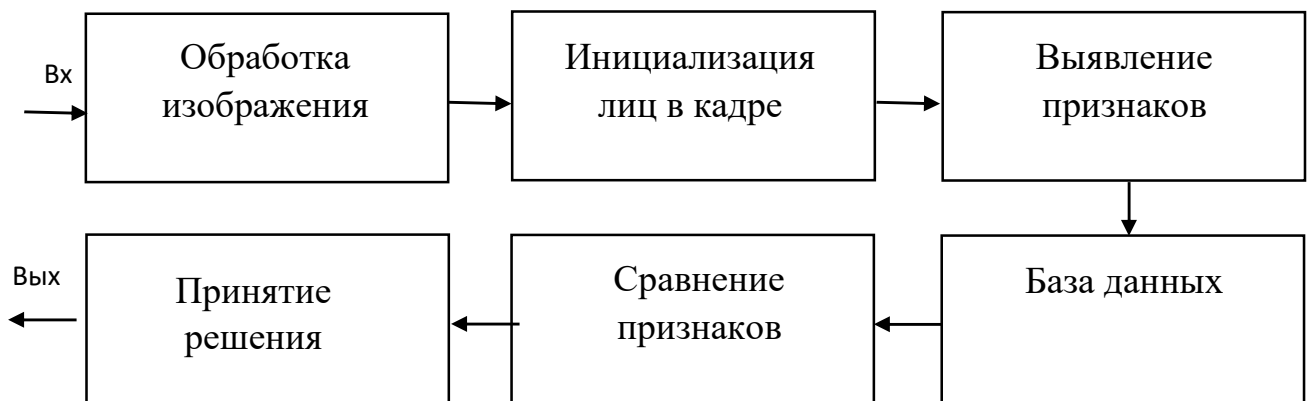


Рисунок 10 – Развернутая структура создаваемого программного обеспечения

3.2 Разработка алгоритма предварительной обработки данных

В настоящее время техника в основном воспринимает и воспроизводит цветные изображения. Из-за этого количество обрабатываемой информации в виде кодирования цветов и оттенков возросло во много раз. Чтобы анализировать и удобно обращаться к структурно сложной информации, были разработаны несколько подходов, например RGB, HSV. В цветовая палитра HSV строится на цилиндрической системе координат. Распределение цветов представляет собой шестигранный конус, перевернутый на вершину. RGB обозначает три цвета красный, зеленый и синий. Вся палитра цветов складывается из трех цветов благодаря насыщенности цвета и яркости света. Более распространенным методом является RGB. Благодаря простому строению, модель RGB используется более часто, чем остальные.

Перед тем как подать изображение с камеры в систему, оно преобразуется из цветных каналов RGB в серые оттенки с помощью разложения изображения на три разных канала. Из-за этого система принимает сразу три матрицы одного и того же изображения.

Это позволяет упростить вычисление по алгоритму распознавания лица.

В процессе прохождения алгоритма, матрицы обратно суммируются по своим значениям, и на выходе изображение выдается в цвете.

3.3 Разработка алгоритма нахождения лица

Для решения нахождения лица в кадре будет использована сверточная нейронная сеть. Этот метод включает в себя несколько этапов [1][2][4]:

- сбор данных для обучения нейронной сети;
- подготовка и нормализация данных;
- выбор и формирование структуры сети;
- экспериментальный подбор параметров обучения;
- обучение.

Сбор данных включает в себя формирование базы данных лиц, либо же применение уже готовой базы данных. В этой работе будет использована уже готовая библиотека с множеством лиц, которую можно открыто скачать в общем доступе. В этой библиотеке находятся несколько наборов классов для применения в машинном обучении для разных задач.

Подготовка и нормализация данных в данном случае не нужны, так как используется уже готовая библиотека.

При выборе структуры сети сделан упор на несколько признаков. А именно, такие как меньшая нагрузка на аппаратную платформу, легкость реализации и быстроту выполнения алгоритма. Таким образом, архитектура нейронной сети представлена на (рисунок 11).

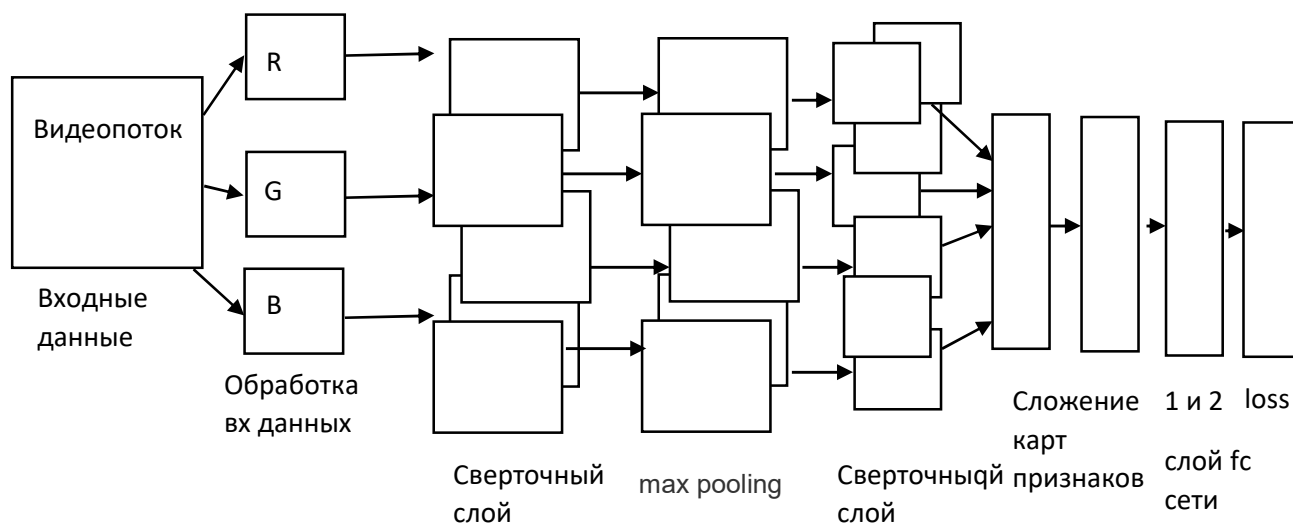


Рисунок 11 – Архитектура нейронной сети

В каждом из слоев вычисляются операции. Рассмотрим каждую операцию отдельно. Суть работы первого слоя свертки заключается в перемножении матриц входного изображения на ядро свертки. При этом ядро свертки меньше, чем исходное изображение, и передвигается с указанным шагом. На этапе выявления первых признаков изображения ядро выделяет вертикальные края объекта (рисунок 12).

