

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____ М.Х. Уртенев
_____ 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

СИСТЕМА НЕЙРОСЕТЕВОГО РАСПОЗНАВАНИЯ ЛИЦ В
МЕДИЦИНСКИХ МАСКАХ

Работу выполнил(а) _____ М.В. Сиухин

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Системный анализ, исследование операций и управление (Математическое и информационное обеспечение экономической деятельности)

Научный руководитель
канд. физ.-мат. наук, доцент _____ А.В. Письменский

Нормоконтролер
канд. физ.-мат. наук, доцент _____ Г.В. Калайдина

Краснодар
2021

РЕФЕРАТ

Выпускной квалификационной работы содержит 36 страниц, 26 рисунка, 12 источников.

НЕЙРОННЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, ИДЕНТИФИКАЦИЯ ЧЕЛОВЕКА, РАСПОЗНОВАНИЕ ОБРАЗОВ

Объектом выпускной квалификационной работы являются существующие нейронные сети и методы машинного обучения, применяемые в распознавания объектов.

Целью выпускной квалификационной работы является разработка приложения для идентификации человека в реальном времени, при этом лицо объекта частично закрыто медицинской маской. Итог проделанной работы – работающее приложение для идентификации человека, использующее нейронные сети. Реализована нейронная сеть с помощью библиотеки машинного обучения PyTorch и тестирование её в реальном времени используя библиотеку OpenCV.

СОДЕРЖАНИЕ

Введение.....	3
1 Основные понятия.....	5
1.1 Нейронная сеть.....	5
1.2 Функция активации.....	5
1.3 Функция потерь.....	8
1.4 Сверточная нейронная сеть.....	8
1.5 Субдискретизация.....	12
1.6 Связь сверточной нейронной сети с полносвязной нейронной сетью ...	13
2 Обучение модели	15
2.1 Обучение функции потерь	15
2.2 Обучение функции активации	15
2.3 Обучение полносвязного перцептрона.....	16
2.4 Обучение слоя макспулинга	17
2.5 Обучение свёрточного слоя	18
3 Программная реализация	20
3.1 Постановка задачи	20
3.2 Подготовка данных и построение модели.....	20
3.3 Тестирование	28
Заключение	34
Список используемой литературы	35
Приложение А	37
Приложение Б.....	38

ВВЕДЕНИЕ

Понятие нейронных сетей возникло при изучении процессов, протекающих при мышлении, и при попытке смоделировать эти процессы. Нейронная сеть представляет из себя систему соединенных между собой простых процессоров. Они довольно простые, и каждый из них обрабатывает входные сигналы и посылает другим процессорам. Если они соединены в большую сеть с управляемыми связями между ними, то такие локально простые процессоры вместе могут выполнять довольно сложные задачи.

В современных реалиях пандемии контроль над масочным режимом является важной функцией, но также масочный режим ставит новые задачи перед системами идентификации человека. Когда лицо каждого человека частично прикрыто маской, целесообразно является разработка средств идентификации человека, лицо которого частично прикрыто медицинской маской. Такое приложение может использоваться в медицинских учреждениях, в местах большого скопления людей, на любых контрольно-пропускном пункте. Благодаря высокой скорости работы приложение сможет одновременно оценивать и идентифицировать несколько человек.

В данной выпускной квалификационной работе будут рассмотрены понятия, связанные со свёрточными нейронными сетями, достаточными для построения работающей модели. Будут определены параметры для успешной работы модели. Выбраны средства реализации, тестирования и развёртывания модели. Модель будет реализована с помощью ЯП Python и библиотек для машинного обучения. Такой подход позволяет использовать преимущества Python и не терять в скорости работы программы, так как все вычисления выполняются на языке C.

Целью выпускной квалификационной работы является разработка приложения для идентификации человека в реальном времени, при этом лицо объекта частично закрыто медицинской маской. Итог проделанной работы –

работающее приложение для идентификации человека, использующее нейронные сети. Реализована нейронная сеть с помощью библиотеки машинного обучения PyTorch и тестирование её в реальном времени используя библиотеку OpenCV.

1 Основные понятия

1.1 Нейронная сеть

Нейронная сеть – это система процессоров (узлов), соединённых между собой, что позволяет обрабатывать простые признаки в большом количестве, искать между ними взаимосвязь. Это понятие пришло прямоком из биологии. Благодаря такой структуре, компьютер способен анализировать информацию или даже воспроизводить её в своей памяти. Нейронные сети способны решать сложные задачи, с которыми не справятся другие алгоритмы. К этим задачам относятся:

Классификация – распределение объектов работы нейронной сети по группам на основе входных данных. Например, на вход дается группа людей и нужно решить, кому дать кредит, а кому нет. Эту работу может сделать нейронная сеть, анализируя такую информацию как: возраст, заработная плата, длительность работы, кредитная история и т.д.

Предсказание – возможность предсказывать следующие события или анализ временных рядов. Например, предсказание роста заболеваемости, основываясь на данных прошлых лет.

Распознавание – в настоящее время, самое широкое применение нейронных сетей. Например, поиск объектов на фотографии. Когда нам нужно распознать рукописный текст на фото, это делает именно нейронная сеть.

1.2 Функция активации

Функция активации – функция, оценивающая результат работы одного конкретного узла и решающая передавать ли информацию с одного узла на другой. Фактически функция активации оценивает важность работы узла при определённых входных данных. Если значение, полученное в результате

работы узла достаточно большой, то информация с него передаётся дальше, иначе результатом работы становится 0.

Ниже формулы функций активации, которые мы будем использовать в будущей модели. Фактически, это просто перевод Y^l в X^l таким образом:

$$f(Y^l) = X^l \quad (1)$$

Функция активации позволяет сделать сеть нелинейной, и если бы мы не использовали функции активации или использовали бы линейную функцию, то неважно какое количество слоев тогда было бы в сети: их все можно было бы заменить одним единственным слоем с линейной функцией активации [5].

Функция ReLu:

$$F_{ReLU}(x) = \max(0, x) \quad (2)$$

И сигмоида:

$$F_{sigm}(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

Как мы и говорили эти функции нелинейные. На самом деле можно использовать любую нелинейную функцию, но предложенные здесь являются наиболее удачными для их использования в машинном обучении.

Сигмоида используется только если классов (для задачи классификации) не больше двух. Выход модели будет числом от нуля (первый класс) до

единицы (второй класс) [1]. Это связано с поведением функции сигмоиды на графике.

Графики этих функций показаны на рисунке 1:

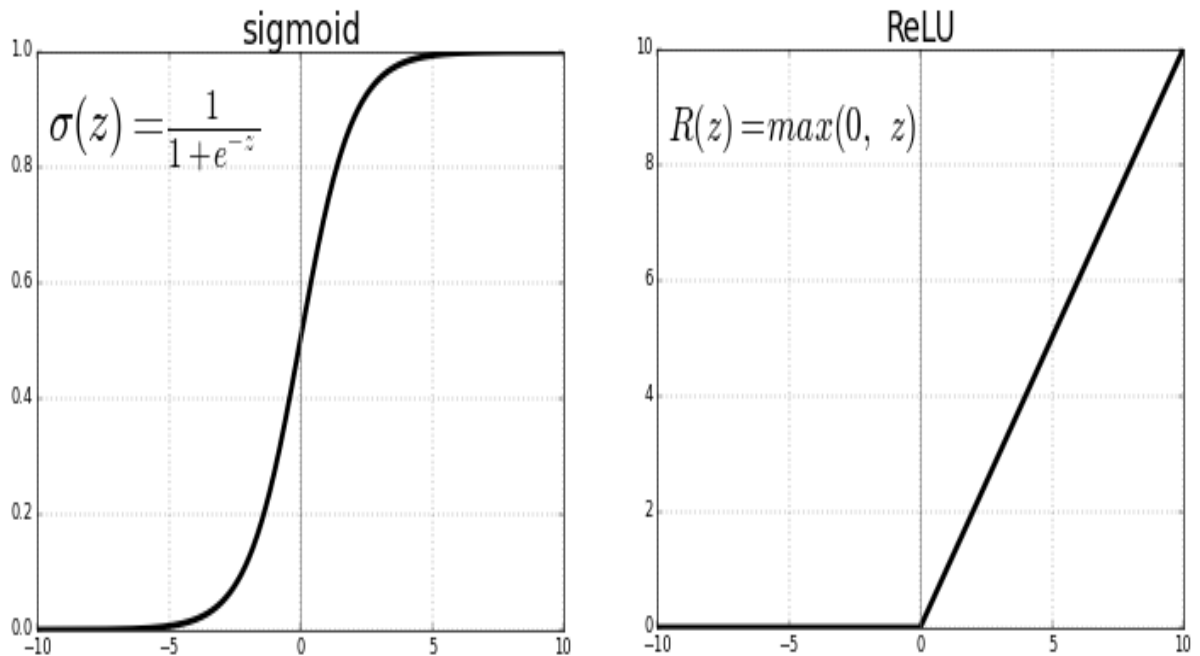


Рисунок 1 – Графики функций сигмоиды и ReLU

Для большего же числа классов, чтобы выход модели отражал вероятность этих классов (и сумма вероятностей по выходам сети равнялась единице), используется softmax.

