1. **АНАЛИЗ СУЩЕСТВУЮЩИХ НЕЙРОННЫХ СЕТЕЙ**
   1. **Искусственный нейро́н - нейрон Маккалока - Питтса**

Математически искусственный нейрон обычно представляют как некоторую нелинейную функцию от единственного аргумента – линейной комбинации всех входных сигналов. Эту функцию называют функцией активации, функцией срабатывания или передаточной функцией нейрона. Полученный результат отправляется на единственный выход нейрона. Математическая модель искусственного нейрона представлена на рис. 1.

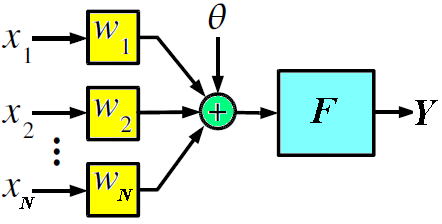


Рис. 1. Структура искусственного нейрона

Линейная комбинация входных сигналов с определенными весами подается на вход нейрона. Кроме того, иногда ко входу нейрона специально добавляют некоторую случайную величину, которая называется смещением. Смещение можно рассматривать как сигнал на дополнительном, всегда нагруженном синапсе.

Результирующую реакцию формального нейрона можно представить следующим образом. После сумматора получаем ответ 

,

где - скалярное произведение векторов ; N – число входов нейрона. После прохождения порогового устройства и преобразования пороговой функцией получаем выходной ответ, равный

.

Взвешенную сумму иногда представляют в виде  , где - порог нейронного элемента, характеризующий сдвиг функции активации по оси абсцисс.

Теперь формальный нейрон можно определить подобно конечному автомату. К основным компонентам искусственного нейрона относятся умножители, сумматор-аккумулятор и блок функции активации нейрона, что иллюстрируется рис.3

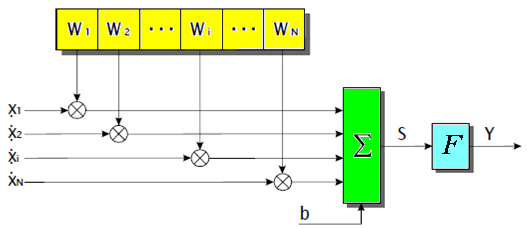


Рис. 2. Структура искусственного нейрона

Нейрон содержит следующие блоки:

Умножители. Количество умножителей должно быть равно количеству входов нейрона.

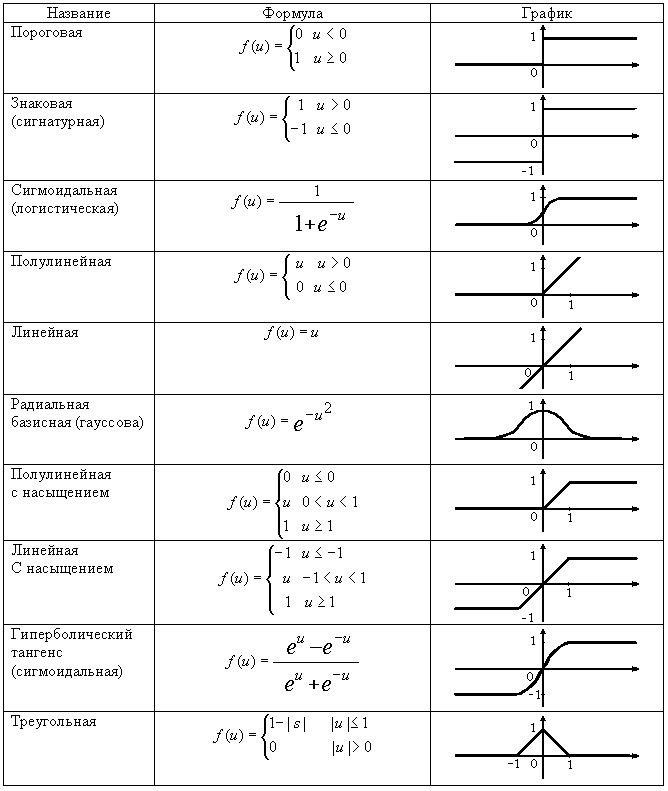
Сумматор - сглаживающий фильтр. Сумматор один, и с увеличением количества входов его сложность не меняется.

Функция активации. Функция активации определяет зависимость сигнала на выходе нейрона от взвешенной суммы сигналов на его входах.

**1.2.Различные виды функций активации**

Различные виды функций активации приведены в таблице 1

Таблица 1



Рассмотрим различные модификации искусственного нейрона.

**1.2. Звезды Гроссберга**

Идеи, отраженные в исследованиях Стефана Гроссберга на заре биологической кибернетики, положены в основу многих нейросетевых разработок.

Начинаем наше рассмотрение конфигураций входных и выходных звезд Гроссберга [Grossberg, 1988]. Входная и выходная звезда - фрагмент нейронных сетей, предложенный и использованный Гроссбергом во многих нейросетевых моделях. Состоит из нейрона, управляющего группой весов. Рассмотрим входную звезду Гроссберга. Выходные звезды в сочетании с входными могут соединяться в сети любой сложности. Входная звезда Гроссберга показана на рис. 3.

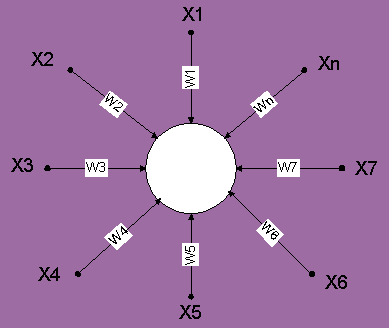


Рис. 3. Входная звезда Гроссберга

Входная звезда состоит из нейрона, на который подается группа входов через синаптические веса. Входные и выходные звезды могут быть взаимно соединены в сети любой сложности. Гроссберг рассматривал входные звезды как модели отдельных участков биологического мозга.

Выходная звезда Гроссберга показана на рис. 4.

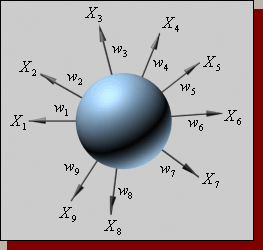


Рис. 4. Выходная звезда Гроссберга

Входная звезда обучается реагировать на определенный входной вектор X и ни на какой другой. Это обучение реализуется путем настройки весов таким образом, чтобы они соответствовали входному вектору. Выход звезды определяется как взвешенная сумма ее входов. С другой точки зрения, выход можно рассматривать как свертку входного вектора с весовым вектором. Следовательно, нейрон должен реагировать наиболее сильно на входной образ, которому был обучен.

Нейрон в форме входной звезды имеет  входов , которым соответствуют веса , и один выход , являющийся взвешенной суммой входов.

Структурная схема нейрона данного типа представлена на рис. 5.

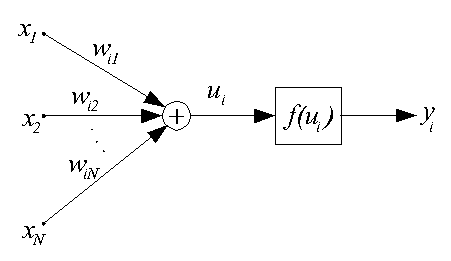


Рис. 5. Структурная схема нейрона - звезды Гроссберга

Особенностями звезды Гроссберга, отличающими его от нейронов ранее рассмотренных типов, являются следующие: функция активации  часто линейна, т.е. ; входной вектор  нормализован так, что его эвклидова норма равна 1; обучение такой звезды возможно как с учителем, так и без него.

Нормализация элементов вектора  производится по следующей формуле:

.



Входная звезда обучается выдавать сигнал на выходе всякий раз, когда на входы поступает определенный вектор. Таким образом, входная звезда является детектором совокупного состояния своих входов. Процесс обучения представляется в следующей итерационной форме:



Темп обучения  имеет начальное значение масштаба 0.1 и постепенно уменьшается в процессе обучения. В процессе настройки нейрон учится усредненным обучающим векторам.

Выходная звезда Гроссберга выполняет противоположную функцию - функцию командного нейрона, выдавая на выходах определенный вектор при поступлении сигнала на вход. Нейрон этого типа имеет один вход и  выходов с весами , которые обучаются по формуле:



Рекомендуется начать c  порядка единицы и постепенно уменьшать до нуля в процессе обучения. Итерационный процесс будет сходиться к собирательному образу, полученному из совокупности обучающих векторов.

Недостатки: Каждая звезда в отдельности реализует слишком простую функцию. Из таких звезд невозможно построить нейронную сеть, которая реализовала бы любое заданное отображение. Это ограничивает практическое применение входных звезд.

Преимущества: Входная звезда хорошо моделирует некоторые функции компонент биологических нейронных сетей. Сеть, включающая в себя входные звезды, может быть достаточно хорошей моделью отдельных участков мозга.

Особенностью нейронов в форме звезд Гроссберга является локальность памяти. Каждый нейрон в форме входной звезды помнит "свой" относящийся к нему образ и игнорирует остальные. Каждой выходной звезде присуща также конкретная командная функция. Образ памяти связывается с определенным нейроном, а не возникает вследствие взаимодействия множества нейронов в сети.

**3. Нейрон WTA** (Принцип Winner Take All - Победитель Забирает Все ).

Рассмотрим задачу о принадлежности образа  некоторому классу , определяемому заданными библиотечными образами . Каждый из заданных образов обучающей выборки непосредственно определяет свой собственный класс, и таким образом, задача сводится к поиску "ближайшего" образа. В случае двух двоичных (0-1) образов расстояние между ними может быть определено по Хеммингу, как число несовпадающих компонент. Теперь после вычисления всех попарных расстояний  искомый класс определяется по наименьшему из них.

Нейросетевое решение этой задачи может быть получено на основе архитектуры Липпмана-Хемминга (Lippman R., 1987).

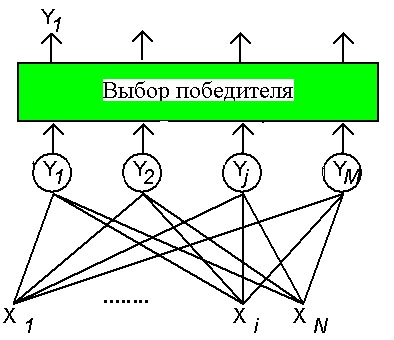


Рис. 6. Нейронная сеть Липпмана-Хемминга.

Сеть имеет один слой одинаковых нейронов, число которых равно количеству классов. Таким образом, каждый нейрон "отвечает" за свой класс. Каждый нейрон связан с каждым из входов, число которых равно размерности рассматриваемых библиотечных образов. Веса связей полагаются равными нормированным библиотечным образам:



Здесь  - значение веса связи от *n*-го входа к *m*-му нейрону (см. рис.7.1.). Процесс поступления информации о векторе  в нейронную сеть является безитерационным. При этом входной вектор сначала нормируется:



и нейроны принимают начальные уровни активности:

.

Здесь  - пороговая функция нейрона, которая выбирается равной нулю при *x*<0, и  при *x*>0. Пороги  полагаются обычно равными нулю.

При поступлении входного вектора начальное возбуждение получают все нейроны, скалярное произведение векторов памяти которых с входным вектором превышает порог. В дальнейшем среди них предстоит выбрать один, для которого оно максимально. Это достигается введением дополнительных обратных связей между нейронами, устроенных по принципу "латерального торможения". Каждый нейрон получает тормозящее (отрицательное) воздействие со стороны всех остальных нейронов, пропорционально степени их возбуждения, и испытывает возбуждающее (положительное) воздействие самого на себя. Веса латеральных связей в нейронном слое нормируются таким образом, что суммарный сигнал является возбуждающим только для нейрона с максимальной исходной активностью. Остальные нейроны испытывают торможение:



По выполнении некоторого числа итераций *t* для всех нейронов кроме одного значение аргумента функции *f(x)* становится отрицательным, что обращает их активность  в нуль. Единственный, оставшийся активным, нейрон является победителем. Он и указывает на тот класс, к которому принадлежит введенный образ. Такой механизм получил название "Победитель-Забирает-Все" ( Winner Take All - WTA ). Механизм WTA используется и в других нейросетевых архитектурах. Заложенный в его основе принцип латерального торможения имеет глубокие биологические основания и весьма широко распространен в нейронных сетях живых организмов.

Для обучения нейронов типа WTA не требуется учитель, оно практически полностью аналогично обучению входной звезды Гроссберга. Начальные значения весовых коэффициентов всех нейронов выбираются случайным образом с последующей нормализацией относительно 1. При предъявлении каждого обучающего вектора Xk определяется нейрон-победитель, что дает ему право уточнить свои весовые коэффициенты по упрощенному правилу Гроссберга. Все проигравшие нейроны оставляют свои весовые коэффициенты неизменными. Понятно, что в каждом цикле обучения побеждает тот нейрон, чей текущий вектор входных весов Wi наиболее близок входному вектору . При этом вектор Wi корректируется в сторону вектора Xk. Поэтому в ходе обучения каждая группа близких друг другу входных векторов (кластер) обслуживается отдельным нейроном.

Нейросетевая парадигма нейрона Хемминга является моделью с прямой структурой памяти. Информация, содержащаяся в библиотечных образах, никак не обобщается, а непосредственно запоминается в синаптических связях. Память здесь не является распределенной, так как при выходе из строя одного нейрона полностью теряется информация обо всем соответствующем ему образе памяти.

**4. Нейрон Хебба**

Д. Хебб, исследуя поведение природных нервных клеток, зафиксировал усиление связи двух взаимодействующих клеток при их одновременном возбуждении [Hebb, 1949]. Это позволило ему предложить правило уточнения входных весов нейрона в следующем виде:

,

где  - коэффициент обучения, значение которого выбирается из интервала (0, 1).

Правило Хебба применимо для нейронов с различными функциями активации. Обучение нейрона может производиться как с учителем, так и без него. В первом случае в правиле Хебба вместо фактического значения выходного сигнала используется ожидаемая реакция .

Особенностью правила Хебба является возможность достижения весом  произвольно большого значения за счет многократного суммирования приращения в циклах обучения. Одним из способов стабилизации процесса обучения по Хеббу служит уменьшение уточняемого веса  на величину, пропорциональную коэффициенту забывания gamma. При этом правило Хебба принимает вид

.

Значение коэффициента забывания gamma выбирается из интервала (0, 1), рекомендуется соблюдать условие gamma<0,1. К сожалению, при обучении по правилу Хебба нейрона с линейной функцией активации стабилизация не достигается даже при использовании забывания. В 1991г. Е. Ойя предложил модификацию правила Хебба, имеющую следующий вид:

.

**5. Радиальный нейрон**

Нейроны данного типа существенно отличаются от ранее рассмотренных. Они используются только группами, составляя первый слой в многослойных радиальных сетях. Структурная схема такого нейрона дана ниже.

