для инженерных, технических, математических, естественнонаучных направлений/специальностей

федеральное государственное бюджетное образовательное учреждение высшего образования
«Вологодский государственный университет»

Институт математики, естественных и компьютерных наук
наименование института

Кафедра автоматики и вычислительной техники
наименование кафедры

Мудрик Роман Сергеевич
Ф.И.О. обучающегося полностью в именительном падеже

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.04.	43.10	10	1
код направления подготовки/ специальности	код выпускающей кафедры	порядковый номер темы ВКР по приказу	код формы обучения

Программная инженерия
направление подготовки/специальность

Разработка программно-информационных систем
направленность (профиль/специализация)

Разработка нейронной сети для распознавания эмоции человека
наименование темы

Допустить к защите:

Директор института_____(_____)
подпись, дата, расшифровка

Заведующий кафедрой_____(_____)
подпись, дата, расшифровка

Руководитель ВКР_____(_____)
подпись, дата, расшифровка

Нормоконтролёр_____(_____)
подпись, дата, расшифровка

Обучающийся_____(_____)
подпись, дата, расшифровка

Вологда
2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

За последние годы мир становится более автоматизированным. Искусственный интеллект вошел в обиход человеческой жизни.

Ученые и программисты всего мира разработали большое количество различных алгоритмов машинного обучения, такие как: распознавание номеров у машин, детектирование лиц, медицинская диагностика и т.д. В данной работе рассматривается одна из развивающихся идей компьютерного зрения – распознавание эмоций человека.

Но что же такое эмоция? Эмоция – это вид психических процессов, отражающий субъективную оценку человеком ситуации, носит исключительно временный характер.

Определение эмоционального состояния человека особенно актуально в вопросе безопасности [2]. В некоторых современных автомобилях имеется система определения состояния человека. Благодаря такой системе исключены случаи вождения в нетрезвом виде, либо случайные аварии, вызванные плохим самочувствием или сонливостью водителя.

Так же распознавание эмоций широко используется для клиентоориентированности. Система позволяет определить заинтересованность клиента при виде определенной рекламы, или потребления продукции. Способствуя дальнейшему развитию товара.

Для определения эффективности подобных алгоритмов была разработана модель нейронной сети, определяющей эмоции человека по изображению.

1 АНАЛИТИЧЕСКИЙ ОБЗОР

1.1 Постановка задачи

Основной целью работы является – разработка нейронной сети для распознавания эмоции человека на основе его выражения лица во входном изображении.

Для определения эмоций, необходимо их сначала классифицировать. Существуют две основополагающие группы эмоций – положительные и отрицательные.

Положительные эмоции – это эмоции, возникающие в ответ на приятные события или ситуации. Помогают уберечься от опасности и принимать решения на основе опыта и знаний. Воспринимаются как приятные для переживания. Оксфордский справочник определяет их как «приятные или желательные ситуативные реакции, отличные от приятных ощущений и не дифференцированного положительного аффекта» [3].

Негативные эмоции – это те эмоции, которые как правило нам не нравятся. Определены как «неприятные или несчастные эмоции, которые вызываются людьми, чтобы выразить негативное влияние на событие или человека».

Так же обозначим особую эмоцию, не относящуюся к вышеперечисленным группам – нейтральная эмоция, которая характеризует состояние человека как обычное, не возбужденное.

И так для достижения цели работы необходимо выполнить следующие задачи:

- Рассмотреть существующие системы определения эмоций человека и выявить их достоинства и недостатки;
- На основе изученного материала определить собственный алгоритм;
- Определение средств реализации, импорт необходимых библиотек;
- Реализация предложенного алгоритма.
- Подведение итогов, результаты.

1.2 Обзор на существующие алгоритмы

На данный момент идея о распознавании эмоций человека с помощью нейронной сети достаточно актуальна. Существует не малое количество подобных алгоритмов, работающих с разными входными данными, такие как изображения, видеопотоки, последовательность кадров в реальном времени [4]. Но действительно хороших и качественных алгоритмов достаточно мало, а методы требуют совершенства.

Подробнее рассмотрим некоторые из них:

1.2.1 FaceReader

Один из самых успешных аналогов – программа FaceReader Noldus [5]. Данная система с точностью в 88% верно определяет эмоции человека по выражению лица (рисунок 1.1). Программа определяет такие эмоции, как «нейтральное», «счастливое», «грустное», «недовольное», «сердитое», «испуганное» и «удивленное». Так же FaceReader способен по лицам людей определить их возраст, пол и расу человека.

Рисунок 1.1 – Интерфейс FaceReader

Достоинства:

- Работает с широким спектром данных, таких как, статичные и динамичные изображения, видео;
- Имеет высокий процент точности распознавания;
- Вывод результата реализован в различных видах, например, диаграммы, гистограммы, накладываемая сетка и др.

Недостатки:

- При определении лиц детей, возрастом до 7 лет, точность весьма сильно падает;
- Определение эмоций может быть неточным при наличии второстепенных объектов (очки, шрамы, пирсинг);
- Не определяет повернутые лица;
- Результаты могут ощутимо разниться от расы человека.

1.2.2 FaceSecurity

Система, разработанная для обработки уникальных баз данных определенных категорий людей. Идентифицирует по эталону образцов из базы (рисунок 1.2). Данная разработка активно применяется для определения лиц в местах преступления по фотографии и видеонаблюдению [6].

Рисунок 1.2 – Интерфейс FaceSecurity

Достоинства:

- Модель работает в реальном времени и отслеживает опознанное лицо на нескольких потоках данных одновременно;
- Помимо автоматического управления имеется и ручное.
- Большая база лиц;

Недостатки:

- Низкая точность определения при бликах и тенях на объекте;
- Определяет исключительно фронтально расположенные лица;
- Распознавание неизвестного лица может быть проблемным из-за отсутствия в базе данных.

1.2.3 MMER_FEASy

Система, основанная на концепции наложения на лицо определенной деформируемой маски (рисунок 1.3). Распознает 5 эмоций, также возраст, пол и расу человека. Программа работает, используя три модуля – MMER_GPU (ядро программы, обеспечивает взаимодействие всей системы), MMER_Lab (классификация определенных признаков изображения) и MMER_Locate (осуществляет поиск лиц на изображении) [7].

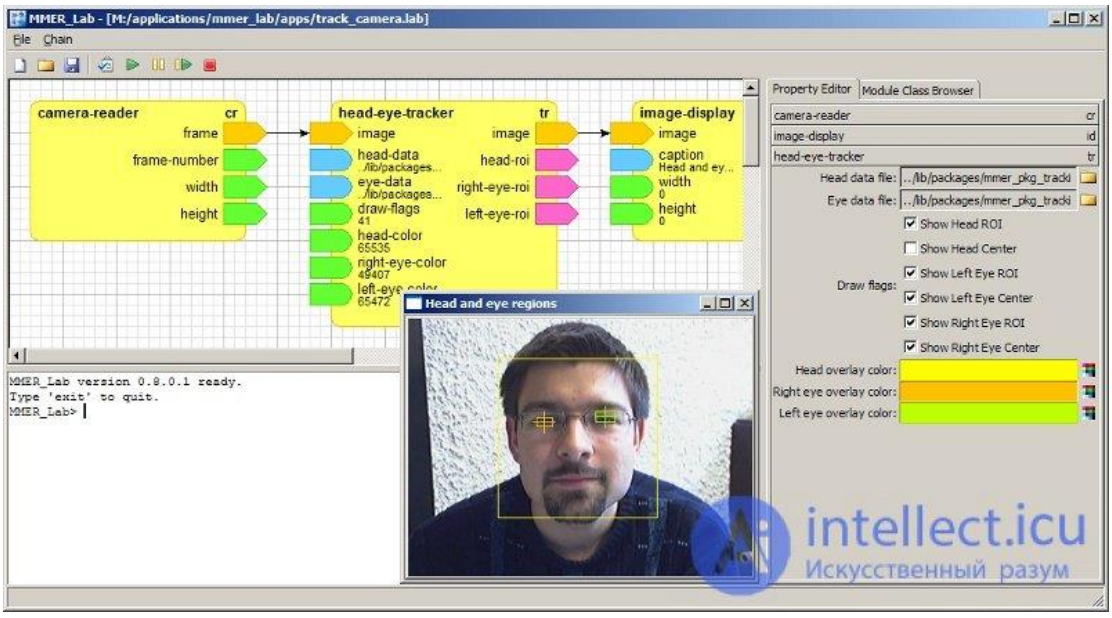

Рисунок 1.3 – Интерфейс MMER_FEASy

Достоинства:

- Имеется возможность подключения к иным программам в роли модуля;
- Возможности распознавания дополнительных признаков человека.

Недостатки:

- Малый охват загружаемых данных;
- Взаимодействие осуществляется только с веб-камеры;
- Низкокачественная выгрузка данных.

Выделив ключевые параметры из обзора аналогов, можно сделать вывод, о том, что весь спектр эмоций разделяется на разное множество классов. Каждый подход имеет определенные преимущества перед остальными, как и некоторые недостатки. А результаты далеки от идеала.

Определить самый эффективный и при этом достаточно точный метод, крайне сложно. Объединяет их только одна деталь - все вышеупомянутые приложения, реализованы в виде сверточных нейронных сетей.

2 ТЕОРЕТИЧЕСКОЕ ВВЕДЕНИЕ

2.1 Нейронные сети

Нейронная сеть – математическая модель, основанная на нейронной сети живого организма [8]. Но в отличие от биологической, машинная нейросеть учится решать только те задачи, которые поставил ей человек.

Если же сравнивать биологический и искусственный нейроны, следует подчеркнуть, что искусственный нейрон является лишь упрощенной концепцией естественного, то есть он только напоминает чем-то биологический нейрон, а не копирует его.

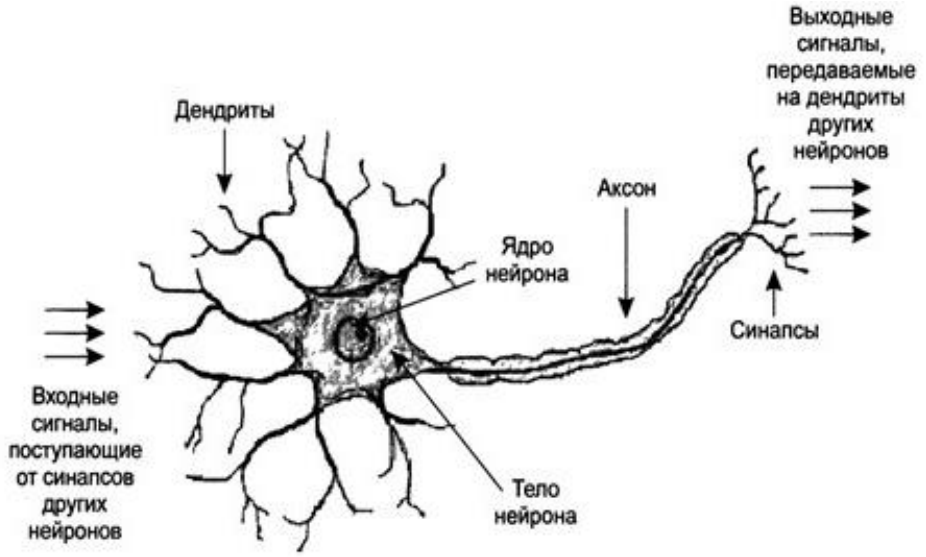

Рисунок 2.1 – Схема биологического нейрона

Нейрон состоит из следующих основных частей: ядро, аксон (отросток, пропускающий импульсы), дендрит (отвечает за прием сигналов) и синапсы (размещаются на конце аксона), контактирующие с другой клеткой. Импульсы от одного нейрона к другому проходят со скоростью до 100 м/с.

