МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Крымский федеральный университет имени В.И. Вернадского»

Таврический колледж

(структурное подразделение)

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

на тему

**СОЗДАНИЕ СИСТЕМЫ РАСПОЗНАВАНИЯ ЛИЦ С ИСПОЛЬЗОВАНИЕМ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ**

Обучающийся: **ВОЛОЧАЙ ИГОРЬ АЛЕКСАНДРОВИЧ \_\_\_\_\_\_\_\_\_\_\_\_\_**

Специальность: 09.02.01 Компьютерные системы и комплексы

Группа 1КСК21

Научный руководитель: Стацевич Анна Викторовна \_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| **К защите допустить**  Преподаватель дисциплины  Основы  научно-исследовательской деятельности  М.Н. Щербакова | Индивидуальный проект  выполнен на оценку  Волочая Игоря Александровича  Дата защиты «\_\_\_» \_\_\_\_\_\_\_ 2022 г. |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc104138311)

[ГЛАВА 1 ТЕОРИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ 7](#_Toc104138312)

[1.1 Сравнение биологической и искусственной нейронной сети 7](#_Toc104138313)

[1.2 Изучение работы многослойных нейронных сетей 10](#_Toc104138314)

[1.3 Градиентный спуск. Методы вычисления ошибки нейронной сети 12](#_Toc104138315)

[1.4 Обучение модели. Метод обратного распространения ошибки 14](#_Toc104138316)

[1.5 Представление графической информации в виде, пригодном для работы модели искусственной нейронной сети 16](#_Toc104138317)

[Вывод к 1 главе 18](#_Toc104138318)

[ГЛАВА 2 РАЗРАБОТКА СИСТЕМЫ РАСПОЗНОВАНИЯ ЛИЦ С ИСПОЛЬЗОВАНИЕМ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ 19](#_Toc104138319)

[2.1 Подготовка датасетов для обучающей и тестовой выборки 19](#_Toc104138320)

[2.2 Описание работы нейронной сети 22](#_Toc104138321)

[2.3 Выбор языка программирования и среды разработки программы. Установка и описание библиотек и модулей 25](#_Toc104138322)

[2.4 Разработка системы распознавания лиц с использованием глубоких нейронных сетей 29](#_Toc104138323)

[2.5 Обучение модели глубокой нейронной сети. Тест работоспособности. Приведение гипотез по улучшению работоспособности 33](#_Toc104138324)

[Вывод ко 2 главе 35](#_Toc104138325)

[ЗАКЛЮЧЕНИЕ 36](#_Toc104138326)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc104138327)

[ПРИЛОЖЕНИЕ А 41](#_Toc104138328)

[ПРИЛОЖЕНИЕ Б 42](#_Toc104138329)

[ПРИЛОЖЕНИЕ В 43](#_Toc104138330)

[ПРИЛОЖЕНИЕ Г 44](#_Toc104138331)

[ПРИЛОЖЕНИЕ Д 46](#_Toc104138332)

[ПРИЛОЖЕНИЕ Е 47](#_Toc104138333)

[ПРИЛОЖЕНИЕ Ж 48](#_Toc104138334)

# ВВЕДЕНИЕ

Первые электронно-вычислительные устройства появились в середине сороковых годов прошлого века. Парадигмы построения программ того времени строились на принципе императивного программирования. Данные программы должны работать по определённому последовательному алгоритму, при котором все вероятные входные, промежуточные и выходные данные будут заведомо известны.

В 1958 году Фрэнк Розенблатт публикует статью «The perceptron: а probabilistic model for information storage and organization in the brain» («Перцептрон: вероятная модель хранения и организации информации в головном мозге») [1]. В этой статье впервые описана модель работы перцептрона - математической модели восприятия информации биологическим мозгом. Именно перцептрон положил начало развитию направления изучения и разработки машинного обучения. Программы, использующие машинное обучение не нуждаются в точном последовательном алгоритме решения задач, так как для их работы необходимы некие входные, и желаемые выходные данные. Все необходимые взаимосвязи и решения, программа определяет самостоятельно, без участия человека.

Алгоритмы машинного обучения на сегодняшний день применяются во многих отраслях, таких как: медицина и фармацевтика, логистика, производственная отрасль, банки и бизнес, маркетинг и др. [2, 3]. Технологии машинного обучения способны многократно сократить расходы и увеличить производительность в отдельных аспектах монотонного труда.

Одной из областей машинного обучения является технология компьютерного зрения. Данная технология связана с обработкой графической информации (изображение и видео). Данная технология получила большое развитие в начале 21 века. Связано это с ростом вычислительных мощностей микроэлектроники, а также с уменьшением размеров микрочипов и интегральных схем.

По причине такого роста, компьютерное зрение стали применять во многих отраслях производства, управления и быта. На сегодняшний день, многие конвейеры на фабриках и заводах оснащены системами компьютерного зрения, с целью обнаружения бракованных изделий. Помимо этого, технология применяется в транспортных системах, с целью обнаружения дорожно-транспортных происшествий и дорожных правонарушений, в медицине, автопилотах и системах помощи водителям и др. [4].

Так же, системы компьютерного зрения нужны для распознавания лиц на изображении с последующей идентификацией личности. Подобный процесс необходим в любом паспортном контроле для подтверждения личности, например в банке при выдаче кредита, на любом контрольно- пропускном пункте, а также в качестве системы блокировки. Многие отечественные и зарубежные банки уже давно начали использовать биометрические данные клиентов (в том числе и данные формы лица) для подтверждения их личности [5]. Однако, подобная задача является монотонной и однотипной для человека. Компьютерная система может справиться с этой задачей быстрее и точнее, чем человек, при этом, система может работать круглосуточно и без перерывов, что доказывает актуальность темы.

Современные системы распознавания и идентификации лиц разрабатываются частными компаниями, из-за чего исходный код таких программ невозможно увидеть и изменить под свои нужды, в связи с этим актуальной является разработка системы распознавания лиц.

Цель проекта: изучить и разработать систему распознавания лиц на изображении при помощи глубоких нейронных сетей.

Объект исследования: модель глубокой нейронной сети «классификатора».

Предмет исследования: создание и обучение глубокой нейронной сети «классификатора».

Задачи:

1. описать базовые принципы работы глубоких нейронных сетей, методы их обучения, а также базовые принципы работы компьютерного зрения;
2. собрать и обработать тренировочные и тестовые данные для обучения нейросети и последующей оценки работоспособности;
3. проанализировать принципы работы сторонних модулей и библиотек для разработки;
4. создание, обучение и тестирование модели глубокой нейронной сети «классификатора»;
5. достижение работоспособности системы с точностью свыше 95%.

Методы, применяемые в ходе исследования: поисковый, анализ информации, описание, эксперимент, моделирование, тестирование.

Данная научно-исследовательская работа состоит из введения, 1 и 2 главы, заключения и списка использованных источников. Также включает в себя 48 страниц, 23 рисунка, 1 таблица, 27 источников литературы.

# ГЛАВА 1 ТЕОРИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

## Сравнение биологической и искусственной нейронной сети

Искусственные нейронные сети были созданы по подобию нейронных связей биологического мозга живых существ. Электрический заряд, возникающий из-за какого-то возбуждения в организме, поступает от дендритов, в тело нейрона, что представлено на (Рисунок 1). Далее, если общего заряда, полученного через дендриты, хватает, то нейрон возбуждается, и передаёт уже свой заряд по аксону в синапсы, а синапсы передают заряд к другим нейронам. Наш мозг в среднем состоит из 80 млрд. нейронов. Каждый нейрон по отдельности - малоэффективен, но создавая нейронные сети, наш мозг способен на логическое мышление, прогнозирование, выведение закономерностей, запоминание чего-либо.

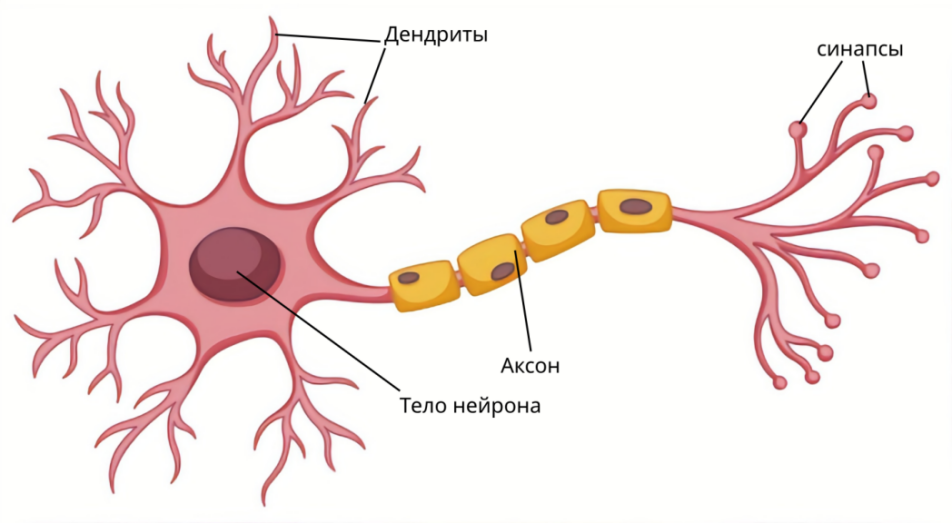


Рисунок 1 - Строение биологического нейрона

Искусственные нейронные сети работают точно по такому же принципу, что и биологические нейронные сети [6]. Простейшую искусственную нейронную сеть можно представить, как связь 2 отдельных нейронов. Таким образом мы получаем входной и выходной нейрон. Во входной нейрон мы подаём число (аналог заряда, передаваемого через дендриты вследствие возбуждения), после чего это число перемещается по связям в другой нейрон следующего слоя. При перемещении, число умножается на определённый весовой коэффициент. От значений весовых коэффициентов будет зависеть итоговый ответ любой искусственной нейронной сети.

