САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

САНАКОЕВ БАТРАДЗ АЛЕКСАНДРОВИЧ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА СИСТЕМЫ ТРЕНИРОВКИ И РАСПОЗНАВАНИЯ ЭМОЦИЙ

НАПРАВЛЕНИЕ 010400

«ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

НАУЧНЫЙ РУКОВОДИТЕЛЬ,
СТАРШИЙ ПРЕПОДАВАТЕЛЬ
КАФЕДРЫ ТП,
СТУЧЕНКОВ А.Б.

Санкт-Петербург

2017

Оглавление

Введение

В конце XVIII и в начале XIX веков эмоциональная сфера человека впервые всерьез заинтересовала философов и физиологов того временного периода. Поначалу психология не касалась эмоций ни коим образом. В течение довольно длительного времени в основе направления изучения эмоций лежало несоответствие понятий человека о том, что будет и тем, что есть. В соответствии с этой задумкой эмоции являются показателем личности индивидуума. Но существовало и противоположное мнение, согласно которому происхождение человеческих эмоций таится в реакциях человеческого организма.

Тема эмоций не обошла стороной и психологию. Многие ученые-психологи занялись изучением данного понятия, что повлекло за собой огромное количество написанных книг, статей, докторских диссертаций и др. С течением времени изучение эмоций разделилось на гораздо более мелкие подобласти, которые занимаются исследованием отдельно взятых понятий, касающихся эмоций. Благодаря глубоким систематическим исследованиям в этой области даже обычные люди стали задаваться вопросами о причине возникновения эмоций, их разновидности, методах распознавания и изменения эмоционального фона человека с течением времени. При этом интерес этот затронул огромное количество областей и профессий: от церковного служителя до работника банка, от полицейского до телеведущего.

Роль эмоций в современном мире настолько возрос, что множество компаний уделяют особое внимание специальным «эмоциональным» тренингам для того, чтобы работники были максимально устойчивыми эмоционально и психологически или могли определить нужную эмоцию клиентов, с которыми работают, для более эффективного и верного выполнения работы. И ведь причины, которые побуждают на такие меры вполне оправданы и, что самое главное, действительно важны. По человеческим эмоциям можно составить первоначальный психологически

портрет человека, чем пользуются многие работодатели во время собеседований, когда большое внимание уделяется поведению кандидата на работу, его мимике и реакции на определенные вопросы. Опытные следователи при допросе пристально следят за эмоциями подозреваемых, пытаясь элементарно определить истинность данных ему показаний. Конечно же нельзя забывать и про самую главную область изучения эмоций – психологию. Все вышеперечисленные области в своем большинстве имеют в своем штате психологов, которые нужны для определенных задач в зависимости от отрасли компании.

Анализ всей этой информации неизбежно способствует подключению к изучению эмоций новейших технологических средств. Еще в самом начале XXI века многие АйТи компании занялись разработкой софта, направленного на распознавание и определение эмоций по фотографиям. При этом первичные результаты подобных исследований не останавливали программистов на достигнутом, и, в дальнейшем, все реализованные способы дополнялись новыми методиками для достижения более качественных результатов. Изучение различных методов распознавания эмоций с помощью средств компьютерных технологий идет до сих пор, так как с каждым годом повышается количество, а самое главное, и качество реализуемых продуктов.

Изучению некоторых таких методов и посвящено данное исследование, целью которого является показать на примере некоторых реализованных средств диагностики эмоций вероятность успешности их распознавания. Сначала будут описаны основные понятия и термины, касающиеся определения эмоций. Далее, будет изложен сам, выбранный для исследования, алгоритм, после чего будут продемонстрированы и разобраны результаты исследования, реализованные в среде Visual Studio на языке C# с использованием библиотеки машинного зрения OpenCV.

Постановка задачи

Цель работы. В рамках данной работы поставлена следующая цель:

- Проектирование и разработка приложения для распознавания и тренировки эмоций;

Решаемые задачи. Поставленной в работе цели соответствует решение следующих задач:

- Распознавание лиц при помощи каскадов Хаара и алгоритма Виолы-Дэвиса;
- Формирование качественной и обширной обучающей выборки;
- Обучение нейронной сети распознаванию эмоций;
- Программная реализация представленной задачи;
- Получение приемлемых результатов написанной программы;

План решения. Для осуществления реализации и выполнения задач используется план:

- Провести анализ предметной области;
- Изучить методики, выбранные для реализации цели;
- Составить схему исследования;
- Разработать понятный, простой и компактный интерфейс приложения для легкости эксплуатации им пользователями;
- Написать программу распознавания эмоций человека;

Краткий обзор литературы

В данной выпускной квалификационной работе в качестве литературы использовались учебно-методические пособия, статьи из журналов и электронные виды информации в сети интернет. Ниже список основных источников литературы и их небольшой обзор:

1. **P. Viola and M.J. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features», proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001**

   В данной статье описаны и разобраны способы детектирования объектов на изображении при помощи усиленных каскадов простых функций.

2. **Буй Т.Т.Ч., «Алгоритмы распознавания лиц и жестов на основе вейвлет-преобразований и метода главных компонент», 2011**

   В этой диссертации очень подробно и досконально рассказывается о многих различных методах распознаваниях объектов на изображении, таких как PCA, алгоритм Виолы-Джонса и другие.

3. **Статья из https://habrahabr.ru/post/278435/**

   В приведенной статье описывается алгоритм бинаризации изображений, которое переводит изображение в черно-белые тона, приводит входные данные в виде числового вектора и уменьшает размерность данных.

4. **Duda R., Hart P., «Pattern Classification and Scene Analysis», 1973**

   В статье Дуда Р. и Харта П. рассказывается о паттерной классификации и операторе Собеля.

5. **http://it-claim.ru/Persons/Semenova_Yana/SemenovaPlakats.pdf**

   Работа Семеновой Я. показывает, как должна строиться программа распознавания эмоций, в каком порядке проводить обучение нейронной сети и наглядно демонстрирует, каких результатов следует ожидать.

Глава 1. Описание существующих методик и алгоритмов по распознаванию эмоций

Задача распознавания лиц и объектов является достаточно востребованной в современном мире, в частности, в таких областях, как компьютерное зрение, биометрия, интеллектуальные системы безопасности и другие. Людям не сложно решать задачу распознавания объектов, однако, как показывает практика эта легкость не относится к компьютерам. В последние года задача распознавания объектов находит применения во многих отраслях человеческой деятельности, к примеру, для идентификации личности.

На сегодняшний день разработано и реализовано множество методик и алгоритмов для решения задач по распознаванию объектов. Но все они, в качестве базиса, используют стандартные два этапа: нахождение и выделение объектов на изображениях и, собственно, распознавание объекта. Однако существуют различные факторы, которые усложняют, к примеру, распознавание лица на изображении:

- Маленькое разрешение изображения;
- Условия освещение, углы обзора, наличие солнцезащитных очков и различные перекрытия лица посторонними предметами;
- Наличие объектов на заднем плане;
- Количество лиц на изображении;

Распознавание эмоций является едва ли не самой популярной областью среди задач распознавания объектов на изображении. Подходом для распознавания эмоций состоит из преобразования оригинального изображения в виде вектора, выделение основных признаков изображения и классификация по признакам.

Итак, первым делом требуется выделить признаки объектов на изображении, в нашем случае лиц. Для выполнения этой задачи существует алгоритм Виолы-Джонса, основанный на вейвлет-преобразованиях Хаара, но

являющийся более доработанным и усовершенствованным, а также можно использовать совместное применение преобразований Хаара и Дебоши. Процесс выделения лица на изображении при применении вейвлетов Хаара и Дебоши выглядит так:

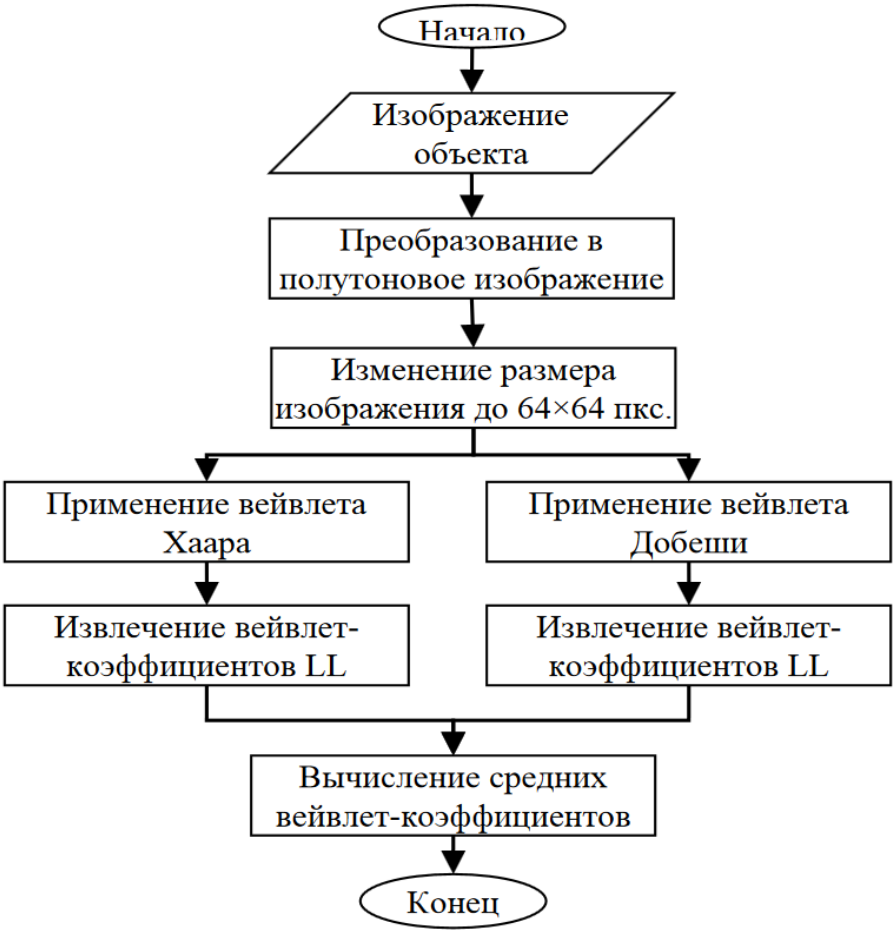


Рисунок 1.1 Вейвлеты Хаара и Дебоши

Как показано на схеме, в самом начале входное изображение переводится из RGB формата в пространство YUV и, затем, приводится к размеру изображения 64×64 пикселя. Далее, после применения вейвлетов Хаара и Дебоши, мы получаем извлеченные вейвлет коэфициенты X_{LL} и D_{LL}, уже на основе которых и выделяются основные признаки изображения.

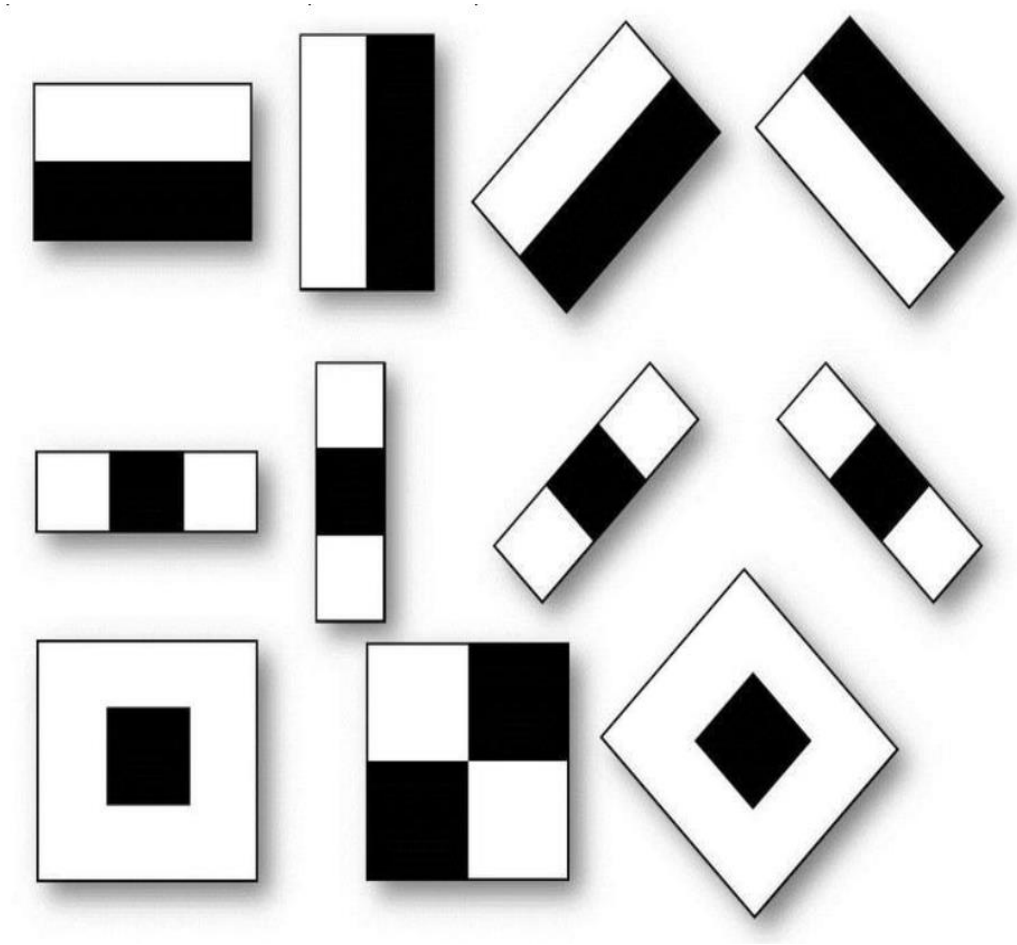


Рисунок 1.2 Признаки Хаара

Что касается алгоритма Виолы-Джонса, то он базируется на каскадах Хаара, но, как было написано выше, является более усовершенствованной их версией. При применении данного алгоритма используется представление изображения в интегральном виде, для того, чтобы быстро вычислять необходимые нам объекты, и, разумеется, уже описанные выше каскады Хаара. Когда нам требуется распознать лицо на изображении, то само изображение мы представляем в виде двумерной матрицы, в котором содержится информация о искомых объектах изображения. Далее, при работе алгоритма используется так называемое сканирующее окно. Это работает так: для нашего изображения выбирается размер окна сканирования и, соответственно, берется признак Хаара, далее, сканирующее окно начинает двигаться по изображению с шагом в одну ячейку окна, при этом в самом окне происходит сканирование выбранного участка изображения, в котором вычисляется около 200000 признаков, при этом в каждом окне сканирование

проходит последовательно для различных масштабов, дальше все признаки, найденные во время сканирования, попадают к классификатору, который выдает результат.

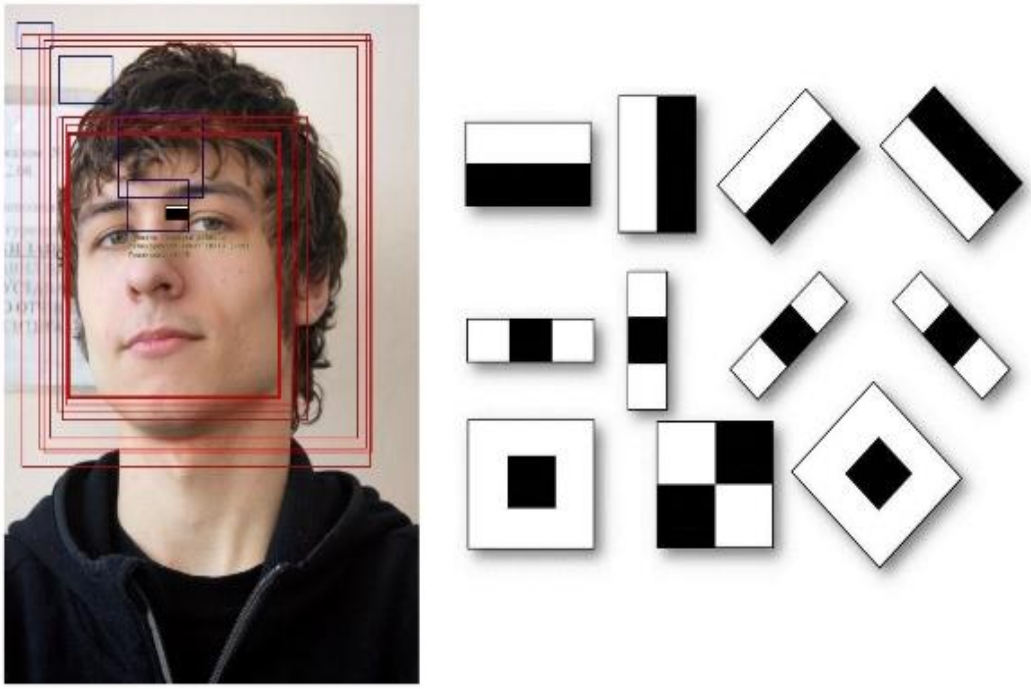


Рисунок 1.3 Алгоритм Виолы-Джонса

Также одним из самых основных методов для распознавания лица, а также уменьшения размерности входных данных является метод главных компонент – PCA. Данный метод позволяет отобразить одномерный вектор пикселей, который строится из двумерного изображения исследуемого объекта, в виде компактных компонентов пространства признаков. Алгоритм PCA выглядит так:

Процесс создания собственных объектов

Шаг 1. Создать набор изображений $I_1,...,I_M$, отражающий вариации объектов. Представить их в виде векторов $\vec{I}_1,...,\vec{I}_M$.

