электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.931

ТЕХНОЛОГИЯ РАСПОЗНАВАНИЯ ОБРАЗОВ С ИСПОЛЬЗОВАНИЕМ СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ

***Федоренко Ю.С.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Автоматизированные системы обработки информации и управления»

Научный руководитель: Гапанюк Ю.Е..,к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
bauman@bmstu.ru

1. Введение

В современном мире все чаще говорят об автоматизации тех или иных процессов в различных областях национальной экономики. На этом пути немалую роль играют интеллектуальные системы, в которых крайне важны задачи распознавания образов. К примеру, эту проблему требуется решать, когда необходимо осуществить автоматизированный анализ данных в каких-либо документах (например, в паспортах или визах при прохождении пограничного контроля), обработку банковских чеков и т.п.

Целью любого распознавания является классификация объектов по нескольким категориям или классам[1]. При этом входные символы, которые необходимо классифицировать, после сегментации могут оказаться не только в искаженном виде, но и в различных конфигурациях (сдвинутые или повёрнутые). Помимо этого нередко возникает необходимость создания алгоритма распознавания, инвариантного к масштабированию.

Решения, существующие на сегодняшний день, можно разделить на две большие группы[3]:

1. Нахождение признаков, инвариантных к имеющимся пространственным искажениям. Далее по ним производится сравнение распознаваемого образа с эталонными образцами.

2. Нормализация изображения, т.е. приведение его к такому виду, в котором находятся все образцы из обучающей выборки. Затем производится распознавание при помощи нейронных сетей. Для этого может потребоваться предварительно сдвинуть изображение, повернуть его, а также произвести масштабирование.

В данной работе кратко рассматриваются данные подходы, а затем исследуются возможности применения сверточных нейронных сетей в задачах распознавания образов.

2. Инвариантные признаки

Существует большое число признаков, инвариантных к искажениям. Ниже приведены некоторые из них.

2.1. Топологические и яркостные признаки

1. Рассмотрим площадь фигуры (количество пикселей внутри неё). Её значение инвариантно к сдвигу, но зависит от масштаба. Также возьмем другую величину — длину контура фигуры. Она тоже зависит от масштаба. Однако если обозначить через S(E) площадь фигуры, а через L(E) — длину её контура, то признак $V(E) = \frac{\sqrt{S(E)}}{L(E)}$ будет являться инвариантным не только к сдвигу, но и к масштабу[3]. Его недостаток состоит в том, что подсчёт длины контура может потребовать значительных вычислительных затрат (отметим, что вычисление площади фигуры не требует значительных затрат, т.к. по сути представляет из себя подсчет количества черных пикселей).

Отсюда стоит сделать важный вывод. Многие признаки сами по себе не являются инвариантными по отношению к повороту, масштабу и т.д., но если взять два или более таких свойств, а затем использовать их отношения в различных комбинациях, то можно получить достаточно эффективные инвариантные признаки.

Например, рассмотрим кратчайшее и наибольшее расстояния от центра масс символа до его крайних точек[2]. Отношение этих расстояний будет признаком, инвариантным по отношению к масштабу.

3. Возможно использование признаков, основанных на вычислении яркости изображения[3]. К примеру, средняя яркость изображения инвариантна по отношению к сдвигам и поворотам:

$$B_{CP} = \frac{1}{N}\sum_{i,j} B(i,j),$$

где B(i, j) — яркость пикселя в позиции (i, j); N — число пикселей.

Безусловным плюсом данного признака является относительно легкая вычислимость.

4. Также возможно использование признаков на основе моментов[2]. Для бинарного изображения момент может быть вычислен по формуле *(1)*:

$$m_{pq} = \sum_{i,j=1}^{N} j^p \cdot i^q \cdot B(i,j), \quad (1)$$

где p и q определяют порядок момента, а *B(i, j)* — значение яркости пикселя (для бинарного изображения это 0 или 1). Далее на основании *(1)* можно вычислить центральные моменты *(3)*:

$$\mu_{pq} = \sum_{i,j=1}^{N} (j - \frac{m_{10}}{m_{00}})^p \cdot (i - \frac{m_{01}}{m_{00}})^q \cdot B(i, j).$$

Они инвариантны к смещениям.

Также возможен расчет моментов более высокого порядка, которые не зависят также от поворотов и масштабирования.

Существуют и другие инвариантные характеристики[3]. Однако большинство из них являются топологическими признаками, и их вычисление требует значительных затрат. Поэтому они представляют интерес лишь с теоретической точки зрения.

2.2. Контурный анализ

На основе данного метода вполне возможно создание системы распознавания образов. Способ достаточно эффективен, и в данной статье он рассматривается прежде всего как часть системы распознавания образов для улучшения качества распознавания.