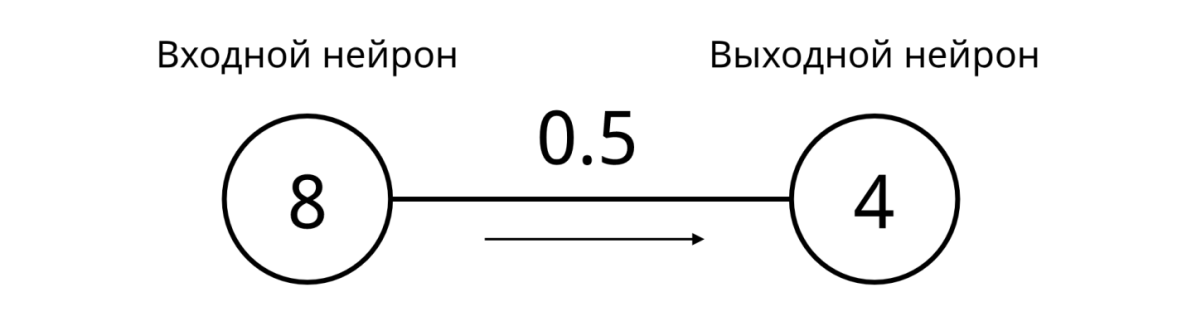


Рисунок 2 - Пример простейшей нейронной сети с двумя слоями

Каждый искусственный нейрон имеет функцию активации, аналогичную порогу возбуждения биологического мозга. В примере нейронной сети (Рисунок 2), функция активации имеет линейный вид уравнения

(1)

где - функция активации;

- значение суммы входящих сигналов в нейроне.

На сегодняшний день существует большое количество эффективных функций активации [7]. Каждая функция активации подбирается с учётом условий задачи и сферой применения нейронной сети. Самые распространённые функции активации: линейная, ступенчатая, линейный выпрямитель (ReLU), логистическая (сигмоида) (таблица 1)

|  |  |  |  |
| --- | --- | --- | --- |
| 180px-Activation_identity.svg  Линейная | 180px-Activation_binary_step.svg  Ступенчатая | 180px-Activation_rectified_linear.svg  ReLU | 180px-Activation_logistic.svg  Sigmoid |

Таблица 1 - Функции активации и их графики

В любом слое нейронной сети может быть неограниченное количество нейронов, следовательно, неограниченное количество связей с каждым нейроном, из-за чего появляется необходимость в суммировании подаваемых на нейрон входных числовых значений.

(2)

где - итоговое значение суммы;

xi - входные сигналы, совокупность всех входных сигналов нейрона образует вектор x.

Исходя из всей вышеописанной информации, простейшую нейронную сеть, состоящую из 2 полносвязных слоёв можно представить как математическую модель, имитирующую работу биологической нейронной сети

(3)

где - результат работы нейронной сети;

()- функция активации нейрона;

wi - весовые коэффициенты, совокупность весовых коэффициентов образует вектор весов w;

xi - входные сигналы, совокупность всех входных сигналов нейрона образует вектор x.

## Изучение работы многослойных нейронных сетей

Простейший двухсловный перцептрон способен решать узкий круг примитивных задач. В большинстве случаев для решения реальных задач с использованием машинного обучения применяют многослойные нейронные сети. Подобные сети имеют помимо входного и выходного слоя, определённое количество «скрытых» слоёв (рисунок 3). В случае стандартных полносвязных нейронных сетей, подобные слои находятся между входным и выходным слоем. Из-за невозможности повлиять на начальное значение в таких нейронах их назвали скрытыми.



Рисунок 3 - Пример строения полносвязной многослойной нейронной сети

На сегодняшний день, многослойные нейронные сети способны решать практически любую задачу, связанную с обнаружением корреляции входных и выходных данных. Из-за большего количества промежуточных нейронов, в сравнении с двухслойными нейронными сетями, подобные сети способны точнее решать уникальные задачи.

Математически, многослойную нейронную сеть можно представить следующей формулой:

(4)

где - результат работы многослойной нейронной сети;

()- функция активации нейрона;

I - номер входных данных;

j - номер нейрона в слое;

n - номер слоя;

w - весовой коэффициент;

x - входной сигнал.

Однако, из-за возможности добавления неограниченного количества слоёв и нейронов, возникают случаи переобучения нейронной сети. Данное состояние характерно для задач с большим количеством входных и выходных нейронов, чрезмерно большими или ненормализованными входными данными, а также, неочевидной корреляцией между входными и выходными данными. Подобная проблема может быть решена следующими способами:

* Нормализация входных данных: входные данные, представленные числовым форматом, не должны быть чрезмерно большими. Максимальный и минимальный порог входных данных может изменяться в зависимости от задачи, но в идеале, для каждого входного нейрона значение не должно превышать число в интервале (-10; 10);
* Разработка и тестирование иных архитектур нейронных сетей с меньшим количеством скрытых слоёв и нейронов; иных функций активации и методов вычисления ошибки;
* Использование прореживания (dropout). Так как переобучение напрямую связано с количеством нейронов, самостоятельно не оказывающих существенное значение на итоговое выходное значение, то возникает потребность отключить подобные нейроны при обучении нейронной сети и её последующей эксплуатации [8].

## 1.3 Градиентный спуск. Методы вычисления ошибки нейронной сети

Ключевое отличие классических алгоритмов от алгоритмов машинного обучения заключается в гибкости решения поставленных задач. Искусственные нейронные сети достигают этого благодаря процессу обучения. Из-за этого происходит корректировка параметров (весовых коэффициентов) искусственной нейронной сети. Изменяя определённым образом весовые коэффициенты, нейронная сеть минимизирует итоговую ошибку при решении поставленной задачи.

Базовым методом машинного обучения является «градиентный спуск». Данный алгоритм лежит в основе большинства более сложных алгоритмов обучения [9]. Данный метод в математике предназначен для нахождения экстремумов функции, то есть, нахождения локальных и глобальных минимумов и максимумов функции. В машинном обучении градиентный спуск необходим для нахождения направления корректировки весовых коэффициентов, а также скалярное расстояние до точки минимума.

Графически градиентный спуск для всей модели искусственной нейронной сети можно представить как некую многомерную кривую. На данном графике (рисунок 4), горизонтальные оси () обозначают численное значения каждого весового коэффициента. Вертикальная ось обозначает численное значение ошибки при данных значениях весовых коэффициентов.

Вычисление ошибки является важным этапом для обучения моделей искусственных нейронных сетей. Методов вычисления ошибки существует большое количество, однако, в большинстве задач применяется метод вычисления «среднеквадратичной ошибки» (mean squared error):

(5)

где, ERROR – среднеквадратичная ошибка;

x – входные значения;

N\_OUT – выходное значение нейронной сети при данных входных значениях;

R\_OUT – истинное выходное значение нейронной сети при данных входных значениях.

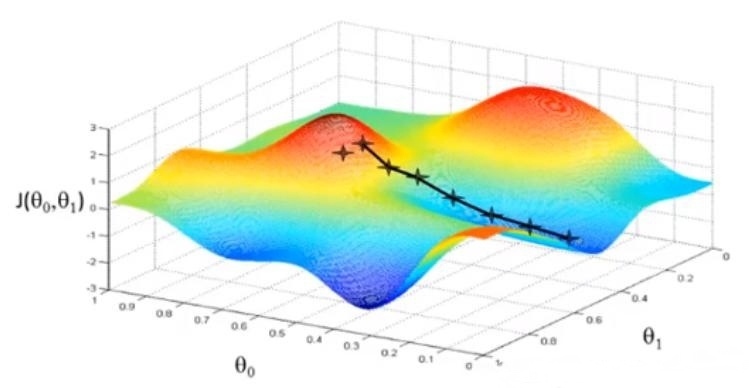


Рисунок 4 – многомерное представление градиентного спуска

Для каждого отдельного нейрона можно так же вычислить ошибку и отобразить функцию на графике. Для этого необходимо отобразить ось весового коэффициента, подаваемого на нейрон, и ось среднеквадратичной ошибки для этого нейрона. По итогу на графике будет отображена квадратичная парабола, с глобальным минимумом в точке минимальной среднеквадратичной ошибки со значением весового коэффициента рваного 3 (рисунок 5).

Рисунок 5 – график градиентного спуска для отдельного нейрона

## 1.4 Обучение модели. Метод обратного распространения ошибки

Процесс обучения нейронной сети строится на основе корректировки всех весовых коэффициентов, присутствующих в данной модели искусственной нейронной сети. Этот процесс позволяет нейронной сети находить корреляцию между входными данными и, заранее известными, истинными выходными значениями. Данный способ обучения называется обучением «с учителем», так как каждый набор входных данных подкреплён заранее известными «ответами» [10]. Подобный метод обучения наиболее распространён, так как разработчик может заранее предугадать возможное поведение модели нейронной сети, что позволяет частично или полностью избежать случаев неверного или чрезмерного обучения модели.

Корректировка основывается на вычислении разности между результатом вычисления нейронной сети для конкретного набора данных и заранее известным истинным результатом. Данная разность позволяет определить направление корректировки весовых коэффициентов (увеличение или уменьшение), а также численное представление несоответствия результата нейронной сети и истинного значения.

