4. Реализация конкретного примера распознавания изображений

4.1. Введение

4.2. Типы входных изображений

4.3. Создание объектной модели нейронной сети

4.4. Использование Баз Данных в распознавании

4.5. Анализ помех в изображении

4.6. Заключение

4.1. Введение

Рассмотрев основные вопросы и методы теории распознавания изображений на основе возможностей нейронных сетей, можно перейти к реализации конкретного задания. Целью которого, будет являться демонстрация возможностей математических алгоритмов, основанных на реальных природных процессах, которые происходят в мозгу человека, при решении им подобной задачи. Как уже отмечалось ранее, невозможно создать реальный эмулятор сети на базе современных последовательных компьютеров, вследствие невозможности реализации основного принципа работы НС – параллельности происходящих в ней процессов. Таким образом, создание реально действующего эмулятора сети будет возможно только на основе специальных компьютеров - нейрокомпьютеров, которые в настоящее время находятся только в стадии разработки. Однако за счет постоянного увеличения быстродействия ЭВМ за последнее время, стало возможно создание эмулятора сети, в определенной степени приближенного к реальности, который смог бы решать не очень сложные задачи, не требующие большого количества нейронов в слоях сети. В нашем случае эмулирование сети будем производить на персональном компьютере, на базе процессора Intel Pentium-200MMX, с объемом оперативной памяти 64 Мб. Мощности данного, в настоящее время не столь дорогого, компьютера должно вполне хватить на эмулирование параллельной работы сравнительно небольшого количества нейронов.

В качестве языка программирования был выбран объектно-ориентированный Паскаль (Pascal), вследствие довольно мощной реализации в нем принципов объектного программирования, что облегчит создание модели сети, разложив данный процесс на отдельные простые шаги: создание каждого нейрона в отдельности, затем сборка из них слоев сети, и, наконец, сборка самой сети из созданных слоев. Моделирование сети на основе объектной модели облегчит и сделает более наглядным весь процесс программирования. В качестве интерпретатора данного языка была выбрана среда Borland Delphi, так как в ней реализованы возможности быстрого создания удобного интерфейса конечного продукта, коим будет являться программа. Однако такой выбор не является единственным приемлемым способом программирования НС. В качестве примера можно назвать реализацию самоорганизующихся карт Кохонена, для решения задачи распознавания лингвистических символов, основанную на языке JavaScript, и представленную в сети Интернет (Internet). Пример ее работы можно посмотреть по адресу *http://www.hav.com/nnhtml.htm*. Также очень часто программирование сетей производят на языке C++, вследствие большой популярности данного языка среди программистов, и огромного количества его интерпретаций на различных платформах, что влечет за собой его некоторую универсальность. Примеры программирования различных видов НС на C++ можно посмотреть также в сети Интернет по адресу http://www.geocities.com/CapeCanaveral/1624/ .

В данном параграфе будут рассмотрены конкретные вопросы программирования по шагам:

- постановка задачи;
- вид входных изображений и объектов для распознавания;
- выбор и моделирование архитектуры сети;
- использование Баз Данных для хранения эталонных изображений, структуры сети и результатов ее обучения.

4.2. Типы входных изображений

Так как основной задачей реализуемой НС является простой пример распознавания объектов из изображений, то зададимся характером этих изображений, ограничениями, налагаемыми на них, а также типом распознаваемых объектов.

4.2.1. Получение и предварительная обработка изображений

Нами уже были рассмотрены вопросы, связанные с большим разрешением входных изображений, влекущих за собой большой объем сети и ее низкое быстродействие. Поэтому ограничимся размером входных изображений 32×32 пиксела, каждый из которых может принимать значение 0 или 1, то есть изображение является черно-белым. Такого рода ограничения не являются столь жесткими, как может показаться в начале, так как разрешение современных камер и видиконов СТЗ не является столь высоким, или, например, разрешение современных телевизоров равно примерно 700×500 пикселей, однако это не очень мешает нам его смотреть, не напрягая глаз. Не усложняя принципиально архитектуру сети можно добиться изменения состояния каждого пиксела в изображении в диапазоне от 0 до 255, то есть сделать входное изображение либо цветным, с палитрой 256 цветов, либо черно-белым, с 256-ю градациями серого цвета. Однако часто в результате предварительной обработки, которая производится непосредственно перед подачей изображения на модуль СТЗ, занимающийся распознаванием объектов, конечный сигнал содержит только однотонные контуры изображений на также однотонном фоне. То есть обработанное изображение представляет собой черно-белую картину.

Так как сами изображения каким-либо образом будут подаваться на вход программы на ЭВМ, то следует также выбрать формат графических файлов, посредством которых будет осуществляться передача. Как известно на персональных компьютерах есть два основных типа такого рода файлов: растровые и векторные. Так как векторный формат используется не столько для хранения самого изображения в пиксельном виде, сколько для хранения графических объектов, которые находятся в этом изображении, а такого рода априорной информацией мы не обладаем, то остановим свой выбор на растровом формате. Так как само изображение является очень маленьким, то в такого рода выборе мы практически ничем не ограничены. Однако, существуют типы хранения растровых изображений посредством их фрактальной обработки, например формат JPEG, вследствие которой исходная информация, носимая изображением, может в некоторой степени исказиться. Поэтому остановим свой выбор на наиболее известном на данный момент растровом формате Windows Bitmap, или просто BMP. Так как данный формат не использует сжатие изображений и наиболее часто используется в настоящее время, а также не является таким сложным в раскодировке, как TARGA или TIFF.

Рассмотрев ограничения и тип изображений, перейдем к типам объектов, которые мы будем распознавать. Как известно, наиболее частой задачей распознавания для персональных ЭВМ является распознавание лингвистических символов из сканированных изображений, для ускорения ввода в компьютер рукописного текста. В настоящее время уже выпущено в продажу большое количество коммерческих программ, в которых реализованы различные алгоритмы распознавания букв. Однако почти ни один из них не использует для этого методы нейронных сетей, попробуем заполнить этот пробел. Для этого рассмотрим класс объектов, из которых будут составляться изображения, с помощью которых будет производиться обучение сети. Это будет 10

изображений, размером 32×32 пиксела, с черным фоном, в которые будут занесены первые буквы латинского алфавита. Таким образом, выделим еще одно правило, незаметно полученное только что: распознаваемый объект должен иметь белый цвет и располагаться на черном фоне. Для уменьшения количества входов на каждый нейрон входного слоя, а значит и количества весов, которые нам придется хранить, добавим, что размер букв в изображении будет равным 16×16 пикселов.

Выделим основные операции, которые будут производиться над входным изображением перед подачей его на входной слой нейронной сети:

- избавление от единичных помех в изображении;
- нахождение и выделение области 16×16 пикселов, в которой находится объект в изображении;
- получение одномерного массива данных из выделенной области, представляющего собой строку с чередованием нулей и единиц.

Стоит заметить, что совершенно не обязательно обучать сеть только буквам латинского объекта, в конце концов, можно использовать любые графические объекты, которые подходят под вышеперечисленные рамки, не зависимо от их характера.

