САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Нейросетевые технологии распознавания пиксельных изображений

Дипломная работа студента 545 группы

Мальчевского Михаила Андреевича

Научный руководитель ……………… д.т.н., проф. Тимофеев А.В.

/ подпись /

Рецензент ……………… д.ф.-м.н., проф. Косовская Т.М.

/ подпись /

“Допустить к защите” ……………… д.ф.-м.н., проф. Терехов А.Н.

заведующий кафедрой, / подпись /

Санкт-Петербург

2012

SAINT PETERSBURG STATE UNIVERSITY

Mathematics & Mechanics Faculty

Software Engineering Chair

Neural network technologies of pixel images recognition

by

Mikhail, Malchevskiy

Master’s thesis

Supervisor ……………… Professor A. V. Timofeev

Reviewer ……………… Professor T. M. Kosovskaya

“Approved by” ……………… Professor A. N. Terekhov

Head of Department

Saint Petersburg

2012

Оглавление

[Введение 4](#_Toc326658885)

[Постановка задачи 6](#_Toc326658886)

[Обзор существующих решений 7](#_Toc326658887)

[Simbrain 7](#_Toc326658888)

[Neuroph 8](#_Toc326658889)

[Nengo 9](#_Toc326658890)

[Encog 10](#_Toc326658891)

[Существующие реализации сверточных сетей 10](#_Toc326658892)

[Выводы из обзора 10](#_Toc326658893)

[Описание решения 12](#_Toc326658894)

[Варианты слоев нейронной сети 12](#_Toc326658895)

[Полносвязный слой 12](#_Toc326658896)

[Слой свертки 13](#_Toc326658897)

[Слой субдискретизации 14](#_Toc326658898)

[Методы оптимизации работы нейронной сети 16](#_Toc326658899)

[Стохастический и пакетный режимы обучения 16](#_Toc326658900)

[Перемешивание обучающих примеров 17](#_Toc326658901)

[Нормализация входных данных 18](#_Toc326658902)

[Сигмоидальная функция 19](#_Toc326658903)

[Выбор скорости обучения 20](#_Toc326658904)

[Тестирование системы 22](#_Toc326658905)

[Описание базы MNIST 22](#_Toc326658906)

[Предобработка изображений 22](#_Toc326658907)

[Описание решения и топология сети 24](#_Toc326658908)

[Заключение 27](#_Toc326658909)

[Продолжение исследования 27](#_Toc326658910)

[Список литературы 28](#_Toc326658911)

# Введение

Автоматическое (машинное) распознавание, описание, классификация и группирование образов – важные задачи в большом количестве инженерных и научных областей, таких как биология, физиология, медицина, маркетинг, компьютерное зрение, искусственный интеллект. Введем понятие образа. Образ – противоположность хаоса; это определенная сущность, которой может быть дано имя. Существуют два класса задач распознавания/классификации: 1) распознавание с учителем, в котором входной образ считается членом определенного класса; 2) распознавание без учителя (например, классификация), в котором образ считается членом неизвестного класса. Нужно заметить, что задача распознавания считается задачей классификации, в которой классы либо задаются дизайнером системы(в распознавании с учителем), либо строятся на основании схожести образов (в распознавании без учителя).

Одной из наиболее трудных задач распознавания образов является задача распознавания (классификации) изображений. Эта задача возникает в таких областях как распознавание рукописного текста, дорожных знаков, номеров автомобилей, стерео- и мультизрение. Отличительной особенностью данной задачи является огромная размерность входного пространства – что ведет к усложнению распознавателей и вычислительным трудностям. Многие подходы к распознаванию изображений пропускают входные данные через фильтр, проецирующий входной вектор на пространство существенно меньшей размерности, после чего разбивают промежуточные вектора на классы с помощью стандартных распознавателей.

Одним из таких классификаторов являются нейронные сети, единичный элемент которых, нейрон, эмулирует работу биологического нейрона. Классические нейронные сети являются решением для задачи распознавания с учителем. Несмотря на сложность извлечения знаний из нейросетевой системы, они успешно применяются для многочисленных задач классификации, управления, прогнозирования.