-1	0	1
-2	0	2
-1	0	1

Рисунок 12 – Ядро свертки для первого этапа

Если развернуть ядро свертки по горизонтали, то находятся горизонтальные границы изображения (рисунок 13).

1	2	1
0	0	0
-1	-2	-1

Рисунок 13 – Развернутое ядро свертки

Пример вывода работы свертки представлен на рисунке 14.

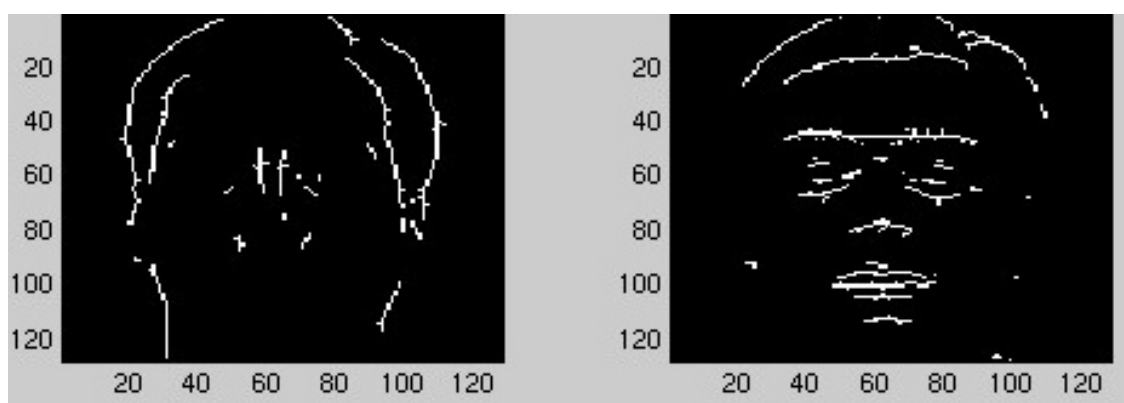


Рисунок 14 – Пример результатов работы свертки

Следующий слой – подвыборочный - используется для уменьшения размерности. Этот шаг позволяет уменьшить вычислительную мощность для облегчения работы алгоритма. Данная операция называется пулинговый слой. Пулинг может выполняться двумя вариантами: нахождение максимального либо среднего значения из обрабатываемой области (рисунок 15).

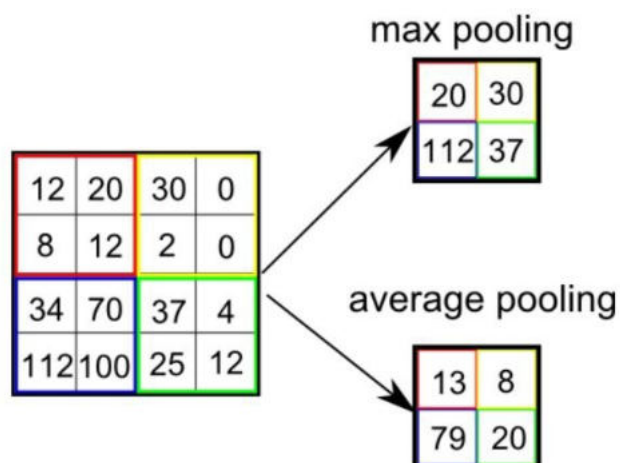


Рисунок 15 – Работа операции пулинга

В созданном проекте применен максимальный пулинг.

Переходя к слою активации скалярный результат свертки попадает на функцию активации. Функция активации – непрерывная, нелинейная функция. В данной работе использована функция сигмойды (рисунок 16).

$$f(x) = \frac{1}{1+e^{-x}} \text{ - сигмоидная функция активации}$$

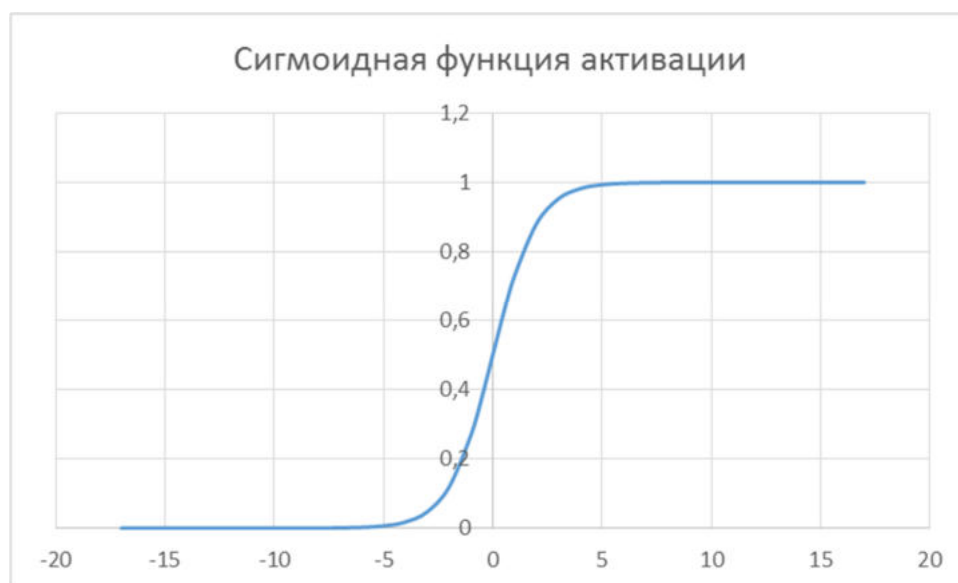


Рисунок 16 – Сигмоидная функция активации

3.4 Алгоритм выявления признаков

Для выявления признаков в данной работе используется алгоритм SIFT. Задачей алгоритма является нахождение особых точек. Для человека идентификация какого-либо предмета дается достаточно легко, но не для компьютера. Для машины это очень сложная задача, состоящая из многих шагов и вычислений, требующая множество ресурсов. Признаки строятся на основании цвета, яркости и текстуры окрестности предполагаемой особой точки. Они могут оказаться где угодно: на углах, границах объекта, на ребре предмета и так далее. Чтобы сформировать точки дескриптора, в начале высчитывается магнитуда, а так же ориентация градиента в каждом пикселе, при этом учитываются веса, пропорциональные значению функции плотности нормального распределения и с математическим ожиданием в рассматриваемой особой точке и стандартным отклонением. Выделенный дескриптор изменяется так, что минимизировать возможные искажения от освещения. Его применение распространено во многих задачах, добавлено во многие готовые библиотеки. В данной дипломной работе применяется уже готовый алгоритм, находящийся в открытой библиотеке компьютерного зрения OpenCV[3].