$$f_{softmax}(X_i^l) = \frac{e^{x_i^l}}{\sum_{k=0}^n e^{x_k^l}} \quad (4)$$

где n равен числу классов. Так же отметим, что на разных слоях может быть разная функция активации. Это связано с тем что для различных операция лучше подходят разные функции.

1.3 Функция потерь

Функция потерь – функция, которая оценивает качество работы всей модели. Функция потерь находится в самом конце, после всех слоев сети. Выглядеть она может так:

$$E = \sum_{i=0}^n \frac{1}{2} (y_i^{try} - y_i^l)^2 \quad (5)$$

где n – это количество классов, y^l – выход модели, а y^{try} – правильные ответы. Коэффициент $\frac{1}{2}$ здесь нужен только для сокращения формулы во время обучения сети. Его можно убрать, что ничего принципиально не изменит [7].

1.4 Сверточная нейронная сеть

Свёрточные нейронные сети – специальная архитектура искусственных нейронных сетей, которая была предложена Яном Лекуном в 1988 году. Основное применение такая архитектура нашла в распознавании образов. Так же является представителем технологии глубокого обучения [3].

Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв и слоёв маспулинга. Структура сети – однонаправленная (без обратных связей), принципиально многослойная, так как всегда к свёрточным слоям добавляется перцептрон [4]. Это нужно так как свёрточные

слои только выделяют признаки в данных, чего недостаточно для достижения нашей цели. Для обучения используются стандартные методы, чаще всего метод обратного распространения ошибки или метод градиентного спуска. По сути, обучение нейронной сети, это поиск таких коэффициентов нейронов, что при прохождении данных сеть должна выдать правильный ответ. Отсюда задача обучения сводится к поиску минимума функции, поэтому все методы обучения являются методами нахождения локального экстремума. Функция активации нейронов - любая, по выбору исследователя.

Слой свёртки (англ. convolutional layer) является основным блоком свёрточной нейронной сети. Слой свёртки для каждого канала включает в себя свой фильтр или ядро свёртки который обрабатывает предыдущий слой по фрагментам (суммируя результаты поэлементного произведения для каждого фрагмента). Иначе говоря – это перебор всех элементов входного слоя и умножение их на какой-то фиксированный вес. Весовые коэффициенты ядра свёртки (как правило небольшой матрицы) изначально неизвестны (как правило используется единичная матрица) и устанавливаются в процессе обучения [1]. Процесс свёртки показан на рисунке 2.

Ядро свёртки итерационно, со сдвигом перемножают с частью обрабатываемого слоя (в самом начале – непосредственно с входным изображением), формируя после каждого шага сигнал активации с помощью соответствующей функции для нейрона следующего слоя с соответствующей позицией. То есть для различных нейронов выходного слоя используются одна и та же матрица весов и такой же функции активации. Тогда следующий слой, получившийся в результате операции свертки такой матрицей весов, показывает наличие данного признака в обрабатываемом слое и её координаты, формируя так называемую карту признаков.

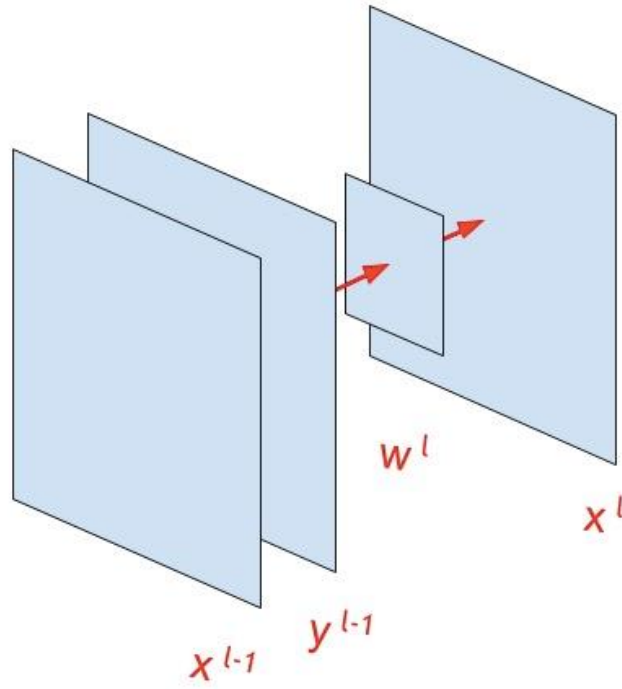


Рисунок 2 – Свёрточный слой

Будем использовать обозначения из рисунка 2. Следующая формула отображает движение ядра свёртки по входному слою:

$$X_{ij}^l = \sum_{a=-n}^n \sum_{b=-m}^m w_{ab}^l * y_{(i*s-a)(j*s-b)}^{l-1} + b^l; \forall i \in (0, \dots, N), \forall j \in (0, \dots, M) \quad (6)$$

Здесь подстрочные индексы i, j, a, b – это индексы элементов в матрицах, а s – величина шага свертки. Надстрочные индексы l и $l - 1$ – это индексы слоев сети.

Коэффициенты формулы имеют следующее обозначение:

y^{l-1} – выход какой-то предыдущей функции, либо входное изображение сети.

w^l – ядро свертки.

b^l – или смещение (на картинке выше отсутствует).

X^l – результат операции конволюции. То есть операции проходят отдельно для каждого элемента матрицы, размерность которой (N, M).

Важный нюанс этой формулы и самой операции вообще: у ядра существует центральный элемент. Причем центральный элемент может находиться не обязательно в центре, а абсолютно в любой ячейке ядра – какой нам нужно и какую мы выберем.

Индексация элементов ядра происходит в зависимости от расположения центрального элемента. Фактически, центральный элемент определяет начало нумерации элементов ядра свёртки. Обратим внимание на рисунок 3: слева ядро с центральным элементом в нулевой строке и нулевом столбце, справа – в первой строке и первом столбце.

На рисунке видно, как изменились индексы матрицы. Заметим, что индексы центрального элемента всегда равны (0,0). Условимся, что будем называть координаты старыми, если центральный элемент расположен в верхнем левом углу, в любом другом случае будем говорить, что координаты новые.

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)
(2,0)	(2,1)	(2,2)

(-1,-1)	(-1,0)	(-1,1)
(0,-1)	(0,0)	(0,1)
(1,-1)	(1,0)	(1,1)

Рисунок 3 – Ядра свёртки с разным центральным элементом

1.5 Субдискретизация

Слой макспулинга (субдискретизации) представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. При этом чаще всего употребляется функция максимума. Здесь, когда мы говорим пиксель, имеем в виду элемент матрицы (не обязательно двумерной, так как и входное изображение может состоять из нескольких каналов, например RGB)

Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в одно число, при этом выбирается значение матрицы, имеющие максимальное значение. Операция пулинга позволяет существенно уменьшить пространственный объём изображения (в зависимости от выбора размера ядра макспулинга). Пример работы слоя пулинга показан на рисунке 4.