4.2.2. Принцип получения сумм. Избавление от единичных помех

Для того, чтобы решить проблему выделения объекта из изображения, независимо от его положения в изображении, а также наличия помех, на основе приведенных в главе 3 соображений, следует разработать алгоритм увеличения описания исходного изображения, с целью увеличения информативности сигнала, подаваемого на входной слой нейронной сети. В п.п. 3.3.1.1. с этой целью было предложено рассчитать суммы пикселов белого цвета (то есть, отличных от цвета фона) по столбцам и строкам изображения, и в дальнейшем оперировать изображением на их основе. Проанализируем

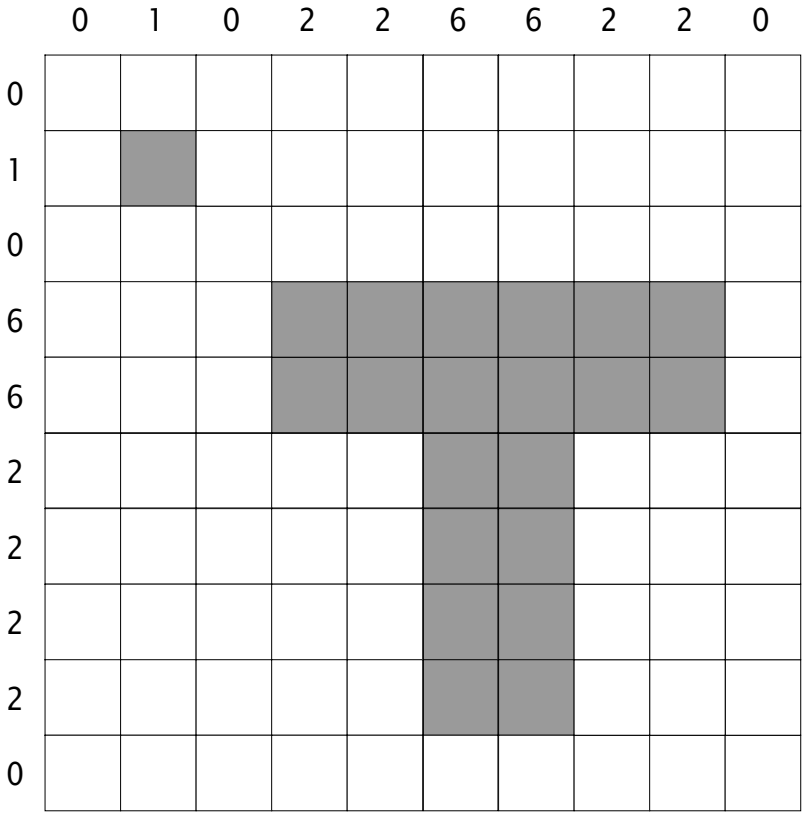


Рис. 4.1

данный метод применительно к нашей конкретной задаче.

Ненадолго вернемся к примеру распознавания буквы «Т», и рассмотрим более подробно каждый шаг. На рис. 4.1 изображена сетка пикселов 10×10, в которой находится данная буква, а также единичная помеха в левом верхнем углу, от которой нам предстоит избавиться. Цифры по краям сетки обозначают вертикальные и горизонтальные суммы активных пикселов.

Если внимательно присмотреться к суммам вокруг единичной помехи, то можно заметить, что по ее краям суммы равны 0, а вертикальные и горизонтальные суммы на ее местоположении малы, что свидетельствует о том, что в наличии имеется ничтожно малый объект (в данном случае единичный пиксель). На основании информации о том, что размеры нашего реального объекта в изображении должны быть около 16×16 пикселов, можно сделать вывод, что в данном случае мы имеем дело не с объектом, а с единичной помехой, которую следует удалить из изображения, что и представлено на рис. 4.2.

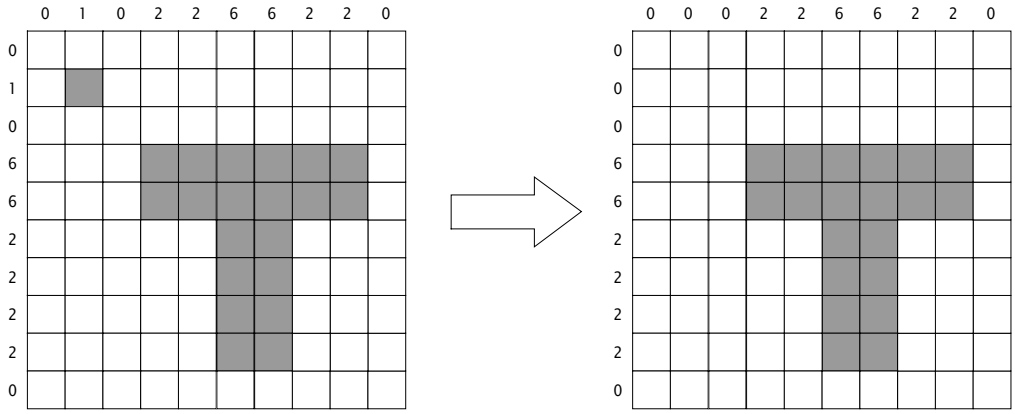


Рис. 4.2

Обобщим данный, довольно простой, алгоритм выделения единичных помех:

1. *Расчет сумм активных пикселов в изображении.*
   Получение двух одномерных массивов, длина каждого из которых равна количеству пикселов по вертикали и горизонтали в изображении.
2. *Чтение полученного массива сумм по вертикали до получения первого члена, неравного 0.*
3. *Проверка рядом стоящих членов массивов на неравенство 0.*
   Запоминаем индекс *i* первого члена массива, неравного 0. И проверяем значения членов *i*-1 и *i*+1 на том же условии. Если условие верно, то заносим данный индекс в стек.
4. *Проведение операций 2 и 3 для горизонтального массива.*
5. *Сравнение совпадений индексов из стека для вертикального массива и горизонтального.*
   В результате проверки на совпадение станут ясны размеры полученного объекта, и если они ничтожно малы, то следует стереть данный объект из изображения, путем установки соответствующих сумм равными 0.

Среди явных недостатков данного метода можно заметить довольно ограниченное количество помех, которые можно распознать через этот алгоритм. Помехи могут быть только единичными, и полностью изолированными от самого объекта в изображении.

Очевидным и главным же достоинством является простота данного метода, наглядность, и очевидная легкость программирования.

