МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙФЕДЕРАЦИИ

Федеральное государственное автономное образовательное   
учреждение высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

**Разработка приложения для распознавания жильцов на видео с камеры домофона на основе нейросетевых технологий**

КУРСОВАЯ РАБОТА

по дисциплине «Программная инженерия»

ЮУрГУ – 02.03.02.2020.308-278.КР

|  |  |
| --- | --- |
| Нормоконтролер,  к.ф.-м.н., доцент кафедры CП  \_\_\_\_\_\_\_\_\_\_С.А. Иванов “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 2020 г. | Научный руководитель,  к.ф.-м.н., доцент кафедры CП  \_\_\_\_\_\_\_\_\_\_ С.А. Иванов  Автор работы,  студент группы КЭ-302  \_\_\_\_\_\_\_\_\_\_ А.А. Владимиров  Работа защищена  с оценкой: \_\_\_\_\_\_\_\_\_\_\_  “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_ 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

«\_\_\_».\_\_\_\_\_\_\_\_\_\_\_\_\_.2020

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине «Программная инженерия»

студенту группы КЭ-302

Владимирову Артёму Александровичу

обучающемуся по направлению   
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы**

Разработка приложения для распознавания жильцов на видео с камеры домофона на основе нейросетевых технологий.

1. **Срок сдачи студентом законченной работы:** 31.05.2020 г.
2. **Исходные данные к работе**
   1. Саймон Хайкин, «Нейронные сети: полный курс, 2-е издание», 2006.
   2. Распознавание лиц с помощью сиамских сетей [Электронный ресурс] URL: https://habr.com/ru/company/jetinfosystems/blog/465279/
3. **Перечень подлежащих разработке вопросов**
   1. Провести обзор аналогов и научной литературы.
   2. Подготовить обучающую и тестовую выборку изображений.
   3. Программно реализовать и обучить нейронную сеть.
   4. Разработать и протестировать приложение.
4. **Дата выдачи задания:** 17.02.2020 г.

Научный руководитель С.А. Иванов

Задание принял к исполнению А.А. Владимиров

оглавление

[ВВЕДЕНИЕ 4](#_Toc50752437)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc50752438)

[1.1. Обзор аналогичных решений 6](#_Toc50752439)

[1.2. Обзор готовых решений по созданию нейронных сетей 7](#_Toc50752440)

[2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8](#_Toc50752441)

[2.1. Источник данных 8](#_Toc50752442)

[2.2. Библиотека Face recognition 8](#_Toc50752443)

[2.3. Свёрточная нейронная сеть 9](#_Toc50752444)

[2.4. Сиамская сеть 11](#_Toc50752445)

[3. ПРОЕКТИРОВАНИЕ 13](#_Toc50752446)

[3.1. Функциональные требования 13](#_Toc50752447)

[3.2. Нефункциональные требования 13](#_Toc50752448)

[3.3. Варианты взаимодействия с системой 13](#_Toc50752449)

[3.4. Диаграмма деятельности 14](#_Toc50752450)

[4. РЕАЛИЗАЦИЯ 16](#_Toc50752451)

[4.1. Средства разработки 16](#_Toc50752452)

[4.2. Формирование обучающей выборки 16](#_Toc50752453)

[4.3. Обучение нейронной сети 19](#_Toc50752454)

[4.4. Реализация распознавания лиц 22](#_Toc50752455)

[5. ТЕСТИРОВАНИЕ 25](#_Toc50752456)

[5.1. Тестирование нейронной сети 25](#_Toc50752457)

[5.2. Модульное тестирование 25](#_Toc50752458)

[ЗАКЛЮЧЕНИЕ 27](#_Toc50752459)

[ЛИТЕРАТУРА 28](#_Toc50752460)

Введение

**Актуальность темы**

В последние годы популярность искусственных нейронных сетей значительно возросла, они используются для решения множества задач, например: распознавание образов – символы текста, изображения, образцы звуков и т.д; классификация — распределение данных по параметрам; принятие решений и управление; кластеризация – разбиение множества входных сигналов на классы; прогнозирование – предсказание будущего значения некой последовательности на основе нескольких предыдущих значений. Основными достоинствами нейронных сетей являются: решение задач в условиях неопределенности, устойчивость к шумам во входных данных, гибкость структуры нейронных сетей.

В данной работе искусственные нейронные сети будут использованы для распознавания образов, в частности, для распознавания лиц. Технология распознавания лиц в наше время применяется во многих областях, начиная от систем слежения и заканчивая смартфонами и фотоаппаратами. Распознавание лиц на видео с камеры домофона пока что применяется в единичных случаях, однако применение данной технологии в этой области может упростить процесс открытия дверей, а также повысить уровень безопасности жильцов подъезда или частного дома.

**Цель и задачи исследования**

Целью представленной работы является разработка приложения на основе компьютерного зрения и искусственных нейронных сетей для распознавания лиц жильцов дома на видео с камеры домофона. На вход программе подаётся видеофайл, с человеком, входящим в подъезд, по завершению его обработки, программа выдаёт id наиболее вероятного жильца. Для реализации поставленной цели потребуется выполнить следующие подзадачи:

1. провести обзор аналогов и научной литературы;
2. подготовить обучающую и тестовую выборку изображений;
3. программно реализовать и обучить нейронную сеть;
4. разработать и протестировать приложение.