Шаг 2. Вычислить среднее изображение по формуле:

$$\vec{I}_{ср} = \frac{1}{M}\sum_{i=1}^{M}\vec{I}_i.$$

Шаг 3. Вычесть среднее изображение из каждого изображения:

$$\vec{\Phi} = \vec{I}_i - \vec{I}_{ср}.$$

Шаг 4. Вычислить собственные объекты $\vec{u}_i$:

$$\vec{u}_i = \sum_{k=1}^{M} v_{ik}\Phi_k, i = 1,...,M,$$

где $\vec{v}_i$ – собственные векторы матрицы $W^T W$, $W = \left\{\vec{\Phi}_1,...,\vec{\Phi}_M\right\}$.

Рисунок 1.4 PCA

Процесс распознавания объектов

Шаг 1. Поступление на вход нового изображения неизвестного объекта $I_{вх}$.

Шаг 2. Вычисление для каждого объекта в БД соответствующего ему вектора ω_i в пространстве собственных объектов V_p:

$$\omega_i = \overrightarrow{u_i^T}(\vec{I}_{вх} - \vec{I}_{ср}), i = 1,...,M.$$

Шаг 3. Вычисление проекции ω любого нового изображения $I_{вх}$ в пространстве собственных объектов V_p.

Шаг 4. Если расстояние $d=|I_{вх}-\omega|$ больше некоторого предопределенного порога ε_1, то классифицировать изображение как «иной объект».

Шаг 5. В случае если минимальное расстояние $d_k=|\omega-\omega_k|$ между проекцией нового изображения и известным представителем набора объектов меньше некоторого заданного порога ε_2, то классифицировать изображение как «объект номер k».

Рисунок 1.5 PCA

Шаг 6. В оставшемся случае ($d<\varepsilon_1$ и $d_k \geq \varepsilon_2$) классифицировать изображения как «неизвестный объект» и если это необходимо, то добавить новое изображение в БД и заново вычислить собственные объекты.

Рисунок 1.6 PCA

На основе вышеописанных методов существует алгоритм, который основан на совместном использовании вейвлетов Хаара, Дебоши и PCA. Первый шаг алгоритма позволяет создать базы признаков объекта:

1. Выделение признаков каждого i-ого изображения объекта из M изображений объектов обучающей выборки на основе совместного применения вейвлет-преобразований Хаара и Добеши.

2. Представление полученных вейвлет-коэффициентов в виде вектора $\vec{I}_i$, где $i = 1,..,M$.

3. Вычисление среднего изображения по формуле:

$$\vec{I}_{cp} = \frac{1}{M}\sum_{i=1}^{M}\vec{I}_i.$$

4. Вычитание среднего изображения из каждого изображения $\vec{\Phi}_i = \vec{I}_i - \vec{I}_{cp}$.

5. Вычисление собственных объектов:

$$\vec{u}_i = \sum_{k=1}^{M} v_{ik}\Phi_k, i = 1,...,M,$$

где $\vec{v}_i$ – собственные векторы матрицы $W^T W$, $W = \left\{\vec{\Phi}_1,...,\vec{\Phi}_M\right\}$.

6. Вычисление для каждого объекта соответствующего ему вектора $\vec{\Omega}_k^T = \{\omega_1,...,\omega_M\}$ в пространстве собственных объектов, где $\omega_i = \vec{u}_i^T(\vec{I}_i - \vec{I}_{cp}), i = 1,...,M.$

Рисунок 1.7 PCA-H-D

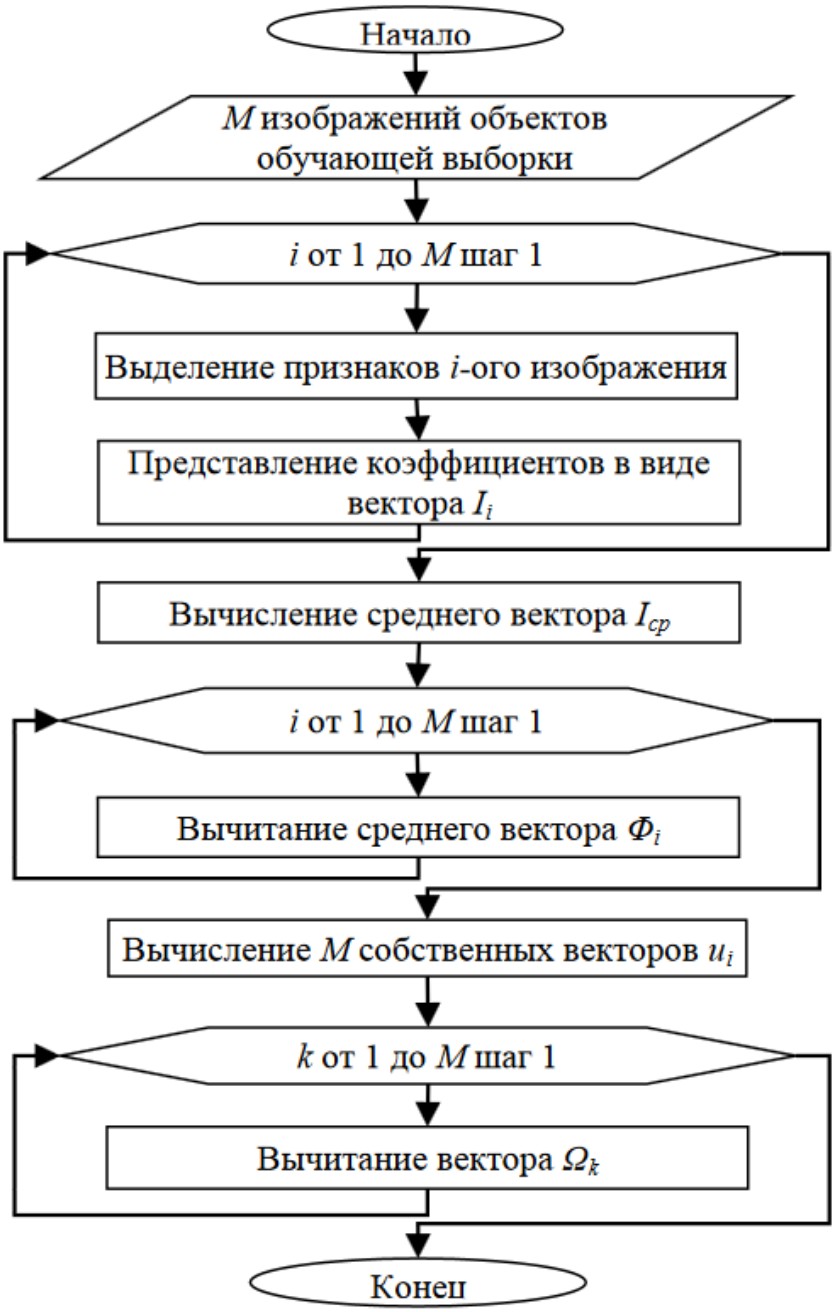


Рисунок 1.8 PCA-H-D

После создания базы признаков объектов алгоритм переходит, непосредственно, к процессу распознавания, схема которой выглядит так:

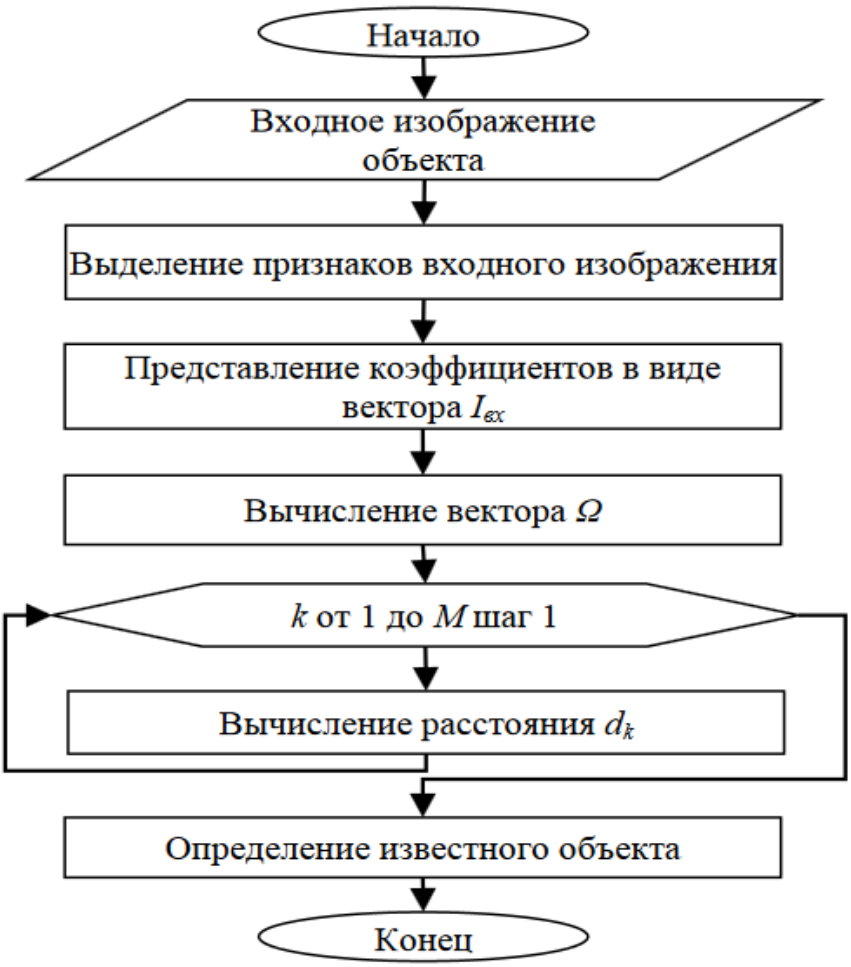
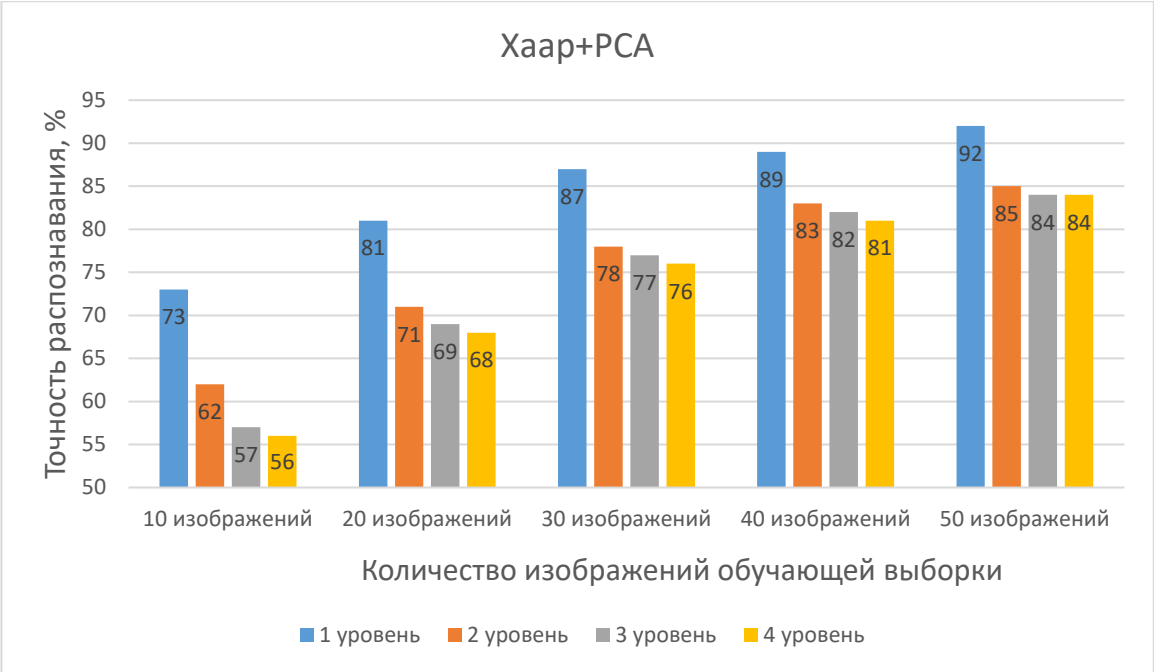
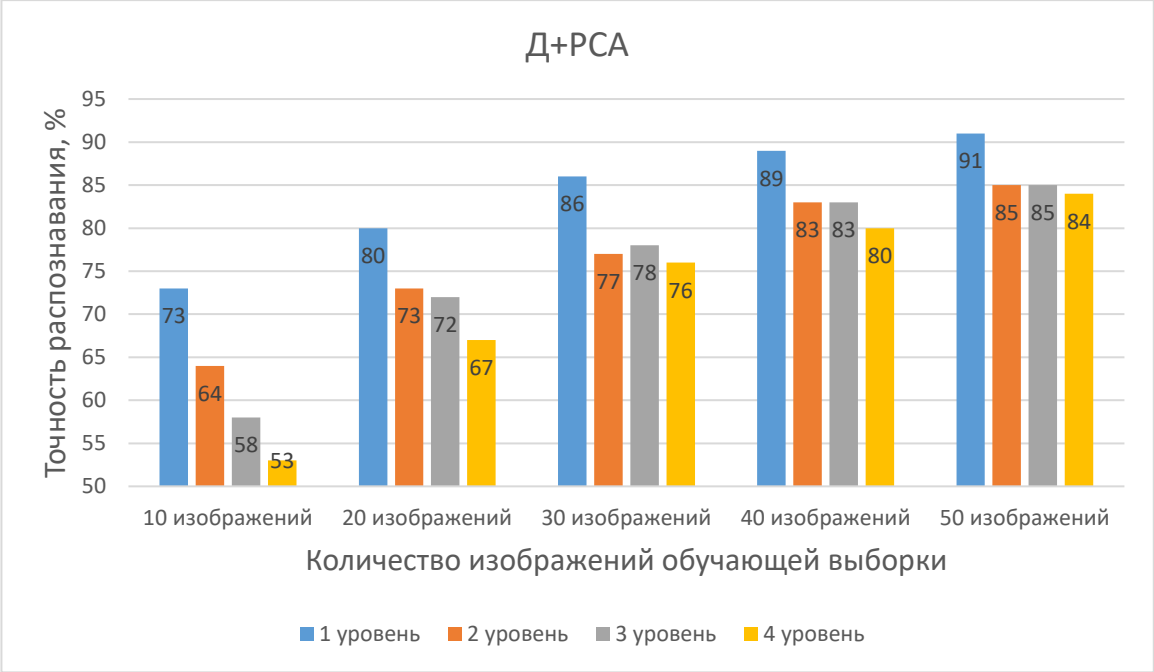
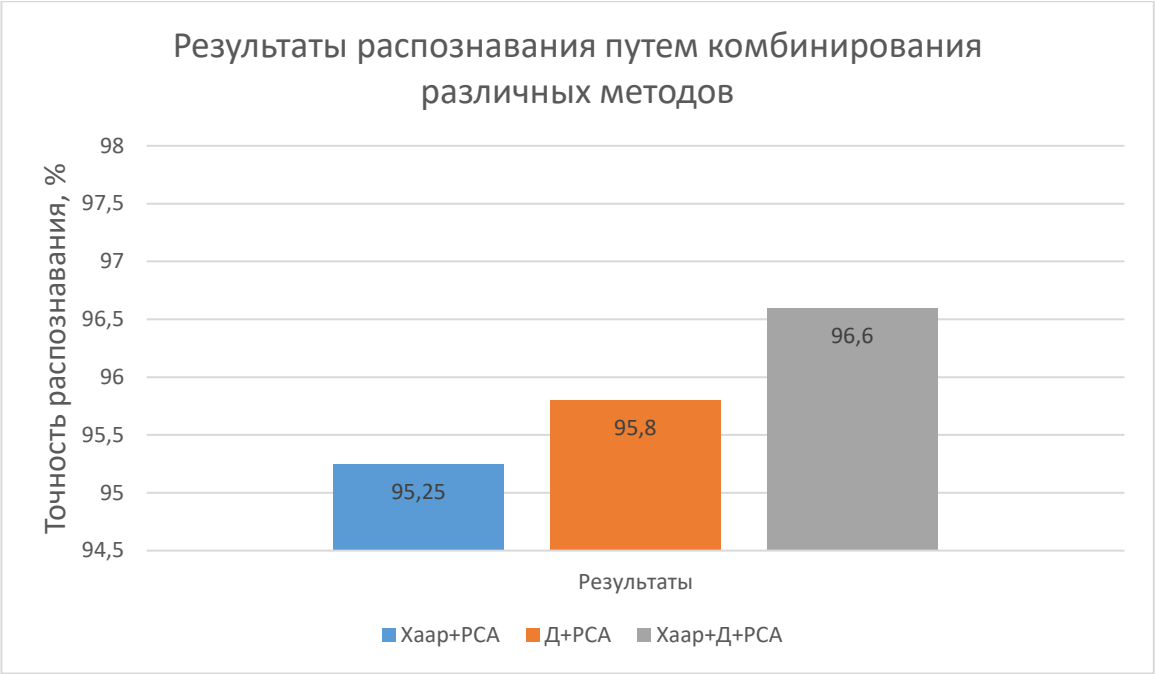