4.2.3. Выделение объекта из изображения

После удаления единичных помех из изображения, второй основной задачей предварительной обработки является выделение области 16×16 пикселов, в которой находится объект. Причем в данной задаче следует учесть инвариантность выделения

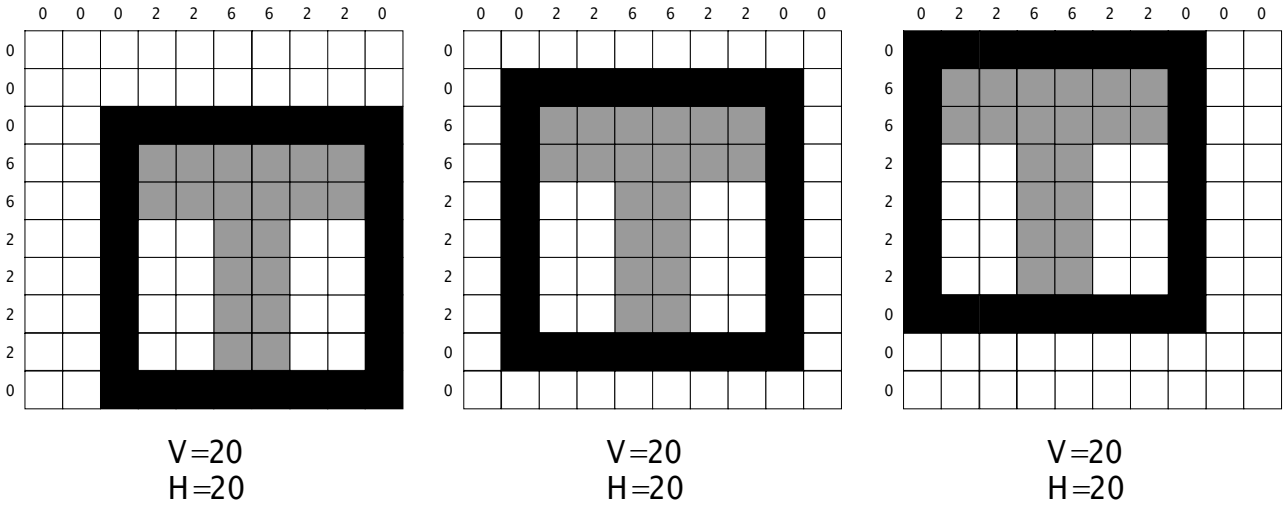


Рис. 4.3

подобной области к перемещению объекта на экране. Решим данную задачу также на основе использования принципа расчета сумм в изображении.

На рис. 4.3 черным цветом обведена область местонахождения объекта в изображении. Как видно из рассчитанных сумм, независимо от расположения объекта на экране, сумма пикселов по вертикали и горизонтали остается постоянной, то есть инвариантной к перемещению объекта, что можно уже взять за основу, при решении вопроса о наличии объекта в изображении. Также при построении области местонахождения объекта на экране на рис. 4.3 можно заметить закономерность значений сумм, которые использовались в данной операции. Таким образом, немного модернизировав метод, описанный в предыдущем подпункте, можно построить алгоритм, единственным отличием которого станет нахождение не единичного объекта в области между нулевыми суммами, а целого массива пикселей, определяющих объект. Причем каждому отдельному объекту в изображении будут соответствовать своя пара массивов по горизонтали и вертикали, по которой можно будет определить не только расположение самого объекта или нескольких объектов в изображении, но также провести первичное, пусть даже не совсем точное, распознавание. Однако в данном случае встает вопрос об инвариантности первичного распознавания к ориентации объектов на экране. Так как, например, малейший поворот буквы «Т» на экране приведет к категоричному изменению значений сумм, соответствующих данному символу. Следовательно, первичное распознавание, основанное на вышеописанном методе, может привести к непредсказуемым результатам, являющимся в высокой мере зависимыми от внешних обстоятельств, что зачастую в заводских условиях является просто неприемлемым.

Таким образом оставим задачу распознавания на долю НС, и ограничимся лишь передачей на ее входы области расположения объекта из в некоторой степени «очищенного» изображения.

4.3. Создание объектной модели нейронной сети

После решения вопроса о характеристиках и виде входного сигнала первого слоя нейронной сети, а также постановки задачи на распознавание, мы имеем достаточно информации, чтобы перейти к выбору архитектуры сети и созданию ее объектной модели.

4.3.1. Выбор архитектуры сети

В качестве первого шага при выборе архитектуры сети мы должны провести анализ существующих на данный момент видов сетей, описанных в параграфе 2. Как уже отмечалось раннее, однослойный перцептрон, либо однослойная НС, могут решать только самые простые задачи распознавания и классификации. Но рассмотрим поближе нашу задачу, основной работой сети будет всего лишь распознавание контуров простых объектов, в данном случае букв, на однородном фоне. Это наиболее простой вид задачи, из всех, возлагаемых на нейронную сеть, и поэтому использовать в данном случае, например, самоорганизующиеся карты Кохонена, просто нецелесообразно, и более того, совсем не экономно. Так как работа и обучение сетей, имеющих сложную структуру, имеет гораздо более низкое быстродействие, чем выполнение аналогичных мероприятий на одно- двухслойных сетях, не имеющих обратных связей. Да, более многофункциональные и развитые сети могут решать задачи несравнимо более высокой степени сложности, чем однослойный перцептрон. Но при выборе сети в первую очередь следует руководствоваться классом задачи и также теми аппаратными возможностями, которыми мы обладаем для моделирования сети.

Итак, простота задачи говорит о том, что с ней вполне может справиться и однослойная НС, прямого распространения, тренируемая на основе пошагового обучения каждого нейрона в ее слое, вследствие независимости решения, принимаемого каждым нейроном в слое, от его ближайших соседей. Что освобождает нас от реализации, пусть на вид довольно простого для программирования, метода обучения на основе обратного распространения ошибки, который имеет ряд побочных эффектов, уже отмеченных ранее, среди которых можно выделить низкое быстродействие. С одной стороны, моделирование работы сети на персональном ЭВМ на базе процессора Intel Pentium 200, в некоторой степени развязывает нам руки в вопросе быстродействия работы сети. Однако, мы не можем заранее предположить реальные заводские аппаратные условия, в которых могут в дальнейшем применяться разработанные алгоритмы. Если задачей СТЗ робота будет элементарное распознавание объектов на рабочем столе, то это совершенно не обязывает нас покупать для него дорогую систему управления и распознавания. Так как в конечном счете в масштабах автоматизированного производства это приведет к огромным затратам, которые просто не смогут окупиться в короткие сроки.

Следовательно, наиболее оптимальным выбором в данном случае будет являться однослойный перцептрон, в котором количество нейронов в слое будет равным количеству объектов, которым сеть была обучена, а разрядность входного вектора сети будет равна $16 \times 16 = 256$. То есть каждый нейрон будет иметь на входе **256** весов, посредством которых, он и будет сохранять данные об объекте, которому был обучен. В качестве функции срабатывания нейрона была выбрана сигмоидальная функция, изображенная на рис. 2.10, порогом которой считается значение 0,88.

Выбрав вид сети, мы можем перейти к реализации объектной модели нейрона, а в дальнейшем и реализации модели всей нейронной сети. Основным принципом при этом будет осуществление динамичности архитектуры сети, посредством динамически изменяющегося количества нейронов в слое, в зависимости от количества объектов,

которые сеть должна распознать. Это будет реализовано с целью экономии оперативной памяти, отведенной для хранения данных сети, а также достижения компактности базы данных, в которой будут храниться основные данные об объектах, посредством хранения значений весов нейронов.

4.3.2. Создание модели нейрона

Для того, чтобы подойти конкретно к вопросу создания объектной модели нейрона, рассмотрим его основные свойства и методы с точки зрения объектно-ориентированного программирования.