Современным подходом к распознаванию изображений являются сверточные нейронные сети. Они имеют большое количество слоев, по сравнению с классическим многослойным персептроном. За счет совместных весов, используемых сразу несколькими нейронами в каждом слое, удается снизить общее количество обучаемых параметров сети и ускорить обучение. Также, в отличие от многослойного персептрона, сверточные сети восприимчивы к топологии входного изображения.

Данная работа посвящена реализации системы, позволящей строить нейросетевые распознаватели для различных задач распознавания изображений. Нейронные сети, используемые в работе, базируются на слоях свертки и субдискретизации. Проведен обзор алгоритмов, ускоряющих обучение нейронных сетей, и алгоритмов, повышающих качество распознавания.

# Постановка задачи

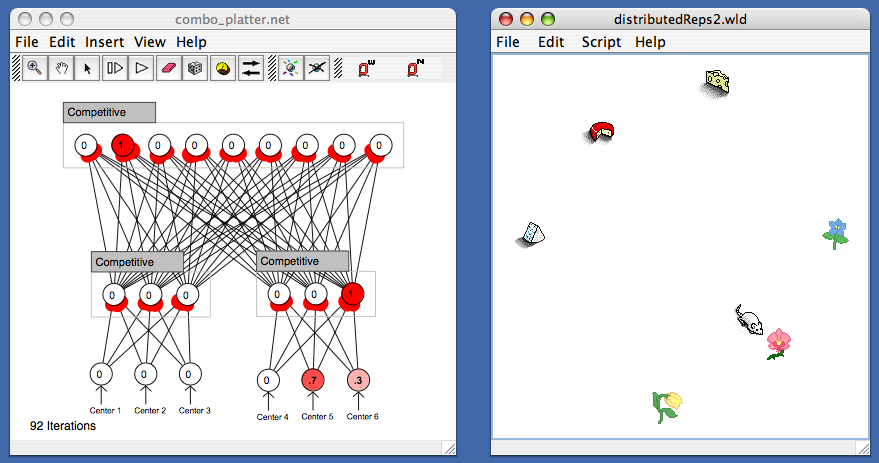
В рамках проводимой работы было необходимо:

* Исследовать существующие подходы к распознаванию изображений
* Изучить существующие нейросетевые технологии
* Реализовать систему для построения нейросетевых распознавателей
* Изучить и внедрить в систему алгоритмы ускорения обучения и повышения качества распознавания
* Опробовать построенный с помощью системы распознаватель на реальных данных

# Обзор существующих решений

На данный момент существует большое количество алгоритмов распознавания, систем для построения нейросетевых (и не только) распознавателей.

## Simbrain

Simbrain – кроссплатформенная система, написанная на языке Java [13, 14]. Система ориентирована на визуальность и простоту. Включена реализация двумерного мира с различными объектами, для тестирования нейронных сетей, используемых для управления. 