Подобную разность именуют «чистой ошибкой», и зачастую, для корректировки весовых коэффициентов эта ошибка не подходит. Перед её использованием, необходимо произвести процесс масштабирования, достигаемый путём умножения чистой ошибки на входной сигнал:

(6)

где, RESIDUAL – итоговое значение изменения весового коэффициента;

N\_OUT – выходные данные нейронной сети;

R\_OUT – истинное выходное значение;

x – входные сигналы, подаваемые на нейроны.

Производя корректировку весовых коэффициентов всего один раз, точность работы нейронной сети существенно не увеличится. Следовательно, необходимо выполнить обучение максимальное количество раз для увеличения точности и уменьшения ошибки нейронной сети. Такие циклы обучения называтся «эпохами» [11], и, исходя из задачи, эпох может быть существенно много, вплоть до нескольких десятков и сотен тысяч. Количество эпох зависит от:

* количества параметров нейронной сети;
* сложность корреляции входных и выходных данных;
* количества данных в обучающей выборке.

Рассматривая процесс обучения полносвязных многослойных нейронных сетей, необходимо использовать метод обучения «обратное распространение ошибки» [12]. Данный метод обучения берёт в свою основу базовый метод вычисления градиента, используя его для обновления всех весовых коэффициентов поочерёдно, начиная с весовых коэффициентов конечного (выходного) слоя и заканчивая начальным (выходным) (рисунок 6).

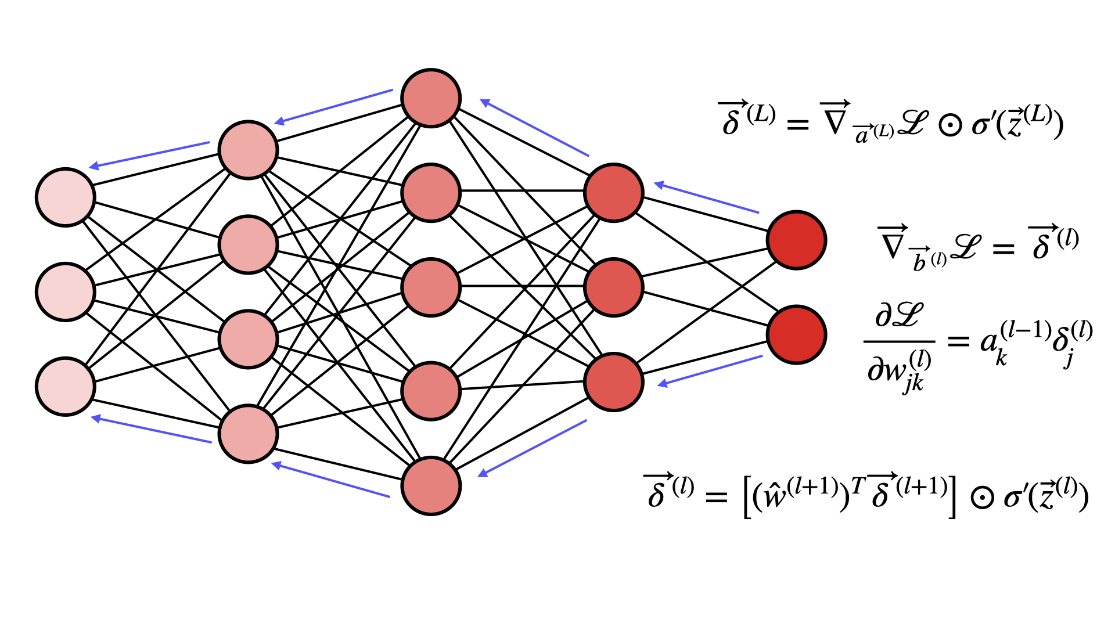


Рисунок 6 – визуализация метода обратного распространения ошибки

## 1.5 Представление графической информации в виде, пригодном для работы модели искусственной нейронной сети

Ключевой проблемой при работе с графической информацией с использованием искусственных нейронных сетей является проблема представления графической информации в цифровом формате. Нейронные сети умеют работать только лишь с числами малых разрядов. При этом современное оборудование для записи фото и видео способны формировать изображение, состоящее из миллионов пикселей, что в разы усложняет построение оптимальной архитектуры нейронной сети. Это создаёт определённые трудности при работе с фото и видео файлами.

Современные методы решения данной проблемы можно глобально разделить на 2 типа:

1. Прямая конвертация - метод, при котором изображение остаётся в изначальном размере и формате. Каждый пиксель записывается в матрицу, по размерам, идентичным размерам изображения. Пиксель принимает формат числа в зависимости от алгоритма конвертации. Самым распространённым является конвертация по яркости пикселя, при котором существует градация цвета от 0 (чёрного цвета) до 1 (белого цвета) (рисунок 7). Промежуточные значения будут являться различными оттенками серого.
2. Свёртка [13]. Методы свёртки позволяют получить числовую матрицу меньшего размера, чем размер изначального изображения (рисунок 8). Достигается это путём анализа одновременно нескольких рядом стоящих пикселей на изображении и записи 1 числа. Самыми распространёнными методами анализа являются: среднее арифметическое, максимальное и минимальное значение, сумма.

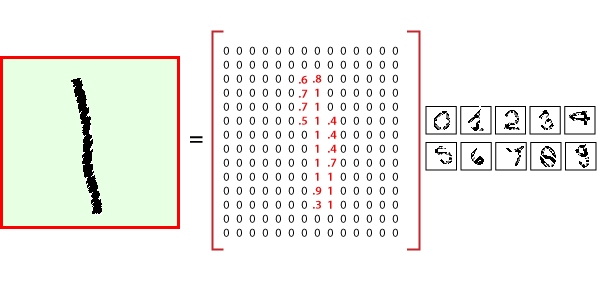


Рисунок 7 – представление изображения в числовом виде по яркости пикселей

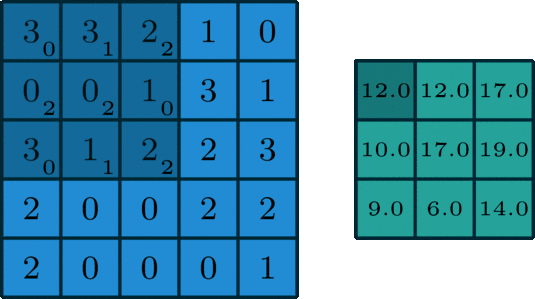


Рисунок 8 – пример свёртки матричных значений

Выбор способа представления графической информации в числовом виде зависит от конкретной задачи. Зачатую, чем больше в изображении пикселей, тем хуже будет справляться метод прямой конвертации, как в 1 варианте, и наоборот, для изображений менее 32x32 пикселей функционал свёртки, как во 2 варианте, будет излишним.

## Вывод к 1 главе

В первой главе индивидуального проекта было подробно описано применение и принцип действия искусственных нейронных сетей. Были рассмотрены методы обучения простейшего перцептрона, полносвязных многослойных нейронных сетей, а также, методы вычислений ошибки моделей. Рассмотрены методы преобразования графической информации в числовом виде с учётом нормализации данных.

Данная теоретическая база необходима для решения поставленной задачи.

# ГЛАВА 2 РАЗРАБОТКА СИСТЕМЫ РАСПОЗНОВАНИЯ ЛИЦ С ИСПОЛЬЗОВАНИЕМ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

## Подготовка датасетов для обучающей и тестовой выборки

Следуя поставленной задачи, необходимо разработать систему, способную идентифицировать определённого человека на изображении.

Для этого было выбрано 4 добровольца, готовых участвовать в проекте. Из них: 3 – мужского пола, возрастом приблизительно от 14 до 18 лет; 1 – женского пола, возрастом приблизительно от 14 до 18 лет. Изображения участников приведены ниже (рисунок 9).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Мужчина 1 | Мужчина 2 | Мужчина 3 | Женщина 1 |

Рисунок 9 – изображения лиц участников проекта

Для создания изображений, используемых для обучения нейронной сети, были сняты 4 небольших видео, на каждого участника. Длинна каждого видео, в среднем была равна 20 секундам, с частотой кадров 30 кадров в секунду. При съёмке, камера плавно перемещалась, чтобы запечатлеть участников с разных ракурсов (вид по бокам, снизу и сверху).

При постобработке, была увеличена контрастность изображения и «резкость». Размеры видео были уменьшены с 1080x1920 пикселей до 1080x1080 пикселей (кроме видео с участником «мужчина 2». Из-за малого поля обзора, необходимо было уменьшить размеры видео до 1080x1256 пикселей). Данные действия необходимы для увеличения точности обучения нейронной сети, и для увеличения скорости обучения.

По окончанию рендера обработанных видео, необходимо было разложить получившиеся видео на кадры. Для решения данной задачи был использован онлайн ресурс «img2go». При использовании конвертации файла с расширением «.mp4» в «.jpg», видеофайл разделяется на отдельные кадры, которые можно скачать одним архивом (рисунок 10). В итоге образовалось 4 папки с фотографиями:

* Папка «Men1»: 177 изображений;
* Папка «Men2»: 185 изображений;
* Папка «Men3»: 233 изображений;
* Папка «Female1»: 202 изображения.

