

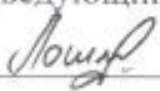
Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Комсомольский-на-Амуре государственный университет»

Кафедра «Информационная безопасность автоматизированных систем»  
Специальность 10.05.03 – «Информационная безопасность автоматизированных систем»

К ЗАЩИТЕ ДОПУСКАЮ

Заведующий кафедрой

 А.Ю. Лошманов

« 25 » 06 2020 г.


## ДИПЛОМНАЯ РАБОТА

Обеспечение безопасности систем «умный дом» с использованием  
компьютерного зрения и свёрточной нейронной сети

Н. КОНТР.

 А.А. Обласов


РУКОВОДИТЕЛЬ

 А.Ю. Лошманов

СТУДЕНТ группы 5ИБ-1

 Г.В. Васильев

РЕЦЕНЗЕНТ

 М.Т. Смирнов

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное  
образовательное учреждение высшего образования  
«Комсомольский-на-Амуре государственный университет»

Кафедра «Информационная безопасность автоматизированных систем»

УТВЕРЖДАЮ

Зав. кафедрой

\_\_\_\_\_ А.Ю. Лошманов

« \_\_\_\_ » \_\_\_\_\_ 2020 г.

**З А Д А Н И Е**  
**на дипломный проект /работу**

Выдано студенту \_\_\_\_\_ Васильеву Гордею Владимировичу  
Тема проекта /работы \_\_\_\_\_ Обеспечение безопасности систем  
\_\_\_\_\_ "умного дома" с использованием компьютерного зрения и сверточной  
\_\_\_\_\_ нейронной сети

\_\_\_\_\_ утверждена приказом по университету № 0987-ЛСС-ПРЧ-ИБАС от 31.10.2019

Срок сдачи студентом законченного проекта /работы \_\_\_\_\_ 25.06.2020

Исходные данные к проекту /работе \_\_\_\_\_

\_\_\_\_\_ Перечень подлежащих разработке вопросов в расчетно-пояснительной записке:

1 Спецчасть \_\_1. Рассмотреть область применения и задачи машинного и компьютерного зрения. 2. Изучить алгоритм Хаара. 3. Рассмотреть область применения и задачи нейронных сетей. 4. Создать нейронную сеть и разработать алгоритмы для обеспечения безопасности систем «Умный дом».

2 Экономическая часть \_\_\_\_\_

Рассчитать затраты, которые возникают при исследовании сверточных нейронных сетей, их тестирование и создании программного обеспечения, по результатам которых определить себестоимость продукта

3 Экологичность и безопасность \_\_\_\_\_

Рассмотреть вопросы безопасности и экологичности

4 \_\_\_\_\_

Перечень графического материала (с точным указанием обязательных чертежей)

Консультанты по проекту /работе (с указанием относящихся к ним разделов)

Задание принял к исполнению \_\_\_\_\_ «\_\_» \_\_\_\_\_ 2020 г.  
(подпись)

Руководитель, \_\_\_\_\_  
(подпись) (Ф.И.О.)

должность, ученая степень \_\_\_\_\_ «\_\_» \_\_\_\_\_ 2020 г.

## **Аннотация**

### **Обеспечение безопасности систем «умный дом» с использованием машинного зрения и сверточной нейронной сети**

Пояснительная записка 102 с., 71 рис., 2 табл., 102 источника, 4 приложения.

Разработано программное обеспечение для распознавания личности по лицу и обнаружения наличия маски на лице. Программное обеспечение умеет создавать обучающую выборку клиентов для обучения сверточной нейронной сети, обучать сверточную нейронную сеть, распознавать личность по лицу или определять, как «неизвестный», и обнаруживать наличия маски на лице или ее отсутствие. Перед разработкой программного обеспечения были исследованы архитектуры нейронных сетей, функции активации и функции потерь. Был проведен анализ нейронных сетей, после чего были сделаны соответствующие выводы. Программное обеспечение внедрено в организацию ООО «ФОГСТРИМ».

## **Abstract**

### **Securing «smart home» systems using machine vision and convolutional neural networks**

Explanatory Note 102, 71 pic., 2 tabl., 102 source, 4 enclosure.

Software has been developed for facial recognition and face mask detection. The software is able to create training data for training a convolutional neural network, train a convolutional neural network, recognize a person by face or determine as «unknown», and detect the presence of a mask on the face or its absence. Before developing the software, neural network architectures, activation functions, and loss functions were investigated. An analysis of neural networks was carried out, after which the corresponding conclusions were drawn. The software has been introduced into the organization of FOGSTREAM LLC.

## Содержание

Введение.....	6
1 Компьютерное зрение.....	8
1.1 Область применения и задачи машинного и компьютерного зрения .....	8
1.2 Библиотеки для машинного и компьютерного зрения.....	9
1.3 Алгоритм Хаара.....	9
2 Нейронные сети.....	10
2.1 Область применения и задачи нейронных сетей .....	10
2.2 Библиотеки для нейронных сетей .....	10
2.3 Топологии нейронных сетей.....	11
2.4 Функции активации .....	12
2.5 Функции потерь.....	15
3 Создание нейронной сети.....	16
3.1 Создание VGGNet подобную архитектуру свёрточной нейронной сети с помощью библиотеки Keras и бэкэнда tensorflow .....	19
3.2 Использование свёрточной нейронной сети из библиотеки face_recognition.....	38
3.3 Распознавание наличия маски на лице используя свёрточной нейронной сети и библиотеки Keras с бэкэндом TensorFlow .....	43
4 Использование компьютерного зрения.....	49
5 Экономическая часть .....	51
6 Безопасность и экологичность.....	57
6.1 Описание рабочего места программиста.....	57
6.2 Освещенность .....	60
6.3 Шум .....	64
6.4 Электробезопасность .....	64

6.5 Возможные аварийные и чрезвычайные ситуации.....	65
6.6 Охрана окружающей среды .....	66
6.7 Вывод по разделу .....	66
Заключение .....	68
Список использованных источников .....	69
ПРИЛОЖЕНИЕ А. Модуль загрузки датасета .....	79
ПРИЛОЖЕНИЕ Б. Модуль распознавания личности по лицу и обнаружение наличия маски.....	81
ПРИЛОЖЕНИЕ В. Модуль обучение нейросети обнаруживать маску .....	91
ПРИЛОЖЕНИЕ Г. Руководство пользователя .....	95

## Введение

За последний год с лишним усовершенствования технологии интеллектуального управления инфраструктурой сделали обслуживание физической инфраструктуры очень простым процессом. Система интеллектуального управления инфраструктурой обеспечивает жизненно важный уровень защиты, позволяющий быстро обнаруживать и фиксировать атаки, предупреждать администраторов и выполнять соответствующие действия. Такую систему можно использовать для документирования и контроля сетевых событий в реальном масштабе времени и предупреждения о любых неразрешенных подключениях и отключениях. Можно настроить систему для вывода информации о проникновении в серверные и офисные помещения и т. п. Средства документирования и контроля, обеспечиваемые первым уровнем, позволяют защищать от определенных запрещенных действий при помощи соотнесения устройств и их физического местоположения в организации. Можно запрещать совершение некоторых действий или подключение устройств в определенных точках.

В 1960 году начались работы в области искусственного интеллекта. Уже в 2019 году искусственный интеллект составляет 2,5 миллиардов долларов мирового рынка, в 2024 году предположительно мировой рынок ИИ (искусственного интеллекта) вырастет до 137,2 миллиардов долларов. 30 стран в мире считают, что искусственный интеллект имеет критическую важность и принимают соответствующие национальные стратегии.

История компьютерного зрения берет свое начало с 50-х годов XX века. В это время компьютеры начали становиться общедоступными для обработки и анализа информации.

Машинное зрение – научное направление в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования получен-

ных данных для решения разного рода прикладных задач без участия (частичного или полного) человека. Компьютерное зрение – общий набор методов, позволяющих компьютерам видеть. Машинное зрение является подразделом инженерии, связанное с вычислительной техникой, оптикой, машиностроением и промышленной автоматизацией.

Области применения машинного зрения: медицина, охранные системы, мультимедиа-приложения, расширенная реальность, системы распознавания рукописного и печатного текста, промышленность.

С использованием компьютерного зрения можно реализовать программное обеспечение, которое будет фиксировать моргание. Данное программное обеспечение можно использовать для дополнительного этапа защиты перед распознаванием лица, таким образом можно отличить снимок от настоящего человека. Еще одно применение – безопасность дорожного движения: программное обеспечение может зафиксировать сонливость водителя, если он засыпает или уснул за рулём.



## **1 Компьютерное зрение**

История компьютерного зрения берет свое начало с 50-х годов XX века. В это время компьютеры начали становиться общедоступными для обработки и анализа информации.

Машинное зрение – научное направление в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования полученных данных для решения разного рода прикладных задач без частичного или полного участия человека. Компьютерное зрение – общий набор методов, позволяющих компьютерам видеть. Машинное зрение является подразделом инженерии, связанное с вычислительной техникой, оптикой, машиностроением и промышленной автоматизацией.

### **1.1 Область применения и задачи машинного и компьютерного зрения**

Области применения машинного и компьютерного зрения:

- 1) крупное промышленное производство;
- 2) ускоренное производство уникальных продуктов;
- 3) системы безопасности в промышленных условиях;
- 4) системы видеонаблюдения;
- 5) системы организации информации, например, для индексации баз данных изображений.

Задачи компьютерного зрения:

- 1) поиск изображений в интернете;
- 2) распознавание текста;
- 3) биометрия;
- 4) видеоаналитика;
- 5) анализ спутниковых снимков.

## **1.2 Библиотеки для машинного и компьютерного зрения**

Библиотека в программировании – это сборник подпрограмм или объектов, используемых для разработки программ. Набор классов, компонентов или модулей для разных задач.

Библиотеки представляют собой уже написанные кем-то переносимые наборы проверенного кода. Это готовые решения, которые программисты могут присоединять к своим программам, вставлять их в свой код по специальным алгоритмам, причем в разных проектах.

Для машинного и компьютерного зрения используются следующие библиотеки:

- 1) OpenCV;
- 2) Dlib;
- 3) Kornia;
- 4) Kornia.

## **1.3 Алгоритм Хаара**

Обнаружение объектов с использованием каскадных классификаторов на основе признаков Хаара – это эффективный метод обнаружения объектов. Этот подход, основанный на машинном обучении, где каскадная функция обучается из множества положительных и отрицательных изображений. Затем ее можно использовать для обнаружения объектов на других изображениях [68].

## **2 Нейронные сети**

В 1960 году начались работы в области искусственного интеллекта. Уже в 2019 году искусственный интеллект составляет 2,5 миллиардов долларов мирового рынка, в 2024 году предположительно мировой рынок ИИ (искусственного интеллекта) вырастет до 137,2 миллиардов долларов. 30 стран в мире считают, что искусственный интеллект имеет критическую важность и принимают соответствующие национальные стратегии.

### **2.1 Область применения и задачи нейронных сетей**

Нейронные сети применяются в следующих областях: военное дело, сельское хозяйство, промышленность, медицина, образование, госслужба, транспорт, дорожное движение, быт, развлечения, издательство, управление личными финансами, алгоритмическая торговля, исследования рынка, рекрутинг и управление человеческими ресурсами. Также различные средства нейронных сетей широко используются в области обеспечения безопасности и информационной безопасности, распознавании текста и речи, фильтрации спама в электронной почте и интеллектуального анализа данных. Также разрабатываются приложения для распознавания жестов, распознавание голоса, распознавание лица для интерпретации эмоций и невербальных сигналов. Другие приложения – это роботизированная навигация, преодоление препятствий и распознавание объектов.

Применять нейронные сети можно на производстве, например, в целях безопасности, контроля, сбора статистики, обеспечения прав доступа.

### **2.2 Библиотеки для нейронных сетей**

Для работы с нейронными сетями используются следующие библиотеки:

1) PyBrain;

- 2) Caffe;
- 3) TensorFlow;
- 4) Keras;
- 5) PyTorch.

## 2.3 Топологии нейронных сетей

Персептрон (рисунок 1). Самая простая и самая старая модель нейрона. Принимает некоторые входные данные, суммирует их, применяет функцию активации и передает их в выходной слой [87].

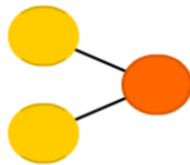


Рисунок 1 – Персептрон

Нейронная сеть с прямой связью (рисунок 2) – этот подход берет свое начало с 50-х годов. Они соответствуют следующим правилам:

- 1) все узлы полностью связаны;
- 2) активация течет от входного слоя к выходу, без обратных петель;
- 3) есть один слой между входом и выходом (скрытый слой).

В большинстве случаев этот тип сетей обучается методом обратного распространения [87].

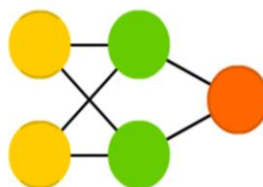


Рисунок 2 – Нейронная сеть с прямой связью

DCN (deep convolutional networks), сети представлена на рисунке 3. Она содержит ячейки свертки или объединяющие слои и ядра, каждое из которых служит разным целям [87].

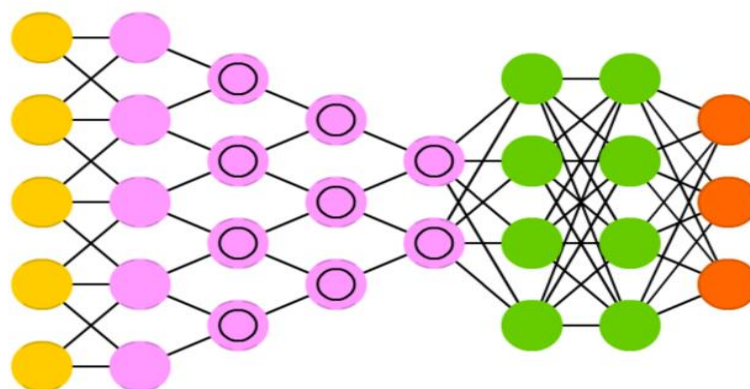


Рисунок 3 – DCN

## 2.4 Функции активации

Функция активации – это нелинейная функция, применяемая нейроном для введения нелинейных свойств в сеть.

Отношение является линейным, если изменение первой переменной соответствует постоянному изменению второй переменной. Нелинейное отношение означает, что изменение первой переменной не обязательно соответствует постоянному изменению второй. Тем не менее, они могут влиять друг на друга, но это кажется непредсказуемым.

Функции с несколькими степенями называются нелинейными функциями. Искусственные нейронные сети спроектированы как универсальные функциональные аппроксиматоры и предназначены для работы над этой целью. Это означает, что они должны иметь возможность рассчитывать и изучать любую функцию. Благодаря нелинейным функциям активации можно добиться более глубокого изучения сетей.

Для расчета значений ошибок, связанных с весами, применяется алгоритм обратного распространения искусственной нейронной сети. Необходимо определить стратегию оптимизации и минимизировать частоту появления ошибок.

Функция линейной активации, диапазон функции от  $-\infty$  до  $\infty$  (рисунок 4)[49]:

- 1) является постоянным значением;
- 2) значения могут стать очень большими;

3) не захватывает сложные шаблоны.

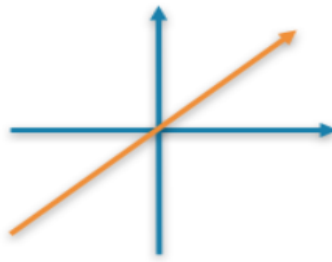


Рисунок 4 – Функция линейной активации

Формула функции линейной активации имеет вид

$$f(z) = a \cdot z. \quad (1)$$

Сигмовидная функция активации, диапазон функции от 0 до 1 (рисунок 5)[49]:

- 1) нелинейная функция, поэтому может захватывать более сложные шаблоны;
- 2) выходные значения ограничены, поэтому не становится слишком большими;
- 3) может страдать от «исчезающего градиента».

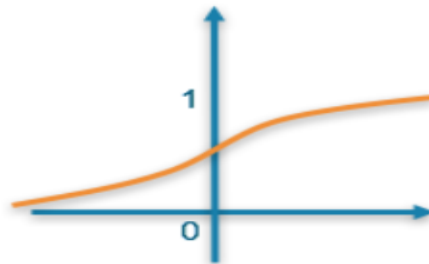


Рисунок 5 – Сигмовидная функция активации

Формула сигмовидной функции имеет вид

$$f(z) = \frac{1}{1 + e^{-z}}. \quad (2)$$

Функция активации выпрямленного линейного блока (ReLU) представлена на рисунке 6, ее характеристики [49]:

- 1) нелинейная функция;
- 2) значения могут стать очень большими;

- 3) так как он не учитывает отрицательные значения, некоторые шаблоны могут быть не зафиксированы;
- 4) градиент может стремиться к 0, поэтому веса не обновляются: «проблема с умирающим ReLU»;
- 5) диапазон от 0 до  $\infty$ .

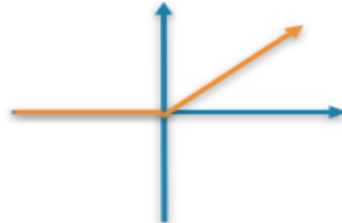


Рисунок 6 – Функция активации выпрямленного линейного блока  
 Формула активации выпрямленного линейного блока имеет вид

$$relu(z) = \max(0, z). \quad (3)$$

Функция активации Softmax (рисунок 7) [49]:

- 1) каждое значение находится в диапазоне от 0 до 1, а сумма всех значений равна 1, поэтому может использоваться для моделирования вероятностных распределений;
- 2) используется только в выходном слое, а не по всей сети.



Рисунок 7 – Функция активации Softmax

Формула активации Softmax имеет вид

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j^n \exp(z_i)}. \quad (4)$$

## 2.5 Функции потерь

Функция потерь – функция, которая в теории статистических решений характеризует потери при неправильном принятии решений на основе наблюдаемых данных.

Квадратичная ошибка потери, также известного как L2 Loss, представляет собой квадрат разности между фактическим и прогнозируемым значениями [18].

Формула функции потерь имеет вид

$$L = (y - f(x))^2. \quad (5)$$

Кросс-энтропийная потеря измеряет перекрестную энтропию между предсказанным и фактическим значением [18].

Кросс-энтропия используется:

- 1) в классификационные задачи;
- 2) для более высокой точности.

Формула кросс-энтропии имеет вид

$$\text{loss}(x, y) = - \sum_i^n x \log y, \quad (6)$$

где  $x$  – вероятность истинной метки;

$y$  – вероятность предсказанной метки.



### 3 Создание нейронной сети

Разработка нейронной сети прямого распространения. Графически ее можно представить следующим образом (рисунок 8).

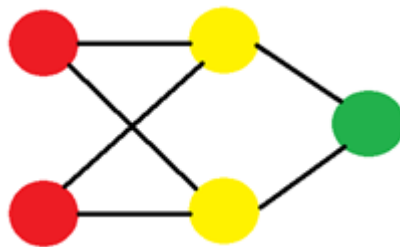


Рисунок 8 – Нейронная сеть прямого распространения

Красные круги – входные данные или входной слой, желтые круги – скрытый слой, зеленый круг – выходные данные или выходной слой. Функция активации была выбрана сигмоид (сигмоидальная функция), данная функция монотонно возрастает всюду дифференцируемая S-образная нелинейная с насыщением. Причины выбора сигмоидной функции активации послужили:

- 1) нелинейная функция, поэтому она может захватывать более сложные шаблоны;
- 2) выходные значения ограничены, поэтому не становится слишком большими.

График функции представлен на рисунке 9.

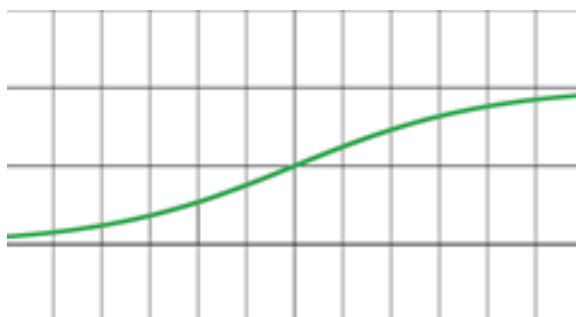


Рисунок 9 – График сигмоиды

Формула:

$$f(x) = \frac{1}{1 + e^{-x}}. \quad (7)$$

Также использовать bias (нейрон смещения), это позволит сдвигать функцию активации вправо и влево таким образом, чтобы найти большее число нужных весов.

Нейронная сеть была реализована на языке программирования Python. Рассмотрим блоки программного кода.

