МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА
Рецензент
Ведущий программист
ООО «ВОРТЕКСКОД»
_____ П.А. Михайлов
“___” _____2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___” _____ 2020 г.

Разработка приложения для распознавания и идентификации человека по биометрическим данным лица с использованием нейросетевых технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.115-103.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____К.Ю. Никольская

Автор работы,
студент группы КЭ-401
_____И.В. Напольских

Ученый секретарь
(нормоконтролер)
_____И.Д. Володченко
“___” _____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Напольских Илье Викторовичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. Тема работы (утверждена приказом ректора от 24.04.2020 № 627)
Разработка приложения для распознавания и идентификации человека по биометрическим данным лица с использованием нейросетевых технологий.
2. Срок сдачи студентом законченной работы: 07.06.2020.
3. Исходные данные к работе
3.1. Хайкин С., «Нейронные сети: полный курс, 2-е издание», 2006.
3.2. Koch G., Zemel R., Salakhutdinov R. Siamese Neural Networks for One Shot Image Learning // ICML Deep Learning Workshop. 2015.
4. Перечень подлежащих разработке вопросов
4.1. Провести обзор аналогов и научной литературы.
4.2. Разработать алгоритм для поиска лиц на изображении.
4.3. Разработать архитектуру нейронной сети для идентификации человека.
4.4. Разработать пользовательский интерфейс.
4.5. Разработать систему хранения записей.
4.6. Провести тестирование приложения.
5. Дата выдачи задания: 09.02.2020.

Научный руководитель
ст. преподаватель кафедры СП К.Ю. Никольская

Задание принял к исполнению И.В. Напольских

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Распознавание и классификация являются одними из основных задач машинного обучения с широкой сферой применения: начиная от определения рукописных цифр и заканчивая распознаванием объектов на видеозаписи и диагноза по снимкам МРТ [1].

Одним из применений нейросетевых технологий на настоящий момент является распознавание лиц на фотографии или в видео. Например, такие корпорации как Фейсбук в настоящее время используют нейросетевые технологии для задач вроде идентификации друзей пользователя на по их лицам на совместных фотографиях [2].

В данной работе будет проведена разработка системы для распознавания лиц по фотографиям или видеофайлам, имеющей относительно небольшие размеры и работающей с достаточно высокой точностью.

ЦЕЛЬ РАБОТЫ

Целью данной работы является разработка системы, позволяющей распознавать лицо человека на видеофайлах в реальном времени, и имеющей достаточно высокую точность – больше 90%.

Для достижения данной цели необходимо:

1) провести анализ имеющихся решений для распознавания лица;
2) реализовать алгоритм для поиска лиц на изображении;
3) реализовать архитектуру нейронной сети для идентификации человека;
4) реализовать пользовательский интерфейс;
5) реализовать систему хранения записей о личностях;
6) протестировать приложение для распознавания лица.

СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из глоссария, введения, 5 глав, заключения и библиографии. Объем работы составляет 46 страниц, объем библиографии – 35 источников.

В первой главе приводятся теоретические сведения предметной области, а также осуществляется обзор существующих подходов к распознаванию лиц.

Во второй главе представлены этапы предобработки видеофайла, а также приводятся теоретические сведения о искусственных нейронных сетях и их разновидностях топологий, применяющихся для решения поставленной задачи.

В третьей главе описывается топология нейронной сети, а также архитектура настольного приложения.

В четвертой главе представлена программная реализация алгоритма предобработки и постобработки входных данных, алгоритма поиска лиц, нейронной сети, идентификации человека, системы хранения записей и пользовательского интерфейса.

В пятой главе приведены результаты тестирования приложения и нейронной сети.

В заключении описаны полученные в ходе выполнения работы результаты.

ГЛОССАРИЙ

Нейронная сеть – математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [3].

Сверточная нейронная сеть (Convolutional Neural Network, CNN) – архитектура нейронной сети, в настоящий момент являющаяся стандартом для решения задачи распознавания, реализующая процесс «свертки» - постепенного уменьшения изображения и получения таким образом его особенностей, используемых в дальнейшем для распознавания [4].

Геометрический коэффициент (Cosine distance) [5] – бинарная мера сходства, предложенная японским биологом Акирой Отиаи в 1957 году, в дальнейшем обобщенная и нашедшая применение в разнообразных приложениях и за рамками биологии. Представляет собой косинус угла между точками пространства.

HOG (Histogram of Oriented Gradients) [6] – алгоритм, преобразовывающий изображение в градиентную карту и классифицирующий его.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор аналогичных решений

В настоящее время существуют различные решения для распознавания лиц на фотографиях и видео, начиная от распознавания лица, встроенного в телефон и заканчивая промышленными системами, в том числе работающих в реальном времени. Далее приведен обзор некоторых аналогов.

Pictriev [7]

Поисковый сервис с функцией распознания лиц, предоставляющий основную информацию о человеке на фото, сканируя лицо на фотографии. Также сервис показывает знаменитостей, похожих на лицо на загруженной фотографии.

При загрузке фотографии сервис определил, что на картинке женщина с вероятностью 81%, при этом, не указав ее саму как похожего на себя двойника рисунке 1.

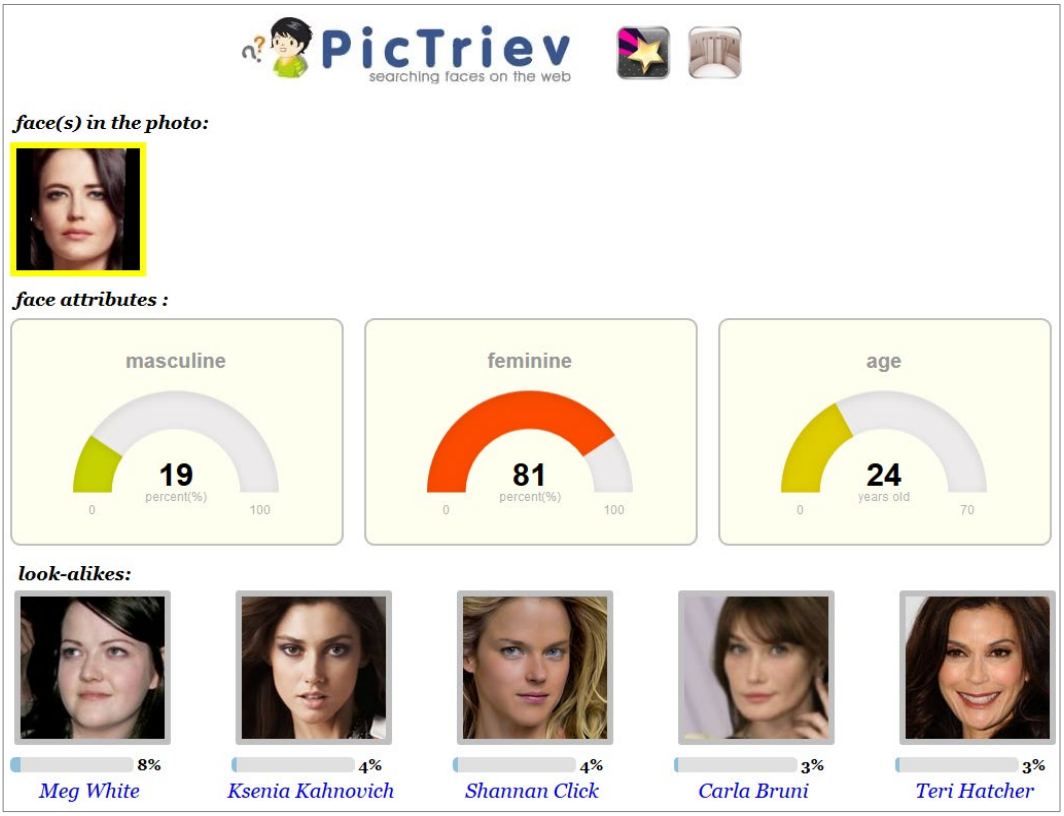

Рис. 1. Результат работы системы PicTriev на первой фотографии

При загрузке другой фотографии сервис распознал ее как женщину с вероятностью 96%, однако в списке двойников указал, что она похожа на саму себя с вероятностью 41% (рисунок 2).

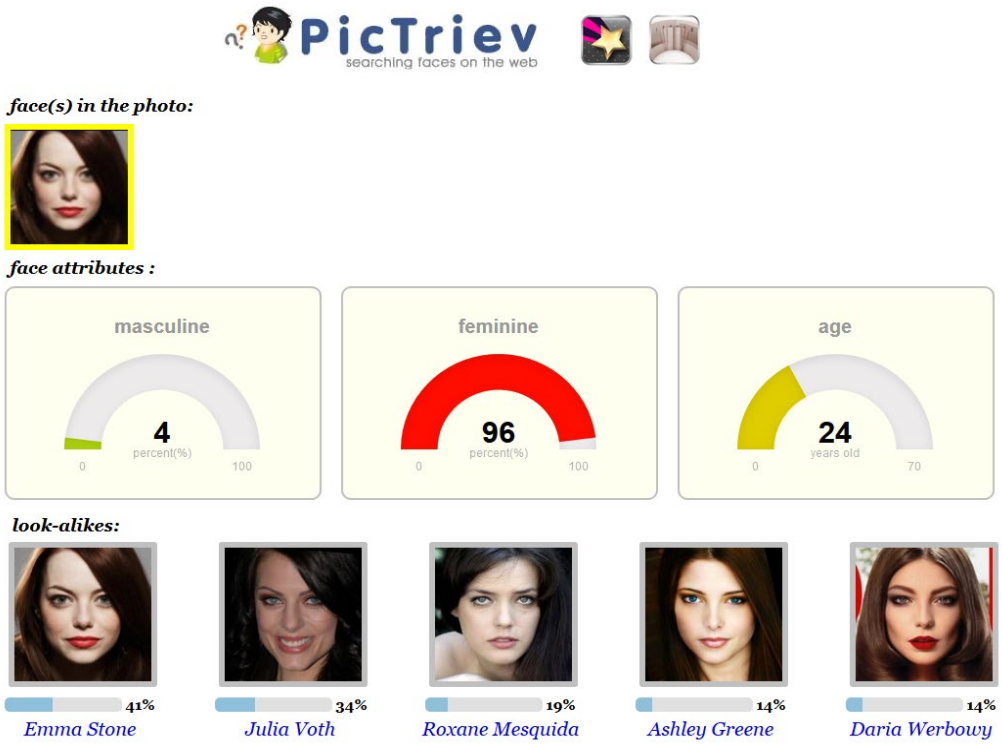

Рис. 2. Результат работы системы PicTriev на второй фотографии

При дальнейшей работе с сервисом было выявлено, что с высокой вероятностью он распознает только фотографии лиц, которые уже были загружены в него. При работе с произвольной фотографией результаты работы, как правило, невысоки – сервис может либо указать возраст с большой разницей, либо не найти в списке похожих знаменитость, фотография которой распознается.