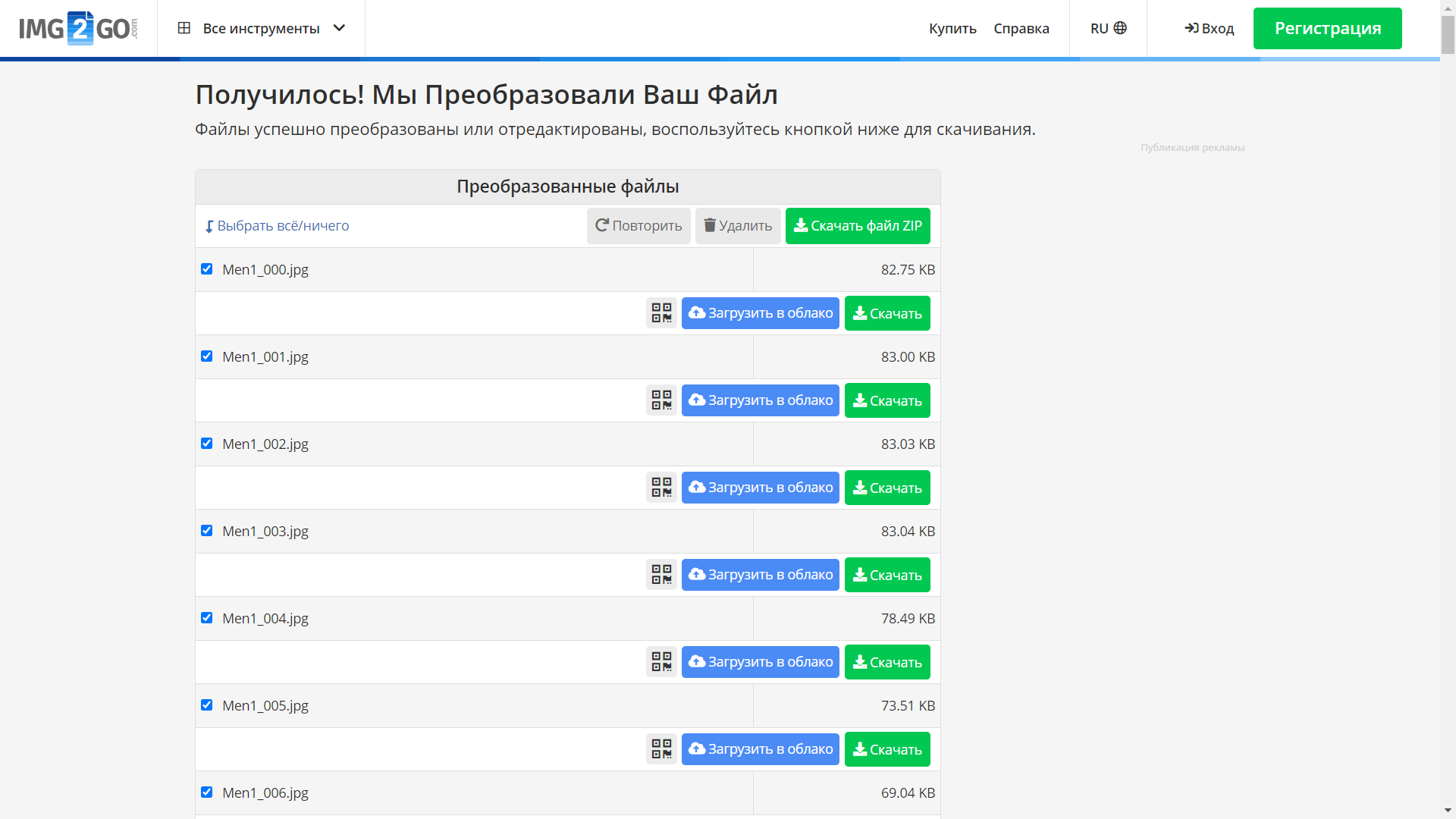


Рисунок 10 – сайт img2go с разложенным на кадры видео

Помимо наборов изображений лиц 4 добровольцев, необходимо использовать некоторое количество изображений лиц случайных людей. Данный метод поспособствует улучшению точности работы модели нейронной сети. Для создания данного набора изображений был использован электронный ресурс (сайт) «This person does not exist». Данный сайт использует алгоритмы машинного обучения для генерации изображений человеческих лиц, не отличимых от настоящих (рисунок 11). Алгоритм способен генерировать изображения лиц людей разных возрастов (включая детей и пожилых людей), а также различных этнических групп.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Рисунок 11 – примеры сгенерированных изображений

Для создания датасета из изображений лиц случайных людей, была написана программа «парсер» на языке программирования Python (приложение А). Данная программа в автоматическом режиме скачивала изображения с сайта «This person does not exist», и сохраняла их в отдельной папке под названием «SomePerson». В итоге папка содержит 100 изображений со случайными лицами людей.

Набор тестовой выборки должен содержать уникальные изображения, не используемые для обучения модели нейронной сети. Для создания данного набора, необходимо было зафиксировать лица кандидатов в необычных условиях (нестандартная мимика, угол обзора камеры, освещение и иные факторы). Помимо этого, необходимо было добавить в набор определённое количество изображений лиц случайных людей. Для этого, необходимо было вновь воспользоваться сайтом «This person does not exist» .

## 2.2 Описание работы нейронной сети

Для решения поставленной задачи, необходимо разработать модель искусственной нейронной сети, способную определять лицо человека на изображении и идентифицировать его.

Для этого, необходимо решить 2 ключевые задачи:

* 1. Задача поиска лица на данном изображении и создание «маски» из ключевых течек;
  2. Определение конкретного человека исходя из данных ключевых точек.

Для решения 1 задачи была использована заранее обученная свёрточная нейронная сеть, используемая совместно с библиотекой «dlib» [14]. Dlib – кроссплатформенная библиотека программного обеспечения, используемая для задач связанных с машинным обучением, написанная на языке программирования C++. Данная библиотека имеет обширный набор инструментов для создания и оптимизации моделей глубоких нейронных сетей [15]. Для решения поставленной задачи были использованы 3 функции библиотеки «dlib»:

* Функция «face\_recognition\_model\_v1» предназначенная для обнаружения человеческого лица на изображении;
* Функция «get\_frontal\_face\_detector» возвращает детектор лица на изображении;
* Функция «shape\_predictor» предназначенная для создания «маски» из ключевых точек.

Используя библиотеку «dlib», можно создать специальную маску, состоящую из ключевых точек (рисунок 12). Данные точки отмечаются в следующих зонах лица: границы лица (щёки и подбородок), брови, контур глаз, нос (вертикальная центральная ось и нижние крайние точки), губы (внешние и внутренние границы). Всего библиотека «dlib» создаёт маску из 64 точек, которые можно преобразовать в одномерный массив чисел (дескриптор), при помощи функции «compute\_face\_descriptor». Данный массив будет содержать 128 чисел, значение которых не превышает область от -1 до 1, включительно.

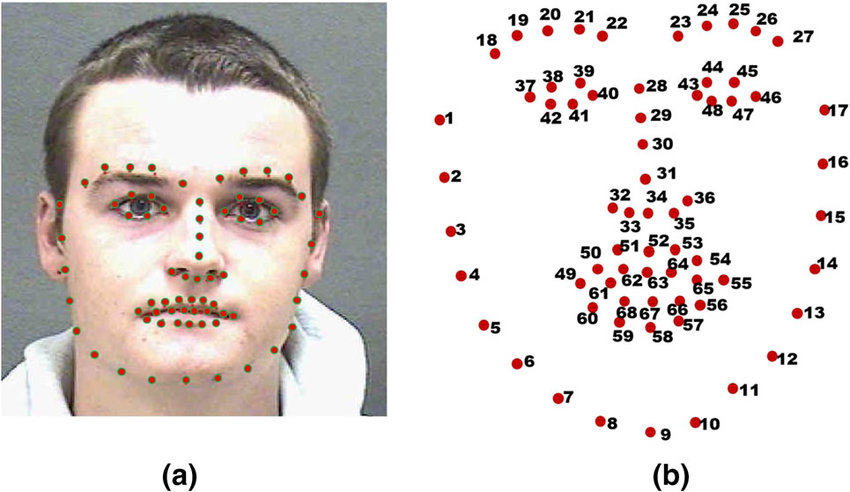


Рисунок 12 – наложение изображения маски лица на изображение лица

Используя данный дескриптор можно создать глубокую нейронную сеть «классификатор», способную находить корреляцию между дескриптором лица и конкретным человеком. Исходя из этого, нейронная сеть должная иметь следующую архитектуру:

* 1. Первый слой является входным, поэтому, количество нейронов в этом слое должно соответствовать количеству чисел в дескрипторе (128). Каждую «эпоху» случайные 20% нейронов в этом слое будут отключаться, для предотвращения запоминания конкретных наборов данных. Функция активации – линейная;
  2. Данный слой является скрытым, и имеет 256 нейронов. Также как и в 1 слое, случайные 20% нейронов не будут активны. Функция активации – «ReLU»;
  3. Выходной слой. Методов представления выходных значений существует большое количество, однако их можно разделить на 2 категории: метод «единичного выхода», когда в выходном слое содержится 1 нейрон, и «множественный выход», когда на каждый уникальный объект классификации выделяется отдельный нейрон. Так как поставленная задача нуждается в идентификации уникального человека, целесообразнее всего прибегнуть ко «множественному выходу». Пусть для каждого уникального человека (в данном случае это: «Men1», «Men2», «Men3» и «Female1») будет выделен отдельный нейрон. В таком случае, выходной слой будет иметь 4 нейрона. Функция активации – линейная.