Тренировать и тестировать нейронную сеть можно с помощью набора данных cifar10, данный набор содержит 70 тысяч нарисованных цифр от 0 до 9, из которых 60 тысяч картинок используются для тренировки и 10 тысяч картинок используются для тестирования (рисунок 10) [56].

```
In [33]: 1 # Загрузка данных
          2 (x_train, y_train), (x_test, y_test) = mnist.load_data ()
```

Рисунок 10 – Загрузка набора данных cifar10

На рисунке 11 представлен блок кода, который производит обучение нейронной сети.

```
14 def train(self, inputs_list, targets_list):
15     # добавляем 1 вход под bias
16     inputs_list=np.concatenate([inputs_list,[1]], axis=0)
17     # вектор-столбцы входных данных и правильных ответов
18     inputs = np.array(inputs_list, ndmin=2).T
19     targets = np.array(targets_list, ndmin=2).T
20     # прямое распространение. сигмоид и линейар.
21     hid_results = self.activation_function(np.dot(self.w_ih, inputs))
22     # применяем активационную функцию к прямому распространению к выходному слою
23     out_results = self.activation_function(np.dot(self.w_ho, hid_results))
24     # ошибка вывода
25     out_errors = (targets - out_results)
26     # ошибка скрытого слоя
27     hid_errors = np.dot(self.w_ho.T, out_errors)
28     # поправки для весов скрытый-выход
29     # средний слой
30     # чикаем обратное распространение
31     # вернули сигмоиду, был линейный выход ( )
32     self.w_ho += self.lr * np.dot(out_errors * out_results * (1.0 - out_results), np.transpose(hid_results))
33     # поправки для весов входа-скрытый
34     # слой выходных сигналов
35     self.w_ih += self.lr * np.dot((hid_errors * hid_results * (1.0 - hid_results)), np.transpose(inputs))
```

Рисунок 11 – Функция Train

Процесс обучения нейронной сети представлен на рисунке 12 После того, как эпохи (обучение) будут давать незначительный результат следует уменьшить `myAI.set_lr` до 0.01, это позволит более точно обучить нейронную сеть.

```

1 def epoch_train():
2     # можно понизить чтобы лучше обучить, более мелкая коррекция ошибок
3     myAI.set_lr(0.10)
4     # длина массива тренировочных данных (x.train = 60к, 128*128)
5     x_train_len = len(x_train)
6     # вызываем
7     for n in range(x_train_len):
8         # изменяем поворот картинки trainR(n)
9         trainR(n)
10        if (n % 10000 == 0):
11            sys.stdout.write("Row: %s\r" % n)
12            sys.stdout.flush()

```

1 epoch\_train()

Row: 50000

Рисунок 12 – Обучение нейронной сети

Результат тренировок представлен на рисунке 13, где epoch\_test\_t, это результат прохождения на тренировочном наборе, а epoch\_test, это результат прохождений на тестовом наборе [56].

```

In [61]: epoch_train()
print (epoch_test_t())
print (epoch_test())

```

0.9991833333333333

0.9752

Рисунок 13 – Результат обучения нейронной сети

Проверка как нейронная сеть распознает рукописные изображения. На рисунке 14 нейронная сеть спрогнозировала, верно, на рисунке 15 нейронная сеть не справилась. Из десяти рукописных картинок нейронная сеть дала четыре верных прогноза.

```

In [81]: 1 image = color.rgb2gray(io.imread("F:/test0.png"))
2 query_vec = np.array(1-image).reshape(784)
3 plt.show()
4 plt.imshow(image, cmap="gray")
5 myAI.query(query_vec).argmax()
6 #print(img)

```

Out[81]: 0

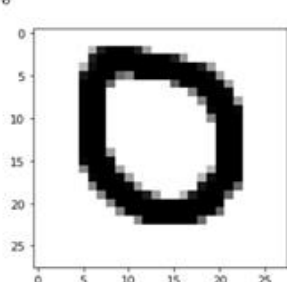


Рисунок 14 – Верный прогноз нейронной сети

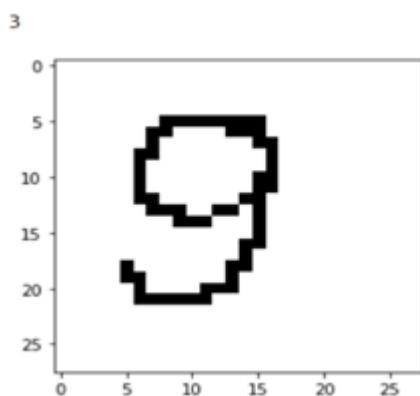


Рисунок 15 – Неверный прогноз нейронной сети

После анализа данного типа нейронной сети были выявлены следующие недостатки, нейронная сеть плохо справляется с распознаванием изображений, которые не так сильно отличаются от тренировочных, также нейронная сеть принимает фиксированное значение пикселей.

Более лучшие результаты в области распознавания лиц показывает сверточная нейронная сеть, которая является логическим развитием архитектур нейронных сетей как когнитрона и неокогнитрона. Преимущества свёрточной нейронной сети является учет двумерной топологии изображения, в отличие от многослойного персептрона [53].

### **3.1 Создание VGGNet подобную архитектуру свёрточной нейронной сети с помощью библиотеки Keras и бэкэнда tensorflow**

Входной слой или входные данные представляют из себя цветные изображения типа JPEG или др., размера 48x48 пикселей или др. Если размер будет слишком велик, то вычислительная сложность повысится, соответственно ограничения на скорость ответа будут нарушены, определение размера в данной задаче решается методом подбора. Если выбрать размер слишком маленький, то сеть не сможет выявить ключевые признаки, например, лиц. Каждое изображение разбивается на 3 канала: красный, синий, зеленый. Таким образом получается 3 изображения размера 48x48 пикселей.

Входной слой учитывает двумерную топологию изображений и состоит из нескольких карт (матриц), карта может быть одна, в том случае, если изображение представлено в оттенках серого, иначе их 3, где каждая карта соответствует изображению с конкретным каналом (красным, синим и зеленым).

Входные данные каждого конкретного значения пикселя нормализуются в диапазон от 0 до 1, по формуле

$$f(p, min, max) = \frac{p - min}{max - min}, \quad (8)$$

где  $p$  – значение конкретного цвета пикселя от 0 до 255;

$min$  – минимальное значение пикселя – 0;

$max$  – максимальное значение пикселя – 255.

Сверточный слой представляет из себя набор карт (карты признаков или матрицы), у каждой карты есть синаптическое ядро (сканирующее ядро или фильтр).

Количество карт определяется требованиями к задаче, если взять большое количество карт, то повысится качество распознавания, но увеличится вычислительная сложность. Исходя из анализа научных статей, в большинстве случаев предлагается брать соотношение один к двум, то есть каждая карта предыдущего слоя (например, у первого сверточного слоя, предыдущим является входной) связана с двумя картами сверточного слоя, в соответствии с рисунком 16, количество карт – 6.

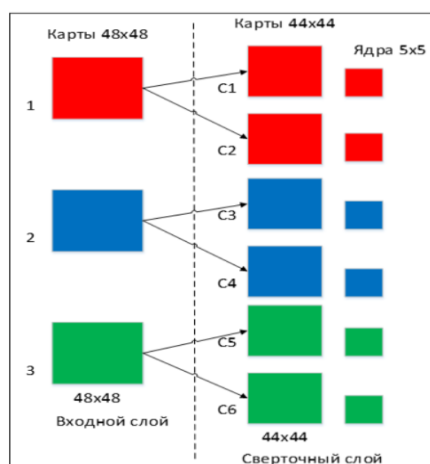


Рисунок 16 – Организация связей между картами сверточного слоя и предыдущего

Размер у всех карт сверточного слоя – одинаковы и вычисляются по формуле

$$(w, h) = (mW - kW + 1, mH - kH + 1), \quad (9)$$

где  $(w, h)$  – вычисляемый размер свёрточной карты;

$mW$  – ширина предыдущей карты;

$mH$  – высота предыдущей карты;

$kW$  – ширина ядра;

$kH$  – высота ядра.

Ядро представляет из себя фильтр или окно, которое скользит по всей области предыдущей карты и находит определенные признаки объектов. Например, если сеть обучали на множестве лиц, то одно из ядер могло бы в процессе обучения выдавать наибольший сигнал в области глаза, рта, брови или носа, другое ядро могло бы выявлять другие признаки. Размер ядра обычно берут в пределах от 3x3 до 7x7. Если размер ядра маленький, то оно не сможет выделить какие-либо признаки, если слишком большое, то увеличивается количество связей между нейронами. Также размер ядра выбирается таким, чтобы размер карт сверточного слоя был четным, это позволяет не терять информацию при уменьшении размерности в подвыборочном слое.

Ядро представляет собой систему разделяемых весов или синапсов, это одна из главных особенностей свёрточной нейросети. В обычной многослойной сети очень много связей между нейронами, то есть синапсов, что весьма замедляет процесс детектирования. В свёрточной сети – наоборот, общие веса позволяют сократить число связей и позволить находить один и тот же признак по всей области изображения. Пример ядра с обученным признаком представлен на рисунке 17 и пример входного изображения представлен на рисунке 18. Где желтое окно, там будет большой отклик (сигнал), что говорит о наличии этого признака на изображении.

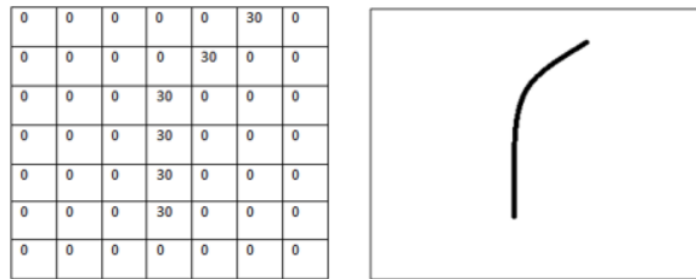


Рисунок 17 – Ядро с обученным признаком

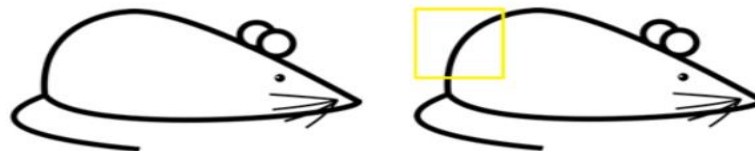


Рисунок 18 – Входное изображение

Изначально значения каждой карты сверточного слоя равны 0. Значения весов ядер задаются случайным образом в области от -0.5 до 0.5. Ядро скользит по предыдущей карте и производит операцию свертки, которая часто используется для обработки изображений.

Формула:

$$(f \times g)[m, n] = \sum_i^n f[m - k, n - l] \times g[k, l]_{k,l}, \quad (10)$$

где  $f$  – исходная матрица изображения;

$g$  – ядро свертки.

Эту операцию можно описать следующим образом – окно размера ядра  $g$  проходим с заданным шагом (обычно 1) все изображение  $f$ , на каждом шаге поэлементно умножаем содержимое окна на ядро  $g$ , результат суммируется и записывается в матрицу результата (рисунок 19).

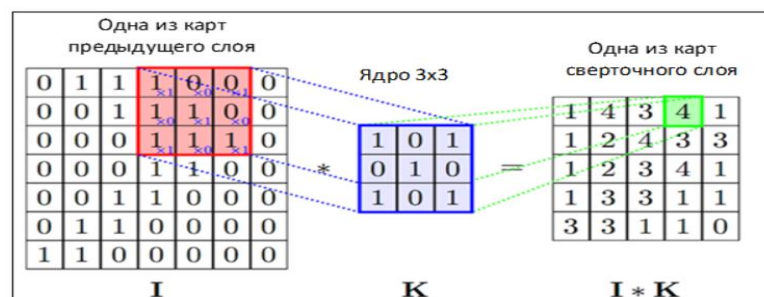


Рисунок 19 – Операция свертки и получение значений свёрточной карты (valid)

В зависимости от метода облюи работки краев исходной матрицы результат может быть меньше исходного изображения (valid), такого же размера (same) или большего размера (full), в соответствии с рисунком 20.

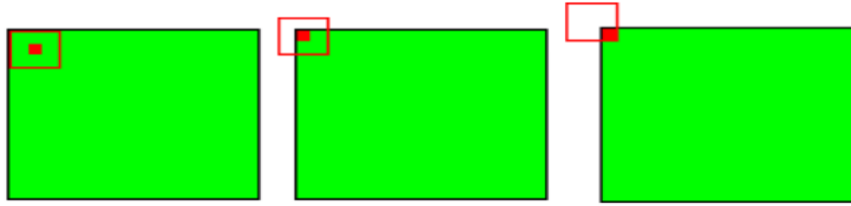


Рисунок 20 – Три вида свертки исходной матрицы

В упрощенном виде этот слой можно описать формулой

$$x^l = f(x^{l-1} * k^l + b^l), \quad (11)$$

где  $x^l$  – выход слоя  $l$ ;

$f()$  – функция активации;

$b^l$  – коэффициент сдвига слоя  $l$ ;

$*$  – операция свертки входа  $x$  с ядром  $k$ .

При этом за счет краевых эффектов размер исходных матриц уменьшается.

Формула

$$x_j^l = f\left(\sum_i^n x_i^{l-1} * k_j^l + b_j^l\right), \quad (12)$$

где  $x_j^l$  – карта признаков  $j$  (выход слоя  $l$ );

$f(x)$  – функция активации;

$b_j^l$  – коэффициент сдвига слоя  $l$  для карты признаков  $j$ ;

$k_j^l$  – ядро свертки  $j$  карты, слоя  $l$ ;

$*$  – операция свертки входа  $x_j^l$  с ядром  $k$ .

Подвыборочный слой также, как и сверточный имеет карты, но их количество совпадает с предыдущим (сверточным) слоем. Цель слоя – уменьшение размерности карт предыдущего слоя. Если на предыдущей операции свертки 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.

В процессе сканирования ядром подвыборочного слоя (фильтром) карты предыдущего слоя, сканирующее ядро не пересекается в отличие от сверточного слоя. Обычно, каждая карта имеет ядро размером  $2 \times 2$ , что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Вся карта признаков разделяется на ячейки  $2 \times 2$  элемента, из которых выбираются максимальные по значению.

В подвыборочном слое обычно применяется функция активации ReLU. Операция подвыборки (или MaxPooling – выбор максимального) в соответствии с рисунком 21. Кроме пулинга с функцией максимума можно использовать и другие функции, например, среднего значения или L2-нормирования. Однако практика показала преимущества именно пулинга с функцией максимума, который включается в типовые системы.

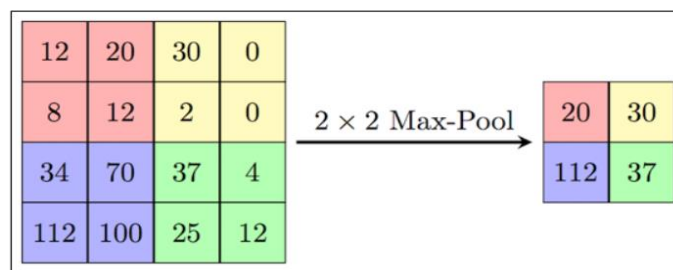


Рисунок 21 – Формирование новой карты подвыборочного слоя на основе предыдущей карты сверточного слоя. Операция подвыборки (Max Pooling)

Слой пулинга, как правило, вставляется после слоя свертки перед слоем следующей свертки. В целях более агрессивного уменьшения размера получаемых представлений, всё чаще находят распространение идеи использования меньших фильтров или полный отказ от слоёв пулинга. Слой можно описать формулой

$$x^l = f(a^l * \text{subsample}(x^{l-1}) + b^l), \quad (13)$$

где  $x^l$  – выход слоя  $l$ ;

$f(x)$  – функция активации;

$a^l, b^l$  – коэффициенты сдвига слоя  $l$ ;

$\text{subsample}()$  – операция выборки локальных максимальных значений.

Исключение или дропаут – метод регуляризации искусственных нейронных сетей, предназначен для уменьшения переобучения сети за счет предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения.

Термин «dropout» (выбивание, выбрасывание) характеризует исключение определённого процента (например, 30%) случайных нейронов (находящихся как в скрытых, так и видимых слоях) на разных итерациях (эпохах) во время обучения нейронной сети. Это очень эффективный способ усреднения моделей внутри нейронной сети. В результате более обученные нейроны получают в сети больший вес. Такой прием значительно увеличивает скорость обучения, качество обучения на тренировочных данных, а также повышает качество предсказаний модели на новых тестовых данных.

Полносвязный слой, это последний из типов слоев. Это слой обычного многослойного персептрона. Цель слоя – классификация, моделирует сложную нелинейную функцию, оптимизируя которую, улучшается качество распознавания. На рисунке 22 представлены расположение данных слоев.

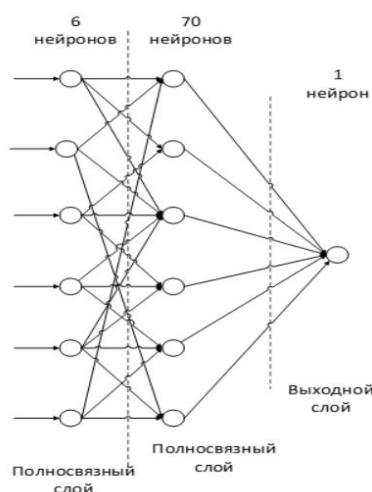


Рисунок 22 – Расположение слоев

Нейроны каждой карты предыдущего подвыборочного слоя связаны с одним нейроном скрытого слоя. Таким образом число нейронов скрытого слоя равно числу карт подвыборочного слоя, но связи могут быть не обязательно такими, например, только часть нейронов какой-либо из карт подвыборочного слоя

быть связана с первым нейроном скрытого слоя, а оставшаяся часть со вторым, либо все нейроны первой карты связаны с нейронами 1 и 2 скрытого слоя. Вычисление значений нейрона можно описать формулой

$$x_j^l = f\left(\sum_i^n x_j^{l-1} * w_j^{l-1} + b_j^{l-1}\right), \quad (14)$$

где  $x_j^l$  – карта признаков  $j$  (выход слоя  $l$ );

$f(x)$  – функция активации;

$b_j^l$  – коэффициент сдвига слоя  $l$ ;

$w_j^{l-1}$  – матрица весовых коэффициентов слоя  $l$ .

Выходной слой или выходные данные. Слой связан со всеми нейронами предыдущего слоя.

Создание и обучение свёрточной нейронной сети будет происходить с помощью библиотеки Keras. Созданная архитектура свёрточной нейронной сети будет напоминать архитектуру VGGNet (рисунок 23, 24) [99].