**Структура и объем работы**

Работа включает в себя введение, пять глав, заключение и список литературы. Объем составляет 29 страниц, список литературы насчитывает 22 источника.

В первой главе проводится обзор аналогичных решений, а также анализ существующих библиотек для работы с нейронными сетями и обработки видео и изображений.

Во второй главе представлены источники данных для обучения нейросети, а также приводятся теоретические сведения о искусственных нейронных сетях и их разновидностях топологий, применяющихся для решения поставленной задачи.

В третьей главе описаны функциональные и нефункциональные требования к системе, представлены диаграммы использования и деятельности.

В четвертой главе представлены этапы создания обучающей выборки, программная реализация предобработки входных изображений, модели нейронной сети, распознавания лиц и идентификации человека.

В пятой главе приведены результаты тестирования приложения и нейронной сети.

В заключении описаны полученные в ходе выполнения работы результаты.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ
   1. Обзор аналогичных решений

В настоящее время существуют различные технологии для распознавания лиц на видео и фотографиях, в том числе технологии, работающие в реальном времени. Они встречаются повсеместно, например, в телефонах или социальных сетях. Однако распознавание лиц по видео с камер домофона довольно узкое направление, в России ведущую роль здесь занимает компания Beward [5].

**Домофон Beward**

Домофон Beward обладает встроенной функцией распознавания лиц, которая позволяет осуществлять автоматизированный доступ в помещение без дополнительных устройств определённому кругу посетителей. Функция может работать автономно – без подключения вызывной панели к интернету.

Алгоритм по умолчанию запускается, когда посетитель длительно удерживает кнопку вызова на вызывной панели. В процессе анализа лица вызывная панель выдает гудок, в течении которого посетитель должен стоять неподвижно и смотреть в объектив камеры вызывной панели. По окончании распознавания вызывная панель выдаёт короткий звуковой сигнал и открывает дверь. Если посетителя нет в базе лиц или вызывная панель его не опознала, то звучат два коротких сигнала.

Определённым недостатком является ограничение базы данных лиц. Она может содержать только 30 снимков, при этом для одного пользователя доступно максимум 5 снимков.

Процесс обучения запускается с помощью дополнительного программного обеспечения. После его активации, вызывная панель начинает периодично производить короткие звуковые сигналы. Для захвата лица пользователь должен встать перед вызывной панелью и нажать кнопку вызова. При успешном захвате лица – панель выдаст один кроткий звуковой сигнал, при неудачном – два звуковых сигнала.

* 1. Обзор готовых решений по созданию нейронных сетей

**TensorFlow [6]**

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.

**Keras [7]**

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

**OpenCV [8]**

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.

**Dlib [9]**

Dlib - это современный набор инструментов C ++, содержащий алгоритмы машинного обучения и инструменты для создания сложного программного обеспечения на C ++ для решения реальных проблем, в частности включает алгоритмы поиска лиц. Используется как в промышленности, так и в академических кругах в широком диапазоне областей. Также существует реализация библиотеки на Python.

**Выводы по первой главе**

В этой главе была рассмотрена часть аналогов, реализующих похожие функции. Стоит отметить большое разнообразие программных библиотек для задач машинного обучения с использованием нейронных сетей. Использование подобных библиотек может значительно облегчить задачу реализации. Совокупность озвученных фактов позволяет сделать вывод о актуальности решаемой задачи.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
   1. Источник данных

В данной работе обучение и последующая работа нейронной сети будет проходить на видео, снятом камерой умного домофона компании «Интерсвязь» [10].

«Умный домофон» это новая домофонная система, разработанная для обеспечения максимального комфорта и безопасности жителей многоквартирных домов. Он позволяет принимать звонки от гостей на свой телефон, связываться с управляющей компанией и звонить в экстренные службы с домофонной панели.

Так же компания «Интерсвязь» предоставляет доступ к видео камере в реальном времени и её видеоархивам, для жильцов подъезда в котором установлен домофон.

* 1. Библиотека Face recognition

Библиотека Face recognition [11] для языка программирования Python это простейший инструмент для распознавания лиц, созданный с использованием [современного распознавания лиц dlib,](http://dlib.net/) построенного на основе глубокого обучения.

Библиотека имеет множество функций, например, обнаружение и получение расположения и очертания глаз, носа, рта и подбородка каждого человека; определение человека, изображенного на фотографии.

Face recognition также можно интегрировать в операционную систему и запускать из командной строки для быстрой обработки изображений.

В данной работе будет использована базовая функция библиотеки – нахождение лица на картинке. Пример её работы представлен на рисунке 1.

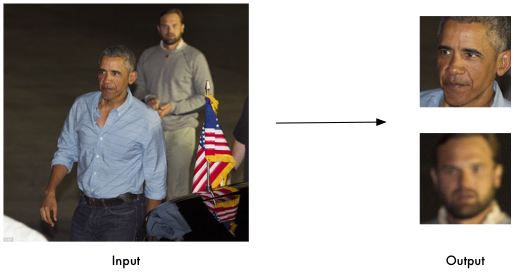


Рис. 1 Пример работы библиотеки Face recognition

* 1. Свёрточная нейронная сеть

Свёрочные нейронные сети [2] повсеместно используются для решения задач распознавания, после успеха этого типа сетей на ImageNet в 2012 году [12].