Пулинг интерпретируется так: если на предыдущей операции свёртки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного [6].

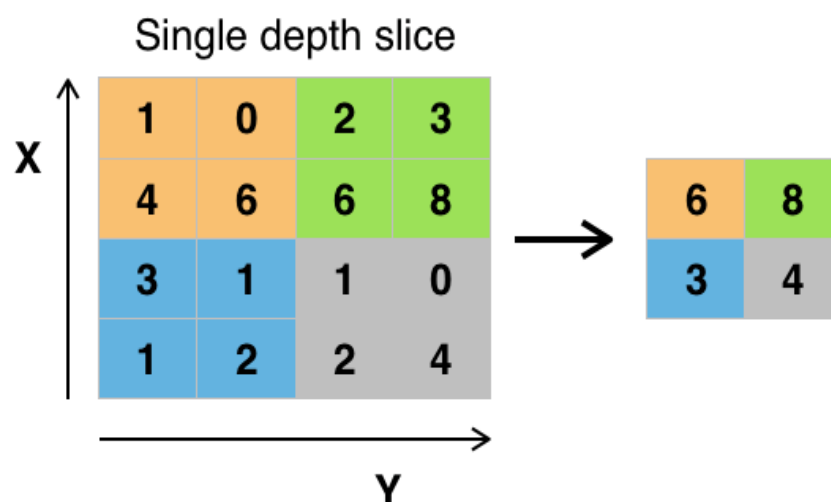


Рисунок 4 – Пример работы слоя пулинга

К тому же фильтрация уже ненужных деталей помогает не переобучаться, оставаться нейронной сети более стабильной и соответственно использовать больше эпох для обучения. Слой пулинга, как правило, вставляется после слоя свёртки перед слоем следующей свёртки.

1.6 Связь сверточной нейронной сети с полносвязной нейронной сетью

После слоев пулинга мы получим множество карт признаков. Их соединим в один вектор и этот вектор подадим на вход полносвязной сети, показанной на рисунке 5. Фактически у нас будет одна большая матрица, которую мы вытянем в вектор.

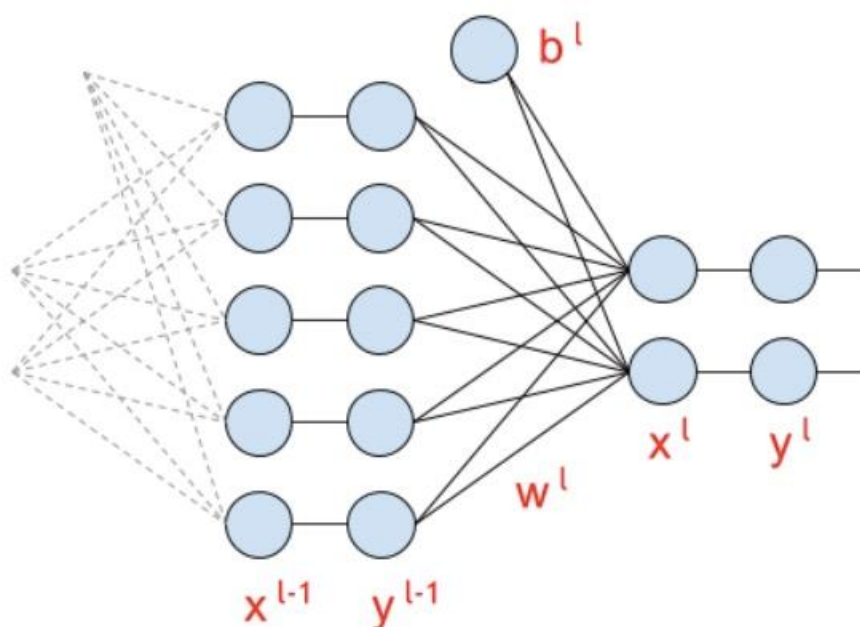


Рисунок 5 – Полносвязная нейронная сеть

Формула для слоя полносвязной сети выглядит следующим образом:

$$X_i^l = \sum_{k=0}^m w_{ki}^l * Y_k^{l-1} + b_i^l; \forall i \in (0, \dots, n) \quad (7)$$

2 Обучение модели

Для обучения модели будем использовать метод обратного распространения ошибки. Ниже будут представлены слои сверточной нейронной сети, описанные выше и идти они будут в обратном порядке для наглядности описания процесса обучения нейронной сети. В этой главе рассмотрим основные математические формулы, которые использует компьютер для корректирования весовых коэффициентов. Формулы помогут нам глубже понять, как работает наша нейронная сеть.

2.1 Обучение функции потерь

Обучение функции потерь – это частная производная. Пусть E это функция потерь, тогда получим следующую формулу 2:

$$\begin{aligned}\frac{\partial E}{\partial y_i^l} &= \frac{\partial \frac{1}{2} \sum_{i=0}^n (y_i^{try} - y_i^l)^2}{\partial y_i^l} = \frac{\partial \frac{1}{2} (y_i^{try} - y_i^l)^2}{\partial y_i^l} = \\ &= \frac{1}{2} \cdot 2 \cdot (y_i^{try} - y_i^l) \cdot \frac{\partial (y_i^{try} - y_i^l)}{\partial y_i^l} = -1 \cdot (y_i^{try} - y_i^l) = \\ &= y_i^l - y_i^{try}\end{aligned} \quad (8)$$

Отсюда видно зачем мы вставили $\frac{1}{2}$ в формулу 1.

2.2 Обучение функции активации

Сразу перейдём сразу к формулам. Производная для функции ReLU:

$$\frac{\partial y_i^l}{\partial x_i^l} = \begin{cases} 1, & x_i^l > 0 \\ 0, & x_i^l \leq 0 \end{cases} \quad (9)$$

Смысл формулы 9 в том, что мы пропускаем через ошибку те элементы, которые во время прямого прохождения через функцию активации были максимальными, и не пропускаем те, которые не были выбраны и не повлияли на результат работы.

Аналогично выведем формулу для функции сигмоиды:

$$\begin{aligned} \frac{\partial (1 + e^{-x_i^l})^{-1}}{\partial x_i^l} &= - (1 + e^{-x_i^l})^{-2} \cdot (-e^{-x_i^l}) = \\ &= \frac{1}{(1 + e^{-x_i^l})} \cdot \frac{1 + e^{-x_i^l} - 1}{(1 + e^{-x_i^l})} = \\ &= \frac{1}{(1 + e^{-x_i^l})} \cdot \\ &\cdot \left(\frac{(1 + e^{-x_i^l})}{(1 + e^{-x_i^l})} - \frac{1}{(1 + e^{-x_i^l})} \right) = f_{sigm} \cdot (1 - f_{sigm}) \end{aligned} \quad (10)$$

2.3 Обучение полносвязного перцептрона

Обучение перцептрона это, по сути, изменение матрицы весов w определённым образом. Ниже представлена формула 11 для обучения полносвязного слоя свёрточной сети:

$$\frac{\partial E}{\partial w_{ki}^l} = \frac{\partial E}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \frac{\partial x_i^l}{\partial w_{ki}^l} = \frac{\partial E}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \cdot \frac{\partial \sum_{j=0}^m w_{ji}^l * y_j^{l-1}}{\partial w_{ki}^l} = \frac{\partial E}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l} \cdot y_k^{l-1} \quad (11)$$

$$\forall i \in (0, \dots, n), \forall k \in (0, \dots, m)$$

Заменяя $\frac{\partial E}{\partial y_i^l} \frac{\partial y_i^l}{\partial x_i^l}$ на δ_i^l в формуле 4 получим $\delta_i^l \cdot y_k^{l-1}$. Также в сумме все элементы нулевые кроме случая $j = k$. В матричном виде это выглядит так:

$$\frac{\partial E}{\partial w^l} = (y^{l-1})^T * \delta^l \quad (12)$$

Но кроме весов w у нас есть матрица b . Выведем формулу и для неё:

$$\frac{\partial E}{\partial b_i^l} = \delta^l * \frac{\partial x_i^l}{\partial b_i^l} = \delta^l * \frac{\partial \sum_{j=0}^m w_{ji}^l * y_j^{l-1} + b_i^l}{\partial b_i^l} = \delta_i^l; \forall i \in (0, \dots, n) \quad (13)$$

2.4 Обучение слоя макспулинга

Ошибка “проходит” только через те значения исходной матрицы, которые были выбраны максимальными на шаге макспулинга. Остальные значения ошибки для матрицы будут равны нулю (что логично, ведь значения по этим элементам не были выбраны функцией макспулинга во время прямого прохождения через сеть и, соответственно, никак не повлияли на итоговый результат) [2]. Рисунок 6 хорошо иллюстрирует это.