Рассмотрим схему созданной модели нейрона, изображенной на рис. 4.4, и основанной на основных выводах, приведенных в п.п. 2.3.2.

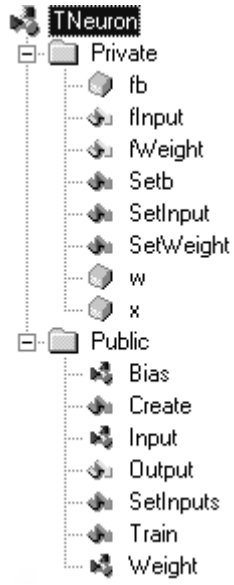


Рис. 4.4

Как известно из теории ООП каждый объект имеет частные (private) и общие (public) свойства и методы. В данном случае в качестве частных полей (то есть полей, которыми могут пользоваться только экземпляры данного объекта), объявлены 2 массива **x** и **w**, первый из которых содержит значения входов нейрона, каждый из которых равен 0 либо 1, а второй содержит значения весов, соответствующих каждому входу нейрона, изменяющихся от 0 до 1. Размерность каждого из массивов равна $16 \times 16 = 256$. Также среди частных методов стоит отметить функции чтения и установки данных полей, имеющих соответствующие названия. Также среди полей объекта «нейрон» отведено место для установки смещения нейрона. Стоит заметить, что установка частных полей и методов нужна для сохранности информации, путем ограничения доступа к ней, что является одним из базовых понятий зрения объектно-ориентированного программирования, и его преимуществом.

Теперь рассмотрим общие поля, посредством которых, в принципе, и происходит общение экземпляров нейрона с внешним миром:

- Bias – смещение нейрона;
- Input - свойство, через которое реализуется доступ к отдельному входу нейрона;
- Output – метод расчета выходного значения нейрона;
- SetInputs -метод подачи текущего входного вектора на нейрон;
- Train - метод, реализующий обучение нейрона по текущему значению его входов и весов, математических базис которого основан на выводах, приведенных в формулах 2.2 и 2.3;
- Create - метод создания объекта «нейрон», отводящий в памяти определенную область хранения его данных, а также реализующий функцию установки начального малого значения весов нейрона;
- Weight - свойство, через которое реализуется доступ к отдельному весу нейрона.

Данная модель является реализацией наиболее общих принципов функционирования элементарного нейрона и может быть применена для построения не только простых однослойных сетей, но также для сетей любой сложности. Так как нейронная сеть абсолютно любой архитектуры использует в своей основе, в качестве строительного материала одну и ту же модель нейрона, то можно считать, что данная модель является в некотором смысле универсальной.

4.3.3. Построение модели нейронной сети

Одной из наиболее сильных сторон объектно-ориентированного программирования является возможность построения некоторой архитектуры объектов, установив между ними соответствующие связи. При этом объекты, из которых производится построение более сложных архитектур, называются *базовыми*. В нашем случае базовой будет являться модель нейрона, а конечный объект, через который мы хотим реализовать взаимосвязи между нейронами в сети, и, таким образом, создать архитектуру всей сети в целом, будет являться основным во всей программной модели, назовем его TNNet.

Рассмотрим архитектуру выбранной нами сети, с целью выявления взаимосвязей между нейронами, которые мы хотим достигнуть при ее моделировании (рис. 4.5).

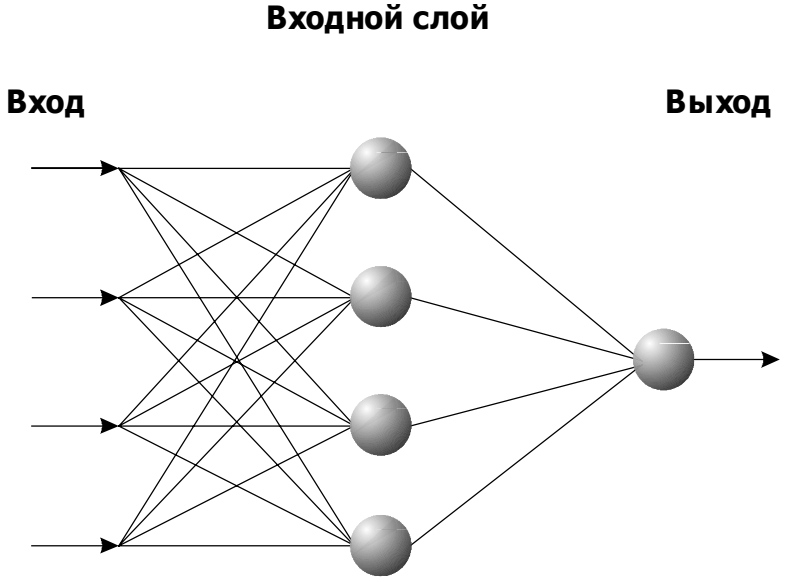


Рис. 4.5

На рис. 4.5 изображена обобщенная структура моделируемой сети. Как видно из рисунка, сеть имеет слой рецепторов, с которых информация о виде входного вектора передается на каждый нейрон входного слоя, число которых, как уже отмечалось ранее, равно числу объектов, которым была обучена сеть. Каждый нейрон имеет 256 входов и один выход, который передается на выходной слой, имеющий в своем составе один нейрон, выходное значение которого и является выходным значением всей сети. С одной стороны, надобность в выходном слое, в свете всего вышесказанного, довольно абстрактна. Так как, при подаче на вход сети объекта для распознавания, только один нейрон из всех во входном слое сети должен иметь значение, превышающее пороговую функцию срабатывания. И, с помощью простого анализа значений всех выходов нейронов, реализованного через их последовательный перебор, можно выяснить, какой именно объект находится в изображении. Однако такой способ распознавания может привести к неоднозначности принятого сетью решения, при слишком зашумленном входном изображении. Так как в этом случае не принимается во внимание ситуация наличия высокого состояния на выходе нескольких нейронов одновременно, что может привести в последствии к непредсказуемым результатам. Поэтому отказаться от наличия выходного слоя полностью мы не сможем, однако сможем ограничиться наличием в нем только одного нейрона, задачей которого и будет выборка из выходных значений нейронов во входном слое.

Теперь, руководствуясь выбранной архитектурой, создадим объектную модель нейронной сети, составив ее из базовых экземпляров объектов нейронов (рис. 4.6).

Как видно из рисунка, модель сети не имеет частных методов и полей, это объясняется тем, что так как сама сеть состоит из моделей нейронов, в которых концепции защиты информации уже были реализованы, и создавать второй уровень защищенности уже не имеет должного смысла.

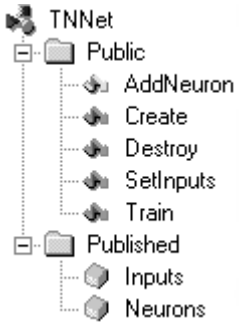


Рис. 4.6

Рассмотрим основные функции, реализованные в данной модели:

- Create – обобщенный метод создания сети, создающий заданное количество нейронов входного слоя, и проводящий их первичную инициализацию;
- Destroy - метод разрушения сети, используемый для освобождения оперативной памяти, занимаемой сетью, после завершения работы;
- AddNeuron – функция добавления очередного экземпляра объекта «нейрон» в сеть, проводящая также его первичную инициализацию;
- SetInputs – обобщенный метод подачи входного вектора на все нейроны входного слоя сети;
- Train - метод обучения сети, основанный на отдельном обучении каждого нейрона из входного слоя, посредством инициализации подобной функции нейрона, рассмотренной ранее;
- Inputs - поле хранения входного вектора, и подачи его на вход каждого нейрона входного слоя;
- Neurons - специализированный список нейронов сети, содержащий в себе указатели на каждый отдельный элемент слоя сети.