В её основе лежат некоторые особенности [зрительной коры](https://ru.wikipedia.org/wiki/%D0%97%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D1%80%D0%B0), в которой были открыты так называемые простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток. Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и субдискретизирующих слоёв (слоёв подвыборки).

**Свёрточный слой**

В свёрточном слое используется ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою, формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов, которую также называют ядром свертки. Особенностью свёрточного слоя является сравнительно небольшое количество параметров, устанавливаемое при обучении. Так, например, если исходное изображение имеет размерность 100×100 пикселей по трём каналам (это значит 30000 входных нейронов), а свёрточный слой использует фильтры c ядром 3x3 пикселя с выходом на 6 каналов, тогда в процессе обучения определяется только 9 весов ядра, однако по всем сочетаниям каналов, то есть 9×3×6 =162, в таком случае данный слой требует нахождения только 162 параметров, что существенно меньше количества искомых параметров полносвязной нейронной сети. Один из этапов процесса свертки представлен на рисунке 2.

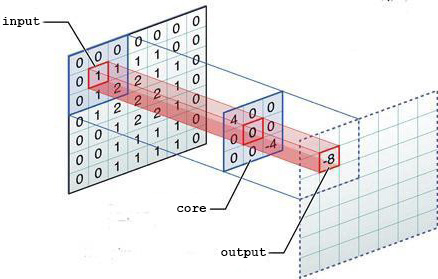


Рис. 2 Пример обработки входного слоя ядром свертки

**Слой субдискретизации**

Слой пулинга (иначе подвыборки, субдискретизации) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель, имеющий максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объём изображения. Пулинг интерпретируется так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не [переобучаться](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5). Слой пулинга, как правило, вставляется после слоя свёртки перед слоем следующей свёртки. Наглядно преобразование можно увидеть на рисунке 3.

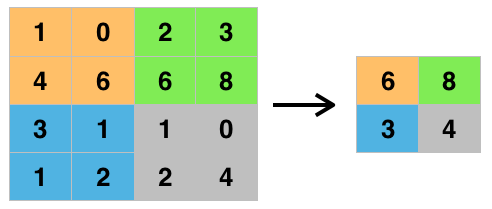


Рис. 3 Процесс подвыборки функцией максимума и фильтром 2×2 с шагом 2

* 1. Сиамская сеть

Сиамская нейросеть [13] — один из простейших и наиболее популярных алгоритмов однократного обучения. Методики, при которой для каждого класса берётся лишь по одному учебному примеру. Таким образом, сиамская сеть обычно используется в приложениях, где в каждом классе есть не так много единиц данных.

По сути, сиамские сети состоят из двух симметричных нейросетей, с одинаковыми весами и архитектурой, которые в конце объединяются и используют функцию энергии — E. Пример простой архитектуры сиамской сети представлен на рисунке 4.



Рис. 4 Пример архитектуры сиамской сети.

**Выводы по второй главе**

В данной главе был представлен источник данных для обучения и тестирования нейросети, а также разобран теоретический аспект работы сверточных нейронных сетей, включая информацию о работе различных слоев её архитектуры. Описан один из алгоритмов однократного обучения – сиамская нейросеть, и представлена её простейшая архитектура.

1. ПРОЕКТИРОВАНИЕ
   1. Функциональные требования

Функциональные требования к разрабатываемому приложению описывают то, как она должна себя вести. Система после разработки должна удовлетворять следующим функциональным требованиям:

1. система должна принимать на вход видео в формате .mp4;
2. система должна выводить рамку вокруг распознанного лица;
3. после обработки входного видео файла, система должна вывести номер наиболее вероятного жильца.
   1. Нефункциональные требования

Нефункциональные требования включают в себя те свойства, которые имеются у приложения. Определяют критерии, которые могут 20 использоваться для оценки работы системы, а не для конкретного поведения. Список нефункциональных требований представлен ниже:

1. система должна быть реализована на языке Python 3.7;
2. система должна использовать библиотеку Keras в связке с TensorFlow для работы с нейронными сетями;
3. система должна использовать библиотеку OpenCV для работы с видеофайлами.
   1. Варианты взаимодействия с системой

Для проектирования системы был использован UML (united modeling language) – универсальный язык моделирования для разработки в области программного обеспечения [14], предназначенным для обеспечения стандартного способа визуализации проектирования системы. Была построена модель взаимодействия актера с приложением распознавания лица. Диаграмма вариантов использования представлена на рисунке 5.

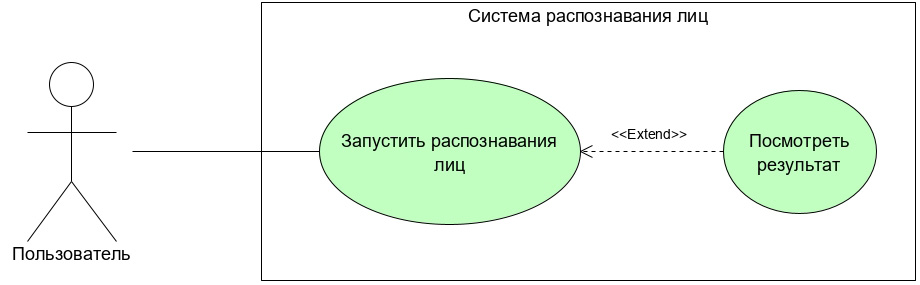


Рис. 5 Диаграмма вариантов использования

**Актеры, участвующие в взаимодействии с приложением:**

Пользователь – человек, который использует функционал системы.