Данная нейронная сеть является полносвязной, то есть, все нейроны слоя n связаны со всеми нейронами слоя n+1. Таким образом, схематично данную архитектуру можно изобразить следующим образом (рисунок 13).

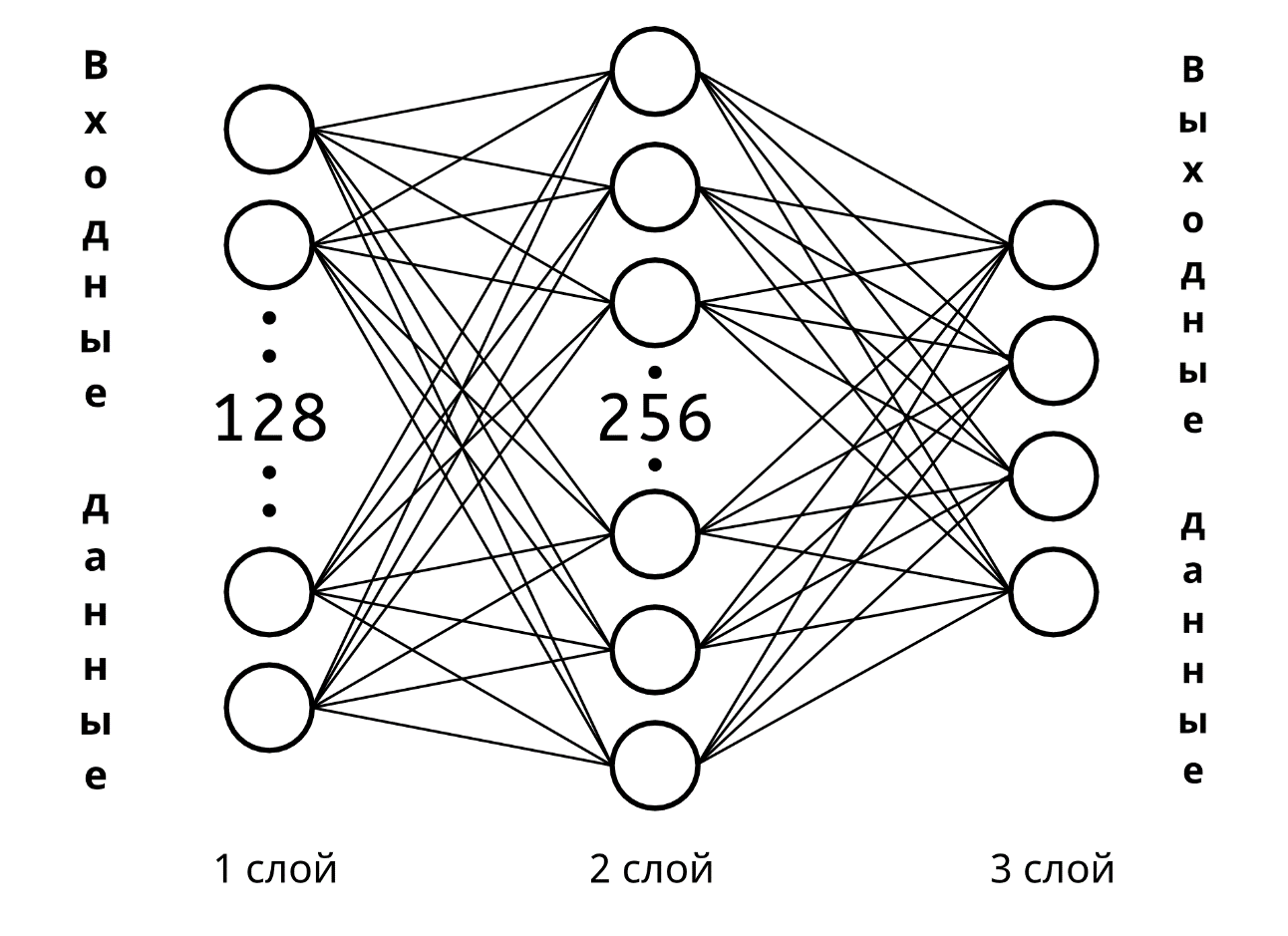


Рисунок 13 – архитектура модели глубокой нейронной сети для решения поставленной задачи индивидуального проекта

## 2.3 Выбор языка программирования и среды разработки программы. Установка и описание библиотек и модулей

На сегодняшний день существует большое количество языков программирования. Связано это со сферами применения тех или иных языков программирования. Если рассматривать языки программирования с точки зрения применения их в машинном обучении, то зачастую рассматриваются следующие языки программирования [18]:

* Python;
* C++;
* Lisp;
* JavaScript;
* R.

Иные языки программирования, не вошедшие в список, тоже могут использоваться для разработки алгоритмов машинного обучения, однако большая часть стороннего функционала (сторонние модули и библиотеки), не будут поддерживаться в подобных языках программирования, из-за чего, весь функционал придётся создавать вручную с нуля.

Для выбора языка программирования из предложенного списка учитывались следующие факторы:

1. Знание синтаксиса языка;
2. Удобство и скорость разработки;
3. Скорость выполнения кода;
4. Простота внедрения сторонних модулей в программу;
5. Наличие большого сообщества разработчиков.

Следуя обозначенным пунктам, был выбран язык программирования Python3.7 [17]. Данный язык программирования является универсальным, однако для создания алгоритмов машинного обучения, Python – является лидирующим языком программирования [18]. Связано это в первую очередь с обширным сторонним функционалом для разработки алгоритмов машинного обучения. Подобные модули и библиотеки легко внедряются в программу и написаны на высокопроизводительных языках (таких как язык «Си» или «C++»), за счёт чего подобные программы будут работать на порядок быстрее, чем программы без данных модулей и библиотек.

В качестве среды разработки для языка программирования Python, был выбран «Jupyter Notebook» [19]. «Jupyter Notebook» является проектом компании «Jupyter» с открытым исходным кодом. Преимуществом и главным отличием «Jupyter Notebook» от классических сред разработки является возможность дробления программы на «блоки кода», которые можно исполнять как отдельно, так и совместно со всей программой (рисунок 14). При этом, при инициализации переменной, функции, класса и подобного в определённом блоке, данные объекты будут доступны во всей программе.

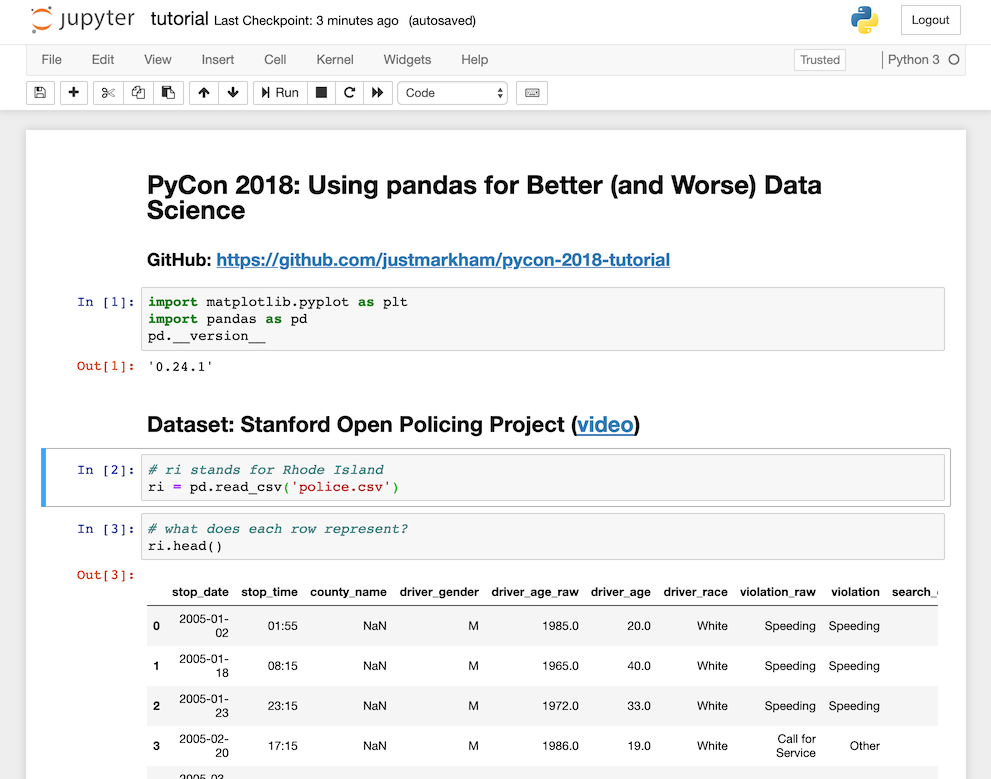


Рисунок 14 – пример программы в Jupyter Notebook

Для решения поставленной задачи индивидуального проекта необходимо воспользоваться определённые списком модулей и библиотек. Язык программирования Python имеет ряд предустановленных модулей, которые скачиваются совместно с интерпретатором языка на компьютер. Из них, для решения поставленной задачи необходим:

* Модуль «random» [20]: позволяет генерировать случайные числа, выбирать случайные объекты из списка, перемешивать списки случайным образом и многое другое.