Рисунок 1.9 PCA-H-D

В качестве обучающей выборки для данного алгоритма было рассмотрено 100 изображений 10 человек, а также проведено пару экспериментов, направленных на исследование зависимости точности распознавания от количества изображений и зависимости точности распознавания от выбора вейвлет-преобразования. Как уже было неоднократно написано выше, процесс распознавания проходит в два этапа: фиксирование признаков изображений лиц обучающей и тестовой выборок, и их сравнение. Результаты экспериментов представлены ниже:


Хаар+PCA
Точность распознавания, %
95
90
85
80
75
70
65
60
55
50
73
62
57
56
81
71
69
68
87
78
77
76
89
83
82
81
92
85
84
84
10 изображений
20 изображений
30 изображений
40 изображений
50 изображений
Количество изображений обучающей выборки
1 уровень
2 уровень
3 уровень
4 уровень


Д+PCA
Точность распознавания, %
95
90
85
80
75
70
65
60
55
50
73
64
58
53
80
73
72
67
86
77
78
76
89
83
83
80
91
85
85
84
10 изображений
20 изображений
30 изображений
40 изображений
50 изображений
Количество изображений обучающей выборки
1 уровень
2 уровень
3 уровень
4 уровень



Данные методы и алгоритмы были разобраны и описаны в этой главе исключительно по причине своей доступности, понятности способа реализации и, что самое главное, отличным результатам точности распознавания.

Глава 2. Основные понятия и определения

2.1 Нейронные сети

2.1.1 Биологический нейрон и его модель

Нейронные сети были предложены У.Маккалоком и У.Питтсом как модель биологических нейронов. Это была попытка смоделировать искусственный интеллект путем создания имитации его внутренней структуры. Предполагалось, что если скопировать внутреннюю состав мозга, то, возможно, получится создать искусственный интеллект. Действительно, это привело к довольно хорошим результатам. Конечно, получить полноценный искусственный интеллект не вышло, но нейронные сети могут решать многие довольно сложные задачи.

Поговорим немного о нервной системе. Нервная система есть практически у всех представителей царства животных, физически у всех

многоклеточных, кроме губок, которые представляют собой, на самом деле, пласты, колонии различных типов клеток, являющихся началом симбиоза множества клеток внутри одного организма, но в самой примитивной его стадии. По мере того, как все больше и больше клеток принимает участие в жизни одного организма, обеспечивает разные функции этого организма, требуется координация этих клеток, требуется передавать информацию от одних клеток к другим. Именно этой цели служит нервная система. Нервная система состоит из нейронов. Выглядит это примерно так:

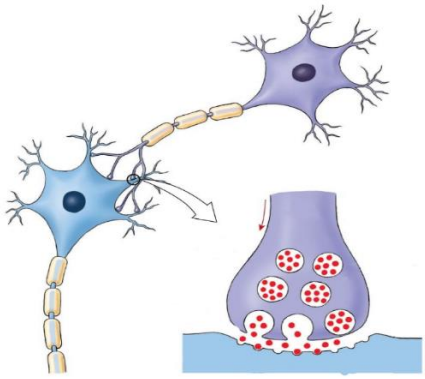


Рисунок 2.1.1 Строение нейрона

Здесь изображен нейрон, его тело, которое называется сомой, а также дендриты и аксон. Внутри сомы можно заметить ядро нейрона. Дендриты – это маленькие ветвящиеся отростки, через которые в нейрон поступает информация, они обычно бывают в большом, а аксон – это более крупный отросток, по которому, наоборот, информация выходит из нейрона и передается следующему нейрону через синапс.

На рисунке 1.1 видно, как из одного нейрона через аксон передается информация через дендриты другому нейрону. Для того, чтобы лучше представлять этот процесс на изображении присутствует в увеличенном виде соединение аксона с дендритом. Таким образом нейроны соединяются между собой, образуя сети.

Каким образом все это работает? Нейроны передают друг другу информацию через синапсы. Каждый нейрон может находится в возбужденном или в стационарном состоянии. В возбужденном состоянии он проводит ток, а в стационарном – не проводит и таким образом в нервных тканях выстраиваются проводящие участки, по которым идут электрические импульсы и, таким образом, информация может передаваться по нервным тканям. Нейроны могут переходить из состояния в состояние и, меняя состояние, они руководствуются «мнением» соседних нейронов. Вот как это происходит: если рассматривать рисунок 1.1, то внутри верхнего нейрона и его аксона есть специальные молекулы, которые называются нейромедиаторы(адреналин, драмин и тд.) и, когда этот первый нейрон находится в возбужденном состоянии, эти молекулы проникают через синапсов клетку второго нейрона, т.е. , когда нейрон находится в возбужденном состоянии, то он отдает свои нейромедиаторы другим нейронам. Некоторые нейромедиаторы оказывают тормозящий эффект, который способствует переходу нейрона в стационарное состояние, а некоторые оказывают, наоборот, возбуждающий эффект. И после того как возбуждающих нейромедиаторов окажется больше, чем тормозящих, то второй нейрон тоже перейдет в возбуждающее состояние и начнет передавать нейромедиаторы уже тем нейронам, которые подключены к нему. Таким образом этот эффект распространяется по нервным тканям, возбужденные нейроны выстраиваются в цепочку и передают сигнал от рецепторов до рефлекторов. Такая цепочка называется рефлекторной дугой.

Рефлекторные дуги бывают короткие и длинные, и у разных живых организмов наблюдаются по-разному. Но, по сути, эти дуги и являются основным представлением нервной системы организма. Есть, к примеру, моллюски с очень сложной нервной системой, затем насекомые с чуть более сложной и уже потом идут люди с большими лобными долями, которые могут обрабатывать информацию очень сложным образом.

И вот именно эти клетки – нейроны, моделировали Маккалок и Питтс в своей оригинальной работе. Они ввели математическую модель нейрона, устроенную следующим образом: фактически нейрон – это функция, которая вычисляет свое значение по нескольким переданным в нее параметрам. Графически ее можно представить так:

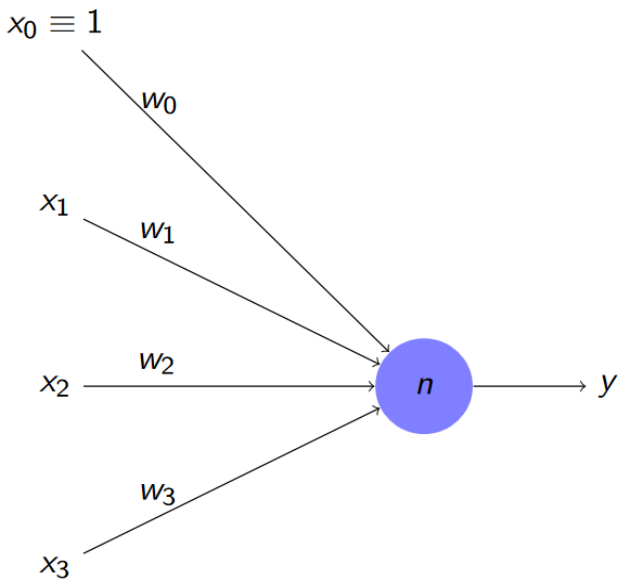


Рисунок 2.1.2 Графическое представление нейрона

На рисунке 1.2 клетка выделена синим кружком, стрелки, направленные к ней, это информация, передаваемая в нейрон, а исходящая из нейрона информация соответствует аксону. На вход нейрону подаются сигналы: x_1, x_2, x_3 – это обычные числа. Далее, нейрон, получив эти сигналы, вычисляет их взвешенное произведение. С каждым входом ассоциирован его вес – w_1, w_2, w_3. Это тоже числа. Взвешенное произведение вычисляется по формуле:

$$y = f\left(\sum_{i=1}^{n} w_i \times x_i\right)$$

Веса позволяют моделировать различия в синапсах: если вес положительный, то этот синапс получается разгоняющим как будто он передает разгоняющие нейромедиаторы, если отрицательный – то как будто

тормозящие. Чем больше сигнал подан на вход, тем меньше будет значение выражения в скобках. Также может быть разная интенсивность влияния входов на общий результат – тоже достигается за счет больших или меньших весов. То есть x_1, x_2, x_3 – это дендриты и выходы из аксонов предыдущего нейрона, w_1, w_2, w_3 – это настройки синапса, а y – это результат. Игрек вычисляется как некая нелинейная функция от взвешенной суммы. Например, это может быть функция sign(x) или tanh(-βx):

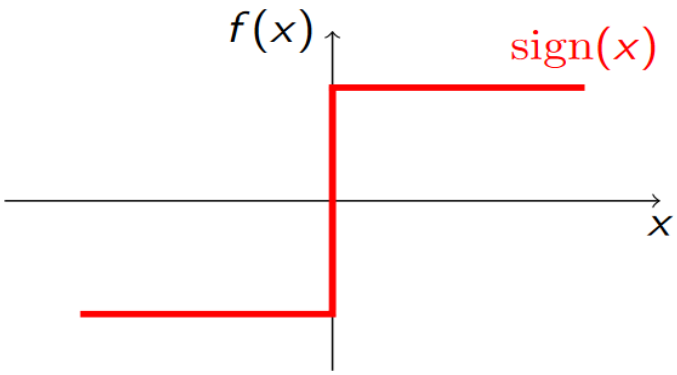


Рисунок 2.1.3 Sign(x)

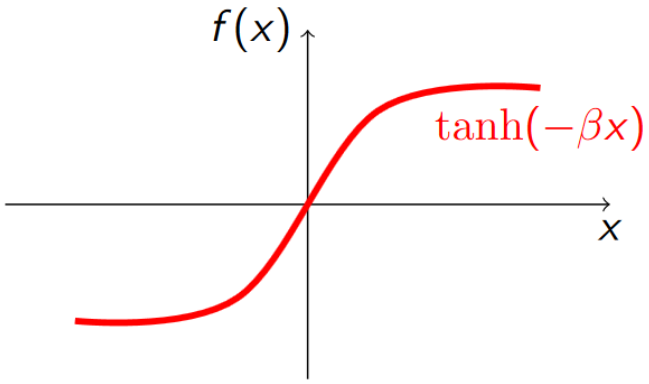


Рисунок 2.1.4 Гиперболический тангенс

Такие функции как sign(x) или tanh(-βx) называются биполярными, потому что у них два полюса: ±1. Есть также униполярная функция: $\frac{1}{1+e^{\beta \times x}}$.

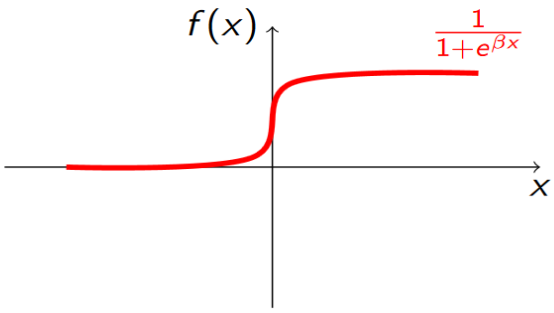


Рисунок 2.1.5 Униполярная функция

Униполярная функция позволяет получить ту же самую нелинейную, похожую на sign(x), картину от 0 до 1. Однако все же большую популярность и распространенность имеют именно биполярные функции.

Также стоит заметить, что на рисунке 1.2 имеется w_0. В модель нейрона часто вводят эту величину, называемую весом активации. Это дополнительный вес, на который всегда подается единичный сигнал. Он как бы соответствует «родному» порогу возбудимости, то есть может быть очень сильно возбудимый нейрон и, если w_0 положительный, то он всегда имеет некую предрасположенность к возбуждению и его, наоборот, надо затормозить, чтобы перевести в стационарное состояние, или, иначе, вес активации может быть отрицательным и тогда надо приложить дополнительные усилия, чтобы нейрон открылся. То есть на самом деле это просто формально добавляется дополнительный вход и считается, что на него всегда подается единичный сигнал, и он не связан с выходами других нейронов и с рефлекторными клетками – на формулу это никак не влияет.

Вот таким образом строятся нейроны. Мы рассмотрели биологические нейроны и довольно грубо смогли построить их модель. На самом деле нужно сказать, что конечно нейронная сеть и нейроны Маккалока и Питса – они не являются хорошими математическими моделями нейронов, то есть нейроны функционируют по более сложным законам и это очень упрощенный подход, который, однако, позволяет добиться хороших результатов. В настоящее время функционирование нейронов тщательно изучено, построены точные дифференциальные уравнения, которые описывают их поведение во времени и, более того, создано программное обеспечение, которое точно позволяет моделировать работу нейронов на каких-то участках нервной системы.

2.1.2 Обучение персептрона

Одним из основных свойств нейронной сети, несомненно, является то, что они могут реализовывать различные логические функции. Для того, чтобы нейрон умел это делать, его необходимо обучать. Способность обучаться – также очень важное и, что не менее важно, полезное свойство нейронов. Они способны сами находить вес, которые позволяют им становится той или иной функцией. Тот процесс, в результате которого обучаются нейроны, называется обучение с учителем. Это означает, что обучение происходит следующим образом: сначала мы задаем вопрос нашему нейрону, то есть подаем на входной вектор, он дает некоторый выход и мы сравниваем этот ответ с правильным. После этого мы проводим обучение нейрона на правильном ответе, то есть для того, чтобы обучить нейрон, нам нужно знать правильный ответ, знать какое правильное соответствие между входом и выходом. Зная это соответствие, мы можем научить нейрон его реализовывать.