**Описание вариантов взаимодействия:**

1. запустить процесс распознавания введя в консоль расположение видеофайла;
2. посмотреть результаты распознавания консоли.
   1. Диаграмма деятельности

На основе функциональных и нефункциональных требований была разработана диаграмма деятельности, показанная на рисунке 6. На данной диаграмме изображен процесс взаимодействия пользователя с системой распознавания лиц. После запуске приложения пользователь вводит имя видеофайла. Затем видеопоток разделяется на отдельные кадры. Следом на каждом кадре ищутся лица и личности идентифицируются. Дальше происходит обработка изображения – лица на отмечаются красным квадратом. После окончания видеофайла выводится идентификационный номер наиболее вероятного человека на данном видео.

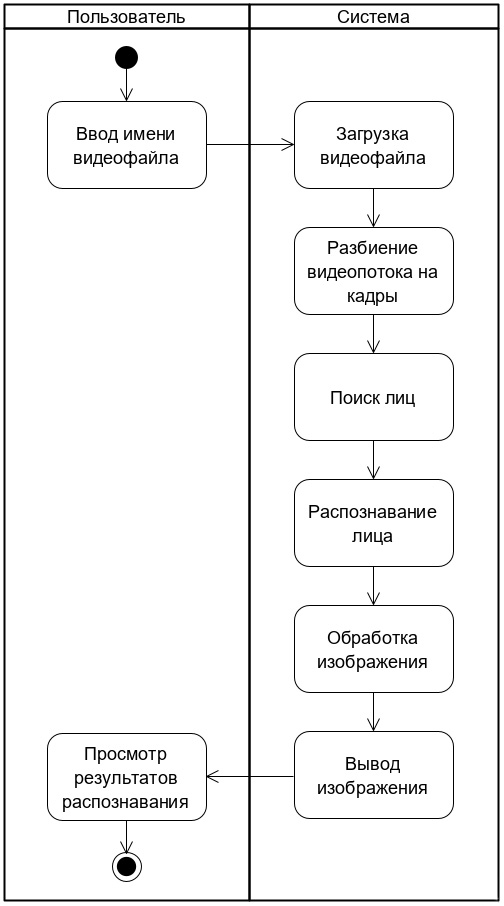


Рис. 6 Диаграмма деятельности

**Выводы по третьей главе**

В данном разделе были сформулированы функциональные и нефункциональные требования к системе, созданы диаграмма вариантов использования и диаграмма деятельности. Так же спроектирована система распознавания лиц на видео, реализация которой представлена в следующей главе.

1. РЕАЛИЗАЦИЯ
   1. Средства разработки

Для реализации программной части был использован язык программирования Python версии 3.7 [15], использовалась среда разработки JetBrains PyChram Professional 2018.3.6 [16]. Обучение и подготовка данных для обучения нейросети велись в JupyterNotebook 5.7.4 [20].

Для языка Python были использованы следующие библиотеки: OpenCV 4.4.0 [8], Numpy 1.18.5 [18], Pillow 5.3.0 [17], Keras 2.4.3 [7], Tensorflow 2.3.0 [6], Face Recognition 1.3.0 [11].

Для подготовки видеофайлов для обучения использовалась программа для захвата видео с экрана Bandicam 4.6.2 [21].

* 1. Формирование обучающей выборки

Как уже говорилось выше обучение нейросети будет проходить на видео, снятым камерой умного домофона компании «Интерсвязь». Компания предоставляет доступ к камерам перекрёстков, а также домофонов на своём сайте [19], представленном на рисунке 7.