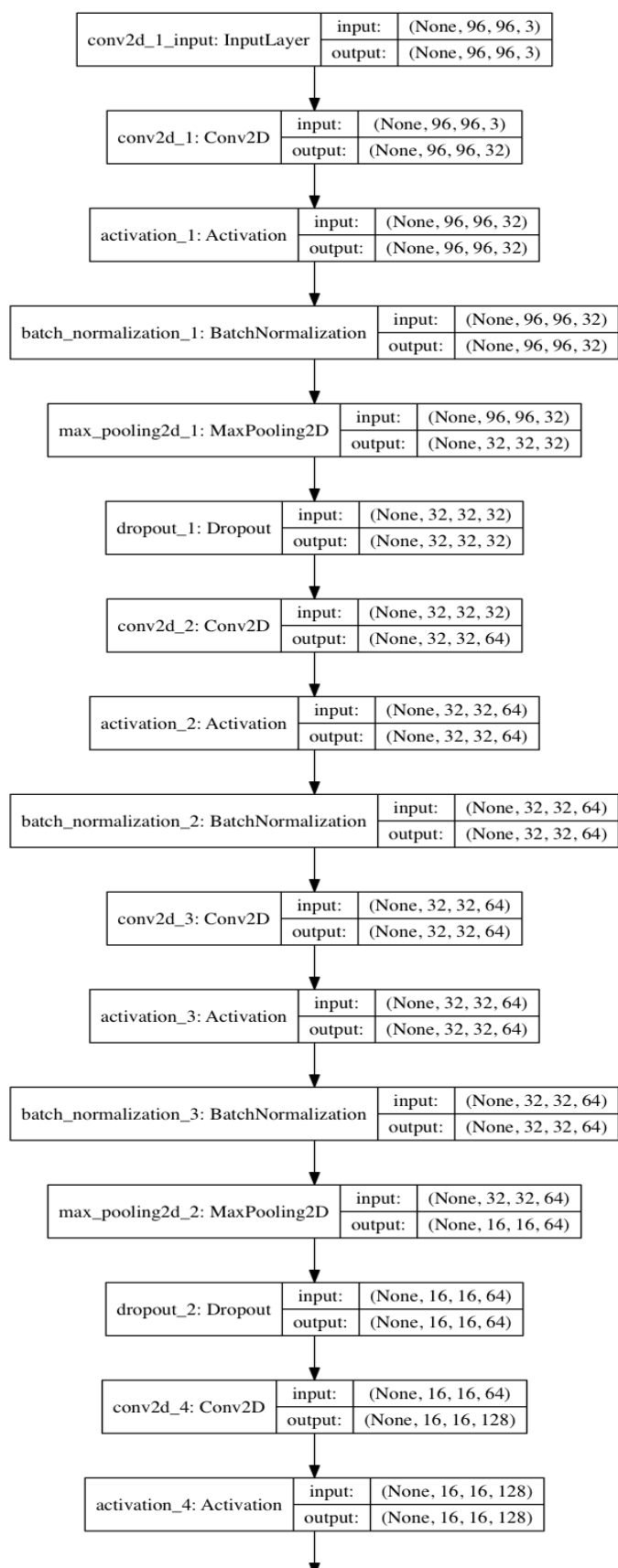


Рисунок 23 – Архитектура свёрточной нейронной сети. Часть 1

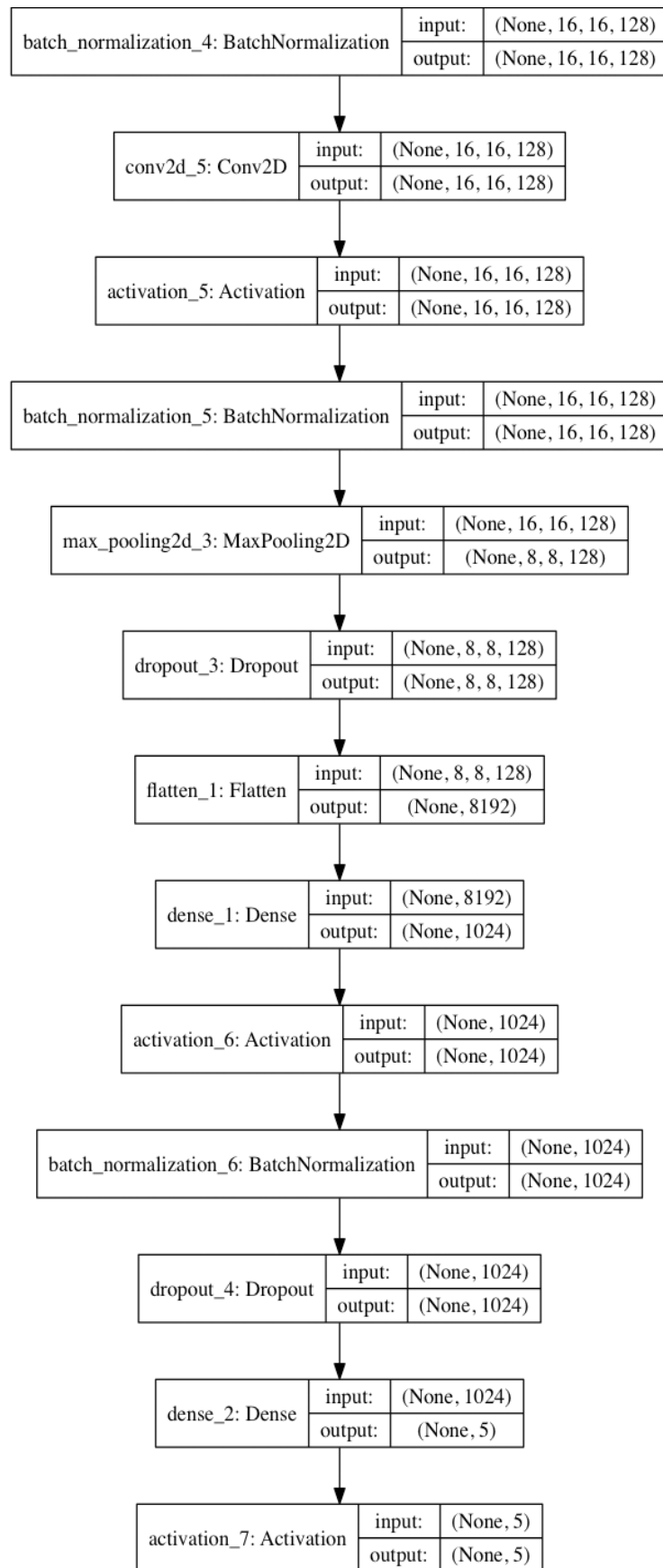
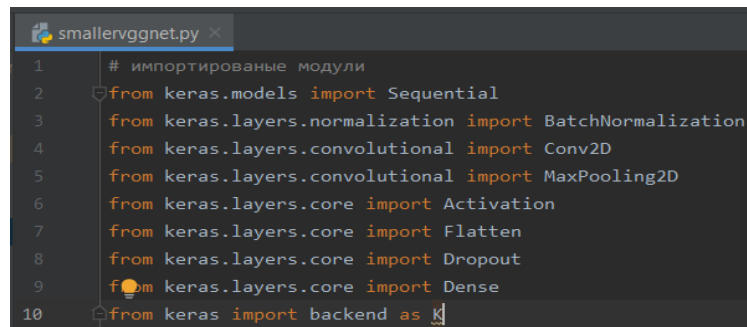


Рисунок 24 – Архитектура свёрточной нейронной сети. Часть 2

VGGNet-подобные архитектуры характеризуются:

- 1) использованием только  $3 \times 3$  сверточных слоев, уложенных друг на друга для увеличения глубины;
- 2) уменьшением размера тома путем максимального объединения;
- 3) полностью связанными слоями в конце сети до классификатора softmax.

Архитектура свёрточной нейронной сети программируется в файле «smallervggnet.py». Для этого нужно импортировать соответствующие модули (рисунок 25).

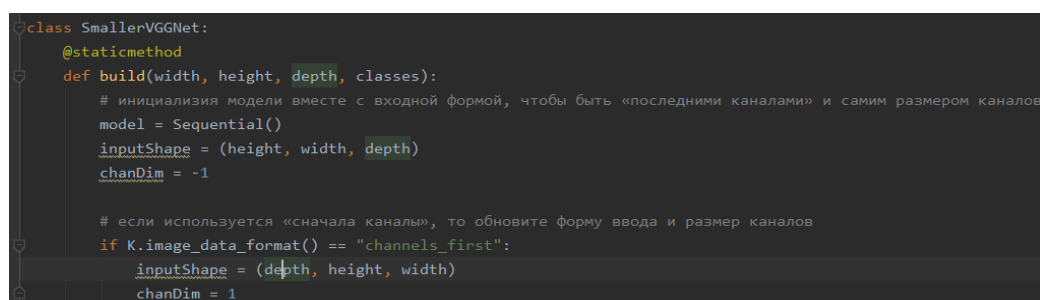


```
1 # импортированные модули
2 from keras.models import Sequential
3 from keras.layers.normalization import BatchNormalization
4 from keras.layers.convolutional import Conv2D
5 from keras.layers.convolutional import MaxPooling2D
6 from keras.layers.core import Activation
7 from keras.layers.core import Flatten
8 from keras.layers.core import Dropout
9 from keras.layers.core import Dense
10 from keras import backend as K
```

Рисунок 25 – Импортированные модули в smallervggnet.py

В данном файле определяется класс SmallerVGGNet. В классе реализован метод build (рисунок 26), метод принимает четыре параметра:

- 1) width: размер изображения по ширине;
- 2) height: размер изображения по высоте;
- 3) depth: глубина изображения – также называется количеством каналов;
- 4) classes: количество классов в наборе данных, которые будут влиять на последний слой модели.



```
class SmallerVGGNet:
    @staticmethod
    def build(width, height, depth, classes):
        # инициализация модели вместе с входной формой, чтобы быть «последними каналами» и самим размером каналов
        model = Sequential()
        inputShape = (height, width, depth)
        chanDim = -1

        # если используется «сначала каналы», то обновите форму ввода и размер каналов
        if K.image_data_format() == "channels_first":
            inputShape = (depth, height, width)
            chanDim = 1
```

Рисунок 26 – Параметры метода в классе SmallerVGGNet

Работа происходит с входными изображениями, которые 96 x 96 с глубиной 3.

Создание первого слоя (рисунок 27).

```
# CONV => RELU => POOL
model.add(Conv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
```

Рисунок 27 – Первый слой в классе SmallerVGGNet

Сверточный слой имеет 32 фильтра с 3 x 3 ядром. Функция активации используется ReLU с последующей нормализацией партии. Далее идет слой пулинга, слой использует 3 x 3 пул размером, чтобы быстро уменьшить пространственные размеры от 96 x 96 в 32 x 32.

Также используется дропаут в сетевой архитектуре. Дропаут работает путем случайного отключения узлов от текущего слоя до следующего слоя. Этот процесс случайного разъединения во время тренировочных пакетов помогает естественным образом ввести избыточность в модель. Ни один узел в слое не отвечает за прогнозирование определенного класса, объекта, ребра.

Далее создаются слои перед нанесением другого пулинга слоя. Это повторяется два раза. (рисунок 28).

```
# (CONV => RELU) * 2 => POOL
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# (CONV => RELU) * 2 => POOL
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

Рисунок 28 – Следующие слои перед нанесением следующего pool слоя

Укладка нескольких сверточных и ReLU слоев вместе (до уменьшения пространственных размеров объема) позволяет изучить более богатый набор функций.

Стоит обратить внимание на:

1) увеличение размера фильтра с 32 на 64 и далее на 128, чем глубже углубляться в сеть, тем меньше пространственные измерения объема и тем больше фильтров изучается;

2) уменьшение максимального размера пула с 3 x 3 в 2 x 2 чтобы не сокращать пространственные размеры слишком быстро;

3) дроп-аут выполняется на каждом этапе. Дропаут 25% узлов выполняется, чтобы уменьшить переоснащение.

И наконец, набор FC (полносвязный слой) и RELU слои и классификатор softmax (рисунок 29).

```
# первый (и единственный) набор слоев FC => RELU
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# классификатор softmax
model.add(Dense(classes))
model.add(Activation("softmax"))

# вернуть построенную сетевую архитектуру
return model
```

Рисунок 29 – Набор слоев FC => RELU

Между сверточными слоями Conv2D и слоем пулинга (Dense) находится слой выравнивания (Flatten). Он служит соединительным узлом между слоями.

Полностью связанный слой определяется как softmax 1024 с выпрямленной линейной активацией единицы и нормализацией партии. Дропаут настроен так что пропускаем 50% узлов во время обучения.

Обучение свёрточной нейронной сети реализовано в файле в «train.py». Им-портируемые необходимые пакеты и библиотеки представлены на рисунке 30.



Для создания графика после обучения используется «Agg» бэкэнд matplotlib, позволяющий сохранять фигуры в фоновом режиме. Класс ImageDataGenerator будет использоваться для дополнения данных.

```
# импортированные модули
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from pyimagesearch.smallervggnet import SmallerVGGNet
import matplotlib.pyplot as plt
from imutils import paths
import numpy as np
import argparse
import random
import pickle
import cv2
import os

# установить фоновые рисунки Matplotlib можно сохранить в фоновом режиме
import matplotlib
matplotlib.use("Agg")
```

Рисунок 30 – Модули в файле train.py

Этот метод используется для съемки существующих изображений в обучающую выборку и применения к ним случайных преобразований (поворотов, сдвига и т. д.) для создания дополнительных обучающих данных. Увеличение данных помогает предотвратить переоснащение. Adam оптимизатор, метод оптимизатора, используемый для обучения нейронной сети. Класс LabelBinarizer бинаризирует метки в едином стиле. Обучение состоит в изучении одного регрессора или двоичного классификатора на класс. При этом необходимо преобразовать мультиклассовые метки в двоичные метки (принадлежит или не принадлежит к классу). LabelBinarizer облегчает этот процесс с помощью метода transform.

Во время прогнозирования присваивается класс, для которого соответствующая модель дала наибольшую достоверность. LabelBinarizer делает это легко с помощью метода inverse\_transform. LabelBinarizer преобразовывает название класс, которое является строкой и берется с названия каталога в целое число и обратно. Это нужно для использования меток в нейронной сети и при получении прогноза от нейронной сети чтобы перевести обратно в читаемый вид название метки.

Функция `train_test_split` используется для создания тренировок и тестирований. Класс `SmallerVGGNet`, это созданная ранее сверточная нейронная сеть с помощью библиотеки `Keras`.

На рисунке 31 видны основные переменные для обучения нейронной сети. Значение переменных:

1) `EPOCHS`, общее количество эпох, это сколько раз нейронная сеть «видит» каждый пример обучения и изучает образцы из него;

2) `INIT_LR`, начальная скорость обучения – значение  $1e-3$  является значением по умолчанию для оптимизатора `Adam`;

3) `BS`. Во время обучения передаются партии изображений в нейронную сеть. В эпохах есть несколько партий. `BS`, это значение, которое контролирует размер пакета;

4) `IMAGE_DIMS`, предоставление пространственных размеров входных изображений. Для обучения требуется, чтобы входные изображения были  $96 \times 96$  пикселей с 3 каналы (т. е. RGB).

Также инициализируются два списка – `data` и `labels`, которые будут содержать предварительно обработанные изображения и метки. Метки, это название классов классификации, которые берутся с названий каталогов, в которых содержатся обучающая выборка, состоящая из фотографий личности. Далее идет программный код, который захватывает все пути к изображениям и случайным образом перетасовывает их.

```
# инициализация количеств эпох для обучения, начальной скорости обучения, размера пакета и размеров изображений
EPOCHS = 100
INIT_LR = 1e-3
BS = 32
IMAGE_DIMS = (96, 96, 3)

# инициализация данных и меток
data = []
labels = []

# захватить пути изображений и случайным образом перемешать их
print("[INFO] загрузка изображений...")
imagePaths = sorted(list(paths.list_images(args["dataset"])))
random.seed(42)
random.shuffle(imagePaths)
```

Рисунок 31 – Основные переменные для обучения нейронной сети

С глубоким обучением или любым машинным обучением в этом отношении обычной практикой является разделение обучения и тестирования. На рисунке 32 программный код, который создает случайное разбиение данных на две части, деление происходит 80 на 20. Далее создается объект дополнения данных к существующим изображениям. Поскольку работа происходит с ограниченным количеством данных (200-330 изображений в классе), поэтому можно использовать расширение данных в процессе обучения, чтобы дать модели больше изображений. Новые изображения создаются на основе существующих изображений для обучения.

```
# разбить данные на разделы обучения и тестирования, используя 80% данных для обучения и оставшиеся 20% для тестирования
(trainX, testX, trainY, testY) = train_test_split(data, labels, test_size=0.2, random_state=42)

# построить генератор изображений для увеличения данных
aug = ImageDataGenerator(rotation_range=25, width_shift_range=0.1,
                        height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                        horizontal_flip=True, fill_mode="nearest")
```

Рисунок 32 – Создания обучающей выборки

На рисунке 33 представлена компиляция модели и обучение нейронной сети. Сначала происходит инициализация созданной модели с 96x96x3 входными пространственными размерами. Далее используется Adam оптимизатор с затуханием скорости обучения, а затем компиляция модели с категорической кросс-энтропией. А двоичная кросс-энтропия используется как потерю только для двух классов. После чего метод `fit_generator` из Keras начинает обучение нейронной сети.

```
# инициализации модель
print("[INFO] составление модели...")
model = SmallerVGGNet.build(width=IMAGE_DIMS[1], height=IMAGE_DIMS[0],
                            depth=IMAGE_DIMS[2], classes=len(lb.classes_))
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])

# тренировка сети
print("[INFO] учебная сеть...")
H = model.fit_generator(
    aug.flow(trainX, trainY, batch_size=BS),
    validation_data=(testX, testY),
    steps_per_epoch=len(trainX) // BS,
    epochs=EPOCHS, verbose=1)
```

Рисунок 33 – Программный код компиляции модели и обучения

После обучения нейронной сети идет сохранение модели и бинаризатора меток (рисунок 34).

```
# сохранить модель на диск
print("[INFO] сериализация сети...")
model.save(args["model"])

# сохранить метку бинаризатора на диск
print("[INFO] сериализатор меток бинаризатор...")
f = open(args["labelbin"], "wb")
f.write(pickle.dumps(lb))
f.close()
```

Рисунок 34 – Экспорт данных после обучения

Создание графика после обучения представлен на рисунке 35. На рисунке 36 изображен созданный график.

```
# график обучения потери и точности
N = EPOCHS
plt.style.use("ggplot")
plt.figure()
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["accuracy"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_accuracy"], label="val_acc")
plt.title("Training Loss and Accuracy")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

Рисунок 35 – Программный код создания графика

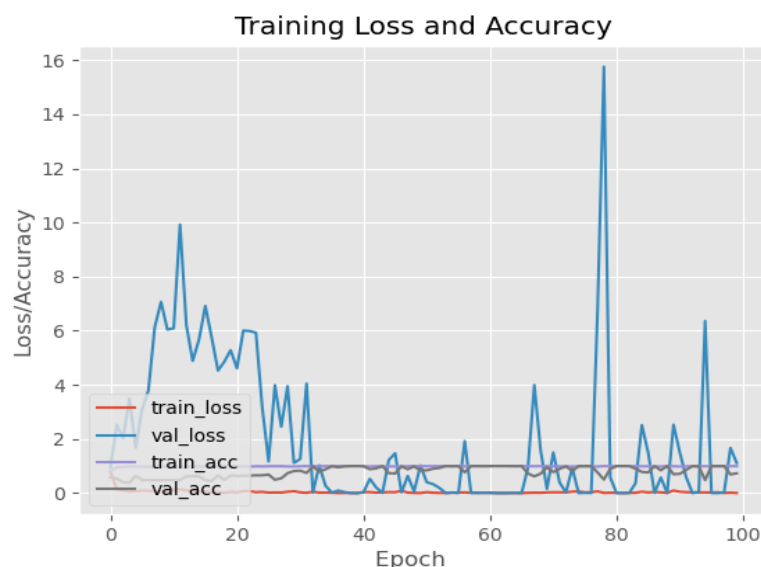


Рисунок 36 – График обучения VGGNet подобной архитектуры свёрточной нейронной сети

На рисунке 37 представлены каталоги с обучающей выборкой. На рисунке 38 продемонстрировано содержание одного из нескольких каталогов.

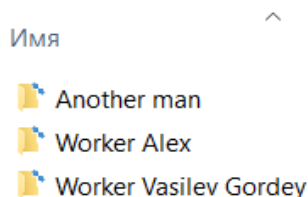


Рисунок 37 – Каталоги с обучающей выборкой

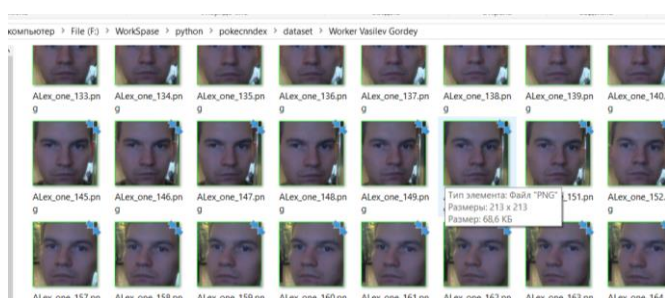


Рисунок 38 – Содержание каталога Worker Vasilev Gordey

Процесс обучения представлены на рисунках 39-40, создания графика представлено на рисунке 41.

```
Using TensorFlow backend.
[INFO] загрузка изображений...
[INFO] матрица данных: 163.30MB
[INFO] составление модели...
2020-05-15 23:34:57.875065: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
[INFO] учебная сеть...
Epoch 1/100
18/18 [=====] - 18s 982ms/step - loss: 0.7798 - accuracy: 0.8077 - val_loss: 0.9446 - val_accuracy: 0.5855
Epoch 2/100
18/18 [=====] - 18s 978ms/step - loss: 0.1430 - accuracy: 0.9528 - val_loss: 2.5260 - val_accuracy: 0.5329
Epoch 3/100
18/18 [=====] - 18s 1s/step - loss: 0.1037 - accuracy: 0.9650 - val_loss: 2.0228 - val_accuracy: 0.4013
Epoch 4/100
18/18 [=====] - 17s 970ms/step - loss: 0.0573 - accuracy: 0.9808 - val_loss: 3.4926 - val_accuracy: 0.3816
Epoch 5/100
```

Рисунок 39 – Начало обучения VGGNet подобной архитектуры свёрточной нейронной сети

```
Epoch 95/100
18/18 [=====] - 15s 833ms/step - loss: 0.0250 - accuracy: 0.9930 - val_loss: 6.3638 - val_accuracy: 0.4803
Epoch 96/100
18/18 [=====] - 15s 840ms/step - loss: 0.0208 - accuracy: 0.9930 - val_loss: 0.0029 - val_accuracy: 1.0000
Epoch 97/100
18/18 [=====] - 14s 791ms/step - loss: 0.0284 - accuracy: 0.9878 - val_loss: 1.3791e-05 - val_accuracy: 1.0000
Epoch 98/100
18/18 [=====] - 14s 750ms/step - loss: 0.0068 - accuracy: 0.9965 - val_loss: 0.0240 - val_accuracy: 1.0000
Epoch 99/100
18/18 [=====] - 14s 791ms/step - loss: 0.0147 - accuracy: 0.9965 - val_loss: 1.6704 - val_accuracy: 0.6908
Epoch 100/100
18/18 [=====] - 14s 757ms/step - loss: 0.0036 - accuracy: 0.9965 - val_loss: 1.1114 - val_accuracy: 0.7303
[INFO] сериализация сети...
[INFO] сериализатор меток бинаризатор...
```

Рисунок 40 – Завершение обучения VGGNet подобной архитектуры свёрточной нейронной сети

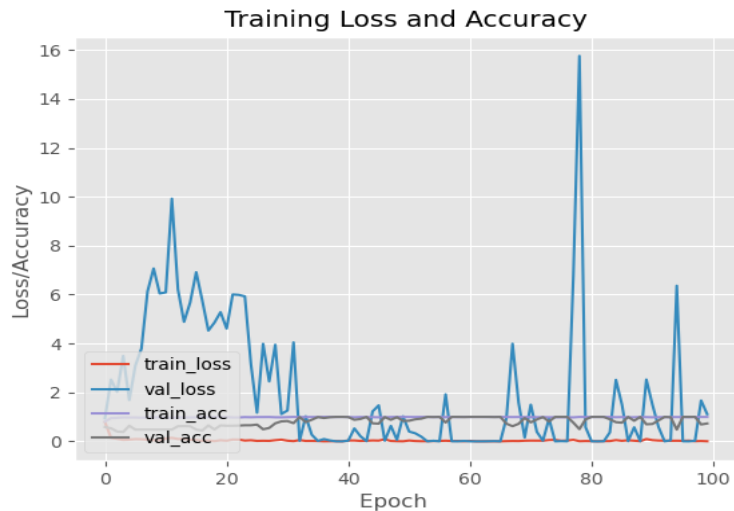


Рисунок 41 – График обучения VGGNet подобной архитектуры свёрточной нейронной сети

После обучение нейронной сети можно распознавать лицо (рисунок 42-43).