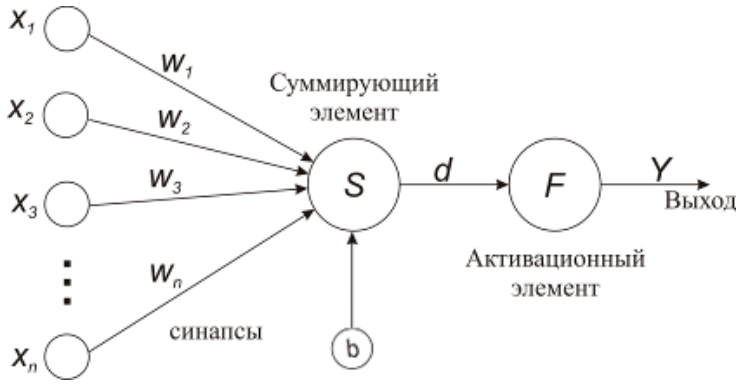

Рисунок 2.2 – Схема искусственного нейрона

В искусственном нейроне (рисунок 2.2) на вход (детрит) подаются сигналы x_i, затем каждый из входов умножается на соответствующий вес W_i (синапсы), b – смещение, значение которого позволяет сдвинуть функцию активации влево или вправо, что может изменить решающее значение для успешного обучения. Функция F – активационная характеристика. И на выходе получаем результат – Y [9]. Исходя из этого получим формулу (1).

$$d = \sum_{i=1}^{n} (x_i\, w_i) + b, \qquad Y = F(d) \tag{1}$$

Подведем итоги сравнения нейронов в таблице (2.1).

Таблица 2.1 – Отличия биологического и искусственного нейронов

Наименование характеристики	Биологический нейрон	Искусственный нейрон
Прием сигнала	Химическая реакция в области синапса при участии усиливающих или тормозящих нейромедиаторов и нейротрансмиттеров	Умножение входных сигналов на весовые коэффициенты
Распространение сигнала	Амплитуда и скорость сигнала зависит от диаметра нервного волокна	Амплитуда и скорость сигналов одинакова в любой точке
Назначение нейронных соединений	Распространение информации, аккумуляция знаний, интеграция, производство белков	Только передача информации
Тип сигналов	Аналоговый	Аналоговый и дискретный

Также отличительной особенностью искусственной нейронной сети является наличие активационной функции.

Функция активации нейрона [10] нужна, чтобы определить при каких входных значениях должен активироваться сигнал нейрона. Если значение Y больше некоторого порогового значения, нейрон считается активированным.

Одним из ключевых этапов обучения нейронной сети является внедрение метода обратного распространения ошибки [11]. Это алгоритм обучения нейронных сетей, который оптимизирует веса между нейронами. Для каждого примера из обучающей выборки вычисляется ошибка, которая "распространяется" обратно по сети, при этом каждому нейрону присваивается вклад в эту ошибку. Для вычисления вклада ошибки каждого нейрона используется производная активационной функции этого нейрона. Затем веса между нейронами корректируются в направлении, противоположном градиенту ошибки (2).

$$w_{ij}(t+1) = w_{ij}(t) - \alpha * \frac{\partial E(k)}{\partial w_{ij}(t)}, \quad (2)$$

где w_{ij} – весовой коэффициент синаптической связи, соединяющий i-ый нейрон слоя n-1 с j-ым нейроном слоя n, α – коэффициент скорости обучения, $0<\alpha<1$, E – среднеквадратичная ошибка сети для одного из k образов, определяется по формуле (3).

$$E = \frac{1}{2}\sum_{l=1}^{m}(y_l - d_l)^2 \quad (3)$$

Основной принцип обучения – если сеть дает неправильный ответ, то веса корректируют так, чтобы уменьшить ошибку.

При обработке и распознавании объектов на изображениях возникает проблема большого объема данных. Чтобы обучить нейронную сеть распознавать, требуется база данных большого количества изображений, каждое

из которых имеет свои собственные размеры в RGB формате. Обычная нейронная сеть не справиться с таким количеством данных даже на супермощных компьютерах. Поэтому требуется нейросеть особой архитектуры для работы с изображениями.

2.2 Сверточные сети

Сверточная нейронная сеть (Convolutional Neural Network) – это нейронная сеть, которая имеет наиболее подходящую архитектуру для обработки изображений, она показала высокую точность в кластеризации и распознании объектов на фотографиях, поэтому имеет большую популярность в решении подобных задач [12].

Сверточная нейронная сеть состоит из следующих видов слоев:

2.2.1 Входной слой

Это начальная часть нейронной сети, предназначена для приема входных данных. Обычно это изображения формата JPG, размером 48x48 пикселей. Такой размер изображения обусловлен оптимизацией алгоритма, при увеличении размера, вычислительная сложность сильно возрастает, что приводит к плохой оптимизации.

Изображение состоит из 3 каналов: красный, синий, зеленый. Итого имеется 3 изображения размеров 48x48 пикселей. Значения каждого определенного пикселя нормализуются в диапазон от 0 до 1, по формуле (4).

$$f(p, min, max) = \frac{p - min}{max - min}, \quad (4)$$

где f – функция нормализации; p – значение конкретного цвета пикселя; min – минимальное значения пикселя – 0; max – максимальное значения пикселя – 255.

2.2.2 Сверточный слой (рисунок 2.3)

Это ключевая операция, во время которой нейросеть «убирает лишнее», например, края или ровные области. Признаки подбираются нейросетью самостоятельно во время распознания на каждом слое.

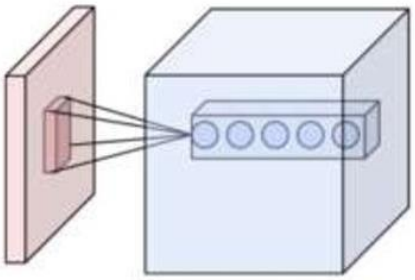

Рисунок 2.3 – Операция в сверточной слое

Свертка представляет собой набор карт признаков (матрицы), у каждой карты имеется свой фильтр. Количество матриц определяется требованиями к задаче. В научных статьях предлагается выбрать соотношение один к двум, иначе говоря, к каждой карте предыдущего слоя привязаны две карты сверточного слоя. Размеры матриц одинаковы и вычисляются по формуле (5).

$$(w, h) = (mW - kW + 1, mH - kH + 1), \quad (5)$$

где *(w, h)* – вычисляемый размер матрицы; mW – ширина матрицы предыдущего слоя; mH – высота матрицы предыдущего слоя; kW – ширина фильтра; kH – высота фильтра.

Фильтр – система разделяемых весов, которая находит конкретные признаки объектов по всей области предыдущей матрицы. В основном размер ядра 5x5.

2.2.3 Слой пулинга (рисунок 2.4)

Он сортирует самые важные признаки из предыдущего слоя, обычно к результату пулинга повторно применяется сверточный слой на несколько циклов, для того чтобы построить иерархию признаков, от самых простых, до наиболее сложных по своей структуре признаков.

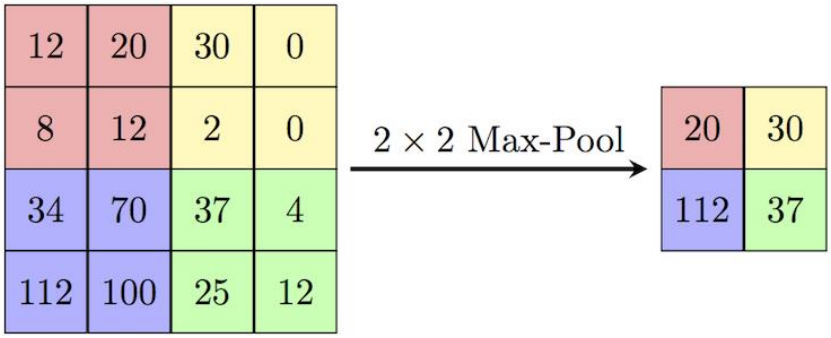

Рисунок 2.4 – Операция в слое пулинга

Основная цель пулинга – уменьшение размерности матриц предыдущего слоя. Реализуется данная операция через формулу (6).

$$x^l = f(a^l * subsample(x^{l-1}) + b^l), \quad (6)$$

где x^l – выход слоя l; $f()$- функция активации; a^l, b^l – коэффициенты сдвига слоя l; *subsample()* – операция пулинга локальных максимальных значений.

2.2.4 Полносвязный слой (рисунок 2.5)

Предназначен для вычисления вероятности принадлежности по уже имеющимся признакам.

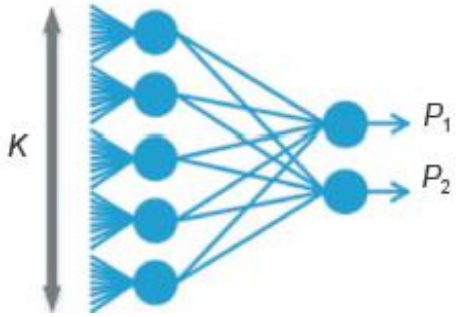

Рисунок 2.5 – Операция в полносвязном слое

Нейроны каждой матрицы предыдущего слоя связаны с одним нейроном скрытого слоя. Информация проходит через каждый нейрон полносвязного слоя, где каждый нейрон разделяет свои выходные данные со всеми остальными нейронами в слое. Таким образом, связи между нейронами в полносвязном слое позволяют нейронной сети извлекать сложные признаки из входных данных Вычисление значений нейрона можно описать следующей формулой (7).

$$x_j^l = f\left(\sum_i x_i^{l-1} * w_{i,j}^{l-1} + b_j^{l-1}\right), \quad (7)$$

где x_i^l – матрица признаков j (выход слоя l); $f()$ – функция активации; b^l – коэффициент сдвига слоя l; $w_{i,j}^l$ – матрица весовых коэффициентов слоя l.

2.2.5 Выходной слой

Этот слой связан со всеми нейронами предыдущего слоя. Количество нейронов зависит от количества распознаваемых классов, в данной работе это значение соответствует количеству распознаваемых эмоций - 7.

2.3 Средства реализации

Обозначив, что из себя представляет сверточная нейронная сеть, следует определить какая функция активации подойдет под решение задачи.

Разберем 4 основных вида функций активации:

2.3.1 Ступенчатая функция (рисунок 2.6)

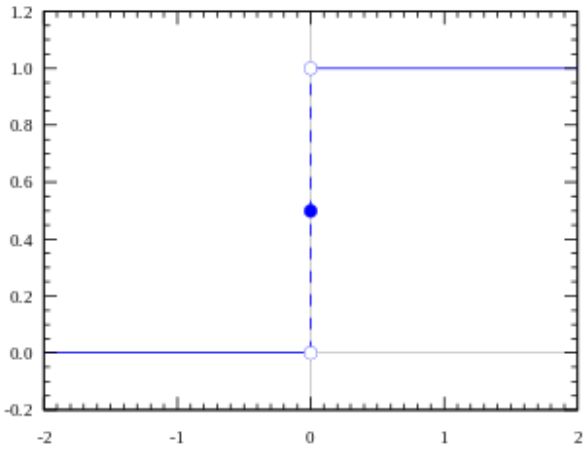

Рисунок 2.6 – Ступенчатая функция активации

Функция принимает значение 1 (активация), когда Y > 0, а в противном случае значение 0 (нет активации). При Y = 0 выходное значение не передается на следующий скрытый слой, а градиент равен нулю, создавая помеху в процессе

обратного распространения. Данная функция является не эффективной, из-за отсутствия промежуточных значений.

2.3.2 Линейная функция (рисунок 2.7)

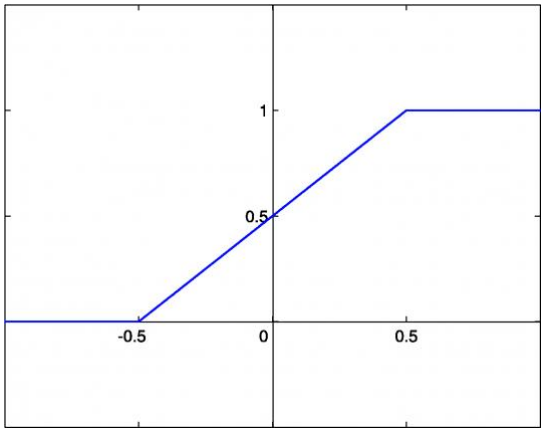

Рисунок 2.7 – Линейная функция активации

Функция представляет собой прямую линию, где входная функция – взвешенная сумма весов и смещений нейронов в слое. Линейная функция решает проблему ступенчатой функции, возвращая плавающие значения.