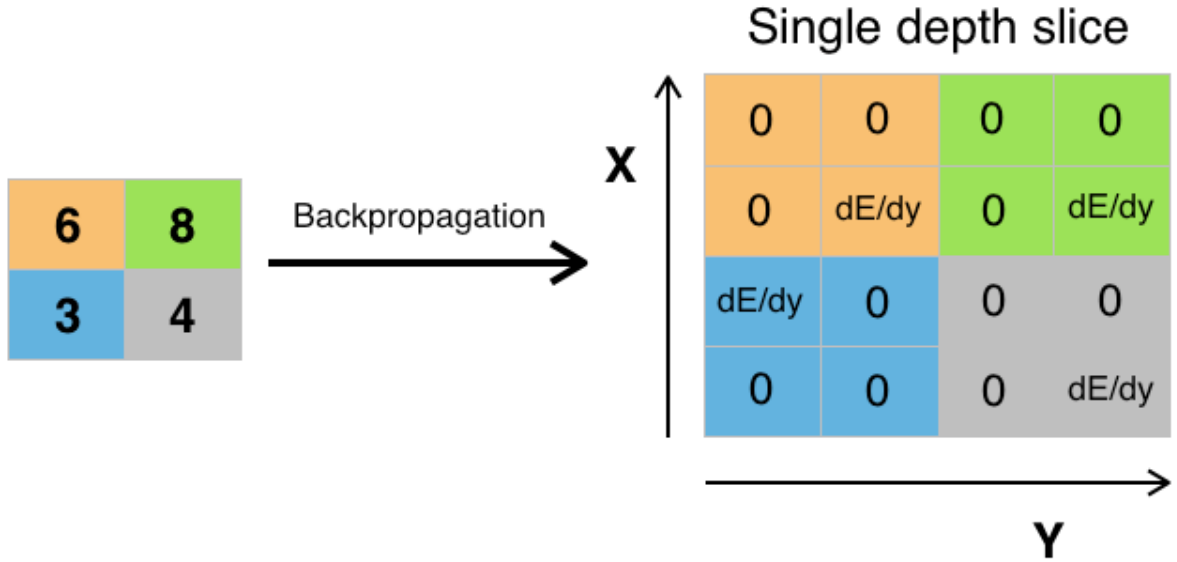


Рисунок 6 – Обучение слоя макспулинга

2.5 Обучение свёрточного слоя

Обучение свёрточного слоя заключается в нахождении весов ядра свёртки, так как именно ядро свёртки выделяет определённый признак на каждом шаге свёртки.

$$\begin{aligned}
 \frac{\partial E}{\partial w_{ab}^l} &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial x_{ij}^l}{\partial w_{ab}^l} = \\
 &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial \sum_{a_1} \sum_{b_1} w_{a_1 b_1}^l \cdot y_{(i-a_1)(j-b_1)}^{l-1} + b^l}{\partial w_{ki}^l} = \\
 &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \cdot y_{(i-a)(j-b)}^{l-1}
 \end{aligned} \tag{14}$$

Параметры a и b в формуле 14 такие же, как и в прямом ходе сети. Заметим что частные производные равны 0 во всех случаях, кроме того, когда $a = a_1$ и $b = b_1$.

Здесь важно увидеть, что в итоговой формуле не участвует само ядро свертки. Происходит некое подобие операции свертки, но с участием уже $\frac{\partial E}{\partial x_{ij}^l}$ и y^{l-1} , причем в роли ядра выступает $\frac{\partial E}{\partial x_{ij}^l}$, но все-таки это мало напоминает свертку.

Так же следует вывести формулу для b^l в ядре свёртке:

$$\begin{aligned}
 \frac{\partial E}{\partial b^l} &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial x_{ij}^l}{\partial b^l} \\
 &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l} \frac{\partial \sum_{a_1} \sum_{b_1} w_{a_1 b_1}^l * y_{(i-a_1)(j-b_1)}^{l-1} + b^l}{\partial b^l} \\
 &= \sum_i \sum_j \frac{\partial E}{\partial y_{ij}^l} \frac{\partial y_{ij}^l}{\partial x_{ij}^l}
 \end{aligned} \tag{15}$$

Приведенных выше формул хватит для обучения модели.

3 Программная реализация

3.1 Постановка задачи

Программной реализацией будет являться каскад нейронных сетей и методов машинного обучения, которые идентифицируют человека в медицинской маске. Задачу можно разбить на несколько подзадач: найти на фото лицо человека, понять в маске он или нет и идентифицировать его. Так же для обучения нейронных сетей потребуются тренировочные данные. Как правило такие данные плохо обработаны, поэтому перед обучением нам нужно будет заняться предобработкой данных.

Для реализации был выбран язык программирования Python. Выбор связан с большим набором инструментов для реализации нейронных сетей, их тестировании и обучении, так же Python обладает мощными средствами для обработки фотографий: изменение размера, применение фильтров. Хотя и Python является достаточно медленным языком программирования в сравнении с другими популярными языками, но это не мешает ему быть лидером в области машинного обучения и нейронных сетей. В сущности, все ресурсозатратные операции выполняются на языке C или C++, а Python предоставляет лишь высокоуровневый интерфейс к этим операциям.

3.2 Подготовка данных и построение модели

Для первой задачи воспользуемся уже готовым решением, включённым в библиотеку OpenCV. Будем использовать модель каскадной классификации с применением признаков Хаара. Это позволяет сети довольно быстро распознавать лицо на изображении и работать программе в высокой частоте кадров в реальном времени.

Каскад Хаара – это набор примитивных признаков, которые программа ищет на изображении. Примерная работа каскада Хаара показан на рисунке 7. Существуют различные каскады, как для поиска объектов, так и их частей. В своей работе я использовал стандартный каскад Хаара для поиска человеческих лиц. Он использует маски показанные на рисунке 8. Такие маски подчёркивают структурную информацию объекта: глаза будут темнее, чем область между ними, так же как область рта будет темнее чем лоб. Чем больше используется различных примитивов, тем точнее можно потом классифицировать объект. Но почему в каскадах Хаара используются только прямоугольники? Дело в том, что прямоугольные фигуры быстрее считаются через интегральное представление изображений. Если использовать синусоиды или похожие кривые, то количество действий будет пропорционально квадрату размера примитива, в то время как для прямоугольных примитивов требуется только 4 обращения к памяти и 3 математических операции.



Рисунок 7 – Пример работы признаков Хаара

Для второй и третьей части задачи воспользуемся библиотекой машинного обучения PyTorch. Библиотека предоставляет пользователю удобный, высокоуровневый интерфейс, быстрое вычисление математических операций, в частности свёртки, так как эта операция является наиболее затратной. Это позволит ускорить и обучение, и работу сети. Основным

конкурентом PyTorch является TensorFlow. Различие библиотек состоит в том, что PyTorch использует динамические вычислительные графы, а TensorFlow – статические вычислительные графы [12]. Это значит, что PyTorch не требуется перекомпилировать вычислительные графы перед выполнением, что даёт прирост в скорости подготовки нейронной сети, так как позволяет экспериментировать с входными данными. Так же к плюсам PyTorch можно отнести Python-подобный синтаксис и встроенные распределённые вычисления [11].