Betaface [8]

Betaface представляет собой профессиональное программное обеспечение для распознавания лиц, ориентированное на медиа-компании, позволяющее им автоматически распознавать лица и находить информацию о них. Чтобы изучить работу этого платного продукта, можно воспользоваться его демоверсией.

Пользователям предлагается загрузить фото на сайт, после чего сервис большой список данных о лице, изображенном на нем. Эти детали включают возраст, расу, выражение лица, наличие бороды и очков, цвет волос и бороды, наличие усов, размер подбородка, цвет и положение глаз, положение, цвет и толщину бровей, длину волос, форму головы, форму и размер рта и носа, зубы и другие более мелкие детали. Несколько результатов работы можно увидеть на рисунке 3 и рисунке 4.

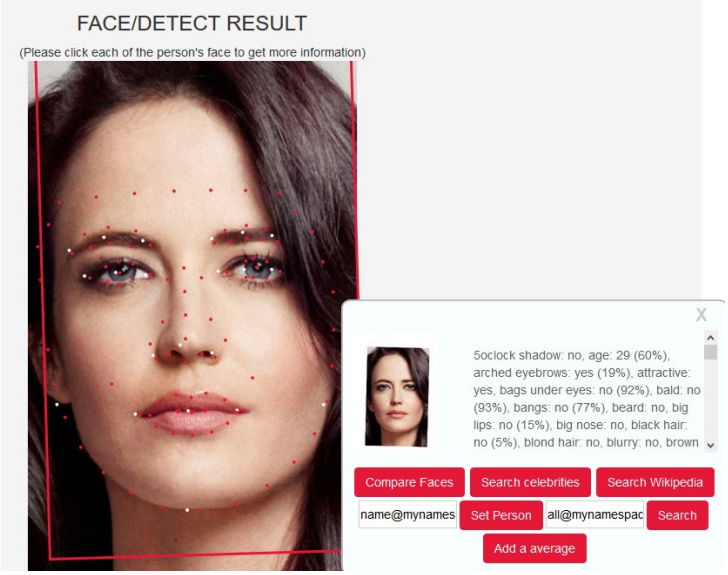

Рис. 3. Результат распознавания первой фотографии системой Betaface

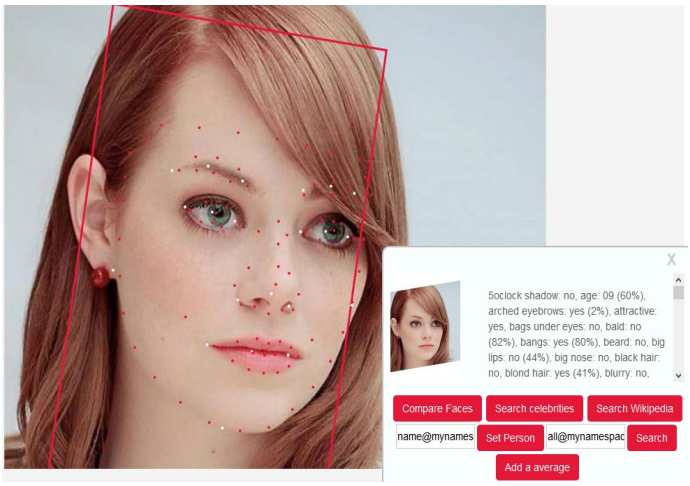

Рис. 4. Результат распознавания второй фотографии системой Betaface

В целом, результат использования этого сервиса лучше, чем предыдущего: данных о лице больше, пол и возраст определяются более кор-

ректно, однако все равно не идеально: при второй фотографии сервис посчитал, что ей 9 лет с вероятностью 60%.

Deepface [9]

Сервис, разработанный в Facebook, позиционируемый авторами как «система, способная распознавать лица лучше человека». Она была обучена на наборе данных Labeled Faces in the Wild и, по словам авторов, способна находить 97,25% лиц на фотографиях.

Сам процесс происходит в несколько этапов. На первом, система накладывает лицо, изображенное на фотографии поверх трехмерной модели человеческой головы. На втором этапе, с помощью многослойной нейронной сети, система дает лицу числовое описание.

Результаты тестирования фотографий приведены на рисунке 5 и рисунке 6. Лица на студийных фотографиях система определяет довольно точно, однако, на непрофессиональных фотографиях система может либо вовсе не найти лица, чего не наблюдается у предыдущих систем, либо ошибочно распознать пол и возраст.

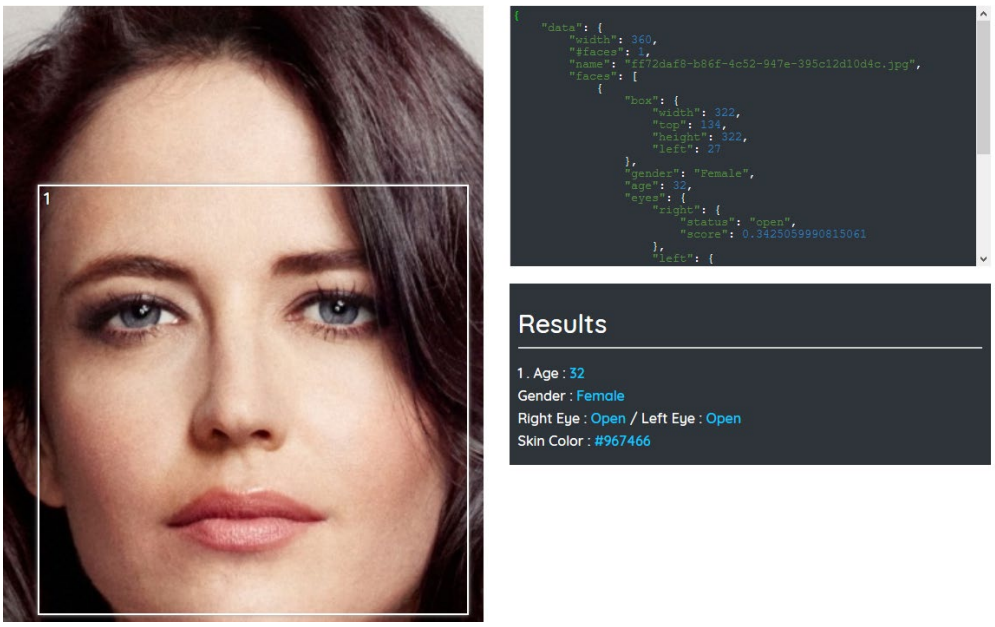

Рис. 5. Результат распознавания первой фотографии системой Deepface

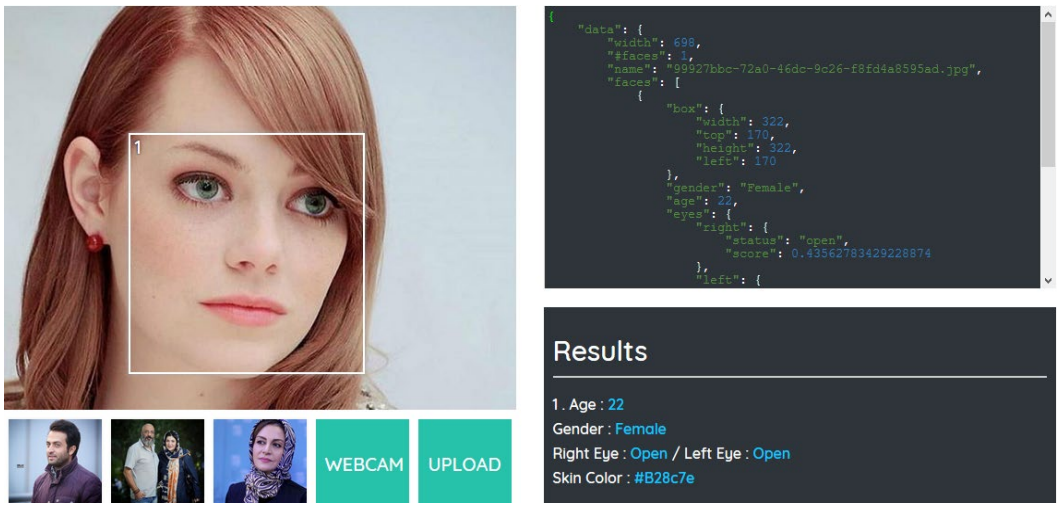

Рис. 6. Результат распознавания второй фотографии системой Deepface

1.2. Обзор готовых решений по созданию нейронных сетей

NumPy

Библиотека языка Python, предназначенная для научных вычислений. Поддерживает большие многомерные массивы и матрицы, а также включает в себя функции, предназначенные для работы с многомерными массивами [10].

Mathplotlib

Библиотека, имеющая в своем составе методы для визуализации вычислений в виде графиков, гистограмм, диаграмм и иных способов графического представления данных, что позволяет наглядно отображать процесс и качество обучения и работы нейронной сети [11].

TensorFlow

Разработанная в Google, библиотека содержит методы и типы для построения нейронных сетей. Библиотека оперирует особым типом данных – тензором, представляющим из себя многомерный массив. Сеть строится в виде графов [12].

Keras

Библиотека, разработанная на языке Python как надстройка над TensorFlow. Повышает уровень абстракции TensorFlow, облегчая процесс построения сети и делая работу со слоями более удобной [13].

OpenCV

Библиотека, содержащая более 2500 алгоритмов машинного зрения. В настоящий момент используется более чем 47 тысячами людей по всему миру. Применяется для широкого списка задач, начиная от навигации складских роботов и заканчивая распознаванием лиц на улицах Японии [14].

Выводы по первой главе

В этой главе была рассмотрена часть аналогов, реализующих похожие функции. Также стоит отметить существование широкого ряда библиотек для создания и обучения нейронных сетей. Использование подобных библиотек может значительно облегчить задачу реализации.

Принимая во внимание все описанные выше факторы, можно сделать вывод, что исследование в данной области является актуальным.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Предварительная обработка входных данных

Для более качественных результатов в обучении и практическом использование нейронной сети, входные данные требуют подготовки, т.е. изменения размера, нормализации и разбиения видеопотока на отдельные кадры.

В данной работе подготовка входного видеопотока перед подачей в модуль поиска лица происходит следующим образом:

1) разделение видеопотока на последовательность кадров;

2) конвертация кадра в другой цветовой формат.

Для первых подготовки изображений используется библиотека OpenCV, включающая в себя алгоритмы для декомпозиции видеопотока, а также конвертации цветового формата. После подготовленные изображения передаются в сверточную нейронную сеть-извлекатель признаков. Перед подачей в сверточную сеть проводится следующая предобработка:

1) изменение размера изображения (приведение к одинаковому для всех кадров);

2) нормализация.

Нормализация представляет собой изменение параметров входных данных, чтобы они находились в пределах от 0 до 1. В текущем примере яркость каждого пикселя находится в пределах от 0 до 255. Для нормализации каждое значение делится на 255, получая в результате значение в искомом пределе.