Впервые процесс обучения персептрона предложил Розенблатт, именно он понял каким образом мы можем обучать нейроны, настраивать их веса не путем решения системы линейных неравенств, а путем итерационного процесса обучения с учителем – посмотрели на вопрос, выдали ответ,

сравнили с правильным, обучились на правильном ответе. Итак, рассмотрим такой нейрон:

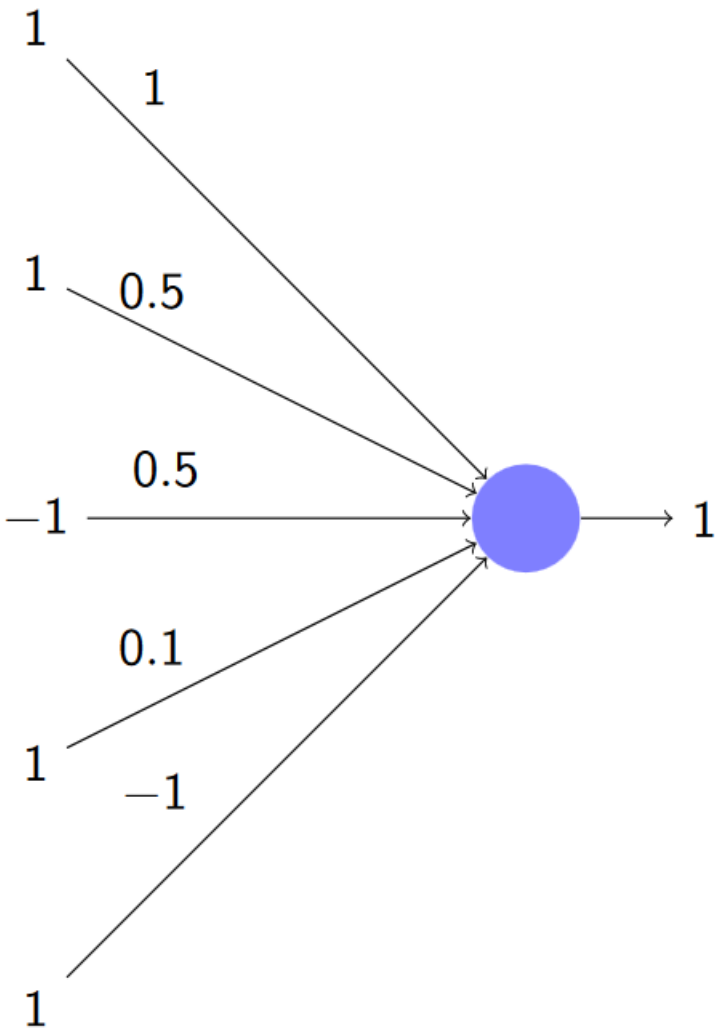


Рисунок 2.2.1 Обучение нейрона

У этого нейрона есть пять входов с разными весами и на эти входы переданы следующие значения: 1, 1, -1, 1, 1. Посчитав выход этого нейрона по формуле взвешенного произведения, мы получаем значение 1. Допустим этот ответ неправильный, а верным является результат – -1. В такие моменты довольно часто применяется аналогия «советчика»: нейрон принимает решение, основываясь на сигналах, то есть «советах от кого-то на тему какой ответ следует дать». На картинке видно, что четверо из пяти «советчиков» советуют дать ответ 1, веса характеризуют степень доверия каждому

«советчику», то есть вот первому «советчику» мы доверяем. Мы слушаем, что он говорит и стремимся сказать именно это. Второго мы тоже слушаем, но не так хорошо, допускаем, что он может ошибаться. Точно также с мнением третьего «советчика», четвертому мы почти не доверяем, а пятому мы доверяем со знаком минус, то есть мы слушаем, что он говорит и делаем наоборот. Это гораздо лучше, чем нулевой вес, который означает, что мы вообще не слушаем данного «советчика». А когда вес отрицательный, то это означает, что в поведении нашего советчика есть какая-то закономерность – он всегда советует неправильно и это та информация, которой мы можем руководствоваться при принятии решений.

Итак, мы «послушали» наших «советчиков» и поняли, что ошиблись. В этом мы, естественно, виним «советчиков», потому что именно на основании их мнения мы сделали свой ответ. Нам придется распределить между ними «вину»:

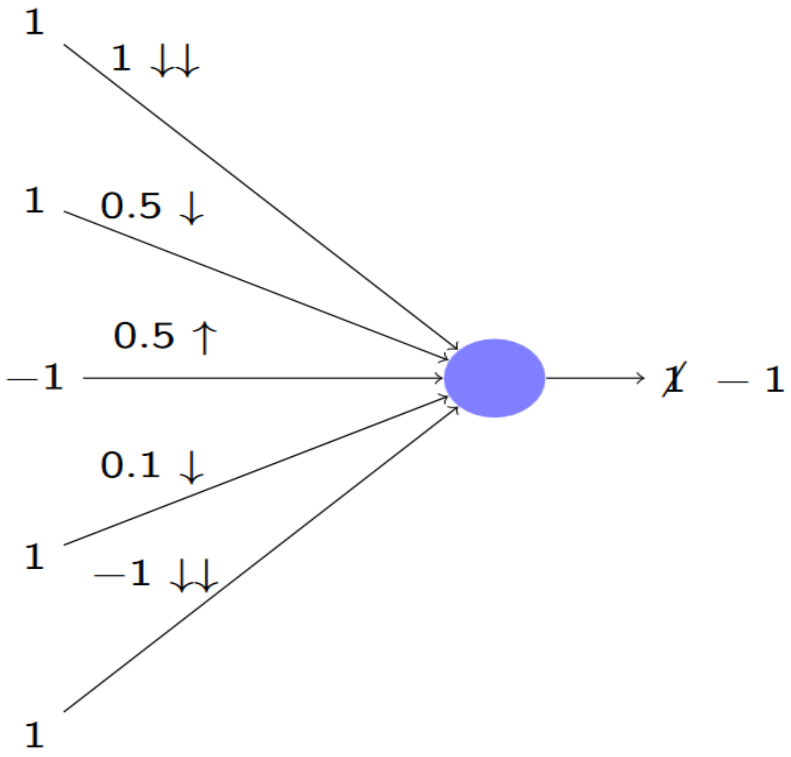


Рисунок 2.2.2 Обучение персептрона

Как видно на рисунке 1.2.2 мнение о первом «советчике» мы радикально поменяли, потому что степень доверия была высокой, но не оправдалась. Второго «советчика» мы тоже понизили в доверии, но не сильно. Рейтинг третьего, наоборот, повысим – он сказал правду, и мы начали доверять ему сильнее. Степень доверия четвертого тоже уменьшается. Пятому мы начали доверять значительно меньше, чем раньше, но это именно то, чего мы от него ждали.

В целом такое направление изменения весов совершенно логично, то есть тех «советчиков», кто советовал нам хорошо, мы начали слушать больше, тех, кто советовал плохо – мы меньше доверяем их мнению. Эту концепцию можно записать с помощью такой формул:

$$\mathrm{d} = \mathrm{a} - \mathrm{y} = -2,$$

которая показывает направление обучения, где a – правильный ответ, y – ответ сети, а также

$$\Delta w_i = \varepsilon \times d \times x_i \times |w_i|,$$

при помощи которой мы можем вычислить изменение весов.

Направление обучения показывает нам насколько мы ошиблись и ошиблись ли вообще. В данном случае, если бы мы угадали правильный ответ, то направление обучения равнялось бы нулю, то есть учится больше не пришлось бы, но мы не угадали и поэтому результат равен -2. Далее, мы изменяем веса по формуле изменения весов, где $d \times x_i$ будет положительным в случае, если совет был правильным, и отрицательным – если неправильным, дальше мы умножаем эту величину на модуль веса, для того, чтобы сильнее уронить доверие к тем «советчикам», которым мы сильно доверяли раньше, как к первому и пятому. А эпсилон – это некое маленькое число, которое нужно для того, чтобы как-то нормировать эффект обучения.

После применения данной формулы, мы получаем примерно такие показатели:

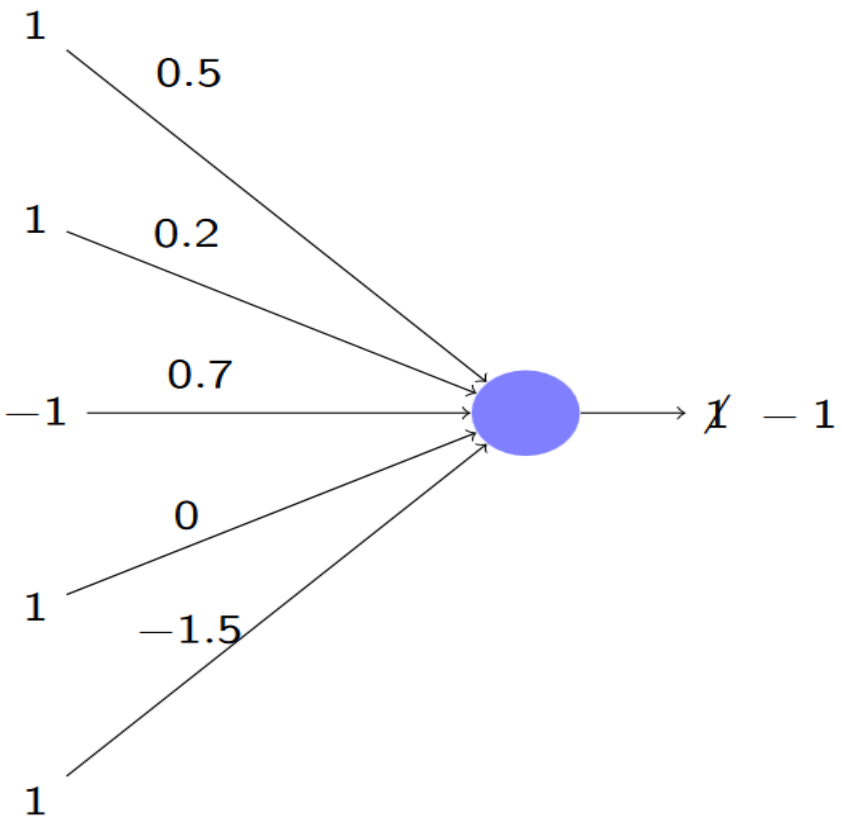


Рисунок 2.2.3 Обучение персептрона

Как мы видим, доверие к первому, второму, четвертому и пятому «советчикам» понизилось, а к третьему возросло. Стоит обратить внимание, что доверие к четвертому «советчику» стало 0, то есть мы больше не будем учитывать его мнение. Таким образом мы обновили веса. Это привело к тому, что на том же самом примере мы получим взвешенную сумму, равную -1.5, sign(x) от которой равен -1. То есть мы научились давать правильный ответ за счет этого изменения весов.

В этом и состоит суть обучения нейрона – сравниваем полученный ответ с ожидаемым, выясняем, что нам нужно делать с доверием к «советчикам» и, собственно, меняем соответствующие веса нейрона, в результате чего, уже с измененными весами, мы на тот же самый вопрос начинаем давать правильный ответ.

2.1.3 Метод обратного распространения ошибки

В предыдущем параграфе мы рассматривали простой пример обучения персептрона, но для решения более сложных и объемных задач требуется сеть из персептронов. Обучить многослойный персептрон гораздо сложнее одного. По своей сути этот процесс рассматривается задача оптимизации функции нескольких переменных. Для этого требуется ввести некоторые обозначения:

Дано:

$\mathcal{X} = (X_1, \ldots, X_k)$	входные вектора, $X_i \in \mathbb{R}^n$
$\mathcal{A} = (A_1, \ldots, A_k)$	правильные выходные вектора, $A_i \in \mathbb{R}^m$
$(\mathcal{X}, \mathcal{A})$	обучающая выборка
W	вектор весов нейронной сети
$N(W, X)$	функция, соответствующая нейронной сети
$Y = N(W, X)$	ответ нейронной сети, $Y \in \mathbb{R}^m$
$D(Y, A) = \sum_{j=1}^{m} (Y[j] - A[j])^2$	функция ошибки
$D_i(Y) = D(Y, A_i)$	функция ошибки на i-ом примере
$E_i(W) = D_i(N(W, X_i))$	ошибка сети на i-ом примере
$E(W) = \sum_{i=1}^{k} E_i(W)$	ошибка сети на всей обучающей выборке

Найти:

вектор W такой, что $E(W) \to \min$ (обучение на всей выборке)

вектор W такой, что $E_i(W) \to \min$ (обучение на одном примере)

Рисунок 2.3.1 Постановка задачи

В конечном итоге задача обучения нейронной сети сводится к тому, чтобы свести ошибку сети на всей обучающей выборке и на одном конкретном примере к минимуму. Возможно нам не удастся идеально обучить нейронную сеть, то есть добиться того, чтобы функция ошибки была в точности равно 0, как правило, этого практически невозможно добиться, поэтому мы минимизируем ошибку нейронной сети путем подбора правильного вектора весов. Получается, что обучить нейронную сеть – это подобрать веса таким образом, чтобы минимизировать ошибку на всей обучающей выборке.

Одним из самых распространенных и популярных методов обучения многослойной нейронной сети является метод обратного распространения ошибки. Впервые открыл этот алгоритм Пол Вербос, а популяризировал Дэвид Румельхарт. Благодаря этим ученым, стало возможным практическое

применение нейронных сетей, потому что нейронные сети из одного слоя могут решать маленькое количество задач, а из многих слоев – большое. Какое-то время было непонятно как обучать такие сети из нескольких слоев, но алгоритм обратного распространения ошибки нам такую возможность предоставляет. Необходимо отметить, что данный метод обратного распространения ошибки – это алгоритм вычисления градиента для того конкретного случая, когда функция является нейронной сетью.

Следует понимать, что алгоритм обратного распознавания весов применим как для задач, в которых используется однослойный персептрон, так и для случаев, в которых задействован многослойный персептрон. И также перед тем, как начать пользоваться методов, требуется знание о методе градиентного спуска и о том, что такое производная сложной функции.

Итак, для того, чтобы понять и разобраться как работает данный алгоритм, рассмотрим следующую нейронную сеть:

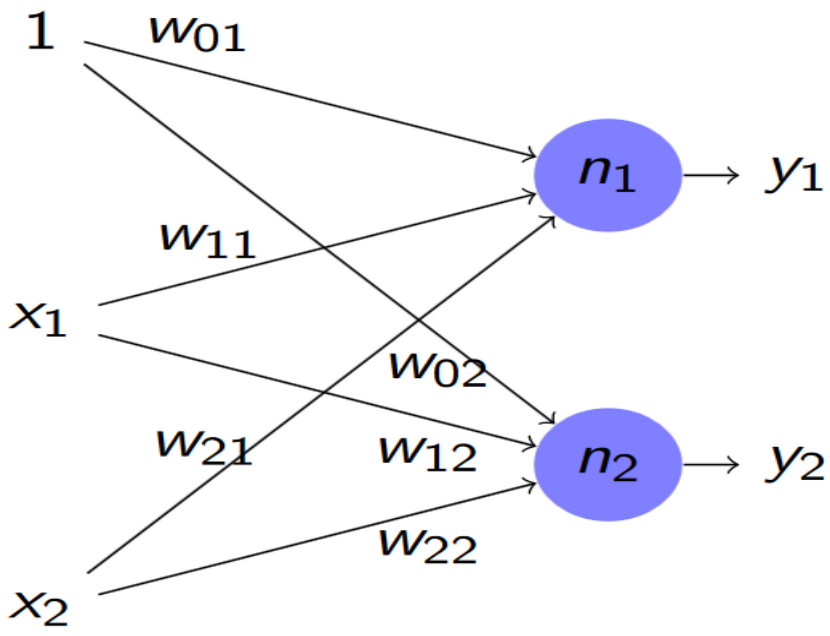


Рисунок 2.3.2 Обратное распространение ошибки

На рисунке 1.3.2 изображена однослойная нейронная сеть с двумя выходами. Необходимо вычислить градиент функции ошибки, с помощью которого мы минимизируем функцию ошибки и, тем самым, подберем те веса, на которых она минимальна, а минимальна она будет тогда, когда выход практически совпадает с правильным ответом, то есть мы подберем веса так, чтобы нейронная сеть научилась давать выход, совпадающий с правильным ответом. Это и будет ее обучением.