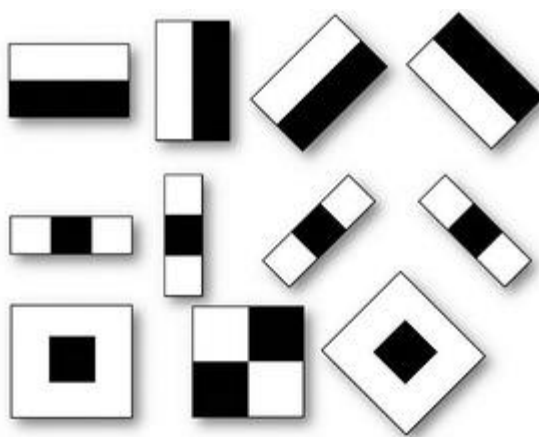


Рисунок 8 – Примитивы Хаара

Перед обучением первой нейронной сети нам нужно подготовить данные для обучения. “Сырые” данные представлены на рисунке 9.



Рисунок 9 – Необработанные данные

Первое, что нужно сделать, это привести изображения к одному размеру. Это сделаем с помощью встроенных инструментов PyTorch. Фреймворк обладает инструментом для предобработки входных данных. Изменим размер изображения на (120x120) пикселей, так же переведем изображение в тензор – векторное представление данных, понятное фреймворку, проведем нормализацию как показано в коде (Приложение А). На рисунке 10 представлено входное изображение после обработки.



Рисунок 10 – Изображение после обработки

Обучение происходило на выборке из 7553 изображений, из которых 3725 содержало изображения лиц людей в масках, а 3828 без масок. Все изображения были трехканальными т. е. изображения в формате RGB. Не будем переводить изображения в серую гамму, чтобы не снижать точность модели, а также это создаст дополнительные нагрузки при работе в реальном времени. Перед началом обучения приведем все данные в один размер – (120x120) пикселя. Обучение будем проводить на CPU, используя стохастический градиентный спуск, а именно его модификацию Adam [9] (метод адаптивной оценки моментов). Функция потерь – кросс энтропийная. Изображения будем “скармливать” нейронной сети пачками по 128 штук, это и ускорит обучение и даст большую точность после обучения. На рисунке 11 показано сравнение обучения с пачками по 64 (слева) и по 128(справа) штук.

[1, 119] loss: 0.470	[1, 60] loss: 0.508
[2, 119] loss: 0.383	[2, 60] loss: 0.411
[3, 119] loss: 0.361	[3, 60] loss: 0.376
[4, 119] loss: 0.368	[4, 60] loss: 0.395
[5, 119] loss: 0.359	[5, 60] loss: 0.321
[6, 119] loss: 0.311	[6, 60] loss: 0.296
[7, 119] loss: 0.288	[7, 60] loss: 0.314
[8, 119] loss: 0.277	[8, 60] loss: 0.361
[9, 119] loss: 0.276	[9, 60] loss: 0.272
[10, 119] loss: 0.263	[10, 60] loss: 0.269
Finished Training	Finished Training
Accuracy of the network on the 7553 test images: 88 %	Accuracy of the network on the 7553 test images: 89 %

Рисунок 11 – Сравнение результатов обучения с разным количеством данных в одной итерации обучения

Сама модель состоит из трёх слоёв свёртки, к каждому из которых подключен слой макспулинга, и трёх полносвязных слоёв. Каждое ядро свёртки будет иметь размер (5x5), а также увеличивать число каналов до 16, 32, 64 соответственно. Слой макспулинга будет выбирать максимальный элемент из матрицы (2x2). Полносвязные слои на входе содержат 7744, 4096, 2048 вершины. Для перехода от слоя свёртки к перцептрону нужно преобразовать матрицу в вектор. Для этого просто «вытянем» его и сделаем его размер равным (1x7744). Для всех слоёв будем использовать ReLU в качестве функции активации. Функцией потерь выберем функцию кросс-энтропии [10]. Обучение будет происходить на 50 эпохах. Нейронная сеть не успеет переобучиться и достигнет высокой точности. На рисунке 12 показано как меняется значение функции потерь при обучении на 10 эпохах. Как видно из графика значение уменьшается, это говорит о том, что модель обучается верно, обучающие данные хорошо подготовлены. На рисунке 13 показано как меняется точность работы модели на 10 эпохах. Как видно, модель уже на 1 эпохе показывает достойные результаты. Замечу, что эти значения актуальны только для наших тестовых данных, на реальных данных точность может незначительно отличаться.

Для второй нейронной сети подготовим данные самостоятельно. Сделаем 10 фотографий человека, которого будем распознавать. Для увеличения выборки отзеркалим эти изображения. Можно наложить ещё фильтров, для увеличения обучающих данных, но для нашей задачи это излишне.

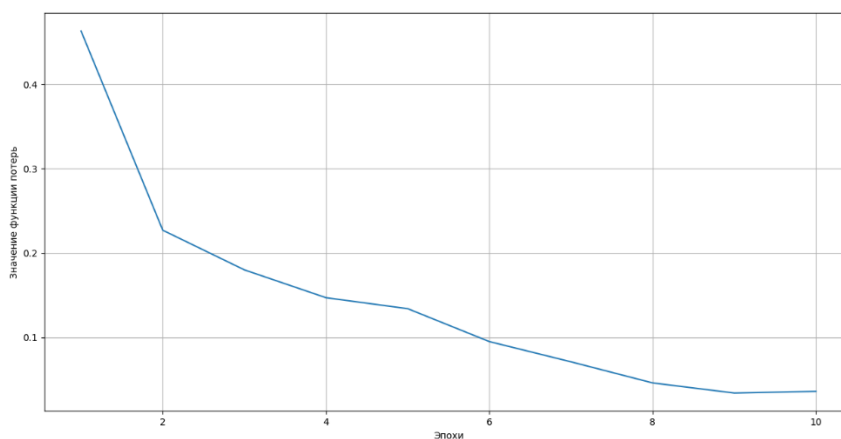


Рисунок 12 – Изменение функции потерь

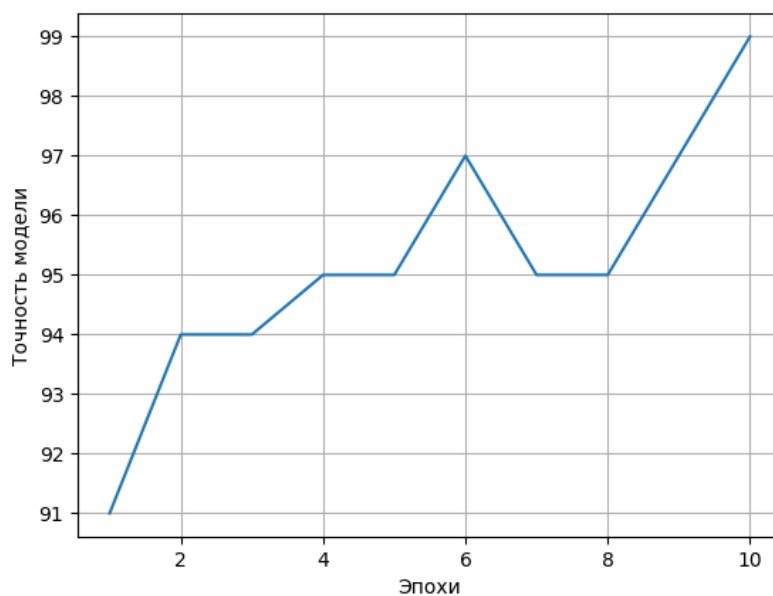


Рисунок 13 – Точность модели на 10 эпохах

Данные для обучения показаны на рисунке 14. Заметим, что данные также как и в первой нейронной сети приведём к размеру (120x120) и нормализуем.