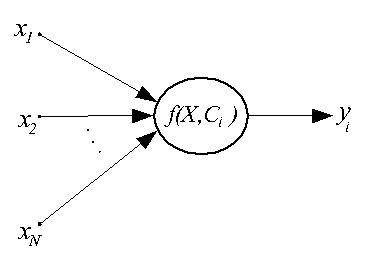


Рис. 7. Структурная схема радиального нейрона

Здесь  - радиальная функция с центром в точке с координатами . Такие функции разнообразны, но на практике чаще всего используется функция Гаусса, имеющая следующий вид:

,

где  - эвклидова норма расстояния между входным вектором  и центром  нейрона,  - параметр, определяющий "ширину" функции.

Принципиальное отличие радиального нейрона от сигмоидального (и персептрона) - в том, что сигмоидальный нейрон разбивает многомерное пространство входных сигналов гиперплоскостью, а радиальный - гиперсферой.

Обучение радиального нейрона заключается в подборе параметров радиальной функции  и . Подробно алгоритм обучения радиальных сетей приведен в соответствующем разделе позже. Здесь же в качестве примера приведено выражение, часто используемое для корректировки положения центра нейрона после предъявления *k*-ого обучающего вектора

,

где  - коэффициент обучения (<<1). Причем, такому уточнению подвергается только центр, ближайший к входному вектору  (подобный подход используется и при обучении нейронов типа WTA). Некоторые реальные нейроны можно рассматривать как радиальные.

**6. Стохастический нейрон**

Выше описана модель детерминистического искусственного нейрона, то есть состояние на выходе нейрона однозначно определено результатом работы сумматора входных сигналов. Рассматривают также стохастические нейроны, где переключение нейрона происходит с определенной вероятностью, то есть передаточная функция определена как

,

где распределение вероятности обычно имеет вид

,

нормировочная константа  вводится для условия нормализации распределения вероятности . Таким образом, нейрон активируется с вероятностью *P(u).* Параметр *T* — аналог температуры и определяет беспорядок в нейронной сети. Если *Т* устремить к 0, стохастический нейрон перейдет в обычный нейрон с передаточной функцией Хевисайда (пороговой функцией).

1. **Бионический нейрон**

В настоящее время популярна разработка моделей нейрона, учитывающих его свойства пластичности. Одна из таких моделей предложена В.Б.Вальцевым [Вальцев, Лавров, 2006]. Нейрон имеет несколько входов и выходов. Входы бывают нескольких типов: возбуждение, регуляция, память, торможение, запрет. Текущее состояние нейрона определяется текущим потенциалом и текущим порогом. Нейрон способен принимать и испускать импульсы.

Текущее состояние нейрона меняется со временем. Если подача на входы импульсов отсутствует, то значение текущего потенциала стремится к нулю по экспоненциальному закону. При этом значение текущего порога со временем стремится к некоторому постоянному значению, называемому порогом покоя. Ограничения потенциала и порога учитываются при расчёте эффекта от поступившей импульсации.

Принимать сигналы (импульсы) нейрон может с помощью входов. Каждый вход нейрона характеризуется весовым коэффициентом W (вес входа). Импульсы, поступая на вход нейрона, изменяют его текущее состояние. Эффект от импульса определяется типом входа, на который он поступил, весом этого входа а также текущим состоянием нейрона. На рисунке 5 представлена модель бионического нейрона.

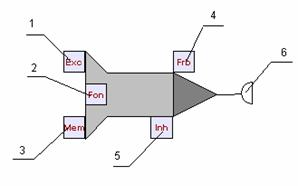


Рис. 8. Модель бионического нейрона 1 - вход возбуждения, 2 - вход регуляции, 3 - вход памяти, 4 - вход запрета, 5 - вход торможения, 6 - выход (синапс) нейрона

Одиночный импульс, пришедший на вход нейрона типа возбуждение, повышает величину потенциала нейрона на некоторое значение. Таким образом: если нейрон не возбуждён (потенциал равен нулю), то импульс увеличивает значение потенциала на величину, равную весу входа. Продолжительная периодическая импульсация поднимает потенциал «ступеньками», высота которых убывает с возрастанием самого потенциала. Высота ступенек становится равной нулю, когда потенциал достигает предельного максимального значения При данном задании функции, потенциал может превышать максимальное значение на величину, пренебрежимо малую. Сходным образом ведёт себя вход Торможения. Однако его роль заключается в уменьшении потенциала на величину H, которая рассчитывается аналогично:

Особенным образом работает вход памяти. Аналогично возбуждению он увеличивает потенциал, но приращение потенциала теперь зависит не только от веса входа, но также от текущего состояния коэффициента обученности. Коэффициент обученности в отличие от веса меняет своё значение динамически в процессе работы нейронной сети. Обучение, переобучение, разобучение – механизмы, регулирующие значение µ и, как следствие, работу входов памяти нейрона.

**1.3. Классификация искусственных нейронных сетей по структуре**

Рассмотрим наиболее известные типы нейронных сетей: Персептрон Розенблатта; Сеть Хопфилда; Сеть Хэмминга; Сеть Кохонена.

Вычислительные системы, основанные на искусственных нейронных сетях, обладают рядом качеств, которые отсутствуют в машинах с архитектурой фон Неймана но присущи мозгу человека:

1. Массовый параллелизм;

2. Распределённое представление информации и вычисления;

3. Способность к обучению и обобщению;

4. Адаптивность;

5. Свойство контекстуальной обработки информации;

6. Толерантность к ошибкам.

С точки зрения структуры или архитектурыискусственная нейронная сеть (ИНС) может рассматриваться как направленный граф с взвешенными связями, в котором искусственные нейроны являются узлами. Связи между нейронами характеризуются весами, отражающими степень влияния передающих нейронов на принимающие нейроны.

По архитектуре связей искусственные нейронные сети могут быть сгруппированы в два класса, как показано на рис.9.

Можно выделить сети прямого распространения, в которых графы не имеют петель, и рекуррентные сети, или сети с обратными связями. В наиболее распространенном семействе сетей первого класса, называемых многослойным перцептроном, нейроны расположены слоями и имеют однонаправленные связи между слоями. На рис. 9 представлены типовые сети каждого класса.

Сети прямого распространения являются статическими в том смысле, что на заданный вход они вырабатывают одну совокупность выходных значений, не зависящих от предыдущего состояния сети. Рекуррентные сети являются динамическими, так как в силу обратных связей в них модифицируются входы нейронов, что приводит к изменению состояния сети. Рассмотрим классификация нейронных сетей по архитектуре.

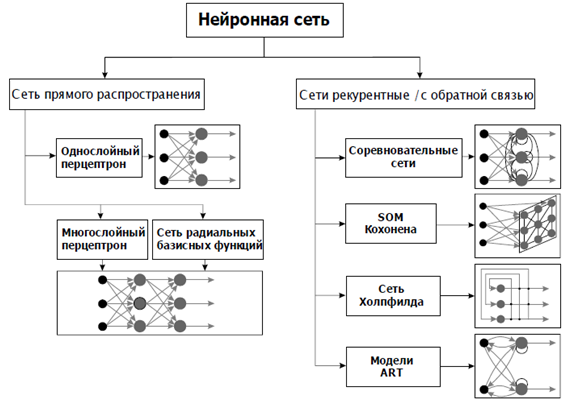


Рис. 9. Классификация нейронных сетей по архитектуре

**1.4 Обучение нейронных сетей**

В контексте искусственной нейронной сети процесс обучения имеет следующее определение. Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения - одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Обучение нейронной сети- настройка архитектуры сети и весов связей для эффективного выполнения специальной задачи. Обычно нейронная сеть должна настроить веса связей по имеющейся обучающей выборке. Функционирование сети улучшается по мере итеративной настройки весовых коэффициентов. Свойство сети обучаться на примерах делает их более привлекательными по сравнению с системами, которые следуют определенной системе правил функционирования, сформулированной экспертами. Для конструирования процесса обучения, прежде всего, необходимо иметь модель внешней среды, в которой функционирует нейронная сеть - знать доступную для сети информацию. Эта модель определяет парадигму обучения. Во-вторых, необходимо понять, как модифицировать весовые параметры сети, какие правила обучения управляют процессом настройки. Алгоритм обучения означает процедуру, в которой используются правила обучения для настройки весов. Процесс обучения нейронной сети с обратным распространением ошибки схематически иллюстрируется рис. 10.

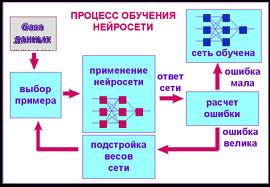


Рис. 10. Процесс обучения нейронной сети

С точки зрения машинного обучения, нейронная сеть представляет собой частный случай методов распознавания образов, дискриминантного анализа, методов кластеризации и т. п. С математической точки зрения, обучение нейронных сетей — это многопараметрическая задача нелинейной оптимизации. С точки зрения кибернетики, нейронная сеть используется в задачах адаптивного управления и как алгоритмы для робототехники. С точки зрения развития вычислительной техники и программирования, нейронная сеть — способ решения проблемы эффективного параллелизма [Александров, Анохин и др.,2008; Дунин-Барковский, Терехин, 1990; Уоссермен, 1992; Хайкин, 2006; Цетлин, 1969; Ясницкий, 2005]. А с точки зрения искусственного интеллекта, искусственная нейронная сеть является основой философского течения коннективизма и основным направлением в структурном подходе по изучению возможности построения (моделирования) естественного интеллекта с помощью компьютерных алгоритмов.

Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искаженных данных.

По характеру обучения нейронные сети подразделяются на группы. Обучение с учителем — выходное пространство решений нейронной сети известно.

Обучение без учителя — нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называют самоорганизующимися. Обучение с подкреплением — система назначения штрафов и поощрений от среды.

* 1. **Прикладные задачи, решаемые нейронными сетями**

Рассмотрим некоторые задачи, решаемые искусственными нейронными сетями, с дополнением кратких пояснений.

1. Распознавание образов;
2. Кластер-анализ и классификация;
3. Ассоциативная память (сети Хопфилда);
4. Сжатие данных;

5. Аппроксимация функций;

6. Задачи управления;

7. Управление движением транспортного средства и т. д.;

8. Прогнозирование;

9. Принятие решений;

10. Задачи оптимизации;

Приведем краткую классификацию искусственных нейронных сетей, не претендующую на полноту и основанную на учете приложений искусственных нейронных сетей, сведенную в таблицу 2.

Таблица 2.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Автор | Год создания | Применение |
| Однослойный перцептрон (Single layer perceptron) | R. Rosenblatt | 1959 | Распознавание образов/категоризация |
| Обратное распространение ошибки (Back Propagation) | R. Rosenblatt,  M.Minsky, S. Papert | 1960-е | Распознавание образов, классификация, прогнозирование |
| Встречное распространение (Counter Propagation) | R. Hecht-Neilsen | 1986 | Распознавание образов, восстановление образов (ассоциативная память), сжатие данных |
| Входная звезда (Instar Network) | S. Groosenberg | 1974 | Распознавание образов |
| Выходная звезда (Outstar Network) | S. Groosenberg | 1974 | Распознавание образов |
| Искусственный резонанс-1 (ART-1 Network) | S. Grossenberg, G. Carpenter | 1986 | Распознавание образов, кластерный анализ |
| Сеть Хопфилда (Hopfield Network) | J. J. Hopfield | 1982 | Поиск и восстановление данных по их фрагментам |
| Сеть Хэмминга (Hamming Network) | R. W. Hamming | 1987 | Распознавание образов, классификация, ассоциативная память, надежная передача символов в условиях помех |
| Сеть Кохонена (Kohonen Network) | T. Kohonen | 1984 | Кластерный анализ, распознавание, образов, классификация |
| Сеть поиска максимума (MAXNET) | R. P. Lippman | 1987 | Совместно с сетью Хэмминга, в составе НС систем распознавания образов |
| Сеть поиска максимума с прямыми связями (Feed-Forward MAXNET) | R. P. Lippman | 1987 | Совместно с сетью Хэмминга, в составе НС систем распознавания образов |
| Двунаправленна ассоциативная память (BAM Network) | B. Kosko | Вторая половина 80-х | Ассоциативная память, распознавание образов |
| Обучение Больцмана (Boltzman machine) | J. Hinton, T. Sejnovsky, H. Szu | 1985 | Распознавание изображений, сигналов радара, радара |
| Классификатор Гаусса (Neural Gaussian Classifier) | R. P. Lippman | 1987 | Распознавание, образов, классификация |
| Генетические алгоритмы (Genetic training algorithm) | J. Holland, D. Goldberg | 1975 1988 | Обучение НС, распознавание сигналов сонаров |

Искусственная нейронная сеть успешно используется и в других областях и число таких приложений непрерывно возрастает. Дадим краткие пояснения прикладным задачам, решаемым с помощью искусственной нейронной сети.

1. Классификация образов.Задача состоит в указании принадлежности входного образа (например, речевого сигнала или рукописного символа), представленного вектором признаков, одному или нескольким предварительно определенным классам. К известным приложениям относятся распознавание букв, распознавание речи, классификация сигнала электрокардиограммы, классификация клеток крови. В робототехнике одним из основных приложений является распознавание объектов из видеоинформации, полученной от системы технического зрения.
2. Кластер-анализ и классификация (без учителя) (сети Кохонена).

К задачам распознавания относятся также кластеризация*.* При решении задачи кластеризации, которая известна также как классификация образов "без учителя", отсутствует обучающая выборка с метками классов. Кластеризация- метод, применяющийся для анализа больших наборов данных заключающийся в разбиении всего множества на группы близкородственных элементов (кластеры). Кластеризация может быть использована для решения таких задач как обработка изображений, классификация, тематический анализ коллекций документов, построение репрезентативной выборки. Известны случаи применения кластеризации для извлечения знаний, сжатия данных исследования свойств данных.

1. Ассоциативная память*.* В модели вычислений фон Неймана, которая используется в современных вычислительных машинах, обращение к памяти доступно только посредством адреса, который не зависит от содержания памяти. Более того, если допущена ошибка в вычислении адреса, то может быть найдена совершенно иная информация. Ассоциативная память, или память, адресуемая по содержанию, доступна по указанию заданного содержания. Содержимое памяти может быть вызвано даже по частичному входу или искаженному содержанию. Ассоциативная память чрезвычайно желательна при создании мультимедийных информационных баз данных.
2. Сжатие данных. Способность нейросетей к выявлению взаимосвязей между различными параметрами дает возможность выразить данные большой размерности более компактно, если данные тесно взаимосвязаны друг с другом. Обратный процесс — восстановление исходного набора данных из части информации — называется (авто)ассоциативной памятью. Ассоциативная память позволяет также восстанавливать исходный сигнал/образ из зашумленных/поврежденных входных данных. Решение задачи гетероассоциативной памяти позволяет реализовать память, адресуемую по содержимому.
3. Аппроксимация функций*.* Предположим, что имеется обучающая выборка  (пары данных вход-выход), которая генерируется неизвестной функцией, искаженной шумом. Задача аппроксимации состоит в нахождении оценки неизвестной функции . Аппроксимация функций необходима при решении многочисленных инженерных и научных задач моделирования.