3.5 Алгоритм принятия решений

В качестве алгоритма принятия решения в данной дипломной работе используется алгоритм ближайших соседей. Этот алгоритм нам подходит, так как он относится к группе методов, работа которых основывается на хранении информации в памяти для сравнения с новыми элементами. При добавлении новой записи для прогнозирования, появляются отклонения между этой записью и похожими наборами данных, и наиболее похожая идентифицируется. Для этого алгоритм сравнивает новые данные со всеми уже существующими признаками в базе данных. Он подразумевает вычисление квадратов евклидова расстояния между точками сравнения. Этот метод довольно прост и распространен довольно широко для решения задач регрессии и классификации. Поэтому он внесён во многие библиотеки машинного обучения. Воспользуемся возможностью и используем в своей работе одну из таких библиотек.

4 Тестирование системы

4.1 Описание набора данных

Для обучения нейронной сети в дипломной работе воспользуемся базой данных Olivetti¹. Olivette – набор данных лиц людей, сформированный в период с 1992 по 1994 годы в AT & T Laboratories Gambridg.

Взятая база данных включает в себя 400 образов лиц человека. Изображения подобраны и сделаны в разное время, с различными методами освещения, несколькими вариантами эмоций и мимикой лица, а так же с различными деталями искусственного искажения лица (очки, макияж). Изображения получены на темном фоне, с вариантами движения в различные стороны.

Яркость каждого изображения раскладывается до 256 рядов уровней серого цвета. После чего сохраняется как 8-разрядное целое число.

Изображения в наборе данных состоит из фотографий 64х64 пикселей (рисунок 17).



Рисунок 17 – Пример изображения в базе данных Olivette

4.2 Описание метода тестирования

Для выявления качества работы программы используем экспериментальный метод. Он позволит понять насколько программа точна. Используем два варианта выявления ошибок:

- поднесем в кадр изображение лица человека, имеющего доступ в систему и, соответственно, занесенного в базу данных;
- поднесем в кадр изображения лица человека, не имеющего доступа к системе и не занесенного в базу данных.

Правильная работа программы:

- человек есть в базе:

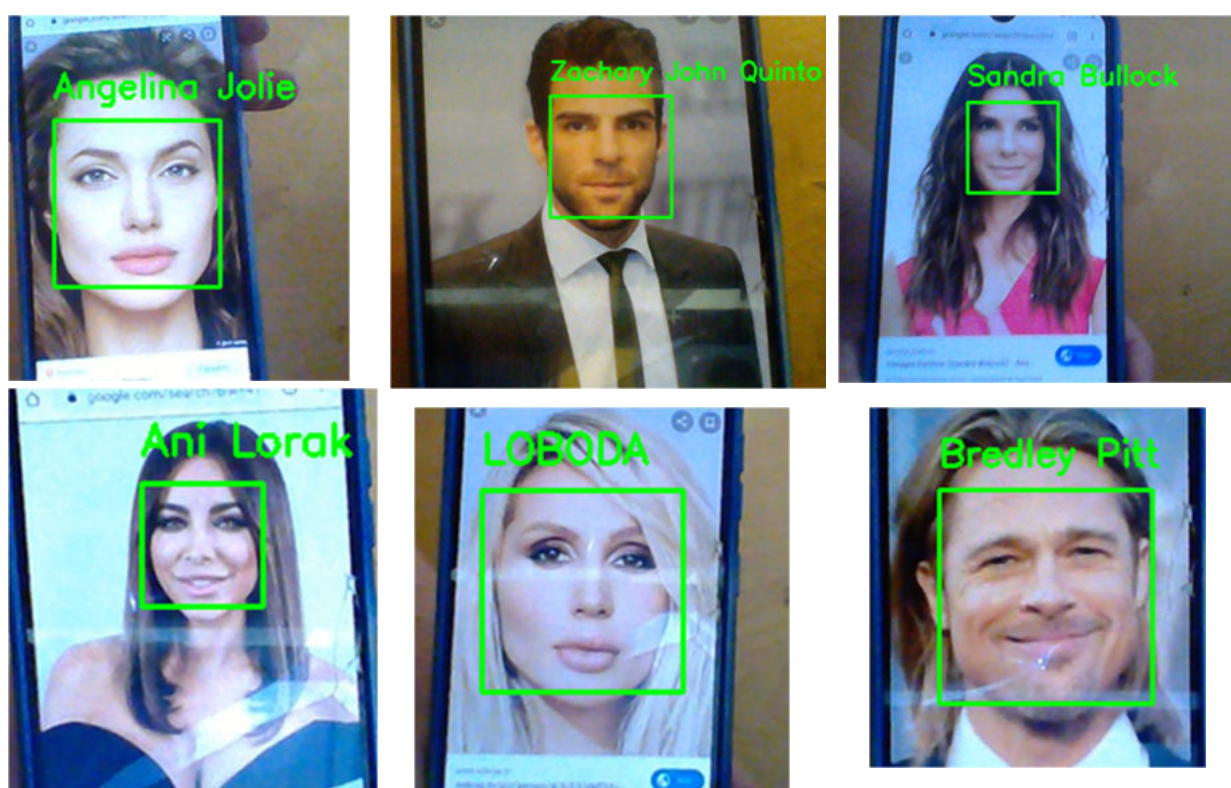


Рисунок 18 – Примеры работы программы при внесенных в базу данных сведениях о лице человека