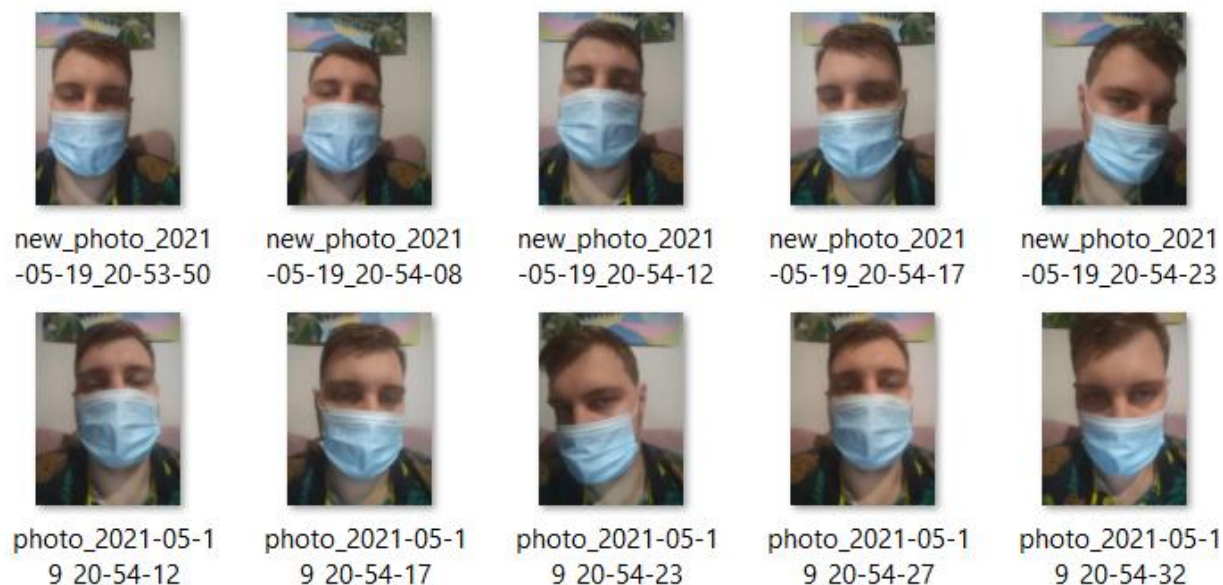


Рисунок 14 – Набор для обучения второй нейронной сети

Подготовленные данные представлены на рисунках 15,16.



Рисунок 15 – Первый класс после обработки изображений



Рисунок 16 –Второй класс после обработки изображений

На рисунке 17 показано изображение, которое использует нейронная сеть непосредственно при обучении.



Рисунок 17 – Изображение используемое второй нейронной сетью

Нейронная сеть будет иметь 1 слой свёртки с ядром свёртки размером (5x5) и макспулингом для матрицы (2x2), так же два полносвязных слоя, с функцией активации ReLU. Входные вектора для этих слоёв равны 215296 и 2048, итоговое число выходов равно количеству человек для распознавания, в нашем случае выхода два. Некоторые классы представлены на рисунке 18.



Имя	Дата изменения	Тип	Размер
 Mihail	19.05.2021 21:28	Папка с файлами	
 Yaroslav	19.05.2021 21:28	Папка с файлами	

Рисунок 18 – Классы для идентификации

Обучение будет происходить на 10 эпохах для того, чтобы сеть не переобучилась.

3.3 Тестирование

После обучения наших сетей получим хорошие результаты. Первая сеть обучилась с точностью 91 процент (рисунок 19)

```
[25, 60] loss: 0.234
[26, 60] loss: 0.185
[27, 60] loss: 0.190
[28, 60] loss: 0.186
[29, 60] loss: 0.177
[30, 60] loss: 0.179
[31, 60] loss: 0.183
[32, 60] loss: 0.172
[33, 60] loss: 0.173
[34, 60] loss: 0.168
[35, 60] loss: 0.190
[36, 60] loss: 0.171
[37, 60] loss: 0.166
[38, 60] loss: 0.198
[39, 60] loss: 0.209
[40, 60] loss: 0.171
[41, 60] loss: 0.176
[42, 60] loss: 0.158
[43, 60] loss: 0.163
[44, 60] loss: 0.147
[45, 60] loss: 0.159
[46, 60] loss: 0.162
[47, 60] loss: 0.157
[48, 60] loss: 0.165
[49, 60] loss: 0.153
[50, 60] loss: 0.189
Finished Training
Accuracy of the network on the 7553 test images: 91 %
```

Рисунок 19 – Результат обучения первой нейронной сети

Вторая сеть обучилась с точностью 87 процентов на двух классах. (рисунок 20).

```
[1,      1] loss: 0.694
[2,      1] loss: 1.176
[3,      1] loss: 50.618
[4,      1] loss: 3.896
[5,      1] loss: 17.688
[6,      1] loss: 5.957
[7,      1] loss: 4.431
[8,      1] loss: 15.382
[9,      1] loss: 12.948
[10,     1] loss: 2.771
Finished Training
Accuracy of the network on the 40 test images: 87 %
```

Рисунок 20 – Результат обучения второй нейронной сети с 2 классами

Если обучить сеть всего одной эпохой получим точность 75 процентов (рисунок 21).

```
[1,      1] loss: 0.696
Finished Training
Accuracy of the network on the 40 test images: 75 %
```

Рисунок 21 – Результат обучения второй нейронной сети на 1 эпохе

Добавим классов для обучения и посмотрим, что у нас получится. Результаты обучения представлены на рисунке 22. Точность повысилась, что говорит нам о том, что с увеличением классов, точность нейронной сети не уменьшается.

```
['Mary', 'Mihail', 'Sergay', 'Serik', 'Yaroslav']  
5 {'train': 98, 'test': 98}  
[1,      1] loss: 1.607  
[2,      1] loss: 5.084  
[3,      1] loss: 11.388  
[4,      1] loss: 1.829  
[5,      1] loss: 0.431  
[6,      1] loss: 0.446  
[7,      1] loss: 0.397  
[8,      1] loss: 0.202  
[9,      1] loss: 0.082  
[10,     1] loss: 0.096  
Finished Training  
Accuracy of the network on the 98 test images: 98 %
```

Рисунок 22 – Результат обучения второй нейронной сети с 5 классами

Работа программы в реальном времени показана на рисунках 23 и 24, подключим только первую нейронную сеть и проверим её работоспособность. Программа показывает хорошие результаты в точности прогноза, а также работает в высокой частоте кадров.

На рисунке 23 видно, что программа точно определяет человека без маски, даже с учётом того, что условия освещения не идеальные

На рисунке 24 наблюдаем аналогичную картину. Отметим, что каскады Хаара найдут на фото человеческое лицо, даже частично скрытое медицинской маской.

Добавим вторую нейронную сеть и повторим наши тесты. Идентифицировать человека будем только в том случае, если надета маска. Результаты тестирования показаны на рисунке 25.