Помимо предустановленных модулей, язык программирования Python даёт возможность устанавливать библиотеки и модули из интернета, при помощи пакетного менеджера «PIP» [21]. При помощи пакетного менеджера были установлены следующие библиотеки и модули:

* Библиотека «TensorFlow» [22, 23]: библиотека, созданная компанией «Google» в 2015 году. На сегодняшний день, TensorFlow является крупнейшей и самой популярной библиотекой для разработки алгоритмов машинного обучения. Функционал данной библиотеки позволяет создавать модели нейронных сетей любой сложности. Очень сложен для изучения, поэтому для данной библиотеки разработали специальную библиотеку-надстройку;
* Библиотека «Keras» [24]: является надстройкой для библиотеки «TensorFlow». Крайне простая библиотека для обучения и для внедрения в код. Использует уже готовые функции и шаблоны, написанные при помощи библиотеки «TensorFlow»;
* Библиотека «Dlib» [14]: кроссплатформенная библиотека программного обеспечения, используемая для задач, связанных с машинным обучением, написанная на языке программирования C++;
* Библиотека «scikit-image» (иное название: «skimage») [25]: это библиотека обработки изображений с открытым исходным кодом для языка программирования Python. Имеет большой инструментарий для работы с компьютерным зрением;
* Библиотека «Python Imaging Library» (сокращённо: «PIL») [26]: библиотека языка Python, предназначенная для работы с растровой графикой;
* Библиотека «MatPlotLib» [27]: библиотека на языке программирования Python для визуализации данных двумерной и трёхмерной графикой (рисунок 15). Необходим для визуализации числовых данных.

Установив соответствующие модули, можно было приступить к написанию системы распознавания лиц с использованием глубоких нейронных сетей.

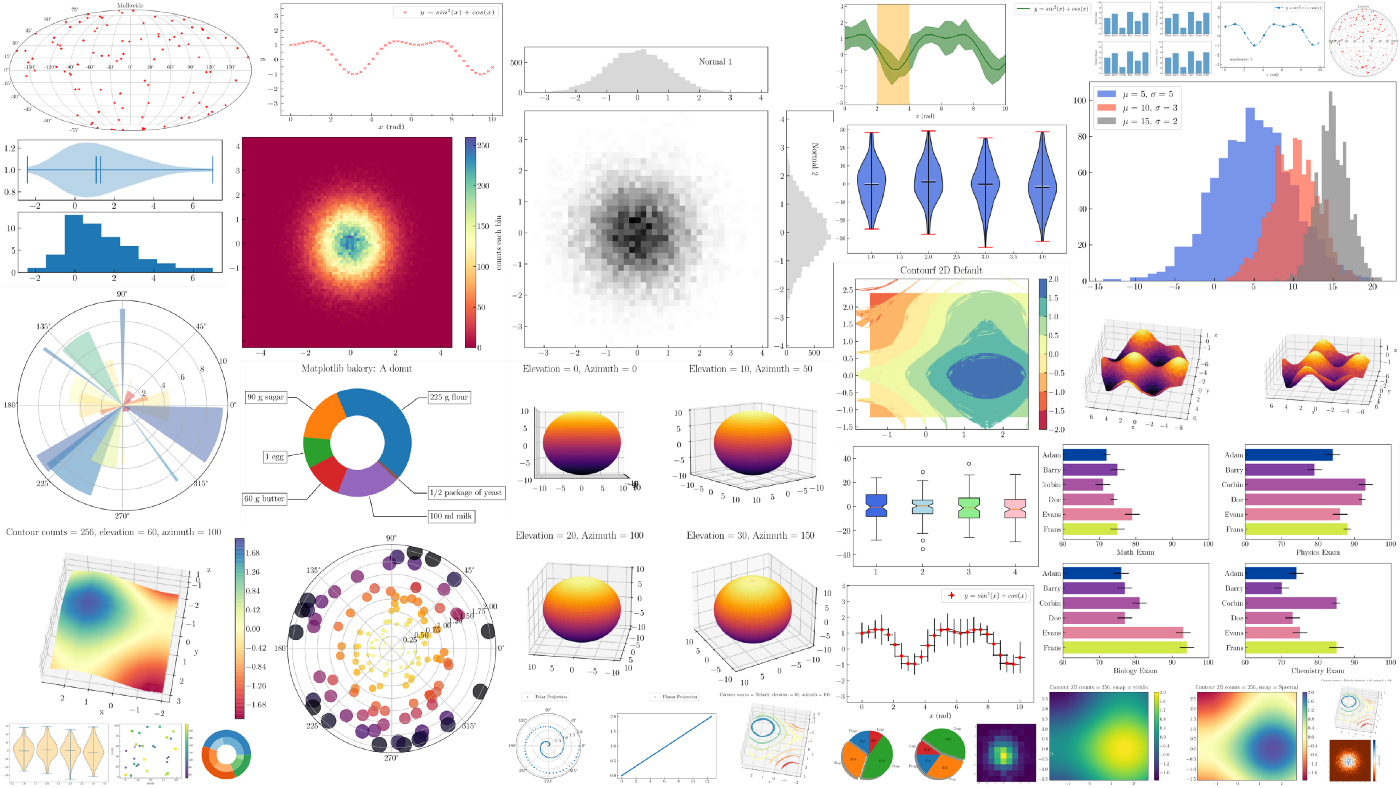


Рисунок 15 – примеры визуализаций данных при помощи MatPlotLib

## 2.4 Разработка системы распознавания лиц с использованием глубоких нейронных сетей

Перед началом написания программы, необходимо настроить рабочую область (папку), в которой будет работать программа. Для этого, необходимо в корневую папку программы поместить 4 папки с изображениями лиц добровольцев. Названия папок соответствуют названиям, выданных в главе 2.1. Таким же образом, необходимо поместить папку «SomePerson», с изображениями лиц, случайных людей. Помимо данных папок, необходимо скачать и переместить в корневую папку программы файлы предварительно обученных моделей для распознавания лиц и создания «маски» из ключевых точек лица. Таким образом, рабочая область полностью готова к работе (рисунок 16)

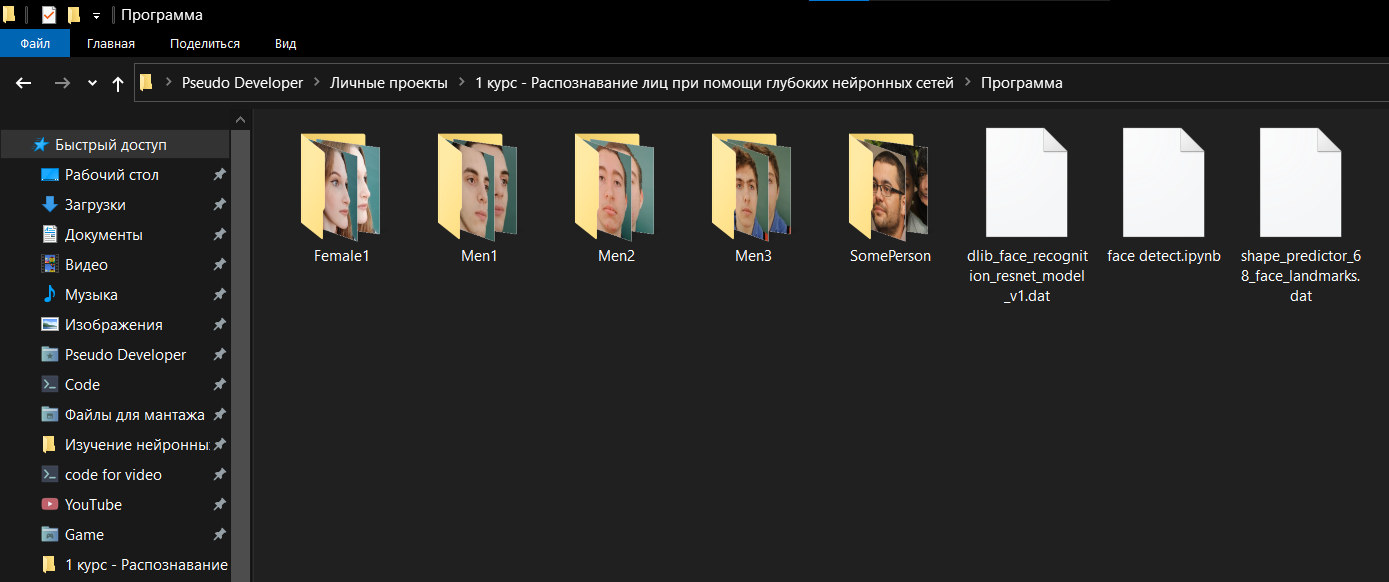


Рисунок 16 – рабочая область программы

Запустив среду разработки «Jupyter Notebook» в данной папке, необходимо создать файл программы с расширением «.ipynb». После чего, можно приступать к разработке самой программы.

В самом начале, необходимо подключить те библиотеки и модули, что будут использованы в самом проекте. Для того чтобы не использовать избыточный функционал некоторых библиотек, было использовано частичное подключение. В данном случае, в программу подключаются лишь отдельные класс и функции, которые будут задействованы в проекте. Таким образом мы получаем блок кода, описанный в приложении Б (рисунок 17).

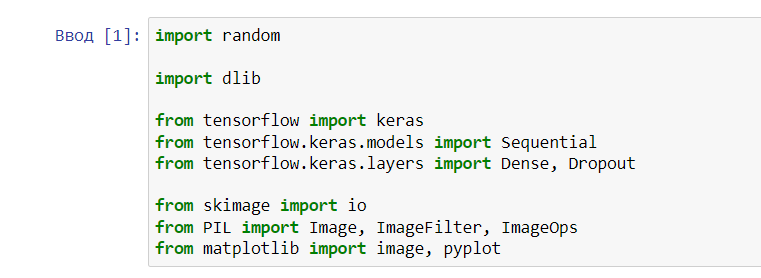


Рисунок 17 – блок подключений библиотек и модулей

Далее следует алгоритм подключения предварительно обученных моделей библиотеки «Dlib». В этом блоке кода необходимо воспользоваться скаченными моделями, что расположены в папке вместе с программой. Данный блок кода описан в приложении В (рисунок 18).