6. Задачи управления. Рассмотрим динамическую систему, заданную совокупностью , где  является входным управляющим воздействием, а  - выходом системы в момент времени . В системах управления с эталонной моделью целью управления является расчет такого входного воздействия , при котором система следует по желаемой траектории, диктуемой эталонной моделью. Примером является оптимальное управление приводами роботов.

7. Управление движением транспортного средства и т. д.. Задачи управления движением в настоящее время являются актуальными и активно развиваются.

8. Прогнозирование**.**Пусть заданы n дискретных отсчетов в последовательные моменты времени  Задача состоит в предсказании значения в некоторый будущий момент времени, что можно выразить в виде формулы . Предсказание/прогноз имеют значительное влияние на принятие решений в бизнесе, науке и технике. Предсказание цен на фондовой бирже и прогноз погоды являются типичными приложениями техники предсказания/прогноза.

9. Принятие решений. Эта задача близка к задаче классификации. Классификации подлежат ситуации, характеристики которых поступают на вход нейронной сети. На выходе сети при этом должен появиться признак решения, которое она приняла. При этом в качестве входных сигналов используются различные критерии описания состояния управляемой системы.

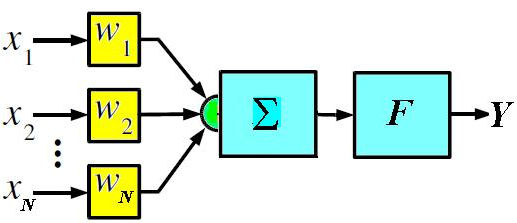
10**.** Оптимизация.Многочисленные проблемы в математике, статистике, технике, науке, медицине и экономике могут рассматриваться как проблемы оптимизации. Задачей алгоритма оптимизации является нахождение такого решения, которое удовлетворяет системе ограничений и максимизирует или минимизирует целевую функцию. Задача коммивояжера, является классическим примером задачи оптимизации.

**Глава 2.**

**Математическая модель нейрона**

Первой формальной моделью нейронных сетей (НС) была модель МакКаллока-Питтса. Суть её заключается в следующем.

Имеется n входных величин импульсов x1,…,xn в бинарном формате, описывающих объект x. Попадая в нейрон, импульсы сталкиваются с весами ω1,…,ωn. Если вес положительный, то соответствующий синапс возбуждающий, если отрицательный, то тормозящий. После импульсы суммируются и преобразуются и на выходе мы получаем 0 либо 1.



Ри. 11. Схема нейрона МакКаллока – Питтса

**Избирательный нейрон**

Избирательный нейрон относится ко второй волне построения нейронных сетей. Именно он используется в данной работе в качестве основы нейронной сети. Эта модель нейрона более приближена к модели биологического нейрона, чем нейрон МакКаллока-Питтса, за счёт того, что отражение входной информации в ней отражается не в весовых коэффициентах, а в изменении количества и качества дендритов.

Может ли система дендритов на входе нейрона быть столь пластичной и лабильной, чтобы адекватным образом отражать входную информацию за счет изменения числа возбуждающих и тормозных дендритов? Ранее этот вопрос был изучен недостаточно. Однако в последнее время были сделаны ряд удивительных открытий, показавших наличие у дендритов совершенно необычных возможностей их роста, изменения количества и качества с помощью специальной системы шипиков. Установлено также наличие у дендритов активных свойств, похожих на свойства аксонов. Все это вместе взятое доказывает возможность адаптации к отражению любой входной информации не за счет весовых функций дендритов, а за счет изменения их количества и качества дендритов. Под различным качеством дендритов понимаются их свойства: дендриты возбуждающие и тормозные, различные по диаметру и электрической мощности и т.д.; возможность роста и гибели отдельных дендритов. В связи с этим было предложено устройство — избирательный нейрон, выполняющее функции нейрона, структурная схема которого показана на рисунке.

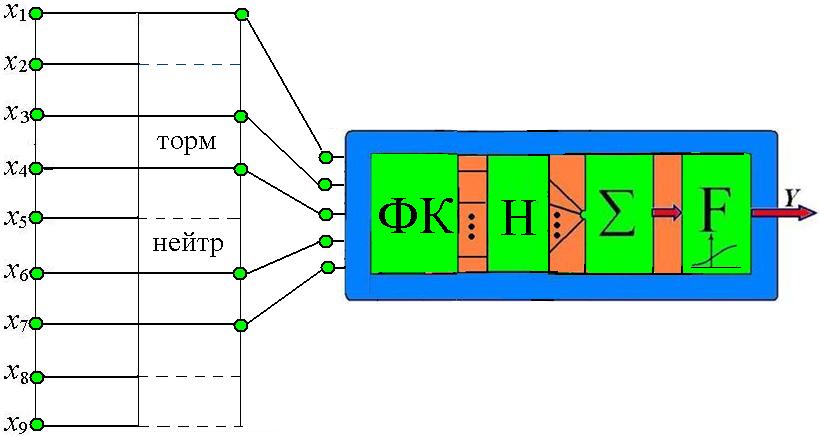


Рис. 12. Схема избирательного нейрона. На рисунке приняты следующие обозначения: x1….xn – входные сигналы; ФК – формируемые кластеры дендритов; Н – нормирующее устройство; ∑ - сумматор; F - пороговый преобразователь импульсных последовательностей.

Рассмотрим кратко его устройство и отличие от обычного искусственного нейрона МакКаллока – Питтса. Схематически устройство, формирующее кластеры дендритов ФК, вынесено за блок схему нейрона и показано на рисунке с левой стороны от блок-схемы.

Нормирующее устройство производит выравнивание максимального числа активных входов при использовании нейрона в качестве элемента нейронной сети. Нейронной сетью является, например, однослойный перцептрон с несколькими нейронами в слое.

Модель избирательного нейрона в большей степени учитывает особенности реального биологического нейрона. Отражение входной информации в избирательного нейроне достигается не за счет весовых коэффициентов, а за счет изменения количества и качества (например, возбуждающие или тормозные дендриты) дендритов.

Математическая модель избирательного нейрона в большей степени учитывает реальные свойства биологического нейрона. Решение основной задачи нейрона — сжатие сенсорной или промежуточной информации, распознавание образов — достигается за счет изменения числа дендритов на входе нейрона, возбуждающих и тормозных. Надобность в регулировании весовых функций дендритов отпадает. Таким образом, в соме (теле) i-го нейрона формируется результирующий сигнал, поступающий в i-ый канал, равный

,



где ,



- число дендритов отдельного кластера, характеризующего внешний паттерн.



Принципиальным отличием предлагаемого избирательного нейрона от искусственного нейрона МакКаллока—Питтса и от динамического нейрона на основе математической модели Ходжкина—Хаксли и других моделей подобного рода является: полное отсутствие весовых коэффициентов для входных воздействий и всех недостатков нейронных сетей, связанных с наличием весового суммирования.

Для обработки информации используются нейронные сети. Основой этих сетей – элементарным элементом – является однослойный перцептрон. Более подробно нейронные сети перцептронного типа описаны ниже.

**Структурные схемы нейронных сетей типа перцептронов. Нейронные сети типа однослойного и многослойного перцептронов**

Рассмотрим основные структурные схемы нейронных сетей. Однослойные нейронные сети для обработки импульсных потоков от рецепторов и промежуточных нейронов показаны на рис. 12.

Для распознавания возможно использование многослойных перцептронов с различным количеством скрытых слоев. Перцептрон, включающий 2 слоя нейронов, показан на рис. 13.

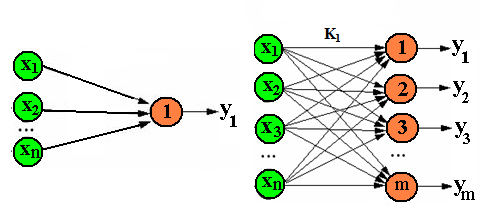


Рис. 12. Перцептрон, включающий 1 слой нейронов. K1 – системы каналов

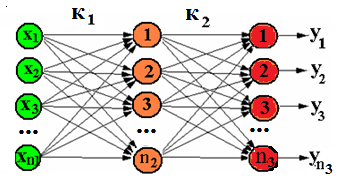


Рис. 13. Перцептрон, включающий 2 слоя нейронов. K1, K2 – системы каналов.

Перцептрон, включающий 3 слоя нейронов, показан на рис. 14.

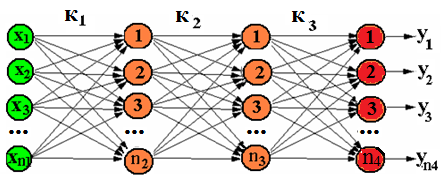


Рис. 14. Перцептрон, включающий 3 слоя нейронов. K1, K2, K3 – системы каналов.

**Математическая теория избирательного перцептрона**

Рассмотрим математическую теорию рассматриваемого избирательного нейрона. Обозначим возможные характеристические кодовые комбинации объектов на входе нейрона в виде векторов

; … ; ,

где n - число элементов кодовой комбинации; m = - число объектов. Все возможные кодовые комбинации входных объектов образуют матрицу A, которую можно представить в виде



Пусть конкретный нейрон содержит кластер связей, характеризуемый кодовой комбинацией . При подаче на вход нейрона с номером  кодовой комбинации входного объекта получим



Значения сумм  равны элементам матрицы *B*, равной

,

где - транспонированная матрица к *А*. Всего мы получим х сумм . Наибольшей будет сумма

,

где - число единиц в кодовой комбинации . Свойство сумм в том, что , используется для распознавания входных объектов.

Можно дать красивую интерпретацию избирательных свойств однослойного перцептрона, представив значения матрицы  в виде графика в трехмерном пространстве. С физической точки зрения значения  это значения матрицы , численно равные сумме после прихода входного сигнала  через кластер каналов связи, соединяющий входы нейрона с номером i на выходе перцептрона с его пороговой нелинейной системой. Для построения графика в трехмерном пространстве воспользуемся программой в Matlab-7. По осям Ох и Оу отложены значения i и j .

В качестве примера было реализовано избирательное распознавание для 33 букв русского алфавита в поле 150х100. Графическая интерпретация избирательных свойств системы иллюстрируется рисунком, приведенным ниже.

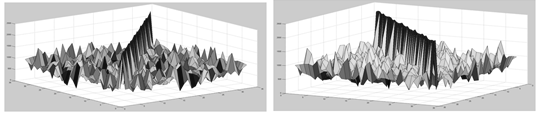


Рис. 14.

Характеристики информационных фильтров входных объектов. На приведенных рисунках показаны графики, характеризующие избирательные свойства системы при двух разных углах наблюдения. Избирательность характеризуется диагональной частью графика и выражена достаточно эффективно.

**Ввод данных на вход нейронной сети**

Ввод данных на вход нейронной сети, в нашем случае многослойного перцептрона, может быть осуществлен следующими двумя способами. Первый способ, как у создателя перцептрона А. Розенблата, – с экрана передающей электронно-лучевой телевизионной трубки (ЭЛТ). Второй – непосредственно с клавиатуры компьютера с предварительным преобразованием букв в бинарную кодовую строку или последовательность.

Рассмотрим эти способы подробнее. При вводе с экрана ЭЛТ производится сканирование экрана электронным лучом, в результате чего образуется токовая последовательность из импульсов с амплитудами 0 и 1. Эта последовательность содержит nxm число пикселей, где m – число движений электронного луча по вертикали, n - число градаций по горизонтали. Схема обработки информации в данной системе иллюстрируется рис.

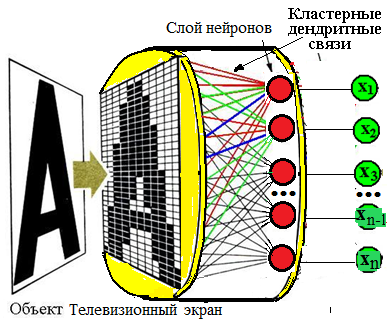


Рис. 14. Схема считывания входной информации в перцептроне А. Розенблата при использовании передающей электронно-лучевой трубки

При использовании первого способа на выходе рассмотренной системы получают бинарный код любой буквы русского алфавита, одной из 33. Эти бинарные коды подаются на входы последующих многослойных перцептронов с 2-мя, 3-мя и 4-мя слоями, показанных выше на рис.

Второй способ ввода букв художественного произведения является упрощенным вариантом, когда кодовые комбинации образуются в компьютере, а сами буквы вводятся с клавиатуры. Этот способ ввода букв иллюстрируется рис. 15

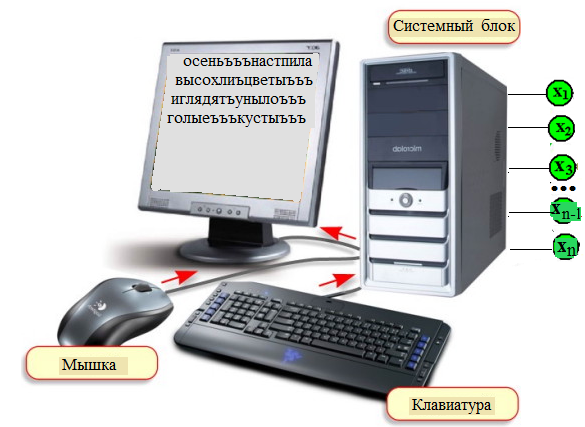


Рис. 15. Ввод букв с клавиатуры с последующим получением бинарных кодовых комбинаций с помощью компьютера.

**Алгоритмы, используемые в программе**

1. **Работа с эталонным изображением.**

Производится необходимая работа с каждым эталонным изображением для его приведения к бинарной комбинации.

* 1. Читаем из jpg-файла изображение и помещаем в три различные матрицы, где каждая матрица хранит значения интенсивностей пикселей определенного цвета.