Из недостатков:

1) производная постоянная на всей области определения;

2) многослойная сеть невозможна, из-за линейной комбинации входных данных, то есть любое количество слоев можно заменить одним слоем.

2.3.3 Сигмоида (рисунок 2.8)

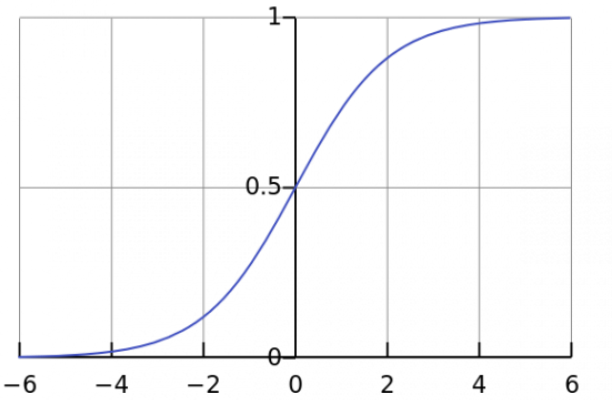

Рисунок 2.8 – Сигмоидная функция активации

Функция нелинейная, следовательно, нейронная сеть может быть многослойной. Сигмоида принимает бесконечное количество значений и нормализована на интервале от 0 до 1.

Недостаток функции – при приближении к границам интервала значений функции происходит исчезновение градиента. Из-за этого процесс обучения замедляется.

2.3.4 Функция ReLu (рисунок 2.9)

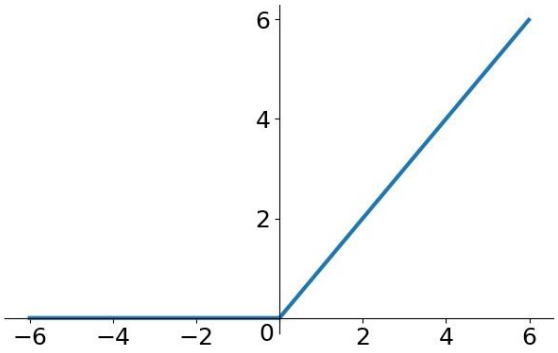

Рисунок 2.9 – Функция активации ReLu

Функция возвращает значение, если оно положительно, в противном случае 0. ReLu является нелинейной, принимает бесконечное количество значений, простое вычисление как функции, так и ее производной. Также разреженность активации, при котором количество включаемых нейронов станет меньше, из-за возвращающихся нулевых значений для отрицательных чисел.

Так как, часть функции представляет собой горизонтальную линию, градиент этой части равен 0, из-за чего веса не будут корректироваться во время градиентного спуска. Поэтому значительная часть нейронной сети пассивна. Однако существует не мало решений такой проблемы, главное избежать нулевой градиент.

Подводя итог, можно определить наиболее эффективную функцию активации – ReLu:

1) функция нелинейна;

2) возможность многослойной сети;

3) хорошая оптимизация.

3 РЕАЛИЗАЦИЯ МОДЕЛИ НЕЙРОННОЙ СЕТИ

3.1 Обнаружение лица на изображении

Распознавание эмоций включает в себя еще одну достаточно сложную задачу – обнаружение лица на изображении. Данной задаче посвящено множество работ, однако она еще далека от решения. Основной проблемой является распознавание человека независимо от ракурса, освещенности и различных индивидуальных изменений черт лица. Среди алгоритмов обнаружения лиц на изображении выделяют следующие основные категории [13]:

- эмпирический метод;
- метод характерных инвариантных признаков;
- распознавание с помощью шаблонов, заданных разработчиком;
- метод обнаружения по внешним признакам, обучающиеся системы.

3.1.1 Эмпирический метод

Основан на наблюдении, экспериментировании и сборе данных. С помощью метода «сверху-вниз», предполагающего создание алгоритма сравнения пикселей, вычисления контуров, локализацию ключевых точек и другие. Такой метод является попыткой формализовать эмпирические знания о том, как выглядит человеческое лицо на изображениях и чем руководствовался бы человек при принятии решения: видит ли он лицо или нет. Однако это может быть затруднительно и неэффективно.

Среди основных правил выделяется:

- центральная часть лица имеет однородную яркость и цвет;
- разница в яркости между центральной частью и верхней частью лица значительна;
- лицо содержит в себе симметрично расположенные глаза, нос, рот, которые отличаются по яркости относительно остальной части лица.

Метод сильного упрощения изображения для уменьшения вычислительных операций подвергает изображение сильному сжатию (рисунок 3.1). Благодаря подобным изменениям, определение основных параметров эмпирического метода становится гораздо проще.

Рисунок 3.1 – Метод сжатия изображения

Следующим примером является метод гистограмм для определения области лица на изображении. Такой метод заключается в построении прямых гистограмм в наиболее подходящих зонах (рисунок 3.2). Главным преимуществом данного подхода является малое требование к вычислительной мощности процессора.

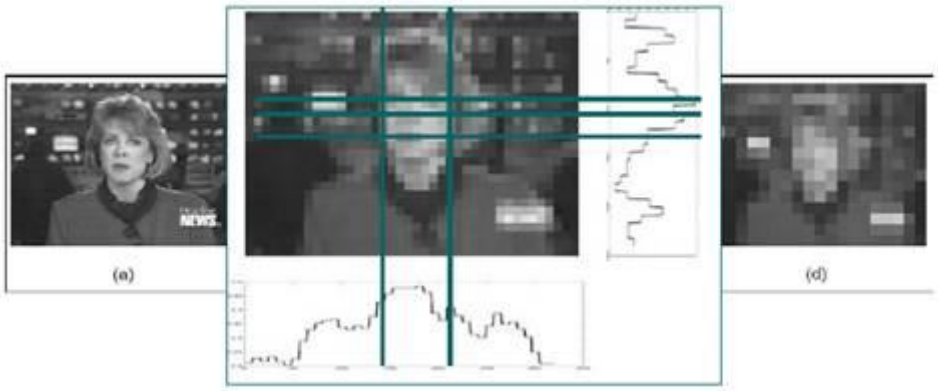

Рисунок 3.2 – Построение диаграмм на изображении

Вышеупомянутые методы имеют достаточно неплохие показатели по выявлению лица на цифровом изображении, но только на однородном фоне. При наличии небольшого наклона головы или посторонних объектов точность значительно снижается. В связи с чем они стали непригодны.

3.1.2 Метод характерных инвариантных признаков

Основная идея метода заключается в том, что для каждого лица можно выделить некоторые уникальные характеристики, которые не меняются при изменении положения, масштаба либо освещения, например, глаза, нос, рот, брови.

Выделим основные параметры данного метода:

- детектирование на изображении ключевых признаков;
- дескриптор ключевых точек;
- объединение всех найденных инвариантных признаков и их последующее сопоставление.

Одним из примеров вышеупомянутого подхода является метод обнаружения лиц в сложных сценах. Основная суть заключается в определении правильных геометрически-расположенных черт лица, реализуется это с помощью инструмента гауссовского производного фильтра с множеством масштабов и ориентаций для поиска. Используя случайный перебор и пороговый фильтр, производится соответствие выявленных признаков и их взаимного расположения (рисунок 3.3). Завершающим шагом происходит объединение всех найденных признаков и последующая оценка комбинирования при помощи байесовской сети.

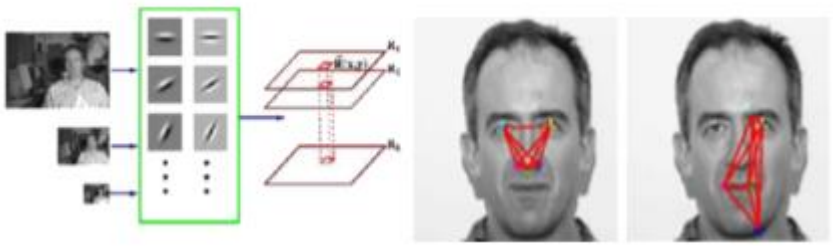

Рисунок 3.3 – Метод группировки признаков

Преимущественным отличием данного класса детектирования лиц является возможность распознавать лицо в разных наклонах. Основа подходов – эмпирика, сводит задачу к нескольким простым проверкам. Однако, процессы, происходящие в мозгу человека во время классификации изображения далеко не

полностью изучены, из чего можно сделать вывод, что инструментарий эмпирических знаний не может быть использован с тем же функционалом, что человеческий мозг. Проблема шумов или засветки в данных подходах не решена и крайне сильно влияет на результативность.

3.1.3 Распознавание с помощью шаблонов, заданных разработчиком

Для распознавания задается стандартный шаблон (образ изображения) путем описывания основных свойств определенных параметров лица и их взаимного расположения. Такой процесс состоит из, предварительной обработки изображения, выделение основных черт лица путем проверки каждой области изображения на соответствие заданному шаблону, предоставление результата сравнения.

Метод трехмерных форм для обнаружения лица – использование шаблона для анализа геометрических характеристик лица, описанного как трехмерная модель, которая может быть повернута и масштабирована для анализа специфических признаков лица (рисунок 3.4).

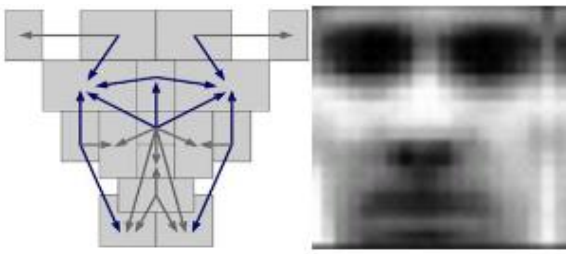

Рисунок 3.4 – Шаблон трехмерной формы

Модель распределения опорных точек – структура деформирующихся объектов (рисунок 3.5). Основана на идее выбора определенного количества опорных точек, которые являются наиболее репрезентативными для заданной выборки. Данная система классификации является компактной и достаточно точной.

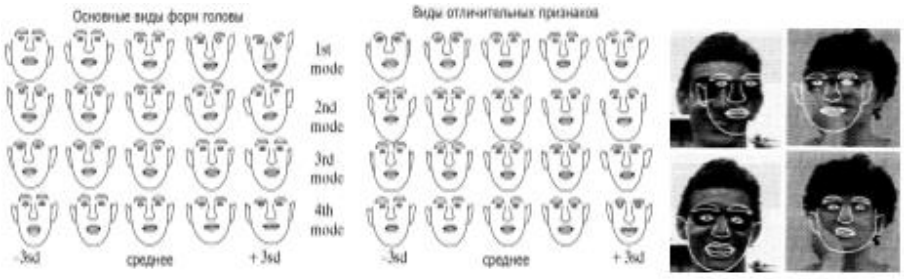

Рисунок 3.5 – Модель распределения опорных точек

Простота реализации является главным преимуществом использования шаблонов, но при этом существует необходимость постоянно калибровать шаблоны из-за сложности определения лиц при наличии малейших посторонних дефектов.

3.1.4 Метод обнаружения по внешним признакам, обучающиеся системы.

Обнаружение лица на изображении заключается в построении математической модели лица и переборе всех прямоугольных фрагментов для проверки отношения к одному из классов: лицо / не лицо (рисунок 3.6).

Рисунок 3.6 –Разбиения изображения на прямоугольные фрагменты

В связи с высокими вычислительными требованиями применяются различные методы для сокращения количества рассматриваемых объектов.

Обучение производится на базе подготовленных изображений лиц и любых других объектов, не имеющих отношения к лицам.