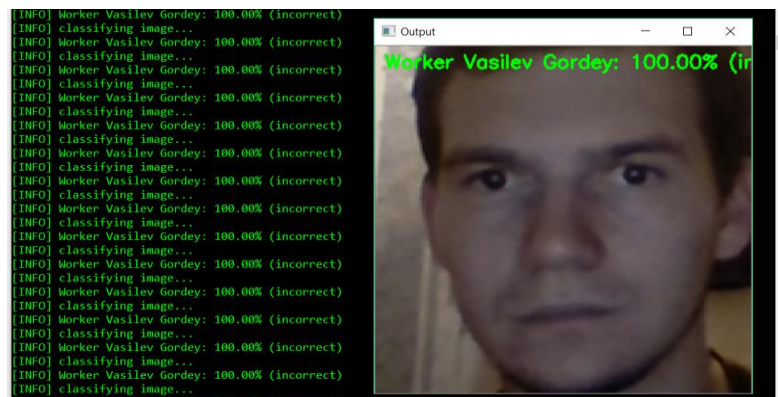


Рисунок 42 – Распознавание работника

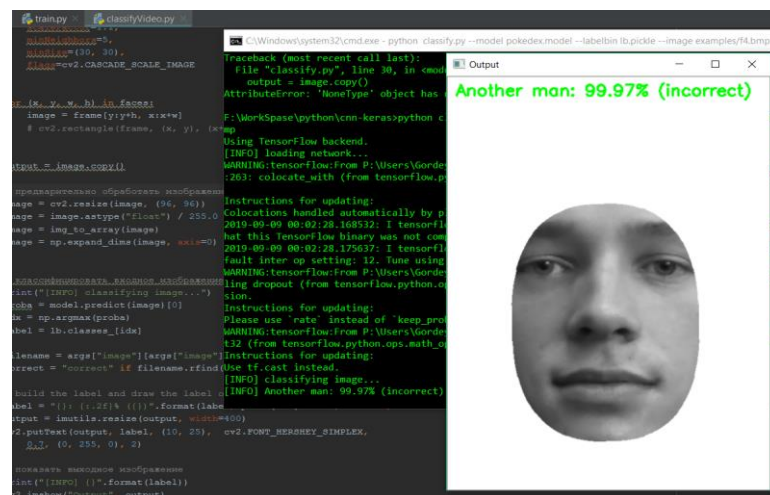


Рисунок 43 – Неизвестный человек

Главный недостаток созданной VGGNet подобной архитектуры свёрточной нейронной сети, это недостаточность обучаемости. Процент верных прогнозов нейронной сети падает спустя несколько дней. Это следует из-за того, что были всего создано 200-320 изображений для обучающей выборки. Все изображения сделаны с конкретными условиями в данное место и времени.

Решение данной проблемы будет обучение нейросети на большой обучающей выборке (несколько миллионов данных). Таким образом можно использовать предобученную нейронную сеть и дообучать ее на со нужной обучающей выборке.

### **3.2 Использование свёрточной нейронной сети из библиотеки face\_recognition**

В данной библиотеки используется распознавание лиц на основе глубокого обучения, включая концепцию «глубокого метрического обучения». Целью обучения метрики является измерение сходства между образцами при использовании оптимальной метрики расстояния для задач обучения. Метрические методы обучения, которые обычно используют линейную проекцию, ограничены в решении реальных задач, демонстрирующих нелинейные характеристики. Глубокое изучение метрик дало впечатляющие результаты в таких задачах, как кластеризация и извлечение изображений, благодаря использованию нейронных сетей для получения встраиваемых объектов с высокой степенью дискриминации, которые можно использовать для группировки выборок в разные классы. Много исследований было посвящено разработке функций интеллектуальных потерь или стратегий интеллектуального анализа данных для обучения таких сетей. Большинство методов рассматривают только пары или тройки выборок в мини-пакете для вычисления функции потерь, которая обычно основана на расстоянии между вложениями. Можно использовать Group Loss, функцию потерь, основанную на дифференцируемом методе распространения меток, который обеспечи-



вает встраивание сходства во все выборки группы, в то же время продвигая области с низкой плотностью среди точек данных, принадлежащих к различным группам.

В глубоком обучении обычно обучается сеть следующим образом:

- 1) принимается одно входное изображение;
- 2) далее вывод классификации или ярлык (метку) для этого изображения.

Глубокое изучение метрики отличается. Вместо того, чтобы пытаться вывести одну метку, выводится вектор с равнозначными объектами. Для сети распознавания лиц dlib вектор выходных объектов равен 128 (список из 128 вещественных чисел), который используется для количественного определения лица. Обучение в сети осуществляется с помощью триплетов (рисунок 44). Триплет состоит из 3 уникальных изображений лица, которые 2 из 3 являются одним и тем же человеком. Нейронная сеть генерирует 128 вектор для каждого из 3 изображений лица. Для двух изображений лица одного и того же человека настраивается веса нейронной сети, чтобы сделать вектор ближе с помощью метрики расстояния [81].

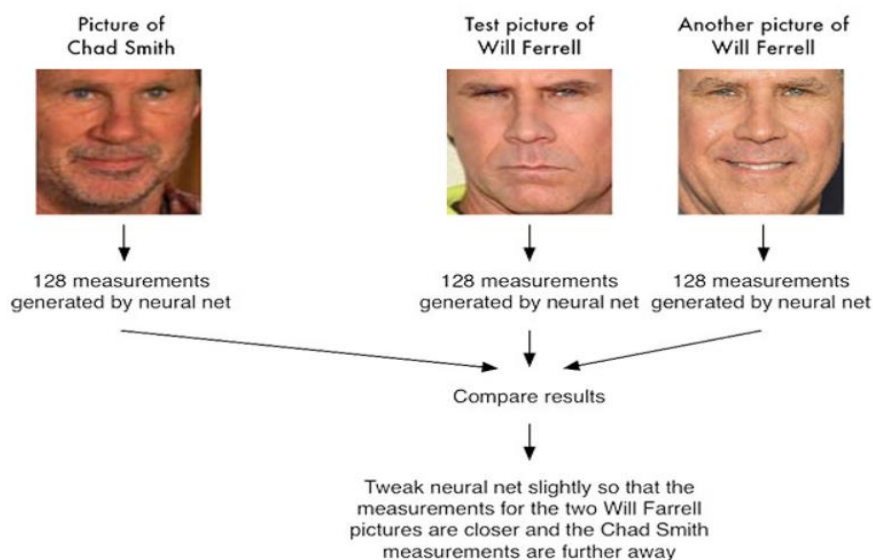


Рисунок 44 – Распознавание лиц с помощью глубокого обучения метрике включая триплетный этап обучения

Значения трех переданных изображений в нейронную сеть:



1) два из них изображения являются примерами лиц одного и того же человека;

2) третье изображение – это случайное лицо из набора данных, которое не совпадает с двумя другими изображениями.

Процесс обучения работает, глядя на 3 изображения лица одновременно:

1) загрузка тренировочного изображения лица известного человека;

2) загрузка другой фотографии того же известного человека;

3) загрузка фотографии совершенно другого человека.

Нейросеть количественно оценивает грани, создавая 128-мерное вложение (квантификация) для каждого изображения. Исходя из этого, общая идея заключается в том, что вес нейронной сети настраиваются так, чтобы 128 измерения двух изображений были ближе друг к другу и дальше от измерений третьего случайного изображения.

Сетевая архитектура для распознавания лиц основана на ResNet-34, но с меньшим количеством слоев и уменьшением количества фильтров вдвое. Нейросеть была обучена на наборе данных около 3 миллионов изображений. Нейросеть сравнивается с другими современными методами, достигая точности 99,38%. Во время классификации используется модель k-NN + голоса для окончательной классификации лиц.

Для данной нейросети достаточно 20 изображение в датасете. Созданный датасет представлен на рисунке 45. Датасет состоит из 46 изображений без маски и 43 изображений с маской, данная нейронная сеть может распознавать личность в маске, если в датасете будут соответствующие изображения на класс.

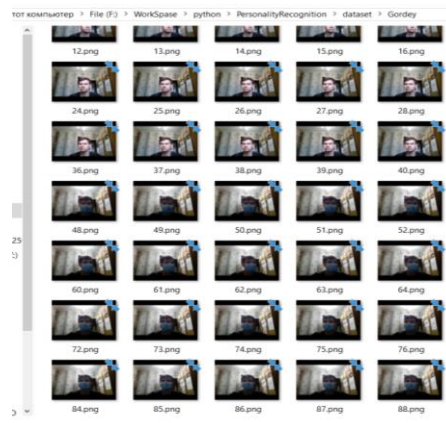


Рисунок 45 – Созданный датасет состоящий из фотографий с наличием маски и без

Результат работы нейросети представлен на рисунках 46-49. На рисунке 48 нейросеть может распознавать личность в медицинской маске, так как в датасете были добавлены соответствующие фотографии. На рисунке 49 можно увидеть, что нейросеть распознает личность по распечатанной фотографии. Избежать данных ситуаций можно с помощью машинного зрения или дополнительной нейросети.

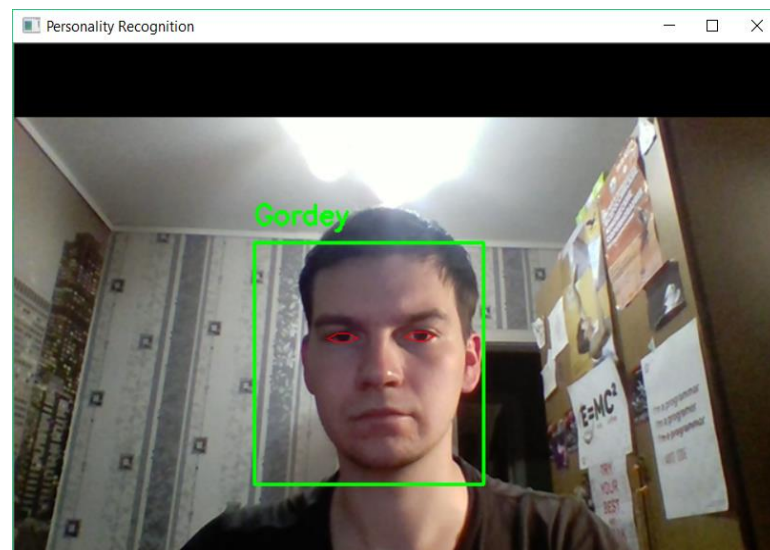


Рисунок 46 – Распознавание личности

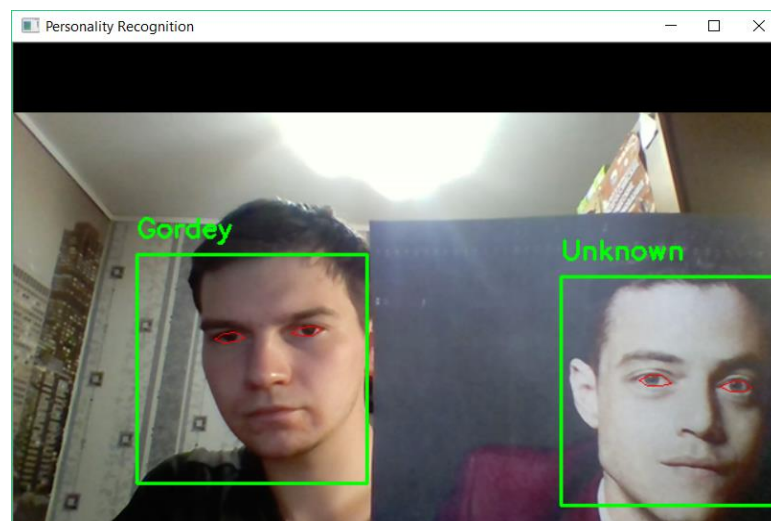


Рисунок 47 – Обнаружения неизвестной личности

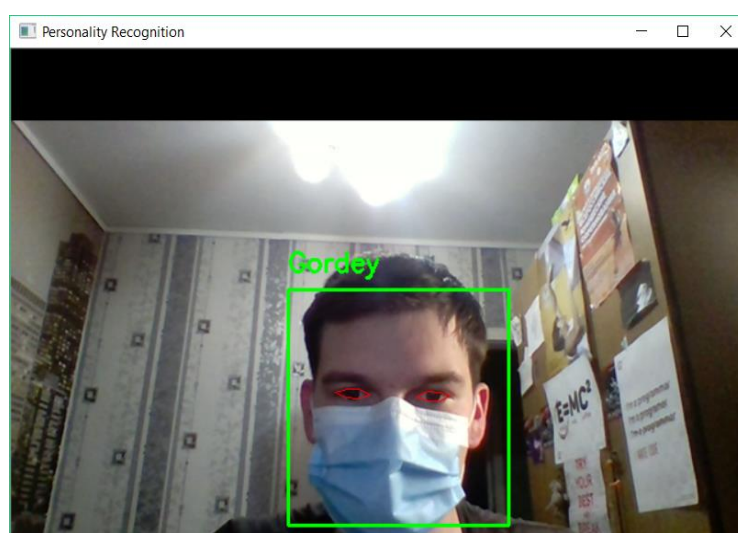


Рисунок 48 – Распознавание личности в маске

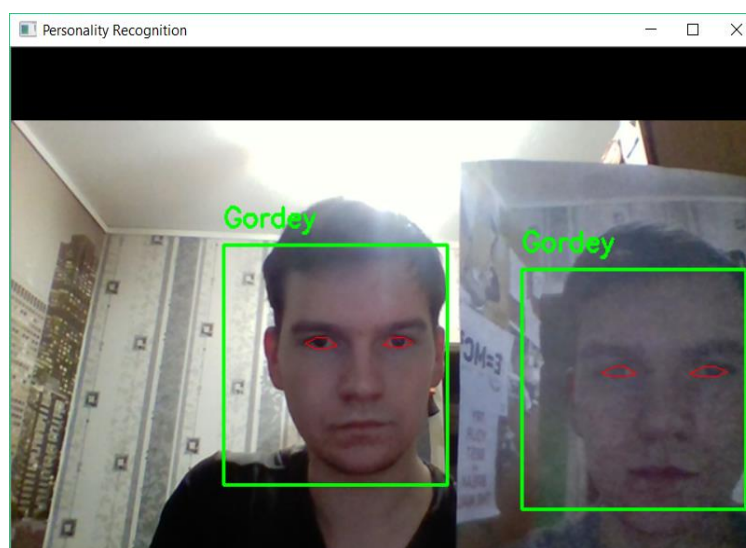


Рисунок 49 – Уязвимость в нейросети, распознавание личность по фотографии

### 3.3 Распознавание наличия маски на лице используя свёрточную нейронную сеть и библиотеки Keras с бэкэндом TensorFlow

В 2020 году стали актуальны меры по предостережениям распространения вирусов. В компаниях, предприятиях, заводах, государственных учреждениях ввели масочный режим. Именно эту меру можно начать контролировать с помощью нейросети.

Нейросеть разбивается на два этапа (рисунок 50).

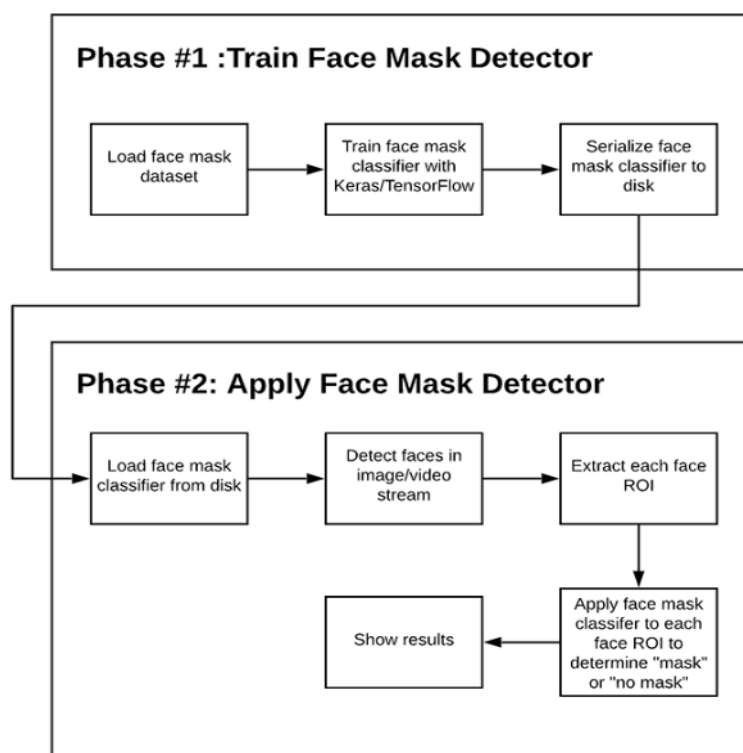


Рисунок 50 – Разбиение нейросети на этапы и подэтапы

Чтобы обучить пользовательскому детектору лицевой маски нужно разбить проект на две отдельные фазы, каждая со своими соответствующими подэтапами:

1) обучение. Здесь концентрация на загрузке набора данных обнаружения маски лица с диска, обучении модели, используя Keras и TensorFlow, а затем сериализация детектора маски лица на диск;

2) развертывание. После того, как детектор маски для лица обучен, можно перейти к загрузке детектора маски, выполнению обнаружения лица, а затем классифицировать каждое лицо как `with_mask` или `without_mask`.

Набор данных для обучения нейросети состоит из 690 изображений лиц с маской или 686 без маски. Для данной задачи была выбрана архитектуру MobileNetV2. MobileNetV2, это высокоэффективная архитектура, которая может применяться к встроенным устройствам с ограниченной вычислительной мощностью, например, Raspberry Pi, Google Coral, NVIDIA Jetson Nano и т. д. Развертывание детектора маски на встроенных устройствах может снизить стоимость производства таких систем, поэтому было решено использовать эту архитектуру.

Обучение нейронной сети реализовано в файле «`train_mask_detector.py`». На рисунке 51 представлены импорты.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import os
```

Рисунок 51 – Импорт модулей в файле `train_mask_detector.py`

Набор `tensorflow.keras` импорт позволяет:

- 1) увеличение данных;
- 2) загрузка классификатора MobileNetV2, далее модель дорабатывается с помощью предварительных обученных весов ImageNet;
- 3) строительство новой полностью подключенной головы нейросети;
- 4) предварительная обработка;

5) загрузка данных изображения.

Scikit-learn (sklearn) используется для бинаризации меток классов, сегментирования набора данных и печати отчета о классификации. Imutils поможет найти и перечислить изображения в наборе данных. Matplotlib позволит просить график обучения.

На рисунке 52 представлен программный код, который собирает данные для обучения нейросети.

```
# цикл по путям изображения
for imagePath in imagePaths:
    # извлечь метку класса из имени файла
    label = imagePath.split(os.path.sep)[-2]

    # загрузить входное изображение (224x224) и предварительно обработать его
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)

    # обновить данные и метки списки
    data.append(image)
    labels.append(label)

# преобразовать данные и метки в массивы NumPy
data = np.array(data, dtype="float32")
labels = np.array(labels)
```

Рисунок 52 – Сбор данных для обучения

Данный блок кода выполняет:

- 1) проходит все пути к изображениям;
- 2) инициализация данных, а также список меток;
- 3) этапы предварительной обработки включают изменение размера до 224×224 пикселей, преобразование в формат массива и масштабирование интенсивности пикселей во входном изображении до диапазона [-1, 1] через preprocess\_input;
- 4) добавление предварительно обработанного образа и связанной метки к данным, а также списка меток;
- 5) преобразование обучающих данных в формат массива NumPy.

Подготовка MobileNetV2 к тонкой настройке представлена на рисунке 53.

```
# загрузить сеть MobileNetV2 при отключенном заголовке FC
baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))

# построение головы модели, которая будет размещена поверх базовой модели
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# Поместить модель головы FC поверх базовой модели (она станет реальной моделью, которую мы будем тренировать)
model = Model(inputs=baseModel.input, outputs=headModel)

# перебрать все слои в базовой модели и заморозить их, чтобы они не обновлялись во время первого тренировочного процесса
for layer in baseModel.layers:
    layer.trainable = False
```

Рисунок 53 – Подготовка MobileNetV2 к тонкой настройке

Тонкая настройка состоит из трех этапов[80]:

- 1) загрузка MobileNetV2 с предварительно настроенными весами ImageNet, не прибегая к голове сети;
- 2) создание новой головы FC и добавление ее вместо старой головы;
- 3) заморозит базовые слои сети. Веса этих базовых слоев не будут обновляться в процессе обратного распространения, тогда как веса головного слоя будут настраиваться.

Компиляция, обучение и тестирование нейросети, а также сериализация модели представлено на рисунке 54.

```
# компиляция модели
print("[INFO] компиляция модели...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# обучить головы сети
print("[INFO] тренировка головы...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# прогнозы на тестовом наборе
print("[INFO] тестирование сети...")
predIdxs = model.predict(testX, batch_size=BS)

# для каждого изображения в тестовом наборе нужно найти индекс метки с соответствующей наибольшей предсказанной вероятностью
predIdxs = np.argmax(predIdxs, axis=1)

# показать отформатированный отчет о классификации
print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

# сериализовать модель на диск
print("[INFO] сохранение модели детектора маск...")
model.save(args["model"], save_format="h5")
```

Рисунок 54 – Компиляция и обучение MobileNetV2

Процесс обучения MobileNetV2 представлен на рисунке 55-56.