Также у всех входных изображений меняется и приводится к выбранному стандартному значению размер. Это необходимо для корректной работы сверточной нейронной сети.

2.2. Алгоритм HOG

Предложенный в 2005 году Навнитом Далалем и Биллом Триггсом алгоритм был изначально разработан для распознавания пешеходов на

изображении. К тому моменту HOG имел гораздо более высокую производительность, чем существовавшие способы распознавания людей [6].

Алгоритм использует метод скользящего окна. Основная идея заключается в поочередном сравнении яркости одного пикселя с окружающими и определении, в каком направлении изображение становится темнее. Каждый пиксель заменяется вектором, указывающем в этом направлении, т.н. «градиентом», после чего изображение разбивается на несколько более мелких квадратов (ячеек) размера, к примеру, 8x8, объединенных в блоки небольшого размера (2x2), в которых подсчитывается количество градиентов, указывающих в различные стороны. Ячейки пересекаются между собой.

Полученные данные представляются в виде гистограммы векторов, из-за чего произошло название способа. Затем этот квадрат заменяется градиентами, указывающими в том направлении, в котором указывало большинство градиентов в текущем квадрате. Таким образом, оригинальное изображение предстает в простом виде (векторе, имеющем некоторое количество измерений), показывающем направление изменения яркости в оригинальном изображении. Полученный вектор (дескриптор) подается на вход алгоритма для классификации изображений (авторы использовали SVM).

Нормализация

Перед началом работы алгоритма исходное изображение возможно преобразовать различными способами: нормализовать по гамме, перевести в другое цветовое пространство, привести к черно-белому виду. Однако делать это необязательно: в оригинальной статье отмечено, что в зависимости от вида, преобразование дает либо совсем небольшой прирост производительности – около 1%, либо и вовсе ухудшает ее.

Подсчет градиентов

Оригинальное изображение разбивается на несколько более мелких квадратов.

Авторы испробовали несколько различных способов и пришли к выводу, что наиболее простой путь оказывается наилучшим: центрированные маски в одном измерении, например, (-1, 0, 1), дают наилучший результат. Если увеличить количество измерений, т.е. применить квадратную маску, не центрированную, или, например, фильтр Собеля, то наблюдается падение производительности и точности.

Подсчет направления и длины градиента возможно подсчитать следующим образом: к пикселю применяются два фильтра в одном измерении – горизонтальный (-1, 0, 1) и вертикальный (он же, но повернутый на 90 градусов). Таким образом подсчитывается S_x и S_y. Зная эти значения, длину можно найти по формуле, представленной на рисунке 7.

$$S = \sqrt{S_x^2 + S_y^2}|$$

Рис. 7. Длина градиента

Направление – по формуле, представленной на рисунке 8.

$$\theta = tan^{-1}\left(\frac{S_y}{S_x}\right)$$

Рис. 8. Направление градиента

Если изображение цветное, то градиенты вычисляются для каждого канала.

Группировка векторов

После определения направления и длины векторов выполняется операция группировки векторов (binnig) с определенным шагом, создавая гистограмму векторов. В оригинальной статье векторы с направлением от 0 до 180 градусов разбивались на 9 частей с шагом 20°, т.е. 0° -20°, 20°-40°, 40°-60° и т.д. Векторы с большей длиной имеют больший вес. Таким обра-

зом, все значения направления градиентов в текущем квадрате сокращаются до 9 значений, т.е. можно сказать, что происходит квантизация.

В случае, если направление имеет промежуточное значение, например, 85°, то его значение распределяется в равной степени между ближайшими значениями гистограммы. Опишем это более подробно:

Есть вектор с направлением 85°. Нужно разместить его на гистограмме, на которой нет этого значения, но есть ближайшие к нему: 70° и 90°. Чтобы распределить его, нужно под отдельности подсчитать разницу между ними и вектором. В данном случае разница будет: 85 - 70 = 15 для 70° и 90 – 85 = 5 для 90°. После этого вычисляется соотношение для границ 90° и 70°: 5/20 = ¼ и 15/20 = 3/4. Эти значения прибавляются к значениям на гистограмме – 90 и 70 соответственно. Наглядно это можно увидеть на рисунке 9.

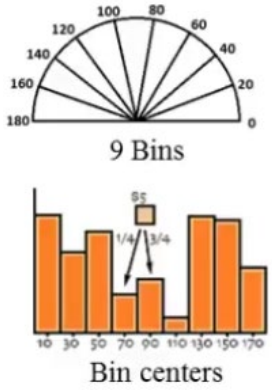

Рис. 9. Распределение на гистограмме

Нормализация градиентов

Нормализация применяется для того, чтобы исключить влияние изменения яркости и контраста в изображении. Таким образом значение каждого пикселя в любом случае остается неизменным.

Допустим, мы нормализуем значение вектора. Для этого нужно знать значение пикселей сверху, снизу, слева и справа. Нормализованное значение будет равно среднеквадратичной сумме разности верхнего и нижнего

пикселя, и разности левого и правого. Наглядно это можно видеть на рисунке 10:

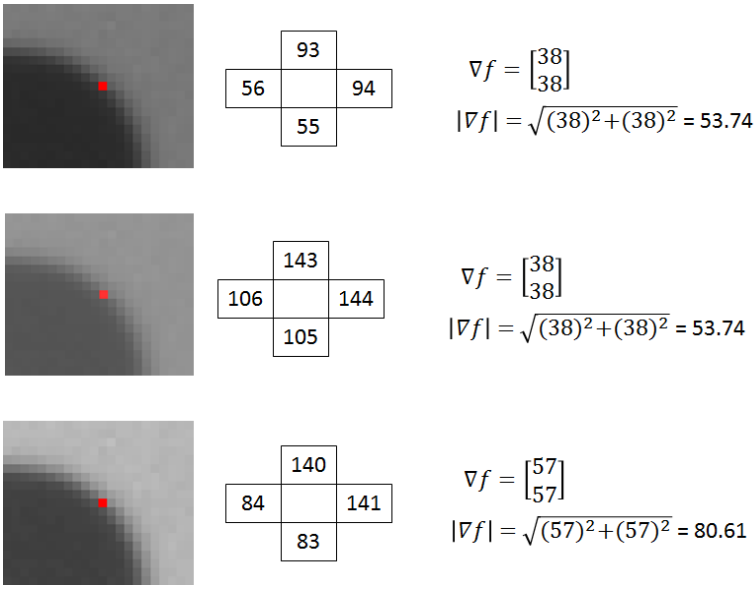

Рис. 10. Нормализация значений

Затем полученное значение делится на длину вектора и получается итоговый нормализованный вектор.

Нормализация блоков

Вместо нормализации каждой гистограммы по отдельности, ячейка разбивается на несколько блоков и нормализуется по всем гистограммам в каждом блоке. В оригинальной статье авторы утверждают, что лучше всего работают блоки 3x3 и 2x2, пересекающие друг друга.

К примеру, есть блок 2x2 (2 – не пиксель, а клетка из 8 пикселей, т.е. блок имеет размер 16x16 пикселей). Объединив все гистограммы в блоке, мы получим вектор с 36 компонентами (4 гистограммы и 9 значений в ней). После деления вектора на его длину мы нормализуем его.

Конечный вектор

Конечный вектор состоит из всех векторов блоков, объединенных в один. Подсчитаем количество измерений в нем: для этого нужно знать размер примененных блоков, их количество и измерение вектора в каждом блоке. Например, если исходное изображение имеет размер 64x128 пиксе-

лей, каждая клетка имеет размер 8x8, каждый блок имеет размер 2x2, т.е. блок имеет размер 16x16 пикселей, то оно будет поделено на 7 * 15 = 105 квадратов. В гистограмме 9 частей, а значит вектор блока имеет 36 измерений. Таким образом, размер конечного вектора будет равен 105 * 36 = 3780 измерений.

Конечный вектор подается на вход алгоритма для классификации изображений, называемом Support Vector Machine (SVM), тренированном для распознавания дескрипторов алгоритма.

2.3. Многослойная нейронная сеть

Искусственная нейронная сеть – математическая модель, схожая по принципу работы с биологической нейронной сетью. Основным элементом является искусственный нейрон, представляющий собой элемент с несколькими входами и одним выходом, выдающим сигнал, значение которого зависит от входных данных [3]. Для нормализации выходного сигнала, т.е. приведение его, например, в предел от 0 до 1 применяется функция активации. Таким образом, искусственная нейронная сеть представляет собой ориентированный взвешенный граф. Наглядно модель нейрона представлена на рисунке 11.

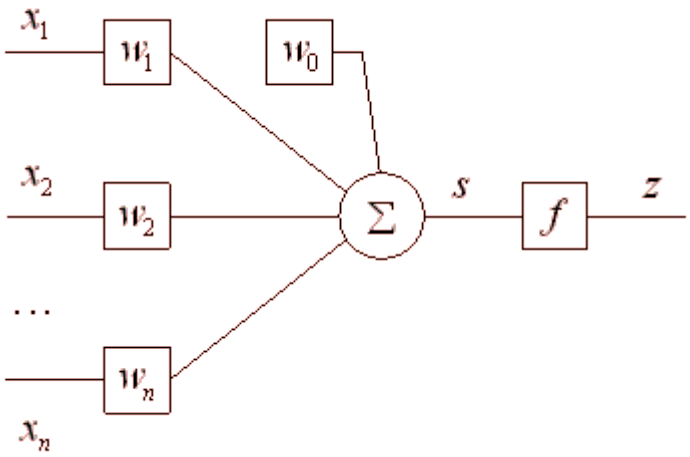

Рис. 11. Модель нейрона

Здесь x_i – значение сигнала, w_i – его вес, w_0 – вес нейрона смещения, S – сумма произведений входных весов и значений, f – функция активации, Z – выходной сигнал.

Таким образом, нейрон представляет собой сумматор входных сигналов с функцией активации. Математическая формула нейрона представлена на рисунке 12.

$$Z = \sum_{i=0}^{n} f(w_0 + (x_i * w_i)$$

Рис. 12. Формула нейрона

Нейрон смещения представляет собой нейрон без входов, с выходом, всегда равным 1. Он используется для сдвига графика функции активации, что порой необходимо при обучении [15]. Также нейрон активации позволяет применять нейронную сеть, если на вход подаются сигналы, равные 0. Без нейрона смещения в этом случае независимо от весов нейроны передадут 0 * веса = 0 [16]. Для реализации более сложных вычислений множество нейронов могут соединяться в слои, соединяемые между собой. Современная нейронная сеть состоит из входного слоя, одного или нескольких скрытых слоев и выходного слоя. Наглядно «слоеная» модель представлена на рисунке 13.

Процесс обучения представляет собой подбор таких значений весов в графе, что выходной результат будет наиболее близок к истинному. Для этого используются различные алгоритмы, например, алгоритм градиентного спуска [17]. Для предотвращения переобучения может использоваться алгоритм dropout [18], случайным образом отключающий некоторые ребра в графе.