Наиболее известным и популярным методом является метод нейронных сетей [14]. Суть которого состоит в нахождении коэффициентов связей между нейронами. Для снижения размерности пространства признаков и перехода к базису пространства, используется метод опорных векторов (рисунок 3.7).

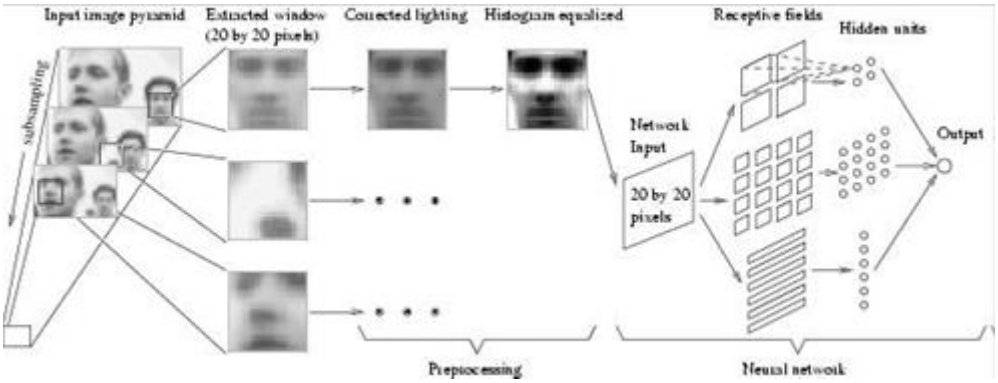

Рисунок 3.7 – Обучение системы при помощи нейронной сети

Основным принципом является – минимизация ошибки классификации на тренировочном наборе.

3.1.5 Практическое применение

Самый перспективный и популярный метод на сегодняшний день – метод Виолы-Джонса [15]. Данный метод имеет один из самых больших процентов обнаружения лиц и обладает высокой производительностью. Основан он на следующих принципах:

- изображение переводится в интегральную форму, для быстрого вычисления объектов;
- определение искомого объекта осуществляется за счет каскадов Хаара;
- каждый найденный признак подается на вход классификатора, для определения бинарного значения;
- использование каскадов признаков для быстрого отбрасывания окон, не имеющих лиц.

Система принимает на вход изображение в виде двумерной матрицы, в которой пиксели имеют значение от 0 до 255^n, где n – количество каналов изображения.

Определение лиц производится через поиск в активной части изображения прямоугольными признаками (рисунок 3.8), с помощью которых описывается найденное лицо, данная функция определяется по формуле (8).

$$Rect_i = \{x, y, w, h, a\}, \quad (8)$$

где x, y – координаты центра i-го прямоугольника, w – ширина, h – высота, a – угол наклона прямоугольника к вертикальной оси.

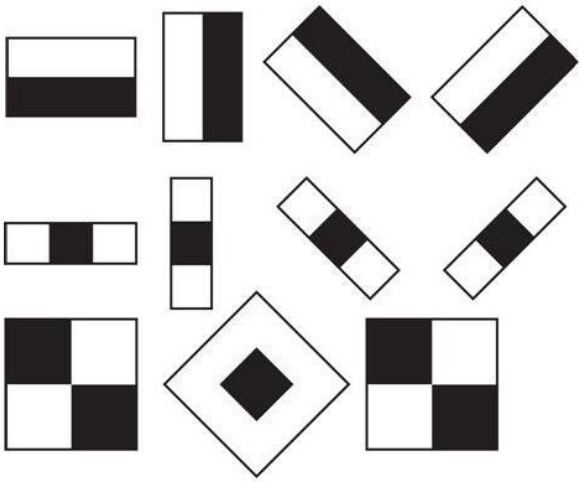

Рисунок 3.8 – Примитивы Хаара

Следующим шагом производится расчет суммарной яркости прямоугольника за счет интегрального представления изображения [16]. Элементы такой матрица рассчитываются по формуле (9).

$$L(x, y) = \sum_{i=0, j=0}^{i \le x \le y} I(i, j), \quad (9)$$

где $I\ (i, j)$ – яркость пикселя изображения. В каждом элементе матрица хранится сумма интенсивностей всех пикселей до x, y.

Для использования каскадов Хаара [17] требуется импортировать соответствующий под задачу каскад, в виде xml документа.

Прежде чем производить поиск лиц, требуется обработать изображение, а именно перевести его в черно-белый формат. Изображение станет одноканальным, и его обработка будет более оптимальной и точной.

Применение каскадов реализовано при помощи библиотеки OpenCV. Используя команду cv2::CascadeClassifier, мы определяем классификатор, указывая к нему путь.

Для детектирования лиц на изображении используется метод cv2::CascadeClassifier::detectMultiScale, который возвращает граничные прямоугольники для обнаружения лиц. Реализация отражена в Приложении 1.

3.2 Данные для обучения нейронной сети

Для обучения сверточных нейронных сетей требуются большие объемы данных. Чем сложней нейронная сеть, тем обширней и разнообразней должна быть обучающая выборка.

Один из самых больших и распространенных наборов данных является – FER2013 [18]. Он состоит из 35887 изображений лиц в серых оттенках, размер 48x48 пикселей (рисунок 3.9). Датасет состоит из следующих категорий:

- «счастье» (Happy) 8989 изображений;
- «нейтральный» (Neutral) 6198 изображений;
- «грусть» (Sad) 6077 изображений;
- «страх» (Fear) 5121 изображение;
- «злость» (Angry) 4953 изображения;
- «удивление» (Surprised) 4002 изображения;
- «отвращение» (Disgusted) 547 изображений.

Рисунок 3.9 – Примеры изображений набора FER2013

Для полноценного обучения сети требуется выделить два основных этапа: тренировка и тестирование. Для обоих этапов требуется свой набор изображений, поэтому набор данных будет разделен на 2 части, с отношением 80% на 20%, train и test соответственно.

Помимо самих изображений, потребуется обучающая выборка в формате векторов. Файл icml_face_data.csv (рисунок 3.10) организован в виде:

1) Класс эмоций в виде числа;
2) Предназначенное использование (Training, PrivateTest или PublicTest);
3) Значения пикселей в виде вектора.

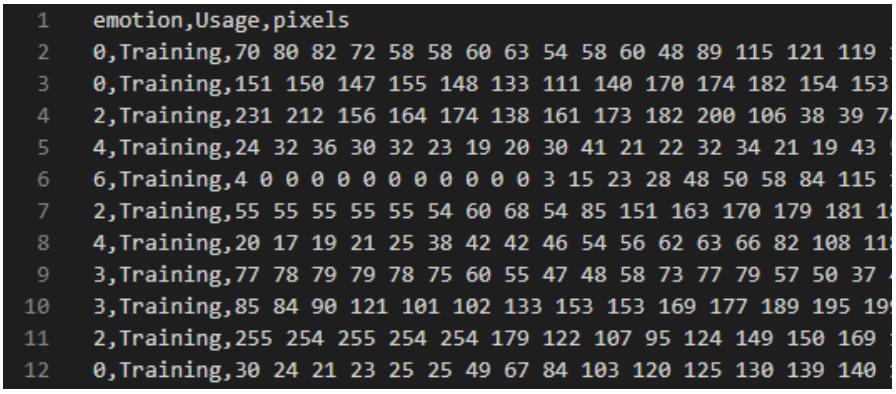

Рисунок 3.10 – Пример структуры csv файла

Пример использования и интеграции набора данных в программный код представлен в Приложении 2.

3.3 Разработка уникальной модели

В данной работе для решения задачи распознавания эмоций используется сверточная нейронная сеть (CNN). В первую очередь требуется разработать архитектуру модели и определить необходимые слои.

Входной слой, будет принимать размеры 48x48x1 (соответствующий изображениям из датасета). Выходной слой же будет являться вектором из семи значений, каждое представляет вероятность принадлежности к одному определенному классу эмоций. Как итог, исходное изображение будет относиться к тому классу, который имеет максимальное значение вероятности.

Каждая версия нейронной сети будет пронумерована в порядке разработки, первая версия имеет имя – CNN_v1.

Модель будет состоять из набора следующих слоев:

1) Conv2D – слой двумерной свертки, создающий ядро, которое «наматывается» на входные слои, тем самым помогая создавать тензор выходных данных;

2) MaxPool2D – слой подвыборки с группой уплотнения 2x2, основная суть в уменьшении размерности полученных карт признаков;
3) Flatten – используется для преобразования многомерных тензоров данных в одномерный вектор, который может быть передан в полносвязный слой для дальнейшей обработки;
4) Dense – добавление полносвязного слоя, в котором каждый нейрон в слое принимает входные данные от всех нейронов предыдущего слоя и вычисляет свою линейную комбинацию с весами, которые обучаются в процессе обучения [19].

Архитектура слоев с параметрами для первой модели представлена в таблице 3.1.

Таблица 3.1 – Слои CNN_v1

Слой	Выходное значение	Параметры
conv2d (Conv2D)	(None, 46, 46, 32)	320
max_pooling2d (MaxPooling2D)	(None, 23, 23, 32)	0
conv2d_1 (Conv2D)	(None, 21, 21, 64)	18 496
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 64)	36 928
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 64)	262 208
dense_1 (Dense)	(None, 7)	455
Total params: 318,407 Trainable params: 318,407 Non-trainable params: 0		

В качестве оптимизатора использовался алгоритм Adam [20]. Так как помимо адаптации скорости обучения параметров на основе среднего значения, Adam еще использует среднее значение вторых моментов градиентов. Что является самым эффективным способом оптимизации обучения. Для функции

потерь была выбрана кросс-энтропия. Основная метрика – accuracy (точность).

Для моделирования обучения необходимо задать два важных начальных параметра. Первый параметр – эпоха, это одна итерация в процессе обучения, состоящая из всех примеров обучающего множества. Одна эпоха обозначает, что весь датасет прошел через модель нейронной сети в двух направлениях один раз. Количество эпох будет различаться в зависимости от метода тестирования. Следующий параметр - размер батчей, это деление данных на пакеты. Его значение можно сделать постоянным, поэтому во всех версиях нейронной сети batch_size = 32.

Обучение и тестирование будет проводиться в каждой модели одинаковым образом, единственным отличием является количество эпох и добавление нескольких дополнительных параметров. Реализация в виде программного кода предоставлена на рисунке 3.11:

```
# Основные параметры
epochs = 50
epochs_range = range(epochs)
batch_size = 32

# Запуск итераций
history = model.fit(train_images, train_labels,
                    validation_data=(valid_images, valid_labels),

                    epochs=epochs,
                    batch_size=batch_size)

print('\t')
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc, '\n')

# Присваивание параметров тестирования
loss = history.history['loss']
acc = history.history['accuracy']
valid_loss = history.history['val_loss']
valid_acc = history.history['val_accuracy']

# Построение гистрограмм
plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, valid_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')
plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, valid_loss, label='Validation Loss')
plt.legend(loc='lower left')
plt.title('Training and Validation Loss')
plt.show()
```

Рисунок 3.11 – Программная реализация тестирования

После проведения тестирования на 50 эпох для первой модели получаем результаты в виде гистограмм (рисунок 3.12), а также точность, которая вычисляется на тестовых данных. Скорость прохождения одной эпохи

составляет примерно 35 секунды, что является достаточно быстрым результатом.