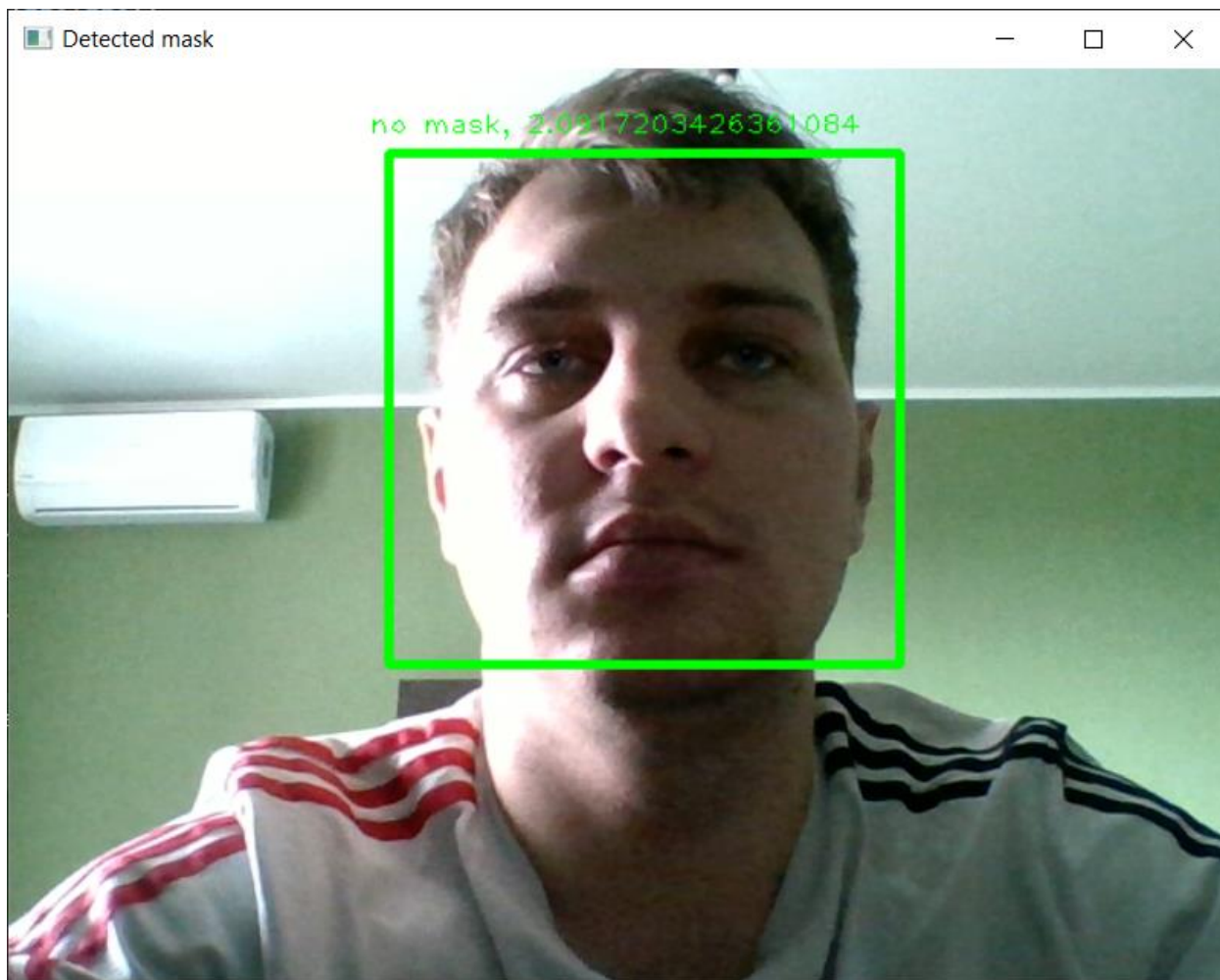


Рисунок 23 – Тестирование на человеке без маски

Для того, чтобы у нас не было идентификации людей, не входящих в нашу обучающую выборку, нужно ввести “коэффициент схожести”.

Так как нейронная сеть выдаёт вектор чисел, значение которых можно интерпретировать как, схожесть с обучающей выборкой, то сделаем минимальным допустимым числом схожести 20 как показано в коде, представленном в приложении Б.

На рисунке 26 показано как программа работает с несколькими людьми на одном фото.