В описании модели сети ограничимся только общими определениями, и не будем вдаваться в подробное описание программных алгоритмов реализации каждой из них. Заметим только, что алгоритмы работы представленных функций основаны на выкладках, приведенных в главе 2. Основной же нашей целью в данном случае являлось стремление как можно больше упростить вид сети, с целью доказательства ее широких возможностей даже для такого, заметно упрощенного, рассмотренного случая.

Результаты применения сети для задач распознавания лингвистических символов будут приведены несколько позже, а сейчас следует перейти к не менее важному вопросу, а именно, каким образом архитектура сети и результаты ее обучения могут быть сохранены для постоянного использования. Наиболее приемлемым решением данного вопроса может стать применение наиболее передовой современной технологии хранения информации - Баз Данных.

4.4. Применение Баз Данных в распознавании изображений

Прежде чем приступить к конкретным вопросам применения Баз Данных (БД) в обработке изображений и распознавании объектов следует рассмотреть вопрос о целесообразности их использования. С одной стороны данные о структуре сети и ее характеристиках можно хранить в специализированных файлах, структура которых может быть подобрана соответствующим образом. С точки зрения программирования такой выход был бы наиболее простым. Но встает вопрос о массовости доступа к данной информации, то есть система распознавания одного робота, основанная на уже обученной сети может с успехом применяться и для других роботов, выполняющих аналогичные операции. Так как сам робот должен иметь возможность быстрого перепрограммирования и переобучения на новую задачу, то говорить о некой централизованной нейронной сети, управляющей одновременно целой ячейкой, здесь было бы неуместно. Таким образом, каждый робот, оснащенный СТЗ и системой распознавания, должен иметь свою систему управления, в состав которой должна входить нейронная сеть. Однако данные о каждой сети совершенно необязательно хранить в каждой системе управления отдельной ячейкой, тем более не стоит забывать о возможности отсутствия должных аппаратных средств, для реализации такой возможности. Следовательно, следует организовать некую общую базу, в которой хранилась бы информация о задаче, выполняемой, например, производственной ячейкой, обстоятельств, при которых должна быть выполнена задача, то есть рабочей обстановке, а также о данных локальных систем управления, к которым они должны иметь постоянный доступ. Выходом в подобной ситуации является хранение информации подобного рода в специализированных Базах Данных. Рассмотрим пример организации данных той части систем управления производством, в которой хранилась бы информация о структурах НС, применяющихся в системах распознавания роботов.

4.4.1. Выбор типа Базы Данных

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД).

Увеличение объема и структурной сложности хранимых данных, расширение круга пользователей информационных систем привели к широкому распространению наиболее удобных и сравнительно простых для понимания реляционных (табличных) СУБД. Для обеспечения одновременного доступа к данным множества пользователей, нередко расположенных достаточно далеко друг от друга и от места хранения баз данных, созданы сетевые мультипользовательские версии СУБД. В них тем или иным путем решаются специфические проблемы параллельных процессов, целостности (правильности) и безопасности данных, а также санкционирования доступа.

Ясно, что совместная работа пользователей в сетях с помощью унифицированных средств общения с базами данных возможна только при наличии стандартного языка манипулирования данными, обладающего средствами для реализации перечисленных выше возможностей. Таким языком стал SQL, разработанный в 1974 году фирмой IBM для экспериментальной реляционной СУБД System R. После появления на рынке двух пионерских СУБД этой фирмы - SQL/DS (1981 год) и DB2 (1983 год) - он приобрел статус стандарта де-факто для профессиональных реляционных СУБД. В 1987 году SQL

стал международным стандартом языка баз данных, а в 1992 году вышла вторая версия этого стандарта.

Реляционные базы данных состоят из таблиц, и только из них, которые в свою очередь состоят из строки заголовков столбцов и одной или более строк значений данных под этими заголовками. Эти столбцы и строки должны иметь следующие свойства:

- всякому столбцу таблицы присвоено имя, которое должно быть уникальным для этой таблицы;
- столбцы таблицы упорядочиваются слева направо, т.е. столбец 1, столбец 2, ..., столбец n. С математической точки зрения это утверждение некорректно, потому что в реляционной системе столбцы не упорядочены. Однако с точки зрения пользователя, порядок, в котором определены имена столбцов, становится порядком, в котором должны вводиться в них данные, если не предварять при вводе каждое значение именем соответствующего;
- строки таблицы не упорядочены (их последовательность определяется лишь последовательностью ввода в таблицу);
- в поле на пересечении строки и столбца любой таблицы всегда имеется только одно значение данных и никогда не должно быть множества значений;
- всем строкам таблицы соответствует одно и то же множество столбцов, хотя в определенных столбцах любая строка может содержать пустые значения (NULL-значения), т.е. может не иметь значений для этих столбцов;
- все строки таблицы обязательно отличаются друг от друга хотя бы единственным значением, что позволяет однозначно идентифицировать любую строку такой таблицы;
- при выполнении операций с таблицей ее строки и столбцы можно обрабатывать в любом порядке безотносительно к их информационному содержанию.

Почему же база данных, составленная из таких таблиц, называется реляционной? А потому, что отношение - relation - просто математический термин для обозначения неупорядоченной совокупности однотипных записей или таблиц определенного специфического вида.

Составим таблицы для хранения данных НС. В качестве реляционной БД для хранения полученных таблиц и организации доступа к ним была выбрана БД PARADOX 7.0, вследствие простоты и наглядности SQL-запросов и небольшого объема данных, которые должны быть в ней сохранены.

4.4.2. Хранение характеристик НС

Итак, перед непосредственным составлением таблиц, разберемся с форматом, в котором мы собмираемся сохранить данные о сети. Для того, чтобы получить исчерпывающую информацию из Базы Данных для восстановления архитектуры сети, в нее должна быть занесена информация о:

- текущем количестве нейронов во входном слое сети;
- текущих значениях весов для каждого нейрона.

В то время как заранее количество нейронов в сети, до проведения операции обучения, является числом неизвестным, то количество весов для каждого нейрона всегда равно 256. С одной стороны, для проведения операции поиска было бы удобным отвести для каждого веса нейрона свой столбец в таблице, но, однако, таким образом мы достигнем избыточночти информации и усложнения SQL-запросов, что является очень неприятным фактом. Другим вариантом решения является сохранение значений весов в

16-ти переменных типа String, в которых данные можно разделить условными знаками. В нашем случае в качестве разделителя использовалась «;».

Таким образом, конечный вариант таблицы должен иметь три столбца или поля:

1. номер нейрона в сети;
2. номер строки, содержащей веса нейрона;
3. непосредственно сама строка.