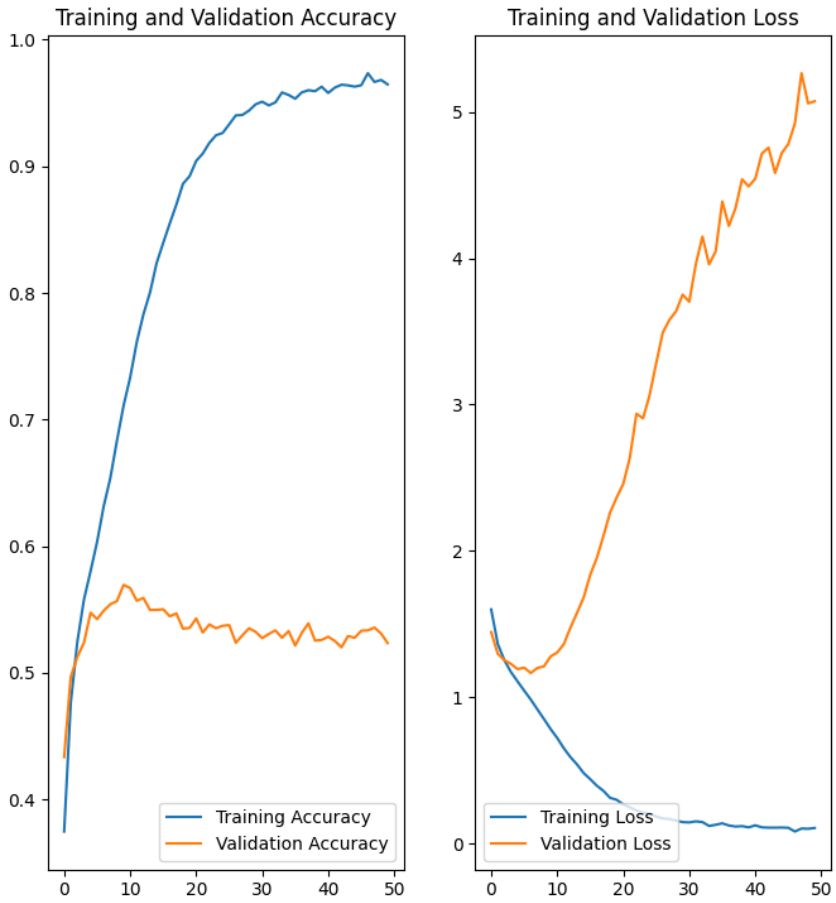

Рисунок 3.12 – Результативные гистограммы CNN_v1

Исходя из результатов можно определить, что сеть достигает 51,65% точности с минимальным значением потерь 1,1661 на тестовой выборке. Как можно увидеть на графике, на последующих эпохах точность не меняется, а иногда даже снижается, что говорит о плохой эффективности сети. Вывод – модель требует доработок.

Для улучшения точности нейронной сети, была разработана следующая модель – CNN_v2 ее структура указана в таблице 3.2, среди основных изменений:

1) Добавление новых слоев свертки (Conv2D), с дополнительными параметрами strides = (1, 1) – задает шаг дискретизации для ядра свертки в конкретной оси свертки. Он определяет, на сколько пикселей смещается ядро свертки за одну итерацию в каждом направлении;

2) Новые слои ZeroPadding2D – слой, добавляющий нулевые значения по контуру входного изображения, параметр padding = (1,1) является постоянным;
3) Еще один полносвязный слой (Dense) размерность = 32, функция активации ReLu;
4) После каждого полносвязного слоя будет происходить блокировка определенного процента нейронов предыдущего слоя, для оптимизации (Dropout (0.3)).

Таблица 3.2 - Слои CNN_v2

Слой	Выходное значение	Параметры
conv2d (Conv2D)	(None, 46, 46, 16)	160
conv2d_1 (Conv2D)	(None, 44, 44, 16)	2 320
zero_padding2d (ZeroPadding2D)	(None, 46, 46, 16)	0
conv2d_2 (Conv2D)	(None, 44, 44, 32)	4 640
conv2d_3 (Conv2D)	(None, 42, 42, 32)	9 248
conv2d_4 (Conv2D)	(None, 40, 40, 64)	18 496
conv2d_5 (Conv2D)	(None, 38, 38, 64)	36 928
max_pooling2d (MaxPooling2D)	(None, 19, 19, 64)	0
zero_padding2d_1 (ZeroPadding2D)	(None, 21, 21, 64)	0
conv2d_6 (Conv2D)	(None, 19, 19, 128)	73 856
conv2d_7 (Conv2D)	(None, 18, 18, 128)	65 664
conv2d_8 (Conv2D)	(None, 16, 16, 128)	147 584
conv2d_9 (Conv2D)	(None, 15, 15, 128)	65 664
flatten (Flatten)	(None, 28800)	0

dense (Dense)	(None, 64)	1 843 264
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2 080
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 7)	231
Total params: 2,270,135 Trainable params: 2,270,135 Non-trainable params: 0		

Обучение проводилось на 50 эпох, среднее время прохождения одной эпохи составило 400 секунд. Столь высокая разница скоростей заключается в большом количестве параметров – 2 270 135. На выходе получаем также две гистограммы точности и ошибок (рисунок 3.13).

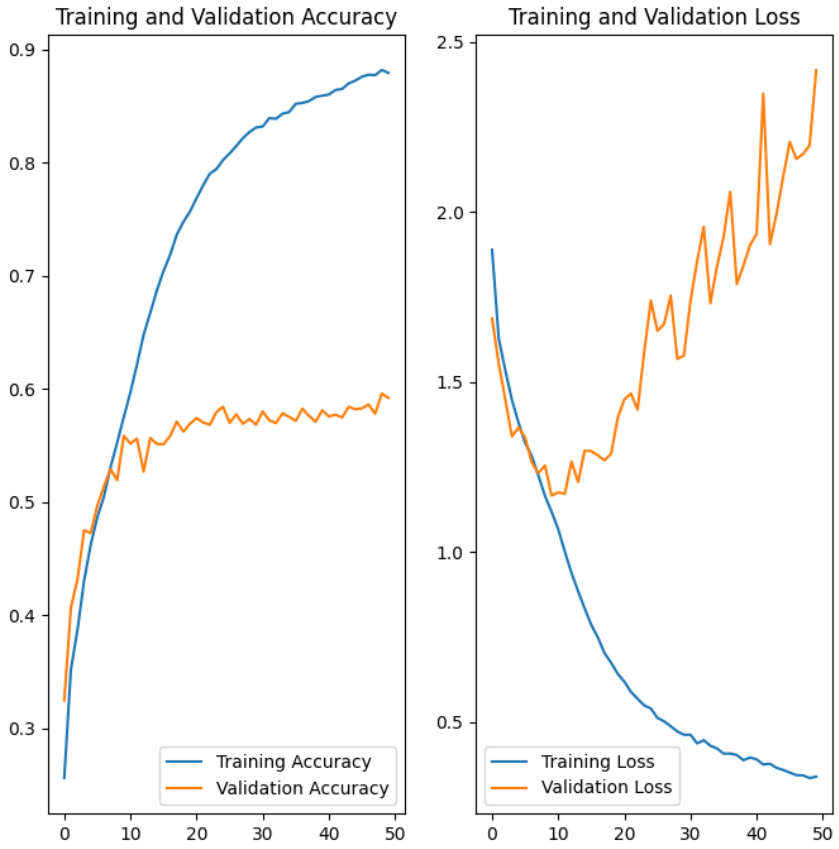

Рисунок 3.13 – Результативные гистограммы CNN_v2

По результатам обучения, общая точность достигла 58,42% с минимальным значением потерь 1,2317. Значение точности повысилось на 7%, но это все равно

является неудовлетворительным результатом. Начиная с 20 эпохи сеть перестала прогрессировать, а значение потерь только повышалось, это связано с проблемой переобучения, это можно заметить по значениям тренировочной и тестовой точностей, первое значение значительно прогрессирует, второе же застывает на определенном отрезке значений.

Используя механизмы регуляризации можно избежать проблему переобучения, одним из таких механизмов является метод Dropout, его суть заключается в блокировке заданной части сигналов из предыдущего слоя, тем самым предотвращая чрезмерную адаптацию нейронов друг к другу. Важно отметить, что данный метод активируется только при обучении, при использовании уже обученной сети dropout никак не функционирует.

Следующей проблемой является внутренний ковариантный сдвиг – во время обучения модели распределение данных меняется во внутренних слоях, из-за этого модель обучается гораздо медленней. Для такой проблемы существует решение в виде слоя BatchNormalization, это метод, повышающий производительность и стабилизацию работы модели. Основная суть состоит в том, что некоторые слои принимают на вход данные, предварительно обработанные и имеющие нулевое математическое ожидание [21].

В разработке модели третьей версии необходимо:

1) Уменьшить количество параметров для слоев, для этого в конец каждого модуля добавим слой подвыборки (MaxPooling2D) размерностью (2, 2) и сразу после метод dropout (0.25). Для половины слоев двумерной свертки добавим параметр padding = 'same';
2) Ускорить процесс обучения, с помощью нового слоя – BatchNormalization. Метод размещается в каждом модуле сразу после первичной свертки, а также после каждого полносвязного слоя, так как именно эти методы обрабатывают наибольшее количество параметров. Собственных параметров не имеет.

После проделанных манипуляций получаем структуру слоев в таблице из Приложения 3.

Модель CNN_v3 обучалась в течение 50 эпох, среднее выполнение одной эпохи составило 100 секунд, что в 4 раза быстрее предыдущей версии. Точность модели составила 69,87% с минимальным значением потерь 0,9135 (рисунок 3.14).

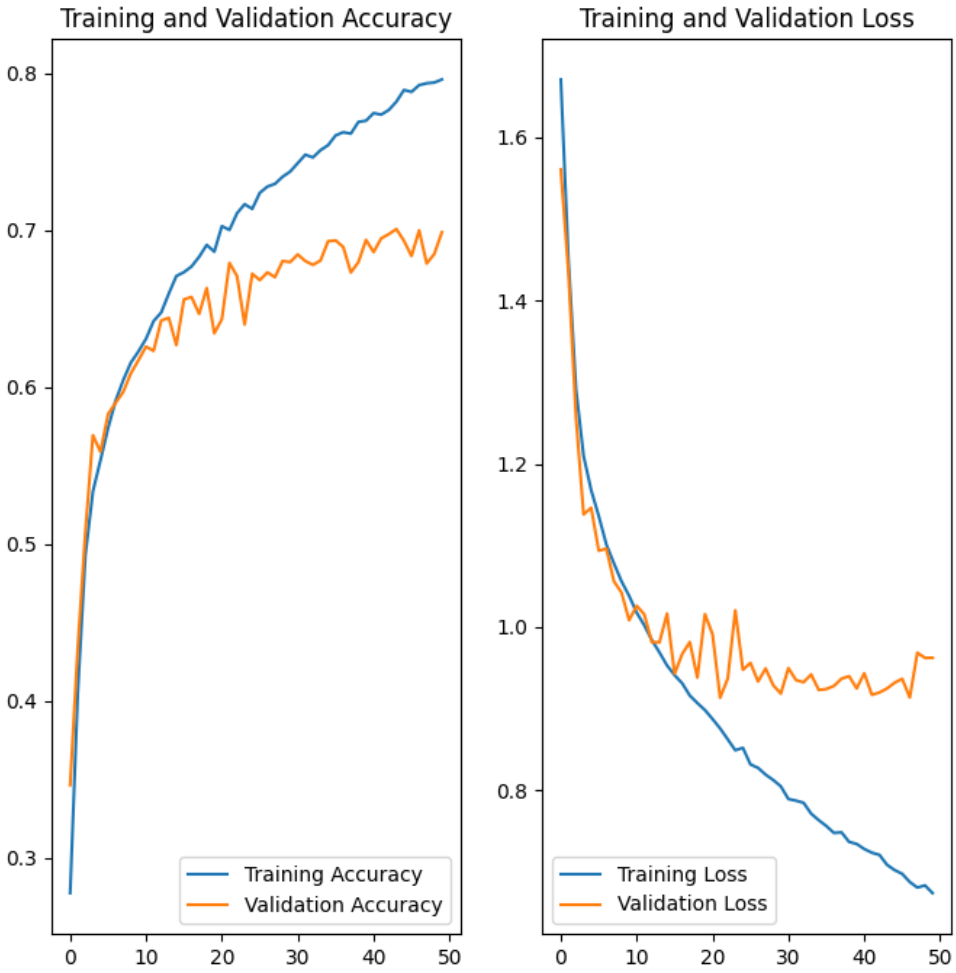

Рисунок 3.14 – Результативные гистограммы CNN_v3

Несмотря на то, что различия значений тестирования и тренировки стали менее значимы, проблема переобучения все еще актуальна. Для ее решения требуется добавить L2 регуляризацию, ее суть заключается в изменении первоначальной функции, добавляя «штраф» на большие весовые коэффициенты, чтобы направлять линию наилучшего соответствия от основной тенденции. Для этого импортируется соответствующая библиотека l2 из keras. regularizers [22].