```
[INFO] загрузка изображений...
C:\WorkSpace\python\venv\NeuralNetworksAndComputerVision\python3.7\env\lib\site-packages\keras_applications\mobilenet_v2.py:294: UserWarning: 'input_shape' is undefined or non-square, or 'rows' is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
warnings.warn("'input_shape' is undefined or non-square, '
2020-05-17 21:28:34.823295: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
[INFO] компиляция модели...
[INFO] тренировка головы...
Train for 34 steps, validate on 276 samples
Epoch 1/20
34/34 [=====] - 101s 3s/step - loss: 0.7393 - accuracy: 0.6096 - val_loss: 0.3469 - val_accuracy: 0.8242
Epoch 2/20
34/34 [=====] - 102s 3s/step - loss: 0.4362 - accuracy: 0.7959 - val_loss: 0.1974 - val_accuracy: 0.9336
Epoch 3/20
34/34 [=====] - 110s 3s/step - loss: 0.3131 - accuracy: 0.8755 - val_loss: 0.1660 - val_accuracy: 0.9375
Epoch 4/20
34/34 [=====] - 97s 3s/step - loss: 0.2654 - accuracy: 0.8970 - val_loss: 0.0878 - val_accuracy: 0.9805
Epoch 5/20
34/34 [=====] - 96s 3s/step - loss: 0.2238 - accuracy: 0.9148 - val_loss: 0.0780 - val_accuracy: 0.9844
```

Рисунок 55 – Начало обучения MobileNetV2

```
34/34 [=====] - 89s 3s/step - loss: 0.0940 - accuracy: 0.9644 - val_loss: 0.0462 - val_accuracy: 0.9844
Epoch 18/20
34/34 [=====] - 89s 3s/step - loss: 0.0986 - accuracy: 0.9616 - val_loss: 0.0409 - val_accuracy: 0.9961
Epoch 19/20
34/34 [=====] - 91s 3s/step - loss: 0.0897 - accuracy: 0.9660 - val_loss: 0.0595 - val_accuracy: 0.9766
Epoch 20/20
34/34 [=====] - 89s 3s/step - loss: 0.0849 - accuracy: 0.9700 - val_loss: 0.0522 - val_accuracy: 0.9805
[INFO] тестирование сети...
      precision    recall  f1-score   support

with_mask      0.97      1.00      0.98       138
without_mask    1.00      0.96      0.98       138

   accuracy      0.98
  macro avg      0.98      0.98      0.98       276
weighted avg      0.98      0.98      0.98       276

[INFO] saving mask detector model...
```

Рисунок 56 – Завершение обучения MobileNetV2

Результат программы представлены на рисунках 57-59.

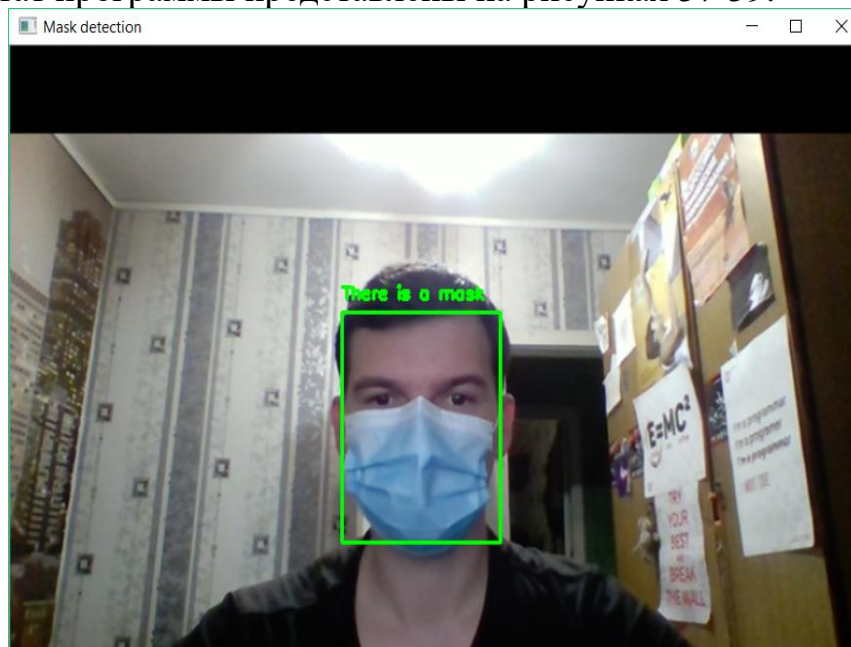


Рисунок 57 – Обнаружения наличия маски на лице



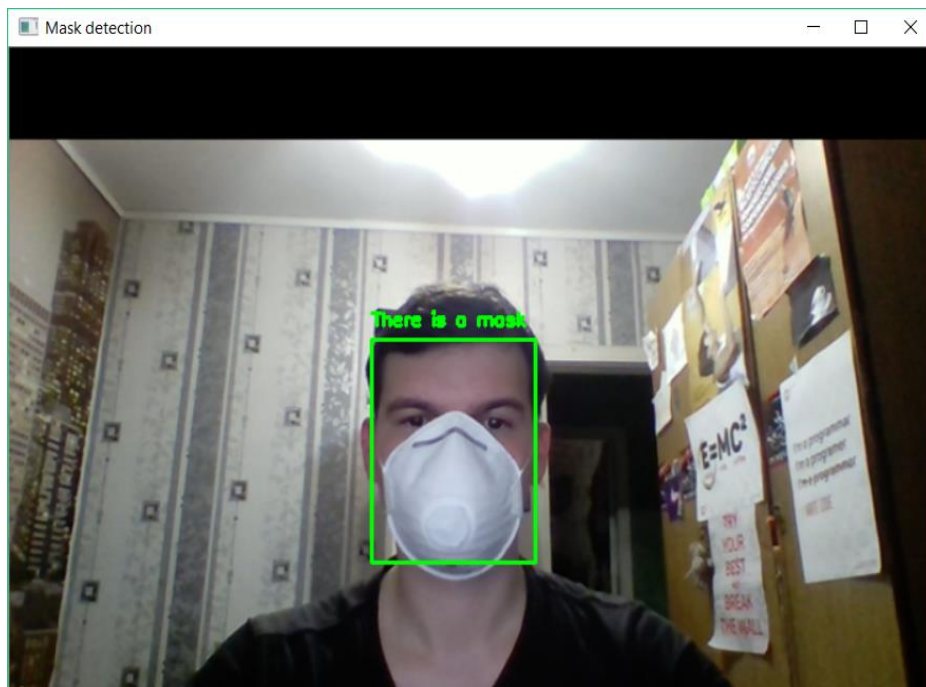


Рисунок 58 – Респиратор и т.п. присваиваться к классу «в маске»



Рисунок 59 – Обнаружения отсутствия маски на лице

## 4 Использование компьютерного зрения

Существует уязвимость при определении личности по лицу с помощью свёрточной нейронной сети, которая позволяет распознать личность по фотографии. Одно из решений этой проблемы будет использование машинного и компьютерного зрения. Для этого можно использовать библиотеку Dlib.

Детектор лицевых ориентиров, который реализован внутри dlib, выдает 68 точек, которые отображаются на конкретных областях лица. Эти точечные отображения были получены путем обучения предиктора на наборе данных iBUG 300-W. Отображение этих точек представлено на рисунке 60 [71].

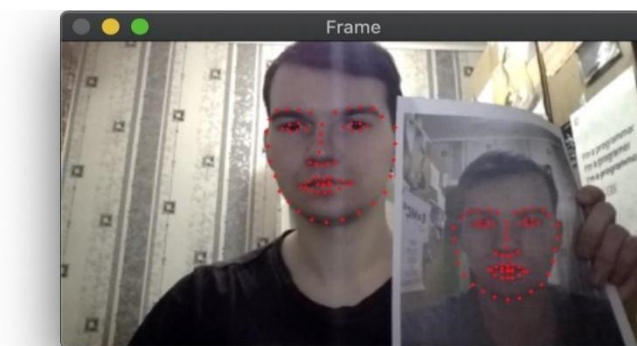


Рисунок 60 – Нахождения лица с помощью обученного детектора и наложение 68 точек

Глаза описываются шестью точками. Вычислив расстояние между ними, можно определить моргнул человек или нет: если состояние глаза стремится к нулю, то глаз закрыт, если состояние глаза больше заданного порога, то глаз открыт. Таким образом задав условие, если состояние глаза было близко к 0, то человек моргнул, следовательно перед камерой реальный человек, а не фотография.

Состояние глаза можно определить, как отношение между суммой расстояний вертикальных точек и удвоенного расстояния горизонтальных точек. Для расчета расстояний между двумя точками можно использовать формулу Евклида. На рисунке 61 представлена функция, которая производит данные вычисления.

```
def eye_condition_calculation(eye):
    # вычисление евклидова расстояния между двумя наборами
    # вертикальных координат глаз (x, y) –координаты
    first_vertical_points = dist.euclidean(eye[1], eye[5])
    second_vertical_points = dist.euclidean(eye[2], eye[4])

    # вычисление евклидова расстояния между горизонтальными
    # ориентирами глаз (x, y) –координаты
    horizontal_points = dist.euclidean(eye[0], eye[3])

    # вычислить соотношение сторон глаз
    eye_condition = (first_vertical_points + second_vertical_points) / (2.0 * horizontal_points)