Контурный анализ позволяет описывать, хранить, сравнивать и производить поиск объектов, представленных в виде своих внешних очертаний — контуров [4]. Также он позволяет эффективно решать основные проблемы распознавания образов — перенос, поворот и изменение масштаба изображения объекта, поскольку методы контурного анализа инвариантны к этим преобразованиям.

Предполагается, что контур содержит всю необходимую информацию о форме объекта. Внутренние точки объекта во внимание не принимаются. Это ограничивает область применимости алгоритмов контурного анализа, но рассмотрение только контуров позволяет перейти от двумерного пространства изображения — к пространству контуров, и тем самым снизить вычислительную и алгоритмическую сложность.

Контур кодируется последовательностью, состоящей из комплексных чисел. Первоначально на контуре фиксируется некоторая точка. Затем контур обходится (допустим – по часовой стрелке), и каждый вектор смещения записывается комплексным числом $a+ib$, где a — смещение точки по оси X, а b — смещение по оси Y. Смещение берется относительно предыдущей точки [4]. Полученная последовательность комплексных чисел называется вектор-контуром. Стоит отметить несколько очевидных свойств вектор-контуров:

1. Сумма всех элементарных векторов вектор-контура равна 0. Это вытекает из замкнутости контура.

2. Вектор-контур не зависит от параллельного переноса исходного изображения. Поскольку контур кодируется относительно начальной точки, то этот способ кодирования инвариантен сдвигу исходного контура.

3. Поворот изображения на определенный угол равносилен повороту каждого элементарного вектора контура на тот же угол.

4. Изменение масштаба исходного изображения можно рассматривать как умножение каждого элементарного вектора контура на масштабный коэффициент.

Далее рассматривается скалярное произведение контуров:

$$\eta = (\Gamma, N) = \sum_{n=0}^{k-1} (\gamma_n, v_n),$$

где k — размерность вектор-контура, γ_n — *n*-ый элементарный вектор контура Г, υ_n — n-ый элементарный вектор контура N, (γ_n, υ_n) — скалярное произведение комплексных чисел, вычисляемого следующим образом:

$$(a+ib, c+id) = (a+ib, c-id) = ac+bd+i(bc-ad)$$

Из линейной алгебры известно, что скалярное произведение ортогональных векторов равно 0, а для коллинеарных векторов оно максимально по модулю. Скалярное произведение контуров обладает этими же свойствами [4]. Итак, чем больше скалярное произведение между векторами, тем меньше угол между ними и, следовательно, тем они «более похожи» друг на друга. Это и есть то свойство, которое позволяет применять методы контурного анализа в задачах распознавания образов.

Данный метод имеет некоторые недостатки [4]. Часть из них связана со сложностью выделения контура в некоторых изображениях. Кроме этого, данный метод не допускает пересечения объектов и искажения их формы, также он неустойчив к помехам. Зато бесспорным преимуществом контурного анализа является простота вычислений и, как следствие, быстродействие.

3. Нормализация изображений

Несмотря на наличие немалого числа признаков, инвариантных к тем или иным пространственным искажениям, на практике очень часто их оказывается недостаточно для осуществления надежного распознавания[3]. Поэтому нередко приходится хотя бы частично устранять пространственные искажения путём различных преобразований.

3.1. Центрирование изображения

Центрирование изображения позволяет устранить ряд часто встречающихся искажений, связанных со сдвигом положения образа относительно центра. Для этого

необходимо найти центр масс символа, а затем сдвинуть его так, чтобы он совпал с центром масс символов из обучающей выборки. Рассмотрим данную процедуру подробнее:

1. Нахождение центра масс. Для любой системы его можно найти, взяв произвольную точку системы и рассчитав нормированную векторную сумму от нее до всех остальных точек. Согласно теореме о группировке масс, после сдвига этой точки на полученный вектор, она оказывается прямо в центре масс системы. Основываясь на данном соображении, можно достаточно легко рассчитать центр масс.

Для удобства рассматривается точка с координатами (0; 0) — левый верхний угол. После расчета взвешенной суммы масс получатся координаты центра масс (в силу того, что исходные координаты точки равны нулям).

2. Далее рассчитываются координаты вектора, соединяющего центр масс с серединой изображения. Затем все точки изображения сдвигаются на данный вектор.

Таким образом, в результате достаточно простой процедуры (в плане вычислений), можно устранить практически любые искажения, связанные со сдвигами.

3.2. Масштабирование и поворот изображения

На практике масштабирование изображения возможно осуществить при помощи библиотеки *OpenCV (Open Source Computer Vision Library)* — библиотека компьютерного зрения с открытым исходным кодом. Её функции позволяют масштабировать исходное изображение в нужное число раз.