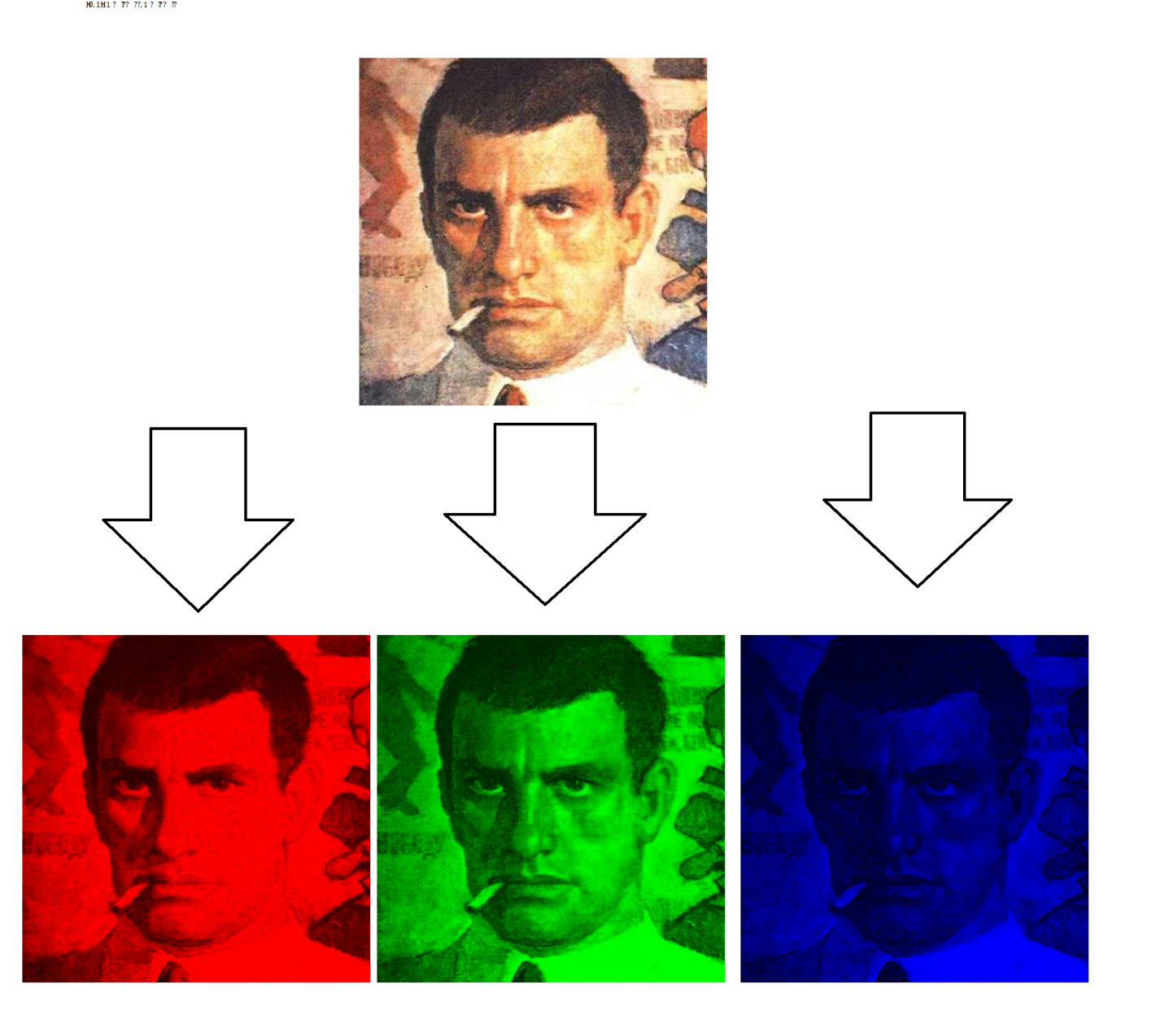


Рис.16. Деление цветного изображение на «красную», «зеленую» и «синюю» матрицы.

* 1. Приведение эталонных изображений к стандартной форме размеру 100х100.

Приводим матрицы к размеру 100х100 с помощью метода ближайшего соседа (см. Приложение 1).

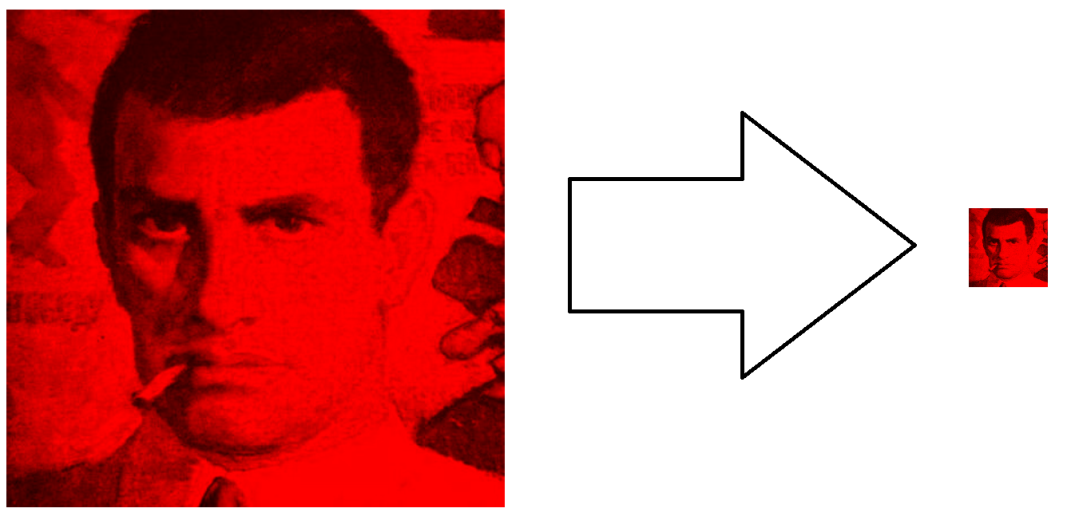


Рис.17. Приведение матрицы к размеру 100х100.

* 1. Приводим матрицы к вектор-строкам длиной 10000.
  2. Приводим каждую вектор-строку к бинарной вектор-строке, где каждому элементу массива (пикселю) ставится в соответствие своя бинарная комбинация.

Рассмотрим метод создания бинарной последовательности на примере одного пикселя:

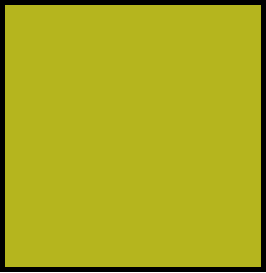
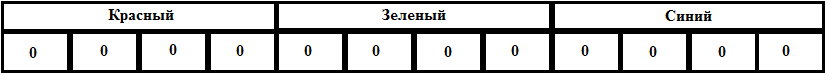


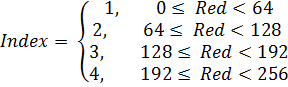
Рис.18. Пример пикселя.

Интенсивность красного цвета – 181, зеленого -181, синего – 30. Пусть предлагается разбить каждый из цветов на 4 градации. Для данного пикселя создается бинарный вектор из 12 элементов.



Вначале все значения массива являются нулевыми.

Затем изменяем значение на одном из «красных» индексов (от 1 до 4) на единицу по следующей формуле.

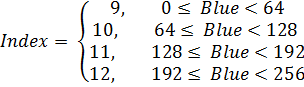
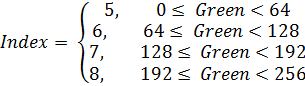


где Index – индекс бинарного вектора, где значение меняется на единицу; Red – интенсивность красного цвета у пикселя.

Промежуточный итог виден на следующей таблице:



Аналогичным образом изменяем значения на одном из «зеленых» (от 5 до 8) и на одном из «синих» (от 9 до 12) индексов на единицу по следующим формулам:



Green - интенсивность зеленого цвета у пикселя;

Blue - интенсивность синего цвета у пикселя;

* 1. Окончательный результат.

Итого получается следующий вектор-бинарная строка или код.



На этом обработка эталонных изображений заканчивается.

Преимуществом данного способа получения бинарной строки является возможность получения нормированной диагонали матрицы A \* AT. На диагонали должно получаться 30 000.

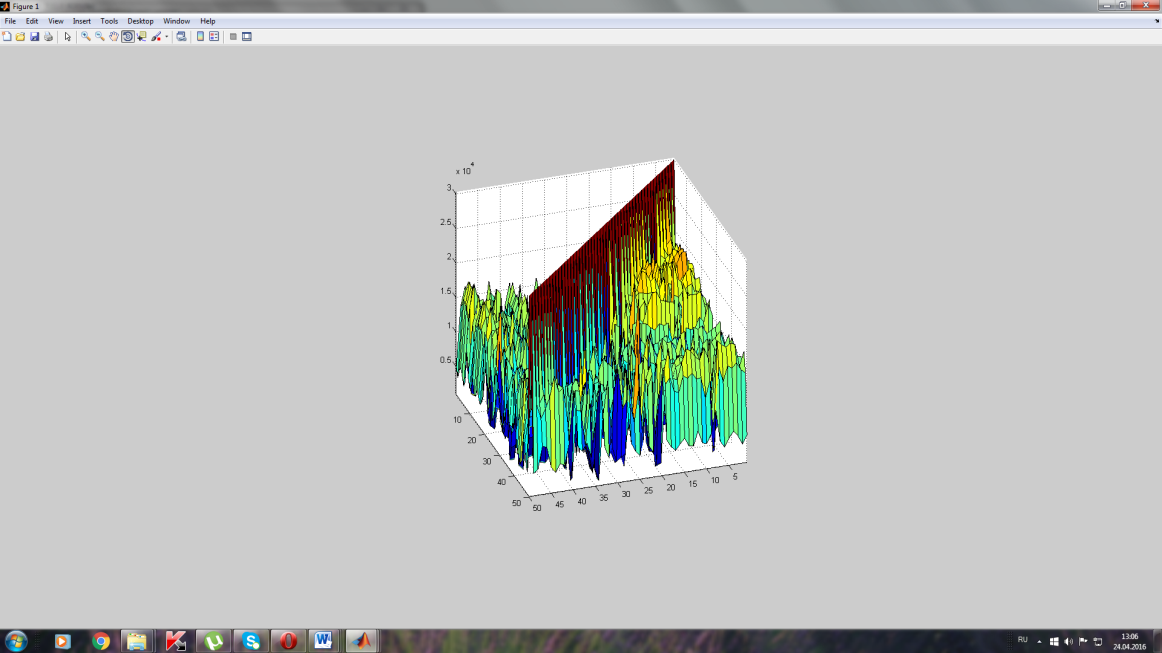


Рис.19. График матрицы A \* AT.

1. **Работа с тестовым изображением.**
   1. Запрашиваем у пользователя посредством диалогового окна тестовое изображение формата jpg для его распознавания.
   2. Производится необходимая работа с тестовым изображением для его приведения к бинарной комбинации (аналогичное пункту 1.4).
   3. На основе избирательного нейрона производится сравнение тестового изображения со всеми эталонными. На основе данных сравнений находится самое похожее эталонное изображение, и происходит его вывод на экран с кратким описанием, либо выводится в консоли сообщение о неудачном распознании (если порог распознания не превышен).

Пример сравнения тестового изображения с наиболее похожим на него эталонным:

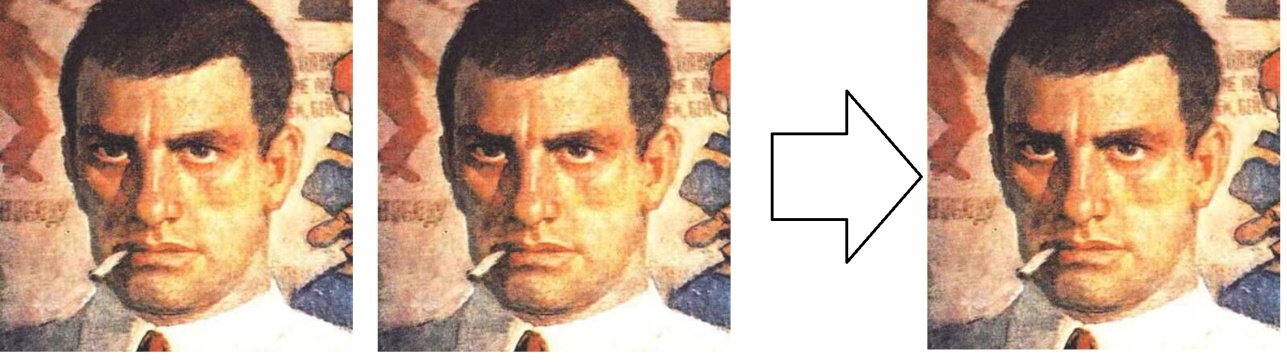


Рис.20. Сравнение тестового изображения с наиболее похожим на него эталонным.

Пример сравнения тестового изображения с непохожим на него эталонным:

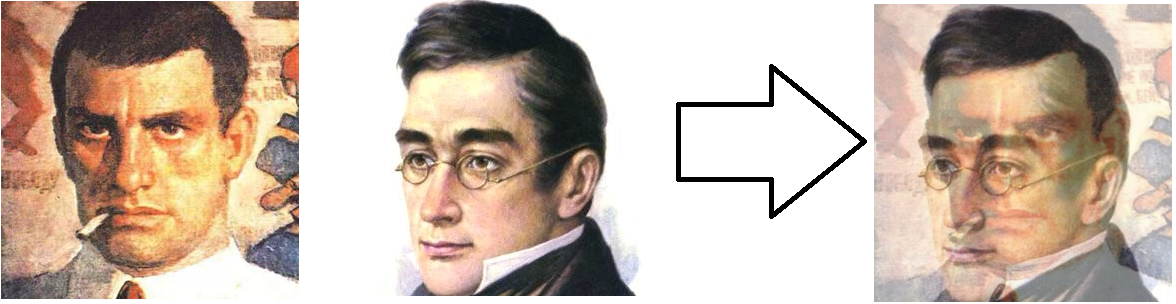


Рис.21. Сравнение тестового изображения с непохожим на него эталонным.

**Методы получения бинарных строк**

Метод 1. Кодирование интенсивности за счет движения единицы по длине элемента.

* 1. Представление пикселя в виде 0 или 1.

Дано бинарное изображение в виде матрицы, где цвет каждого пикселя может быть либо черным, либо белым. Для вычислений преобразуем матрицу в вектор-строку и рассмотрим каждый пиксель. Преобразуем каждое значение вектора в число по формуле:



Где Code – число, характеризующее пиксель, а White – признак, является ли пиксель белым.

После данных преобразований получается бинарная комбинация из нулей и единиц, пригодная для использования в процессе распознавания изображения.

Достоинства:

- Самый маленький размер бинарной комбинации, получаемой на выходе, при использовании данного метода по сравнению с остальными.

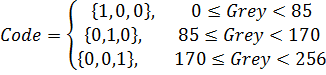
Недостатки:

- Данный метод работает только с бинарными изображениями.

- Входящие изображения должны быть нормализованы. Т. е. количество черных пикселей должно быть примерно равно для всех изображений.