    return eye_condition
```

Рисунок 61 – Программный код, отвечающий за расчет состояния глаза

Демонстрация работы программы представлена на рисунках 62 и 63.



Рисунок 62 – Если количество морганий меньше порога, то перед камерой возможно не человек

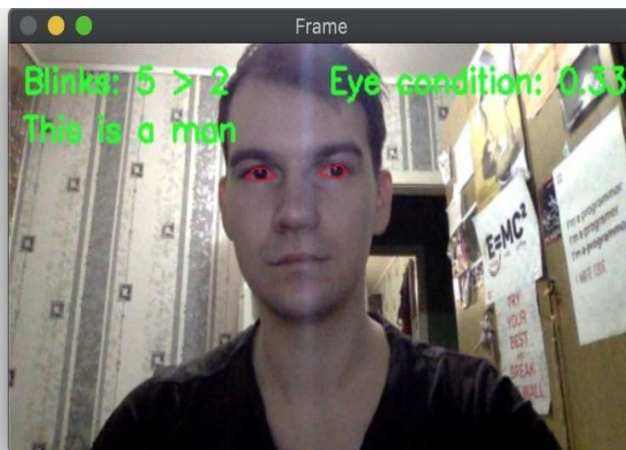


Рисунок 63 – Если количество морганий больше порога, то перед камерой человек

## 5 Экономическая часть

В дипломной работе была решена проблема информационной безопасности на физическом уровне, а именно обнаружение физического доступа к данным третьих лиц. Результатом решением проблемы было создание программного обеспечения. программное обеспечение должно распознавать зарегистрированных личностей (например сотрудников предприятия) или фиксировать вторжения третьих лиц.

В данной главе будут подсчитаны затраты, которые составили при исследовании сверточных нейронных сетей, их тестирование и создании программного обеспечения, по результатам которых определится себестоимость продукта.

Чтобы определить базовую стоимость нужно принять во внимание следующие пункты:

- 1) накладные расходы;
- 2) амортизационные отчисления, которые потребуются на восстановление технических средств и компьютерных программ;
- 3) расходы на бытовые нужды (электроэнергия), которые необходимы техническим устройствам;
- 4) трудоемкость создания программного обеспечения;
- 5) стоимость оплаты программиста.

В научно-исследовательской работе с сверточными нейронными сетями, их тестирование и в текущей разработке программного обеспечения применялись следующие технические устройства:

- 1) 15.6" Ноутбук ASUS TUF Gaming FX505DT-AL023 черный, 1 шт [74];
- 2) 22" (55 см) Телевизор LED LG 22MT49VF-PZ черный, 1 шт [79];
- 3) компактная мышь беспроводная Logitech M171 красный, 1 шт [78];
- 4) 250 ГБ SSD M.2 накопитель Samsung 970 EVO [MZ-V7E250BW], 1 шт [76];

5) оперативная память SODIMM AData [AD4S266638G19-S] 8 ГБ, 1 шт [77];

6) веб-камера Logitech HD C525, 1 шт [73].

Стоимость использованного оборудования:

1) ноутбук – 60499 рублей;

2) телевизор – 9999 рублей;

3) компьютерная мышь – 999 рублей;

4) SSD M.2 накопитель – 5799 рублей;

5) SO-DIMM оперативная память – 2699 рублей;

6) веб-камера – 4899 рублей.

Общая цена технических устройств, руб

$$Ц_{тс} = Ц_{н} + Ц_{т} + Ц_{км} + Ц_{вн} + Ц_{оп} + Ц_{вк}, \quad (15)$$

где  $Ц_{н}$  – цена ноутбука;

$Ц_{т}$  – цена телевизора;

$Ц_{км}$  – цена компьютерной мыши;

$Ц_{вн}$  – цена внутреннего накопителя;

$Ц_{оп}$  – цена оперативной памяти;

$Ц_{вк}$  – цена веб-камеры.

Общая цена технических устройств равна, руб

$$Ц_{тс} = 60499 + 9999 + 999 + 5799 + 2699 + 4899 = 84894.$$

Для создания программного обеспечения, являющегося конечным результатом исследований, применялось следующее программное обеспечение:

1) Windows 10 Pro – 289 USD на одного пользователя;

2) PyCharm – 199 USD за год.

Общая цена программного обеспечения, руб

$$Ц_{по} = (Ц_w + Ц_p) \times Ц_{usd}, \quad (16)$$

где  $Ц_w$  – Windows;

$Ц_p$  – IDE PyCharm;

$Ц_{usd}$  – Курс USD/RUB на 17 июля 2019 года.

Общая стоимость программного обеспечения равна, руб

$$Ц_{тс} = (289 + 199) \times 62,94 = 30714,72.$$

Таким образом общая стоимость технических средств и программного обеспечения составит, руб

$$Ц_о = Ц_{тс} + Ц_{по}. \quad (17)$$

Отсюда, руб

$$Ц_о = 84894 + 30714,72 = 115608,72.$$

Для установки нормы амортизационных расчетов нужно определить срок использования технических средств при исследовании, тестирование и разработке. Срок эксплуатации в месяц равняется 28-31 дням(ю), возьмем 31 день. С помощью линейного способа, можно найти норму амортизационных отчислений по формуле

$$H_a = \frac{1}{n} \times 100\%, \quad (18)$$

где  $n$  – это установленный срок в месяц.

Следовательно норма равна

$$H_a = \frac{1}{31} \times 100\% = 3,25\%$$

Ежемесячные амортизационные отчисления равняются 3,25 %. Срок исследование нейросетей, их тестирование и создания программного обеспечения составляет 91 дней, то размер амортизационных отчислений составили 22,85 %. Ежегодная амортизационные отчисления составляет 37,74%.

Амортизационные отчисления на полное восстановление технических средств и программного обеспечения в год рассчитываются по формуле

$$A_o = Ц_о + H_a, \quad (19)$$

Отсюда

$$A_o = 115608,72 \cdot 0,22 = 25433,92.$$

Амортизационные начисления за время исследования, тестирования и разработки программы

$$A_{\text{п}} = \frac{A_0 \times K_{\text{дн}}}{K_{\text{рг}}}, \quad (20)$$

где  $K_{\text{дн}}$  – количество отработанных дней;

$K_{\text{рг}}$  – количество рабочих дней в году.

Количество затраченных дней составляют 91, количество рабочих дней в году составляют 247.

Отсюда

$$A_{\text{п}} = \frac{25433,92 \times 91}{247} = 9370,39.$$

Далее будет приведен расчет расходов на энергопотребление.

ПЭВМ, на которой были проведены исследования, тестирования и разработка программного обеспечения, является потребителем электрической энергии сети переменного тока, напряжением 220 В. Согласно технической документации, потребляемая мощность ноутбука составляет 96 Вт, а телевизора 26 Вт. суммарная потребляемая мощность, составляет, Вт · ч

$$M_{\text{с}} = 122.$$

Затраты денежных средств, которые связанные с энергопотреблением технических средств можно вычислить по формуле

$$P_{\text{э}} = K_{\text{дн}} \times V_{\text{раб}} \times M_{\text{с}} \times \text{Ццэ}, \quad (21)$$

где  $K_{\text{дн}}$  – период исследования нейросетей, их тестирования и создания программного обеспечения;

$V_{\text{раб}}$  – затраченное время в рабочей смены, ч;

$M_{\text{с}}$  – мощность, потребляемая техническими средствами, кВт · ч;

$\text{Ццэ}$  – стоимость электроэнергии в первое полугодие, р./кВт · ч [52].

Отсюда затраты денежных средств, которые связанные с энергопотреблением технических средств в рублях равняется:

$$P_{\text{э}} = 91 \cdot 5 \times 0,12 \cdot 3.81 = 208,03$$

Далее найдем количество фактически отработанного времени в часах

$$T_{\text{ф}} = K_{\text{дн}} \cdot V_{\text{раб}}, \quad (22)$$

$$T\phi = 91 \cdot 5 = 455.$$

Часовая заработная плата программиста составляет 300 рублей в час, тогда основная зарплата будет определяться по формуле, руб

$$З_{осн} = T\phi \cdot T_{ч}, \quad (23)$$

где  $T_{ч}$  – ставка за час программиста.

$$З_{осн} = 455 \times 300 = 136500$$

Для определения всей суммы расходов на оплату труда, нужно учесть доплаты и надбавки. Установим вид доплат и надбавок в размере 20% от заработной платы, руб

$$У_{доп} = З_{осн} \times 0,2, \quad (24)$$

$$У_{доп} = 136500 \times 0,2 = 27300$$

Отсюда найдем всю сумму расходов на оплату труд, руб

$$Р_{общ} = З_{осн} + У_{доп}, \quad (25)$$

$$Р_{общ} = 136500 + 27300 = 163800$$

Определим единый социальный налог. Общая ставка взносов в 2020 году составляет 30 % от расходов на оплату труда, руб

$$О_{соц} = Р_{общ} \times 0,3, \quad (26)$$

$$О_{соц} = 163800 \times 0,3 = 49140$$

Вычисление всех затрат в рублях составляет

$$T = A_{п} + P_{э} + Р_{общ} + О_{соц}, \quad (27)$$

$$T = 9370,39 + 208,03 + 163800 + 49140 = 222518,42$$

Исходя из всех рассчитанных затрат можно установить цену на программное обеспечение 50000 рублей. Данная сумма не может быть конечной так как не учтены затраты на регистрацию, рекламную кампанию, продажи, зарплаты сотрудников и др.

Экономическая эффективность программного обеспечения достигается за счет автоматизации работы по наблюдению за охраняемыми объектами. Заработная плата охранника составляет в среднем 30500 рублей в месяц. Годовая экономия по заработной плате с учетом начислений вычисляется по формуле



$$\text{ЭЗП} = \text{ЭЗП.мес.} \cdot (1 + \text{Котч}) \cdot n, \quad (28)$$

где  $\text{ЭЗП}$  – годовая экономия по заработной плате с отчислениями на социальные мероприятия;

$\text{ЭЗП.мес.}$  – месячная экономия заработной платы при внедрении программного обеспечения;

$\text{Котч}$  – коэффициент, учитывающий отчисления на социальное страхование и в другие фонды;

$n$  – количество месяцев.

Если взять за пример текущую стоимость программного обеспечения, то годовая экономия по заработной плате с учетом начислений составляет, руб

$$\text{ЭЗП} = 30500 \cdot (1 + 0,3) \times 12 = 475800$$

Для предприятия использование созданного программного обеспечения позволит:

- 1) сэкономить затраты путем отказа от увелечения допалнительных сотрудников охраны;
- 2) автоматизировать физический доступ сотрудникам;
- 3) повысит уровень информационной безопасности.

Окупаемость в предприятии наступит через 2 месяца, если не будет требоваться дополнительного оборудования, например, камер, сервера, замков и т.п.

Окупаемость проделанной работы при текущей стоимости программного обеспечения наступит после 5 продаж.

## **6 Безопасность и экологичность**

Исследование сверточных нейронных сетей, их тестирование, создание программного обеспечения, последующее обслуживание и эксплуатация программного обеспечения, такую работу, можно квалифицировать как творческую, которая связана с персональными электронными вычислительными машинами и другими устройствами.

Сотрудники непосредственно выполняют свою работу за ЭВМ, а в связи с этим, на них влияют вредные факторы, которые уменьшают производительность труда. Данные факторы:

- 1) негативное воздействие вредных излучений от монитора;
- 2) не нормированный уровень шума;
- 3) неправильная освещенность;
- 4) наличие напряжения;
- 5) нарушение микроклимата, и другие факторы.

### **6.1 Описание рабочего места программиста**

Рабочее место – это место, где работник осуществляет свою трудовую деятельность, и проводит значительное количество рабочего времени. Рабочее место, должно быть приспособленное к работнику, правильно. Разумно организованное рабочее место, по отношению формы, пространства и размера обеспечивает работнику благоприятное положение. Так же правильно организованное рабочее место обеспечивает высокую производительность труда при наименьшем психическом и физическом напряжении.

Правильно организованное рабочее место увеличивает производительность труда 8-20%.

В соответствие с физическими, психологическими и антропометрическими требованиями. Характер работы имеет большое значение. Также во время организации рабочего места программиста нужно соблюсти основные условия:

- 1) должно быть обеспечено достаточное рабочее пространство, которое будет позволять осуществлять все необходимые движения и перемещения;
- 2) оборудования, которое входит в состав рабочего места, должно быть оптимально размещено;
- 3) уровень шума не должен быть выше допустимого значения;
- 4) для выполнения поставленных задач, должно быть естественное и искусственное освещение.

Основные элементы рабочего места программиста, следующие:

- 1) письменный стол;
- 2) кресло.

Основным положением для работы является положение сидя. Рабочее место в положении сидя во время выполнения работ, должно организовываться в соответствии с ГОСТ 12.2.032-78.

Во время работы в положении сидя рекомендуются следующие параметры рабочего пространства:

- 1) ширина не меньше 70 см;
- 2) высота поверхности стола над полом 70-75 см;
- 3) глубина не меньше 40 см.

Оптимальные размеры стола являются следующие:

- 1) длина – 130 см;
- 2) ширина – 65 см;
- 3) высота – 71 см.

Поверхность для письма должна быть не меньше:

- 1) 60 см в ширину;
- 2) 4 см в глубину.

Под рабочей поверхностью должно быть место для ног, которое характеризуется следующими параметрами:

- 1) ширина не меньше 50 см;
- 2) высота не меньше 60 см;
- 3) глубина не меньше 40 см.

Кресло программиста должно быть выполнено в соответствии с требованиями ГОСТ 21.889-76. Кресло проектируют таким образом, чтобы поза программиста в любом рабочем положении была физиологически правильно обоснованной, исходя из этого положение частей тела должно быть оптимальным. Для удовлетворения таким требованиям конструкция рабочего сидения должна соответствовать следующим требованиям:

- 1) должна быть слегка с вогнутой поверхностью;
- 2) должна быть возможность регулирования высоты;
- 3) должна быть возможность изменять положения тела;
- 4) иметь небольшой наклон назад.

Далее исходя из вышесказанного, определим параметры стола программиста:

- 1) длина – 130 см;
- 2) ширина – 65 см;
- 3) высота – 71 см;
- 4) глубина – 40 см.

Поверхность для письма:

- 1) в ширину 60 см;
- 2) в глубину 4 см.

Для облегчения труда работников должны быть созданы благоприятные условия и правильное эстетическое оформление, которые имеют большое значение не только для облегчения труда, но и для повышения его производительности труда. Окраска помещений и мебели должны создавать благоприятные условия для зрительного восприятия и хорошего настроения. Служебные помещения,

в которых выполняется однообразная умственная работа, которая требует значительного нервного напряжения и большого сосредоточения, окраска такого помещения должна быть спокойных тонов – малонасыщенные оттенки холодного голубого или зеленого цветов.

## **6.2 Освещенность**

Искусственное освещение в помещениях, где используется мониторы и ЭВМ должно реализовываться системой равномерного освещения. В административно-общественных и производственных помещениях, при работе с документами, возможно комбинированное освещение.

На поверхности стола в зоне, где размещаются рабочие документы, освещенность должна составлять от 300 до 500 лк. Также могут быть установлены светильники местного освещения для подсветки рабочих документов. Такое освещение не должно создавать бликов на поверхности экрана и увеличивать освещенность экрана более 300 лк.

Необходимо ограничивать блесккость отраженную на рабочих поверхностях таких как стол, экран, клавиатура и др. с помощью правильного размещения рабочих мест по отношению источников света, а также необходимо производить правильный выбор типа светильников.

Яркость между рабочими поверхностями и поверхностями стен и оборудования не должна быть больше, чем 10:1, между рабочими поверхностями не должно быть больше, чем 3:1-5:1.

При искусственном освещении, в качестве источников света должны быть применены люминесцентные лампы типа ЛБ. При устройстве отраженного освещения административно-общественных и производственных помещениях, могут быть установлены металлогалогенных ламп мощностью до 250 Вт. Или могут быть установлены лампы накаливания в светильниках местного освещения.

Светильники должны быть выполнены в виде прерывистых или сплошных линий и находиться сбоку от рабочих мест параллельно по отношению к линии зрения пользователя при разном положении ЭВМ. При расположении таком как периметральное – линии светильников должны быть распределены над рабочим столом ближе к переднему краю стола, и быть обращенными к программисту.

Для того чтобы достичь нормируемые значения освещенности в помещениях, в которых используются мониторы и ЭВМ, необходимо проводить чистку светильников и стекол не меньше, чем два раза в год, и также следует своевременно проводить замену перегоревших ламп.

Далее будет приведен расчет искусственного освещения для рабочего места.

Характеристики помещения, следующие:

- 1) высота помещения равняется 2,7 м;
- 2) ширина помещения равняется 3 метрам;
- 3) длина помещения равняется 6 метрам;
- 4) количество рабочих мест одно;
- 5) количество окон одно;
- 6) присутствует искусственное освещение и естественное освещение.

Поскольку работа программиста не относится к категории особо точных, то выберем систему общего освещения. Нормативная величина показателя ослепленности составляют 20%, уровень пульсаций освещенности составляет 10%.

Вычисление системы общего освещения производится методом коэффициента использования светового потока, который выражается отношением светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп. Эта величина зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемой коэффициентами отражения стен и потолка.

Необходимый световой поток лампы в каждом светильнике вычисляется по формуле, Лм

$$F_0 = \frac{E \cdot K \cdot S \cdot Z}{\eta}, \quad (29)$$

где  $E$  – заданная минимальная освещенность, лк (500);

$K$  – коэффициент запаса, учитывающий уменьшение светового потока лампы в результате загрязнения светильников в процессе эксплуатации (для люминесцентных ламп 1,5);

$S$  – освещаемая площадь, м<sup>2</sup> (18);

$Z$  – отношение средней освещенности к минимальной (обычно принимается равным 1.1-1.2, для люминесцентных ламп – 1,1);

$\eta$  – коэффициент использования светового потока в долях единицы (отношение светового потока, падающего на расчетную поверхность, к суммарному потоку всех ламп).

Формула расчета высоты светильника над рабочей поверхностью имеет вид, м

$$h = H - h_p - h_c \quad (30)$$

где  $H$  – высота помещения, м;

$h_p$  – высота рабочей поверхности от пола, м;

$h_c$  – высота свеса светильника от основного потолка, м.

Тогда

$$h = 2,7 - 0,7 - 0 = 2$$

Коэффициент использования  $\eta$  зависит от типа светильника, от коэффициентов отражения потолка  $\rho_{\text{п}}$ , стен  $\rho_{\text{с}}$ , расчетной поверхности  $\rho_{\text{р}}$ , индекса помещения

$$i = \frac{S}{h \cdot (a + b)}, \quad (31)$$

где  $a$  – длина помещения;

$b$  – ширина помещения.

Тогда

$$i = \frac{18}{2 \times (6 + 3)} = 0,5.$$

Так как помещение является административно-офисным, то для светлого фона примем, что  $\rho_{\text{п}} = 70$ ,  $\rho_{\text{с}} = 50$ ,  $\rho_{\text{р}} = 10$ , следовательно  $\eta = 24\%$ . Световой поток одной лампы рассчитывается по формуле, Лм

$$F_{\text{л}} = \frac{F_0}{N}, \quad (32)$$

где  $N$  – число ламп.

Тогда

$$F_0 = \frac{500 \times 1,5 \times 18 \times 1,1}{0,24} = 61875.$$

Далее рассчитаем количество ламп, шт

$$N = \frac{61875}{4300} = 14,3.$$

Следовательно, нужно использовать 15 ламп.

Для освещения выбираем люминесцентные лампы типа ЛТБ 80-4, световой поток которых 4300 Лм.

Количество светильников выбирается в зависимости от размеров освещаемого помещения, при этом количество светильников должно быть таким, чтобы отношение расстояния между ними к высоте их подвеса над поверхностью было равно  $1,5 \div 2$ .

Для осветительных приборов используются светильники типа ЛПО 1/20. Каждый светильник комплектуется одной лампой. Размещаются светильники тремя рядами, по два в каждом ряду.

Допускается отклонение ( $\varepsilon$ ) светового потока выбранной лампы от расчетного от -10 % до 20 %, лк

$$E_{\text{факт}} = \frac{\Phi_{\text{л}} \cdot N \cdot N_{\text{лсв}} \cdot n}{S \cdot z \cdot k}, \quad (33)$$

$$E_{\text{факт}} = \frac{4300 \cdot 15 \cdot 1 \cdot 0,24}{18 \cdot 1,1 \cdot 1,5} = 521.$$

Отличие от нормированного уровня определяется по формуле

$$\frac{E_{\text{факт}} - E_{\text{норм}}}{E_{\text{норм}}} \times 100\%, \quad (34)$$



$$\frac{521 - 500}{500} \times 100\% = 4\%.$$

Для исключения засветки экранов дисплеев прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения программиста и стене с окнами. Данное размещение светильников позволяет производить последовательное включение в зависимости от величины естественной освещенности и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

### **6.3 Шум**

Уровень шума на рабочих местах при работе с ЭВМ не должен превышать значений, которые установлены в СанПиН 2.2.4/2.1.8.562-96 и быть не более 50 дБА. На рабочих местах в помещениях, где размещаются шумные агрегаты, то уровень шума не должен превышать 75 дБА, а уровень вибрации в помещениях не должен быть выше таких значений по СН 2.2.4/2.1.8.566-96 категория 3, тип «в».

С помощью использования звукопоглощающих материалов коэффициентом звукопоглощения в области частот 63-8000 Гц для отделки стен и потолка помещений, уровень шума можно снизить, если использовать однотонные шторы из плотной ткани, которая повешена на расстоянии 15-20 см от ограждения, также создает звукопоглощающий эффект. Но при этом ширина штор должна быть в 2 раза больше, чем ширина окна.

### **6.4 Электробезопасность**

Ноутбук – это электрическое устройство напряжение питания, которого составляет 220.

Для того чтобы не получить поражения электрическим током, возникновения пожара и повреждения ноутбука нужно выполнять следующие меры безопасности:

- 1) запрещается использование ноутбука при неисправном шнуре питания;
- 2) запрещается включать ноутбук и периферийные устройства со снятой крышкой;
- 3) запрещается подключать к ноутбуку периферийные устройства с включенным питанием;
- 4) при эксплуатации ноутбука следует принять меры, которые исключат удары или падения ноутбука;
- 5) запрещается эксплуатация ноутбука в помещении с высокой влажностью или сильно загрязненным воздухом;
- 6) запрещается оставлять без присмотра работающий ноутбук;
- 7) запрещается устанавливать ноутбук рядом с источником тепла;
- 8) запрещается перегибы, натяжения и передавливания кабелей питания;
- 9) запрещается закрывать вентиляционные отверстия ноутбука и периферийных устройств.

Для того чтобы обеспечить электробезопасность при установке, наладке и работе нужно принять во внимание создание защитных мер во избежание попадания пользователей и обслуживающего персонала под напряжение.

Электропитание рабочего места должно быть выполнено при помощи рубильника, который в свою очередь установленный в месте, так чтобы можно было быстро выключать питание рабочего места, в случае аварийных ситуаций.

## **6.5 Возможные аварийные и чрезвычайные ситуации**

Одно из наиболее возможных чрезвычайных ситуаций – возможность возгорания. Пожар – это неконтролируемое горение вне контролируемой зоны, влекущий за собой материальный ущерб. Его можно охарактеризовать:

- 1) нанесением урона зданиям, сооружениям и установкам;
- 2) появлением вредных продуктов горения и дыма;
- 3) появление открытого огня и искр;
- 4) уменьшенной концентрацией кислорода в воздухе;
- 5) увеличение температуры воздуха.

Для уменьшения пожарной опасности на рабочем месте программиста разработаны и используются определенные методы и устройства защиты от пожара. Используются активные и пассивные средства огнезащиты. На рабочем месте программиста находятся порошковый огнетушитель, в помещении имеется пожарная сигнализация. Также имеется система автоматического пожаротушения.

В здании, где находится рабочее место программиста используются малотоксичные строительные материалы.

В организации применяется набор мер по электробезопасности (заземление, зануление, молниезащита). Используемый набор мер по обеспечению пожарной безопасности является полным.

## **6.6 Охрана окружающей среды**

При исследовании сверточных нейронных сетей, их тестирование и создании программного обеспечения негативных воздействий на экологичность окружающей среды не осуществляется. Эксплуатация программного обеспечения также не влияет негативно на окружающую среду.

## **6.7 Вывод по разделу**

При написании данного раздела были подробно раскрыты вопросы экологичности и безопасности.

Для того что бы не быть пораженным электрическим током при замыкании на корпус ЭВМ следует устанавливать нулевой защитный проводник;

На рабочем месте программиста нужно обеспечить защиту от статического электричества. Также был проведен анализ освещенности рабочих мест.

Произведен осмотр экологичности помещения. Был произведен анализ воздействия электромагнитных излучений на программиста. Однозначно, что на расстоянии в 30 см интенсивность электромагнитного излучения, не превышает 2,4 В/м, что ниже, чем допустимый уровень, в связи с этим вредное воздействие на человека исключено.

В итоге можно сделать вывод, что рассматриваемое помещение удовлетворяет всем требованиям безопасности и экологичности.

## Заключение

Современные научные достижения в таких областях информатики как математическое моделирование состояния внешнего мира, искусственный интеллект, обработка изображений, сигналов и сцен распознавание образов, оптимальное управление, и др. позволяют говорить о реальной возможности перехода к новому поколению средств информационной защиты – интеллектуальным системам информационной безопасности. Стремительно растет и перечень задач информационной безопасности, решаемых с использованием интеллектуальных методов и средств. Одна из первых актуальных задач в сфере информационной безопасности, потребовавшей использования мощного арсенала методов и средств искусственного интеллекта, стала задача обнаружения вторжений и атак на автоматизированные информационные системы.

Задача разработки новых методик защиты информации является весьма актуальной областью деятельности, поскольку развитие и внедрение информационных технологий проходят параллельно с совершенствованием методов несанкционированного доступа к информации. При этом особое внимание следует обратить на интеллектуальные методы, например, искусственные нейронные сети.

Для решения этой задачи в данной работе были приведены исследования нейросетей, тестирования сверточных нейронных сетей. И создано программное обеспечение Personality Recognition.

## Список использованных источников

- 1 А.В. Васильков, И.А. Васильков. Безопасность и управление доступом в информационных системах. – М. : Форум, 2010. – 368 с.
- 2 Анна. Kornia: библиотека для компьютерного зрения на PyTorch // Neurohive : интернет портал. 2020. 16 янв. URL: <https://cutt.ly/4ua213B> (дата обращения: 22.03.2020).
- 3 Банковские информационные системы и технологии. Часть 1. Технологии банковского учета. – М. : Финансы и статистика, 2010. – 384 с.
- 4 Баранова, Е. Информационная безопасность и защита / Е. Баранова, А. Бабаш – М. : РИОР, 2017 – 324 с.
- 5 Белиовский, Н. А. Основы машинного зрения в среде LabVIEW / Н. А. Белиовский, Л. Г. Белиовский. 2017. – 88 с.
- 6 Бил, Д. Snort 2.1. Обнаружение вторжений / Джей Бил. – М. : Бином-Пресс, 2012. – 656 с.
- 7 Бирюков И. Разнообразие нейронных сетей. Часть вторая. Продвину-тые конфигурации // Tproger : интернет портал. 2016. 13 окт. URL: <https://tproger.ru/translations/neural-network-zoo-2/> (дата обращения: 22.03.2020).
- 8 Бирюков И. Шпаргалка по разновидностям нейронных сетей. Часть первая. Элементарные конфигурации // Tproger : интернет портал. 2016. 04 окт. URL: <https://cutt.ly/Vua226B> (дата обращения: 22.03.2020).
- 9 Боммана Х. Loss Functions Explained // Medium : интернет портал. 2019. 30 сен. URL: <https://cutt.ly/tua23DT> (дата обращения: 22.03.2020).
- 10 Бузов, Г. А. Защита от утечки информации по техническим каналам: Учебное пособие / Г. А. Бузов, С. В. Калинин, А. В. Кондратьев. – М. : Горячая линия-Телеком, 2005. – 416 с.
- 11 Васильков, А. В. Безопасность и управление доступом в информацион-ных системах / А.В. Васильков, И.А. Васильков. – М. : Форум, 2010. – 368 с.

12 Вебер Б. Custom Loss functions for Deep Learning: Predicting Home Values with Keras for R // towards data science : интернет портал. 2018. 13 май. URL: <https://cutt.ly/eua2481> (дата обращения: 22.03.2020).

13 Величко, В. В. Модели и методы повышения живучести современных систем связи: моногр. / В.В. Величко, Г.В. Попков, В.К. Попков. – М. : Горячая линия – Телеком, 2014. – 272 с.

14 Гейтджи А. Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning // Medium : интернет портал. 2016. 25 июл. URL: <https://cutt.ly/Dua9qbb> (дата обращения: 22.03.2020).

15 Гейтджи А. Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning // Medium : интернет портал. 2016. 25 июл. URL: <https://cutt.ly/aua9epf> (дата обращения: 22.03.2020).

16 ГОСТ Р 50922-2006 Защита информации. Основные термины и определения. – Введ. 2006. – М. : Федеральное агентство по техническому регулированию и метрологии: Национальный стандарт РФ, 2006. – 12 с.

17 ГОСТ Р 52069.0-2003. Защита информации. Система стандартов. Основные положения. – Введ. 2003. – М. : Государственный научно-исследовательский испытательный институт проблем технической защиты информации: Государственный стандарт РФ, 2003.

18 Джа П. A Brief Overview of Loss Functions in Pytorch // Medium : интернет портал. 2019. 07 янв. URL: <https://cutt.ly/Iua9tiC> (дата обращения: 22.03.2020).

19 Елези И. The Group Loss for Deep Metric Learning // Cornell University : интернет портал. 2020. 19 мар. URL: <https://cutt.ly/Tua9y3z> (дата обращения: 22.03.2020).

20 Избачков, Ю. Информационные системы / Ю. Избачков, В. Петров. – М. : Питер, 2012. – 656 с.

21 Информационная система для предприятия розничной торговли мобильными телефонами и аксессуарами // АСИНХРОННЫЕ ДВИГАТЕЛИ ДУЮНОВА : интернат портал. 2020. URL: <https://werr.ru/diplom1/economik.php> (дата обращения: 22.03.2020).

22 Информационные системы : учебное пособие для вузов / Ю. С. Избачков, В. Н. Петров, А. А. Васильев, И. С. Телина. – 3-е изд. – СПб. : Питер, 2011. – 539 с.

23 Исключение (нейронные сети) // Википедия свободная энциклопедия : сетевая энциклопедия. 2020. 15 апр. URL: <https://cutt.ly/xua9d05> (дата обращения: 22.03.2020).

24 Кизрак А. Comparison of Activation Functions for Deep Neural Networks // towards data science : интернет портал. 2017. 09 май. URL: <https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a> (дата обращения: 22.03.2020).

25 Когнитрон // википедия свободная энциклопедия : сетевая энциклопедия. 2019. 2 авг. URL: <https://cutt.ly/6ua9hvu> (дата обращения: 22.03.2020).

26 Криницкий, Н.А. Автоматизированные информационные системы / Н.А. Криницкий, Г.А. Миронов, Г.Д. Фролов. – М. : Наука, 2011. – 382 с.

27 Кулурчелло Э. Neural Network Architectures // towards data science : интернет портал. 2017. 24 мар. URL: <https://cutt.ly/Oua9klM> (дата обращения: 22.03.2020).

28 Куприянов, А.В. Основы защиты информации / А. В. Куприянов. – М. : Академия, 2008. – 256 с.

29 Ле Д. The 10 Neural Network Architectures Machine Learning Researchers Need To Learn // Medium : интернет портал. 2018. 02 авг. URL: <https://cutt.ly/uua9zZI> (дата обращения: 22.03.2020).

30 Макаров, И. М. Искусственный интеллект и интеллектуальные системы управления / И. М. Макаров. – М. : Наука, 2017. – 336 с.



- 31 Малюк, А. А. Защита информации в информационном обществе / А.А. Малюк. – М. : Горячая линия-Телеком, 2015. – 229 с.
- 32 Махендру К. A Detailed Guide to 7 Loss Functions for Machine Learning Algorithms // Medium : интернет портал. 2019. 14 авг. URL: <https://cutt.ly/cua9cg0> (дата обращения: 22.03.2020).
- 33 Мезенцев, К. Н. Автоматизированные информационные системы / К.Н. Мезенцев. – М. : Академия, 2010. – 176 с.
- 34 Неокогнитрон // википедия свободная энциклопедия : сетевая энциклопедия. 2019. 2 авг. URL: <https://cutt.ly/nua9v49> (дата обращения: 22.03.2020).
- 35 О.В. Казарин. Методология защиты программного обеспечения. Научные проблемы безопасности и противодействия терроризму. – М. : МЦНМО, 2009. – 464 с.
- 36 Об информации, информационных технологиях и о защите информации :федер. закон от 27 июля 2006 г. № 149-ФЗ // Российская газета. – 2006. – 29 июля.
- 37 Об информации, информационных технологиях и о защите информации: [Электронный ресурс] : федер. закон от 27 июля 2007 г. № 149-ФЗ. Доступ из справ.-правовой системы «КонсультантПлюс».
- 38 Об особенностях оценки соответствия продукции (работ, услуг), используемой в целях защиты сведений, относимых к охраняемой в соответствии с законодательством РФ информации ограниченного доступа, не содержащей сведения, составляющие государственную тайну, а также процессов ее проектирования (включая изыскания), производства, строительства, монтажа, наладки, эксплуатации, хранения, перевозки, реализации, утилизации и захоронения, об особенностях аккредитации органов по сертификации и испытательных лабораторий (центров), выполняющих работы по подтверждению соответствия указанной продукции (работ, услуг) : постановление Правительства РФ от 15 мая 2010 г. № 330 // Собрание законодательства РФ. – 2010. – 15 мая.

39 Об утверждении Доктрины информационной безопасности Российской Федерации: указ Президента РФ от 05 декабря 2016 г. № 646 // Указ Президента РФ «Об утверждении Доктрины информационной безопасности Российской Федерации». – 2016.

40 Об утверждении перечня мер, направленных на обеспечение выполнения обязанностей, предусмотренных Федеральным законом «О персональных данных» и принятыми в соответствии с ним нормативными правовыми актами, операторами, являющимися государственными или муниципальными органами» : постановление Правительства РФ от 21 марта 2012 г. № 211 // Собрание законодательства РФ. – 2012. – 21 марта.

41 Об утверждении Положения о порядке обращения со служебной информацией ограниченного доступа в федеральных органах государственной власти : постановление Правительства РФ от 03 нояб. 1994 г. № 1233 // Собрание законодательства РФ. – 1994. – 03 ноября.

42 Павленко А. Библиотеки Python для нейронных сетей // OTUS ОНЛАЙН-ОБРАЗОВАНИЕ : интернет портал. 2019. 20 май. URL: <https://cutt.ly/Oua9mmz> (дата обращения: 22.03.2020).

43 Паджанкар, А. Raspberry Pi for Computer Vision / А. Паджанкар. – Manning Publications, 2015 – 178 с.

44 Петренко, С.А. Политики информационной безопасности / С.А. Петренко. – М. : Книга по Требованию, 2010. – 400 с.

45 Путин назначил ответственных за искусственный интеллект в России // CNEWS : интернет портал. 2019. 14 июн. URL: <https://cutt.ly/fua9Wsw> (дата обращения: 22.03.2020).

46 Путькина, Л. В. Интеллектуальные информационные системы / Л.В. Путькина, Т.Г. Пискунова. – М. : СПбГУП, 2011. – 228 с.

47 Разработка программного продукта // Библиофонд Электронная библиотека студента : сетевая энциклопедия. 2011. 22 июн. URL: <https://www.bibliofond.ru/view.aspx?id=542168> (дата обращения: 22.03.2020).

48 Робастность // википедия свободная энциклопедия : интернет портал. 2020. 02 мая. URL: <https://cutt.ly/Iua9TYC> (дата обращения: 22.03.2020).

49 Ронаган С. Deep Learning: Overview of Neurons and Activation Functions // Medium : интернет портал. 2018. 27 июл. URL: <https://cutt.ly/Bua9Uxm> (дата обращения: 22.03.2020).

50 Росеброк, А. Practical Python and OpenCV / Росеброк А. – PyImageSearch, 2015. – 418 с.

51 РусГидро [Электронный ресурс] / Web-мастер Корняев В.В. – Электрон. дан. – М. : Публичное акционерное общество «Дальневосточная энергетическая компания», 2018 – Режим доступа: <https://cutt.ly/5ua9Prg>, свободный. – Загл. с экрана. Яз. рус., англ.

52 РусГидро [Электронный ресурс] / Web-мастер Корняев В.В. – Электрон. дан. – М. : Публичное акционерное общество «Дальневосточная энергетическая компания», 2020 – Режим доступа: <https://cutt.ly/Mua9Shy>, свободный. – Загл. с экрана. Яз. рус., англ.

53 Нейронная сеть // Википедия свободная энциклопедия : сетевая энциклопедия. 2020. 22 апр. URL: <https://cutt.ly/sua9FtV> (дата обращения: 22.03.2020).

54 Сейф Д. Understanding the 3 most common loss functions for Machine Learning Regression // towards data science : интернет портал. 2019. 21 май. URL: <https://cutt.ly/Tua9G6Q> (дата обращения: 22.03.2020).

55 Современный предприниматель [Электронный ресурс] / Невзаров А.В. ; Web-мастер Костойкин Н.Д. – Электрон. дан. – М. : ООО Издательский дом «В-Медиа», 2018 – Режим доступа: <https://cutt.ly/aua9K23>, свободный. – Загл. с экрана. Яз. рус., англ.

56 СОЗДАЕМ СВЕРТОЧНУЮ НЕЙРОННУЮ СЕТЬ (CNN) В KERAS // Robohunter : сетевое сообщество робототехников. 2020. URL: <https://cutt.ly/Lua9XrK> (дата обращения: 22.03.2020).

57 Траск, А. Grokking Deep Learning / Траск А. – Manning Publications, 2019. – 336с.

58 Тч А. The 10 Neural Network Architectures Machine Learning Researchers Need To Learn // towards data science : интернет портал. 2017. 04 авг. URL: <https://cutt.ly/Lua9Van> (дата обращения: 22.03.2020).

59 Федорова, Г. В. Информационные системы / Г. В. Федорова. –М. : Изд-во Academia, 2013. – 208 с.

60 Хе К. Labeled Faces in the Wild Home // Labeled Faces: интернет портал. 2016. 25 июл. URL: <https://cutt.ly/5ua9No1> (дата обращения: 22.03.2020).

61 Хе К. SoftTriple Loss: Deep Metric Learning Without Triplet Sampling // Cornell University : интернет ресурс. 2019. окт. URL: <https://cutt.ly/9ua9MWn> (дата обращения: 22.03.2020).

62 Хоффер Э. Deep metric learning using Triplet network // Cornell University: интернет портал. 2018. 14 дек. URL: <https://arxiv.org/abs/1412.6622> (дата обращения: 22.03.2020).

63 Хоффман, Л. Дж. Современные методы защиты информации / Л.Дж. Хоффман. – М. : Советское радио, 2012. – 264 с.

64 Цянь Ц. Deep metric learning using Triplet network // Cornell University: интернет портал. 2020. 15 апр. URL: <https://arxiv.org/abs/1909.05235> (дата обращения: 22.03.2020).

65 AndrewShmig. CS231n: Сверточные нейронные сети для распознавания образов // Хабр: интернет портал. 2016. 13 окт. URL: <https://habr.com/ru/post/456186/> (дата обращения: 22.03.2020).

66 Arnis71. Нейронные сети для начинающих // Хабр : интернет портал. 2017. 12 фев. URL: <https://habr.com/ru/post/313216/> (дата обращения: 22.03.2020).

67 Caffe // Caffe : документация. URL: <http://caffe.berkeleyvision.org/> (дата обращения: 22.03.2020).

68 Cascade Classifier // OpenCV : документация. 2020. 03 июн. URL: <https://cutt.ly/Gua1IX8> (дата обращения: 22.03.2020).

69 Computer Vision Library for PyTorch // Kornia : документация. 2019. URL: <https://kornia.github.io/> (дата обращения: 22.03.2020).

70 Deep Metric Learning: A Survey // MDPI : интернет портал. 2019. 21 авг. URL: <https://cutt.ly/Yua1LfJ> (дата обращения: 22.03.2020).

71 Detect eyes, nose, lips, and jaw with dlib, OpenCV, and Python // pyimagesearch : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/jua1Ctc> (дата обращения: 22.03.2020).

72 Dlib // Dlib C++ Library : документация. 2019. 14 дек. URL: <http://dlib.net/> (дата обращения: 22.03.2020).

73 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/xua1M4l>, свободный. – Загл. с экрана. Яз. рус., англ.

74 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/6ua128C>, свободный. – Загл. с экрана. Яз. рус., англ.

75 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://www.dns-https://cutt.ly/fua18uO>, свободный. – Загл. с экрана. Яз. рус., англ.

76 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/tua17xV>, свободный. – Загл. с экрана. Яз. рус., англ.

77 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/Tua16Y4>, свободный. – Загл. с экрана. Яз. рус., англ.

78 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/sua0eH9>, свободный. – Загл. с экрана. Яз. рус., англ.

79 DNS-Shop [Электронный ресурс] / Web-мастер Хамченко К.А. – Электрон. дан. – М. : Компания DNS, 2018 – Режим доступа: <https://cutt.ly/1ua0lFV>, свободный. – Загл. с экрана. Яз. рус., англ.

80 COVID-19: Face Mask Detector with OpenCV, Keras/TensorFlow, and Deep Learning: [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Husrdbc> (дата обращения: 22.03.2020).

81 Face recognition with OpenCV: [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Qua0vDe> (дата обращения: 22.03.2020).

82 Facial landmarks with dlib, OpenCV, and Python // pyimagesearch : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Uua0mJy> (дата обращения: 22.03.2020).

83 Getting started // Keras : документация. 2010. URL: [https://keras.io/getting\\_started/](https://keras.io/getting_started/) (дата обращения: 22.03.2020).

84 High Quality Face Recognition with Deep Metric Learning // Dlib : блог. 2017. 12 фев. URL: <https://cutt.ly/jua0Exk> (дата обращения: 22.03.2020).

85 How to install TensorFlow 2.0 on macOS : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/iua0Trv> (дата обращения: 22.03.2020).

86 How to install TensorFlow 2.0 on Ubuntu : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/mua0UsZ> (дата обращения: 22.03.2020).

87 HybridTech. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество // Хабр : интернет портал. 2016. 13 окт. URL: <https://cutt.ly/Iua0O18> (дата обращения: 22.03.2020).

88 Installing Keras for deep learning : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Yua0Aux> (дата обращения: 22.03.2020).

89 Keras and Convolutional Neural Networks (CNNs) : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/fua0SnG> (дата обращения: 22.03.2020).

90 kornia // GitHub : документация. 2020. URL: <https://github.com/kornia/kornia/> (дата обращения: 22.03.2020).

91 Microsoft [Электронный ресурс] / Web-мастер Spark D. – Электрон. дан. – М. : Компания Microsoft, 2018 – Режим доступа: <https://cutt.ly/Rua0FZm>, свободный. – Загл. с экрана. Яз. рус., англ.

92 Microsoft [Электронный ресурс] / Windows 10 Pro. – Электрон. дан. – М. : Компания Microsoft, 2018 – Режим доступа: <https://cutt.ly/hua0JIJ>, свободный. – Загл. с экрана. Яз. рус., англ.

93 Microsoft [Электронный ресурс] / Windows 8.1. – Электрон. дан. – М. : Компания Microsoft, 2018 – Режим доступа: <https://cutt.ly/wua0LnK>, свободный. – Загл. с экрана. Яз. рус., англ.

94 Microsoft. Deep Learning: Transfer learning и тонкая настройка глубоких сверточных нейронных сетей // Хабр : интернет портал. 2019. 18 июн. URL: <https://cutt.ly/Eua0XcG> (дата обращения: 22.03.2020).

95 Multi-class SVM Loss : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Zua2AQW> (дата обращения: 22.03.2020).

96 NIX компания. Архитектуры нейросетей // Хабр : интернет портал. 2018. 21 ноя. URL: <https://cutt.ly/Cua2Hmf> (дата обращения: 22.03.2020).

97 PYTORCH DOCUMENTATION // PyTorch : документация. 2019. URL: <https://pytorch.org/docs/stable/index.html> (дата обращения: 22.03.2020).

98 Real-time facial landmark detection with OpenCV, Python, and dlib // pyimagesearch : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/dua2KQ1> (дата обращения: 22.03.2020).

99 Search Results for: pokedex : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Yua2CqZ> (дата обращения: 22.03.2020).

100 sklearn.preprocessing.LabelBinarizer // SCIKIT learn: документация. 2019. URL: <https://cutt.ly/pua2Vii> (дата обращения: 22.03.2020).

101 Welcome to PyBrain // PyBrain : документация. 2010. URL: <http://pybrain.org/> (дата обращения: 22.03.2020).

102 Why is my validation loss lower than my training loss? : [Электронный ресурс] / сост. Р. Адриан; URL: <https://cutt.ly/Sua2Nnu> (дата обращения: 22.03.2020).

## ПРИЛОЖЕНИЕ А

### (Обязательное)

#### Модуль загрузки датасета

##### Листинг А.1 – Текст модуля загрузки датасета