Устранение искажений, связанных с поворотом, также реализуется при помощи библиотеки OpenCV. Вначале находятся хотя бы две точки прямой, про которую точно известно, что она должна быть горизонтальной. Далее изображение поворачивается так, чтобы эта прямая стала горизонтальной.

После нормализации изображения необходимо произвести его распознавание при помощи нейронной сети. Это обсуждается в следующем разделе.

4. Сверточная нейронная сеть

4.1. Почему сверточная сеть?

Одной из наиболее распространенных нейросетевых моделей является многослойный персептрон [5]. Однако он не слишком хорошо подходит для решения реальных задач распознавания образов. Большая размерность входных изображений приводит к резкому увеличению числа нейронов и синапсических связей такой сети. В результате сильно увеличивается время и вычислительная сложность процесса обучения, а в ряде случаев добиться сходимости такой сети вообще не удается. Кроме того, обычный

персептрон игнорирует топологию входных данных, не учитывая четкую двумерную структуру изображений. Использование сверточной сети позволяет устранить данные недостатки.

Она была изобретена относительно недавно американским профессором Ян ЛеКуном, вдохновленным исследованиями зрительной коры головного мозга кошки. Она представляет собой особый класс многослойных персептронов, специально созданных для распознавания двумерных поверхностей с высокой степенью инвариантности к масштабированию, смещению, поворотам и прочим искажениям. Недавно сверточные сети были успешно применены для распознавания снимков трехмерных объектов[9] и номеров домов на улице[12]. Топология классической нейронной [6] сети приведена на рис. 1.

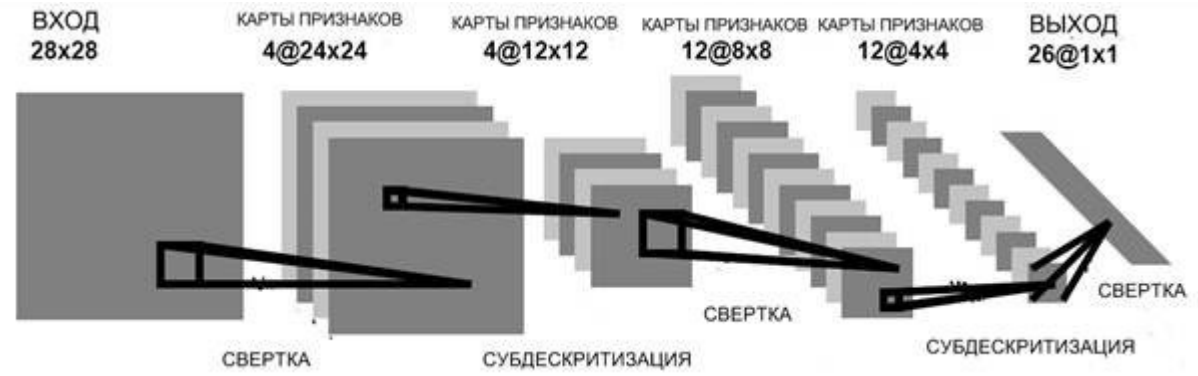

Рис. 1. Топология классической сверточной нейронной сети

Архитектура данной сети реализует следующие аспекты [6]:

1. Локальное восприятие. Оно подразумевает, что на вход одного нейрона подается не все изображение (или выходы предыдущего слоя), а лишь некоторая его область. Такой подход позволил сохранить топологию изображения от слоя к слою при значительном сокращении объема вычислений. В результате каждый нейрон следующего слоя получает информацию не от всего предыдущего слоя, а только от его части, которая описывается локальным рецептивным полем (обычно его выбирают размером 5x5). Благодаря сканированию целой области, а не отдельных точек, данный подход позволяет учесть «тонкие», «глубинные» свойства изображения, что позволяет значительно увеличить качество распознавания.

2. Разделяемые веса. Данная концепция позволяет для большого количества связей использовать относительно малое число параметров — весовых коэффициентов. Достигается это засчет того, что все нейроны одной карты имеют одинаковые весовые коэффициенты. В результате процесс обучения сети проходит намного быстрее и точнее, чем у обычного персептрона. При этом от такого допущения точность распознавания не уменьшается.

3. Субдискретизация. Её суть заключается в уменьшении пространственной размерности изображения. Она выражается в том, что существуют слои с одноименным названием, которые не выделяют новые черты изображения, но уменьшают его размер, не теряя уже выделенные признаки. Субдискретизация позволяет добиться частичной инвариантности к масштабу.

4.2. Структура используемой сети.