* 1. Представление пикселя полутонового изображения в виде бинарной комбинации.

Дано полутоновое изображение в виде матрицы, где интенсивность серого цвета каждого пикселя описано числом от 0 до 255 (0 – черный цвет, 255 - белый). Для вычислений преобразуем матрицу в вектор-строку и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 3 оттенков серого. Преобразуем каждый значение вектора в двоичный код по формуле:



Где Code – двоичный код, характеризующий пиксель, а Grey – интенсивность серого цвета.

Достоинства:

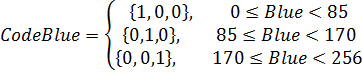
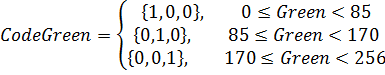
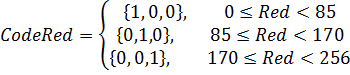
* Лучше всего подходит при работе с полутоновыми изображениями;
* Входящие изображения не обязаны проходит процесс нормализации, так как количество единиц в итоговых бинарных комбинациях для полутоновых изображений с одинаковым разрешением будет одинаковым вне зависимости от самих пикселей.

Недостатки:

* Данный метод не работает с полноцветными изображениями.
  1. Представление пикселя полноцветного изображения в виде бинарной комбинации.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 3 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в двоичный код по формуле:



(“+” – операция конкатенации)

CodeRed (CodeGreen, CodeBlue) – двоичный код, характеризующий красный (зеленый, синий) цвет пикселя;

Red (Green, Blue) – интенсивность красного (зеленого, синего) цвета;  
TotalCode – двоичный код, характеризующий пиксель.

Достоинства:

* Лучше всего подходит при работе с полноцветными изображениями;
* Входящие изображения не обязаны проходит процесс нормализации, так как количество единиц в итоговых бинарных комбинациях для полноцветных изображений с одинаковым разрешением будет одинаковым вне зависимости от самих пикселей.

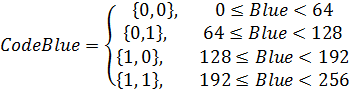
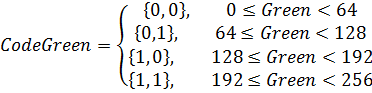
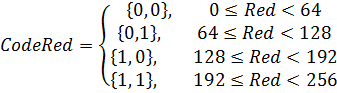
Недостатки:

* Неоптимален с точки зрения памяти, если на вход подаются полутоновые или бинарные изображения.

Метод 2. Двоичное кодирование интенсивности.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 4 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в двоичный код по формуле:



(“+” – операция конкатенации)



CodeRed (CodeGreen, CodeBlue) – двоичный код, характеризующий красный (зеленый, синий) цвет пикселя;

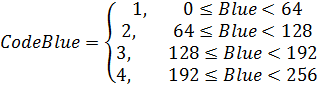
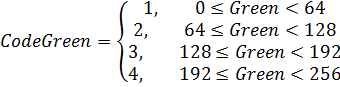
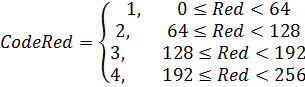
Red (Green, Blue) – интенсивность красного (зеленого, синего) цвета;  
TotalCode – двоичный код, характеризующий пиксель.

Данный метод не подходит для решения задачи представления пикселя в виде бинарной комбинации. С одной стороны, количество единиц в бинарной комбинации становится нефиксированным, поэтому входные изображения нуждаются в нормализации. С другой, представляется слишком сложным, а также нецелесообразным при наличии доступных альтернатив, найти способ провести эту нормализации. Реализация возможна только в компьютере фон Неймана.

Метод 3. Цифровое кодирование.

Дано полноцветное изображение в виде трех матриц, каждая из которых отвечает за интенсивность красного, зеленого и синего цвета пикселя соответственно. Интенсивность любого из трех цветов описано числом от 0 до 255 включительно.

Для вычислений преобразуем матрицы в вектор-строки и рассмотрим каждый пиксель. Пусть пиксели изображения могут иметь всего один из 4 оттенков красного, зеленого и синего цвета. Преобразуем каждый значение вектора в число по формуле:



(“+” – операция конкатенации)



Данный метод не подходит для решения задачи представления пикселя в виде бинарной комбинации, так как пикслель по данному методу не представляется в итоге как бинарная комбинация. Реализация возможна только в компьютере фон Неймана.

**Руководство пользователя программного обеспечения по распознаванию изображения.**

1. Запуск программы.
   1. Запустить MATLAB 7.0.1;
   2. Настроить директорию, в которой находится данное ПО;



Рис. 22. Директория, в которой расположено ПО.

* 1. Ввести в консоли BeginProgram и нажать Enter;

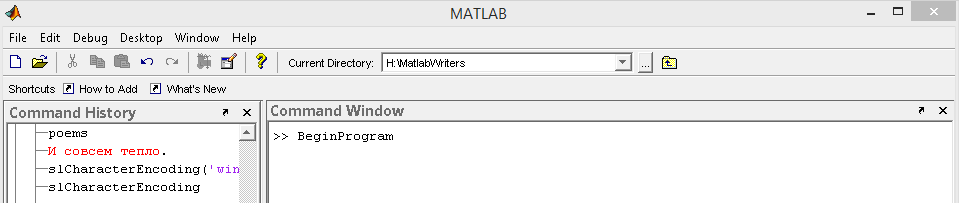


Рис. 23. Ввод в консоли команды запуска.

* 1. Программа запущена;

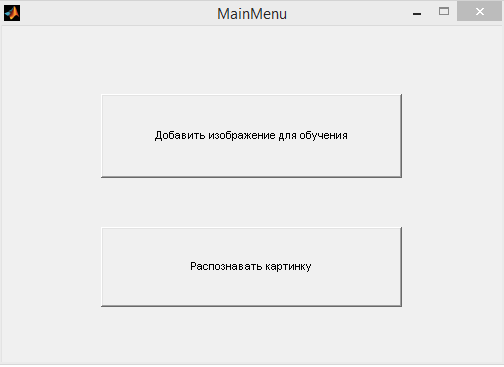


Рис. 24. Окно главного меню.

1. Добавление изображения для обучения.
   1. Нажать кнопку «Добавить изображение для обучения»;

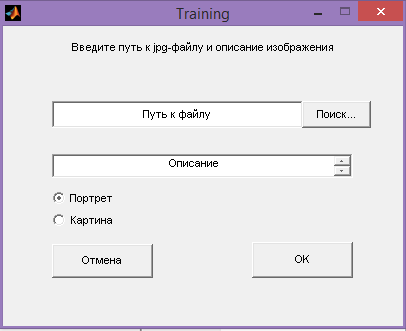


Рис. 25. Окно ввода изображения для обучения.

* 1. C помощью кнопки «Поиск…» найти нужное изображение для обучения программы. После успешного поиска путь к файлу отобразится в поле «Путь к файлу»;

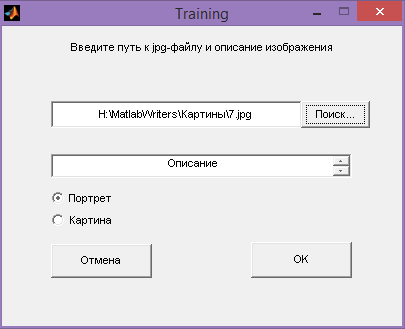


Рис. 26. Окно ввода изображения для обучения с введенным путем.

* 1. В поле «Описание» ввести описание изображения для запоминания. Выбрать, в какую базу изображений сохранить данное изображения посредством выбора одного из вариантов «Портрет» и «Картина» и нажать «ОК»;
  2. Выведется окно, подтверждающее, что данное изображение сохранено.

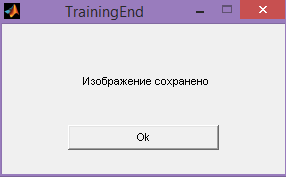


Рис. 27. Окно для подтверждения сохранения изображения в определенной базе.

1. Распознавание тестового изображения.
   1. Нажать кнопку «Распознавать картинку»;
   2. C помощью кнопки «Поиск…» найти нужное изображение для распознавания программой. После успешного поиска путь к файлу отобразится в поле «Путь к файлу»;

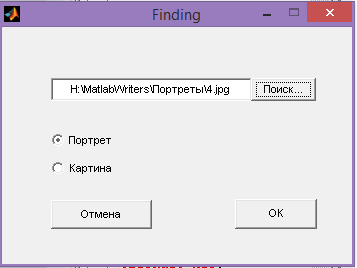


Рис. 28. Окно ввода изображения для его распознания.

* 1. Выбрать, среди эталонных изображений какой базы программа будет искать тестовое изображение посредством выбора одного из вариантов «Портрет» и «Картина» и нажать «ОК»;
  2. Выведется окно с изображением, которое требуется распознать. Нажать «ОК».

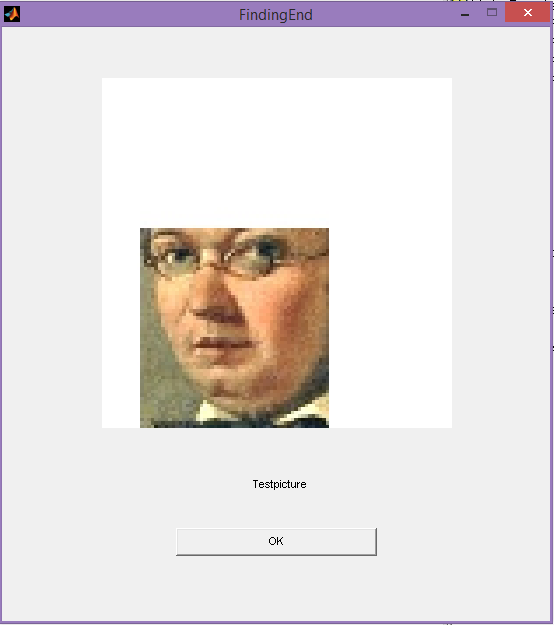


Рис. 29. Окно с изображением, которое нужно распознать.

* 1. Выведется окно, подтверждающее, что данное изображение распознано программой в виде изображения из базы, на которое тестовое изображение больше всего похоже, с соответствующим описанием.

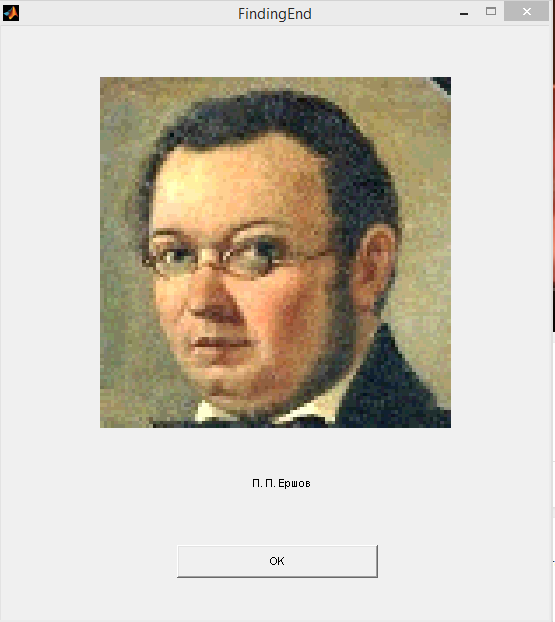


Рис. 30. Окно с изображением, наиболее похожим на тестовое, с соответствующим описанием.

**Распознавание цветных, черно-белых и контурных изображений и распознавание буквенных и цифровых текстов.**

Ранее была разработана программа для распознавания стихотворных текстов. Основные трудности в этой программе были связаны с получением бинарных кодов букв русского алфавита. В данной разработанной программе был исследован более сложный случай распознания полутоновых черно-белых изображений и полноцветных изображений. Также было разработано распознавание контурных изображений, которые по своей сути близко к распознаванию буквенных и цифровых изображений. Между этими вариантами распознавания существуют и определенные различия:

1. На каждую букву при распознавании стихотворений выделяется 33 ячейки.
2. При распознавании полутоновых изображений на каждый пиксель при наличии «к» градации используется «к» ячеек. В частном случае, при наличии черно-белой градации используется 2 ячейки.
3. При распознавании контурных изображений используется модификация в виде 1 ячейки, поскольку этого достаточно. При этом используется проверка интенсивности серого цвета пикселя. Если интенсивность больше 127, то в ячейку заносится «1», иначе заносится «0». Указанные процедуры приводят к уменьшению времени выполнения программы и экономии места в памяти.
4. Различия также присутствуют во время ввода данных для распознавания. Различия заключаются в следующем:
   1. При распознавании стихотворений необходимо вводить построчно части стихотворения.
   2. При распознавании изображения необходимо вводить путь к файлу тестового изображения и указывать, в базе каких изображений необходимо искать тестовое.

**Состав и структура программного обеспечения.**

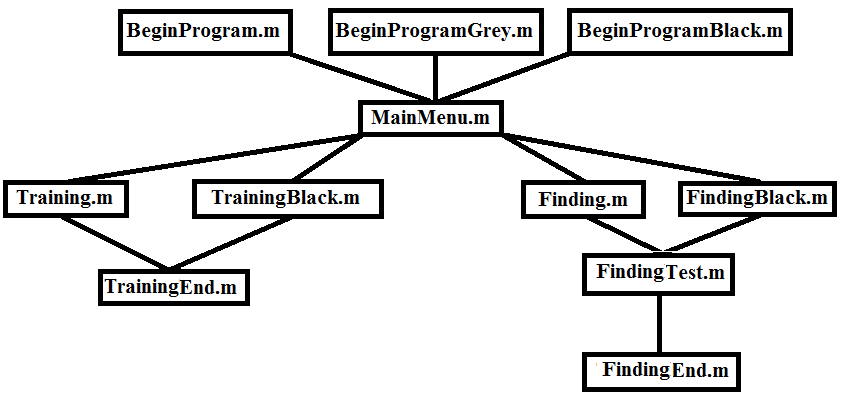


Рис. 31. Структура программного обеспечения.