Итак, ранее на рисунке 1.3.1 мы вводили понятие функции ошибки. Напомню, что функцию ошибки мы определили, как сумму разностей координат векторов – чем векторы ближе друг к другу, тем меньше значение ошибки на этих векторах. Нейронная сеть на рисунке 1.3.2 на выход выдает два числа, то есть выходной вектор и вектор правильного ответа имеют размерность 2. Сумма квадратов разностей координат этой сети равна:

$$D_k(y_1, y_2) = (y_1 - a_1)^2 + (y_2 - a_2)^2$$

Дальнейшие наши действия будут выглядеть так:

$$D_k(y_1, y_2) = (y_1 - a_1)^2 + (y_2 - a_2)^2$$

$$\frac{\partial D_k}{\partial y_1} = 2(y_1 - a_1) \qquad \frac{\partial D_k}{\partial y_2} = 2(y_2 - a_2)$$

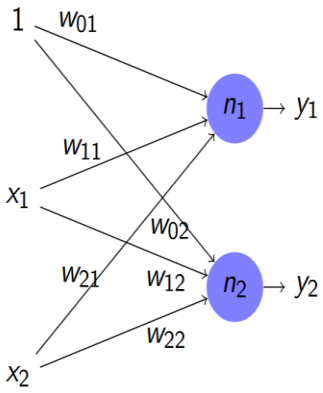
$$\frac{\partial y_1}{\partial w_{21}} = f'(S_1)x_2 \qquad \frac{\partial y_2}{\partial w_{21}} = 0$$

$$E_k(W) = D_k(y_1(w_{01}, w_{11}, w_{21}), y_2(w_{02}, w_{12}, w_{22}))$$

$$\frac{\partial E_k}{\partial w_{21}} = \frac{\partial D_k}{\partial y_1}\frac{\partial y_1}{\partial w_{21}} + \frac{\partial D_k}{\partial y_2}\frac{\partial y_2}{\partial w_{21}} = 2(y_1 - a_1)f'(S_1)x_2$$

Рисунок 2.3.3 Обратное распределение ошибки

Итак, что тут выполняется? Первым делом мы считаем частную производную от нашей функции ошибки по $y_1(y_2)$. Дальше, учитывая, что результат нейронной сети – это функция активации, выраженная формулой взвешенного произведения, мы считаем частную производную функции активации по, к примеру, весу w_{21}. После проведенных операций вычисляем частную производную от функции $E_k(W)$ по этому весу, пользуясь формулой производной от сложной функции. Таким образом, эти формулы дают нам уже сейчас возможность вычислить градиент функции ошибки по весам нейронной сети, когда она состоит из одного слоя. В данном случае у сети шесть весов, по одному весу мы вычислили производную, а по остальным она вычисляется полностью аналогично. Чтобы в этом убедиться, сделаем этот вывод еще раз, но только уже в общем виде:



$$D_k(y_1, \ldots, y_n) = (y_1 - a_1)^2 + \ldots + (y_n - a_n)^2$$

$$\frac{\partial D_k}{\partial y_i} = 2(y_i - a_i)$$

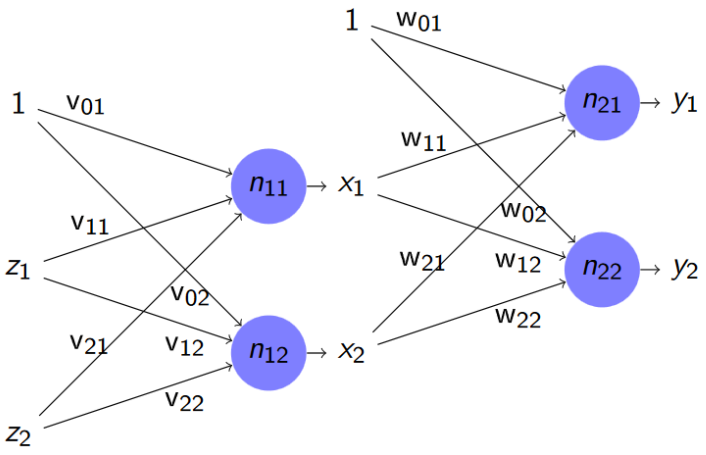
$$S_i = \sum_{j=0}^{m} x_j w_{ji} \qquad y_i = f(S_i) \qquad \frac{\partial y_i}{\partial w_{ji}} = f'(S_i)x_j$$

$$E_k(W) = D_k(y_1(w_{01}, \ldots, w_{mn}), \ldots, y_n(w_{0n}, \ldots, w_{mn}))$$

$$\frac{\partial E_k}{\partial w_{ji}} = \sum_{l=1}^{n} \frac{\partial D_k}{\partial y_l} \frac{\partial y_l}{\partial w_{ji}} = 2(y_i - a_i) f'(S_i) x_j$$

Рисунок 2.3.4 Обратное распространение ошибки

Вот таким образом, применяя производную сложной функции и правило вычисления частных производных, мы можем посчитать градиент функции ошибки по весам нейронной сети в том случае, когда она состоит из одного слоя. Для многослойного персептрона все будет выглядеть так:



$$E_k(W) = D_k(y_1, \ldots, y_n) \quad y_i = y_i(x_1, \ldots, x_m) \quad x_j = x_j(v_{0j}, \ldots, v_{rj})$$

$$\frac{\partial E_k}{\partial v_{rs}} = \sum_{j=1}^{m} \frac{\partial D_k}{\partial x_j} \frac{\partial x_j}{\partial v_{rs}} \quad \frac{\partial D_k}{\partial z_l} = \sum_{j=1}^{m} \frac{\partial D_k}{\partial x_j} \frac{\partial x_j}{\partial z_l}$$

Рисунок 2.3.5 Обратное распространение ошибки

То есть, для выхода сети мы считаем функцию ошибки, далее, применяя формулу производной сложной функции, мы получили функцию ошибки по входам нейронной сети – было известно по выходам, стало известно по весам и по входам. Но эти входы – это выходы предыдущего слоя, известны распределения ошибки по выходам, а значит можно вычислить функцию ошибки по весам и по входам. Таким образом, можно совершенно спокойно применять те же самые формулы, которые мы применяли для случая однослойного персептрона, потому что этот алгоритм обратного распространения ошибки позволяет распространять эту ситуацию от выхода к предыдущему слою полностью – считается распределение ошибки по весам предыдущего слоя и затем вычисляется распределение ошибки по выходам предпредыдущего слоя и так далее. Вот такой алгоритм обратного распределения ошибки, который позволяет нам вычислить градиент ошибки по всем весам нейронной сети – по выходному слою, по предыдущему слою и так далее и так далее.

2.1.4 Классификация и распознавания образов

Задача классификации и распознавания образов подразумевает в первую очередь распознавание образов. Не обязательно, чтобы речь шла о визуальных

образах, очень часто многие ошибочно считают, что распознавание образов – это распознавание изображений и задача сводится к тому, чтобы посмотреть на изображение и понять, что на нем нарисовано. Но это лишь частный случай задачи распознавания, общий случай определяет образ как вектор в n-мерном пространстве. Этот вектор называется вектором признаков, к примеру, для списка пациентов можно выделить признаки: температура, давление и так далее. В таком примере каждый пациент превращается в вектор и это не единственный пример, который можно привести для представления вектора задачи распознавания.

Изображения тоже превращаются в вектор: мы берем картинку и каждый пиксель превращаем, например, в три числа RGB. У нас есть база векторов и необходимо провести распознавание этих векторов, то есть определить по каждому вектору к какому классу он относится. Например, у нас могут изображения, содержащие мяч или кубик, и, посмотрев на вектор, нужно определить к какому из этих классов относится вектор. Нейронные сети учатся распознаванию на прецедентах: у нас уже есть накопленная статистика, по которой мы знаем, что те или иные векторы относятся к таким-то классам, а другие векторы относятся к иным. Грубо говоря, у нас есть накопленная статистика, которую мы хотим изучить, и далее, по результатам этого изучения, диагностировать уже новые векторы. То есть у нас есть знания о том, что было раньше, и мы хотим использовать эти знания для того, чтобы выносить решения по тому, что есть сейчас. На задачу распознавания можно посмотреть, как на регрессию целочисленной функции, то есть функции, которая на вход получает вектор, а на выходе дает число (0, 1, 4, 5, 10 и так далее), соответствующее номеру класса. Но такой подход весьма неудобен, ведь наша цель облегчить работу нейронной сети, а в данном случае получается, что нейронная сеть должна не только решить сложную задачу разбиения векторов на классы, но еще и научиться кодировать все это в довольно-таки странном виде. Поэтому, как правило, у нас есть десять

классов, то делают нейронную сеть, у которой десять выходов, на каждый из которых подается сигнал -1, если вектор не принадлежит этому классу, и 1, если принадлежит – мы учим нейронную сеть, исходя из этих соображений. Соответственно, после того, как мы начинаем ее использовать, мы подаем на нее вектор, снимаем показания с выходных нейронов и относим вектор к тому классу, у которого сигнал, соответствующего ему выходного нейрона, больше. Все эти манипуляции мы выполняем для облегчения жизни нейронной сети. Когда мы требуем от сети выдать число от -1 до +1, то фактически это реализация некой сложной булевой функции, то есть посмотреть на вектор и определить его принадлежность к тому или иному множеству. Когда мы требуем выдать одно число, то мы надеемся провести не только процедуру классификации, но еще потом суммировать все выходы в одном нейроне – это усложняет задачу и для нас это нежелательно.

У нас может быть реальная задача: распознать зашумленные символы на, к примеру, табличках с номерами домов и так далее. Мы понимаем, что собирать большое количество подлинных фотографий (а требуется действительно большое количество фотографий, потому что нейронов много и входов много, и функция получается о 256 аргументах, для обучения которой требуется много прецедентов) не получится. Для этого мы делаем генератор похожих на настоящие фотографий, которые создаются по тем же принципам, что и настоящие: берутся буквы, поворачиваются под различными углами, накладывается на них белый шум и так далее. Далее, мы обучаем нейронную сеть на этих сгенерированных образах, а потом применяем нейронную сеть для анализа настоящих. И это работает, потому что сгенерированные образы похожи на настоящие, они реализуют ту же самую модель зашумления, что и настоящие. Были проведены исследования и эксперименты, на основе которых и было доказано и подтверждено, что такая методика действительно работает – обучить сеть на искусственно-сгенерированных примерах и потом использовать ее для анализа примеров настоящих. Нашей целью является

правильно подделать такой генератор, чтобы он реально выдавал образы с той же самой моделью шума, что существует в реальности. Всегда нужно учитывать следующие проблемы нейронной сети: ее конфигурация, слишком зашумленная обучающая выборка (для конкретно данного примера), переобучение нейронной сети, нерепрезентативная обучающая выборка и так далее.

По всему вышенаписанному в данном параграфе могло сложиться ощущение, что нейронная сеть должна всегда буквально за доли секунды взять и выучить всю обучающую выборку – так не происходит. Обучение нейронной сети – это процесс, который может длиться часами, днями и поэтому в «экспериментаторстве» мы весьма ограничены, потому что необходимо дожидаться результатов эксперимента на реальных данных долгое время, для чего важно получить опыт с работой на простых примерах, чтобы использовать этот опыт на больших. Нужно понимать, что скорость не главный фактор хорошо обученной сети – быстро работают только учебные примеры, а в настоящих условиях эти алгоритмы всегда работают долго.

Рассмотрим также задачу сегментации изображения:

Рисунок 2.4.1 Сегментация

На рисунке 1.4.1 показано игровое поле с точки зрения шахматного робота, к которому была прикреплена камера, обозревавшая стол перед ним и фиксировавшая такую картину, то есть так выглядел изображение на камеру. Конечно, для того, чтобы работать с этим изображением, нам необходимо перевести его в логический вид. Нам необходимо, чтобы все пиксели красной или синей шашки были атрибутированы, соответственно, как синие или красные. То есть на самом деле нам требуется каждому пикселю исходного изображения сопоставить логический класс: стол, желтая, красная, синяя и зеленая шашки. Глядя на каждый пиксель изображения, нужно найти цифру от 0 до 4, которая бы соответствовала классу того объекта, который находится в этом пикселе.

Графическая репрезентация сегментации примерно выглядит так:

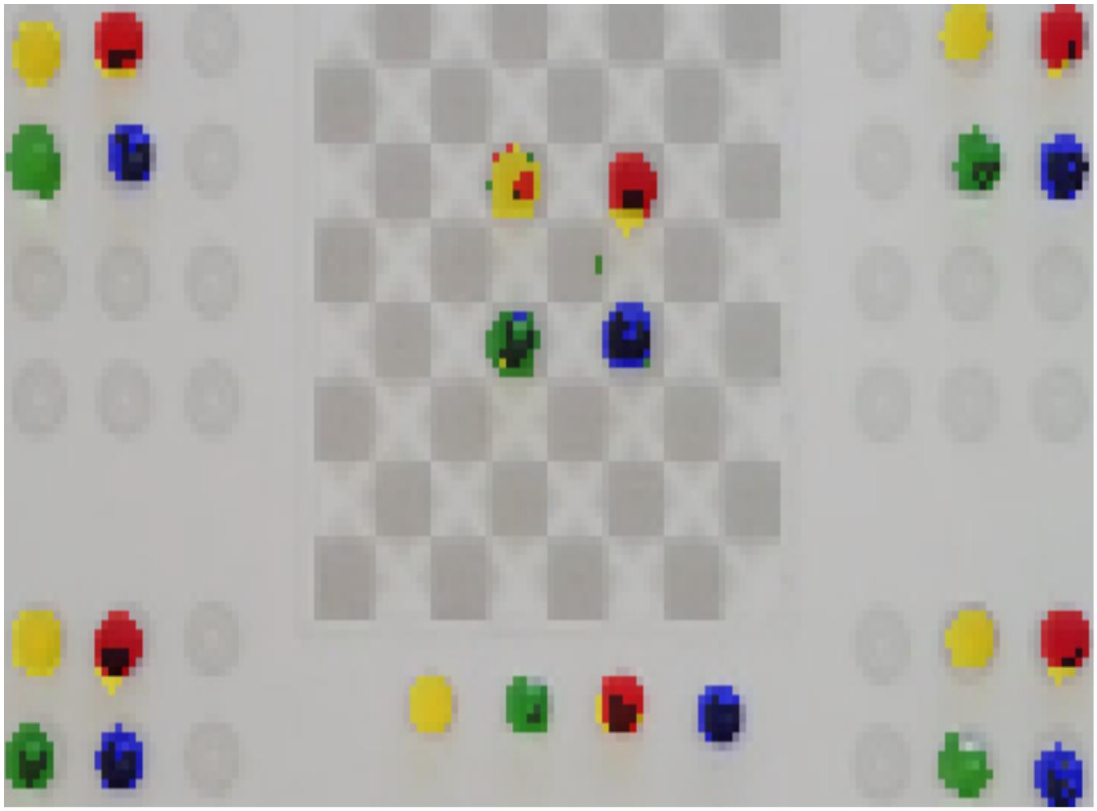


Рисунок 2.4.2 Сегментация

На рисунке 1.4.2 видно, что для каждого пикселя указан его цвет и наложена логическая матрица в виде маски на исходное изображение. То есть видны остатки исходного изображения после наложения маски, но это,

наоборот, позволяет увидеть и оригинал изображения, и ту логическую матрицу, которая из него получилась. Некоторые классы не точно распознаны, это заметно в местах, выделенных черным цветом.

Вот так и выглядит задача сегментации изображения – требуется взять исходное изображение, и каждому пикселю этого изображения сопоставить номер класса того объекта, который расположен в этой точке, то есть мы занимаемся задачей распознавания – мы смотрим на пиксель и определяем, к какому классу он относится.

Очевидным решением является персептронная нейронная сеть с тремя входами (RGB-каналы – три входа конкретно для случая из рисунка 1.4.1 и 1.4.2) и, соответственно, пятью выходами – на каждый класс по одному выходу. Грубо говоря, на вход мы подаем образ, а с выхода мы принимаем, собственно, класс того объекта, который изображен – какой нейрон нам выдал максимальное значение, значит тот класс и изображен. Выходной слой нейронной сети готовит нам ответ, определяя по выходному набору, к какому классу что относится, а вот первый слой осуществляет перевод исходного пространства в пространство признаков. То есть нейронная сеть работает в два этапа: первый этап – это перевод из исходного образа в образ в пространстве признаков, для каждой картинки устанавливается, какими признаками она обладает, а потом, во втором этапе – мы смотрим на признаки изображения и делаем вывод о принадлежности их к определенному классу.