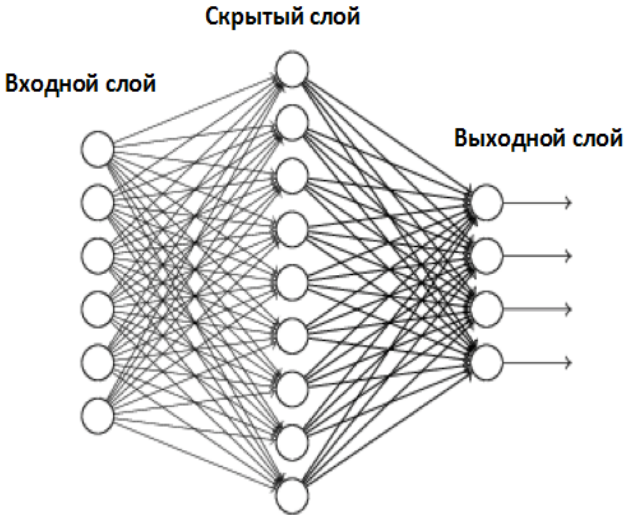

Рис. 13. Модель многослойной нейронной сети

2.4. Сверточная нейронная сеть

Для обнаружения лица на изображении используется сверточная нейронная сеть [19]. После успеха такой сети на конкурсе ImageNet в 2012 году [20], ее архитектура является мировым стандартом для распознавания изображений. Особенностью архитектуры являются сверточные слои и слои субдискретизации (подвыборки).

Сверточный слой используется для выделения признаков из изображения. Квадрат, названный «ядром», движется по изображению слева-направо, просматривая его и проводя скалярное перемножение значений ячеек в ядре и изображении. Этот процесс называется «сверткой». Наглядно его можно увидеть на рисунке 14.

Каждое ядро последовательно применяется ко всему изображению, составляя «карту признаков». Одна карта для одного ядра. Количество ядер и их размер задается произвольно. Значения в ячейках ядра задается случайным образом самой нейронной сетью.

Слой подвыборки уменьшает размер изображения, уплотняя признаки. Обычно ставится после слоя свертки. Также его применение оправдано для уменьшения объема вычислений и предотвращения переобучения [21].

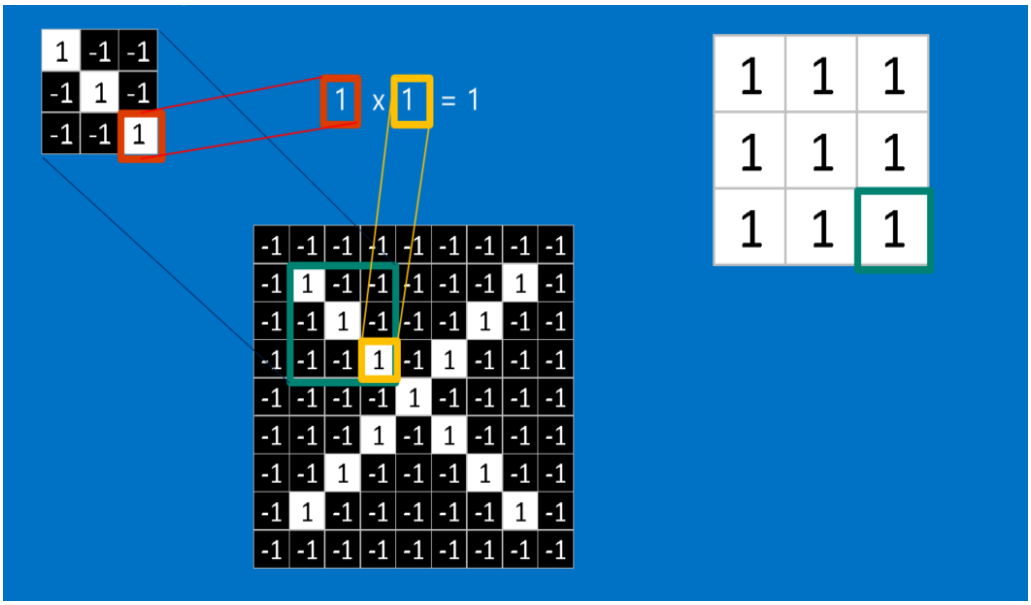

Рис. 14. Сверточный слой

Алгоритм работы слоя несложен: для каждой карты признаков применяется скользящее квадратное окно произвольного размера, преобразующее все пиксели в нем в один. Как правило, используется размер 2x2, что позволяет вдвое уменьшить изображение [22]. Немного похоже на свертку, но, в отличие от нее, нет ядра как такового - есть только окно. Еще одно отличие от свертки в том, что окна не могут пересекаться, т.е. шаг окна всегда равен его длине. Наглядно преобразование можно увидеть на рисунке 15.

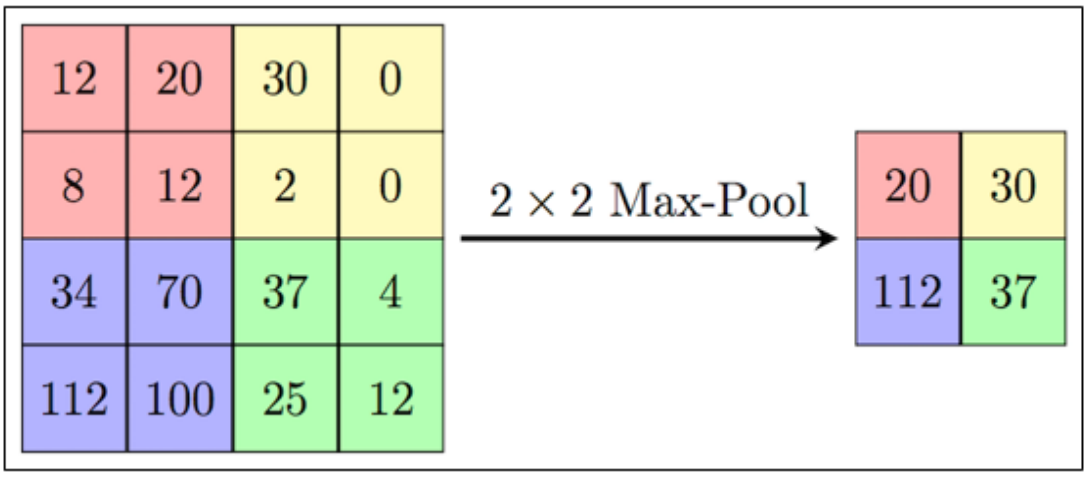

Рис. 15. Преобразования в слое подвыборки. Здесь из четырех ячеек, отмеченных одним цветом, выбирается одна с максимальным значением

2.5. Сиамская сеть

Сеть, предложенная в 1993 году [23], для сравнения двух входных данных. Представляет из себя две абсолютно одинаковые сверточные сети с одинаковыми весами. Каждая сеть принимает одно из изображений в качестве входных данных. Затем из них извлекаются признаки и выходы передаются в функцию, которая сравнивает вектор признаков по определенной метрике, определяя похожесть изображения. Пример архитектуры приведен на рисунке 16.

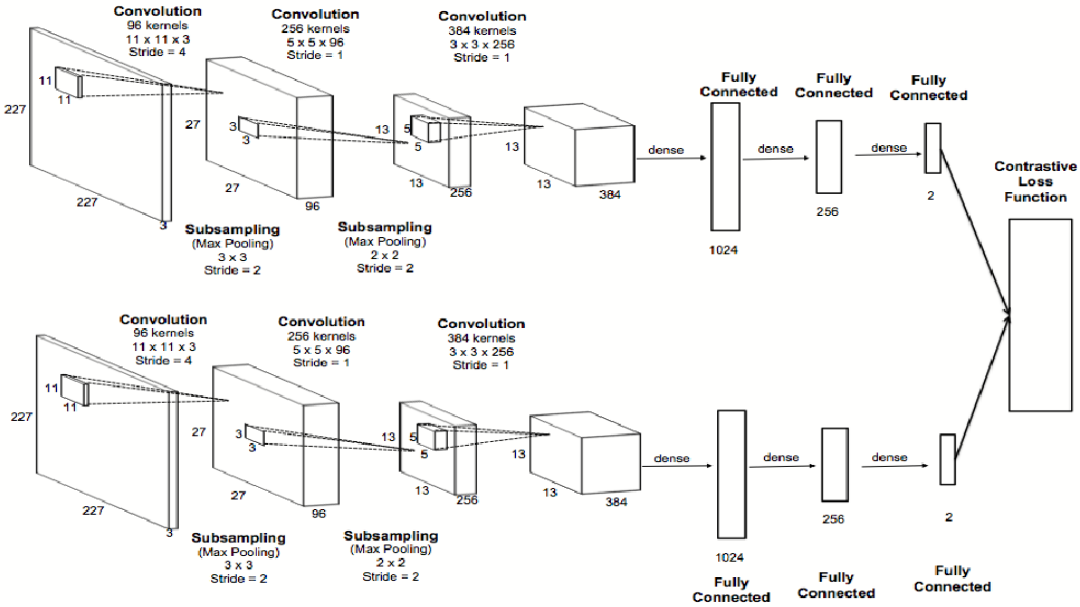

Рис. 16. Пример архитектуры сиамской нейронной сети

Следует понимать, что сиамская сеть – не нейронная сеть как таковая. Она скорее представляет собой идею объединения двух сверточных сетей таким образом, чтобы различать данные, подающиеся на их входы. Совершенно необязательно при предсказании должны одновременно работать две одинаковые сверточные сети. Достаточно из всех входных изображений извлечь дескрипторы лиц, а затем сравнить их по метрике.

Выводы по второй главе

В этой главе были даны теоретические сведения о предварительной обработке входных данных, определения алгоритма HOG, искусственной нейронной сети, сверточной нейронной сети и сиамской сети.

3. ПРОЕКТИРОВАНИЕ

3.1. Функциональные требования

Функциональные требования к системе описывают поведение системы. Разрабатываемая система должна удовлетворять следующим функциональным требованиям:

1) система должна принимать на вход видеопоток;

2) система должна определять, есть ли в кадре лица;

3) система должна идентифицировать человека, сравнивая его лицо с уже существующими записями;

4) система должна обозначать на изображении расположение лица и имя распознанного человека;

5) система должна быть способна распознавать несколько лиц на изображении;

6) системе должна иметь возможность добавлять и удалять записи о человеке с фотографией;

7) система должна иметь возможность изменить имя записи.

3.2. Нефункциональные требования

К нефункциональным требованиям системы относятся свойства, которыми она должна обладать. Например, удобство использования, безопасность, расширяемость и т.д.

Система должна удовлетворять следующим нефункциональным требованиям:

1) система должна быть реализована на языке Python 3.7;

2) система должна использовать библиотеку Keras с связке с TensorFlow для работы с нейронными сетями;

3) для хранения записей должна использоваться база данных PostgreSQL 12 [24].