1. Основное приложение распознавания полноцветных изображений с помощью избирательного нейрона запускается из модуля BeginProgram. Для вывода главного меню на экран из модуля BeginProgram вызывается модуль MainMenu.
   1. Для вывода на экран меню ввода эталонного полноцветного изображения для обучения программы из модуля MainMenu вызывается модуль Training. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.
   2. Для вывода на экран меню ввода тестового полноцветного изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.
2. Основное приложение распознавания полутоновых изображений с помощью избирательного нейрона запускается из модуля BeginProgramGrey. Для вывода главного меню на экран из модуля BeginProgramGrey вызывается модуль MainMenu.
   1. Для вывода на экран меню ввода эталонного полутонового изображения для обучения программы из модуля MainMenu вызывается модуль Training. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.
   2. Для вывода на экран меню ввода тестового полутонового изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.
3. Основное приложение распознавания контурных изображений с помощью избирательного нейрона запускается из модуля BeginProgramBlack. Для вывода главного меню на экран из модуля BeginProgramBlack вызывается модуль MainMenu.
   1. Для вывода на экран меню ввода эталонного контурного изображения для обучения программы из модуля MainMenu вызывается модуль TrainingBlack. Он запрашивает у пользователя путь к изображению, которому должна быть обучена программа, описание изображения и базу, в которую запишется само изображение с описанием. Далее из модуля Training вызывается модуль TrainingEnd. Он выводит на экран сообщение об успешном сохранении изображения с описанием в определенной базе. После из модуля TrainingEnd вызывается модуль MainMenu для возвращения в главное меню.
   2. Для вывода на экран меню ввода тестового полноцветного изображения для его распознавания программой из модуля MainMenu вызывается модуль Finding. Он запрашивает у пользователя путь к изображению, которое необходимо распознать программой, и базу эталонных изображений, в которой предлагается найти изображение, наиболее похожее на тестовое. Далее из модуля Finding вызывается модуль FindingTest. Он выводит на экран изображение, которое необходимо распознать. Затем из модуля FindingTest вызывается модуль FindingEnd для вывода эталонного изображения из выбранной базы, наиболее похожее на тестовое. После из модуля FindingEnd вызывается модуль MainMenu для возвращения в главное меню.

**Описание программных модулей.**

1. Модуль BeginProgram.m

Функция BeginProgram реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных полноцветных изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 1, необходимая для других модулей, чтобы они работали в режиме работы с полноцветными изображениями.

После создается глобальная переменная palitra, обозначающая количество оттенков красного, зеленого и синего цвета, которые будут участвовать в палитре.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages. Каждый из массивов хранит бинарные комбинации портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные четырехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages. Каждый из массивов хранит изображения портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, четвертый – номер цвета (1 - красный, 2 - зеленый, 3 - синий), а значение представляет собой степень интенсивности определенного цвета пикселя (от 0 до целой части 256/palitra).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers. Каждый из массивов хранит описания портретов и картин соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход полноцветных изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages; их бинарных комбинаций в portraitimages, pictureimages; их описаний в portraitanswers, pictureanswers.

[portraitimages, notpreparedportraitimages]=tobk(palitra, portraitimages,… notpreparedportraitimages, pictureanswers, 'Портреты/',…

'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra,…

pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/',…

'Картины (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

* 1. Функция tobk.

function [images, notpreparedimages, answers]=tobk(palitra, images,…

notpreparedimages,answers, imagesPath, answersPath)

На вход функции подается количество оттенков красного, зеленого и синего цвета palitra, неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

images = zeros(MyJpgFilesQuantity, 10000\*palitra\*3);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

* + 1. Получение набора из трех двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за интенсивность красного, зеленого и синего цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]);

* + 1. Перевод массивов в вектор-строки и последующий перевод в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

for j = 1:3

for k=1:10000

images(i,palitra\*3\*(k-1) + fix(V(1, k, j)/(256/palitra)) + palitra\*(j-1) + 1) = true;

end

end

* + 1. Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for j = 1:3

for k=1:100

for l=1:100

notpreparedimages(i,k,l,j)=fix(MToDisplay(k, l, j)/(256/palitra))\*fix(256/palitra);

end

end

end

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль BeginProgramGrey.m

Функция BeginProgramGrey реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных полутоновых изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 2, необходимая для других модулей, чтобы они работали в режиме работы с полутоновыми изображениями.

После создается глобальная переменная palitra, обозначающая количество оттенков серого цвета, которые будут участвовать в палитре.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages. Каждый из массивов хранит бинарные комбинации портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные трехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages. Каждый из массивов хранит изображения портретов и картин соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, а значение представляет собой степень интенсивности серого цвета пикселя (от 0 до целой части 256/palitra).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers. Каждый из массивов хранит описания портретов и картин соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход полутоновых изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages; их бинарных комбинаций в portraitimages, pictureimages; их описаний в portraitanswers, pictureanswers.

[portraitimages, notpreparedportraitimages]=tobk(palitra, portraitimages,… notpreparedportraitimages, pictureanswers, 'Портреты/',…

'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers]=tobk(palitra,…

pictureimages, notpreparedpictureimages, pictureanswers, 'Картины/',…

'Картины (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

* 1. Функция tobk.

function [images, notpreparedimages, answers]=tobk(palitra, images,…

notpreparedimages,answers, imagesPath, answersPath)

На вход функции подается количество оттенков серого цвета palitra, неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

images = zeros(MyJpgFilesQuantity, 10000\*palitra);

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

* + 1. Получение двумерного массива, взятого из jpg-файла. Массив отвечает за интенсивность серого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay= rgb2gray(imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]));

* + 1. Перевод массивов в вектор-строки и последующий перевод в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

for j = 1:3

images(i, (m-1)\*palitra + fix(V(1, m, 1)/(256/palitra)) + 1)=true;

end

* + 1. Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

for k=1:100

for l=1:100

notpreparedimages(i,k,l)=fix(MToDisplay(k, l, 1)/(256/palitra))\*fix(256/palitra);

end

end

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль BeginProgramBlack.m

Функция BeginProgramBlack реализует работу избирательного нейрона путем обучения с помощью заранее подготовленных контурных изображений и соответствующих описаний к ним.

Вначале создается глобальная переменная strategy со значением 3, необходимая для других модулей, чтобы они работали в режиме работы с контурными изображениями.

Затем определяются глобальные двумерные массивы, хранящие бинарные комбинации эталонных контурных изображений: portraitimages, pictureimages, ECGimages. Каждый из массивов хранит бинарные комбинации портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй означает номер бинарной строки, а значение представляет собой «1» или «0».

Потом определяются глобальные трехмерные массивы, хранящие информацию о цвете пикселей контурных эталонных изображений, готовые к отображению на экран в качестве ответа: notpreparedportraitimages, notpreparedpictureimages, notpreparedECGimages. Каждый из массивов хранит изображения портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов первый индекс означает номер эталонного изображения, второй и третий означают координаты пикселя изображения, а значение представляет собой цвет («1» или «0»).

После определяются глобальные одномерные массивы, хранящие описание соответствующего изображения в виде строки: portraitanswers, pictureanswers, ECGanswers. Каждый из массивов хранит описания портретов, картин и электрокардиограмм соответственно. У каждого из данных массивов индекс означает номер эталонного изображения.

Затем вызывается функция tobk для принятия на вход контурных изображений и их описаний для внесения самих изображений в notpreparedportraitimages, notpreparedpictureimages, notpreparedECGimages; их бинарных комбинаций в portraitimages, pictureimages, ECGimages; их описаний в portraitanswers, pictureanswers, ECGanswers.

[portraitimages, notpreparedportraitimages, portraitanswers] = tobk(portraitimages,…

notpreparedportraitimages, portraitanswers, 'Портреты/', 'Портреты (описание)/');

[pictureimages, notpreparedpictureimages, pictureanswers] = tobk(pictureimages, …

notpreparedpictureimages, pictureanswers, 'Картины/', 'Картины(описание)/');

[ECGimages, notpreparedECGimages, ECGanswers] = tobk(ECGimages, notpreparedECGimages,…

ECGanswers, 'ЭКГ/', 'ЭКГ (описание)/');

Затем происходит вызов модуля MainMenu.m.

MainMenu;

* 1. Функция tobk.

function [images, notpreparedimages, answers]=tobk(images, notpreparedimages,answers,…

imagesPath, answersPath)

На вход функции подается неинициализированные массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers и пути к директориям с jpg-файлами изображений и txt-файлами описаний imagesPath и answersPath. По завершению функции возвращаются готовые к работе массивы изображений notpreparedimages, бинарных строк images, описаний к изображениям answers.

Вначале происходит поиск jpg-изображений внутри директории imagesPath и вычисления количества найденных подходящих файлов.

MyJpgFiles=dir([imagesPath, '\*.jpg']);

MyJpgFilesQuantity=length(MyJpgFiles);

Затем происходит заполнение массива images и notpreparedimages нулями.

notpreparedimages = zeros(MyJpgFilesQuantity, 100, 100);

После происходит работа с каждым найденным эталонным изображением:

* 1. Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread([imagesPath, MyJpgFiles(i).name]), [100 100]), 0.5);

* 1. Перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация, например, в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

for m=1:1:10000

images(i, m)=~V(1,m);

end

* 1. Сохранение изображения массива для вывода (если тестовое изображение будет распознано)

notpreparedimages(i, :, :)=MToDisplay;

Затем происходит поиск txt-файлов в директории answersPath и вычисление кол-ва найденных файлов.

MyTxtFiles=dir([answersPath, '\*.txt']);

MyTxtFilesQuantity=length(MyTxtFiles);

В конце происходит заполнение массива answers первыми строками из найденных файлов.

for i = 1 : MyTxtFilesQuantity

%Взятие первой строки из файла и копирование в соответствующую ячейку массива

fileID = fopen([answersPath, MyTxtFiles(i).name]);

buf = textscan(fileID, '%s','delimiter','\n');

answers{i} = buf{1};

fclose(fileID);

end

1. Модуль MainMenu.m

Модуль выводит на экран меню с двумя кнопками: «Добавить изображение для обучения» и «Распознавать картинку».

* 1. При нажатии на кнопку «Добавить изображение для обучения» запускается модуль Training.m (при работе с полноцветными и полутоновыми изображениями) или TrainingBlack.m (при работе с контурными изображениями).
  2. При нажатии на кнопку «Распознавать картинку» запускается модуль Finding.m (при работе с полноцветными и полутоновыми изображениями) или FindingBlack.m (при работе с контурными изображениями).

1. Модуль Training.m

Модуль Training.m реализует дополнительное обучение обучение нейронной сети для работы с полноцветными и полутоновыми изображениями. Модуль запрашивает у пользователя путь к контурному изображению для обучения нейронной сети, описания изображения и базу данных, в которую следует добавить данное изображение с описанием. После получения необходимых данных происходит запуск функции addimage, которая на их основе добавляет изображение в нейронную сеть.

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

Функция addimage получает на вход массив бинарных строку images, массив изображений notpreparedimages, массив описаний answers, путь к директории, где хранятся изображения imagepath и путь к директории, где хранятся описания descriptionpath.

Если идет работа с полутоновыми изображениями, то вначале происходит получение двумерного массива, взятого из jpg-файла. Если идет работа с полноцветными изображениями, то вначале происходит получение набора из трех двумерных массивов, взятых из jpg-файла. Массив отвечает за интенсивность цветов каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

switch strategy

case 1

MToDisplay=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

MToDisplay=rgb2gray(imresize(imread(get(handles.PathEdit, 'String')), [100 100]));

end

Затем идет перевод массива в вектор-строку, а полученная вектор-строка переводится в бинарную вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.

Потом происходит добавление бинарной строки в общую матрицу бинарных строк images, изображения в общую матрицу изображений notpreparedimages, описания в общую матрицу описаний answers и исходных jpg- и txt-файлов в соответствующие директории.

В конце происходит переход к модулю TrainingEnd.m

1. Модуль TrainingBlack.m

Модуль TrainingBlack.m реализует дополнительное обучение обучение нейронной сети для работы с контурными изображениями. Модуль запрашивает у пользователя путь к контурному изображению для обучения нейронной сети, описания изображения и базу данных, в которую следует добавить данное изображение с описанием. После получения необходимых данных происходит запуск функции addimage, которая на их основе добавляет изображение в нейронную сеть.

function addimage(images, notpreparedimages, answers, imagepath, descriptionpath, handles)

Функция addimage получает на вход массив бинарных строку images, массив изображений notpreparedimages, массив описаний answers, путь к директории, где хранятся изображения imagepath и путь к директории, где хранятся описания descriptionpath.

Вначале происходит получение двумерного массива, взятых из jpg-файла. Массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

MToDisplay=im2bw(imresize(imread(get(handles.PathEdit, 'String')), [100 100]), 0.5);

Затем идет перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

BufBinary = zeros(1, 10000);

for m=1:1:10000

BufBinary (i, m)=~V(1,m);

end

Потом происходит добавление бинарной строки в общую матрицу бинарных строк images, изображения в общую матрицу изображений notpreparedimages, описания в общую матрицу описаний answers и исходных jpg- и txt-файлов в соответствующие директории.

images(size(images, 1)+1, :) = BufBinary(1, :);

notpreparedimages(size(notpreparedimages, 1)+1, :, :)=MToDisplay;

answers(length(answers)+1)=get(handles.DescriptionEdit, 'String');

imwrite(MToDisplay, [imagepath, num2str(size(notpreparedimages, 1)), '.jpg']);

fileID = fopen([descriptionpath, num2str(length(answers)), '.txt'], 'a');

cell=answers(length(answers));

fprintf(fileID, '%s' , cell{1});

fclose(fileID);

В конце происходит переход к модулю TrainingEnd.m

1. Модуль TrainingEnd.m.

Модуль выводит на экран окно с уведомлением «Изображение сохранено» и вызывает модуль MainMenu.m для перехода в главное меню.

1. Модуль Finding.m.

Модуль Finding.m реализует распознавание тестового полноцветного или полутонового изображения. Модуль запрашивает у пользователя путь к полноцветному или полутоновому изображению для распознавания, и базу эталонных изображений, среди которых следует искать данное изображение. После получения необходимых данных происходит следующее:

* 1. Получение двумерных массивов, взятых из jpg-файла. Каждый массив отвечает за интенсивность своего цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

switch strategy

case 1

notpreparedtest=imresize(imread(get(handles.PathEdit, 'String')), [100 100]);

case 2

notpreparedtest=imresize(rgb2gray(imread(get(handles.PathEdit, 'String'))), [100 100]);

end

* 1. Перевод массива в вектор-строку. Кодирование интенсивности происходит за счет движения единицы по длине элемента.
  2. Выбор базы изображений, среди которых будет идти поиск наиболее подходящего эталонного изображения и перемножение матрицы эталонных бинарных строк и тестовой бинарной строки.

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

elseif

res = pictureimages \* testimage';

imagekind='picture';

end

Результатом будет вектор-столбец, где каждое значение будет хранить количество пикселей тествого изображения, совпадающих с соответствующими пикселями эталонных изображений.

* 1. Поиск максимального элемента в полученном векторе, т. е. наиболее схожего эталонного изображения с тестовым.

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

* 1. Сохранение номера наиболее похожего эталонного изображения, если порог превышен. Иначе сохраняем нулевой номер (отсутствие похожего эталонного изображения).

global ImageToShow;

if max > threshold

ImageToShow=imax;

else

ImageToShow=0;

end

В конце происходит вызов модуля FindingTest.m.