Архитектура сверточной сети, представленная на рисунке 1, может быть упрощена практически без потери качества распознавания [8]. Сокращение числа слоев достигается благодаря объединению сверточных и субдискретизирующих слоев в один. Уменьшение размерности изображения происходит засчет смещения рецептивных полей сверточных нейронов на два пикселя. При этом ширина и высота рецептивного поля должны быть нечетными числами, чтобы обеспечить попадание нейрона слоя свертки в центр поля. Выбранный размер рецептивного поля 5x5 удовлетворяет описанным выше условиям, а кроме того обеспечивает достаточное наложение рецептивных полей друг на друга. В результате каждый слой свертки уменьшает размер карты признаков с размера n до (n-3)/2. В результате структура используемой нейронной сети имеет следующий вид (рис. 2):

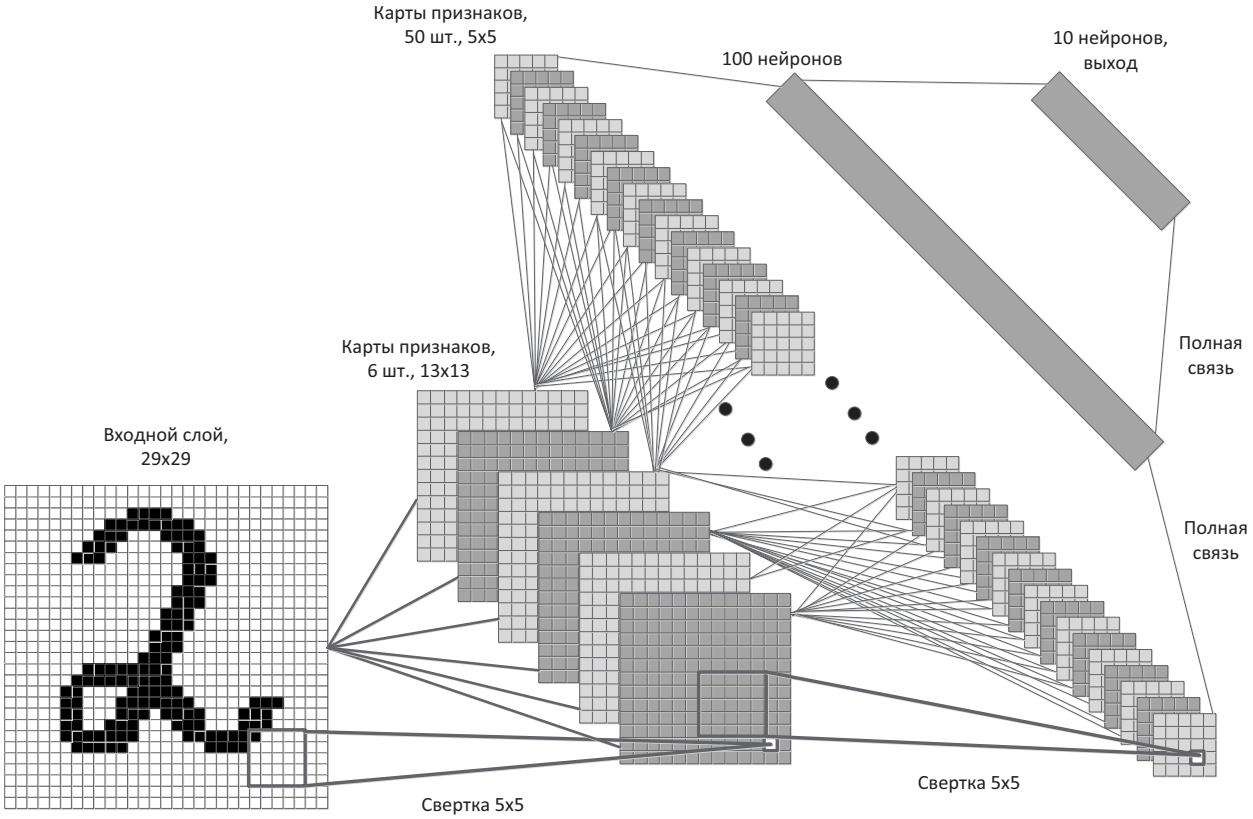

Рис. 2. Структура исследуемой нейронной сети: входной слой, слои свертки и слои классификации

Входными данными нейронной сети является изображение размером 29×29 пикселей. Такой размер выбран для того, чтобы число нейронов в промежуточных слоях было целым числом. Таким образом, на входном слое сети находится 841 нейрон.

Первый скрытый слой является слоем свертки. Он содержит 6 карт признаков размером 13×13 ((29 – 3)/2 = 13). Меньшее число карт признаков приводит к тому, что сеть работает менее точно, а большее число карт признаков значительно не улучшают свойств сети[8]. Каждый элемент карты признаков соединен с рецептивным полем 5×5 на входном изображении (рис. 2). При этом сдвиг локального рецептивного поля производится не на одну позицию, как в классических работах Яна ЛеКуна [6,7,9], а на две, что позволяет уменьшать пространственную сложность изображения, не прибегая к слоям субдискретизации.