В табл. 4.1 показан пример сохранения характеристик одного нейрона сети, где первое и второе поля имеют тип Number, а третье – тип Alpha.

Табл. 4.1

Neuron	WeightStr	SameString
0.00	1.00	0.0003000000; 0.0009000000; ……..; 0.0008000000;
0.00	2.00	0.0004000000; 0.0000000000; ……..; 0.0004000000;
0.00	3.00	0.0008000000; 0.0006000000; ……..; 0.0003000000;
0.00	4.00	0.0008000000; 0.0006000000; ……..; 0.0005000000;
0.00	5.00	0.0008000000; 0.0000000000; ……..; 0.0003000000;
0.00	6.00	0.0002000000; 0.0004000000; ……..; 0.0006000000;
0.00	7.00	0.0206356368; 0.0009000000; ……..; 0.0007000000;
0.00	8.00	0.0200356368; 0.0207356368; ……..; 0.0000000000;
0.00	9.00	0.0200356368; 0.0208356368; ……..; 0.0204356368;
0.00	10.00	0.0002000000; 0.0006000000; ……..; 0.0209356368;
0.00	11.00	0.0008000000; 0.0009000000; ……..; 0.0005000000;
0.00	12.00	0.0207356368; 0.0202356368; ……..; 0.0206356368;
0.00	13.00	0.0203356368; 0.0205356368; ……..; 0.0202356368;
0.00	14.00	0.0209356368; 0.0203356368; ……..; 0.0207356368;
0.00	15.00	0.0009000000; 0.0006000000; ……..; 0.0009000000;
0.00	16.00	0.0204356368; 0.0005000000; ……..; 0.0009000000;

Напишем пример организации SQL-запроса к данной таблице. В случае, если нам нужно выделить все веса, принадлежащие какому-либо нейрону, то синтаксис запроса будет выглядеть следующим образом :

```
select WeightStr, SameString from NNetData where Neuron=:SameNeuron ,
```

где NNetData – название таблицы;

SameNeuron – номер нейрона, данные о котором мы хотим получить.

Как видно из формы запроса, при такой организации таблиц осуществление поиска нужного нейрона и данных о нем из таблицы не представляется сложной задачей. Таким же образом организуется и запись в таблицу данных, полученных в результате обучения сети:

```
insert into NNetData (Neuron, WeightStr, SameString) values (:Neur, :Weight, :Str) ,
```

где Neur – номер обученного нейрона,

Weight – номер текущей строки весов нейрона;

Str - сама строка с очередными 16-ю весами.

Учитывая равенство количества нейронов во входном слое сети количеству объектов, которым сеть была обучена можно сделать простой вывод о том, что даже при обучении сети всем буквам латинского алфавита, количество записей в базе данных не будет столь велико, а значит быстродействие сети, при такой организации хранения данных не будет заметно ухудшено.

4.4.3. Хранение характеристик эталонных изображений

С одной стороны надобность в дополнительных таблицах, в которых будут храниться эталонные изображения, представляется несколько абстрактной. Так как таблица характеристик нейронной сети уже содержит данные об эталонах в виде значений весов нейронов. Однако стоит заметить, что в результате обучения не всегда веса нейонов адекватно реагируют на эталонные изображения, что может привести к некоторым, обычно незначительным, но ощутимым ошибкам. Поэтому таблицу эталонных изображений можно использовать для проведения анализа результатов обучения сети, а также анализа результатов распознавания, для получения наиболее полной информации о протекающих в сети процессах.

Организация подобных таблиц такжде не представляется сложной в силу однообразности хранимой в них информации. Известно, что эталонные изображения используют разрешение 16×16 пикселов, причем значения пикселов в изображении равно 0 либо 1. Таким образом, применив принцип записи информации, описанный в предыдущем подпункте можно получить формат таблиц, примером которого может послужить таблица 4.2.

Табл. 4.2

Name	StrNum	String
0.00	1.00	0;0;0;0;0;1;1;1;1;1;0;0;0;0;0;0;0;
0.00	2.00	0;0;0;0;0;1;1;1;1;1;0;0;0;0;0;0;0;
0.00	3.00	0;0;0;0;0;1;1;1;1;1;0;0;0;0;0;0;0;
0.00	4.00	0;0;0;0;1;1;1;0;1;1;1;0;0;0;0;0;0;
0.00	5.00	0;0;0;0;1;1;1;0;1;1;1;0;0;0;0;0;0;
0.00	6.00	0;0;0;1;1;1;1;0;1;1;1;1;0;0;0;0;0;
0.00	7.00	0;0;0;1;1;1;0;0;0;1;1;1;0;0;0;0;0;
0.00	8.00	0;0;0;1;1;1;0;0;0;1;1;1;0;0;0;0;0;
0.00	9.00	0;0;1;1;1;0;0;0;0;0;1;1;1;0;0;0;0;
0.00	10.00	0;0;1;1;1;0;0;0;0;0;1;1;1;0;0;0;0;
0.00	11.00	0;1;1;1;1;1;1;1;1;1;1;1;1;1;0;0;0;
0.00	12.00	0;1;1;1;1;1;1;1;1;1;1;1;1;1;0;0;0;
0.00	13.00	0;1;1;1;1;1;1;1;1;1;1;1;1;1;0;0;0;
0.00	14.00	1;1;1;0;0;0;0;0;0;0;0;1;1;1;0;0;
0.00	15.00	1;1;1;0;0;0;0;0;0;0;0;1;1;1;0;0;
0.00	16.00	1;1;1;0;0;0;0;0;0;0;0;1;1;1;0;0;

Как видно из таблицы 4.2, в данный момент в Базу Данных занесены данные об эталонном изображении буквы «А» латинского алфавита.

Поиск в данной таблице можно организовать по приведенным ранее алгоритмам, выделяя нужный эталон из таблицы по его имени. В данном случае именем эталонного изображения является его порядковый номер.

4.5. Анализ результатов распознавания

Наконец, после проведенных выкладок, можно приступить к наиболее важной части всей дипломной работы, а именно выяснению вопроса – а может ли созданная нейронная сеть действиетльно распознавать? На входные изображения было наложено большое число ограничений, которые в реальных условиях могут сильно осложнить процесс настройки СТЗ. Также для того, чтобы применять сеть, нужны совсем другие аппаратные средства, напомним что процесс симуляции нейронной сети проходил на персональном компьютере класса Pentium, что может привести к повышению дороговизны СТЗ. Стоило ли действительно идти на все эти трудности? В данной главе будет дан ответ на этот вопрос.

4.5.1. Получение результатов распознавания

Итак, все подготовлено для проведения экспериментов по применению разработанной нейронной сети в вопросе распознавания изображений. Перед проведением экспериментов сеть была обучена 10-ти буквам латинского алфавита, таким образом во входном слое сети находилось 10 нейронов. Для получения изображений эталонов, имеющих шумы, использовался встроенный в программу редактор, с видом которого можно ознакомиться в приложении. Результаты распознавания будут сниматься в виде 4-х основных параметров:

1. вид подаваемого изображения;
2. сопоставленный эталонный объект;
3. анализ ошибок во входном изображении;
4. график состояний выходов нейронов.