3.3. Варианты использования

Для проектирования системы был использован унифицированный язык моделирования UML. Была построена модель взаимодействия актера с приложением распознавания лица. Диаграмма вариантов использования представлена на рисунке 17.

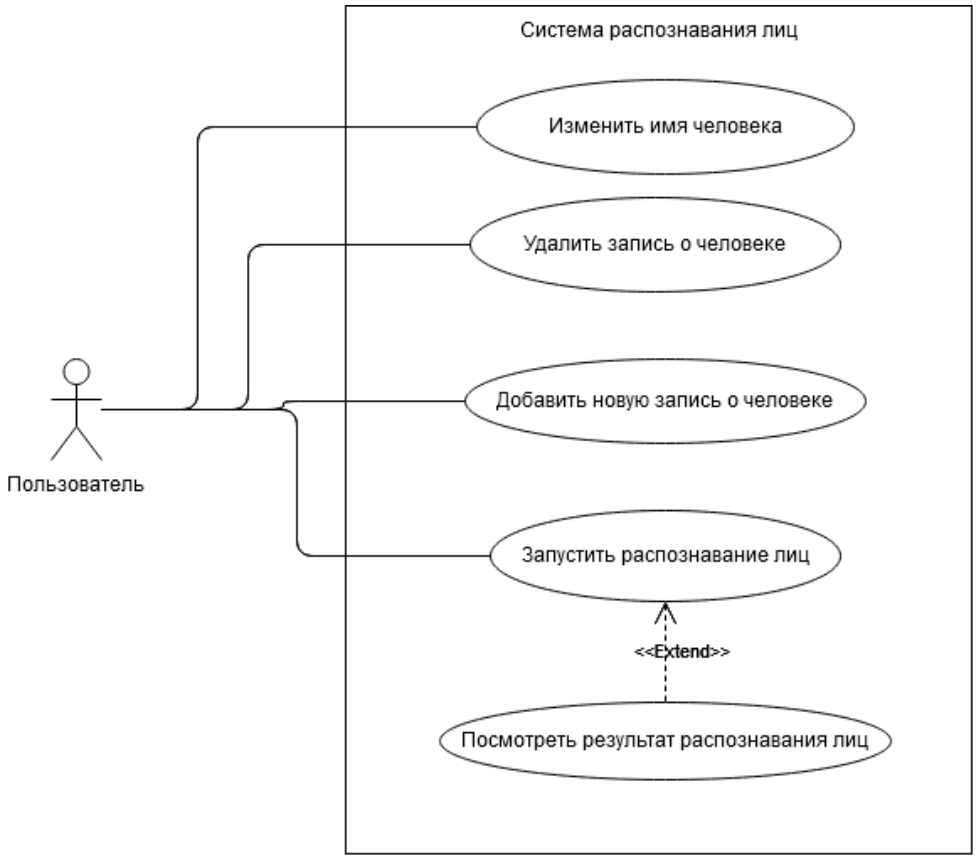

Рис. 17. Диаграмма вариантов использования

Актеры, взаимодействующие с системой:

Пользователь – человек, использующий систему и ее функционал.

Краткое описание вариантов использования:

Пользователю доступны следующие функции:

1) запустить распознавание лиц;
2) добавить новую запись в систему;
3) удалить запись из системы;
4) изменить имя записи в системе;
5) посмотреть результат распознавания.

3.4. Проектирование системы хранения записей

Для хранения записей о записях используется база данных PostgreSQL 12 [24]. Схема базы данных приведена на рисунке 18.

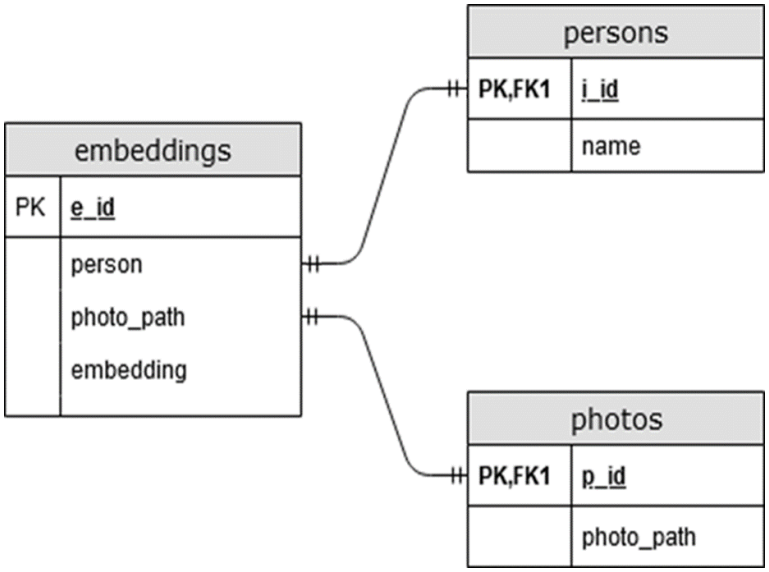

Рис. 18. Схема базы данных

Здесь embeddings – таблица, содержащая уникальный ключ e_id, внешний ключ person, указывающий на поле i_id таблицы persons, внешний ключ photo_path, указывающий на поле p_id таблицы photos, и поле embedding, содержащее дескриптор лица. В таблице persons поле i_id явзяется уникальным ключом, поле name содержит имя пользователя. В таблице photos поле p_id является уникальным ключом, поле photo_path содержит название изображения, сохраненного на диске.

3.5. Диаграмма деятельности

На основе функциональных и нефункциональных требований была разработана диаграмма деятельности, показанная на рисунке 19. На данной диаграмме изображен процесс взаимодействия пользователя с системой распознавания лиц в контексте запуска приложения. При запуске приложения автоматически захватывается видеопоток с веб-камеры. Затем видеопоток разделяется на отдельные кадры, после чего каждый кадр предварительно обрабатывается. Следом на каждом кадре ищутся лица и лич-

ности идентифицируются, если найдены. Дальше происходит постобработка – лица на изображении отмечаются синим квадратом с подписанным именем (если запись о человеке уже есть в системе), а само изображение трансформируется в формат, пригодный для вывода. В результате выводится финальное изображение с отмечеными и идентифицированными лицами, на которое может посмотреть пользователь.

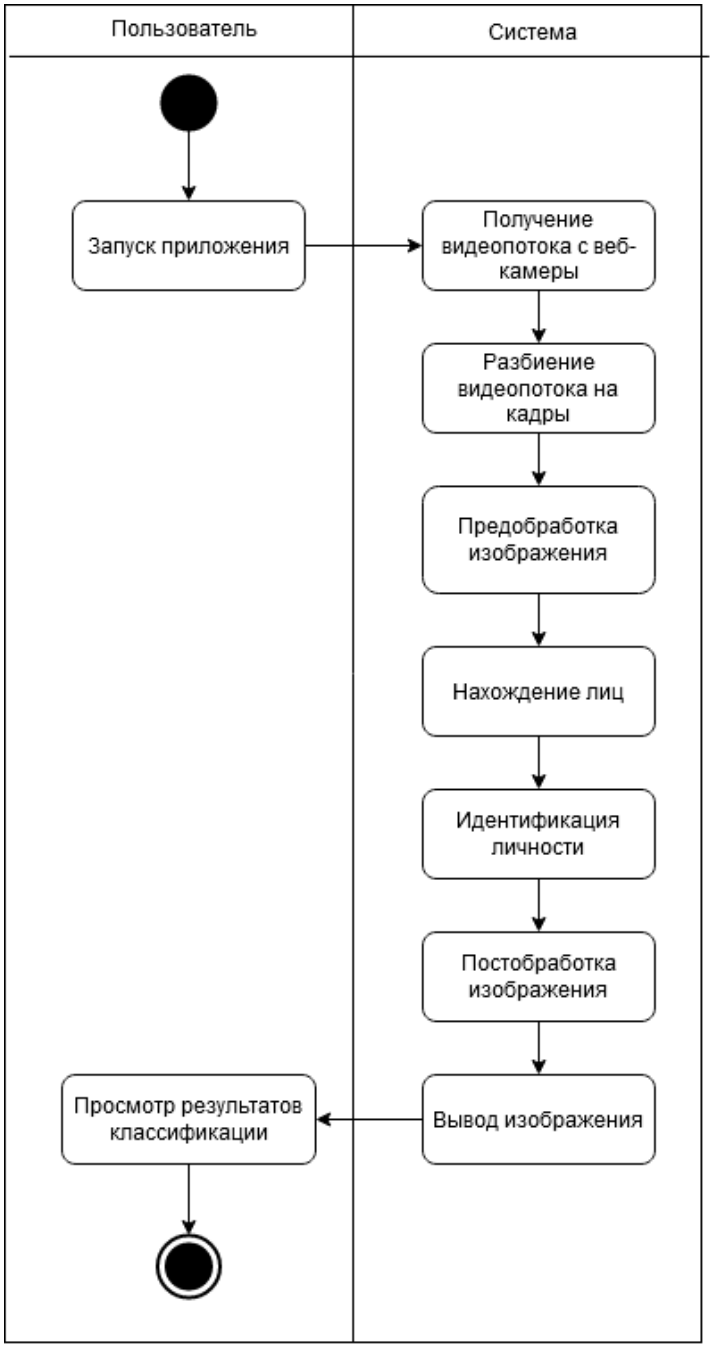

Рис. 19. Диаграмма деятельности

3.6. Топология нейронной сети для извлечения дескрипторов

Для решения поставленной задачи была использована предобученная на наборе данных LFW топология сверточной нейронной сети, представленная на рисунке 20.

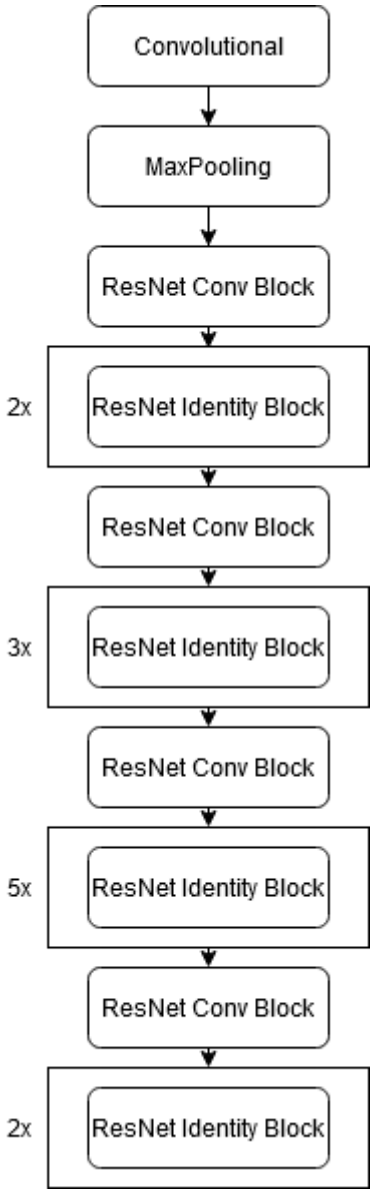

Рис. 20. Топология сверточной нейронной сети

Топология представляет собой модифицированную архитектуру ResNet-50 с удаленным слоем классификации. Сеть представляет собой только извлекатель дескрипторов, которые затем сравниваются по геометрической метрике [5]. Особенность этой архитектуры в том, что в ней используются остаточные блоки (residual blocks), реализующие конкатена-

цию входного слоя блока с выходным, улучшая таким образом точность предсказания [25].