1. Модуль FindingBlack.m.

Модуль FindingBlack.m реализует распознавание тестового контурного изображения. Модуль запрашивает у пользователя путь к контурному изображению для распознания, и базу эталонных изображений, среди которых следует искать данное изображение. После получения необходимых данных происходит следующее:

* 1. Получение двумерного массива, взятого из jpg-файла. Массив отвечает за наличие/отсутствие белого цвета каждого пикселя соответственно. После идет перевод изображения N\*N к размеру 100\*100 по методу ближайшего соседа.

notpreparedtest=imresize(im2bw(imread(get(handles.PathEdit, 'String')), 0.5),…

[100 100]);

* 1. Перевод массива в вектор-строку. Так как она уже бинарная, то перевод в бинарную строку не требуется. Но, так как важная информация в ЭКГ черного цвета, а фон – белого, то происходит инвертирование бинарной строки (черному пикселю теперь соответствует «1», белому – «0»).

V(:,:)=reshape(notpreparedtest, 1, 10000);

testimage = zeros(1, 10000);

for m=1:1:10000

testimage(1, m)=~V(1,m);

end

* 1. Выбор базы изображений, среди которых будет идти поиск наиболее подходящего эталонного изображения и перемножение матрицы эталонных бинарных строк и тестовой бинарной строки.

if get(handles.PortraitRadio, 'Value') == 1

res = portraitimages \* testimage';

imagekind='portrait';

elseif get(handles.PictureRadio, 'Value') == 1

res = pictureimages \* testimage';

imagekind='picture';

else

res = ECGimages \* testimage';

imagekind='ECG';

end

Результатом будет вектор-столбец, где каждое значение будет хранить количество пикселей тествого изображения, совпадающих с соответствующими пикселями эталонных изображений.

* 1. Поиск максимального элемента в полученном векторе, т. е. наиболее схожего эталонного изображения с тестовым.

max = 0;

imax = 0;

for i = 1 : length(res)

if res(i) > max

max = res(i);

imax = i;

end;

end

* 1. Сохранение номера наиболее похожего эталонного изображения, если порог превышен. Иначе сохраняем нулевой номер (отсутствие похожего эталонного изображения).

global ImageToShow;

if max > threshold

ImageToShow=imax;

else

ImageToShow=0;

end

В конце происходит вызов модуля FindingTest.m.

1. Модуль FindingTest.m.

Модуль выводит на экран окно с тестовым изображением и уведомлением «Test Picture» и переходит к модулю FindingEnd.m.

1. Модуль FindingEnd.m.

Модуль выводит на экран окно с наиболее похожим эталонным изображением с соответствующим описанием и переходит к модулю MainMenu.m. для возврата к главному меню.

Вначале в зависимости от того, какая база изображений использовалась для распознавания тестового, берутся соответствующие массивы бинарных строк images и массивы изображений notpreparedimages.

switch imagekind

case 'portrait'

notpreparedimages=notpreparedportraitimages;

answers=portraitanswers;

case 'picture'

notpreparedimages=notpreparedpictureimages;

answers=pictureanswers;

case 'ECG'

notpreparedimages=notpreparedECGimages;

answers=ECGanswers;

end

Если тестовое изображение распознано, то в зависимости от того, с какой цветностью изображений работает программа выводится либо полноцветное, либо полутоновое, либо контурное эталонное изображение с описанием. Иначе выводится сообщение 'Image not found'.

**Выводы ко 2 главе.**

На основании используемой математической модели избирательного нейрона и алгоритмов, связанных с распознаванием изображения можно сделать следующие выводы.

1. Модель избирательного нейрона в большей степени учитывает особенности реального биологического нейрона по сравнению с моделью искусственного нейрона МакКаллока – Питса и моделью динамического нейрона Ходжкина—Хаксли.
2. Принципиальным отличием избирательного нейрона от искусственного нейрона МакКаллока—Питтса и от динамического нейрона на основе математической модели Ходжкина—Хаксли и других моделей подобного рода является полное отсутствие весовых коэффициентов для входных воздействий и всех недостатков нейронных сетей, связанных с наличием весового суммирования.
3. Ввод данных на вход используемой нейронной сети осуществляется с помощью перевода входных изображений квадратной формы формата jpg в бинарную кодовую строку.
4. В программе используются такие алгоритмы, как:

4.1. алгоритм работы с эталонными изображениями квадратной формы формата jpg (обучение);

4.2. алгоритм работы с тестовым изображением квадратной формы формата jpg (распознавание);

4.3. метод ближайшего соседа (масштабирование изображений).

1. Из представленных выше методов бинарных строк могут использоваться только методы кодирования интенсивности за счет движения единицы по длине элемента.

# Глава 3.

**Экспериментальная проверка и оценка программного обеспечения для распознавания изображений с помощью нейронных сетей на избирательных нейронах.**

## 3.1. Описание результатов испытаний

Испытания проводятся для того, чтобы убедиться, что разработанное программное обеспечение удовлетворяет требованиям поставленной задачи.

При испытании были проведены следующие эксперименты.

1. Распознавание писателя по изображению.
2. Распознавание картины по ее изображению.
3. Распознавание писателя по изображению, отсутствующего в базе.

При проведении перечисленных экспериментов были получены следующие результаты.

1. Распознавание писателя по изображению происходит верно.
2. Распознавание картины по ее изображению происходит верно.
3. Программа выдает сообщение о том, что введенное изображение отсутствует в базе.

Во время проведения испытаний программа работала без сбоев.

Результаты проведенных испытаний говорят о том, что разработанное программное обеспечение удовлетворяет поставленным требованиям. Программа успешно обработала некорректно введенные данные и работала стабильно, без аварийных завершений и другого непредвиденного поведения. Обозначенные в постановке задачи на дипломное проектирование требования удовлетворены в полном объеме.

## 3.2. Оценивание надежности программного обеспечения

Надежность программного обеспечения – это его способность с достаточно большой вероятностью безотказно выполнять определенные функции в течение заданного периода времени при заданных условиях. Считается, что в программе произошел отказ, если она не выполнила функции, предусмотренные в техническом задании. Однако наличие ошибок в программе не всегда приводит к отказу. Программа считается надежной, если при ее практическом использовании в требуемых условиях отказы происходят достаточно редко. Убедиться в надежности программы можно при проведении тестирования, а впоследствии и при использовании в практических целях.

Для подсчета количественных показателей надежности в данной дипломной работе будем использовать модель Коркорэна. Она относится к статическим моделям надежности ПО. Модель рассчитывается на основе N испытаний, в которых произошло Ni ошибок i-го типа, и не связывает возникновение отказов со временем. Для различных типов ошибок модель учитывает изменяющиеся вероятности отказов. Модель Коркорэна выгодно отличается от других моделей тем, что ее относительно легко рассчитать.

В модели Коркорэна для оценки вероятности безотказного выполнения программы используется следующая формула:



где R – показатель уровня надежности,

N0 –число безотказных выполнений программы,

N – общее число выполнений,

K – известное заранее число типов ошибок,



Ni – количество ошибок i-го типа,

ai – вероятность появления ошибки i-го типа.

В модели Коркорэна вероятность ai определяется на основе заранее известных результатов работы схожих по функциональности программных средств.

При тестировании программы было выполнено 80 прогонов. Число ошибок каждого типа и вероятности их появления указаны в таблице 6. 71 испытание из 80 было произведено успешно. Всего во время проведения испытаний произошло 9 отказов программы, вызванных одной ошибкой вычисления, одной логической ошибкой, тремя ошибками ввода-вывода, двумя ошибками манипулирования данными и двумя ошибками в базе данных.

**Таблица 6.**

**Выявленные ошибки**

|  |  |  |
| --- | --- | --- |
| Тип ошибки | Вероятность появления (ai) | Количество выявленных ошибок (Ni) |
| Ошибки вычисления | 0.09 | 1 |
| Логические ошибки | 0.26 | 1 |
| Ошибки ввода-вывода | 0.16 | 3 |
| Ошибки манипулирования данными | 0.18 | 2 |
| Ошибки сопряжения | 0.17 | 0 |
| Ошибки определения данных | 0.08 | 0 |
| Ошибки в БД | 0.06 | 2 |

Вычислим показатель надежности программы, используя вышеприведенную формулу. В нашем случае N0 = 71, N = 80, K = 7. Отсюда получаем вероятность безотказной работы программы



Таким образом, вероятность того, что программа будет работать без сбоев, равна 89,45%. Такое высокое значение говорит о том, что программа для распознавания текстов с помощью нейронных сетей на избирательных нейронах полностью готова к практическому использованию. Программное обеспечение показало высокую степень надежности.

## 3.3. Обоснование экономической эффективности

### 3.3.1. Методика расчета экономической эффективности

Каждый проект внедрения программного обеспечения обработки информации должен быть обоснован расчетом его ожидаемой экономической эффективности.

Для расчета экономической эффективности внедрения данного программного обеспечения сравним затраты при двух вариантах решения задачи:

1. Базовый вариант - работник компании решает задачу собственными силами без использования специально разработанного для этих целей программного средства.
2. Проектируемый вариант - для решения задачи используется программное обеспечение из данной дипломного работы.

Экономическая эффективность подразделяется на две составляющие: косвенный эффект и прямой эффект.

Косвенный эффект сложно подсчитать, так как он по большей части характеризуется улучшением качественных показателей, например, снижением затрат на сырье и материалы, уменьшением процента брака, привлечением большего числа клиентов, повышением прибыли, уменьшением сумм штрафов. Из-за этого осложнено выявление доли косвенного эффекта от программной обработки данных в общей экономической эффективности, тесно связанной с комплексом мер по повышению производительности компании.

Прямой эффект, в свою очередь, легко формализуется, потому что он отражает экономию материально-трудовых ресурсов и денежных средств, полученную в результате автоматизации решения задач.

К трудовым относятся следующие показатели.

1. Абсолютный показатель снижения трудовых затрат - разность между годовыми трудовыми затратами базового и проектируемого вариантов обработки данных.



где Tб – годовая трудоемкость обработки информации по базовому варианту,

Tп – годовая трудоемкость обработки информации по проектируемому варианту.

1. Коэффициент снижения трудовых затрат отображает значение относительного снижения трудовых затрат.

.



1. Индекс снижения трудовых затрат отображает повышение производительности труда по причине внедрения автоматизированной обработки данных.



К стоимостным показателям относятся:

1. Экономия финансовых затрат за счет автоматизации решения задачи рассчитывается как разница между затратами по базовому и проектируемому вариантами.



где Сб – стоимостные затраты на решение задачи по базовому варианту,

Сп – стоимостные затраты на решение задачи по второму варианту.

Стоимостные затраты на обработку информации по проектируемому варианту рассчитываются по формуле



где Цмч – стоимость машинного часа,

Зч – средняя зарплата оператора за один час,

Кнр – коэффициент накладных расходов.

1. Коэффициент относительного снижения стоимостных затрат отображает, на сколько процентов снизятся затраты при переходе на предлагаемый вариант решения задачи.



1. Индекс снижения затрат показывает, во сколько раз снизятся затраты на решение задачи после внедрения проектируемого варианта.



Кроме показателей, перечисленных выше, будет также полезно подсчитать и другие.

1. Годовой экономический эффект равен сумме прямого и косвенного эффектов.



где Эп – прямой экономический эффект,

Эк – косвенный экономический эффект.

Прямая составляющая экономической эффективности рассчитывается по формуле



где Ен – нормативный коэффициент эффективности вложений капитала,

Кб – затраты капитала при базовом варианте обработки информации,

Кп – затраты капитала при проектируемом варианте обработки информации,

∆К – дополнительные затраты капитала при миграции с базового варианта на проектируемый.

1. Расчетный коэффициент эффективности равен отношению абсолютного снижения стоимостных затрат к дополнительным капитальным вложениям на переходный период.



1. Время окупаемости вложений на внедрение проекта автоматизированной обработки информации обратно пропорционально расчетному коэффициенту эффективности.



### 4.3.2. Расчет показателей экономической эффективности

Используем информацию, приведенную выше, для подсчета показателей экономической эффективности программного обеспечения, разработанного в рамках данной дипломной работы. В первую очередь определим трудовые ресурсы и денежные средства, затрачиваемые на распознавание портретов при базовом и предлагаемом вариантах. Рассчитанные экономические показатели представим в виде таблиц и графиков для удобства восприятия.

В год сотруднику требуется распознать около тысячи портретов. До внедрения предлагаемого программного средства на каждое распознавание он тратил в среднем 12 минут. Таким образом, годовая трудоемкость решения задачи по базовому варианту равна



Зарплата сотрудника составляет 42000 рублей в месяц или приблизительно 250 рублей в час. Отсюда получаем годовые стоимостные затраты на решение задачи по базовому варианту



C использованием разработанного в рамках данного дипломной работы программного обеспечения для распознавания изображений затраты материально-трудовых ресурсов и денежных средств снизятся.

Трудовые затраты сотрудника, решающего задачу по проектируемому варианту включают в себя две составляющие. Во-первых, время на ознакомление с программным средством, изучение способов работы с ним составит 80 минут. Во-вторых, время, затрачиваемое на распознавание одного художественного произведения, составит в среднем 1 минуту. Отсюда получаем годовую трудоемкость решения по проектируемому варианту



Теперь посчитаем абсолютные показатель снижения трудовых затрат.



Коэффициент относительного снижения трудовых затрат равен



И наконец индекс снижения трудовых затрат получаем равным приблизительно



Теперь рассчитаем стоимостные экономические показатели. Нам уже известно, что сотрудник при работе по предлагаемому варианту в год на решение задачи затрачивает 18 часов. Допустим, что при переходе на автоматизированную обработку данных его зарплата не изменится. Стоимость машинного часа примем равной 10 рублям и возьмем стандартное значение коэффициента накладных расходов в 0,1. Стоимостные затраты на решение задачи по проектируемому варианту будут равны



Теперь рассчитаем абсолютное снижение стоимостных затрат.



Коэффициент относительного снижения стоимостных затрат равен



В конце получим индекс снижения стоимостных затрат



В таблице 7 представлены все вычисленные трудовые и стоимостные показатели экономической эффективности перехода на предлагаемый вариант решения задачи с использованием программного обеспечения для распознавания портретов.

Следующим шагом рассчитаем оставшиеся показатели экономической эффективности всего проекта. Сначала вычислим годовой экономический эффект. Для этого определим себестоимость разработанного в рамках данной дипломной работы программного обеспечения. Она складывается из капитальных вложений на покупку, поддержку и эксплуатацию программно-аппаратного комплекса, использованного при разработке, и расходов на заработную плату разработчиков. Затраты на заработную плату разработчикам вычислим по формуле



где n – число разработчиков, участвовавших в проекте,

C – заработная плата специалиста (в час),

T – количество человеко-часов, потребовавшихся для разработки данного дипломного проекта.