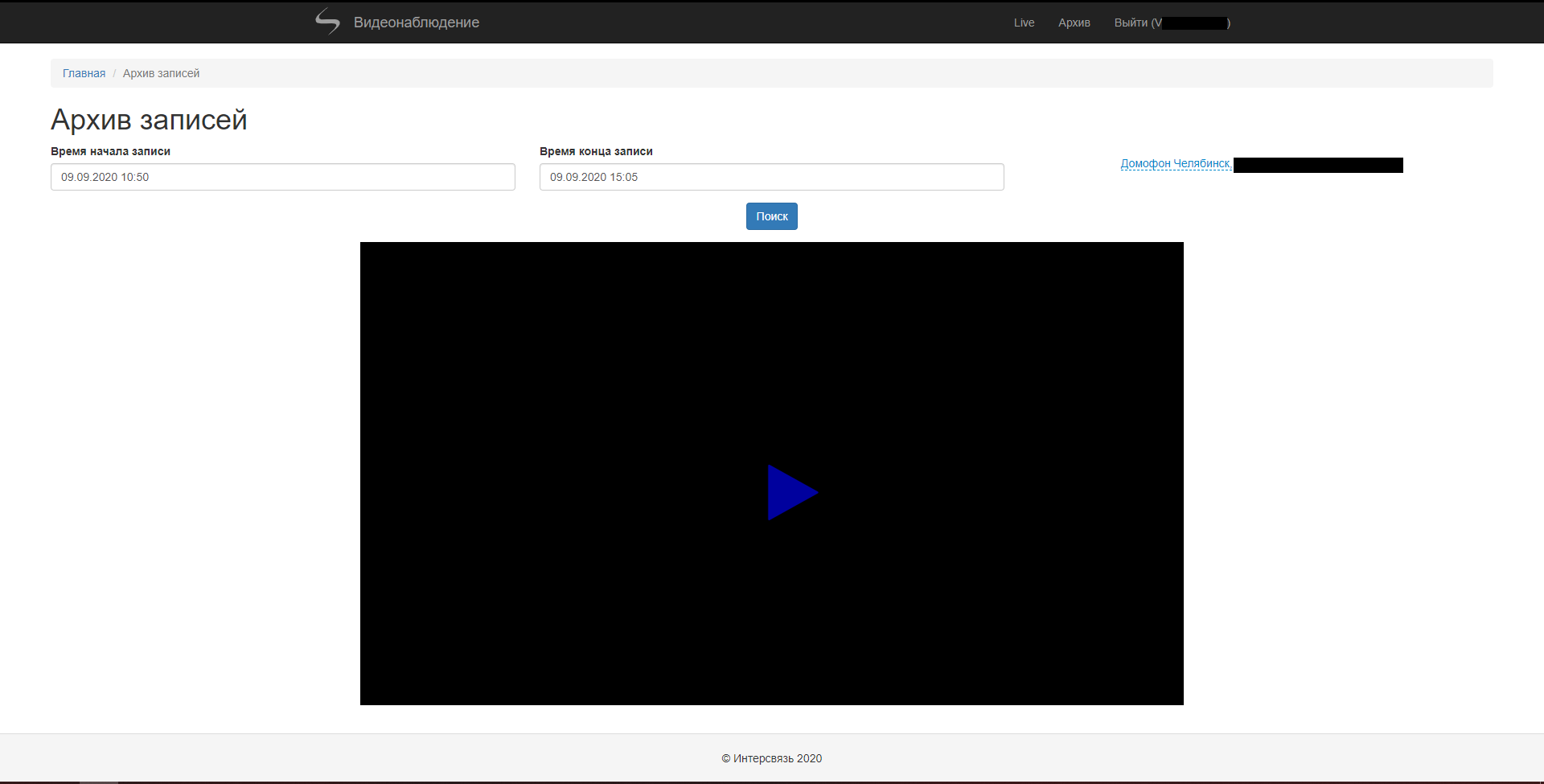


Рис. 7 Архив записей с камер компании «Интерсвязь»

К сожалению, для камер домофонов функция скачивания архива отсутствует, вследствие этого была использована программа для захвата видео Bandicam.

После просмотра двух световых дней было получено 92 видеоролика с 59 различными людьми, входящими в подъезд.

**Получение изображений лиц**

На рисунке 8 представлен цикл обработки видеофайла с распознаванием лиц и сохранением их в отдельной директории. Поскольку после захвата видео с рабочего стола его разрешение получилось больше чем оригинальное разрешение видеокамеры, изображение уменьшается перед использованием на нём распознавания, это также позволяет значительно ускорить процесс обработки видео.

**while True**:  
 *#Получение одного кадра видео* exist, frame = video\_capture.read()  
 **if not** exist:  
 **break** small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)  
  
 *#Конвертация изображения из BGR формата (используемого OpenCV)*

*#в RGB формат (для использования в библиотеке face\_recognition)* rgb\_frame = small\_frame[:, :, ::-1]  
  
 *# Нахождение всех лиц в текущим кадре видео* face\_locations = face\_recognition.face\_locations(rgb\_frame)  
  
 *# Сохранение результата* **for** top, right, bottom, left **in** face\_locations:  
 top \*= 4; right \*= 4; bottom \*= 4; left \*= 4  
 count += 1  
 cv2.imwrite(curDir + **"/frame%d.jpg"** % count, frame[top:bottom, left:right])

Рис. 8 Распознавание и сохранение лиц

В результате обработки видеофайлов данным алгоритмом осталось 26 людей, чьё лицо было распознано более 20 раз. Так получилось из-за низкого качества видео, не на всех из них библиотеке face recognition удалось распознать лица, пример такого кадра представлен на рисунке 9.



Рис. 9 Пример нераспознанного кадра

**Предобработка изображений**

Полученные в результате предыдущей обработки изображения получились разного размера, для того чтобы их можно было использовать для обучения нейронной сети нужно привести их к стандартному размеру.

На рисунке 10 представлен алгоритм для выборки 20 случайных изображений и приведения их к единому размеру 100х100 пикселей. Сначала изменяется ширина изображения с сохранением пропорций. После если высота изображения больше 100 пикселей оно обрезается снизу, если меньше – остаток заполняется белым цветом. Также в процессе изображение конвертируется из цветного в оттенки серого.

**for** i **in** range(setSize):  
 ind = np.random.randint(len(images))  
  
 *#Изменение ширины с сохнанением пропорции* img = Image.open(directory + catalog + **'/'** +

images[ind]).convert(**'L'**)  
 percent = (width / float(img.size[0]))  
 hsize = int((float(img.size[1]) \* float(percent)))  
 img = img.resize((width, hsize), Image.ANTIALIAS)  
  
 background = Image.new(**'L'**, (width, width))  
 background.paste(img)  
 background.save(saveDir + str(curCatalog) + **'/%d.jpg'** % (i + 1))  
  
 **del** images[ind]

Рис. 10 Стандартизация изображений

Результат работы данного алгоритма представлен на рисунке 11.