Второй слой имеет $6 \times 13 \times 13 = 1014$ нейронов, при этом каждый нейрон имеет 25 весовых коэффициентов (поскольку он связан с локальной рецептивной областью размера 5×5) и одно значение смещения. Весовые коэффициенты и значения смещения одинаковы для всех нейронов карты. Именно это свойство позволяет во много раз сократить количество обучаемых параметров сети и делает возможным добиться сходимости сети при обучении, что практически невозможно при использовании обычных многослойных персептронов. В результате второй слой имеет $26 \times 6 = 156$ обучаемых параметров. Интересной особенностью слоев свертки является тот факт, что при сдвиге входного изображения значения карт признаков будут сдвинуты на ту же самую величину. Благодаря этому сверточные сети обладают инвариантностью ко сдвигам и незначительным искажениям входного сигнала. Таким образом, в первом скрытом слое находится $13 \times 13 \times 6 \times 26 = 26634$ синапсических связей и всего лишь 156 обучаемых параметров!

Второй слой свертки содержит 50 карт признаков размером 5×5 . Принцип его построения аналогичен предыдущему слою. Нетрудно вычислить, что в этом слое находится $50 \times 5 \times 5 = 1250$ нейронов. Каждый элемент в карте признаков связан с шестью областями размеров 5x5 шести карт предыдущего слоя. В результате получаем $50 \times 6 \times 25 + 50 = 7550$ обучаемых параметров и $7550 \times 25 = 188750$ связей.

Два первых сверточных слоя можно рассматривать как слои извлечения признаков из изображения. Следующие два слоя являются слоями классификации (рис. 2). В них каждый нейрон соединен со всеми нейронами предыдущего слоя. Третий скрытый слой состоит из 100 нейронов, каждый из которых связан со всеми картами предыдущего слоя. В результате получаем $100 \times 50 \times 25 + 100 = 125100$ синапсических связей и столько же обучаемых параметров.

Четвертый выходной слой содержит 10 нейронов (каждый из которых соответствует своей цифре). В результате получаем $100 \times 10 + 10 = 1010$ обучаемых параметров. В результате сеть имеет 3215 нейронов, 133816 обучаемых параметров и 341224 синапсических связи.

4.3. Работа сети

Принцип работы данной сети в целом аналогичен принципу работы многослойного персептрона. На вход каждого нейрона поступает взвешенная сумма — скалярное произведение между входным вектором сигналов и вектором весов, которая затем подается в качестве аргумента функции активации:

$$y_n^i = \sum_{k=0}^{k=C_{n-1}} \omega_n^{ik} \cdot x_{n-1}^k$$

$$x_n^i = F(y_n^i),$$

где x_n^i — выход i-го нейрона n-го слоя, y_n^i — скалярное произведение между входным вектором сигналов и вектором весов, F — функция активации, ω_n^{ij} — весовой коэффициент между j-ым нейроном (n–1)-го слоя и i-ым нейроном n-го слоя. Для удобства записи формул, а также для удобства реализации программы смещение каждого нейрона принимается равным 1, а весовой коэффициент обозначается через ω_n^{i0} (рис. 3):

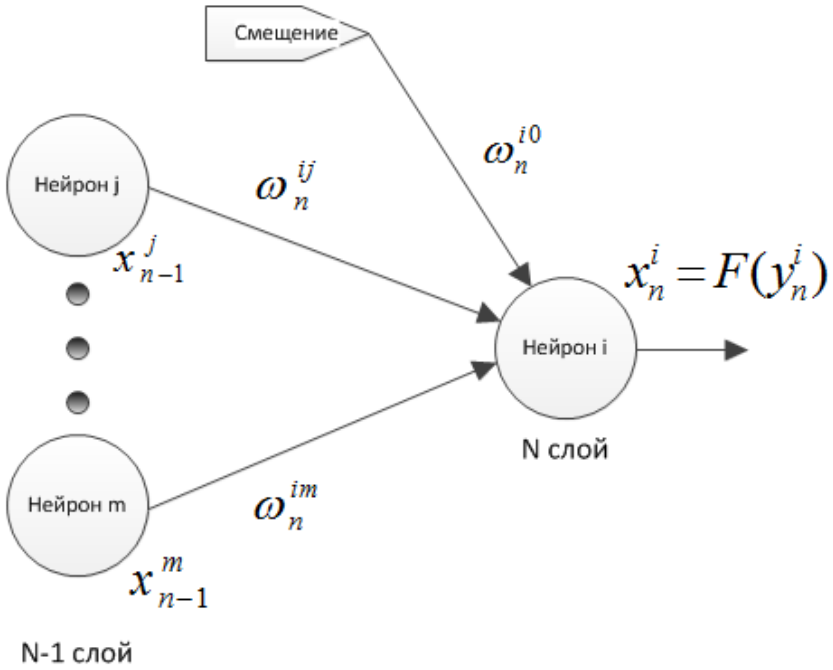

Рис. 3. Принцип работы нейронной сети: небольшой фрагмент, иллюстрирующий принцип работы сети