Сейчас немного подробнее поговорим о задаче классификации, которая чем-то напоминает задачу распознавания образов. В задаче распознавания образов у нас есть учитель, который «говорит» нам, к какому классу каждый из образов должен быть отнесен. Образы – это векторы, имеется также обучающая выборка, в которой учитель некоторые образцы векторов отнес к определенным классам. Затем, мы обучаем нейронную сеть на этой обучающей выборке, и, дальше, она способна определить по каждому вектору, к какому из классов он относится. То есть все пространство образов делится

на определенные области, и, затем, когда мы берем какой-то образ, мы определяем, в какой точке пространства он оказался. Однако иногда у нас нет учителя, который бы мог обучить нашу нейронную сеть, бывает, что есть образы и векторы, но нет учителя, который знает к какому классу относится какой вектор. В таком случае на первый план выходит задача классификации – то есть образы мы хотим разделить на те или иные классы, не зная заранее на какие именно классы и по каким критериям. Задача классификации – тот случай, когда у нас нет учителя, но есть данные, которые мы самостоятельно должны разбить на классы, руководствуясь какими-то соображениями.

Какими соображениями? Давайте посмотрим на картинку:

Рисунок 2.4.3 Задача классификации

Нам не нужен учитель, для того, чтобы понять сколько классов выпечки изображено на рисунке 1.4.3. Мы видим, различные образы объектов, идентичные образы и понимаем, что они похожи друг на друга. То есть за счет особенностей нашего восприятия, глядя на визуальные образы, мы определяем, что некоторые из них похожи. И дальше, если попытаться

разложить все печеньки из рисунка 1.4.3 на классы, то с этим без труда можно справиться, кладя элементы выпечки рядом в одну кучу. Еще раз напомню, что это можно делать без учителя, самостоятельно анализируя схожести тех или иных элементов. То же самое делается и математически – у нас есть пространство векторов, есть метрика на этом пространстве, которая говорит насколько два вектора похожи друг на друга, и дальше мы должны выполнить алгоритм, который бы на основании этой похожести разделял векторы на кучки похожих между собой векторов.

Вот в этом и заключается суть задач классификации и распознавания объектов, которые имеют непосредственное отношение к проделанной в данном проекте работе и помогают достичь поставленную цель. В этой подглаве были описаны основные понятия и алгоритмы, касающиеся нейронных сетей. Благодаря своим полезным свойства и возможности обучаться, именно нейронные сети были выбраны, как основной метод для реализации поставленной в работе задачи.

Глава 3. Описание решения поставленной задачи

В данной главе будет содержаться теоретическое представление проделанной работы: описание алгоритмов и методики решения задачи.

3.1 Выбранные для решения задачи средства и алгоритмы

Итак, для достижения и реализации поставленной задачи в качестве основного инструмента реализации была выбрана нейронная сеть вида многослойный персептрон. Все преимущества, фишки и понятия, касающиеся нейронных сетей подробно изложены в предыдущей главе, но, стоит повторить, что выбор конкретно данного вида нейронной сети обусловлен, в первую очередь, удобством и понятностью реализации, и, соответственно, тем, что целью данной работы не является реализация задачи конкретно

определенным способом, что позволяет выбрать любой удобный для нас метод, который будет изучен и реализован и, с помощью которого будет достигнута конечная цель.

Помимо нейронной сети используются, разумеется, и вспомогательные алгоритмы, роль которых очень важна для решения задачи, так как без них цель не может быть достигнута. Итак, в первой главе было сказано несколько слов об алгоритме Виолы-Джонса и каскадах Хаара, в данной работе эти методы также задействованы, так как позволяют распознать лицо на изображении, что является первым этапом изученного в работе процесса. Среди прочих, также используются методы бинаризации изображения и фильтр Собеля.

Пройдемся на предназначении каждого из методов, кроме нейронных сетей, о которых уже было написано достаточно. Так как об алгоритме Виолы-Джонса было уже сказано не мало в первой главе, то здесь просто будут повторены некоторые наиболее важные моменты и, непосредственно, роль этого алгоритма в данной работе. Итак, алгоритм Виолы-Джонса предназначен для того, чтобы распознавать на изображении определенные признаки, такие как лица, предметы и так далее. В нашем случае нас интересует распознавание лиц. Напомню, что алгоритм Виолы-Джонса в качестве основы использует каскады Хаара, но только более усовершенствованные. При получении входного изображения, по алгоритму, создается сканирующее окно определенного заданного размера, в котором происходит сканирование данного участка изображения на предмет различного расположения признаков Хаара путем изменения их масштаба, после чего сканирующее окно переходит на другой участок изображения, где также проводится сканирование. Алгоритм работает так по всему изображению, последовательно увеличивая масштаб сканирующего окна.

По своей сути, бинаризация изображения и фильтр Собеля предназначены для уменьшения размерности данных, но в данном случае

ключевую роль выполняет именно фильтр Собеля, который выделяет контуры лица и, при помощи внедрения в нее алгоритма бинаризации изображения, преобразует изображение в черно-белые тона, после чего сравнивается яркость каждого пикселя с заданным пороговым значением, по результатам которого пикселям белого цвета присваивается значение 0, а черного – 1. Все вышеописанные алгоритмы также применяют функцию уменьшения размера изображения, которая уменьшает размерность данных. Как видим из вышенаписанного, всю лишнюю информацию мы благополучно теряем, оставляя только самое необходимое.

3.2 Реализация методов

Стоит отметить, что обучающая выборка состоит из 75 изображений лиц людей, 5 эмоций, по 15 изображений на каждую.

Первым делом, на поступившем на обработку изображении, при помощи алгоритма Виолы-Джонса выделяется лицо, и приводится к размеру 40×40 пикселей. О том, как это происходит выше написано предостаточно.

После этого, полученное изображение обрабатывается при помощи фильтра Собеля: переводится в черно-белый цвет, выделяются контуры изображения, уменьшается размерность данных. Фильтр Собеля выполняет часть работы, которую должен выполнять алгоритм бинаризации, и даже облегчает ему работу тем, что на обработку бинаризации уже подается черно-белое изображение с уменьшенной размерностью, так что алгоритму бинаризации остается только перевести данные в числовой вектор из 0 и 1.

Наш многослойный персептрон получает входные данные в размере ширины картинки, помноженной на высоту, а количество выходов равно количеству наших классов-эмоций. При работе сети на выходе получаем вектор из 5 элементов, который сравнивается с «шаблоном». При неудовлетворительном результате, применяя алгоритм обратного

распространения ошибки, сеть учится до тех пор, пока не выдает правильный результат.

Глава 4. Результаты проделанной работы

4.1 Описание продукта

Программный продукт, реализующий систему распознавания эмоций по фотографии, выполненную на языке программирования C# с использованием библиотеки OpenCV.

В системе реализованы следующие модули:

- Модуль распознавания эмоции по фотографии (testing mode);
- Модули обучения (training mode):
    1. Обучение пользователя на правильное распознавание эмоций;
    2. Обучение нейронной сети на подготовленной обучающей выборке по методу «обучение с учителем»;

4.2 Архитектура ПО

В данной подглаве будет повторено все то, о чем уже было написано в предыдущей главе.

В качестве распознающего инструмента в программе используется нейронная сети типа многослойный персептрон.

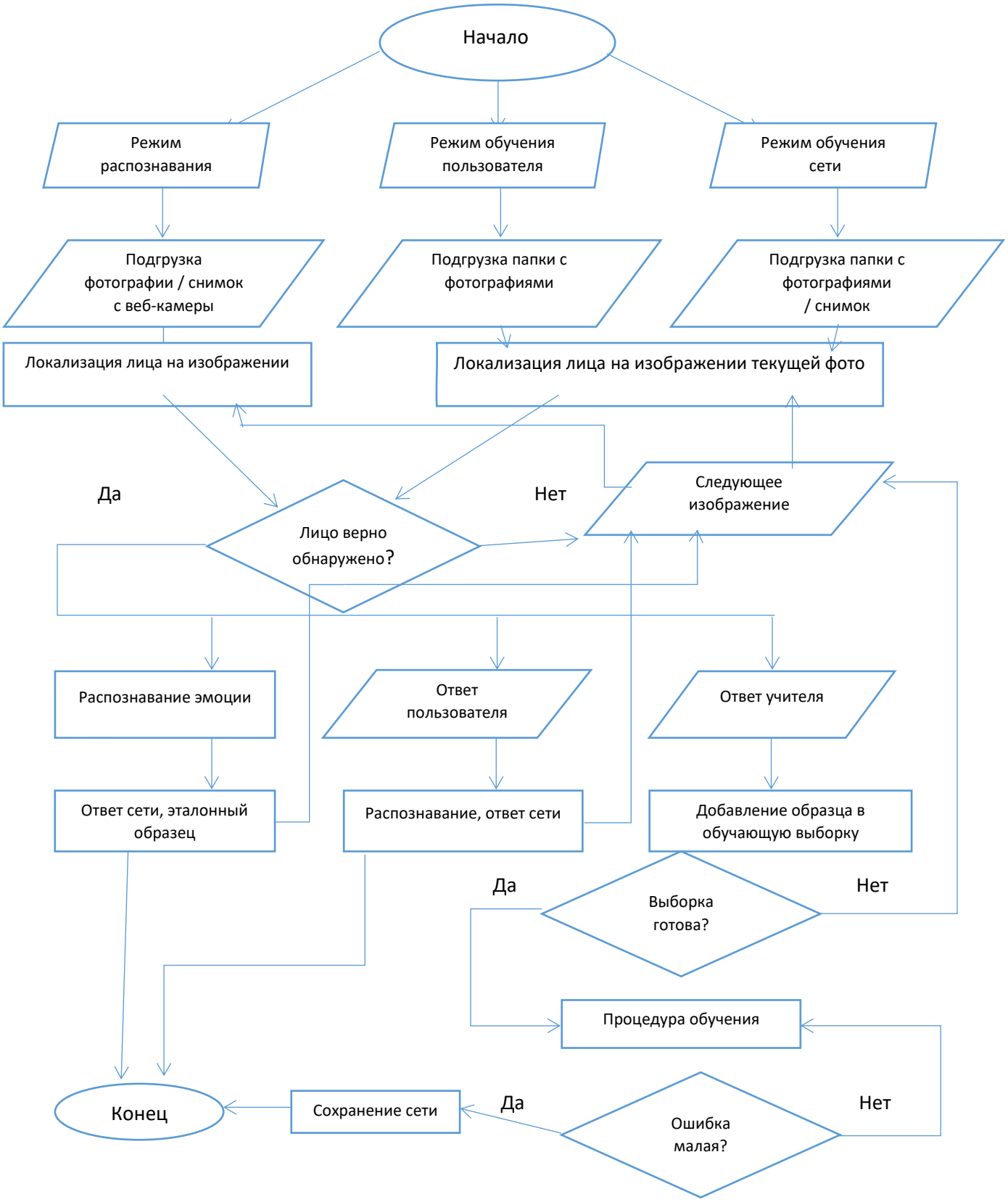
В реализации системы реализованы следующие алгоритмы:

1. Алгоритм Виолы-Джонса – для локализации лица на фотографии. В качестве фильтров используются каскады Хаара.
2. Алгоритм, реализующий фильтр Собеля – для выделения контуров лица. В функцию алгоритма внедрен алгоритм бинаризации изображения, чтобы по применению функции получать сразу черно-белые контуры.

3. Функция уменьшения размера изображения – уменьшает размерность входных данных и делает число входов элементов фиксированным, равным заданной величине.
4. Функция перевода черно-белого изображения в числовой вектор, элементы которого принимают значения 0 и 1, где 0 – белый пиксель, а 1 – черный.
5. Алгоритм обратного распространения ошибки – для обучения нейронной сети.

Перечисленные алгоритмы и функции в программе чередуются в порядке их нумерации.

4.3 Блок-схема работы программы



4.4 Инструкция по работе в программе

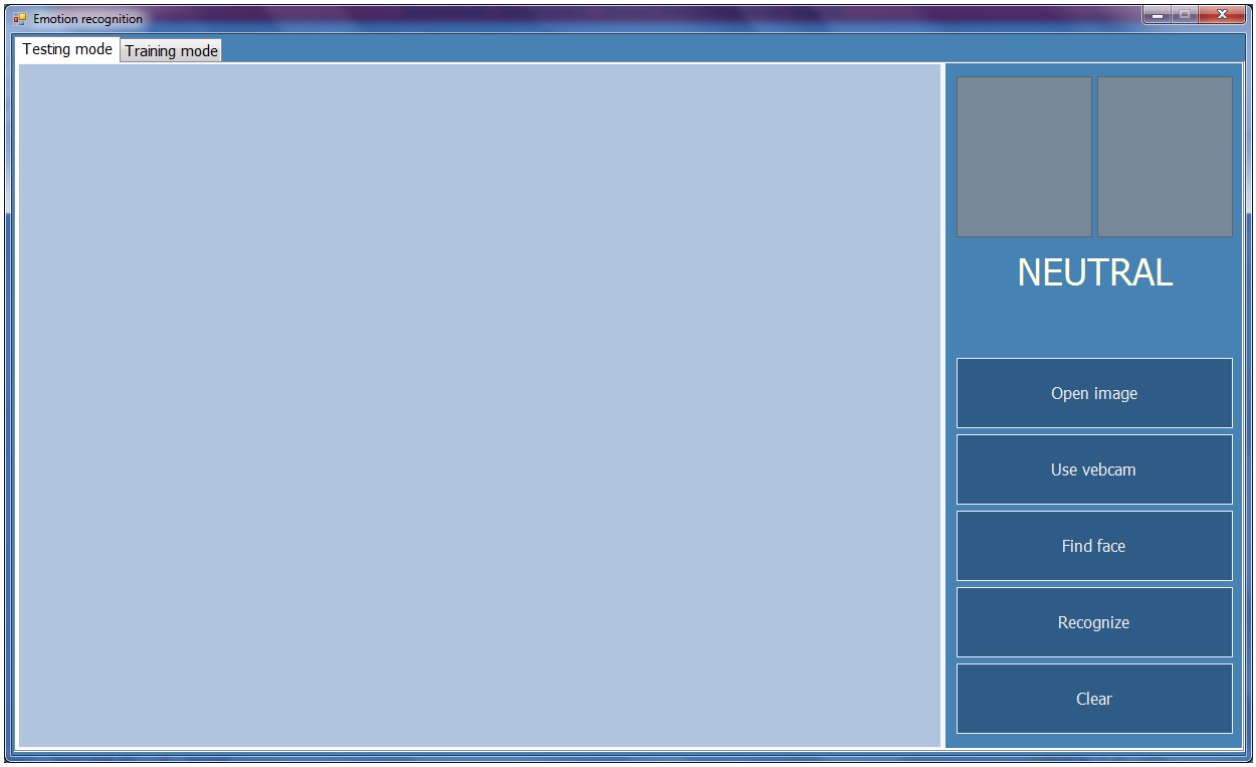


Рисунок 4.1 Окно запущенной программы

После запуска программы отображается начальное окно с режимом распознавания (тестирования ПО) – рис 4.1. Пользователь выбирает режим, в котором хочет работать. Переход на вкладку Training mode переведет программу в режим обучения (Рис. 4.2). В режиме обучения по выбору можно переключаться между режимом обучения пользователя (по умолчанию) и режимом обучения нейронной сети (Рис. 4.3).