Итоговые значения модели приблизились к желаемому результату, но потенциал разрабатываемой модели позволяет добиться большего процента. Для выполнения этой задачи была разработана завершающая версия модели, которая содержит следующие изменения:

1) Использование метода SeparableConv2D, который предназначен для свертки входного тензора. Данный метод разделяет процесс свертки на две фазы: сначала выполняется глубинное свертывание, обрабатывающее каждый канал входного тензора независимо от остальных. Затем выполняется поперечно-связанная свертка, для объединения всех каналов и создания выходного тензора.
2) Переменная regularization, которая вычисляется с помощью метода l2, с начальным значением 0.01. Данная переменная используется в параметре kernel_regularizer метода Conv2D первого модуля, далее в каждом слое SeparableConv2D.
3) Метод GlobalAveragePooling2D, один из завершающих слоев, используется для сокращения размерности тензора признаков, что уменьшает количество параметров для обучения и ускоряет процесс обучения. Данный метод суммирует значения всех признаков каждого канала изображения, а затем усредняет их, получая таким образом одно число для каждого канала, которое затем передается на следующий уровень, тем самым упрощая архитектуру, не уменьшая при этом качество ее работы.
4) Разделение слоев на переменные и их использование в методе layers.add, позволяет более гибко настраивать и изменять архитектуру нейронной сети. Полное управление весами: разделение слоев на переменные позволяет управлять весами каждого слоя независимо, благодаря чему можно переиспользовать определенный слой в разных частях сети с разными весами.
5) Новые изменения сильно повлияют на производительность сети, поэтому требуется оптимизировать модель. Один из способов – замена метода

Dense другими типами слоев, которые позволяют более гибко настраивать параметры модели и улучшать качество ее предсказаний.

Структура слоев заключительной версии сверточной нейронной сети CNN_v4 указана в Приложении 4.

Тестовое обучение проводилось в количестве 70 эпох, среднее выполнение одной эпохи 110 секунд. Точность на тестировании составила 80,58% при значении потерь 0,7601 (рисунок 3.15).

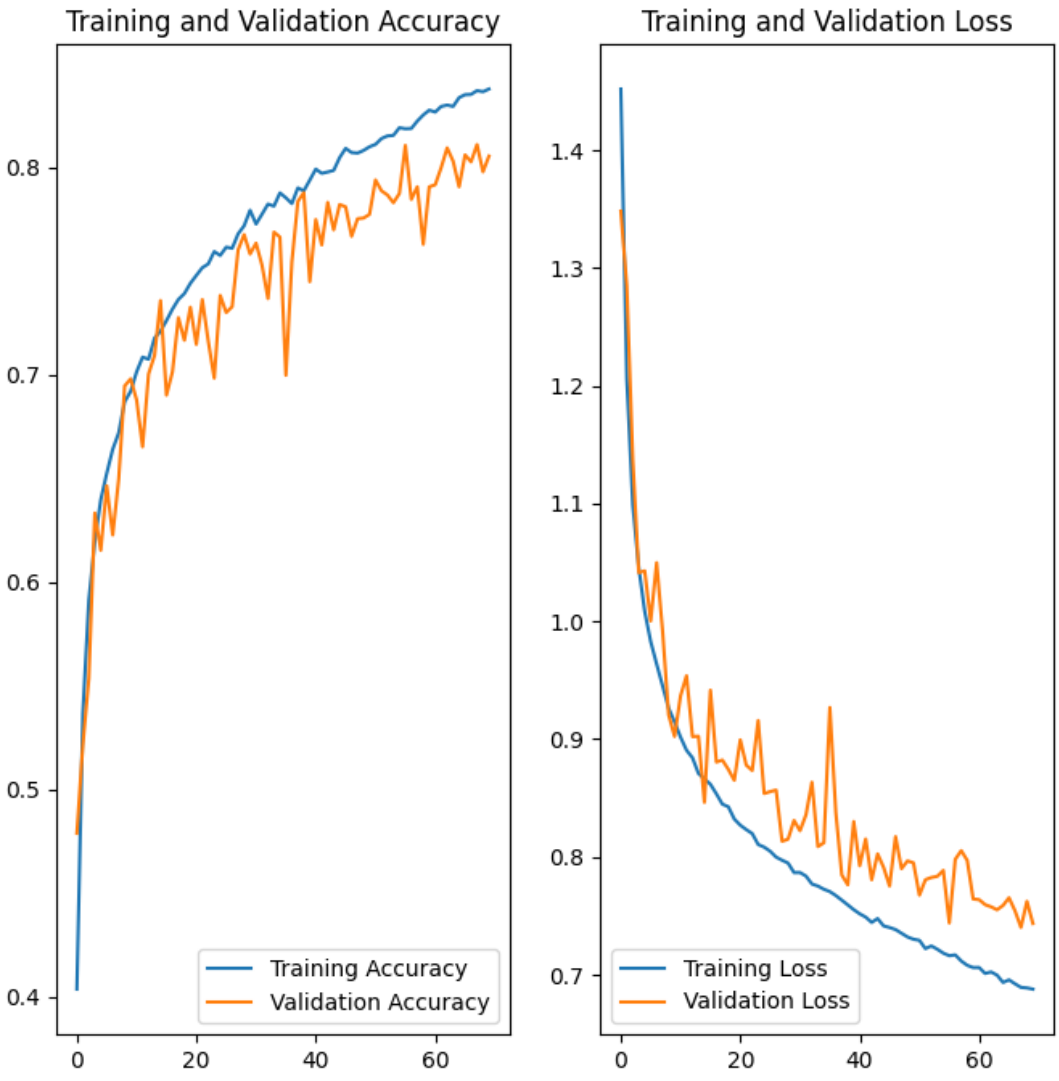

Рисунок 3.15 – Результативные гистограммы CNN_v4

Как видно по графикам кривые тренировки и тестирования практически совпадают, что говорит о решенной проблеме переобучения. Итоговая точность достигла хорошего результата, следовательно, с данной моделью можно проводить дальнейшее тестирование.

В первую очередь необходимо реализовать запись, полученных при обучении весов. Форматом файла был выбран HDF5, из-за своих преимуществ в: быстром доступе к данным через API, поддержке больших объемов данных, автоматическом сжатии данных и возможности параллельной записи. Для сохранения весов используется обратный вызов (callbacks). Программная реализация продемонстрирована на рисунке 3.16.

```
# callbacks
log_file_path = path + '_' + dataset_name + '_emotion_training.log'
csv_logger = CSVLogger(log_file_path, append=False)
early_stop = EarlyStopping('val_loss', patience=patience)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1,
                              patience=patience)
trained_models_path = path + dataset_name + '_Neurons_model_CNN_v4'
model_names = trained_models_path + '.{epoch:02d}-{val_accuracy:.2f}.hdf5'
model_checkpoint = ModelCheckpoint(model_names, 'val_loss',
                                   save_best_only=True)
callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr]
```

Рисунок 3.16 – Реализация записи весов

Запись осуществляется каждую эпоху с соответственным результатом точности. Затем среди всех файлов выбирается тот, чья точность достигла наивысшего результата. Таким образом был выбран файл 68 эпохи, точность которой составила 81%.

4 ТЕСТИРОВАНИЕ

4.1 Метод тестирования

Работа с изображением реализована с помощью библиотек OpenCV и NumPy. Обнаружение лица на изображении производиться с помощью каскадных классификаторов Хаара, в виде файла xml формата [23]. Программная реализация описана в Приложении 1. Для тестирования обученной модели, данный код будет модернизироваться.

Для начала требуется загрузить модель (рисунок 4.1), с использованием метода load_model из библиотеки keras.models. Так же присвоить значения искомых классов эмоций:

```
# Загрузка обученной модели по определению эмоций
model_path = "D:/.Labs/VKR/Neural_Emotions/Project/Project_fer2013_Neurons_model_CNN_v4.68-0.78.hdf5"
emotion_data = load_model(model_path, compile=False)
# Присваивание имен классам
labels = {0: 'злость', 1: 'отвращение', 2: 'страх', 3: 'счастье',
          4: 'грусть', 5: 'удивление', 6: 'нейтрально'}
```

Рисунок 4.1 – Интеграция нейронной сети

После выполнения поиска лица методом Хаара, для улучшения точности классификации необходимо преобразовать изображение, выполнив ряд действий (рисунок 4.2):

1) Присваивание искомой зоны отдельной переменной (лицо);
2) Масштабирование изображения до размера, заданного моделью (48x48x1);
3) Предварительная обработка входных данных;
4) Добавление новых размерностей в массив изображения;
5) Классификация.

```
size = len(detect_image)
if size != 0:
    for (a, b, w, h) in detect_image:
        # Выделяем лицо квадратом
        cv2.rectangle(image, (a, b),
                      (a + h, b + w), # Определение размеров квадрата
                      (0, 0, 255), 5) # Определение цвета и толщины квадрата

        # 1
        emotion_image = image_gray[b:b+h, a:a+w]
        # 2
        emotion_image = cv2.resize(emotion_image, (emotion_data.input_shape[1:3]))
        # 3
        emotion_image = first_input(emotion_image, True)
        # 4
        emotion_image = np.expand_dims(emotion_image, 0)
        emotion_image = np.expand_dims(emotion_image, -1)
        # 5
        classif_emotion = np.argmax(emotion_data.predict(emotion_image))
        emotion_text = labels[classif_emotion]
        draw_text((a, b, w, h), image, emotion_text, (0, 0, 255), 0, -15, 2, 2)
```

Рисунок 4.2 – Обработка изображения

Функция first_input из третьего пункта имеет следующую структуру (рисунок 4.3):

```
def first_input(x, v2=True):
    # массив приводится к типу float, уменьшая объем памяти
    x = x.astype('float32')
    # Нормализация значений в диапазон [0: 1]
    x = x / 255.0
    if v2:
        # Дополнительное масштабирование в диапазон [-1: 1]
        x = x - 0.5
        x = x * 2.0
    return x
```

Рисунок 4.3 – Функция обработки входных данных

В четвертом пункте метод expand_dims используется для добавления новой размерности в массив изображения. Например, при значении 0, происходит преобразование изображения размером 32x32x3 в массив размером 1x32x32x3. Это требуется для подачи изображения в модель глубокого обучения, которая ожидает массив изображений заданного размера, включая размерность пакета.

Метод argmax из пункта №5, используется для поиска индекса элемента с максимальным значением в массиве (обычно вектор вероятностей). Например, если мы имеем вектор вероятностей [0.1, 0.5, 0.4], то метод np.argmax([0.1, 0.5, 0.4]) вернет индекс 1, так как 0.5 - это максимальное значение вектора. В контексте классификации изображений, когда модель выдает вероятностное распределение по нескольким классам, метод argmax используется для выбора предсказанного класса для каждого изображения.

Затем полученное изображение передается на вход к классификатору модели. В результате чего, исходное изображение получает тот класс эмоций, который имеет максимальное значение среди всех классов. В заключающем этапе обработки производится визуализация результатов: вокруг найденного лица отображается красная рамка, сверху которой, соответствующая эмоция (рисунок 4.4).