```
# python encode_faces.py --dataset dataset --encodings encodings.pickle

from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
                help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
                help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
                help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# захватить пути к входным изображениям в нашем наборе данных
print("[INFO] количественное определение лиц...")
imagePaths = list(paths.list_images(args["dataset"]))

# инициализировать список известных кодировок и известных имен
knownEncodings = []
knownNames = []

# цикл по путям изображения
for (i, imagePath) in enumerate(imagePaths):
    # извлечь имя человека из пути к изображению
    print("[INFO] обработка изображения {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # загрузить входное изображение и преобразовать его из RGB (порядок
    OpenCV) в порядок dlib (RGB)
    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```



```

        # определить (x, y) -координаты ограничительных рамок, соответ-
        ствующих каждому лицу во входном изображении
        boxes = face_recognition.face_locations(rgb,
                                                model=args["detection_method"])

        # вычислить вложение лица для лица
        encodings = face_recognition.face_encodings(rgb, boxes)

        # перебрать кодировки
        for encoding in encodings:
            # добавьте каждую кодировку + имя к нашему набору известных имен
и песен кодирования
            knownEncodings.append(encoding)
            knownNames.append(name)

        # сбросить лицевые кодировки + имена на диск
        print("[INFO] сериализационные кодировки...")
        data = {"encodings": knownEncodings, "names": knownNames}
        f = open(args["encodings"], "wb")
        f.write(pickle.dumps(data))
        f.close()

```

## ПРИЛОЖЕНИЕ Б

### (Обязательное)

#### Модуль распознавания личности по лицу и обнаружение наличия маски

Листинг Б.1 – Текст модуля распознавания личности по лицу и обнаружение наличия маски не лице

```
# python recognize_faces_video.py --encodings encodings.pickle --
shape-predictor shape_predictor_68_face_landmarks.dat

from imutils.video import VideoStream
import face_recognition
import argparse
import imutils
import pickle
import dlib
import time
import cv2

from scipy.spatial import distance as dist
from imutils.video import FileVideoStream
from imutils import face_utils

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
import numpy as np
import os

from serial import Serial, time

def eye_condition_calculation(eye):
    # вычисление евклидова расстояния между двумя наборами
    # вертикальных координат глаз (x, y) -координаты
    first_vertical_points = dist.euclidean(eye[1], eye[5])
    second_vertical_points = dist.euclidean(eye[2], eye[4])

    # вычисление евклидова расстояния между горизонтальными
    # ориентирами глаз (x, y) -координаты
    horizontal_points = dist.euclidean(eye[0], eye[3])

    # вычислить соотношение сторон глаз
```

```

        eye_condition = (first_vertical_points + second_vertical_points)
/ (2.0 * horizontal_points)

    return eye_condition

def detect_and_predict_mask(frame, faceNet, maskNet):
    # взять размеры рамки и затем создать из нее блов
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300), (104.0,
177.0, 123.0))

    # передача блова через сеть и получение распознавания лиц
    faceNet.setInput(blob)
    detections = faceNet.forward()

    # инициализация списка лиц, их соответствующих местоположений и
списка прогнозов из сети масок для лица
    faces = []
    locs = []
    preds = []

    # цикл на обнаружения
    for i in range(0, detections.shape[2]):
        # извлечь достоверность (то есть вероятность), связанную с
обнаружением
        confidence = detections[0, 0, i, 2]

        # отфильтровывать слабые обнаружения, гарантируя, что досто-
верность превышает минимальную достоверность
        if confidence > args["confidence"]:
            # вычисление (x, y) -координат ограничительной рамки для
объекта

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")

            # проверка, что ограничивающие рамки попадают в размеры
рамки

            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

            # извлечь лицо ROI, преобразовать его из BGR в порядок
каналов RGB, изменить размер до 224x224 и
            # препроцессировать
            try:
                face = frame[startY:endY, startX:endX]

```

```

        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
        face = cv2.resize(face, (224, 224))
        face = img_to_array(face)
        face = preprocess_input(face)
        face = np.expand_dims(face, axis=0)

        # добавить лицо и ограничивающие рамки в соответств-
ющие списки
        faces.append(face)
        locs.append((startX, startY, endX, endY))
    except:
        print('face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)')

    # делать прогнозы только в случае обнаружения хотя бы одного лица
    if len(faces) > 0:
        # для более быстрого вывода делаются пакетные предсказания на
* всех * гранях одновременно, а не одной за
        # другим в вышеуказанном цикле `for`
        preds = maskNet.predict(faces)

    # возвращение 2 кортежей местоположений лиц и их соответствующих
местоположений
    return (locs, preds)

def personality_recognition(total):
    # цикл по лицевым вложениям
    for encoding in encodings:
        # попытка сопоставить каждое лицо на входном изображении с
известными
        matches = face_recognition.compare_faces(data["encodings"],
encoding)
        name = "Unknown"

        # проверка соответствий
        if True in matches:
            # найти индексы всех совпадающих граней, затем их иници-
ализировать
            # словарь для подсчета общего количества раз каждого лица
            matchedIdxs = [i for (i, b) in enumerate(matches) if b]
            counts = {}

            # цикл по сопоставленным индексам и счет для
            # каждого распознанного лица
            for i in matchedIdxs:
                name = data["names"][i]

```

```

        counts[name] = counts.get(name, 0) + 1

        # определить распознанное лицо с наибольшим номером
        # количеств голосов (примечание: в случае маловероятного
связывания Python
        # выберут первую запись в словаре)
        name = max(counts, key=counts.get)

        # обновить список имен
        names.append(name)
    # перебрать распознанные лица
    for ((top, right, bottom, left), name) in zip(boxes, names):
        # изменить масштаб координат лица
        top = int(top * r)
        right = int(right * r)
        bottom = int(bottom * r)
        left = int(left * r)

        # вывод предсказанного имя по лицу
        y = top - 15 if top - 15 > 15 else top + 15
        cv2.putText(frame, name, (left + 65, y + 10), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 0, 0), 2)
        if name == "Unknown":
            total = 0
    return total

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
                help="path to serialized db of facial encodings")
ap.add_argument("-o", "--output", type=str,
                help="path to output video")
ap.add_argument("-y", "--display", type=int, default=1,
                help="whether or not to display output frame to
screen")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
                help="face detection model to use: either `hog` or
`cnn`")

ap.add_argument("-p", "--shape-predictor", required=True,
                help="path to facial landmark predictor")
ap.add_argument("-v", "--video", type=str, default="",
                help="path to input video file")

# маска
ap.add_argument("-f", "--face", type=str,

```

```

        default="face_detector",
        help="path to face detector model directory")
ap.add_argument("-m", "--model", type=str,
                default="mask_detector.model",
                help="path to trained face mask detector model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")

args = vars(ap.parse_args())

# com1 = Serial('COM6', 9600, timeout=0)

# определить две константы, одну для соотношения сторон глаза, чтобы
указать
# моргания, а затем вторая константа для числа последовательных
# кадров глаз, которые должны быть ниже порога
EYE_AR_THRESH = 0.28
EYE_AR_CONSEC_FRAMES = 2

# инициализация счетчика кадров и общее количество морганий
COUNTER = 0
TOTAL = 0

# инициализация детектора лица dlib (на основе HOG) и создание
# предиктора ориентира лица
print("[INFO] загрузка предиктора ориентир лица...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# загрузка известных лиц и вложения
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())

# захват индексов лицевых ориентиров слеваго и праваго глаз
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

# маска
# загрузка сериализованной модели детектора лиц с диска
print("[INFO] загрузка модели обнаружения лица...")
prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])
weightsPath = os.path.sep.join([args["face"],
"res10_300x300_ssd_iter_140000.caffemodel"])
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# загрузка модели обнаружения маски с диска

```

```

print("[INFO] загрузка модели обнаружения маски...")
maskNet = load_model(args["model"])

# запуск потока видео
print("[INFO] начало видеопотока...")
vs = FileVideoStream(args["video"]).start()
fileStream = True
# 1 - веб камера, 0 - веб камера с ноутбука
vs = VideoStream(src=1).start()
fileStream = False
time.sleep(1.0)

# зацикливание кадров из потока
while True:
    # захватить кадр из потокового видео
    frame = vs.read()

    # преобразовать входной кадр из BGR в RGB, а затем изменить его
размер, чтобы
    # ширина была 750px (для ускорения обработки)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    rgb = imutils.resize(frame, width=130)
    r = frame.shape[1] / float(rgb.shape[1])
    # преобразование его в оттенки серого
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # обнаружение лица в кадре серого
    rects = detector(gray, 0)

    # определить (x, y) -координаты ограничительных рамок
    # соответствует каждому лицу во входном кадре, затем вычислить
    # лицевые вложения для каждого лица
    boxes = face_recognition.face_locations(rgb, model=args["detection_method"])
    encodings = face_recognition.face_encodings(rgb, boxes)
    names = []

    try:
        # определения лица в кадре и определения носит ли он/она маску
или нет
        (locs, preds) = detect_and_predict_mask(frame, faceNet,
maskNet)
    except:
        print('(locs, preds) = detect_and_predict_mask(frame,
faceNet, maskNet)')