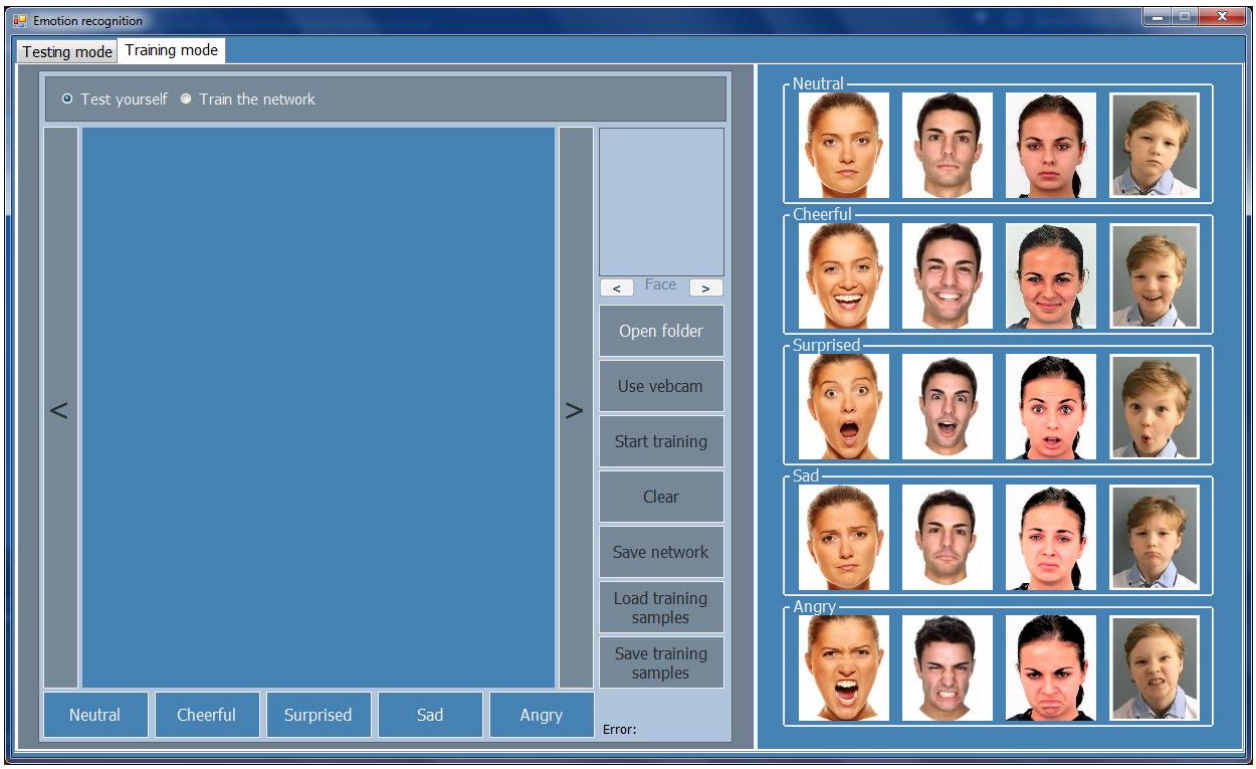


Рисунок 4.2 Режим обучения

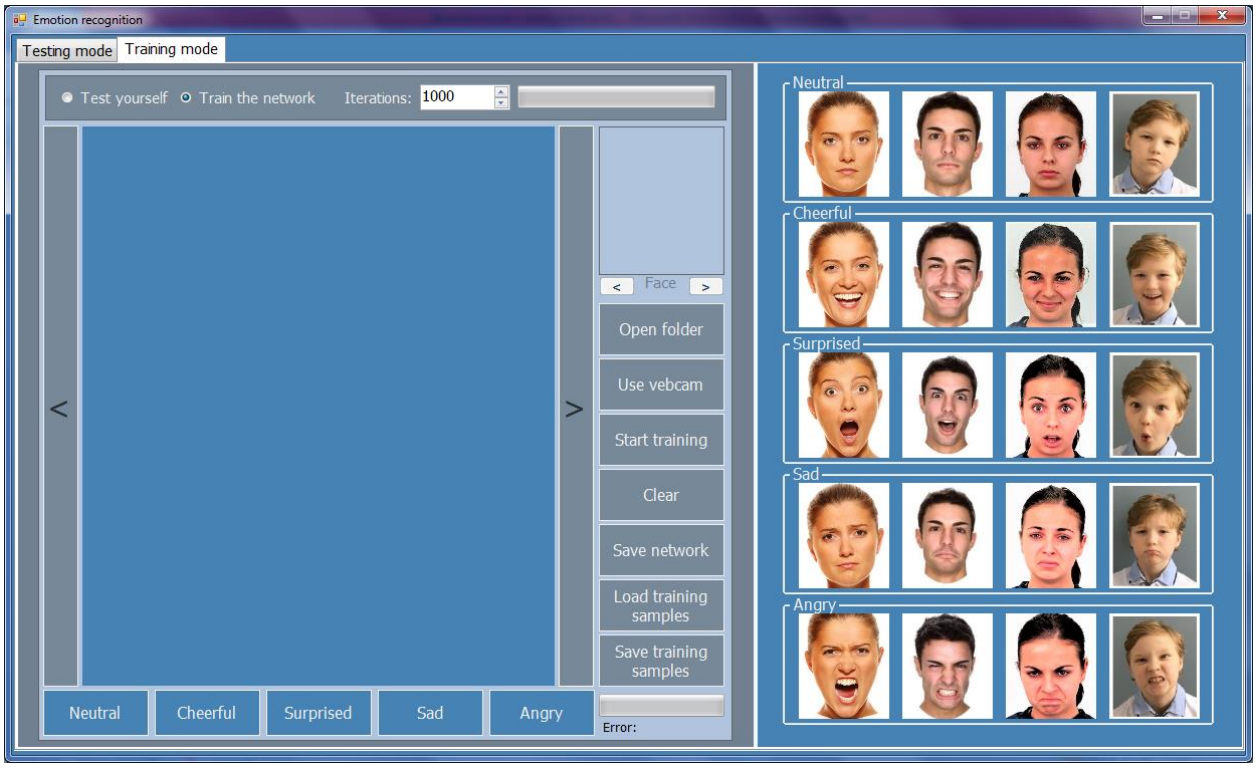


Рисунок 4.3 Режим обучения нейронной сети

Работа в режиме распознавания

1. По кнопке «Open image» подгружается изображение из диалогового окна выбора файла (Рис 4.4).

2. После выбора изображения фотография размещается в панели отображения (Рис. 4.5).

Рисунок 4.4 Подгрузка фотографии

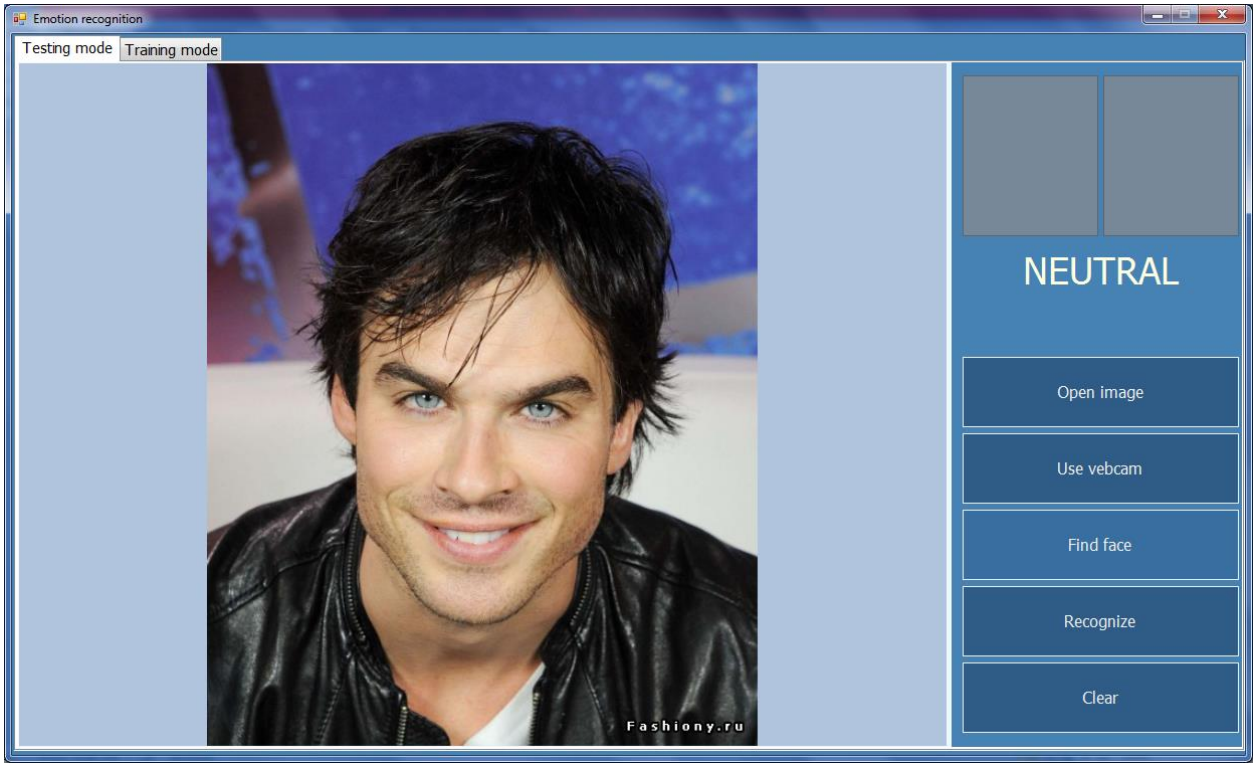


Рисунок 4.5 Отображение выбранной фотографии

Также, фотографию можно получить, сделав снимок с веб-камеры (кнопка Use vebcam). По нажатию кнопки, начнет захват видеопотока с камеры, чтобы сделать фотоснимок, нужно нажать ту же самую кнопку, теперь на ней будет надпись «Get photo».

После того, как пользователь получил в отображении нужную фотографию, необходимо найти лицо (кнопка Find face). Изображение лица отобразится в левом верхнем боксе справа, над табличкой с отображением названия эмоции. Если лицо локализовано корректно, можно получать результаты распознавания (кнопка Recognize). В правом боксе рядом с лицом отобразится эталонный образец распознанной эмоции, а в табличке снизу ее название. Всё это можно увидеть на рисунке 6. По нажатию на кнопку «Clear» все панели и боксы очистятся.

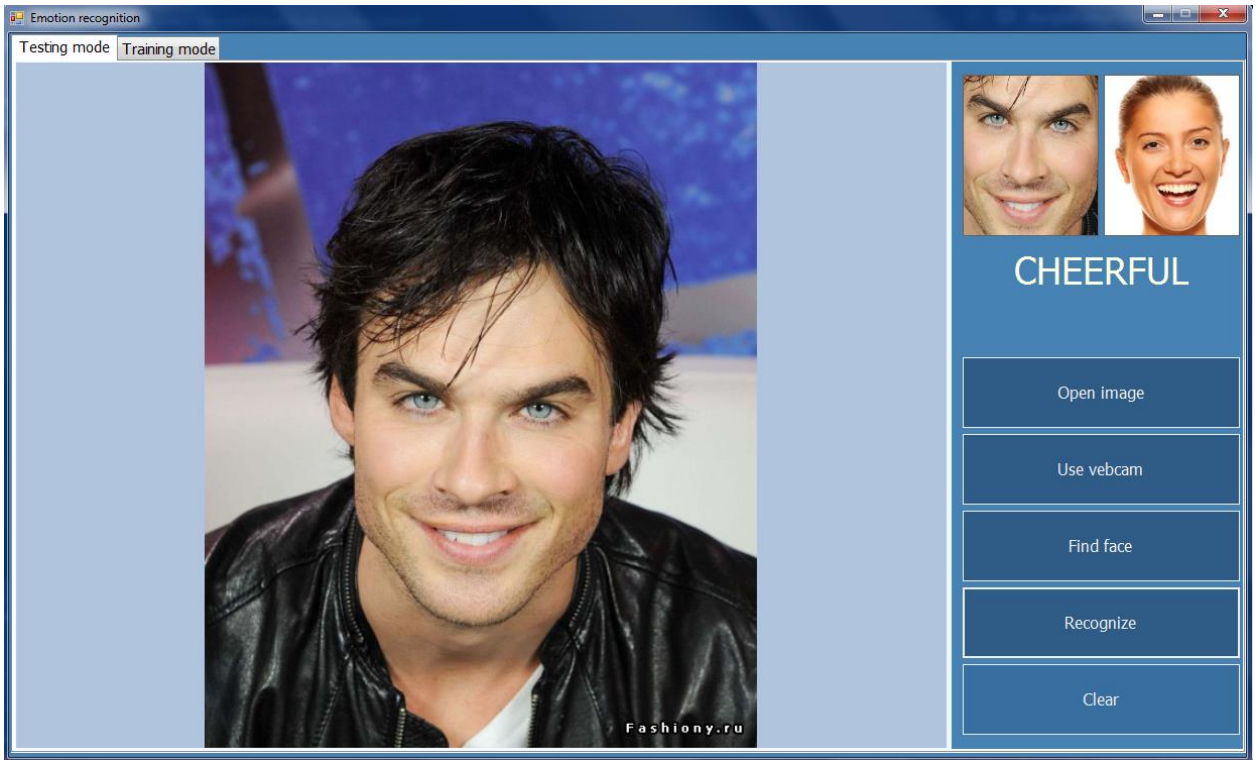


Рисунок 4.6 Результат локализации лица, распознавание эмоции.

Работа в режиме обучения пользователя

В режиме обучения пользователя в первую очередь необходимо подгрузить папку с набором фотографий. Это осуществляется по нажатию кнопки «Open folder». Из каталога компьютера выбирается папка целиком (Рис. 7).

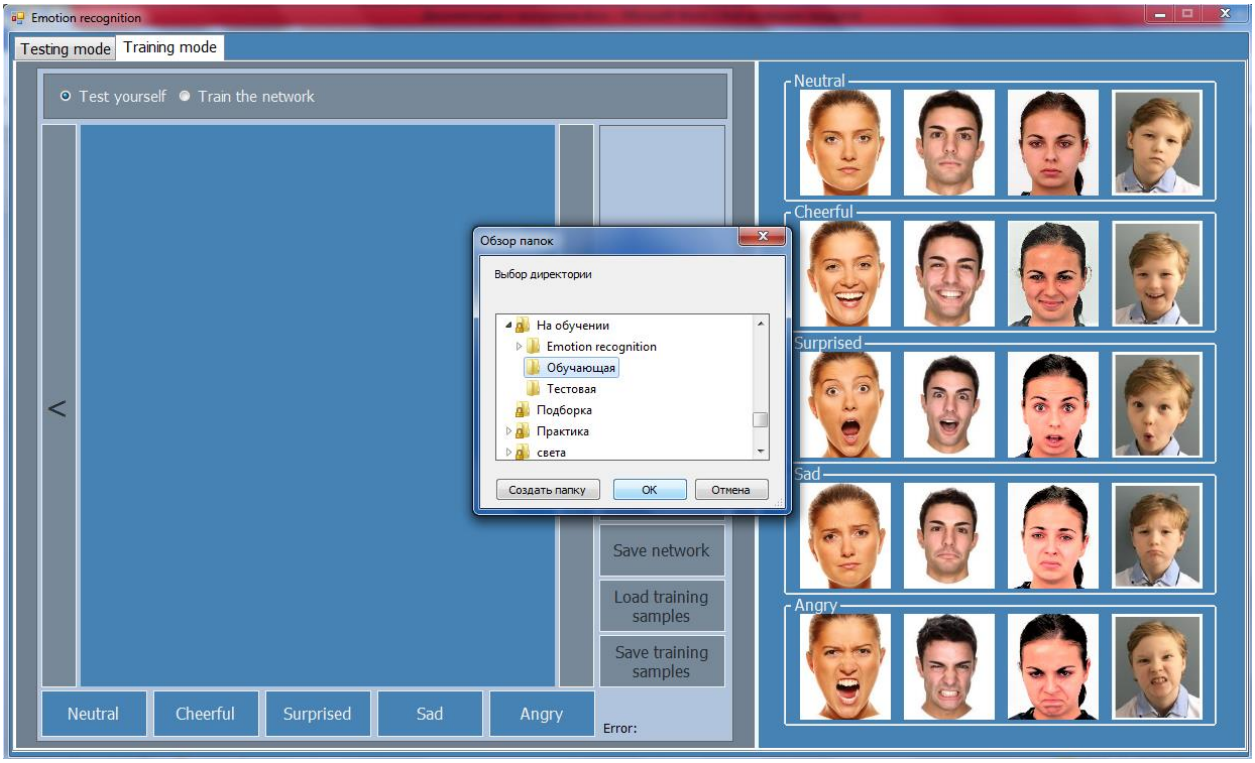


Рисунок 4.7 Выбор папки с изображениями

После выбора папки, первое изображение из нее сразу отобразится в панели. В маленьком боксе вверху сразу можно увидеть локализованное лицо, так что пользователь может сразу понять, будет ли верно обработано программой текущее изображение. В правой части окна можно увидеть несколько примеров эмоций, которые обучаемый может использовать в качестве подсказок. Фотографии увеличиваются при клике по ним. Всё перечисленное видно на рис. 4.8. Снизу окна расположены кнопки для выбора эмоций. В режиме обучения пользователя они служат для интерактивного диалога обучаемого с программой. Если пользователь верно распознал эмоцию, щелкнув на соответствующую кнопку, она подсветится зеленым светом, если неверно – красным, а верная – зеленым, как показано на рис. 8.