Рис. 11 Пример обучающей выборки для одного человека

* 1. Обучение нейронной сети

Для данного проекта была выбрана модель сиамской нейронной сети [3]. Сиамским сетям нужно подавать на вход парные значения с маркировками, поэтому создадим такие наборы. Возьмём из одной папки две случайные фотографии и пометим как «подлинную» пару (genuine). Затем возьмем две фотографии из разных папок и пометим как «ложную» пару (imposite). Код создания «подлинных» и «ложных» пар представлен на рисунке 12.

Было сгенерировано 10000 «подлинных» и 10000 «ложных» пар. Из них 15000 были использованы для обучения и 5000 для тестирования нейросети.

*# инициализация numpy массива с размерами [total\_sample, no\_of\_pairs, dim1, dim2]  
# dim1, dim2 - размеры исходного изображения*x\_geuine\_pair = np.zeros([total\_sample\_size, 2, dim1, dim2, 1])y\_genuine = np.zeros([total\_sample\_size, 1])  
  
**for** i **in** range(26):  
 **for** j **in** range(int(total\_sample\_size / 26)):  
 ind1 = 0  
 ind2 = 0  
  
 **while** ind1 == ind2:  
 ind1 = np.random.randint(20)  
 ind2 = np.random.randint(20)  
  
 *# считать два изображения как numpy массивы* img1 = read\_image(**'data/set/'** + str(i + 1) + **'/'** + str(ind1 + 1) + **'.jpg'**)  
 img2 = read\_image(**'data/set/'** + str(i + 1) + **'/'** + str(ind2 + 1) + **'.jpg'**)  
  
 *# сохранить изображения в инициализированном numpy массиве* x\_geuine\_pair[count, 0, :, :, 0] = img1  
 x\_geuine\_pair[count, 1, :, :, 0] = img2  
  
 *# для изображений из одной директории значение равно 1. (подлинная пара)* y\_genuine[count] = 1  
 count += 1  
  
count = 0  
x\_imposite\_pair = np.zeros([total\_sample\_size, 2, dim1, dim2, 1])  
y\_imposite = np.zeros([total\_sample\_size, 1])  
  
**for** i **in** range(int(total\_sample\_size / 20)):  
 **for** j **in** range(20):  
  
 *# считывание изображений из разных директорий (ложные пары)* **while True**:  
 ind1 = np.random.randint(26)  
 ind2 = np.random.randint(26)  
 **if** ind1 != ind2:  
 **break** img1 = read\_image(**'data/set/'** + str(ind1 + 1) + **'/'** + str(j + 1) + **'.jpg'**)  
 img2 = read\_image(**'data/set/'** + str(ind2 + 1) + **'/'** + str(j + 1) + **'.jpg'**)  
  
 x\_imposite\_pair[count, 0, :, :, 0] = img1  
 x\_imposite\_pair[count, 1, :, :, 0] = img2  
  
 *# для изображений из разных директорий значение равно 0. (ложные пары)* y\_imposite[count] = 0  
 count += 1  
  
*# объединение подлинных и ложных пар для получения общего массива данных*X = np.concatenate([x\_geuine\_pair, x\_imposite\_pair], axis=0) / 255  
Y = np.concatenate([y\_genuine, y\_imposite], axis=0)

Рис. 12 Подготовка массива данных для обучения и тестирования

**Реализация нейронной сети**

Так как сиамская нейронная сеть состоит из двух одинаковых сетей, для начала надо создать базовую сеть. Её код представлен на рисунке 13, это свёрточная нейросеть для извлечения свойств состоящая из двух свёрточных слоёв с функцией активации ReLU и слоем с определением максимального значения после flat-слоя.

**def** build\_base\_network(input\_shp):  
  
 seq = Sequential()  
 nb\_filter = [64, 128]  
 kernel\_size = 3  
  
 *# convolutional layer 1* seq.add(Convolution2D(nb\_filter[0], kernel\_size, kernel\_size,  
input\_shape=input\_shp, padding=**'valid'**))  
 seq.add(Activation(**'relu'**))  
 seq.add(MaxPooling2D(pool\_size=(2, 2)))  
 seq.add(Dropout(.25))  
  
 *# convolutional layer 2* seq.add(Convolution2D(nb\_filter[1], kernel\_size, kernel\_size,  
padding=**'valid'**))  
 seq.add(Activation(**'relu'**))  
 seq.add(MaxPooling2D(pool\_size=(2, 2)))  
 seq.add(Dropout(.25))  
seq.add(Flatten())  
 seq.add(Dense(128, activation=**'relu'**))  
 seq.add(Dropout(0.1))  
 seq.add(Dense(50, activation=**'relu'**))  
   
 **return** seq

Рис. 13 Функция создания базовой нейронной сети

После прохода через нейросети, выходные данные нужно сравнить, для этого воспользуемся функцией евклидового расстояния, её код представлен на рисунке 14.

**def** euclidean\_distance(vects):  
 x, y = vects  
 **return** K.sqrt(K.maximum(K.sum(K.square(x - y), axis=1, keepdims=**True**), K.epsilon()))

Рис. 14 Функция евклидового расстояния

Код создания модели сиамской нейронной сети представлен на рисунке 15. Функция принимает на вход размер обрабатываемых изображений, в данном случае 100х100 пикселей.