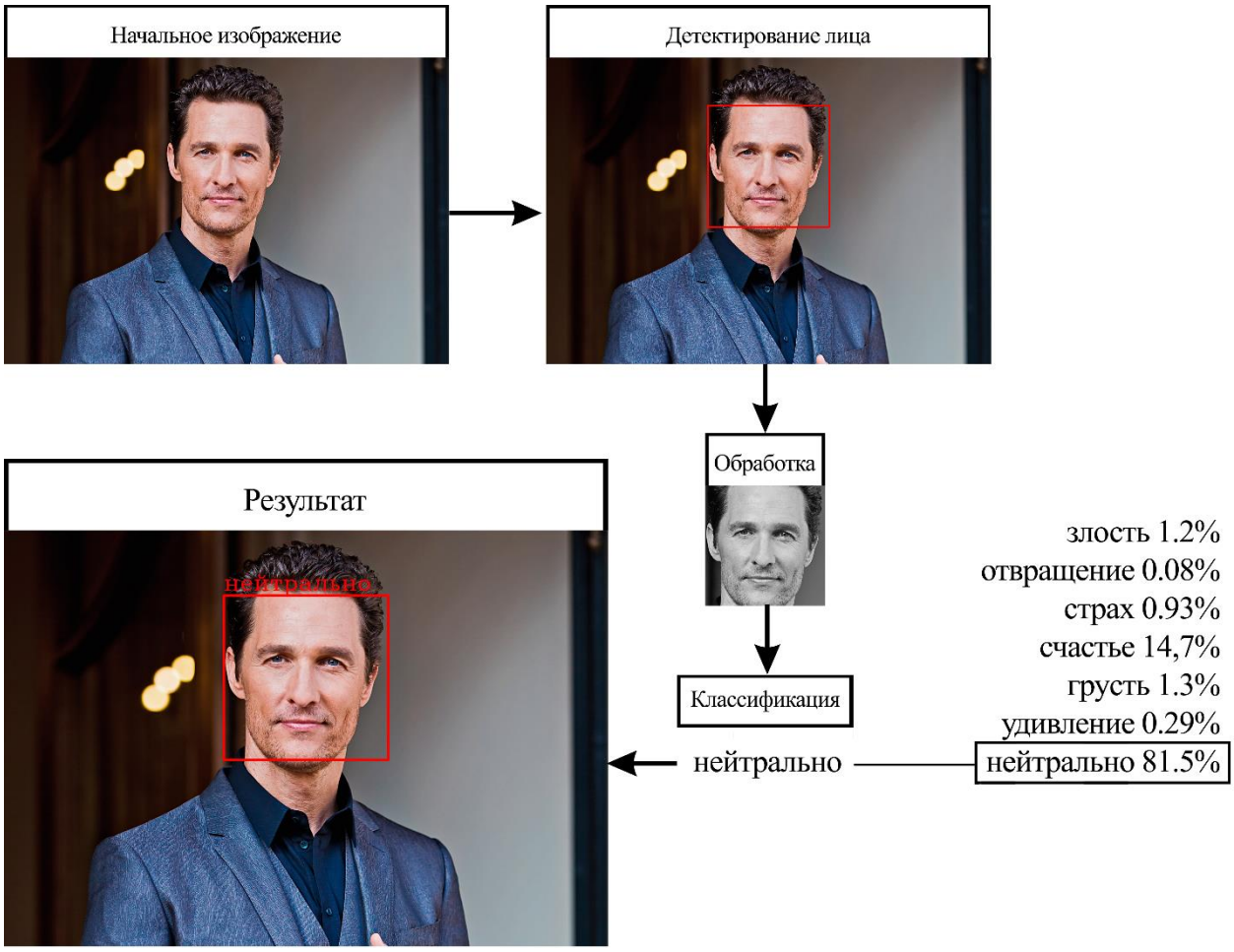

Рисунок 4.4 – Обработка изображения

4.2 Результат тестирования

Итоговое тестирование проводится в формате подачи изображения в нейронную сеть для последующего сравнения ожидаемого результата с получившимся.

Для этого были отобраны изображения в количестве 100 штук для каждого класса эмоций, в общей сумме тестируется 700 лиц. Начальные изображения имеют имя соответствующей эмоции и порядковый номер (рисунок 4.5).

Рисунок 4.5 – Тестируемые изображения

В качестве результата, получаем изображения с определенной эмоцией, которая записывается в имя изображения (рисунок 4.6).

Рисунок 4.6 – Результат

Процесс такого тестирования записывается в xml файл с параметрами: имя (начальная эмоция + порядковый номер); имя (начальная эмоция + порядковый номер) + полученная эмоция.

В результате получаем данные по идентификации каждой эмоции отдельно, для визуализации была построена диаграмма точности определения эмоции (рисунок 4.7).

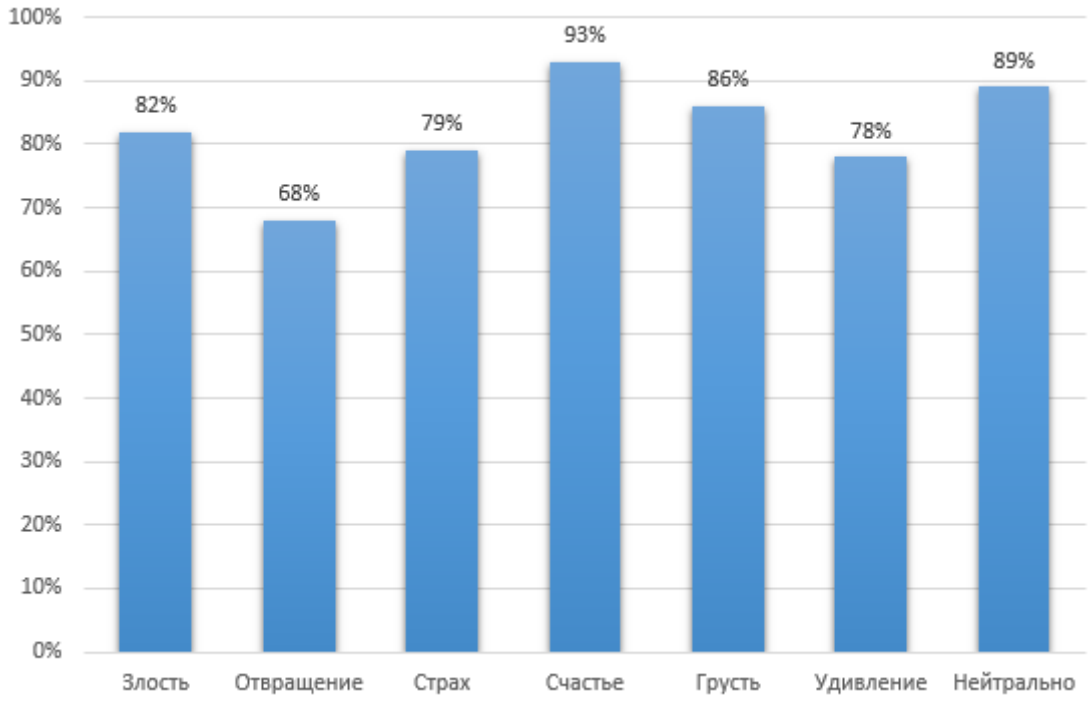

Рисунок 4.7 – Диаграмма итоговых данных

Максимальной точности модель достигает по определению эмоции «Счастье» составляет 93%, минимальный значение у «Отвращение» 68%. Такие данные обусловлены в большей части используемым датасетом FER-2013. Из сравнения точности и количества изображений, соответствующих эмоций, выявляется закономерность.

Количество обучаемой выборки оказывает сильное влияние на результативность, например, эмоция «отвращение», в сравнении с остальными, имеет крайне малое количество изображений и как результат – самый низкий уровень предсказаний.

Так или иначе, общий процент точности итогового тестирования достиг значения в 82%.

4.3 Методические указания

Среда разработки нейронной сети - Visual Studio Code, язык программирования Python 3.10 на кодировке UTF-8. Набор данных для определения эмоций использовался FER2013 формата csv. Детектирование лиц реализовано с помощью каскадов Хаара в виде xml файла.

Тестирование проводилось на компьютере следующей конфигурации:

- операционная система – Windows 10;
- процессор: Intel Core i3-4130, 3400MHz;
- оперативная память 4 GB.

Среди интегрируемых Python-библиотек были использованы:

- Keras версии 2.12.0 [24]

Библиотека Keras – это высокоуровневый API для создания нейронных сетей. Она написана на языке Python и является частью фреймворка TensorFlow. Keras позволяет создавать нейронные сети с минимальным уровнем абстракции и предоставляет простой интерфейс для создания сложных моделей.

Среди всего функционала библиотеки, использовались: слои Dense, Conv2D, BatchNormalization и другие; оптимизатор Adam; функции активации ReLu и Softmax; методы обратного вызова callbacks; класс models для импортирования обученной модели.

- Pandas версии 2.0.1 [25]

Библиотека Pandas – это инструмент для анализа данных в языке Python. Она предоставляет удобные и быстрые средства для работы с табличными данными, включая csv и excel форматы из материалов, используемых в процессе разработки нейронной сети.

Помимо работы с файлами, библиотека использовалась в качестве обработки и форматирования данных для подбора формата, требуемого разработанной моделью.

- MatPlotLib версии 3.7.0 [26]

MatPlotLib – это библиотека предназначенная для создания графиков и визуализации данных в языке Python. Она широко используется в нейронных сетях для визуализации результатов и процессов обучения. Данная библиотека позволяет создавать различные типы графиков, например, гистрограммы, для быстрого и эффективного анализа результатов.

Все полученные в процессе обучения гистограммы реализованы с помощью MatPlotLib.

- OpenCV версии 4.7.0.72 [27]

OpenCV – библиотека с открытым исходным кодом, предоставляющая набор функций для обработки и анализа изображений и видео. Она включает в себя множество функций для обработки, фильтраций, классификаций и много другое.

В рамках разработки нейронной сети, OpenCV использовалась для: импортирования каскадов Хаара; загрузки и всей последующей обработки изображений во время тестирования.

- NumPy версии 1.23.5[28]

Библиотека NumPy – это инструмент для выполнения математических операций в Python. Благодаря своей высокой эффективности и удобстве в работе с матрицами, библиотека широко используется в нейронных сетях.

С помощью NumPy были реализованы представления весов; хранение данных и результатов промежуточных вычислений; преобразование формата матриц под требования разработанной модели.

- h5py версии 3.8.0 [29]

h5py – это Python пакет, который позволяет работать с файлами в формате HDF5. Этот формат позволяет хранить большие объемы данных с иерархической структурой, что делает его удобным для использования в нейронных сетях.

Использование библиотеки h5py заключалось в хранении и загрузке весов и

параметров разработанной модели, которые используются в процессе обучения и предсказаний. Благодаря формату HDF5, сохранение и загрузка модели происходит с минимальными потерями данных и занимает меньше места на диске.

ЗАКЛЮЧЕНИЕ

В ходе дипломной работы, была разработана сверточная нейронная сеть, распознающая эмоции человека на цифровом изображении. На основе простой CNN-модели, спустя 4 версии разработки, была сформирована сложная архитектура, состоящая из 50 слоев.

Разработанная модель требует гораздо меньшего объема вычислительной мощности по сравнению с многими существующими моделями CNN. Обучение производится в малом количестве эпох, следовательно, нейросеть имеет высокую скорость обучения. Результат в 81% точности достигается лишь за 2 часа обучения.

Увеличение результативности нейронной сети с 51,65% точности на 30% обусловлено следующими факторам:

1) Уменьшение количества параметров за счет использования слоя подвыборки;
2) Решение проблемы переобучения с помощью механизмов регуляризации;
3) Полный отказ от полносвязных слоев в пользу комбинаций из нескольких слоев, которые позволяют более гибко настраивать параметры модели;
4) Использование более вариативной свертки входного тензора;
5) Разделение слоев на переменные для полного управления весами.