Моделирование работы сети, как уже было отмечено раньше, производилось на персональном компьютере на базе процессора Intel Pentium 200MMX, с объемом оперативной памяти – 64 Мб. Все результаты снимались с экрана монитора, с активной страницы программы, приведенной также в приложении.

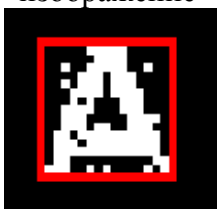
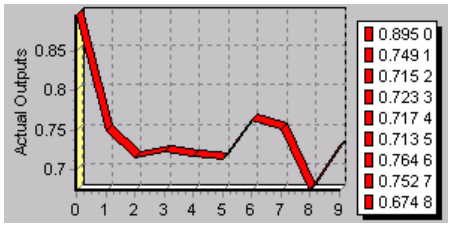
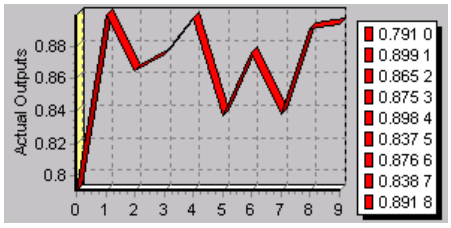
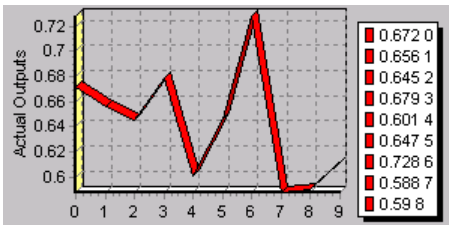
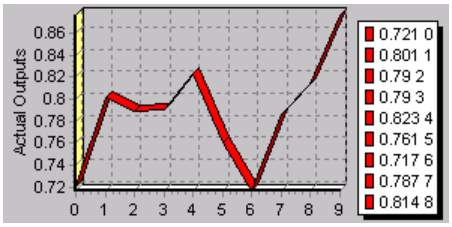
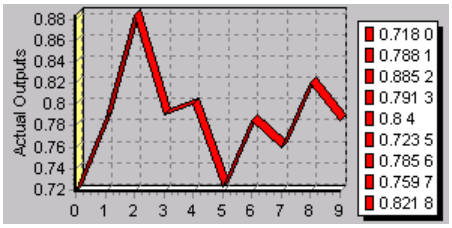
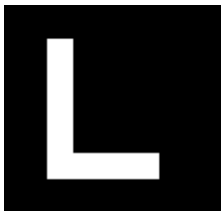
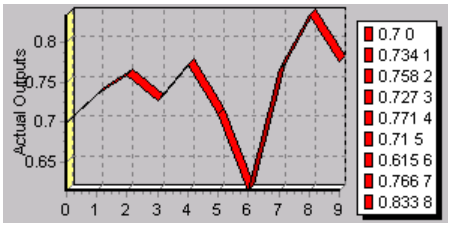
Сведем полученные результаты в таблицу 4.3.

На графиках по оси X расположены номера нейронов сети, а по оси Y их выходные значения, изменяющиеся от 0 до 1. Каждый нейрон соответствует одной из 10 занесенных в память сети букв латинского алфавита:

0. «A»
1. «B»
2. «C»
3. «D»
4. «E»
5. «H»
6. «J»
7. «K»
8. «L»
9. «P»

Количество объектов, которым сеть была обучена, намерено было выбрано большим, чем количество изображений, подаваемых на вход сети. Это было сделано для того, чтобы избежать малейших соответствий между эталонами и входными изображениями.

Табл. 4.3

Входное изображение	Распознанный эталон	Выходные значения нейронов
		
		
		
		
		
		

4.5.2. Анализ полученных результатов

Как видно из таблицы 4.3 сеть распознает заложенные в нее объекты из изображений с шумами. Причем уровень шумов, при котором сеть может распознать текущую букву варьируется, в зависимости от вида самих букв. Как например в случае распознавания буквы «L», уровень шумов в изображении может достигать 50%, а в случае распознавания буквы «C» - до 40%. Однако при высоком уровне шумов для буквы «B», как видно из соответствующего графика, сеть может спутать ее с буквами «E» и «P», соответствующим скачкам графика на 4-м и 9-м нейронах. Наиболее вероятным, судя по данным из графиков, представляется распознавание буквы «A», а наименее – буквы «B».

Таким образом можно сделать вывод, что если изображение эталонного объекта не похоже на остальные, которым сеть была обучена, то нейронная сеть может распознать его из изображения с достаточно высоким уровнем шумов, как например в случае букв «A» и «L». Данный факт не является большой проблемой в заводских условиях применения нейронных сетей, так как вариант совпадения деталей на столе сборочного робота, например, не столь реален, как в случае распознавания букв. Стоит заметить также, что разрешение изображений очень мало, то есть наличие даже одной единичной ошибки, при общем количестве пикселов равном 256, может сильно повлиять на результат распознавания, в случае использования алгоритмов помимо нейронных сетей. Это говорит о том, что разработанная сеть, при очень несложной архитектуре, обладает высокой устойчивостью в шумам. Также не стоит забывать об инвариантности результата распознавания к расположению объекта на экране.

Учитывая высокую устойчивость сети к шумам можно сделать вывод, что сеть может распознать даже объекты, в некоторой степени развернутые вокруг оси, перпендикулярной плоскости изображения. Количество объектов, которые могут быть заложены в сеть практически не регламентируется. Единственной проблемой, которая может возникнуть при этом, может стать увеличение количества знаков после запятой для выходов нейронов, которые следует при этом учитывать. При наличии 10 объектов в памяти сети, требовалось рассматривать 3 знака после запятой. Это не очень высокая точность.

Быстродействие сети при моделировании было очень высоким, например при операции обучения, скорость обработки данных зависела практически только от скорости доступа к Базе Данных, в которую записывались результаты обучения. Очень высокая скорость распознавания говорит о том, что данная сеть может с успехом моделироваться на компьютерах более низкого класса. Обьем памяти, используемый сетью при работе не превышал 2 Мб, то есть сеть вполне может быть работоспособна и на 4-х Мб оперативной памяти, что также понижает затраты на аппаратную часть СТЗ.

Из вышесказанного можно сделать вывод, что разработанная нейронная сеть действительно является работоспособной,и показатели распознавания при ее применении очень высоки. Ни один современный способ распознавания не может выдержать 50% ошибку в изображении при таком невысоком разрешении. Следовательно подобные алгоритмы не только можно, но и нужно использовать в современных системах СТЗ роботов.

4.6. Заключение

В заключении проведенной работы следует отметить новизну предпринятых решений, в силу того, что в современных робототехнических системах, или в системах исскуственного интеллекта в настоящее время еще не проводилось внедрения нейронных сетей, однако прменение НС для распознавания букв не является новой идеей. Разработка архитектуры нейронной сети и методов обучения проводилась на основе современных научных достижений в данной области, в силу того, что ученые вернулись к идее разработки нейронных сетей только в середине восьмидесятых годов, и большинство литературы по данному вопросу было выпущено за последние десять лет.