Кнопки «<» и «>» (назад и вперед) позволяют переключаться между фотографиями из выбранной папки. То же самое работает и в режиме обучения нейронной сети.

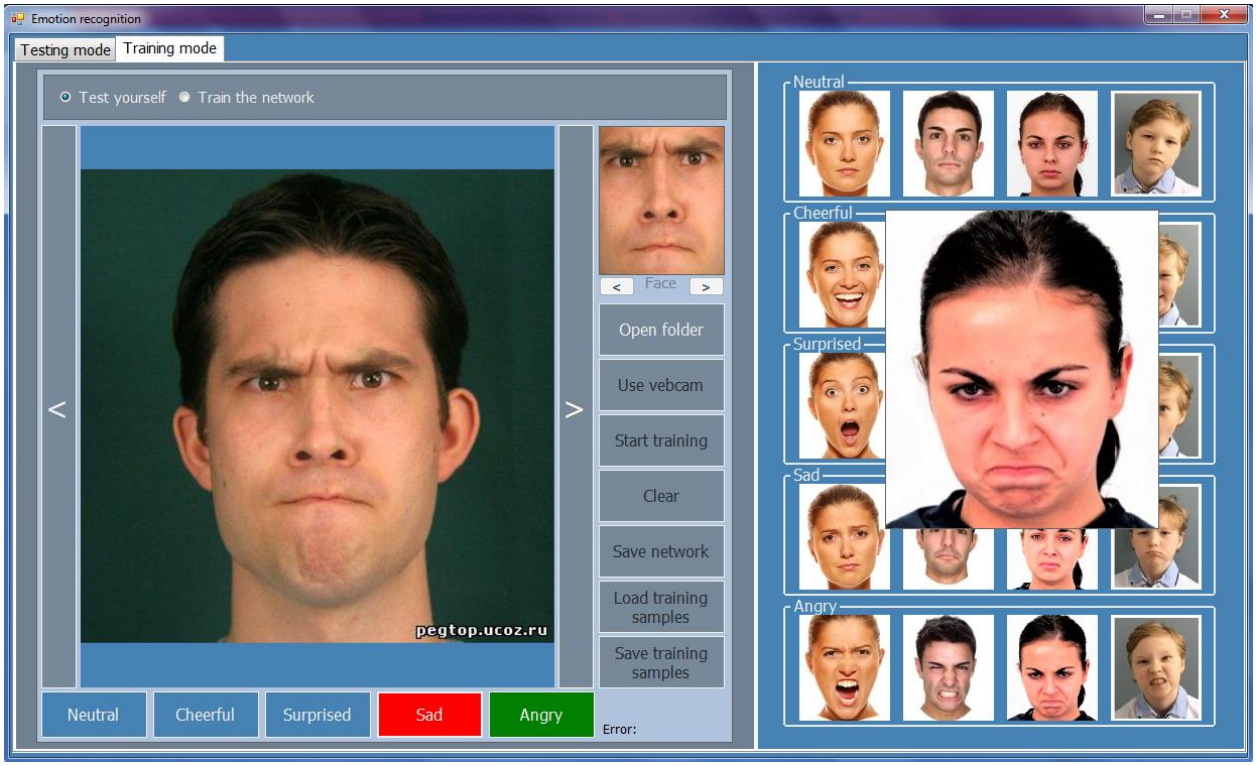


Рисунок 4.8 Работа в режиме обучения пользователя.

Работа в режиме обучения нейронной сети

Подгрузка папки и отображения в этом режиме осуществляются точно так же, как в предыдущем. Кнопки эмоций снизу, однако, теперь выполняют другие функции. После того, как изображение подгружено и корректно отображается, пользователь (теперь уже учитель) по нажатию на кнопку, соответствующую верной эмоции, добавляет таким образом текущий образец в обучающую выборку, после чего переходит к следующей фотографии из набора. Прогресс-бар снизу отображает процесс добавления экземпляра в выборку. После того, как все образцы из набора были добавлены в обучающую выборку, учитель сохраняет ее в файл, нажав при этом кнопку «Save training

samples». В дальнейшем для обучения можно легко подгрузить обучающую выборку нажатием кнопки «Load training samples», и весь набор загрузится сразу, без необходимости обрабатывать все фотографии заново.

После того, как выборка была создана и сохранена, можно посмотреть, как отображаются её обработанные образцы, которые в последующем будут поданы на вход нейронной сети. Эти образцы (маленькие бинаризированные картинки) можно увидеть в боксе сверху при нажатии маленькой кнопки «>» вверху под боксом отображения лица (рис. 9). Если все образцы хорошо различимы, можно начинать обучение по кнопке «Start training». Прогресс-бар вверху сообщает о процессе обучения и его завершении. По умолчанию программа проведёт 1000 итераций обучения, это число можно изменить вверху в поле «Iterations».

Также, в этом режиме тоже можно использовать веб-камеру и добавлять в выборку изображения из снимков.

После того, как обучение завершилось, внизу появится информация об ошибке на момент окончания обучения, например, Error: 0.1. Если величина ошибки удовлетворительная, обученную нейронную сеть можно сохранить, нажав кнопку «Save network». При запуске программы каждый раз подгружается имеющаяся нейронная сеть.

Кнопка Clear будет очищать панель и бокс отображения и зачищать несохраненные данные.

Рисунок 9. Процесс обучения на 1000 итераций

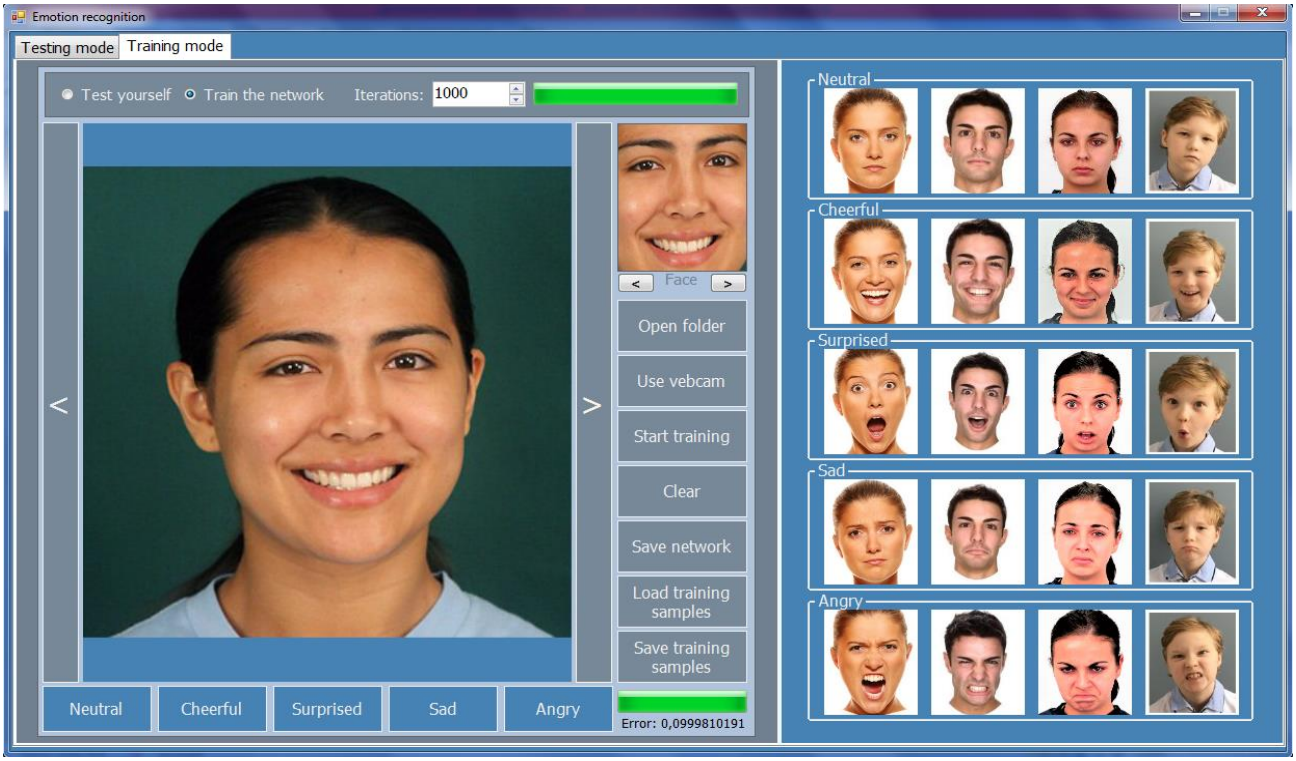


Рисунок 10. Результат обучения. Значение ошибки.

Выводы

В рамках проделанной работы были получены следующие результаты:

- Была спроектирована программа для распознавания и тренировки эмоций на языке C#;
- Удалось добиться приемлемого уровня точности для имеющейся обучающей выборки;
- Реализованы все вспомогательные алгоритмы: алгоритм Виолы-Джонса, алгоритм бинаризации изображений и фильтр Собеля;
- Успешное обучение нейронной сети;

Стоит также отметить, что точность распознавания составила 90%, что является весьма неплохим результатом. Однако не всегда этот показатель точности работает на примерах не из обучающей выборки. Это зависит от количества нейронов в скрытых слоях сети. Если нейронов больше, чем нужно, то сеть может хорошо выучить образцы из выборки, но будет плохо распознавать то, чего нет в обучающем наборе. Если же нейронов меньше, чем нужно, то сеть не сможет обучиться, поэтому оптимальное количество нейронов лежит между этими значениями. Таким образом подбирать количество нейронов требуется экспериментально.

Заключение

Распознавание эмоций является актуальной задачей в различных сферах человеческой деятельности. На данный момент разработано достаточно много методик и алгоритмов, которые позволяют решать такие цели. В данной работе было предложено комплексное решение поставленной задачи в виде обучения нейронной сети. Был разработан понятный и легко осваиваемый интерфейс, с помощью которого можно не просто распознавать эмоции по

фотографиям, но и проверять себя на предмет их знаний. Результаты проделанной работы оказались приемлемыми, но, в перспективе, они способны только улучшаться при большем объеме обучающей выборки и более удачном подборе количества нейронов в сети.

Список литературы

1. P. Viola and M.J. Jones, «Rapid Object Detection using a Boosted Cascade of Simple Features», proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001
2. Местецкий Л. М., «Математические методы распознавания образов», МГУ, ВМиК, Москва, 2002–2004
3. Буй Т.Т.Ч., «Алгоритмы распознавания лиц и жестов на основе вейвлет-преобразований и метода главных компонент», 2011
4. Р.Гонсалес, Р.Вудс, «Цифровая обработка изображений», ISBN 5-94836-028-8, изд-во: Техносфера, Москва, 2005
5. Viola and M. Jones. Robust real-time face detection. IJCV 57(2), 2004
6. Лалаян К.К., «Распознавание эмоций и классификация пола человека в реальном времени», 2016
7. James. The Principles of Psychology. — New York: H. Holt and Company, 1890
8. Друки А.А., «Система поиска, выделения и распознавания лиц на изображениях», 2011
9. Аркадьев А. Г., Браверман Э. М. «Обучение машины распознаванию образов», 1964
10. Л. Шапиро, Дж. Стокман, «Компьютерное зрение», 2006
11. Вапник В.Н., Червоненкис А.Я., «Теория распознавания образов», 1974
12. Duda R., Hart P., «Pattern Classification and Scene Analysis», 1973

Ссылка на документ в интернете

6. https://habrahabr.ru/post/278435/

7. https://habrahabr.ru/post/133826/
8. https://habrahabr.ru/post/322392/
9. http://it-claim.ru/Persons/Semenova_Yana/SemenovaPlakats.pdf
10. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.420.7883
11. https://computersciencesource.wordpress.com/2010/09/03/computer-vision-the-integral-image/
12. https://habrahabr.ru/post/198338/
13. http://neurobotics.ru/phychophysiology/emodetect

Приложение

```
double[] inputarray = new double[inpcount]; // массив входного вектора
double[][] inputvectors = new double[0][]; // массив набора входных векторов обучения
double[][] outputvectors = new double[0][]; // массив набора выходных векторов обучения
double[] neutralvector = new double[5] { 1, 0, 0, 0, 0 }; // вектор нейтральной эмоции
double[] cheerfulvector = new double[5] { 0, 1, 0, 0, 0 }; // вектор эмоции радости
double[] surprisedvector = new double[5] { 0, 0, 1, 0, 0 }; // вектор - удивление
double[] sadvector = new double[5] { 0, 0, 0, 1, 0 };  // вектор - грусть
double[] angryvector = new double[5] { 0, 0, 0, 0, 1 }; // вектор - гнев

//--------------------------------------------------------------------------------вектора для нейросети
```

```
Image<Bgr, Byte> image = null; // исходное изображение
Bitmap faceImage; // для изображения лица

int pathindex = 0;
int k = 0;
int j = 0; // вспомогательные счётчики
byte camera = 0;

VideoCapture myCapture = new VideoCapture(); // переменная видеопотока
CascadeClassifier faceCascade = new CascadeClassifier("haarcascade_frontalface_default.xml"); // каскадный классификатор Хаара
Image<Gray, Byte> grayImage; // серый фильтр фотографии

string path = null; // путь к файлу
string[] paths = null; // массив путей
int Fcount = 0; // сколько файлов в папке подгружено

int maxindex = 0; // индекс элемента верного ответа

//--------------------------------------------------------------------------------глобальные переменные
```

```
public void GetInputVector()
{
    inputarray = new double[inpcount];
    faceImage = Sobel(faceImage);
    faceImage = ResizePicture(faceImage, size, size);
    inpindex = 0;

    for (int x = 0; x < size; x++)
    {
        for (int y = 0; y < size; y++)
        {
            inputarray[inpindex] = faceImage.GetPixel(x, y).GetBrightness() > 0.5 ? 0 : 1;
            inpindex++;
        }
    }

} // процедура получения входного вектора из картинки
```

```
public void Recognizing()
{
    var output = mynetwork.Compute(inputarray);
    var max = output.Max();
    var maxIndex = Enumerable.Range(0, output.Length).Where(z => output[z] == max).First();

    maxindex = maxIndex;

} // процедура распознавания
```

```
private Bitmap TakeFace(Image<Gray, Byte> grayimg, Image<Bgr, Byte> img, Bitmap faceimg, ImageBox imgbox)
{
    Rectangle face1 = new Rectangle(0, 0, 100, 100);
    var faces = faceCascade.DetectMultiScale(grayimg);

    foreach (var face in faces)
    {
        imgbox.Refresh();
        face1 = face;
    }

    face1.Y += 40; face1.X += 30; face1.Width -= 60; face1.Height -= 50;
    faceimg = faceimg.Clone(face1, faceimg.PixelFormat);

    return faceimg;

} // функция локализации лица
```

```
public void TeacherVectors(double[] outvector)
{
    progressBar2.Value = 0;

    GetInputVector();
    Array.Resize(ref inputvectors, inputvectors.Length + 1);
    inputvectors[inputvectors.Length - 1] = new double[inputarray.Length];
    inputvectors[inputvectors.Length - 1] = inputarray;

    Array.Resize(ref outputvectors, outputvectors.Length + 1);
    outputvectors[outputvectors.Length - 1] = new double[5];
    outputvectors[outputvectors.Length - 1] = outvector;

    progressBar2.Maximum = 5;
    for (int j = 0; j < 5; j++)
    {
        progressBar2.Value += 1;
    }
} // процедура добавления обучающего образца в выборку
```

Обучение:

```
var teacher = new BackPropagationLearning(mynetwork); // экземпляр класса обучения нейросети
teacher.LearningRate = 0.05;
teacher.Momentum = 0.1;

progressBar1.Maximum = Convert.ToInt32(numericUpDown1.Value);
progressBar1.Value = 0;

double prErr = 10000;
double error = 10000;
double minerr = 10;
double neederror = 0.1;

Random rnd = new Random();

while (error > neederror)
{
    error = teacher.RunEpoch(inputvectors, outputvectors);

    if (Math.Abs(error - prErr) < 0.00000000001)
    {
        teacher.LearningRate /= 2;
        if (teacher.LearningRate < 0.01)
            teacher.LearningRate = 0.01;
    }
```