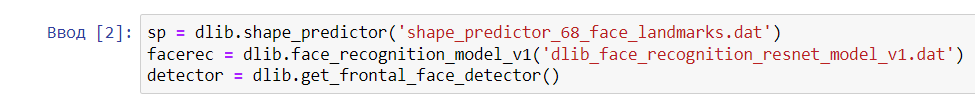


Рисунок 18 – блок кода подключения предварительно обученных моделей

Следующий блок кода отвечает за подключение всех изображений что находятся в корневой папке программы, и создание списков для обучающей и тестовой выборки, которые будут состоять из дескрипторов, созданных по лицевой маске каждого изображения, и истинным выходным значением «pred» (приложение Г).

Объект истинного значения «pred» является списком, состоящим из 4 чисел, которые могут принимать значение либо 0, либо 1. Каждому уникальному человеку присваивается уникальный индекс (положение) в списке: «Men1» - первое число, «Men2» - второе число, «Men3» - третье число, «Female1» - четвёртое число. Если данный человек присутствует на изображении, то, присеваемый ему индекс изменяется с 0 на 1, при этом, все остальные числа в списке остаются со значениями 0. Если на изображении нет лиц уникальных людей, как в случае с изображениями в папке «SomePerson», то все числа в списке остаются со значениями 0 (рисунок 19).

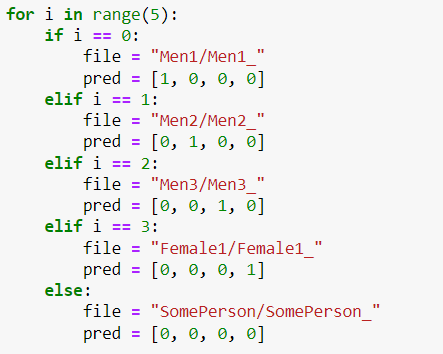


Рисунок 19 – присвоение истинных значений в зависимости от каталога с изображениями

Для создания списка тестовой выборки случайным образом выбирается 10% обработанных объектов от общего списка. Оставшиеся 90% переносятся в список обучающей выборки. Таким образом, создаётся случайны набор данных, не участвующий в обучении модели, тем самым, исключается фактор запоминания признаков нейронной сетью.

После отработки данного блока кода, будут полноценно созданы 2 списка: обучающей и тестовой выборки. Список обучающей выборки будет иметь 809 объектов, а список тестовой выборки – 88.

Следующий блок кода будет отвечать за рандомизацию объектов в обучающей и тестовой выборке, а также, за разделение входных значений (дескрипторов) и выходных (истинные значения «pred») (приложение Д). В дальнейшем, подобный способ хранения данных будет практичнее для работы с моделью искусственной нейронной сети. Рандомизация объектов необходима для предотвращения запоминания порядка и признаков объектов.

После разделения входных и выходных значений, можно приступать к написанию модели глубокой нейронной сети. Для этого, опираясь на архитектуру, описанную в главе 2.2 и рисунке 14, создаём блок кода для инициализации и обучения нейронной сети (приложение Е).

Данная модель нейронной сети принимает в качестве входных значений – значения дескриптора из списка обучающей выборки, а для выходных значений – истинные значения из списка обучающей выборки. Нейронная сеть будет обучатся на протяжении 500 эпох. Для вычисления ошибки будет использована «средняя квадратичная ошибка».

После блока кода с инициализацией и обучением модели глубокой нейронной сети следует блок кода, с тестированием работоспособности нейронной сети (приложение Ж). Данный скрипт производит вычисления исходя из параметров обученной нейронной сети, при этом, не используя методы обучения нейронной сети. Таким образом, можно будет увидеть чистую ошибку нейронной сети и чистую работоспособность в процентном значении.

В качестве аргументов функции, передаются списки входных значений (дескрипторов) тестовой выборки и выходных (истинных значений) тестовой выборки.

## 2.5 Обучение модели глубокой нейронной сети. Тест работоспособности. Приведение гипотез по улучшению работоспособности

При обучении модели нейронной сети (приложение Е), в консоли рабочей среды «Jupyter Notebook» были отображены данные по процессу обучения нейронной сети (рисунок 20). В этой информации храниться: номер эпохи, процесс-бар выполнения эпохи, примерное оставшееся время на выполнение эпохи и затрачиваемое время на выполнение 1 шага, средняя ошибка эпохи, средняя работоспособность (в процентах) эпохи.

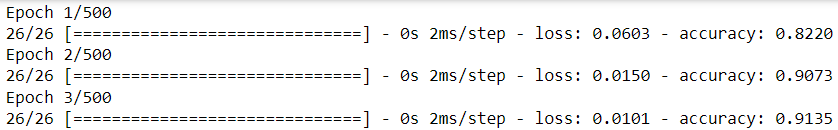


Рисунок 20 – информация по процессу обучения нейронной сети по первым 3 эпохам

По завершению обучения модели нейронной сети, данные в консоли имеют такие значения (рисунок 21): средняя ошибка последней эпохи – 0.0023, средняя точность последней эпохи – 91,22%.

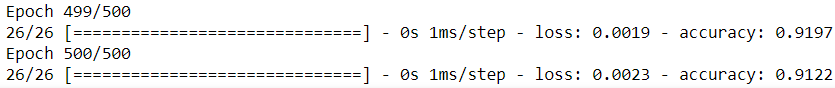


Рисунок 21 – данные об обучении нейронной сети по последним 2 эпохам

При помощи библиотеки «MatPlotLib» были визуализированы 2 графика (рисунок 22): средняя ошибка каждой эпохи, средняя точность каждой эпохи.

|  |  |
| --- | --- |
|  |  |

Рисунок 22 – графики средней ошибки и точности

Несмотря на то, что примерно после 200 эпохи значения ошибки и работоспособности нейронной сети по обучающей выборки почти никак не изменялись, на результатах тестовой выборки, дополнительные эпохи повлияют в лучшую сторону.

Для проверки работоспособности нейронной сети на тестовой выборке используем функцию «evaluate». Пропустив через неё данные тестовой выборки, в консоли «Jupyter Notebook» появились следующие значения (рисунок 23): средняя ошибка тестовой выборки составляет – ≈0.0003, точность работы – 97.72%.

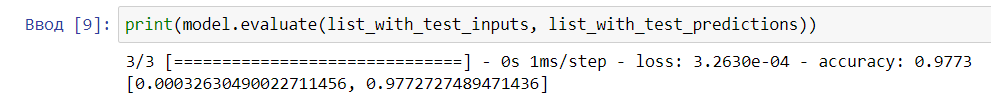


Рисунок 23 – значения работоспособности модели нейронной сети по тестовой выборке

Значение точности работы нейронной сети в значении 97.7% - является хорошим результатом для подобной модели реализации задачи. Однако для коммерческих и государственных служб, подобная точность является низкой. Для увеличения точности работы можно прибегнуть к нескольким методам: увеличение количества устанавливаемых ключевых точек лица; создание объёмного слепка лица; создание общей маски по изображениям с нескольких камер одновременно; изменение архитектуры нейронной сети; изменение метода определения итогового выходного значения.

## Вывод ко 2 главе

Во второй главе индивидуального проекта была описана, разработана и протестирована система, предназначенная для обнаружения и идентификации человека на изображении по биометрическим параметрам его лица.

Были выбраны добровольцы, готовые участвовать в разработке индивидуального проекта. Были собраны наборы изображений лиц добровольцев, а также, изображения лиц случайных людей.

Во второй главе были выбраны и описаны: среда разработки системы, язык программирования, а также модули, необходимые для разработки системы распознавания и идентификации лиц.

Была создана, обучена и протестирована модель глубокой нейронной сети, предназначенная для решения поставленной задачи индивидуального проекта. Выявлены гипотезы по улучшению работоспособности системы.

# ЗАКЛЮЧЕНИЕ

Рассмотрим результаты исследования, целью которого являлось изучить и разработать систему распознавания лиц на изображении при помощи глубоких нейронн ых сетей.

В данном индивидуальном проекте была изучена теория работы искусственных нейронных сетей. Были рассмотрены и описаны следующие элементы разработки и работы нейронных сетей:

* перемещение сигнала от нейрона к нейрону;
* функции активации нейронов;
* алгоритм работы полносвязных нейронных сетей;
* алгоритм работы многослойных нейронных сетей;
* вычисление ошибки нейронной сети;
* нормализация входных, промежуточных и выходных данных;
* методы обучения нейронных сетей.

Были рассмотрены и описаны базовые принципы работы компьютерного зрения, включая представление графической информации в числовом виде, путём прямой конвертации и при помощи алгоритмов свёртки.

После изучения теоретической части было выбрано 4 добровольца, готовых участвовать в разработке системы распознавания лиц с использованием глубоких нейронных сетей. Были отобраны изображения лиц добровольцев крупным планом с различных ракурсов. Помимо изображений лиц добровольцев, были отобраны изображения лиц случайных людей при помощи электронных ресурсов для генерации лиц людей.

Была описана архитектура и алгоритм работы системы по распознаванию лиц с использованием глубоких нейронных сетей. Были рассмотрены и изучены: язык программирования, среда разработки, используемые библиотеки и модули.