Программа разрабатывалась в течение трех месяцев с участием одного разработчика, заработная плата которого составляла 21000 рублей в месяц или примерно 125 рублей в час. Трудоемкость разработки составила 504 человека-часа. Расходы на заработную плату разработчиков за весь период равны



Единовременные капитальные вложения считаем равными K = 7000 рублей. Отсюда получаем суммарную себестоимость данного программного средства:



**Таблица 7.**

**Показатели экономической эффективности.**

|  |  |  |
| --- | --- | --- |
| Тип показателя | Трудовые показатели | Стоимостные показатели |
| Затраты при базовом варианте решения задачи | 200 часов | 50000 руб. |
| Затраты при проектируемом варианте решения задачи | 18 часов | 5130 руб. |
| Абсолютное снижение затрат | 192 часа | 44870 руб. |
| Относительное снижение затрат | 91 % | 89,74% |
| Индекс снижения затрат | 11,1 раз | 9,7 раз |

Себестоимость использования базового варианта считаем равным нулю. Тогда дополнительные затраты капитала при миграции с базового варианта на проектируемый равны



Нормативный коэффициент эффективности считаем равным Ен = 0,15. Теперь можем рассчитать прямой экономический эффект. Он равен



Косвенный экономический эффект от миграции на автоматизированный вариант обработки информации проявляется, например, из-за сокращения количества ошибок в расчетах. В итоге мы получили, что годовой экономический эффект равен



Определим теперь расчетный коэффициент эффективности.



Время окупаемости вложений составит



Так как годовой экономический эффект получился достаточно невысоким, требуется принимать во внимание снижение трудовых затрат сотрудников, занятых решением задачи, а также учитывать косвенный экономический эффект, который непросто формализовать и выразить в цифрах, как уже было сказано выше. По итогам общей оценки рассчитанных значений экономических показателей можно сделать вывод о том, что с экономической точки зрения переход на использование разработанного программного средства эффективен.

Диаграммы снижения трудовых и стоимостных затрат на распознавание портретов при базовом и предлагаемом вариантах приведены на рисунках 21 и 22.

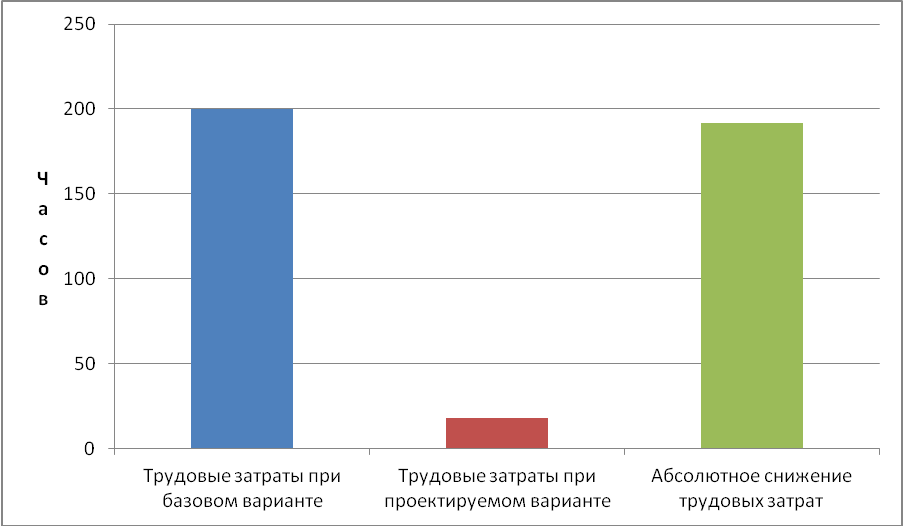


Рис. 22. Диаграмма снижения трудовых затрат

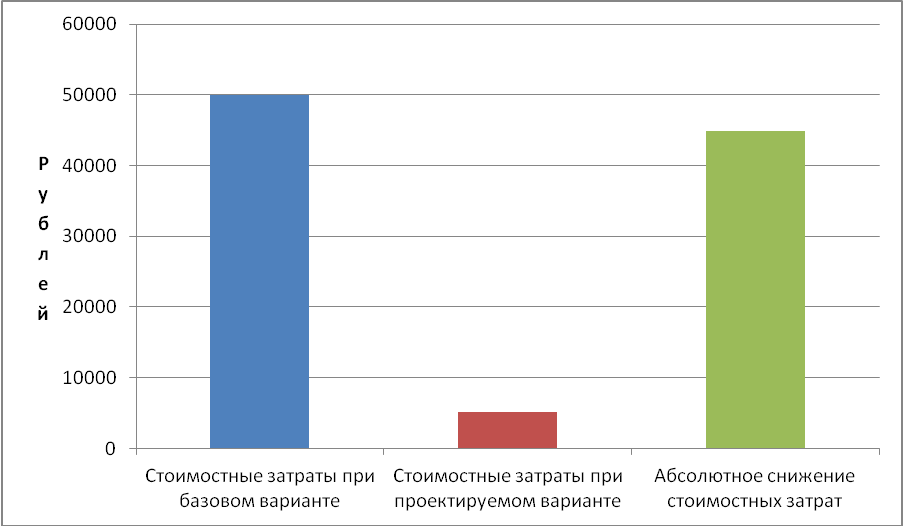


Рис. 23. Диаграммы снижения стоимостных затрат

Из этих диаграмм хорошо видно, что относительные снижения трудовых и стоимостных затрат примерно равны между собой. Так происходит по причине того, что основная часть затрат на решение задачи представляет собой расходы на заработную плату сотрудников.

## Выводы по главе 3

Третья глава дипломной работы посвящена экспериментальному тестированию и оценке экономической эффективности разработанного программного обеспечения для распознавания изображений с помощью нейронных сетей на избирательных нейронах.

В ходе проведения тестирования не было зафиксировано сбоев в работе программы. Программа работала корректно, выдавала правильные результаты, на введенные пользователем данные отвечала согласно ожиданиям.

Были рассмотрены основные модели оценки надежности программных средств. Для оценки надежности разработанного программного обеспечения была выбрана модель Коркорэна как наиболее подходящая в данном случае.

После всех вычислений коэффициент надежности программы получился равным примерно 89%. Такой относительно высокий результат говорит о готовности программы к внедрению в практическую эксплуатацию.

В последнем параграфе были вычислены основные показатели экономической эффективности программного обеспечения. Эти вычисления заключались, главным образом, в сравнении между собой затрат при двух вариантах решения задачи:

1. Базовый вариант - работник компании решает задачу собственными силами без использования специально разработанного для этих целей программного средства.

2. Проектируемый вариант - для решения задачи используется программное обеспечение из данной дипломной работы.

В результате проведенных исследований было вынесено решение о том, что разработанное программное обеспечение для распознавания изображений с помощью нейронных сетей на избирательных нейронах имеет высокую экономическую эффективность.

# ЗАКЛЮЧЕНИЕ

Итогом написания данной дипломной работы является решение задачи разработки программного обеспечения для распознавания изображений при помощи нейронных сетей на избирательных нейронах. Разработанная программа удовлетворяет всем условиям, поставленным в момент постановки задачи. Сформулированная цель данной дипломной работы полностью достигнута.

Исследования возможностей существующего программного обеспечения, которое базируется на нейроне Маккалока – Питтса, показало необходимость разработки новой программы, которая будет обладать рядом преимуществ и не будет иметь недостатков предшественников.

Разработанная программа была протестирована и отлажена, а также была подвергнута оценке надежности и экономической эффективности.

В результате вычислений установлена высокая вероятность того, что разработанная программа будет работать без сбоев. Сделан вывод, что программа для распознавания изображений с помощью нейронных сетей на избирательных нейронах готова к практической эксплуатации.

На основании результатов вычисления основных экономических показателей вынесено заключение о высокой экономической эффективности разработанного в рамках данной дипломной работы программного обеспечения.

Программное обеспечение распознает портреты и картины из числа сохраненных в ее базе. При этом для успешного распознавания достаточно ввести путь к файлу тестового изображения.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Александров Ю.И., Анохин К.В., Соколов Е.Н., Греченко Т.Н. и др. Нейрон. Обработка сигналов. Пластичность. Моделирование. Фундаментальное руководство //Изд-во Тюменского государственного университета. 2008 . 548 с.
2. Анохин П.К. Принципиальные вопросы общей теории функциональных систем // Принципы системной организации функций. – М.: Наука, 1973.
3. Беркинблит М.Б. Нейронные сети – М.: Издательство МИРОС, 1993.
4. Вальцев В.Б, Лавров В.В. Целесообразное фрагментирование информации на входе в мозг // "Информационные технологии", №2, 2006, с.22-30.
5. Веденов А. А. Моделирование элементов мышления. М.: Наука, 1988. 159 с.
6. Виноградова О.С. Нейронаука конца второго тысячелетия: смена парадигм // Журнал высш. нервн. деятел. 2000. Т. 50. С. 743-774.
7. Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. — Харьков: Основа, 1997. — 112 с. — ISBN 5-7768-0293-8.
8. Голубев Ю. Ф. Нейросетевые методы в мехатронике. — М.: Изд-во Моск. унта, 2007. — 157 с. — ISBN 978-5-211-05434-9.
9. Горбань А. Н., Дунин-Барковский В. Л. и др. Нейроинформатика. — Новосибирск: Наука, 1998.
10. Дунин-Барковский В. Л., Терехин А. Т. Нейронные сети и нейрокомпьютеры: тенденции развития исследований и разработок // Микропроцеcсорные средства и системы. 1990. N 2. C. 12-14.
11. Еремин Д. М., Гарцеев И. Б. Искусственные нейронные сети в интеллектуальных системах управления. — М.: МИРЭА, 2004. — 75 с. — ISBN 5-7339-0423-2.
12. Клини С. Математическая логика// М.: Мир, 1973. 480 с.
13. Комарцова Л. Г., Максимов А. В. Нейрокомпьютеры. — 1-е. — Изд-во МГТУ им. Н.Э. Баумана, 2002. — С. 320. — ISBN 5-7038-1908-3
14. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия - Телеком, 2001. — 382 с. — ISBN 5-93517-031-0.
15. Мак-Каллок У., Питтс В. Логическое исчисление идей, относящихся к нервной активности // Нейрокомпьютер. –1992. - №3/4. – Р.40-50.
16. Минский М., Пейперт С. Перцептроны / пер. с англ. М.: Мир, 1971. 261 с. (Англ. оригинал: Minsky M., Papert S. Perceptrons: An Introduction to Computational Geometry. Cambridge, MA: MIT Press, 1969.).
17. Миркес Е. М. Нейрокомпьютер. Проект стандарта. — Новосибирск: Наука, 1999. — 337 с. — ISBN ISBN 5-02-031409-9.
18. Николлс Дж., Мартин Р., Валлас Б., Фукс П. От нейрона к мозгу – М.: Издательство Эдиториал УРСС, 2003.
19. Розенблат Ф. Принципы нейродинамики. Перцептроны и теория механизмов мозга / пер. с англ. М.: Мир, 1965. 175 с. (Англ. оригинал: Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, D.C.: Spartan Books, 1962.)
20. Савельев А. В. На пути к общей теории нейросетей. К вопросу о сложности // Нейрокомпьютеры: разработка, применение. — 2006. — № 4—5. — С. 4—14.
21. Тадеусевич Рышард, Боровик Барбара, Гончаж Томаш, Леппер Бартош. Элементарное введение в технологию нейронных сетей с примерами программ / Перевод И. Д. Рудинского. — М.: Горячая линия — Телеком, 2011. — 408 с. — ISBN 978-5-9912-0163-6..
22. Терехин А.Т., Будилова Е.В. Сетевые механизмы биологической регуляции // Успехи физиологических наук. 1995. Т. 26. N 4. С.75-97.
23. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.
24. Уоссермен Ф. Нейрокомпьютерная техника. Теория и практика // М.: Мир, 1992. 240 с.
25. Фриман Дж.У., Динамика мозга в восприятии и сознании: творческая роль хаоса // В сб. «Синергетика и психология». Вып.3. "Когнитивные процессы", Издательство «Когито-Центр", 2004.
26. Хайкин С. Нейронные сети: полный курс // Neural Networks: A Comprehensive Foundation.- 2-е. М.: «Вильямс», 2006. -1104 с.
27. Цетлин М.Л. Исследования по теории автоматов и моделированию биологических систем. — М.: Наука, 1969. 316с.
28. Швырков В.Б. Теория функциональной системы как методологическая основа нейрофизиологии поведения // Успехи физиологических наук. 1978. Т. 9. №1. С.81-105.
29. Ясницкий Л. Н. Введение в искусственный интеллект. — М.: Издат. центр «Академия», 2005. — 176 с. — ISBN 5-7695-1958-4.
30. Bechtel W, Abrahamsen A. Connectionism and the Mind: Parallel Processing, Dynamics, and Evolution in Networks, 2002.
31. Grossberg Stephen. 1988. Neural Networks and Natural Intelligence. MIT Press, Cambridge, MA.
32. Hebb, D. O. The organization of behavior: a neuropsychological theory. New York (2002) (Оригинальное издание — 1949)
33. Lipman R. An introdaction to computing with neural nets // IEEE Acoustic,Speech and Signal Processing Magazine,1987,no 2,p 4-22.
34. Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Washington, DC: Spartan Books (1962).

Приложение 1.

Метод ближайшего соседа.

Метод ближайшего соседа заключается в следующем. Исходное изображение имеет большое количество пикселей (большее 10000). Изображение подготовлено в виде квадратной формы. Распознавание расчитано на исходные объекты 100х100=10000 пикселей. Поэтому необходимо приведение исходного изображения к стандартной форме. Это преобразование осуществляется с помощью метода ближайшего соседа. В Matlab 7 метод ближайшего соседа реализуется с помощью функции imresize. Идея метода ближайшего соседа заключается в следующем:

Группа близлежащих пикселей заменяется одним пикселем, наиболее близким к этой группе. Близость оценивается с помощью критерия близости.

Один из типовых подходов заключается в разбиении данного отображения прямоугольной сеткой на квадраты, содержащие больше 1 пикселя. Например, сетку 100х100=10000 пкиселей разбиваем на прямоугольники 70х70=4900 пикселей. При этом в каждый прямоугольник попадает от 1 до 3 пикселей. Простейшим способом является взятие среднего арифметического интенсивностей пикселей, попавших внутрь элемента более крупной сетки. Другие способы описаны в программах реализации данного метода.

Для каждого пикселя конечного изображения выбирается один пиксель исходного, наиболее близкий к его положению с учетом масштабирования.