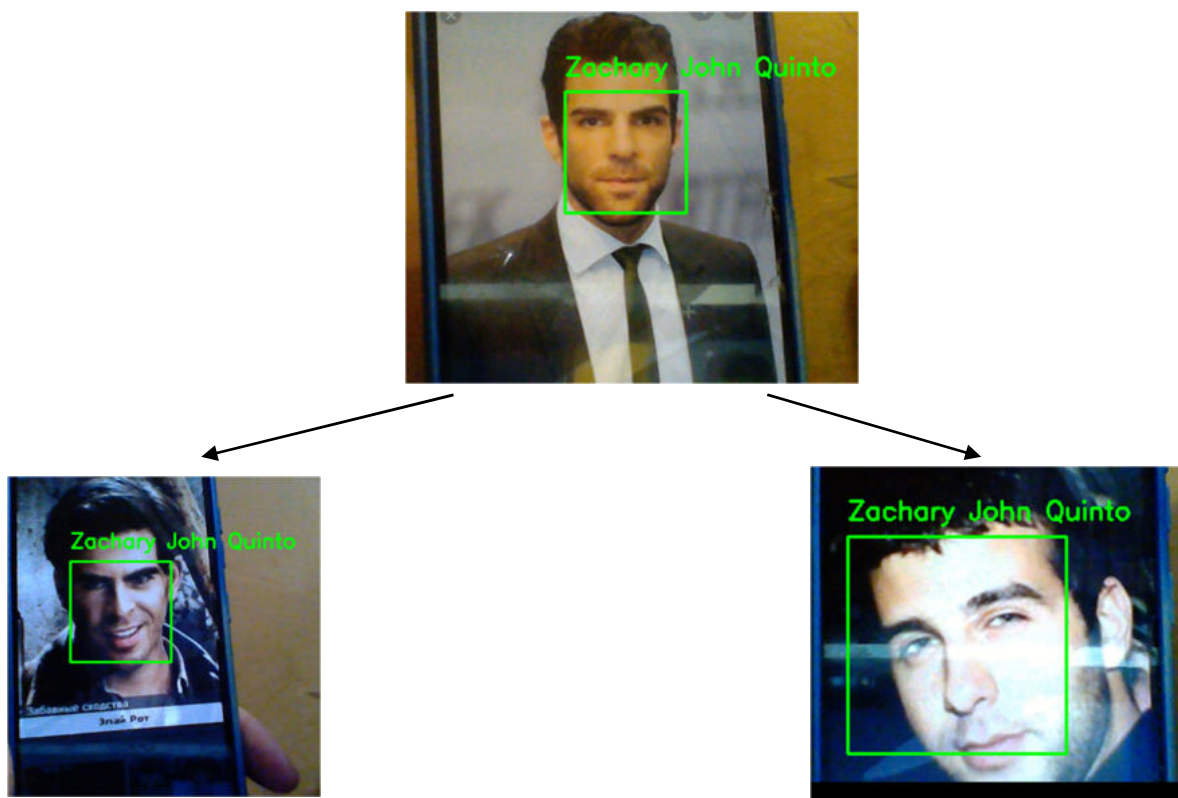
- человек в базу данных не внесен:



Рисунок 19 – Примеры работы программы при не внесенных в базу данных сведениях о лице человека

При тестировании программы был выявлен ряд ошибок. Одной из наиболее часто встречаемой из них является распознавание неизвестного лица как присутствующего в базе. Данный тип ошибки, как показал анализ, возникает из-за схожести черт лиц.

Например, в базе находится Zachary Quinto, при распознавании лица которого происходит ошибка (рисунок 20).



На фото изображён Eli Roth

На фото изображен И. Ургант

Рисунок 20 – Пример ошибки распознавания из-за схожести черт лица

Вторая распространенной ошибкой является не распознавание лица в кадре при том, что черты лица этого человека зарегистрированы в базе.

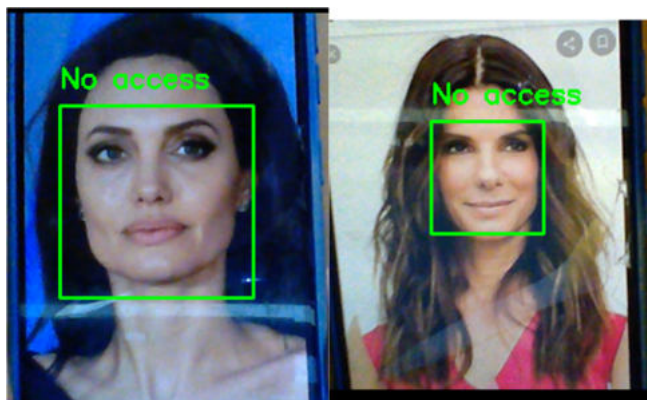


Рисунок 21 – Пример ошибки распознавания черт лица при наличии данных о человеке в базе

Заключение

В написаной квалификационной работе спроектирована и реализована программа распознавания человека по его лицу. Для каждого этапа программы были подобраны алгоритмы решения.

Для этапа нахождения лица в кадре спроектирована сверточная нейронная сеть. В ходе реализации программного обеспечения выбран и описан алгоритм выявления особых точек объекта, а так же алгоритм принятия решений.

В качестве дальнейшего развития созданной программы идентификации людей по лицу можно рассматривать ее использование для установки паролей скрытых папок или приложений.

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

Список литературы

1. Брилюк Д.В., Старовойтов В.В. Распознавание человека по изображению лица нейросетевыми методами. – Минск, 2002
2. Гафаров Ф.М, Галимянов А.Ф Искусственные нейронные сети и приложения: учеб.пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. Ун-та 2018
3. Викторов А.С. Алгоритм детектирования объектов на фотоснимках с низким качеством изображения. Костромской государственный университет 2017
4. Микелуччи У. Прикладное глубокое обучение. Подход к пониманию глубоких нейронных сетей на основе метода кейс: Перс анг - Санкт-Петербург «БХВ-Петербург» 2020
5. Силен Дэви, Мейсман Арно Основы Data Science и Big Data. Python и наука о данных – СПб.: Питер, 2017
6. Барский А.Б. Нейронные сети: распознавание, управление, принятие решений. – М.: Финансы и статистика, 2004
7. Рашид, Тарик., Создаем нейронную сеть.: Пер. с англ. – СПб. : ООО «Альфа-книга», 2017

Приложение

import numpy as np

```
def mix_roster(images_roster, verity_roster, dir_roster):  
    roster_custody = zip(images_roster, verity_roster, dir_roster) # zip  
    соединяет детали картежа  
    roster_custody = list(roster_custody)  
    np.random.shuffle(roster_custody)  
    images_roster, verity_roster, dir_roster = zip(*roster_custody) #  
    return images_roster, verity_roster, dir_roster
```

```
def build_shft_ind (scale_shaft, focus_w_l):  
    coord = []  
    for i in range(-focus_w_l, scale_shaft-focus_w_v):  
        coord.append(i)  
    return coord
```

```
def build_ind(scale_shart, focus_w_v):  
    # Подсчет координат  
    coord_n = build_shft_ind(scale_shaft = scale_shaft[0], focus_w_v=  
focus_w_v[0])  
    coord_m = build_shft_ind(scale_shaft = scale_shaft[1], focus_w_l=  
focus_w_l[1])  
    return coord_n, coord_m
```

```
def convolution_innings_a_v(b_v_sans_l, w_v, c_p):  
    ind_n, ind_m = build_ind(scale_shaft=w_v.form, focus_w_v=c_p  
['focus_w_v'])
```

```

step = c_p ['step']

#Переменная выхода пополняется новыми элементами
a_v = np.zeros((1,1))
if c_p ['convolution']:
    k = 1 # КОНВОЛЮЦИЯ
else:
    k = -1 # корреляция
for i in range(b_v_sans_1.form[0]):
    for j in range(b_v_sans_1.form[1]):
        show = np.zeros([b_v_sans_1. form [0], b_v_ sans_1.form[1]])
        outcome = 0
        cell_be = False
        for n in ind_n:
            for m in ind_m:
                # проверка выхода границ
                if i*step - k*n >= 0 and j*step - k*m >= 0 \
                    and i*step - k*n < y_l_ sans_1. form[0] and j*step -
k*m < b_v_sans_1. form[1]:
                    outcome += b_v_ sans_1[i*step - k*n][j*step
- k*m] * w_l[ind_n.index(n)][ind_m.index(m)]
                    show[i*stride - g*a][j*stride - g*b] =
w_l[indexes_a.index(a)][indexes_b.index(b)]
                    cell_be = True
            if cell_be:
                if i >= a_v. form[0]:
                    a_v = np.vstack((a_v, np.zeros(x_l. form[1])))
                if j >= a_v. form[1]:
                    a_v = np.hstack((a_v, np.zeros((a_v. form[0],1))))