В перспективе разработки, модель имеет хороший потенциал на высокую результативность в точности по определению эмоций человека на изображении. Благодаря увеличению обучающей выборки определенных классов эмоций и повышению вычислительных мощностей машины, разработанная нейронная сеть может достигать более 90% точности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. СТО ВоГУ-МУ.1-2019 «Выпускная квалификационная работа. Требования к структуре, содержанию и оформлению»;
2. В.А. Головко: «Нейросетевые технологии обработки данных» / В.А. Головко, В.В. Краснопрошин;
3. Джун Грубер: «Оксфордский справочник по позитивным эмоциям и психопатологии» // Издательство оксфордского университета;
4. Татаренков Д. А.: «Анализ методов обнаружения лиц на изображении» // Издательство Молодой ученый, 2015. № 4;
5. FaceReader – Facial expression recognition software – Noldus // Режим доступа: https://www.noldus.com/facereader/new;
6. FaceSecurity – DevPost // Режим доступа: https://devpost.com/software/facesecurity;
7. Ronald Muller «A System for Automatic Face Analysis Based on Statistical Shape and Texture Models» С. 11-15;
8. Ф.М. Гафаров: «Искусственные нейронные сети и их приложения» учебное пособие // издательство Казанского университета;
9. В.В. Вьюгин: «Математические основы машинного обучения и прогнозирования»;
10. А.С. Соснин: «Функции активации нейросети» / Соснин А.С., Суслова И. А. // издательство ФГАОУ ВО;
11. Соколинский Л.Б.: «Глубокие нейронные сети. Метод обратного распространения ошибки» // издательство ЮУрГУ;
12. Введение в сверточные нейронные сети (Convolutional Neural Networks) // Режим доступа: https://habr.com/ru/articles/454986/;
13. А.Л. Горелик «Методы распознавания» / С. 208;
14. G. Yang: «Human face detection in a complex background. Pattern Recognition» / G. Yang, Thomas S. Huang;
15. P. Viola, M.J. Jones: «Rapid Object Detection using a Boosted Cascade of

Simple Features» // proceedings IEEE Conf. on CVPR 2001;

16. Р. Гонсалес: «Цифровая обработка изображений» / Р. Гонсалес, Р. Вудс // издательство Техносфера, Москва;

17. Обучение OpenCV каскада Хаара // Режим доступа: https://habr.com/ru/articles/208092/;

18. Dataset FER2013 / Kaggle – 2013 // Режим доступа: https://www.kaggle.com/datasets/msambare/fer2013;

19. Документация по слоям Keras // Режим доступа: https://keras.io/api/layers/;

20. Документация по оптимизатору Adam // Режим доступа: https://keras.io/api/optimizers/adam/;

21. Пакетная нормализация // Режим доступа: https://neerc.ifmo.ru/wiki/index.php?title=Batch-normalization;

22. Layer weight regularizers // Режим доступа: https://keras.io/api/layers/regularizers/;

23. haar-cascade-files // Режим доступа: https://github.com/anaustinbeing/haar-cascade-files/blob/master/haarcascade_frontalface_alt2.xml;

24. Документация по библиотеке keras Python // Режим доступа: https://keras.io/;

25. Pandas - Python Data Analysis Library // Режим доступа: https://pandas.pydata.org/;

26. Библиотека для работы с графикой Matplotlib // Режим доступа: https://matplotlib.org/;

27. OpenCV-Python Tutorials // Режим доступа: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html;

28. Документация по библиотеке NumPy Python // Режим доступа: https://numpy.org/;

29. Documentation HDF5 python // Режим доступа: https://docs.h5py.org/en/stable/.

ПРИЛОЖЕНИЕ 1

(обязательное)

Определение лица и обработка изображения

```
import cv2

image_path = "D:/.Labs/VKR/Neural_Emotions/test.jpeg"
# Загрузка изображения
image =  cv2.imread(image_path)
# Преобразование в черно-белый формат
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# Импортирование каскада Хаара
haar_detection =
cv2.CascadeClassifier("D:/.Labs/VKR/Neural_Emotions/haarcascades/haarcascade_fron
talface_alt2.xml")
# Детектирование лица с помощью импортированных каскадов
detect_image = haar_detection.detectMultiScale(image_gray)

size = len(detect_image)
if size != 0:
    for (a, b, w, h) in detect_image:
        # Выделяем лицо квадратом
        cv2.rectangle(image, (a, b),
                      (a + h, b + w), # Определение размеров квадрата
                      (0, 0, 255), 5) # Определение цвета и толщины квадрата
# Вывод изображения
cv2.imshow('Detection Face', image)
cv2.waitKey(0)
```

ПРИЛОЖЕНИЕ 2

(обязательное)

Интеграция набора данных

```
# Загрузка csv файла с изображениями
data = pd.read_csv('D:/.Labs/VKR/Neural_Emotions/FER-2013/icml_face_data.csv')
data.head()

def data_lead(data):
    # Создаем массив, заполненный нулями
    array_image = np.zeros(shape=(len(data), 48, 48))
    # Создаем массив, заполненный параметром emotion из csv файла
    label_image = np.array(list(map(int, data['emotion'])))

    # Заполняем массив array_image значениями pixels из csv файла
    for i, row in enumerate(data.index):
        image = np.fromstring(data.loc[row, 'pixels'], dtype=int, sep=' ')
        image = np.reshape(image, (48, 48))
        array_image[i] = image
    return array_image, label_image

# Применяем созданный класс для распределения данных на 3 этапа
# 1 этап - тренировка (поступают все параметры изображений предназначенных для
Training)
train_array_image, train_label_image = data_lead
(data[data['Usage']=='Training'])
# 2 этап - проверки (поступают все параметры изображений предназначенных для
PrivateTest)
valid_array_image, valid_label_image = data_lead
(data[data['Usage']=='PrivateTest'])
# 3 этап - тестирование (поступают все параметры изображений предназначенных
для PublicTest)
test_array_image, test_label_image = data_lead
(data[data['Usage']=='PublicTest'])
```

ПРИЛОЖЕНИЕ 3

(обязательное)

Слои модели CNN_v3

Слой	Выходное значение	Параметры
conv2d (Conv2D)	(None, 48, 48, 16)	160
batch_normalization (BatchNormalization)	(None, 48, 48, 16)	64
activation (Activation)	(None, 48, 48, 16)	0
conv2d_1 (Conv2D)	(None, 46, 46, 16)	2 320
max_pooling2d (MaxPooling2D)	(None, 23, 23, 16)	0
dropout (Dropout)	(None, 23, 23, 16)	0
conv2d_2 (Conv2D)	(None, 23, 23, 32)	4 640
batch_normalization_1 (BatchNormalization)	(None, 23, 23, 32)	128
activation_1 (Activation)	(None, 23, 23, 32)	0
conv2d_3 (Conv2D)	(None, 22, 22, 32)	4 128
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 32)	0
dropout_1 (Dropout)	(None, 11, 11, 32)	0
conv2d_4 (Conv2D)	(None, 11, 11, 64)	18 496
batch_normalization_2 (BatchNormalization)	(None, 11, 11, 64)	256
activation_2 (Activation)	(None, 11, 11, 64)	0
conv2d_5 (Conv2D)	(None, 10, 10, 64)	16 448

Слой	Выходное значение	Параметры
max_pooling2d_2 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_2 (Dropout)	(None, 5, 5, 64)	0
conv2d_6 (Conv2D)	(None, 5, 5, 128)	73 856
batch_normalization_3 (BatchNormalization)	(None, 5, 5, 128)	512
activation_3 (Activation)	(None, 5, 5, 128)	0
conv2d_7 (Conv2D)	(None, 4, 4, 128)	65 664
dropout_3 (Dropout)	(None, 4, 4, 128)	0
conv2d_8 (Conv2D)	(None, 4, 4, 128)	147 584
batch_normalization_4 (BatchNormalization)	(None, 4, 4, 128)	512
activation_4 (Activation)	(None, 4, 4, 128)	0
conv2d_9 (Conv2D)	(None, 3, 3, 128)	65 664
dropout_4 (Dropout)	(None, 3, 3, 128)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 128)	147 584
batch_normalization_5 (BatchNormalization)	(None, 128)	512
activation_5 (Activation)	(None, 128)	0
dropout_5 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8 256

Слой	Выходное значение	Параметры
batch_normalization_6 (BatchNormalization)	(None, 64)	256
activation_6 (Activation)	(None, 64)	0
dropout_6 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455
Total params: 557,495 Trainable params: 556,375 Non-trainable params: 1,120		

ПРИЛОЖЕНИЕ 4

(обязательное)

Слои модели CNN_v4

Слой	Выходное значение	Параметры
input_1 (InputLayer)	[(None, 48, 48, 1)]	0
conv2d (Conv2D)	(None, 46, 46, 8)	72
batch_normalization (BatchNormalization)	(None, 46, 46, 8)	32
activation (Activation)	(None, 46, 46, 8)	0
conv2d_1 (Conv2D)	(None, 44, 44, 8)	576
batch_normalization_1 (BatchNormalization)	(None, 44, 44, 8)	32
activation_1 (Activation)	(None, 44, 44, 8)	0
separable_conv2d (SeparableConv2D)	(None, 44, 44, 16)	200
batch_normalization_3 (BatchNormalization)	(None, 44, 44, 16)	64
activation_2 (Activation)	(None, 44, 44, 16)	0
separable_conv2d_1 (SeparableConv2D)	(None, 44, 44, 16)	400
batch_normalization_4 (BatchNormalization)	(None, 44, 44, 16)	64
conv2d_2 (Conv2D)	(None, 22, 22, 16)	128
max_pooling2d (MaxPooling2D)	(None, 22, 22, 16)	0

Слой	Выходное значение	Параметры
batch_normalization_2 (BatchNormalization)	(None, 22, 22, 16)	64
add (Add)	(None, 22, 22, 16)	0
separable_conv2d_2 (SeparableConv2D)	(None, 22, 22, 32)	656
batch_normalization_6 (BatchNormalization)	(None, 22, 22, 32)	128
activation_3 (Activation)	(None, 22, 22, 32)	0
separable_conv2d_3 (SeparableConv2D)	(None, 22, 22, 32)	1 312
batch_normalization_7 (BatchNormalization)	(None, 22, 22, 32)	128
conv2d_3 (Conv2D)	(None, 11, 11, 32)	512
max_pooling2d_1 (MaxPooling2D)	(None, 11, 11, 32)	0
batch_normalization_5 (BatchNormalization)	(None, 11, 11, 32)	128
add_1 (Add)	(None, 11, 11, 32)	0
separable_conv2d_4 (SeparableConv2D)	(None, 11, 11, 64)	2 336
batch_normalization_9 (BatchNormalization)	(None, 11, 11, 64)	256
activation_4 (Activation)	(None, 11, 11, 64)	0

Слой	Выходное значение	Параметры
separable_conv2d_5 (SeparableConv2D)	(None, 11, 11, 64)	4 672
batch_normalization_10 (BatchNormalization)	(None, 11, 11, 64)	256
conv2d_4 (Conv2D)	(None, 6, 6, 64)	2 048
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
batch_normalization_8 (BatchNormalization)	(None, 6, 6, 64)	256
add_2 (Add)	(None, 6, 6, 64)	0
separable_conv2d_6 (SeparableConv2D)	(None, 6, 6, 128)	8 768
batch_normalization_12 (BatchNormalization)	(None, 6, 6, 128)	512
activation_5 (Activation)	(None, 6, 6, 128)	0
separable_conv2d_7 (SeparableConv2D)	(None, 6, 6, 128)	17 536
batch_normalization_13 (BatchNormalization)	(None, 6, 6, 128)	512
conv2d_5 (Conv2D)	(None, 3, 3, 128)	8 192
max_pooling2d_3 (MaxPooling2D)	(None, 3, 3, 128)	0
batch_normalization_11 (BatchNormalization)	(None, 3, 3, 128)	512
add_3 (Add)	(None, 3, 3, 128)	0

Слой	Выходное значение	Параметры
conv2d_6 (Conv2D)	(None, 3, 3, 7)	8 071
global_average_pooling2d (GlobalAveragePooling2D)	(None, 7)	0
predictions (Activation)	(None, 7)	0
Total params: 58,423 Trainable params: 56,951 Non-trainable params: 1,472		