В качестве функции активации для скрытых слоев сети был выбран гиперболический тангенс:

$$x = f(y) = A \cdot \tanh(S \cdot y),$$

где y — скалярное произведение на входе нейрона, *x = f(y)* — значение возбуждения на его выходе, *tanh* — гиперболический тангенс, который равен:

$$x = \tanh(y) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

График функции изображен на рис. 4:

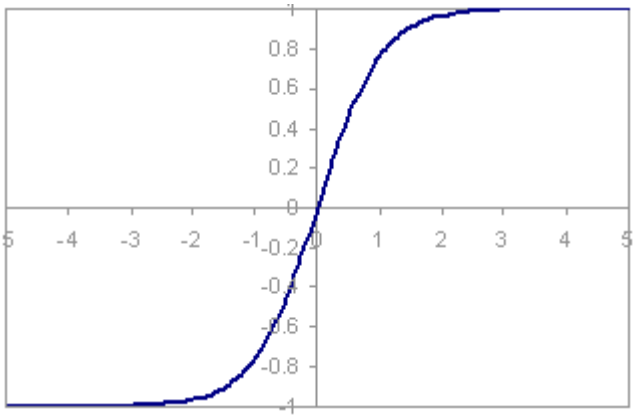

Рис. 4. График функции y = tanh (x)

A и S — числовые коэффициенты. Эмпирическим путем [6] установлено, что наиболее оптимально взять A = 1.7159, S=2/3. Выбор гиперболического тангенса в качестве функции активации не случаен, поскольку он имеет ряд удобных свойств:

1. Симметричность функции: $f(1) = 1, f(-1) = -1$;

2. В начале координат тангенс угла наклона данной функции близок к 1;

3. Ожидаемый отклик сети смещен от границы области значений функции активации в сторону её внутренней части. В этом случае при модификации параметров сети в процессе обучения удается избежать резких колебаний их значений, что позволяет ускорить обучение;

4. Данная функция имеет простую производную [11], что очень удобно при реализации обратного распространения:

$$\frac{dF}{dy} = \frac{d}{dy}\left(\frac{\sinh(y)}{\cosh(y)}\right) = \frac{\cosh^2 y - \sinh^2 y}{\cosh^2 y} = 1 - \tanh^2 y.$$

Т.е. если $x = \tanh(y)$, то $\frac{dF}{dy} = 1 - x^2$.

Для нейронов выходного слоя также удобно применять гиперболический тангенс.

4.4. Обучение сети

Обучение нейронной сети обычно является сложной и вычислительно трудоемкой задачей. Более того, при обучении многослойных персептронов очень часто бывает невозможно добиться сходимости. Сверточная нейронная сеть свободна от этих проблем, поскольку там меньше обучаемых параметров. Однако сама задача обучения всё равно остается достаточно трудоемкой.

При обучении сети используется метод обратного распространения ошибки, который идейно не отличается от аналогичного алгоритма для многослойного персептрона [8], но учитывает особенности архитектуры сверточной сети. При реализации обратного распространения ошибка выходного слоя может быть выбрана следующим образом:

$$E_n^p = \frac{1}{2} \cdot \sum_{k=1}^{M} (x_k - d_k)^2 , \quad (2)$$

где M — количество нейронов выходного слоя, k — номер выходного нейрона, x_k — реальное значение выходного сигнала нейрона, d_k - ожидаемое значение.