3.7. Архитектура системы идентификации лиц

Диаграмма компонентов системы приведена на рисунке 21.

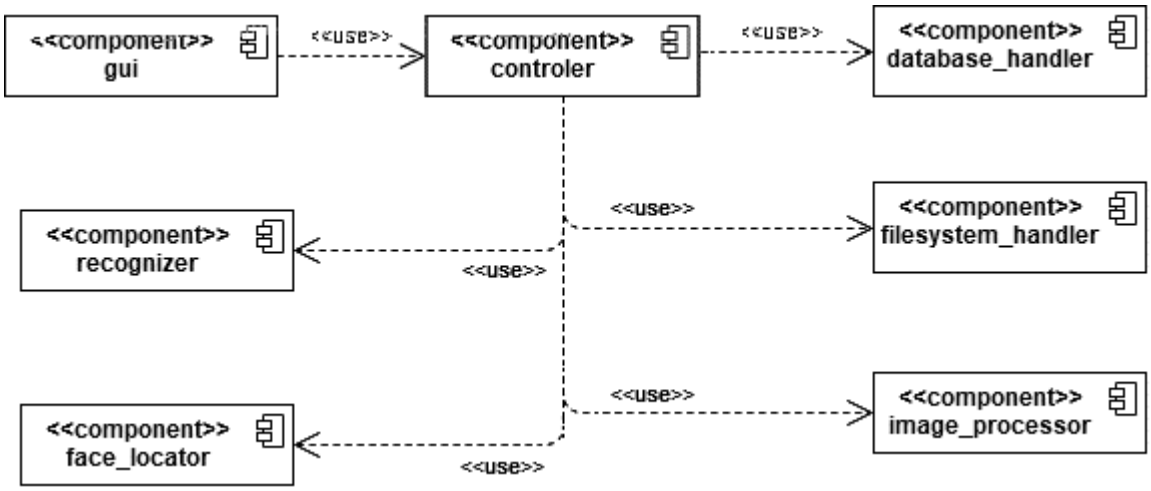

Рис. 21. Диаграмма компонентов

Здесь компонент gui является модулем, реализующим графический интерфейс пользователя. Его взаимодействие с системой происходит через модуль controller.

Модуль database_handler реализует методы для работы с базой данных. Он позволяет создавать записи, читать их, обновлять и удалять, т.е. реализует принципы CRUD.

Модуль filesystem_handler реализует методы для работы с файловой системой. Позволяет работать с фотографиями на диске.

Модуль image_processor реализует методы для работы с изображениями. Позволяет проводить предобработку и постобработку.

Модуль face_locator реализует методы для поиска координат лиц на изображении.

Модуль recognizer реализует методы для извлечения дескриптора из найденного лица и проведения идентификации.

3.8. Проектирование графического интерфейса пользователя

Графический интерфейс приложения должен быть минималистичным и интуитивно понятным для пользователя. Разработанный макет интерфейса приведен на рисунке 22.

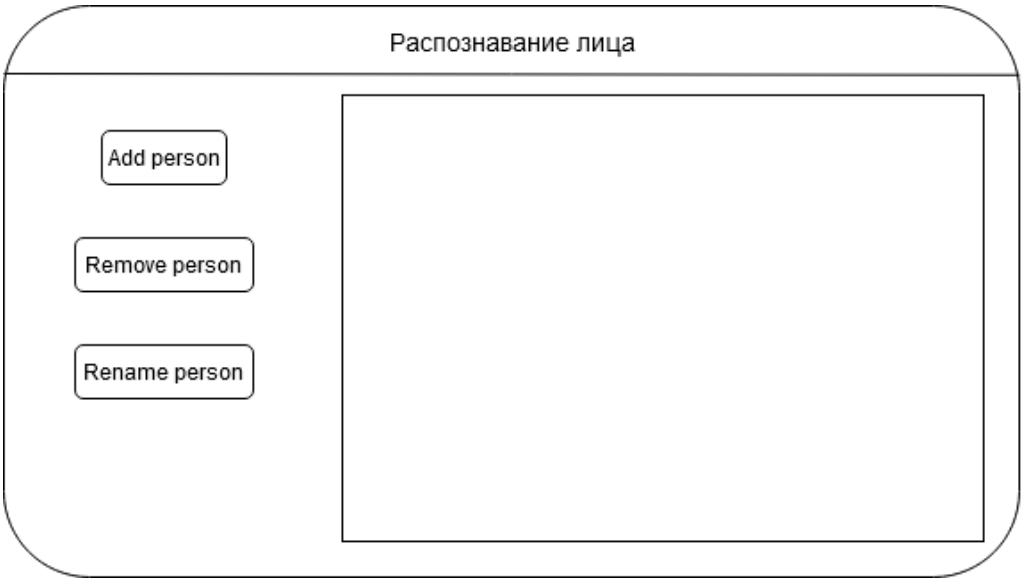

Рис. 22. Макет пользовательского интерфейса

Выводы по третьей главе

В третьей главе были определены функциональные и нефункциональные требования для системы, а также представлены диаграммы вариантов использования и деятельности, схема базы данных. В свою очередь была спроектирована топология нейронной сети, система распознавания лица, система хранения записей и пользовательский интерфейс, реализации которых представлены в следующей главе.

4. РЕАЛИЗАЦИЯ

4.1. Средства реализации

Для реализации программной части системы был выбран язык программирования Python 3.7 [26]. Разработка велась в среде JetBrains PyCharm Professional 2020.1.1 [27].

Для языка Python были использованы следующие библиотеки: Keras 2.3.1 [13], Tensorflow 2.1.0 [12], Numpy 1.16.4 [10], Psycopg2 2.8.4 [28], dlib 19.19 [29], face_recognition 1.3.0 [30], Scikit-learn 0.22.1 [31], PyQt 5.12.3 [32], OpenCV 4.2.0 [14], Pillow 7.1.2 [33].

Для реализации системы хранения записей использовалась реляционная база данных PostgreSQL 12 [24].

4.2. Реализация предобработки входных данных

Перед подачей изображений в алгоритм для поиска лиц и в нейронную сеть для распознавания реализована предобработка входных данных. Она заключается в изменении размера изображения, его нормализации и конвертации цветового пространства.

Реализация конвертации цветового пространства представлена на рисунке 23.

```
def convert_color_format(img):
    """
    Метод конвертирует цветовой формат изображения из BRG в RGB

    :param img: Конвертируемое изображение.
    :return: Сконвертированное изображение
    """
    return cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Рис. 23. Листинг конвертации цветового формата

Реализация изменения размера изображения и его нормализации представлена на рисунке 24.

```
def prepare_for_classificator_nn(face_images):
    """
    Метод, выполняющий нормализацию изображения перед передачей его в нейронную сеть – классификатор.
    Изменяет размер изображения, добавляет измерение.

    :param face_images: Список вырезанных изображений лиц.
    :return: Нормализованные изображения.
    """
    prepared_images_with_coords = []
    for value in face_images:
        resized_img = cv2.resize(src=value[0], dsize=(197, 197))  # Приведение изображения к единому размеру
    normalized_img = np.around(resized_img / 255, 15)
    prepped_img = np.expand_dims(normalized_img, axis=0)  # Добавляет дополнительное измерение в массив изображения (нужно для нейронной сети)
    prepared_images_with_coords.append([prepped_img, value[1]])
    return prepared_images_with_coords
```

Рис. 24. Листинг изменения размера и нормализации изображения

4.3. Реализация постобработки выходных данных

Для отображения результатов работы системы необходима постобработка выходных данных, а именно отметка лиц квадратами, с подписанными именами (если они присутствуют в системе).

Реализация постобработки представлена на рисунке 25.

```
def highlight_identity(img, face_identities_with_coords):
    """
    Метод, рисующий квадраты, центрами которых являются лица. Если передано имя, то оно также будет отображено.

    :param img: Изображение, на котором отмечаются лица.
   :param face_identities_with_coords: Информация об именах найденных на изображении людей и координатах их лиц.
    :return: Обработанное изображение.
    """
    if face_identities_with_coords is None:
        face_identities_with_coords = []
        for value in face_identities_with_coords:
            name = value[0]
            top, right, bottom, left = value[1]
            padding = int(abs(top - bottom * left - right) * 0.0003)
            img = cv2.rectangle(img, (right + padding, bottom + padding), (left - padding, top - padding), (0, 0, 255), 5)
            cv2.rectangle(img, (left, bottom + padding * 2), (right, bottom + padding), (0, 0, 255), cv2.FILLED)
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(img, name, (left + 6, bottom + padding * 2 - 6), font, 1, (255, 255, 255), 1)
    return img
```

Рис. 25. Реализация постобработки выходных данных

4.4. Реализация алгоритма для поиска лиц

Для реализации алгоритма поиска лиц был использован пакет face_recognition [30]. Пакет использует реализацию алгоритма HOG для поиска лиц. Точность работы используемого алгоритма составляет 99.38% на наборе данных Labeled Faces in the Wild [34].

4.5. Реализация нейронной сети для извлечения дескрипторов

На рисунке 26 представлена реализация нейронной сети для извлечения дескрипторов.

```
def RESNET50(include_top=True, weights='vggface',
         input_tensor=None, input_shape=None,
         pooling=None,
         classes=8631):
input_shape = _obtain_input_shape(input_shape,
                default_size=224,
                min_size=32,
                data_format=K.image_data_format(),
                require_flatten=include_top,
                weights=weights)

if input_tensor is None:
    img_input = Input(shape=input_shape)
    else:
        if not K.is_keras_tensor(input_tensor):
            img_input = Input(tensor=input_tensor, shape=input_shape)
        else:
            img_input = input_tensor
if K.image_data_format() == 'channels_last':
        bn_axis = 3
else:
        bn_axis = 1

x = Conv2D(
        64, (7, 7), use_bias=False, strides=(2, 2), padding='same',
        name='conv1/7x7_s2')(img_input)
x = BatchNormalization(axis=bn_axis, name='conv1/7x7_s2/bn')(x)
x = Activation('relu')(x)
x = MaxPooling2D((3, 3), strides=(2, 2))(x)

x = resnet_conv_block(x, 3, [64, 64, 256], stage=2, block=1, strides=(1,
1))
x = resnet_identity_block(x, 3, [64, 64, 256], stage=2, block=2)
x = resnet_identity_block(x, 3, [64, 64, 256], stage=2, block=3)

x = resnet_conv_block(x, 3, [128, 128, 512], stage=3, block=1)
x = resnet_identity_block(x, 3, [128, 128, 512], stage=3, block=2)
x = resnet_identity_block(x, 3, [128, 128, 512], stage=3, block=3)
x = resnet_identity_block(x, 3, [128, 128, 512], stage=3, block=4)

x = resnet_conv_block(x, 3, [256, 256, 1024], stage=4, block=1)
x = resnet_identity_block(x, 3, [256, 256, 1024], stage=4, block=2)
x = resnet_identity_block(x, 3, [256, 256, 1024], stage=4, block=3)
x = resnet_identity_block(x, 3, [256, 256, 1024], stage=4, block=4)
x = resnet_identity_block(x, 3, [256, 256, 1024], stage=4, block=5)
x = resnet_identity_block(x, 3, [256, 256, 1024], stage=4, block=6)

x = resnet_conv_block(x, 3, [512, 512, 2048], stage=5, block=1)
x = resnet_identity_block(x, 3, [512, 512, 2048], stage=5, block=2)
      x = resnet_identity_block(x, 3, [512, 512, 2048], stage=5, block=3)
```

Рис. 26. Топология ResNet50 для извлечения дескрипторов

Реализация блока ResNet Identity Block представлена на рисунке 27.