```

```

        # цикл по обнаруженным местоположениям лица и их соответствующим
местоположениям
        for (box, pred) in zip(locs, preds):

            if rects:
                if len(rects) > 1:
                    cv2.putText(frame, "Error!",
                                (100, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)
                    cv2.putText(frame, "You are more than one.",
                                (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)
                    cv2.putText(frame, "There should be one person in the
frame.",
                                (10, 125), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
(0, 0, 255), 2)
                else:
                    # определения ориентиров лица для области лица, затем
                    # преобразование координат лица (x, y) в Numpy массив
                    shape = predictor(gray, rects[0])
                    shape = face_utils.shape_to_np(shape)

                    # извлечь координаты левого и правого глаза
                    leftEye = shape[lstart:lend]
                    rightEye = shape[rstart:rend]

                    # вычисление пропорций глаз для обоих глаз
                    leftEAR = eye_condition_calculation(leftEye)
                    rightEAR = eye_condition_calculation(rightEye)

                    # усредните соотношение сторон для обоих глаз
                    ear = (leftEAR + rightEAR) / 2.0

                    # вычисление выпуклую оболочку для левого и правого
глаза
                    leftEyeNull = cv2.convexHull(leftEye)
                    rightEyeNull = cv2.convexHull(rightEye)

                    # визуализировать каждый глаз
                    cv2.drawContours(frame, [leftEyeNull], -1, (255, 0,
(0, 1)
                                0), 1)
                    cv2.drawContours(frame, [rightEyeNull], -1, (255, 0,
(0, 1)
                                0), 1)

                    # проверка, находится ли соотношение глаз ниже морга-
ния

```



```

# Порог, и если это так, увеличение счетчика кадров
моргания
    if ear < EYE_AR_THRESH:
        COUNTER += 1

# в противном случае, соотношение сторон глаза не ниже
порога моргания
    else:
        # если глаза были закрыты достаточное значение
        # то увеличить общее количество моргания
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
            TOTAL += 1

        # сбросить счетчик кадров глаз
        COUNTER = 0

# распаковка ограничивающих рамок и прогнозов
(startX, startY, endX, endY) = box
(mask, withoutMask) = pred

# определения метки класса и цвета, чтобы нарисовать ограни-
чивающий прямоугольник и текст
if mask > withoutMask:
    # рисование общее количество морганий на кадре вместе с
    # вычисляемой соотношением сторон для кадра
    if TOTAL >= 2:
        cv2.putText(frame, "This is a man", (10, 50),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        cv2.putText(frame, "Blinks: {} >= 2".format(TOTAL),
(10, 30),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        TOTAL = personality_recognition(TOTAL)
    else:
        cv2.putText(frame, "This photo", (10, 50),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "Blinks: {} <= 2".format(TOTAL),
(10, 30),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        label = "There is a mask"
else:
    if TOTAL >= 2:
        cv2.putText(frame, "This is a man", (10, 50),

```

```

cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        cv2.putText(frame, "Blinks: {} > 2".format(TOTAL),
(10, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        TOTAL = personality_recognition(TOTAL)
    else:
        cv2.putText(frame, "This photo", (10, 50),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "Blinks: {} <= 2".format(TOTAL),
(10, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255,
0), 2)
        label = "No mask"
        color = (0, 255, 0) if label == "There is a mask" else (0, 0,
255)

        try:
            # Вывод значения состояния глаза
            cv2.putText(frame, "Eye condition: {:.2f}".format(ear),
(230, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0),
2)
        except:
            print('cv2.putText(frame, "Eye condition: {:.2f}".for-
mat(ear), (230, 30), '
                'cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)')

        # вероятность метки
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) *
100)

        # отобразить метку и прямоугольник ограничительной рамки на
выходном кадре
        cv2.putText(frame, label, (startX, endY + 15), cv2.FONT_HER-
SHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color,
2)

        # проверка, надо ли отобразить выходной кадр на экран
        if args["display"] > 0:
            cv2.imshow("Personality Recognition", frame)
            key = cv2.waitKey(1) & 0xFF

        # если клавиша 'q' была нажата, то выйти из циклов

```

```
        if key == ord("q"):
            break

# очистка
cv2.destroyAllWindows()
vs.stop()
```

## ПРИЛОЖЕНИЕ В

### (Обязательное)

#### Модуль обучение нейросети обнаруживать маску

Листинг В.1 – Текст модуля обучение нейросети обнаруживать маску на лице

```
# python train_mask_detector.py --dataset dataset

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import os

# построить анализатор аргументов
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True, help="path to input dataset")
ap.add_argument("-p", "--plot", type=str, default="plot.png", help="path to output loss/accuracy plot")
ap.add_argument("-m", "--model", type=str, default="mask_detector.model", help="path to output face mask detector model")
args = vars(ap.parse_args())

# инициализация начальной скорости обучения
INIT_LR = 1e-4
```

```

# количество эпох для обучения
EPOCHS = 20
# размер пакета
BS = 32

# захват изображений в каталог наборов данных, затем инициализация
списков данных (т.е. изображений)
# и изображений классов
print("[INFO] загрузка изображений...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []
labels = []

# цикл по путям изображения
for imagePath in imagePaths:
    # извлечь метку класса из имени файла
    label = imagePath.split(os.path.sep)[-2]

    # загрузить входное изображение (224x224) и предварительно обрабо-
тать его
    image = load_img(imagePath, target_size=(224, 224))
    image = img_to_array(image)
    image = preprocess_input(image)

    # обновить данные и метки списки
    data.append(image)
    labels.append(label)

# преобразовать данные и метки в массивы NumPy
data = np.array(data, dtype="float32")
labels = np.array(labels)

# выполнить горячее кодирование на этикетках
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

# разбить данные на разделы обучения и тестирования, используя 80%
данных для обучения и оставшиеся 20% для тестирования
(trainX, testX, trainY, testY) = train_test_split(data, labels,
test_size=0.20, stratify=labels, random_state=42)

# построить генератор обучающих изображений для увеличения данных
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,

```

```

        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.15,
        horizontal_flip=True,
        fill_mode="nearest")

# загрузить сеть MobileNetV2 при отключенном заголовком FC
baseModel = MobileNetV2(weights="imagenet", include_top=False, input_tensor=Input(shape=(224, 224, 3)))

# построение головы модели, которая будет размещена поверх базовой модели
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# Поместить модель головы FC поверх базовой модели (она станет реальной моделью, которую мы будем тренировать)
model = Model(inputs=baseModel.input, outputs=headModel)

# перебрать все слои в базовой модели и заморозить их, чтобы они не обновлялись во время первого тренировочного процесса
for layer in baseModel.layers:
    layer.trainable = False

# компиляция модели
print("[INFO] компиляция модели...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])

# обучить головы сети
print("[INFO] тренировка головы...")
h = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# прогнозы на тестовом наборе
print("[INFO] тестирование сети...")
predIdxs = model.predict(testX, batch_size=BS)

```

```

        # для каждого изображения в тестовом наборе нужно найти индекс метки
        # с соответствующей наибольшей предсказанной
        # вероятностью
        predIdxs = np.argmax(predIdxs, axis=1)

        # показать отформатированный отчет о классификации
        print(classification_report(testY.argmax(axis=1), predIdxs, target_names=lb.classes_))

        # сериализовать модель на диск
        print("[INFO] сохранение модели детектора маски...")
        model.save(args["model"], save_format="h5")

        # график обучения потери и точности
        N = EPOCHS
        plt.style.use("ggplot")
        plt.figure()
        plt.plot(np.arange(0, N), h.history["loss"], label="train_loss")
        plt.plot(np.arange(0, N), h.history["val_loss"], label="val_loss")
        plt.plot(np.arange(0, N), h.history["accuracy"], label="train_acc")
        plt.plot(np.arange(0, N), h.history["val_accuracy"], label="val_acc")
        plt.title("Training Loss and Accuracy")
        plt.xlabel("Epoch")
        plt.ylabel("Loss/Accuracy")
        plt.legend(loc="lower left")
        plt.savefig(args["plot"])

```

## **ПРИЛОЖЕНИЕ Г**

### **(Обязательное)**

#### **Руководство пользователя**

##### **Введение**

Настоящий документ является руководством пользователя по эксплуатации программного обеспечения «Personality Recognition».

##### **Область применения**

Юридические лица пользуются программным обеспечением для защиты каких-либо данных от физического доступа людей, которые не имеют на это прав. Также в данном программном обеспечении реализовано распознавание наличия маски на лице, таким образом на предприятие можно фиксировать отсутствие маски на работнике.

Физические лица могут использовать программным обеспечением для предотвращения краж и грабежей в личных владениях (дом, дача).

##### **Краткое описание возможностей**

Основными возможностями являются:

- Создание обучающей выборки клиентов.
- Тренировка и тестирование нейросети.
- Определение лица на видео потоки.
- Фиксирование и подсчет морганий глазом.
- Распознавание личности по лицу
- Распознавание наличия маски на лице.

##### **Уровень подготовки пользователей**

Все пользователи программного обеспечения «Personality Recognition» должны иметь навыки работы с использованной операционной системой в предприятии или дома, например Windows, macOS, Linux. Иметь общие знания работы с программами, которые были написаны на языке программирования



Python, а именно уметь установить Python нужной версии и сочувствующие модули с нужными версиями. Уметь запускать консольные приложения и работать с аргументами.

### **Перечень программной и эксплуатационной документации**

На данный момент эксплуатационная документация имеет следующий состав:

- пояснительная записка в соответствии с РД 50-34.698-90;
- акт о внедрении результатов дипломной работы;
- руководство пользователя в соответствии с РД 50-34.698-90;
- выпускная квалификационная работа.

### **Назначение и условия применения**

Данная глава содержит следующие параграфы:

- Предмет автоматизации.
- Условия, обеспечивающие применение средств автоматизации в соответствии с назначением.

### **Предмет автоматизации**

Основным предметом автоматизации являются функции распознавание личности по лицу или обнаружение вторжения неизвестного человек в защищенный объект, а также обнаружение наличие маски на лице.

### **Условия, обеспечивающие применение средств автоматизации в соответствии с назначением**

Работа пользователей программного обеспечения «Personality Recognition» возможна при выполнении следующих требований к рабочему месту:

- Требования к программному обеспечению.
- Требования к техническому обеспечению.

### **Требования к программному обеспечению**

Данный раздел содержит подраздел «клиентская часть».

### **Клиентская часть**

В таблице 1 прописаны требования к конфигурации программного обеспечения клиентской части.

Таблица 1 – Требования к конфигурации программного обеспечения клиентской части

Компонент	Конфигурация
Операционная система	Windows 8.1, Windows 10, macOS Mojave, macOS Catalina, Ubuntu 20.04 LTS, Ubuntu Server 20.04 LTS
Общесистемное ПО	Python

### **Требования к техническому обеспечению**

Данный раздел содержит подраздел “клиентская часть”.

#### **Клиентская часть**

Для работы с программным обеспечением рабочие станции пользователей должны удовлетворять следующим минимальным требованиям:

Таблица 2 – Требования к конфигурации аппаратного обеспечения клиентской части

Компонент	Протестированная конфигурация
Процессор	i7 8750H
Оперативная память	16 Гб
Жесткий диск	1 Тб
Видеоадаптер	GTX 1050 2 Гб
Дополнительное оборудование	LED LG 22MT49VF-PZ

### **Подготовка к работе**

Перед началом работы с Personality Recognition необходимо установить соответствующее программы и дополнительное модули.

Программы и библиотеки, которые нужно установить перед использованием: cmake, python 3, visual studio (для Windows), Xcode (для macOS), pkg-config (для macOS), wget (для macOS), jpeg (для macOS), libpng (для macOS), libtiff (для macOS); openexr (для macOS), eigen (для macOS), tbb (для macOS), hdf5 (для macOS), build-essential (для Ubuntu), gcc-6 (для Ubuntu), g++-6 (для Ubuntu).

Далее нужно будет установить модули, которые попросит Python:

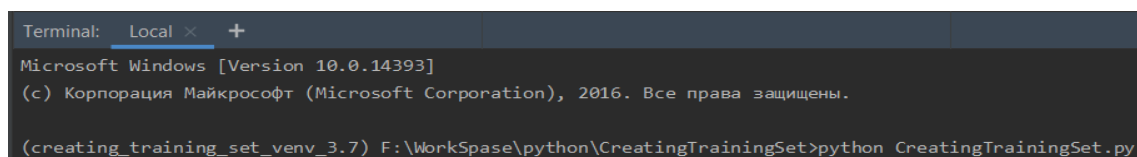
#### **Знакомство с программном обеспечением Personality Recognition**

Перед тем как приступить к работе с Personality Recognition, необходимо ознакомиться со следующей информацией:

- 1) запуск программ;
- 2) главные окна ПО Personality Recognition.

### Запуск программ

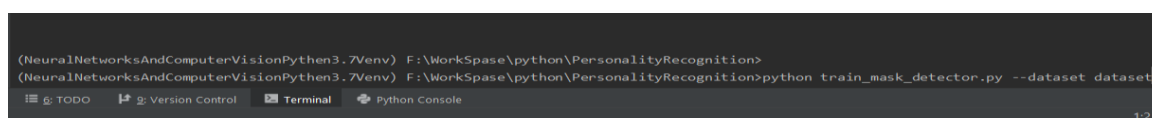
Программное обеспечение Personality Recognition представлено в виде консольного приложения. Для запуска программы, которая создает обучающую выборку клиента нужно в командную строку вести команду «python CreatingTrainingSet.py» (рисунок 64). Для обучения свёрточной нейросети нужно вести команду «python train\_mask\_detector.py --dataset dataset», где аргумент «--dataset» является каталогом откуда будут браться обучающие файлы (рисунок 65). Для запуска программы распознавания личности и обнаружения наличия маски на лице нужно вести команду «python recognize\_faces\_video.py --encodings encodings.pickle --shape-predictor shape\_predictor\_68\_face\_landmarks.dat» (рисунок 66).



```
Terminal: Local x +
Microsoft Windows [Version 10.0.14393]
(c) Корпорация Майкрософт (Microsoft Corporation), 2016. Все права защищены.

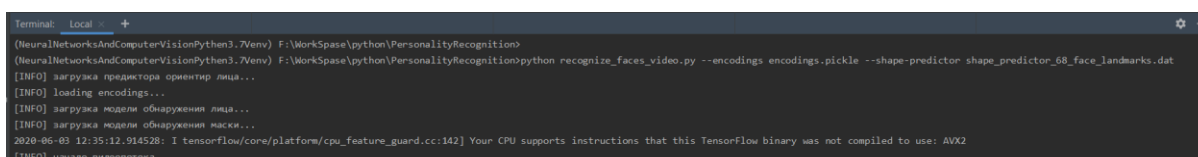
(creating_training_set_venv_3.7) F:\WorkSpace\python\CreatingTrainingSet>python CreatingTrainingSet.py
```

Рисунок 64 – Запуск программы для создания обучающей выборки



```
(NeuralNetworksAndComputerVisionPython3.7Venv) F:\WorkSpace\python\PersonalityRecognition>
(NeuralNetworksAndComputerVisionPython3.7Venv) F:\WorkSpace\python\PersonalityRecognition>python train_mask_detector.py --dataset dataset
```

Рисунок 65 – Запуск обучение нейросети



```
Terminal: Local x +
(NeuralNetworksAndComputerVisionPython3.7Venv) F:\WorkSpace\python\PersonalityRecognition>
(NeuralNetworksAndComputerVisionPython3.7Venv) F:\WorkSpace\python\PersonalityRecognition>python recognize_faces_video.py --encodings encodings.pickle --shape-predictor shape_predictor_68_face_landmarks.dat
[INFO] загрузка предиктора ориентир лица...
[INFO] loading encodings...
[INFO] загрузка модели обнаружения лица...
[INFO] загрузка модели обнаружения маски...
2020-06-03 12:35:12.914528: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
[INFO] начало видеопотока...
```

Рисунок 66 – Запуск основной программы, которая распознает личность или определяет, что человек «неизвестный» и обнаруживает наличие маски на лице или ее отсутствие

Рекомендуется соответствующие файлы и каталоги хранить в одном выбранном каталоге (рисунок 67).

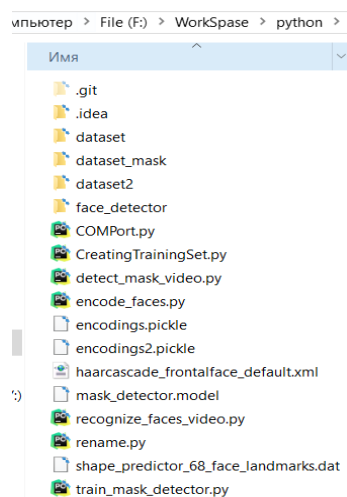


Рисунок 67 – Хранение файлов и каталогов, которые используются в программном обеспечении «Personality Recognition»

Каталог «dataset» хранит каталоги с фотографиями клиентов, которые будут использоваться для обучения и тестирования свёрточной нейронной сети (рисунок 68). Название каталог будет браться программой для отображения, если нейросеть распознает соответствующего клиента.

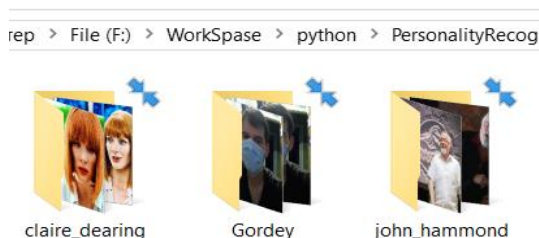


Рисунок 68 – Файлы для обучающей выборки

Файл «haarcascade\_frontalface\_default.xml» нужен для нахождения лица на видеопотоке с помощью библиотеки OpenCV. Файл «shape\_predictor\_68\_face\_landmarks.dat» является предиктором, который использует библиотека Dlib для наложения лицевых ориентиров. Файл «encodings.pickle» содержит кодировки распознавания лиц, которые генерируются из набора данных через «recognize\_faces\_video.py». Файл «mask\_detector.model» является моделью нейросети для обнаружения маски на лице. Файлы «deploy.prototxt» и «res10\_300x300\_ssd\_iter\_140000.caffemodel», которые находятся в каталоге «face\_detector» являются архитектурой и модели нейросети, которая обнаруживает лицо перед обнаружением маски на лице.

## Главные окна ПО Personality Recognition

На рисунке 69 изображено окно программы CreatingTrainingSet, которая создает обучающую выборку клиента. Красная рамка на лице означает, что лицо на видеопотоке обнаружено. В левом верхнем углу счетчик сделанных фотографий для обучающей выборки. Обучающая выборка сохраняется в созданный каталог «dataset».

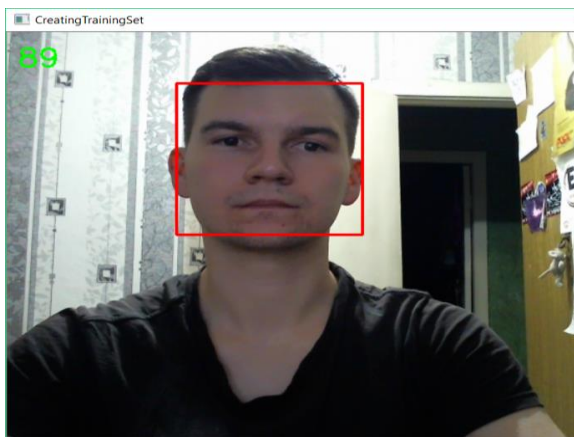


Рисунок 69 – Окно создания обучающей выборки

На рисунке 70 представлено основное окно Personality Recognition.

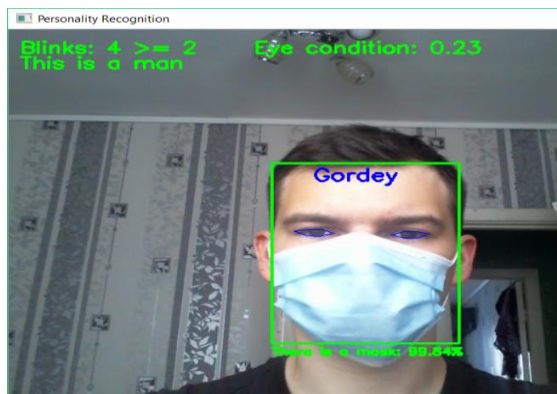


Рисунок 70 – Основное окно ПО

Программа фиксирует, чтобы в кадре не было больше одного человека, если это не так программа выводит сообщение об ошибке красным цветом (рисунок 71).



- --encodings, путь к сериализованной базе данных кодировок лица;
- --display, отображать или нет выводимый кадр на экран, по умолчанию установлена «1»;
- --shape-predictor, путь к предиктору лица;
- --face, путь к каталогу моделей детектора лиц, по умолчанию установлена «face\_detector»;
- --model, путь к обученной модели обнаружения маски, по умолчанию установлена «mask\_detector.model»;
- --confidence, минимальная вероятность фильтрации слабых обнаружений, по умолчанию установлена «0.51».

Для выхода из программы нужно нажать на клавишу «q».

### **Аварийные ситуации.**

Установка ПО «Personality Recognition» осуществляется системным администратором или программистом. Возможны следующие аварийные ситуации:

- Не заполнены обязательные аргументы – для устранения данной ошибки необходимо заполнить указанные аргументы.
- Системные сообщения – для устранения данной ошибки необходимо обратиться к системному администратору или программисту.

### **Рекомендации по освоению**

Для успешной работы с ПО «Personality Recognition» необходимо:

- получить навыки работы с используемой операционной системы, например, Windows, MacOS, Ubuntu;
- получить базовые навыки языка программирования Python;
- ознакомиться с данным руководством пользователя.