Возможен выбор и других функций для ошибки [6, 8], но в данной работе была применена именно ошибка по среднеквадратичному отклонению ввиду его универсальности. Для обучения сети был выбран стандартный алгоритм градиентного спуска[6]. Коррекция весов осуществляется по формуле:

$$(\omega_n^{ij})_{new} = (\omega_n^{ij})_{old} - \eta \cdot (\frac{\partial E_n^p}{\partial \omega_n^{ij}}) , \quad (3)$$

где $(\omega_n^{ij})_{new}$ — значение весов после коррекции, $(\omega_n^{ij})_{old}$ — значение весов до коррекции, η характеризует скорость обучения (изначально обычно выбирается равным 0.0005, затем постепенно увеличивается в процессе обучения). $\frac{\partial E_n^p}{\partial \omega_n^{ij}}$ вычисляется по следующей формуле:

$$\frac{\partial E_n^p}{\partial \omega_n^{ij}} = x_{n-1}^j \cdot \frac{\partial E_n^p}{\partial y_n^i} ,$$

где x_{n-1}^j — выход j-го нейрона (n-1)-го слоя, y_n^i — скалярное произведение всех выходов нейронов (n–1)-го слоя и соответствующих весовых коэффициентов.

$$\frac{\partial E_n^p}{\partial y_n^i} = G(x_n^i) \cdot \frac{\partial E_n^p}{\partial x_n^i} ,$$

где $G(x_n^i)$ — производная функции активации, а

$$\frac{\partial E_n^p}{\partial x_n^i} = x_n^i - d_n^i,$$

что следует из формулы *(2)*.

Ошибку необходимо распространить на предыдущий слой. Это делается по следующей формуле:

$$\frac{\partial E_{n-1}^p}{\partial x_{n-1}^k} = \sum_i \omega_n^{ik} \cdot \frac{\partial E_n^p}{\partial y_n^i}$$

При реализации процесса обучения для сверточных слоев сети необходимо использовать вышеприведенные формулы в матричном виде.

Значения исходных синапсических весов выбираются на основе равномерного распределения с нулевым математическим ожиданием и дисперсией [6], обратной квадратному корню из числа синапсических связей нейрона. Однако также возможно проводить начальную инициализацию весов на основе дисперсии, равной 0.05 [11] (значение выбрано эмпирически).

Для удобства и ускорения обучения сети входные значения пикселей изображения нормируются по формуле:

$$y_i = \frac{x_i}{128} - 1,$$

где x_i -значение пикселя исходного изображения, y_i - значение, подаваемое на вход сети.

Кроме того, для ускорения обучения сети можно пропускать те образцы, для которых ошибка уже мала на данном этапе обучения. Это позволяет также ускорить процесс в несколько раз.

4.5. Обучающая выборка

Для обучения сети была использована база рукописных символов MNIST, которая содержит 60000 обучающих цифр, написанных с различными искажениями, и 10000 тестовых образцов. Данная база доступна на сайте Яна Лекуна[10] в виде бинарного файла специального формата, который также подробно описан в [10].

4.6. Методы второго порядка

Описанная выше методика обучения позволяет добиться сходимости сети, но является достаточно медленной. Так, для обучения на базе MNIST требуется, как минимум, сотня эпох, а то и больше в несколько раз больше[11]. С учетом того, что каждая эпоха даже при хорошей реализации занимает около получаса, время обучения сети может составить около недели непрерывной работы мощного компьютера! Поэтому

является очень актуальным использование различных методов второго порядка, которые позволяют ускорить обучение сети в десятки раз[7].

Прежде всего, следует отказаться от идеи пакетного обучения сети, когда все образцы из обучающей выборки постоянно подаются на вход сети. Гораздо быстрее работают стохастические методы обучения, при которых на данном этапе из обучающей выборки по определенным критериям выбирается лишь часть образов для обучения. Эффективный метод описан в работах Яна ЛеКуна[7, 13] под названием стохастический диагональный метод Левенберга-Марквардта.

Данный алгоритм подробно описан в [7]. Его суть сводится примерно к следующему: перед каждой эпохой из базы выбирается случайным образом 500 изображений. Для каждого из них вычисляется Гессиан, потом они суммируются, и вычисляется среднее значение для всех 500 образцов. Далее производится вычисление коэффициента η, характеризующего скорость обучения:

$$\eta_{ki} = \frac{\varepsilon}{\frac{\partial^2 E}{\partial \omega_{ki}^2} + \mu}, \quad (4)$$

где ε характеризует скорость обучения, μ — коэффициент, препятствующий чрезмерному возрастанию скорости обучения, $\frac{\partial^2 E}{\partial \omega_{ki}^2}$ — гессианы, которые вычисляются отдельно для каждого весового коэффициента.

Сами гессианы также вычисляются путем обратного распространения по следующим формулам:

$$\frac{\partial^2 E}{\partial y_k^2} = \frac{\partial^2 E}{\partial x_k^2} \cdot (F'(y_k)^2),$$

$$\frac{\partial^2 E}{\partial \omega_{ki}^2} = \frac{\partial^2 E}{\partial y_k^2} \cdot (x_{n-1}^i)^2,$$

$$\frac{\partial^2 E}{\partial (x_{n-1}^i)^2} = \sum_k \frac{\partial^2 E}{\partial y_k^2} \cdot \omega_{ki}^2.$$

Эти вычисления проводятся для случайно выбранных 500 образцов перед каждой эпохой.