```
def resnet_identity_block(input_tensor, kernel_size, filters, stage,
block,
                          bias=False):
    filters1, filters2, filters3 = filters
    if K.image_data_format() == 'channels_last':
        bn_axis = 3
    else:
        bn_axis = 1
    conv1_reduce_name = 'conv' + str(stage) + "_" + str(block) +
"_1x1_reduce"
    conv1_increase_name = 'conv' + str(stage) + "_" + str(
        block) + "_1x1_increase"
    conv3_name = 'conv' + str(stage) + "_" + str(block) + "_3x3"

    x = Conv2D(filters1, (1, 1), use_bias=bias, name=conv1_reduce_name)(
        input_tensor)
    x = BatchNormalization(axis=bn_axis, name=conv1_reduce_name +
"/bn")(x)
    x = Activation('relu')(x)

    x = Conv2D(filters2, kernel_size, use_bias=bias,
               padding='same', name=conv3_name)(x)
    x = BatchNormalization(axis=bn_axis, name=conv3_name + "/bn")(x)
    x = Activation('relu')(x)

    x = Conv2D(filters3, (1, 1), use_bias=bias,
name=conv1_increase_name)(x)
    x = BatchNormalization(axis=bn_axis, name=conv1_increase_name +
"/bn")(x)

    x = layers.add([x, input_tensor])
    x = Activation('relu')(x)
    return x
```

Рис. 27. Реализация блока ResNet Identity Block

Реализация блока ResNet Conv Block представлена на рисунке 28.

```
def resnet_conv_block(input_tensor, kernel_size, filters, stage, block,
                      strides=(2, 2), bias=False):
    filters1, filters2, filters3 = filters
    if K.image_data_format() == 'channels_last':
        bn_axis = 3
    else:
        bn_axis = 1
    conv1_reduce_name = 'conv' + str(stage) + "_" + str(block) +
"_1x1_reduce"
    conv1_increase_name = 'conv' + str(stage) + "_" + str(
        block) + "_1x1_increase"
    conv1_proj_name = 'conv' + str(stage) + "_" + str(block) +
"_1x1_proj"
    conv3_name = 'conv' + str(stage) + "_" + str(block) + "_3x3"

    x = Conv2D(filters1, (1, 1), strides=strides, use_bias=bias,
               name=conv1_reduce_name)(input_tensor)
    x = BatchNormalization(axis=bn_axis, name=conv1_reduce_name +
"/bn")(x)
    x = Activation('relu')(x)

    x = Conv2D(filters2, kernel_size, padding='same', use_bias=bias,
               name=conv3_name)(x)
    x = BatchNormalization(axis=bn_axis, name=conv3_name + "/bn")(x)
    x = Activation('relu')(x)

    x = Conv2D(filters3, (1, 1), name=conv1_increase_name,
use_bias=bias)(x)
    x = BatchNormalization(axis=bn_axis, name=conv1_increase_name +
"/bn")(x)

    shortcut = Conv2D(filters3, (1, 1), strides=strides, use_bias=bias,
                      name=conv1_proj_name)(input_tensor)
    shortcut = BatchNormalization(axis=bn_axis, name=conv1_proj_name +
"/bn")(
        shortcut)

    x = layers.add([x, shortcut])
    x = Activation('relu')(x)
    return x
```

Рис. 28. Реализация блока ResNet Conv Block

4.6. Реализация идентификации человека

Для реализации идентификации используется сравнение дескрипторов, полученных из сверточной нейронной сети, по геометрической метрике. Чем меньше разница между дескрипторами, тем больше извлеченный дескриптор похож на уже существующий в системе. Реализация идентификации представлена на рисунке 29.