```

```

        a_v[i][j] = outcome

    return a_v

def maxpool(b_v, c_p):
    ind_n, ind_m = build_ind(scale_shaft=c_p ['window_ form'],
    focus_w_v=c_p['focus_window'])

    step = c_p['step']

    b_v_mp = np.zeros((1,1))

    b_v_mp_to_b_v = np.zeros((1,1), dtype='<U32')

    if c_p['convolution']:
        k = 1

    else:
        k = -1

    for i in range(b_v. form[0]):
        for j in range(b_v. form[1]):
            outcome = -np.inf
            cell_be = False
            for n in ind_n:
                for m in ind_m:
                    if i*step - k*n >= 0 and j*step - k*m >= 0 \
                    and i*step - k*n< b_v.form[0] and j*step - k*m <
b_v.form[1]:
                        if b_v[i*step - k*n][j*step - k*n] > outcome:
                            outcome = b_v[i*step - k*n][j*stride -
k*m]

                            i_round = i*step - k*n
                            j_round = j*step - k*m

                            cell_be = True

            if cell_be:

```

```

        if i >= b_v_mp.form[0]:
            b_v_mp = np.vstack((b_v_mp, np.zeros(b_v_mp.
form[1])))

            b_v_mp_to_b_v =
np.vstack((b_v_mp_to_b_v, np.zeros(b_v_mp_to_b_v.form[1])))

            if j >= b_v_mp. form[1]:
                b_v_mp = np.hstack((b_v_mp,
np.zeros((b_v_mp.form[0],1))))

                b_v_mp_to_b_v = np.hstack((b_v_mp_to_b_v,
np.zeros((b_v_mp_to_b_v.form[0],1))))

                b_v_mp[i][j] = outcome
                b_v_mp_to_b_v[i][j] = str(i_round) + ',' + str(j_round)

    return b_v_mp, b_v_mp_to_b_v

```

```

def maxpool_nurse(b_v, c_p):
    roster_of_b_v_mp = []
    roster_of_b_v_mp_to_y_l = []
    for i in range(len(b_v)):
        b_v_mp, b_v_mp_to_b_v = maxpool(b_v[i], c_p)
        roster_of_b_v_mp.append(b_v_mp)
        roster_of_b_v_mp_to_b_v.append(b_v_mp_to_b_v)
    return roster_of_b_v_mp, roster_of_b_v_mp_to_b_v

```

```

def maxpool_round(Ily_v_mp, b_v_mp_to_b_v, b_v_form):
    roster_of_Ily_v = []
    for i in range(len(Ily_v_mp)):
        Ily_v = np.zeros(b_v_form)
        for f in range(Ily_v_mp[i].form[0]):
            for v in range(Ily_v_mp[i].form[1]):

```



```

        coordinates = b_v_mp_to_b_v[i][f][v]
        coordinate_series = int(coordinates[:coordinates.find(',')])
        coordinate_sed = int(coordinates[coordinates.find(',')+1:])
        Ily_v[coordinate_series][coordinate_sed] =
Ily_v_mp[i][f][v]
        roster_of_Ily_v.append(Ily_v)

    return roster_of_Ily_v

def convolution_round_Ily_v(b_v_sans_1, w_v_form, Ily_1, c_p):
    ind_n, ind_m = build_ind(scale_sheft=w_v_form,
focus_w_v=c_p['focus_w_v'])
    step = c_p['step']
    Ily_v = np.zeros((w_v_form[0], w_v_form[1]))
    if c_p['convolution']:
        k = 1
    else:
        k = -1
    for n in ind_n:
        for m in ind_m:
            show = np.zeros([b_v_sans_1.form[0], b_v_sans_1.form[1]])
            outcome = 0
            for i in range(Ily_v.form[0]):
                for j in range(Ily_v.form[1]):
                    if i*step - k*n >= 0 and j*step - k*n >= 0 \
                        and i*step - k*n < b_v_sans_1.form[0] and j*step -
k*m < b_v_sans_1.form[1]:
                        outcome += b_v_sans_v[i*step - k*n][j*step
- k*m] * Ily_v[i][j]

```

```

        show [i*step - k*n][j*step - k*m] =
Ily_v[i][j]

        Ily_v[ind_n.index(n)][ind_m.index(m)] = outcome

    return Ily_v

def convolution_round_Ily_v_sans_v(Ily_v, w_v, b_v_sans_1_form, c_p):
    ind_n, ind_m = build_ind (skale_shaft=w_v.form,
focus_w_v=c_p['focus_w_v'])

    step = c_p ['step']

    Ily_v_sans_1 = np.zeros((b_v_sans_v_form[0], b_v_sans_v_form[1]))

    if c_p['convolution']:
        k = 1
    else:
        k = -1

    for i in range(Ily_v_sans_1.form[0]):
        for j in range(Ily_v_sans_1.form[1]):
            outcome = 0

            show = np.zeros([Ily_v.form[0], Ily_v.form[1]])

            for i_a_l in range(Ily_v.form[0]):
                for j_a_v in range(Ily_v.form[0]):
                    n = k*i_a_v*step - k*i
                    m = k*j_a_v*step - k*j
                    if n in ind_n and m in ind_m:
                        n = ind_n.index(n)
                        m = ind_m.index(m)
                        outcome += Ily_v[i_a_v][j_a_v] *

w_v[n][m]

        show[i_a_v][j_a_v] = w_v[n][m]