**def** create\_model(input\_dim):  
 img\_a = Input(shape=input\_dim)  
 img\_b = Input(shape=input\_dim)  
  
 base\_network = build\_base\_network(input\_dim)  
 feat\_vecs\_a = base\_network(img\_a)  
 feat\_vecs\_b = base\_network(img\_b)  
  
 distance = Lambda(euclidean\_distance, output\_shape=eucl\_dist\_output\_shape)([feat\_vecs\_a, feat\_vecs\_b])  
  
 rms = RMSprop()  
 model = Model([img\_a, img\_b], distance)  
 model.compile(loss=contrastive\_loss, optimizer=rms)  
  
 **return** model

Рис. 15 Функция создания модели сиамской нейронной сети

После обучения нейронной сети её веса сохраняются в отдельном файле формата HDF5 [22], для дальнейшего использования в основной программе.

* 1. Реализация распознавания лиц

Так как в данном проекте используется сиамская нейросеть, для начала надо подготовить массив данных с которым будут сравниваться распознанные лица. Код его реализации представлен на рисунке 16. Для каждого человека выбирается 3 случайных изображения и сохраняются в общем массиве.

personalCnt = 26  
dim = 100  
personalBase = np.zeros([personalCnt, 3, dim, dim, 1])  
**for** i **in** range(personalCnt):  
 **for** j **in** range(3):  
 ind = np.random.randint(20)  
 img\_load = read\_image(**'data/set/'** + str(i + 1) + **'/'** + str(ind + 1) + **'.jpg'**)  
 personalBase[i, j, :, :, 0] = img\_load  
personalBase /= 255

Рис. 16 Создание массива для сравнения распознанных лиц

После того как пользователь вводит название видеофайла, в цикле представленном на рисунке 17 происходит покадровая обработка изображения.

**while True**:  
 *# Получение одного кадра видео* exist, frame = video\_capture.read()  
 **if not** exist:  
 **break** small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)  
  
 *# Конвертация изображения из BGR формата (используемого OpenCV) в RGB формат (для использования в библиотеке face\_recognition)* rgb\_frame = small\_frame[:, :, ::-1]  
  
 *# Нахождение всех лиц в текущим кадре видео* face\_locations = face\_recognition.face\_locations(rgb\_frame)  
  
 **for** top, right, bottom, left **in** face\_locations:  
 top \*= 4; right \*= 4; bottom \*= 4; left \*= 4  
  
 *# Приведение найденного лица к нужному размеру и формату данных* curImg = Image.fromarray(frame[top:bottom, left:right])  
 curImg = curImg.convert(**'L'**)  
 percent = (100 / float(curImg.size[0]))  
 hsize = int((float(curImg.size[1]) \* float(percent)))  
 curImg = curImg.resize((100, hsize), Image.ANTIALIAS)  
 background = Image.new(**'L'**, (100, 100))  
 background.paste(curImg)  
  
 *# Создание массива для сравнения найденного лица* curBase = np.zeros([personalCnt, 1, dim, dim, 1])  
 **for** i **in** range(personalCnt):  
 curBase[i, 0, :, :, 0] = np.array(background)  
 curBase /= 255  
  
 *# Классификация полученного лица нейросетью* pred = np.zeros([personalCnt, 20])  
 **for** i **in** range(3):  
 p = model.predict([personalBase[:, i], curBase[:, 0]])  
 pred[:, i] = p.reshape(26)  
  
 *# Сохранение 3х наиболее вероятных вариантов в массив со статистикой* res = np.mean(pred, axis=1)  
 personId = np.argmin(res)  
 stat[personId] += 0.5  
 res[personId] = 100  
 personId = np.argmin(res)  
 stat[personId] += 0.3  
 res[personId] = 100  
 personId = np.argmin(res)  
 stat[personId] += 0.2  
  
 *# Отрисовка рамки вокруг распознанного лица* cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)  
  
 *# Отрисовка обработанного изображения* cv2.imshow(**'Video'**, cv2.resize(frame, (0, 0), fx=0.5, fy=0.5))

Рис. 17 Классификация распознанных лиц

Тут также используется библиотека face recognition для распознавания лица на изображении. После чего фрагмент изображения, в котором находится лицо проводится к требуемому размеру 100х100 пикселей и конвертируется в массив данных, который обрабатывается нейросетью. Три наиболее вероятных ответа нейросети добавляются в общую статистику. После чего вокруг распорного лица отрисовывается рамка и обработанное изображение выводится на экран.

После завершения обработки всего видеофайла программа выводит в консоль номер наиболее вероятного человека, код данной операции представлен на рисунке 18.

print(**"Наиболее вероятный Id: "** + str(np.argmax(stat) + 1))

Рис. 18 Заключительный вывод программы

**Выводы по четвертой главе**

В данной главе были описаны используемые средства разработки, процесс создания датасета, а также реализация нейронной сети и программный код самого приложения. Разработанное приложение полностью соответствует ранее упомянутым требованиям.

1. ТЕСТИРОВАНИЕ
   1. Тестирование нейронной сети

**Тестирование на изображениях**

После обучения нейронной сети было проведено тестирование на оставшихся 5000 пар изображений, включающих в себя, как «подлинные» пары, так и «ложные». Результат тестирования показал точность нейронной сети 98,1%.

**Тестирование на видео**

После реализации модуля для классификации лиц, было проведено тестирование на видеофайлах, для этого было выбрано 10 случайных роликов из которых вырезались изображения для обучения нейросети.

В результате первого тестирования программа показала точность 80%. После чего было проведено повторное обучение нейронной сети с увеличением количества итераций. После подбора оптимального количества итераций обучения нейронной сети была получена 100% точность классификации людей на тестовой выборке видеофайлов.