4.7. Тестирование сети и анализ результатов

Уже при первом эксперименте были достигнуты неплохие результаты, которые представлены в таблице 1.

Таблица 1

Результаты распознавания

Точность распознавания обучающей выборки (60000 цифр), %	Точность распознавания тестовой выборки (10000 цифр), %
99.5 %	98.81 %

При этом часть изображений, ошибочно распознанные программой, могут вызывать сомнения даже у человека. Примеры таких изображений представлены на рис. 5:

Рис.5. Примеры образов, неправильно распознанных программой: под каждой цифрой слева представлен правильный ответ, справа — ответ, выданный нейросетью

На первых эпохах обучения значения функции ошибки сильно падает, затем оно снижается уже менее значительно. Для сравнения в таблице 2 приведены результаты работы других классификаторов, протестированных на базе MNIST [6]:

Таблица 2

Процент неверно распознанных символов на тестовой выборке других систем классификации

Классификатор	Количество ошибок на тестовой выборке (%)
Linear	12
[deslant] Linear	8.4
Pairwise	7.6

K-NN Euclidean	5
[deslant] K-NN Euclidean	2.4
40 PCA + quadratic	3.3
40 PCA + quadratic	3.6
[16x16] Tangent Distance	1.1
SVM poly 4	1
RS-SVM poly 5	1
[dist] V-SVM poly 9	0.8
28x28-300-10	4.6
[dist] 28x28-300-10	3.7
[deslant] 20x20-300-10	1.6
28x28-1000-10	4.5
[dist] 28x28-1000-10	3.8
28x28-300-100-10	3.05
[dist] 28x28-300-100-10	2.5
28x28-500-150-10	2.95
[dist] 28x28-500-150-10	2.45
[16x16] LeNet-1	1.7
LeNet-4	1.1
LeNet-4/Local	1.1
LeNet-4/K-NN	1.1
LeNet-5	0.95
[dist] LeNet-5	0.8
[dist] Boosted LeNet-4	0.7

Основываясь на данной таблице, работу системы можно считать высокоэффективной, особенно если учесть, что сети LeNet имеют намного более сложную архитектуру, чем нейронная сеть, описанная выше.

5. Заключение

Для исследования возможностей сверточных нейронных сетей в распознавании рукописных цифр была разработана автоматизированная система, реализующая модифицированную архитектуру четырехслойной сверточной сети без слоев субдискретизации. Была подробно исследована методика обучения сети и методы второго порядка для её оптимизации. Обучение и тестирование сети производилось на основе базы рукописных цифр MNIST. Несмотря на упрощение структуры сети, были показаны достаточно неплохие результаты, вполне сопоставимые с результатами лучших систем по распознаванию цифр.

Список литературы

1. Местецкий Л.М. Математические методы распознавания образов. Курс лекций, ВМиК МГУ. 2002 – 2004. – 85с.

2. Путятин Е.П. Методы и системы распознавания образов. Курс лекций. 2005-2008. – 22с.
3. Ерош И.Л., Сергеев М.Б., Соловьев Н.В. Обработка и распознавание изображений в системах превентивной безопасности: Учебное пособие (Часть 2). 2006. – 47с.
4. Введение в контурный анализ. Под ред. Фурмана Я.А. Москва, Физматлит. 2003. – 590с.
5. Ф. Уоссермен. Нейрокомпьютерная техника: теория и практика. – Пер. с англ., 1992. - 118 с.
6. Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998. [46 pages]
7. Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient BackProp," in Neural Networks: Tricks of the trade, (G. Orr and Muller K., eds.), 1998. [44 pages]
8. Simard, P.Y. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis / P.Y. Simard, D. Steinkraus, J. Platt // International Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society. – Los Alamitos. – 2003. – P. 958-962.
9. LeCun, Y. Scaling learning algorithms towards AI / Y. LeCun, Y. Bengio – MIT Press, 2007.
10. LeCun, Y. The MNIST database of handwritten digits — http://yann.lecun.com/exdb/mnist.
11. Mike O'Neill. Neural Network for Recognition of Handwritten Digits — http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi. 2006.
12. Pierre Sermanet, Soumith Chintala and Yann LeCun: Convolutional Neural Networks Applied to House Numbers Digit Classification, International Conference on Pattern Recognition (ICPR 2012), 2012.
13. Koray Kavukcuoglu, Pierre Sermanet, Y-Lan Boureau, Karol Gregor, Michaël Mathieu and Yann LeCun: Learning Convolutional Feature Hierachies for Visual Recognition, Advances in Neural Information Processing Systems (NIPS 2010), 23, 2010.