```

```

        Ily_v_sans_1[i][j] = outcome

    return Ily_v_sans_1

def conv_heft_therein(form, nomber, heft_title, dir_npy):
    try:
        heft_dile = np.load(dir_npy).item().get(heft_title)
        print('веса для', heft_title, 'подгружены', len(heft.dil)*heft.dil[0].size)
    except:
        heft.dil = []
        for i in range(quantity):
            heft.dil.append(2 * np.random.random(form) - 1)
        print('веса для', heft.titl, 'созданы', len(heft.dil)* heft.dil [0].size)
    return heft.dil

def fc_heft_therein(form, heft.titl, dir_npy):
    try:
        heft_dile = np.load(dir_npy).item().get(heft.titl)
        print('веса для', heft.titl, 'подгружены', heft_dile.size)
    except:
        heft.dile = 2 * np.random.random(form) - 1
        print('веса для', heft.titl, 'созданы', heft_dile.size)
    return heft.dile

def receive_begin_move(dir_npy):
    try:
        begin_move = np.load(dir_npy).item().get('move') + 1
    except:
        begin_move = 0

```

```

        return begin_move

def receive_keep(roster_title, dir_npy):
    try:
        roster_keep = np.load(dir_npy).item().get(list_name)
    except:
        roster_keep = []
    return roster_keep

def convolution_nurse(b_v_sans_1, w_v, w_v_title, w_form_v, m_v, m_v_title,
trait_card, operate_fn, dir_npy, c_p):
    a_v = []
    b_v = []
    if not w_v:
        w_v = conv_helt_init(form=w_form_v,
number=trait_card*len(b_v_sans_v), heft_title=w_v_title, dir_npy=dir_npy)
    for i in range(len(b_v_sans_1)): # для всех b_v_sans_1
        for j in range(i* trait_card, (i + 1)* trait_card):
            a_v.append(convolution_marse_a_v(b_v_sans_1=b_v_sans_1[i], w_v=w_v[j], c_p=c_p))
    if len(m_v) == 0:
        m_v = conv_helt_init(form=(1,1), number= trait_card,
heft_title=m_v_title, dir_npy=dir_npy)
    a_v_final = []
    for i in range(trait_card):
        a_v_final.append(0)
        for j in range(len(y_1_sans_1)):
            n_v_final[-1] += n_v[j* trait_card + i]
        a_v_final[-1] += m_v[len(a_v_final)-1]

```

```

        b_v.append(activation_fn(a_v_final[-1], fn_title=operate_fn,
feed=True))

    return b_v, w_v, m_v

def fc_generation(b_v_sans_l, w_v, w_v_title, m_v, m_v_titele, neurons, operate
_fn, dir_npy):

    if w_v.scale == 0:

        w_v = fc_heft_init(form=(b_v_sans_l.form[1], neurons),
helft_title=w_v_title, dir_npy=dir_npy)

        m_v = fc_helft_init(form=(1, neurons), trait_title=m_v_title,
dir_npy=dir_npy)

        a_v = np.dot(b_v_sans_l, w_v) + m_v

        b_v = fn_activation(x_l, fn_name=act_fn, narse=True)

    return y_l, w_l, b_l

```

```

def fn_activation(matix, fn_title, narse):

    exit_matix = np.copy(matix)

    if feed:

        if fn_name == 'sigmoid':

            exit_matix = 1 / (1+np.exp(-exit_matix))

        if fn_title == 'relu':

            exit_matix[exit_matix<0] = 0

        if fn_title == 'softmax':

            exit_matix = np.exp(exit_matix) / np.exp(exit_matix).sum()

    else:

        if fn_title == 'sigmoid':

            exit_matix = exit_matix * (1 - exit_matix)

        if fn_title == 'relu'

            exit_matix[exit_matix>0] = 1

```

```

if fn_title == 'softmax':
    inlet_matix = np.copy(matix)
    exit_matix = np.zeros((matix.form[1], matix.form[1]))
    for i in range(exit_matix.form[1]):
        for j in range(exit_matix.form[1]):
            if i==j:
                exit_matix[i][i] = inlet_matix[0][i]*(1 -
inlet_matix[0][i])
            else:
                exit_matix[i][j] = -
inlet_matix[0][i]*inlet_matix[0][j]
    return exit_matix

def loss_fn(b_land_sooth, b_foretold, feed):
    if feed:
        # error_matix = (1/2)*( b_land_sooth - b_foretold)**2
        error_matix = - b_land_sooth * np.log(b_foretold)
    else:
        # error_matix = b_foretold - b_land_sooth
        error_matix = - (b_land_sooth / b_foretold)
    return error_matix

def matrix2vec(matrix):
    vec = np.array([[]])
    for i in range(len(matrix)):
        vary_matrix = np.reshape(matrix[i], (1,
matrix[i].form[0]*matrix[i].form[1]))
        vec = np.hstack((vec, vary_matrix))

```

```

return vec

def vec2matrix(vec, matrix_form):
    matrices = []
    matrix_scale = matrix_form[0]*matrix_form[1]
    for i in range(0, vec.scale, matrix_scale):
        matrix = np.reshape(vector[0][i:i+matrix_scale], matrix_form)
        matrices.append(matrix)
    return matrices

def fc_backpropagation(b_v_sans_v, lly_v, b_v, w_v, m_v, operat_fn, alpha):
    # вычисление lly_v, то есть backprop через функцию активации
    if act_fn == 'softmax':
        lly_v = np.dot(lly_v, activation_fn(y_l, fn_title=operat_fn,
narse=False))
    else:
        lly_v = lly_v * activation_fn(y_l, fn_name=act_fn, narse=False)
    # вычисление частных производных
    lly_v = np.dot(b_v_sans_l.T, lly_v)
    lly_v = lly_v
    lly_v_sans_v = np.dot(lly_v, w_v.T)
    w_v = w_v - alpha * lly_v
    m_v = m_v - alpha * lly_v
    return lly_v_sans_v, w_v, b_v

def convolution_backpropagation(y_l_sans_l, y_l, w_l, b_l, lly_v, feature_maps,
act_fn, alpha, conv_params):
    list_of_lly_v_sans_l = []

```

```

list_of_Ily_v = []

for i in range(len(y_l)):

    list_of_Ily_v.append(Ily_v [i] * activation_fn(y_l[i],
fn_title=operat_fn, feed=False))

    Ily_v = list_of_Ily_v [-1].sum()

    b_l[i] = b_l[i] - alpha * Ily_v

for i in range(len(y_l_sans_1)):

    Ily_v_sans_1 = 0

    k = 0

    for j in range(i*feature_maps, (i + 1)*feature_maps):

        Ily_v = convolution_back_Ily_v(
            y_l_sans_1=y_l_sans_1[i],
            w_l_shape=w_l[j].shape,
            Ily_v=list_of_Ily_v [k],

        Ily_v_sans_1 += convolution_back_Ily_v_sans_1(
            Ily_v=list_of_Ily_v [k],
            w_l=w_l[j],
            y_l_sans_1_form=y_l_sans_1[i].form,
            conv_params=conv_params)

        w_l[j] = w_l[j] - alpha * Ily_v

        k += 1

    list_of_Ily_v_sans_1.append(Ily_v_sans_1)

return list_of_Ily_v_sans_1, w_l, b_l

import numpy as np

import matplotlib.pyplot as plt

import PIL