* 1. Модульное тестирование

Модульное тестирование, или юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы. Результаты модульного тестирования приведены в таблице 1.

Табл. 1 Результаты модульного тестирования

| **№** | **Название модуля** | **Назначение модуля** | **Ожидаемый результат** | **Результат теста** |
| --- | --- | --- | --- | --- |
| 1 | Face\_cutter | Модуль предназначен для вырезки и сохранения изображений лиц из видеофайла | Набор изображений лиц для каждого распознанного человека | Пройден |
| 2 | Img\_convector | Модуль предназначен для создания обучающей выборки для каждого человека | Набор из 20 изображений одинакового размера, в оттенках серого для каждого человека | Пройден |
| 3 | My\_model | Модуль предназначен для создания модели сиамской нейронной сети | Готовая к обучению или использования модель нейронной сети | Пройден |
| 4 | Train | Модуль предназначен для превращения готовых наборов изображений в массив данных, обучения и тестирования нейронной сети | Вывод результатов тестирования обученной нейросети, сохранение её весов в отдельный файл | Пройден |
| 5 | Face\_recognizer | Модуль предназначен для распознавания и классификации лиц на видеофайле | Виде с обведёнными распознанными лицами, Id наиболее вероятного человека после обработки всего видео | Пройден |

**Вывод по пятой главе**

В данной главе было произведено тестирование работы нейронной сети, входе него была увеличена точность классификации на видеофайлах. Также было проведено модульное тестирование, в ходе которого нарушений в работе приложения не выявлено.

Заключение

В рамках данной работы была спроектировано и реализовано приложение для распознания жильцов, входящих в подъезд, с применением нейросетевых технологий, а также было проведено тестирование системы и сиамской нейронной сети.

В ходе работы были решены следующие задачи:

1. проведён обзор аналогов и научной литературы;
2. подготовлена обучающая и тестовая выборка изображений;
3. программно реализована и обучена нейронная сеть;
4. разработано и протестировано приложение.

Также были получены навыки программирования на языке Python, разработки нейронных сетей с помощью фреймворка Tensorflow, обработки видео с помощью библиотеки OpenCV, обработки изображений с помощью библиотеки Pillow.

литература

1. Хайкин С. Нейронные сети: полный курс. 2-е изд. Москва: ООО "И.Д. Вильямс", 2017. 1104 с.
2. LeCun Y., Bengio Y. Convolution Networks for Images, Speech, and Time-Series // Igarss 2014, No. 1, 1998. pp. 1-5.
3. Распознавание лиц с помощью сиамских сетей / Блог компании Инфосистемы Джет / Хабр [Электронный ресурс] URL: https://habr.com/ru/company/jetinfosystems/blog/465279/ (дата обращения: 11.09.2020).
4. 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.
5. Системы видеонаблюдения Бевард [Электронный ресурс] URL: <https://www.beward.ru/> (дата обращения: 11.09.2020).
6. Документация TensorFlow [Электронный ресурс] URL: <https://www.tensorflow.org/api_docs> (дата обращения: 11.09.2020).
7. Официальный сайт пакета Keras [Электронный ресурс] URL: https://keras.io/ (дата обращения: 11.09.2020).
8. Официальный сайт пакета OpenCV [Электронный ресурс] URL: https://opencv.org/ (дата обращения: 11.09.2020).
9. Документация пакета dlib [Электронный ресурс] URL: http://dlib.net/python/index.html (дата обращения: 11.09.2020).
10. Умный домофон компании «Интерсвязь» [Электронный ресурс] URL: https://www.is74.ru/home/services/smart\_domofon/ (дата обращения: 11.09.2020).
11. GitHub ageitgey/face\_recognition: The world's simplest facial recognition api for Python and the command line [Электронный ресурс] URL: https://github.com/ageitgey/face\_recognition (дата обращения: 11.09.2020).
12. Large Scale Visual Recognition Challenge [Электронный ресурс] URL: http://image-net.org/challenges/LSVRC/2012/results.html#t1 (дата обращения: 11.09.2020).
13. 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.
14. Официальный сайт UML [Электронный ресурс] URL: <https://www.uml.org> (дата обращения: 11.09.2020).
15. Официальный сайт Python [Электронный ресурс] URL: https:// www.python.org/ (дата обращения: 11.09.2020).
16. Официальный сайт PyCharm [Электронный ресурс] URL: https:// www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 11.09.2020).
17. Официальный сайт пакета Pilllow [Электронный ресурс] URL: 46 https://pillow.readthedocs.io/en/stable/ (дата обращения: 11.09.2020).
18. Документация NumPy [Электронный ресурс] URL: http:// docs.scipy.org/doc/numpy/ (дата обращения: 11.09.2020).
19. Архив видеозаписей компании «Интерсвязь» [Электронный ресурс] URL: https://cams.is74.ru/archive (дата обращения: 11.09.2020).
20. Официальный сайт Jupyter Notebook [Электронный ресурс] URL: https://jupyter.org/ (дата обращения: 11.09.2020).
21. Официальный сайт Bandicam [Электронный ресурс] URL: https://www.bandicam.com/ru/ (дата обращения: 11.09.2020).
22. Документация HDF5 для Python [Электронный ресурс] URL: https://docs.h5py.org/en/stable/ (дата обращения: 11.09.2020).