```
def identify(self, embedding_with_coords, existing_embeddings, thresh-
old=3):
    if len(existing_embeddings) == 0 or len(embedding_with_coords) == 0:
        print('No embedding.')
        return
    identities_with_coords = []
    for embedding, coords in embedding_with_coords:
        distances = {}
        for exist_embed in existing_embeddings:
            score = cosine(exist_embed[0], embedding)
            distances[exist_embed[1]] = score
        min_dist_value = (min(distances, key=distances.get),
max(distances.values()))
        if min_dist_value[1] <= threshold:
            print(f'Face Recognized: {min_dist_value[0]}, distance:
{min_dist_value[1]}')
            identities_with_coords.append([min_dist_value[0], coords])
        else:
            print(f'Face not recognized, distance: {min_dist_value[1]}')
            identities_with_coords.append(['', coords])
    return identities_with_coords
```

Рис. 29. Реализация идентификации

4.7. Реализация системы хранения информации о личностях

Для хранения данных использована СУБД PostgreSQL 12. Фотографии хранятся на жестком диске. В базе данных содержатся три таблицы – persons, в которой хранятся имена людей; photos, в которой хранятся пути до фотографий на диске; embeddings, в которой хранятся дескрипторы лиц и ассоциированные с ними имена людей и пути до фотографий, реализованные внешними ключами.

При добавлении новой записи в базу данных дескриптор лица сериализуется в массив байт. При получении данных дескриптор десериализуется. Таким образом, дескриптор хранится в базе данных целиком, преобразованный в массив байт.

4.8. Реализация пользовательского интерфейса

Для реализации пользовательского интерфейса была выбрана библиотека PyQt. Интерфейс готового приложения представлен на рисунке 30.

В главном окне приложения располагаются три кнопки и поле, в котором показывается изображение с камеры. Кнопка «Add Person» открыва-

ет окно для добавления нового пользователя. Скриншот окна приведен на рисунке 31.

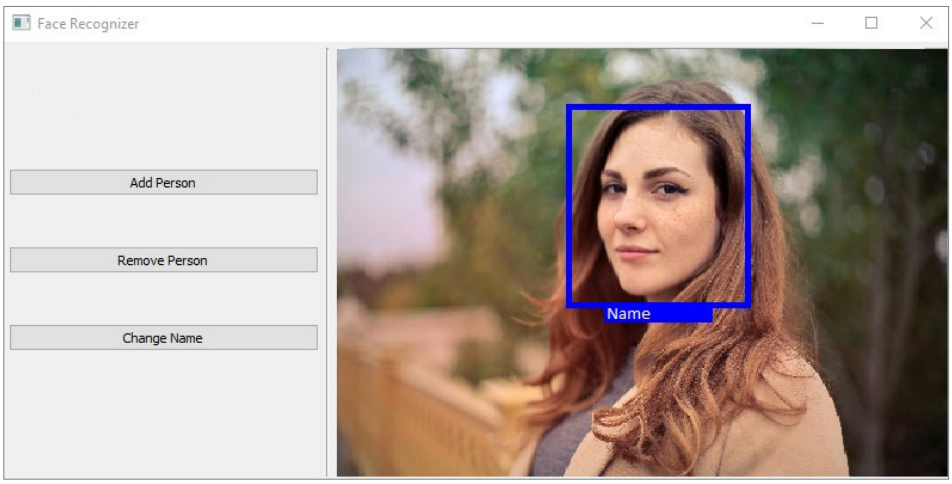

Рис. 30. Интерфейс приложения

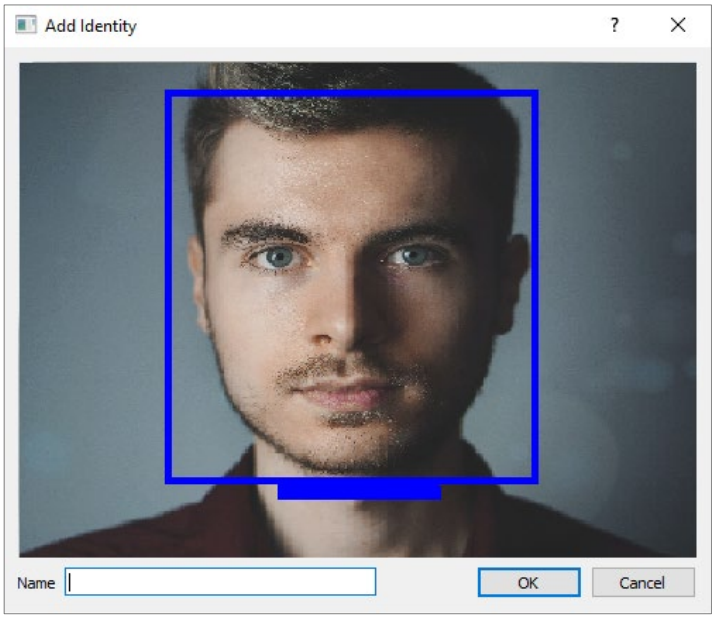

Рис. 31. Интерфейс добавления новой записи

Здесь поле «Name» отвечает за название новой записи. Из изображения вырезается участок с лицом, извлекается дескриптор после чего изображение сохраняется на диск, в базу данных вносится информация о названии, дескрипторе и пути до фотографии на диске.

Кнопка «Remove Person» вызывает окно для удаления записи в системе. Скриншот окна приведен на рисунке 32.

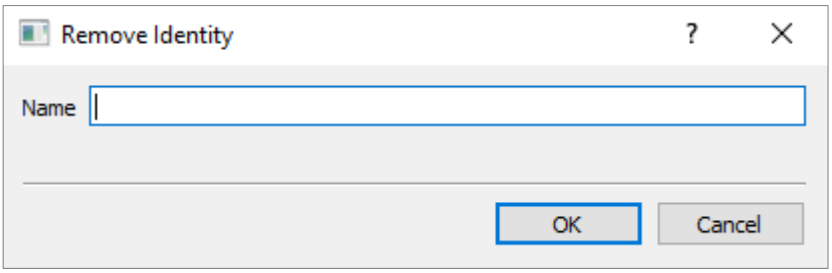

Рис. 32. Интерфейс окна удаления записи

Здесь поле «Name» отвечает за имя записи в системе. При удалении и диска удаляется одноименная папка с фотографией, из базы данных удаляется запись о человеке, его фотографиях и дескрипторах лица.

Кнопка «Change Name» вызывает окно для изменения имени. Скриншот окна приведен на рисунке 33.

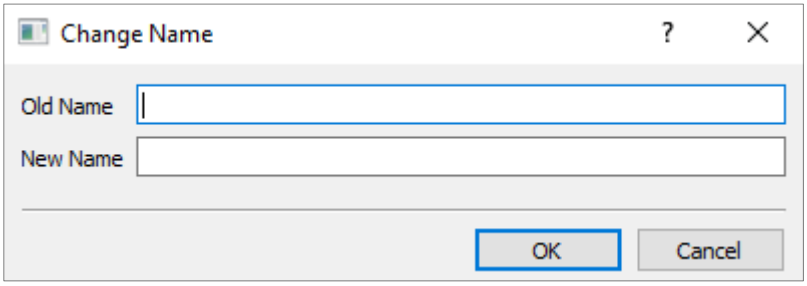

Рис. 33. Интерфейс окна изменения имени

Здесь «Old Name» – старое имя, «New Name» – новое. При изменении имени изменятся имя папки с фотографиями на диске, а также запись об имени в базе данных.

Выводы по четвертой главе

В данной главе были описаны используемые средства разработки, а также реализация программного кода самого приложения. Разработанное приложение полностью соответствует ранее упомянутым требованиям.

5. ТЕСТИРОВАНИЕ

5.1. Тестирование нейронной сети

Для тестирования идентификации человека был использован набор данных AT&T Database of Faces [35], содержащий 10 изображений лиц, разделенных на 40 классов. Сеть показала точность 93.23 %.

5.2. Функциональное тестирование

Было проведено функциональное тестирование системы согласно описанным ранее требованиям. Результаты тестирования приведены в таблице 1.

Табл. 1. Результаты функционального тестирования

№	Предусловие	Действие	Ожидаемый результат	Результат теста
1	Проверка запуска приложения	Пользователь запускает приложение	На компьютере пользователя запускается консольное приложение	Пройден
2	Приложение запущено	Пользователь ничего не делает	В приложении показывается видеопоток с веб-камеры	Пройден
3	Приложение запущено	Лицо пользователя в пределах видимости камеры	На показанном видеопотоке приложения рамкой отмечается лицо и имя пользователя (если есть запись)	Пройден
4	Приложение запущено	Лица нескольких пользователей в пределах видимости камеры	На показанном видеопотоке приложения рамкой отмечаются все лица и имена пользователей (если есть записи)	Пройден
5	Приложение запущено	Пользователь нажимает кнопку «Add Person»	В дополнительном окне открывается форма добавления записи	Пройден

Продолжение табл. 1

№	Предусловие	Действие	Ожидаемый результат	Результат теста
6	Приложение запущено, открыто окно добавления записи	Пользователь ничего не делает	В форме добавления записи отображается застывшее изображение с камеры	Пройден
7	Приложение запущено, открыто окно добавления записи	Пользователь вводит имя и нажимает кнопку «ОК»	Форма закрывается, в систему добавляется запись, на диск сохраняется фотография, в основном окне продолжается показ видеопотока	Пройден
8	Приложение запущено	Пользователь нажимает кнопку «Remove Person»	Открывается форма удаления записи из системы	Пройден
9	Приложение запущено, открыто окно удаления записи	Пользователь вводит имя и нажимает кнопку «ОК»	Запись удаляется из системы, с диска стирается одноименная папка с фотографиями, форма удаления записи закрывается	Пройден
10	Приложение запущено	Пользователь нажимает кнопку «Change Name»	Открывается окно изменения имени	Пройден
11	Приложение запущено, открыто окно изменения имени	Пользователь водит новое и старое имя и нажимает кнопку «ОК»	Окно изменения имени закрывается, имя в системе меняется, имя папка с фотографиями на диске меняется	Пройден

Выводы по пятой главе

В данной главе было произведено тестирование работы нейронной сети, а также функциональное тестирование. Все подготовленные тесты были успешно пройдены.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была спроектировано и реализовано приложение для распознавания и идентификации человека по биометрическим данным лица с использованием нейросетевых технологий, а также было проведено тестирование системы.

Были рассмотрены существующие аналоги систем для распознавания лиц и сделан вывод об актуальности работы.

Была рассмотрена теоретическая часть сверточных нейронных сетей (принципы их работы, информация о слоях свертки и подвыборки), сиамских сетей, а также теоретическая часть алгоритма HOG. Был сформулирован процесс предобработки изображения.

Были определены функциональные и нефункциональные требования и представлены диаграммы деятельности и вариантов использования, а также спроектирована система хранения записей.

Представлена реализация нейронной сети, пользовательского интерфейса и системы хранения записей. Разработанная система соответствует всем требованиям. Была протестирована работа нейронной сети и единой системы распознавания. Система выполнила свою задачу и осуществила классифицирование набора входных изображений.

Были решены следующие задачи:

1) проведен анализ программных аналогов;
2) реализован алгоритм для поиска лиц на изображении;
3) спроектирована топология искусственной нейронной сети;
4) разработан пользовательский интерфейс;
5) разработана система хранения записей;
6) проведено тестирование приложения.

ЛИТЕРАТУРА

1. Lin W., Tong T., Gao Q., Guo D., Du X., Yang Y., Guo G., Xiao M., Du M., Qu X. Convolutional neural networks-based MRI image analysis for the Alzheimer's disease prediction from mild cognitive impairment // Frontiers in Neuroscience, Vol. 12, No. NOV, 2018. pp. 1-13.

2. Что такое распознавание лиц на Facebook и как работает эта функция? [Электронный ресурс] URL: https://www.facebook.com/help/122175507864081 (дата обращения: 15.05.2020).

3. Хайкин С. Нейронные сети: полный курс. 2-е изд. Москва: ООО "И.Д. Вильямс", 2017. 1104 с.

4. LeCun Y., Bengio Y. Convolution Networks for Images, Speech, and Time-Series // Igarss 2014, No. 1, 1998. pp. 1-5.

5. Ochiai A. Zoogeographical studies on the soleoid fishes found Japan and its neighboring regions // Bulletin of the Japanese Society of Scientific Fisheries. 1957. Vol. 22. No. 9. pp. 526-530.

6. Navneet D., Triggs B. Histograms of oriented gradients for human detection // 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). San Diego, CA, USA. 2005.

7. Официальный сайт PicTriev [Электронный ресурс] URL: http://www.pictriev.com/?lang=ru (дата обращения: 12.04.2020).

8. Официальный сайт Betaface [Электронный ресурс] URL: https://www.betafaceapi.com/wpa/ (дата обращения: 12.04.2020).

9. DeepFace [Электронный ресурс] URL: https://research.fb.com/publications/deepface-closing-the-gap-to-human-level-performance-in-face-verification/ (дата обращения: 12.04.2020).

10. Документация NumPy [Электронный ресурс] URL: http://docs.scipy.org/doc/numpy/ (дата обращения: 24.05.2020).

11. Документация Matplotlib [Электронный ресурс] URL: https://matplotlib.org/contents.html (дата обращения: 16.05.2020).

12. Документация TensorFlow [Электронный ресурс] URL: https://www.tensorflow.org/api_docs/python (дата обращения: 16.05.2020).

13. Официальный сайт пакета Keras [Электронный ресурс] URL: https://keras.io/ (дата обращения: 16.05.2020).

14. Официальный сайт пакета OpenCV [Электронный ресурс] URL: https://opencv.org/about.html (дата обращения: 16.05.2020).

15. Neural Network Bias: Bias Neuron, Overfitting and Underfitting [Электронный ресурс] URL: https://missinglink.ai/guides/neural-network-concepts/neural-network-bias-bias-neuron-overfitting-underfitting/ (дата обращения: 16.05.2020).

16. Rashid T. Make your own neural network. CreateSpace Independent Publishing Platform, 2016. 222 pp.

17. Zhang J. Gradient Descent based Optimization Algorithms for Deep Learning Models Training, 2019.

18. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting // Journal of Machine Learning Research, Vol. 15, 2014. pp. 1929-1958.

19. LeCun Y., Bengio Y. Convolution Networks for Images, Speech, and Time-Series // Igarss 2014, No. 1, 1998. pp. 1-5.

20. Large Scale Visual Recognition Challenge [Электронный ресурс] URL: http://image-net.org/challenges/LSVRC/2012/results.html#t1 (дата обращения: 16.05.2020).

21. A Beginner's Guide To Understanding Convolutional Neural Networks Part 2. [Электронный ресурс] URL: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/ (дата обращения: 16.05.2020).

22. Convolutional Neural Networks (CNNs / ConvNets) [Электронный ресурс] URL: http://cs231n.github.io/convolutional-networks/#norm (дата обращения: 16.05.2020).

23. Bromley J., Bentz J.W., Bottou L., Guyon I., LeCun Y., Moore C., Säckinger E., Shah R. Signature Verification Using a “Siamese” Time Delay Neural Network // International Journal of Pattern Recognition and Artificial Intelligence, Vol. 7, No. 4, 1993. pp. 669-688.

24. Официальный сайт PostgreSQL [Электронный ресурс] URL: https://www.postgresql.org/ (дата обращения: 11.04.2020).

25. Kaiming H., Xiangyu Z., Shaoqing R., Jian S. Deep Residual Learning for Image Recognition // Proc. IEEE Conf. Comput. Vis. Pattern Recognit. 2016. pp. 770-778.

26. Официальный сайт Python [Электронный ресурс] URL: https:// www.python.org/ (дата обращения: 13.03.2020).

27. Официальный сайт PyCharm [Электронный ресурс] URL: https:// www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 08.03.2020).

28. Документациея Psycopg2 [Электронный ресурс] URL: https:// www.psycopg.org/docs/ (дата обращения: 02.04.2020).

29. Документация пакета dlib [Электронный ресурс] URL: http:// dlib.net/python/index.html (дата обращения: 05.04.2020).

30. Страница пакета face_rocognition [Электронный ресурс] URL: https://pypi.org/project/face-recognition/ (дата обращения: 11.04.2020).

31. Официальный сайт пакета Scikit [Электронный ресурс] URL: https://scikit-learn.org/ (дата обращения: 05.04.2020).

32. Документация PyQt5 [Электронный ресурс] URL: https:// www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 24.04.2020).

33. Официальный сайт пакета Pilllow [Электронный ресурс] URL:

https://pillow.readthedocs.io/en/stable/ (дата обращения: 23.03.2020).

34. LFW Face Database [Электронный ресурс] URL: http://vis-www.cs.umass.edu/lfw/ (дата обращения: 05.03.2020).

35. AT&T Database of Faces [Электронный ресурс] URL: https://www.kaggle.com/kasikrit/att-database-of-faces (дата обращения: 15.04.2020).