```



```

import os
import model
from sklearn.datasets import fetch_olivetti_faces
import tensorflow as tf
import opencv
import cv2

boat_type = False
image_custody = [] # хранение изображений
truth_custody = []
dir_cus data,
goal = datasets.fetch_olivetti_faces()
tody = [C:\Users\79601\Pictures\Saved Pictures] # путь к изображениям

#функция считывания изображения с камеры
def read_from_camera()
    cap = cv2.VideoCapture(0)
    while(True):
        ret, frame = cap.read()
        frame = cv2.flip(frame, -1)
        pic_from_camera = cv2.cvtColor(frame) #захват статичного фрейма
с камеры
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()

```

					ВКР-НГТУ-ИРИТ-ВСТ-16В1-№151974-2020	Лист
Изм.	Лист	№ докум.	Подпись	Дата		44

```

return pic_from_camera
for door_image in today:
    image_custody.append(np.reshape(door_image, (64, 64)))

# первый и последний шаги
if train_model:
    start_step = model.get_start_step(weight_dir)
    end_step = len(image_storage)
    len_dataset = 1000
else:
    start_step = 0
    end_step = len(image_storage)
    len_dataset = 1000

image_storage, truth_storage, dir_storage = model.shuffle_list(image_storage,
truth_storage, dir_storage)

model_settings = {
    'learning_rate':0.01,
    'conv_form_1':(3,3),
    'conv_form_2':(3,3),
    'maxpool_form_1':(2,2),
    'conv_trait_1':5,
    'conv_trait_2':20,
    'conv_step_1':2,
    'conv_step_2':1,
    'maxpool_step_1':2,
    'fc_neurons_1':2000,
    'conv_fn_1':'relu',

```

```

'conv_fn_2':'sigmoid',
'fc_fn_1':'sigmoid',
'fc_fn_2':'softmax',
'conv_conv_1':False
'conv_conv_2':False,
'maxpool_conv_1':False,
'conv_focus_1':(0,0),
'conv_focus_2':(1,1),
'maxpool_focus_1':(0,0)
}

```

```

conv_w_1 = []
conv_m_1 = []
conv_w_2 = []
conv_m_2 = []
fc_w_1 = np.array([[]])
fc_m_1 = np.array([[]])
fc_w_2 = np.array([[]])
fc_m_2 = np.array([[]])

```

```

if train_model:
    loss_change = model.get_saved('loss_change', weight_dir)
    accuracy_change = model.get_saved('accuracy_change', weight_dir)
else:
    loss_change = []
    accuracy_change = []
for step in range(start_step, end_step):

```

```

# извлечение изображения из хранилища

image_id = step%len(image_storage) # на каждом шаге обновляются веса
для одного изображения

print ('до вывода результатов',
str(round((step%len_dataset)*100/len_dataset)) + '%', end="\r")

input_image = [image_storage[image_id]] # здесь лист, так как
convolution_feed на вход принимает лист, состоящий из feature maps

y_true = truth_storage[image_id]

conv_y_1, conv_w_1, conv_b_1 = model.convolution_feed(
    y_1_minus_1=input_image,
    w_1=conv_w_1,
    w_1_name='conv_w_1',
    w_shape_1=model_settings['conv_shape_1'],
    b_1=conv_b_1,
    b_1_name='conv_b_1',
    feature_maps=model_settings['conv_feature_1'],
    act_fn=model_settings['conv_fn_1'],
    dir_npy=weight_dir,
    conv_params={
        'convolution':model_settings['conv_conv_1'],
        'stride':model_settings['conv_stride_1'],
        'center_w_1':model_settings['conv_center_1']
    }
)

conv_y_1_mp, conv_y_1_mp_to_conv_y_1 = model.maxpool_feed(
    y_1=conv_y_1,
    conv_params={

```

```

        'window_shape':model_settings['maxpool_shape_1'],
        'convolution':model_settings['maxpool_conv_1'],
        'stride':model_settings['maxpool_stride_1'],
        'center_window':model_settings['maxpool_center_1']
    }
)

```

```

conv_y_2, conv_w_2, conv_b_2 = model.convolution_feed(
    y_1_minus_1=conv_y_1_mp,
    w_1=conv_w_2,
    w_1_name='conv_w_2',
    w_shape_1=model_settings['conv_shape_2'],
    b_1=conv_b_2,
    b_1_name='conv_b_2',
    feature_maps=model_settings['conv_feature_2'],
    act_fn=model_settings['conv_fn_2'],
    dir_npy=weight_dir,
    conv_params={
        'convolution':model_settings['conv_conv_2'],
        'stride':model_settings['conv_stride_2'],
        'center_w_1':model_settings['conv_center_2']
    }
)

```

```

conv_y_2_vect = model.matrix2vector_tf(conv_b_2)
fc_b_1, fc_w_1, fc_y_1 = model.fc_multiplication(
    b_1_minus_1=conv_y_2_vect,

```

```

w_l=fc_w_1,
w_l_name='fc_w_1',

b_l=fc_b_1,
b_l_name='fc_b_1',
neurons=model_settings['fc_neurons_1'],
act_fn=model_settings['fc_fn_1'],
dir_npy=weight_dir
)

fc_y_2, fc_w_2, fc_b_2 = model.fc_multiplication(
    y_1_minus_1=fc_y_1,
    w_l=fc_w_2,
    w_l_name='fc_w_2',
    b_l=fc_b_2,
    b_l_name='fc_b_2',
    neurons=len(y_true),
    act_fn=model_settings['fc_fn_2'],
    dir_npy=weight_dir
)

# ошибка модели
fc_error = model.loss_fn(y_true, fc_y_2, feed=True)
loss_change.append(fc_error.sum())
accuracy_change.append(y_true.argmax() == fc_y_2.argmax())

if train_model:
    lly_b_2 = model.loss_fn(y_true, fc_y_2, feed=False)
    lly_b_v, fc_w_2, fc_b_2 = model.fc_backpropagation(
        b_v_stans_v=fc_y_1,

```

```

        lly_l=lly_b_2,
        y_l=fc_b_2,
        w_l=fc_w_2,
        b_l=fc_b_2,
        operat_fn=model_settings['fc_fn_2'],
        alpha=model_settings['learning_rate']
    )
    # backprop через первый слой fc-сети
    lly_b_0, fc_w_1, fc_b_1 = model.fc_backpropagation(
        y_l_minus_1=conv_y_2_vect,
        dEdy_l=dEdfc_y_1,
        y_l=fc_y_1,
        w_l=fc_w_1,
        b_l=fc_b_1,
        act_fn=model_settings['fc_fn_1'],
        alpha=model_settings['learning_rate']
    )
    # конвертация полученного вектора в feature maps
    dEdconv_y_2 = model.vector2matrix_tf(
        vector=dEdfc_y_0,
        matrix_shape=conv_y_2[0].shape # размерность одной из
матриц feature map
    )
    # backprop через второй слой конволюции
    dEdconv_y_1_mp, conv_w_2, conv_b_2 =
model.convolution_backpropagation(
        y_l_minus_1=conv_y_1_mp, # так как слой макспулинга!