Была разработана и протестирована система распознавания лиц с использованием глубоких нейронных сетей. Итоговая работоспособность системы (составляющая 97.7%) удовлетворяет условие поставленной задачи.

Таким образом, система, разработанная в ходе создания данной исследовательской работы, полностью соответствует поставленным задачам. Из этого следует, что цель индивидуального проекта была достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. / F. Rosenblatt // Psychological Review. - 1958. - №65(6). - С.386–408.
2. Бурсов, А.И. Применение искусственного интеллекта для анализа медицинских данных. / А.И. Бурсов // Альманах клинической медицины. – 2019. - №7. – С.630-633.
3. Кораблев, А.Ю. Машинное обучение в бизнесе. / А.Ю. Кораблев, Р.Б. Булатов. // АНИ: экономика и управление. – 2018. - №2 (23). – С. 68-72.
4. Горячкин, Б.С. КОМПЬЮТЕРНОЕ ЗРЕНИЕ. / Б.С. Горяченко, М.А. Китов. // E-Scio. – 2020. - №9 (48). – 29 с.
5. Курьянова, С.Л. БИОМЕТРИЧЕСКАЯ ИДЕНТИФИКАЦИЯ КЛИЕНТОВ В БАНКОВСКОЙ СФЕРЕ: ОТЕЧЕСТВЕННЫЙ И ЗАРУБЕЖНЫЙ ОПЫТ. / С.Л. Курьянова, О.С. Цвигунова. // АНИ: экономика и управление. – 2019. - №4 (29). – С. 238-241.
6. Дремов, И.С. Сравнение механизмов мышления искусственных и биологических нейронных сетей. / И.С. Дремов. // Тенденции развития науки и образования. – 2021. – №. 74-2. – С. 72-74.
7. Рудой, Г.И. Выбор функции активации при прогнозировании нейронными сетями. / Г.И. Рудой. // Машинное обучение и анализ данных. – 2011. – Т.1, №.1. – С. 16-39.
8. Пучков, Е.В. Прореживание синаптических весов искусственных нейронных сетей с целью оптимизации их структуры. / Е.В. Пучков. // Известия Ростовского государственного строительного университета. – 2010. – Т. 1, № 14(14). – С. 330.
9. Александров, В.С. Применение метода градиентного спуска для исследования набора данных. / В.С. Александров, В.В. Мокшин // Молодежь и наука: актуальные проблемы фундаментальных и прикладных исследований: Материалы III Всероссийской национальной научной конференции студентов, аспирантов и молодых ученых. В 3-х частях, Комсомольск-на-Амуре, 06–10 апреля 2020 года. / Комсомольский-на-Амуре государственный университет, 2020. – С. 206-208.
10. Галушкин, А.И. Нейронные сети. Основы теории. Монография. / А.И. Галушкин. – М: Горячая линия – Телеком, 2012. – 496 с.
11. Шалоха, Е.Р. Нейронные сети. / Е.Р. Шалоха // Материалы 73-й студенческой научно-практической конференции. - Белорусский национальный технический университет, Факультет информационных технологий и робототехники. – Минск: БНТУ, 2017. – С. 33-35.
12. Hecht-Nielsen, R. Theory of the backpropagation neural network. / R. Hecht-Nielsen // Neural networks for perception. – Academic Press, 1992. – С. 65-93.
13. Макаров, А.О. Быстрая обработка изображений на основе интегральных матриц изображений. / А.О. Макаров, В.В. Старовойтов. // Искусственный интеллект. – 2006. – Т. 3. – С. 597.
14. Официальная документация библиотеки «Dlib» [Электронный ресурс]. – режим доступа: <http://dlib.net/>
15. King, D.E. Dlib-ml: A machine learning toolkit. / D.E. King. // The Journal of Machine Learning Research. – 2009. – Т. 10. – С. 1755-1758.
16. Маношин, Д.А. Программирование искусственного интеллекта. / Д.А. Маношин. // Colloquium-journal. – Голопристанський міськрайонний центр зайнятості, 2019. – №. 12 (36). – С. 21-23.
17. Python 3.7.0 [Электронный ресурс]. – режим доступа: https://www.python.org/downloads/release/python-370/
18. Таршхоева, Ж.Т. Язык программирования Python. Библиотеки Python. / Ж.Т. Таршхоева // Молодой ученый. – 2021. – № 5(347). – С. 20-21.
19. Jupyter Project Documentation [Электронный ресурс]. – режим доступа: https://docs.jupyter.org/en/latest/
20. random — Generate pseudo-random numbers [Электронный ресурс]. – режим доступа: <https://docs.python.org/3/library/random.html>
21. pip is the package installer for Python [Электронный ресурс]. – режим доступа: <https://pip.pypa.io/en/stable/>
22. TensorFlow — это сквозная платформа с открытым исходным кодом для машинного обучения. Python [Электронный ресурс]. – режим доступа: https://www.tensorflow.org/overview
23. Abadi, M. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. / M. Abadi. // arXiv preprint arXiv:1603.04467. - 2016. – 19 с.
24. About Keras [Электронный ресурс]. – режим доступа: <https://keras.io/about/>
25. scikit-image 0.19.2 docs [Электронный ресурс]. – режим доступа: <https://scikit-image.org/docs/stable/>
26. Pillow Keras [Электронный ресурс]. – режим доступа: <https://pillow.readthedocs.io/en/stable/>
27. Users guide [Электронный ресурс]. – режим доступа: https://matplotlib.org/stable/users/index

# ПРИЛОЖЕНИЕ А

Программа автоматического парсинга изображений «Pars.py»

import requests

def pars\_person():

for © in range(100):

img\_data = requests.get(“https://thispersondoesnotexist.com/image”).content

if len(str(i)) == 1:

count = “00” + str(i)

elif len(str(i)) == 2:

count = “0” + str(i)

else:

count = str(i)

with open(f’SomePerson\_{ count }.jpg’, ‘wb’) as handler:

handler.write(img\_data)

if \_\_name\_\_ == ‘\_\_main\_\_’:

pars\_person()

# ПРИЛОЖЕНИЕ Б

Блок кода, отвечающий за подключение библиотек и модулей

import random

import dlib

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

from skimage import io

from PIL import Image, ImageFilter, ImageOps

from matplotlib import image, pyplot

# ПРИЛОЖЕНИЕ В

Блок кода, отвечающий за подключение предварительно обученных моделей свёрточных нейронных сетей

sp = dlib.shape\_predictor(‘shape\_predictor\_68\_face\_landmarks.dat’)

facerec = dlib.face\_recognition\_model\_v1(‘dlib\_face\_recognition\_resnet\_model\_v1.dat’)

detector = dlib.get\_frontal\_face\_detector()

# ПРИЛОЖЕНИЕ Г

Блок кода, отвечающий за подключение изображений лиц, создание дескрипторов из лицевых масок, присвоение истинных результатов и создание списков обучающей и тестовой выборки

li\_train = list()

li\_test = list()

for © in range(5):

if © == 0:

file = “Men1/Men1\_”

pred = [1, 0, 0, 0]

elif © == 1:

file = “Men2/Men2\_”

pred = [0, 1, 0, 0]

elif © == 2:

file = “Men3/Men3\_”

pred = [0, 0, 1, 0]

elif © == 3:

file = “Female1/Female1\_”

pred = [0, 0, 0, 1]

else:

file = “SomePerson/SomePerson\_”

pred = [0, 0, 0, 0]

li = list()

counter = 0

while True:

try:

if len(str(counter)) == 1:

c = “00” + str(counter)

elif len(str(counter)) == 2:

c = “0” + str(counter)

else:

c = str(counter)

raw\_img = io.imread(file + str© + “.jpg”)

dtc\_img = detector(raw\_img, 1)

for k, d in enumerate(dtc\_img):

shape = sp(raw\_img, d)

li.append([facerec.compute\_face\_descriptor(raw\_img, shape), pred])

counter += 1

except:

for \_ in range(len(li) // 10):

rand\_index = random.randint(0, len(li) – 1)

li\_test.append(li[rand\_index])

del li[rand\_index]

li\_train += li

break

# ПРИЛОЖЕНИЕ Д

Рандомизация объектов в списках обучающей и тестовой выборки и дальнейшее разделение на входные и выходные значения

list\_with\_train\_inputs = list()

list\_with\_train\_predictions = list()

list\_with\_test\_inputs = list()

list\_with\_test\_predictions = list()

random.shuffle(li\_train)

for i in range(len(li\_train)):

list\_with\_train\_inputs.append(list(li\_train[i][0]))

list\_with\_train\_predictions.append(li\_train[i][1])

random.shuffle(li\_test)

for i in range(len(li\_test)):

list\_with\_test\_inputs.append(list(li\_test[i][0]))

list\_with\_test\_predictions.append(li\_test[i][1])

# ПРИЛОЖЕНИЕ Е

Инициализация и обучение нейронной сети

model = Sequential()

model.add(Dense(units=256, input\_shape=(len(list\_with\_train\_inputs[0]),)))

model.add(Dropout(0.2))

model.add(Dense(units=256, input\_shape=(256,), activation="relu"))

model.add(Dropout(0.2))

model.add(Dense(units=4, input\_shape=(256,)))

model.compile(loss="mean\_squared\_error", metrics=['accuracy'])

history = model.fit(list\_with\_train\_inputs, list\_with\_train\_predictions, epochs=500)

# ПРИЛОЖЕНИЕ Ж

Тестирование работоспособности модели нейронной сети

print(model.evaluate(list\_with\_test\_inputs, list\_with\_test\_predictions))