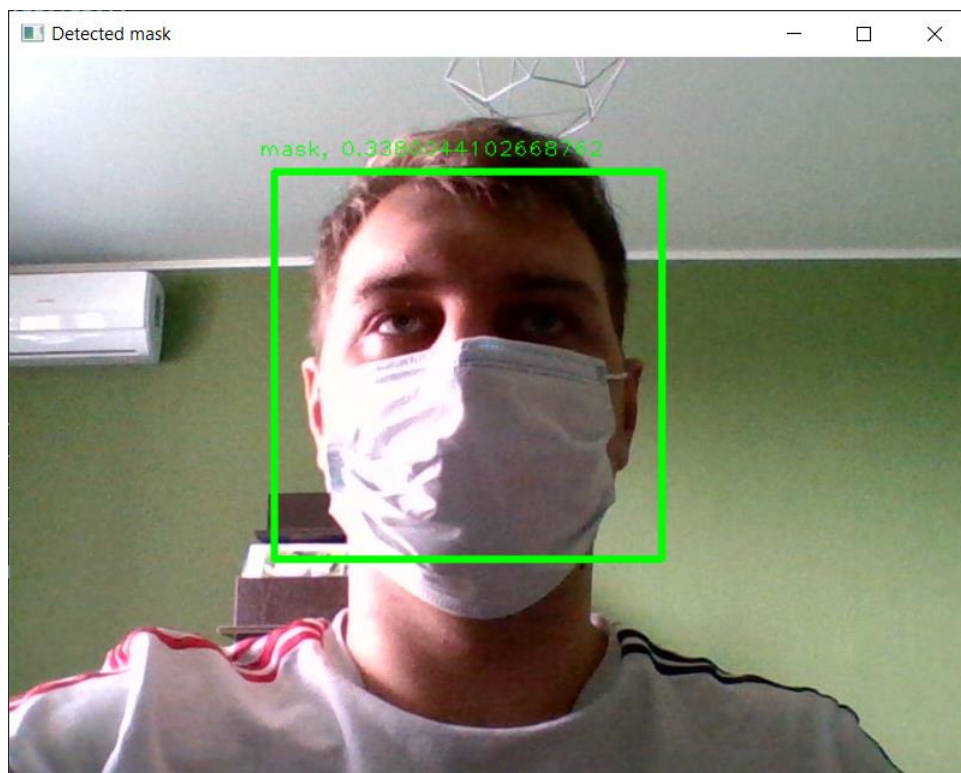


Рисунок 24 – Тестирование на человеке с маской

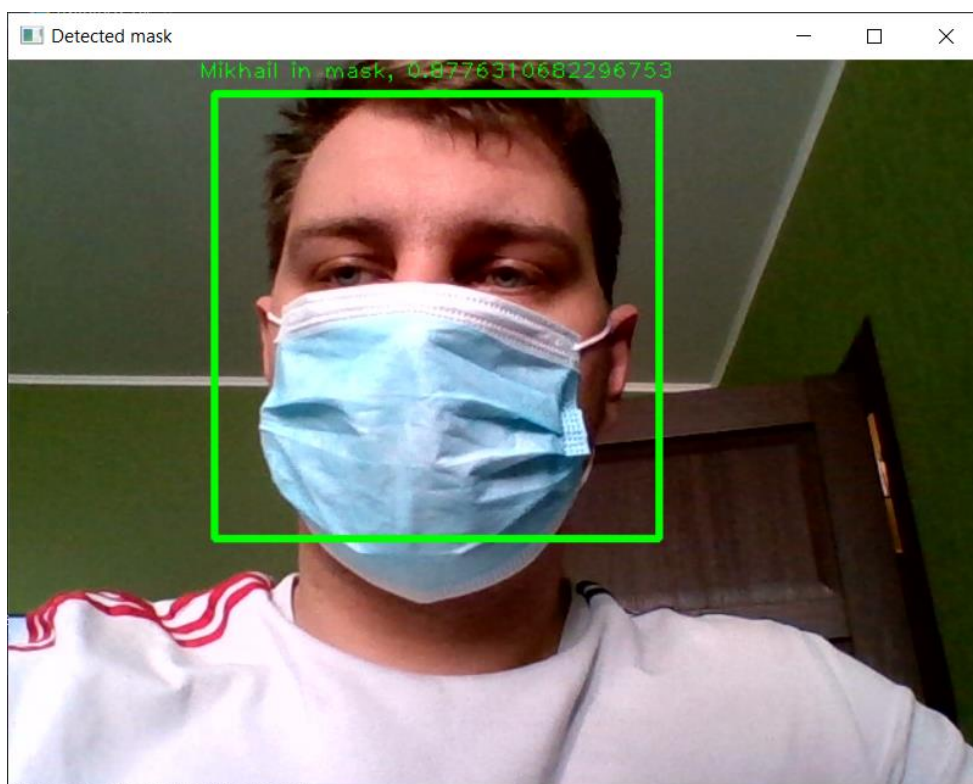


Рисунок 25 – Идентификация человека в маске

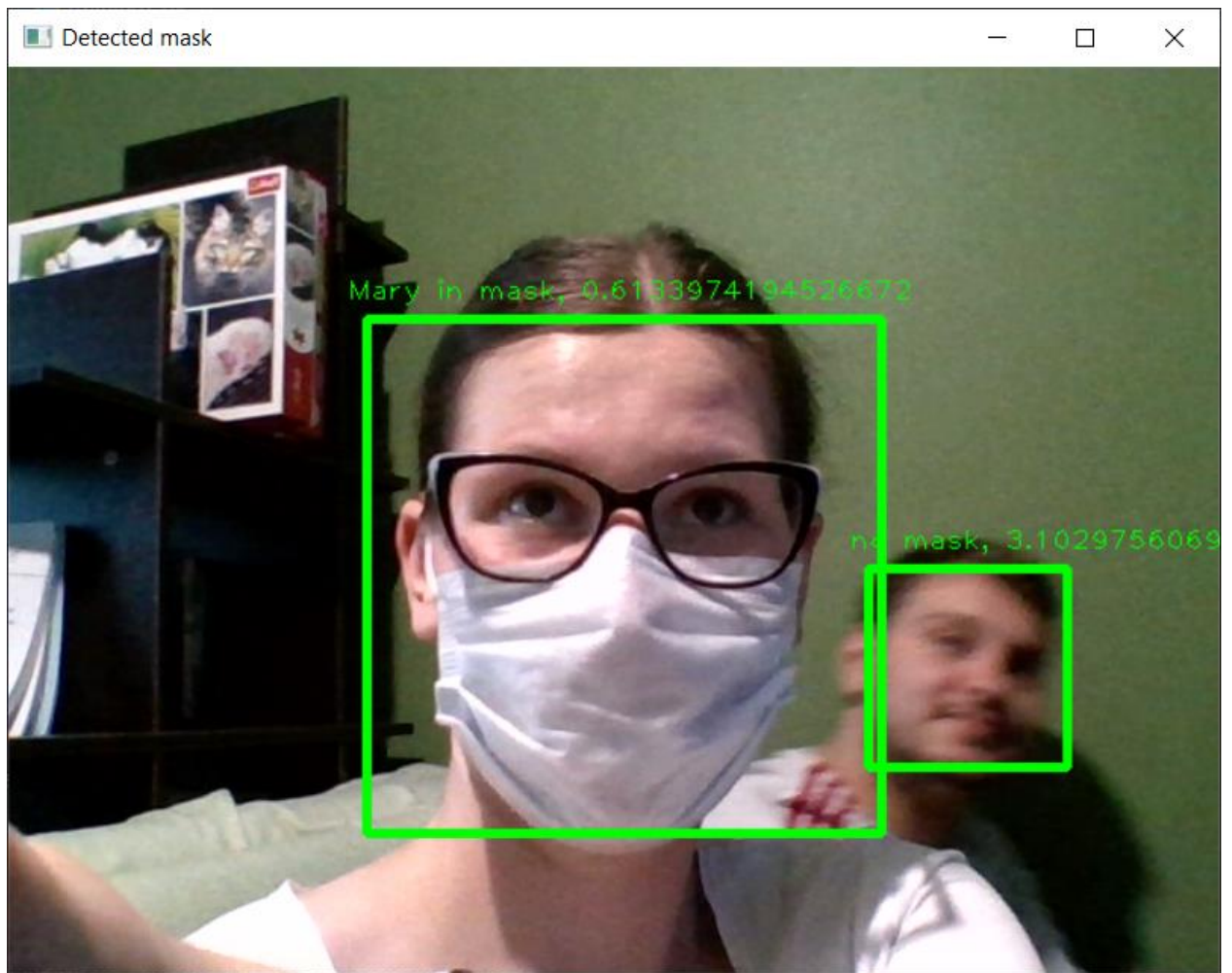


Рисунок 26 – Работа программы с несколькими людьми

Отметим, что благодаря функциональной парадигме Python и используемых нами фреймворков, наше решение может быть интегрировано в любую систему использующую Python, от десктопных приложений до web-приложений.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа посвященная разработка приложения для идентификации человека в реальном времени, при этом лиц объекта частично закрыто медицинской маской.

В ходе выполнения выпускной квалификационной работы были рассмотрены понятия, связанные со свёрточными нейронными сетями, достаточными для построения работающей модели. Были определены параметры для успешной работы модели. Выбраны средства реализации, тестирования и развёртывания модели.

Итог проделанной работы – работающее приложение для идентификации человека, использующее нейронные сети. Реализована нейронная сеть с помощью библиотеки машинного обучения PyTorch и тестирование её в реальном времени используя библиотеку OpenCV.

Разработанное нами приложение может встраиваться в любые системы с использованием Python или же выступать самостоятельным приложением для обеспечения безопасности или выполнения других схожих функций. Приложение показывает хорошие результаты при тестировании и может решить поставленные нами задачи.

В современных реалиях пандемии контроль над масочным режимом является важной функцией, но также масочный режим ставит новые задачи перед системами идентификации человека. Когда лицо каждого человека частично прикрыто маской, целесообразно является разработка средств идентификации человека, лицо которого частично прикрыто медицинской маской. Такое приложение может использоваться в медицинских учреждениях, в местах большого скопления людей, на любых контрольно-пропускном пункте. Благодаря высокой скорости работы приложение сможет одновременно оценивать и идентифицировать несколько человек.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Сверточная сеть на python. Часть 1. Определение основных параметров модели URL: <https://habr.com/ru/company/ods/blog/344008/> (дата обращения 10.05.2021)
2. Сверточная сеть на python. Часть 2. Вывод формул для обучения модели URL: <https://habr.com/ru/company/ods/blog/344116/> (дата обращения 10.05.2021)
3. Свёрточная нейронная сеть URL: https://ru.wikipedia.org/wiki/Сверточная_нейронная_сеть (дата обращения 10.05.2021)
4. Траск Э. Грожаем Глубокое машинное обучение / Э. Траск. – СПб.: Питер, 2019. – 352с.
5. Рохас Р., Neural Networks: A Systematic Introduction / Р. Рохас. – Springer, 1996. – 435с.
6. Neural networks and physical systems with emergent collective computational abilities URL - <https://bi.snu.ac.kr/Courses/g-ai09-2/hopfield82.pdf> (дата обращения 13.05.2021)
7. Learning and relearning in Boltzmann machines URL: https://www.researchgate.net/publication/242509302_Learning_and_relearning_in_Boltzmann_machines (дата обращения 13.05.2021)
8. Использование каскада Хаара для сравнения изображений URL: <https://habr.com/ru/post/198338/> (дата обращения 28.05.2021)
9. Стохастический градиентный спуск URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%BE%D1%85%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B3%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%

82%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D1%83%D1%81%D0%B
A#Adam (дата обращения 28.05.2021)

10. CROSSENTROPYLOS – PyTorch URL:
<https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html> (дата
обращения 28.05.2021)

11. Понимание сверточных нейронных сетей через визуализации в
PyTorch URL – <https://habr.com/ru/post/436838/> (дата обращения 28.05.2021)

12. TENSORFLOW VS PYTORCH URL: [https://python-
school.ru/tensorflow-vs-pytorch/](https://python-school.ru/tensorflow-vs-pytorch/) (дата обращения 28.05.2021)

ПРИЛОЖЕНИЕ А

Фрагмент листинга программы для предобработки входных данных

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop((120, 120)),
        transforms.ToTensor(),
        transforms.Normalize([0.5, 0.5, 0.5], [0.5,
0.5, 0.5])
    ]),
    'test': transforms.Compose([
        transforms.RandomResizedCrop((120, 120)),
        transforms.ToTensor(),
        transforms.Normalize([0.5, 0.5, 0.5], [0.5,
0.5, 0.5])
    ]),
}
```

ПРИЛОЖЕНИЕ Б

Фрагмент листинга программы для идентификации человека

```
result = net_ident(image_tensor)
_, predicted = torch.max(result.data, 1)
print(classes[predicted], result[0])
if result[predicted]>=20:
    cv2.putText(im,
f'{classes[predicted]} in mask, {predict[0][1]}', (x -
10, y - 10),
cv2.FONT_HERSHEY_PLAIN, 1,
(0, 255, 0))
else:
    cv2.putText(im, f'anonymous in mask,
{predict[0][1]}', (x - 10, y - 10),
cv2.FONT_HERSHEY_PLAIN,
1, (0, 255, 0))
```