Даже такой простой вариант решения вопроса об организации сети, для решения задачи распознавания, дает столь ощутимые результаты. Что позволяет надеяться на гораздо более широкие возможности нейронных сетей, при условии дальнейшего продолжения работы в данном направлении.

Результаты проведенной работы говорят о том, что научные изыскания в области нейронных сетей в настоящее время ведутся настолько успешно, что уже в недалеком будущем возможно будет наблюдать конкретные примеры их применения в искусственном интеллекте, и распознавание изображений, хотя и является наиболее сильной стороной применения НС на практике, но является далеко не единственным видом их использования. В области ассоциативной памяти сейчас широко применяются сети Хопфилда, на основе самоорганизующихся карт Кохонена с успехом решаются задачи кластеризации и категоризации хаотично поступающей информации. Уже имеются случаи применения НС в сонарах, для выделения объектов из общего потока сигналов. К сожалению некоторый бум, начавшийся в первой половине 90-х годов в области разработки нейрокомпьютеров (как уже упоминалость ранее, это компьютеры со значительно распределенными возможностями обработки информации) в настоящее время несколько угас из-за нехватки аппаратных возможностей, но стоит надеяться, что в скором времени мы увидим новые решения в этом направлении. В области же распознавания изображений стоит отметить последние достижения, которые были представлены на последней научной конференции по оптимизации и управлению, проводившейся в 99 году в Германии (ECC’99). На которой были представлены разработки в области распознавания цветных изображений, а точнее предметов, расположенных на письменном столе. Что является задачей совершенно другого порядка сложности, чем решенная в данной работе.

Основной целью проведенной работы было не только доказательство широких возможностей математических алгоритмов, основанных на реальных природных явлениях, но также преследовалась идея нахождения способа применения НС непосредственно в роботах, а точнее в СТЗ. Поэтому параллельно с разработками основного модуля программы, содержащего в себе саму модель нейронной сети, также были включены два дополнительных модуля, с помощью которых уже можно решить одну из серьезных в настоящее время проблем – получение эталонных изображений деталей. Более подробно данные модули рассматриваются в приложении. Однако сейчас можно сделать акцент на том, что основная идея, преследуемая при создании модулей, была в нахождении способа получить эталонные изображения деталей непосредственно еще на стадии их моделирования, в современных системах автоматического проектирования. Это поможет нам избавиться от излишних ошибок, которые могут возникнуть при съемке эталонных изображений на камеру.

Список использованной литературы

1. **Описание** и распознавание объектов в системах искусственного интеллекта. // Под ред. : В.С.Гурфинкель, В.С.Файн. М.: Наука,1980.
2. **Путятин Е. П., Аверин С. И.** Обработка изображений в робототехнике. М.: Машиностроение, 1990.
3. **Васильев В. И.** Распознающие системы: Справочник. Киев: Наукова Думка, 1983.
4. **Чукин Ю. В.** Графические информационные системы// Зарубежная радиоэлектроника. 1985. № 10.
5. **Юрков Е. Ф., Нагорнов В. С.** Описание и распознавание объектов в системах искусственного интеллекта. М.: Наука, 1980.
6. **Yoh-Han Pao.** Adaptive Pattern Recognition and Neural Networks. Massachusetts: Western Reserve University, 1989.
7. **W.S. McCulloch and W. Pitts.** A logical Calculus of Ideas Immanent in Nervous Activity. Bull. Mathematical Biophysics, Vol. 5, 1943, pp. 115-133.
8. **R. Rosenblatt.** Principles of Neurodynamics. Spartan Books, New York, 1962.
9. **M. Minsky and S. Papert.** Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge, Mass., 1969.
10. **J.J. Hopfield.** Neural Networks and Physical Systems with Emergent Collective Computational Abilities. in Proc. National Academy of Sciences, USA 79, 1982, pp. 2554-2558.
11. **Anil K. Jain, Jianchang Mao, K.M. Mohiuddin.** Artificial Neural Networks: A Tutorial, Computer, Vol.29, No.3, March/1996, pp. 31-44.
12. **Копосов А.И.**, Щербаков И.Б., Кисленко Н.А., Кисленко О.П., Варивода Ю.В. и др. Отчет по научно-исследовательской работе "Создание аналитического обзора информационных источников по применению нейронных сетей для задач газовой технологии". ВНИИГАЗ, 1995.
13. **Rumelhart, D. E. and McClelland**, **J. L**., editors. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1. MIT Press, Cambridge, MA, 1986.
14. **Резник А.М.** Многорядные динамические перцептроны // Перцептрон - система распознавания образов // Под ред. А.Г.Ивахненко. - Киев: Наук. думка. - 1975. - С. 243-292.
15. **Вороновский Г. К.,** Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности // Харьков: ОСНОВА,1997.
16. **Aleksander I., Morton H.** An introduction to Neural Computing. – London: Chapman&Hall, 1990.
17. **Маункасл В.** Организующий принцип функции мозга – элементарный модуль и распределенная система // Дж. Эдельман, В.Маункасл Разумный мозг: М.:- Мир, 1981.
18. **Edelman G.M.** Molecular recognition in the immune and nervous systems. In: The Neurosciences: Path of Discovery, F.G. Worden, F.G. Swarey and G. Edelman, eds., New York, The Rockfeller University Press, 1975, pp. 65-74.
19. **S. Haykin.** Neural Networks: A Comprehensive Foundation, MacMillan College Publishing Co., New York, 1994.
20. **Holland J. H.** Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. – London: Bradford book edition, 1994 – 211 p.
21. **J. Hertz, A. Krogh**, and **R.G. Palmer**, Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, Mass., 1991.

22. **R.P. Lippman, B. Bold** and **M.L. Malpass,** A comparison of Hamming and Hopfield neural nets for pattern classification, Tech. Rep. 769, Lincoln Laboratory, MIT, 1987.
23. **S. Y. Nof**, Справочник по промышленной робототехнике, Книга 1. М: Машиностроение, 1989.
24. **Brownlee, K.A.,** Statistical Theory and Methodology in Science and Engineering, Wiley, New York, 1965.
25. **Bolles, R.C.,** Robust Feature Matching through Maximal Cliques, Imaging Applications of Automated Industrial Inspection and Assembly, Society of Photo-Optical Instrumentation Engineers, Washington, D. C., April 1979, pp. 140-149.
26. **Chen, H.H., Y.C. Lee, G.Z. Sun, H.Y. Lee, T. Maxwell, and C.L. Giles,** 1986. High order correlation model for associative memory, American Institute of Physics Conference Proceedings, № 151: Neural Networks for Computing, SnowBird, Utah, pp.398-403.
27. **Hinton, C.E., J.M. McClelland, and D.E. Rumelhart**, 1986. Distributed representations. In D.E. Rumelhart and J.L. McClellands (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1: Foundations. MIT Press, Cambridge, MA.
28. **Kloph, H.A.,** 1987. A neuronal model of classified conditioning. U.S. Air Force Wright Aeronautical Laboratories technical report, AFWAL-TR-87-1139, Dayton, OH.