```

```

y_l=conv_y_2,
w_l=conv_w_2,
b_l=conv_b_2,
dEdy_l=dEdconv_y_2,
feature_maps=model_settings['conv_feature_2'],
act_fn=model_settings['conv_fn_2'],
alpha=model_settings['learning_rate'],
conv_params={
    'convolution':model_settings['conv_conv_2'],
    'stride':model_settings['conv_stride_2'],
    'center_w_l':model_settings['conv_center_2']
}
)
# backprop через слой макспулинга
dEdconv_y_1 = model.maxpool_back(
    dEdy_l_mp=dEdconv_y_1_mp,
    y_l_mp_to_y_l=conv_y_1_mp_to_conv_y_1,
    y_l_shape=conv_y_1[0].shape
)
# backprop через первый слой конволюции
dEdconv_y_0, conv_w_1, conv_b_1 =
model.convolution_backpropagation(
    b_v_snas_v=inlet_image,
    b_v=conv_b_v,
    w_v=conv_w_v,
    b_v=conv_b_v,
    lly_v=dEdconv_b_v,

```



```

feature_maps=model_settings['conv_feature_1'],
act_fn=model_settings['conv_fn_1'],
alpha=model_settings['learning_rate'],
conv_params={
    'convolution':model_settings['conv_conv_1'],
    'stride':model_settings['conv_stride_1'],
    'center_w_1':model_settings['conv_center_1']
}
)

if len(loss_change)%len_dataset == 0:
    print('шаг:', len(loss_change), 'loss:', sum(loss_change[-
len_dataset:])/len_dataset, 'accuracy:', sum(accuracy_change[-
len_dataset:])/len_dataset)

# сохранение весов
if train_model:
    np.save(weight_dir, {
        'step':step,
        'loss_change':loss_change,
        'accuracy_change':accuracy_change,
        'conv_w_1':conv_w_1,
        'conv_b_1':conv_b_1,
        'conv_w_2':conv_w_2,
        'conv_b_2':conv_b_2,
        'fc_w_1':fc_w_1,
        'fc_b_1':fc_b_1,
        'fc_w_2':fc_w_2,
        'fc_b_2':fc_b_2
    })

```

```
}
)
```

Сеть

```
For step in range(start_step, end_step)
```

```
Image_id = step%len(image_storage)
```

```
Print('до сохранения весов', str(round((step%len_dataset)*100/len_dataset)) +
input_image = [image_storage[image_id]]
```

```
b-true = truth_storage[image_id]
```

```
conv_b_1, conv_w_1, conv_m_1 = modul.convolution_feed
```

```
conv_b_1_mp, conv_w_1_mp, conv_m_1_mp = model.maxpool_feed
```

```
conv_b_2, conv_w_2, conv_m_2 = model.convolution_feed
```

```
conv_b_2_vect = model.matrix2vector_tf(conv_b_2)
```

```
fc_b_1, fc_w_1, fc_m_1 = model.fc_multiplication
```

```
fc_b_2, fc_w_2, fc_m_2 = model.fc_multiplication
```

```
fc_error = model.loss_fn(y_true, fc_y_2, feed=True)
```

```
Ily_b_2 = model.loss_fn(y_true, fc_y_2, feed=False)
```

```
Ily_b_1 = fc_w_2, fs_m_2 = module.fc_backpropagation
```

```
Ily_b_0 = fc_w_1, fs_m_1 = module.fc_backpropagation
```

```
Ily_b_2 = fc_w_2, fs_m_2 = module.vector2matrix_tf
```

```
Ily_b_1_mp, conv_w_2, conv_m_2 = model.convolution_backpropagation
```

```
Ily_b_1 = model.maxpool_back
```

```
Ily_y_0, conv_w_1, conv_m_1 = model.convolution_backpropagation
```

#функция поиска точек на лице

```
def find_dots_on_face(image)
```

```
    #инициализация SIFT
```

```

sift = cv2.xfeatures2d.SIFT_create()

kp, des = sift.detectAndCompute(image, None)

return kp

#функция сравнения точек лица с базой точек известных лиц
def compare_faces(found_dots)

    base_access = open('base_access', 'r')

    flag = 0

    for line in base_access:

        for i in range(len(line)):

            if line[i] !=(';'):

                e+=line[i]

            else:

                dots_line.append(int(e))

                e=""

        rez_compare = distance.euclidean(dots_line, found_dots)

        if rez_compare < 1000

            flag = 1

    base_access.close()

return flag

#функция добавления точек лица в базу известных лиц
def add_new_face(found_dots)

    base_access = open('base_access', 'a')

    write_string = ""

    for i in found_dots:

        write_string = write_string + str(i) + ";"

```

```

base_access.write(write_string + '\n')

base_access.close()

print("Лицо записано")

def main():

    pic_from_camera = read_from_camera()    # вызов функции считывания
    изображения с камеры

    b,g,r = cv2.split(pic_from_camera)    # разложение фрейма на
    составляющие (RGB)

    my_network = create_network()    # создание нейронной сети

    my_network = train_network(my_network) # вызов функции обучения
    нейронной сети

    pic_with_face = my_face_recognition(model, r, g, b)    # вызов функции
    поиска лица на фото

    found_dots = np.array(find_dots_on_face(pic_with_face))    # вызов
    функции поиска точек на лице

    print("Сделайте выбор: 1 - запрос на доступ, 2 - добавление нового лица
    в базу")

    choise = input()

    if choise = 1:

        rez_access = compare_faces(found_dots)    # вызов функции
        сравнения точек лица с базой точек известных лиц

        if rez_access = 1:

            print("Доступ разрешен")

        else:

            print("Доступ запрещен")

    if choise = 2:

        add_new_face(found_dots)    # вызов функции добавления точек
        лица в базу лиц

```