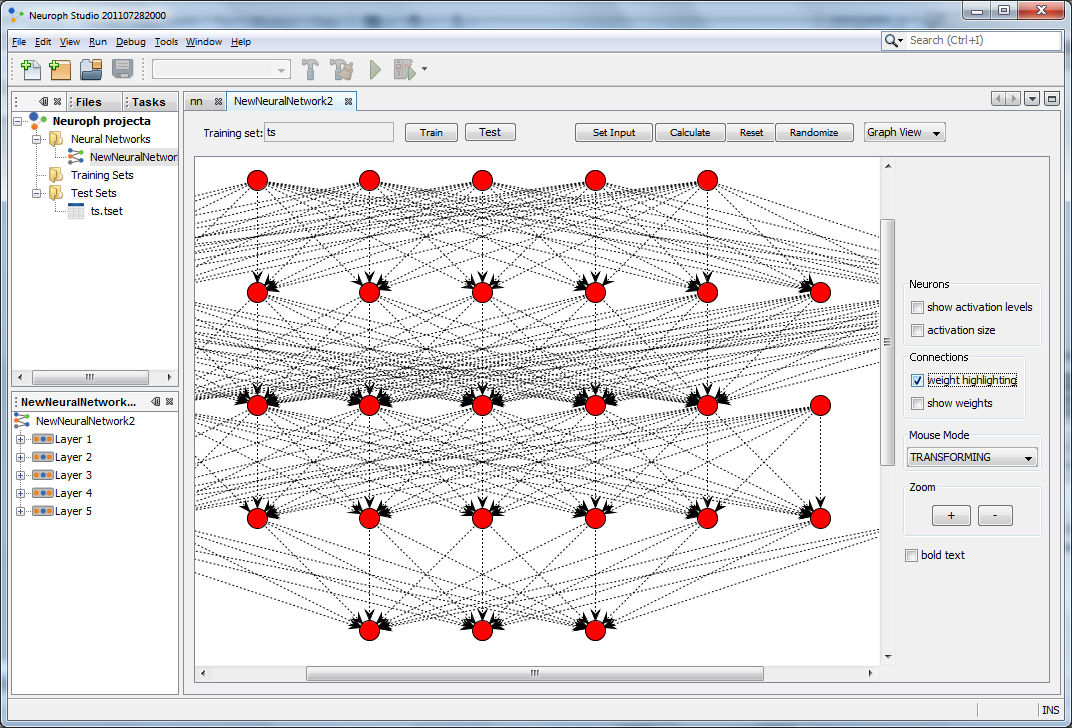
В системе реализовано большое количество различных вариантов нейронов и типов сетей, присутствует обучение без учителя.

Создание и редактирование нейронных сетей в среде Simbrain происходит визуально. В случае визуального построения, пользователь может видеть процесс изменения весов в сети, передачи сигнала. Возможно отслеживать и алгоритм обратного распространения ошибки. Но пользователь также может воспользоваться Java API для конструирования собственных задач и добавления необходимых технологий в систему.

В системе нет реализации сверточных нейронных сетей, предназначенных для распознавания изображений. Ввод выборок в сеть осуществляется вручную, что не всегда удобно.

## Neuroph

Neuroph – система для нейросетевого распознавания, написанная на языке Java [13, 15]. Состоит из Java API, включающего в себя основные классы, классы-утилиты и реализацию конкретных типов нейронных сетей. Также включает в себя среду Neuroph Studio, реализованную на платформе NetBeans.



В среде реализованы следующие нейросетевые архитектуры:

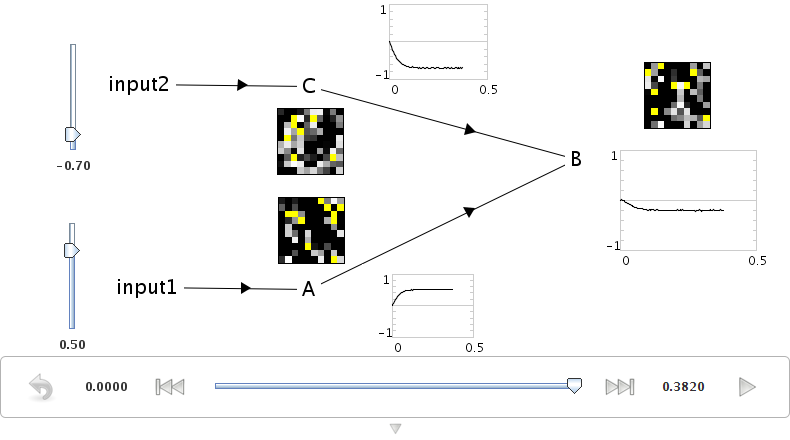
* Адалайн
* Персептрон
* Многослойный персептрон с алгоритмом обратного распространения ошибки, моментом
* Сеть Хопфилда
* Двусторонняя ассоциативная память
* Сеть Кохонена
* Сеть Хебба
* RBF нейронная сеть

В систему включены примеры использования сетей для прогнозирования цен на финансовых рынках, примеры классификации животных и другие. В сеть включена подержка распознавания изображений с помощью многослойного персептрона с алгоритмом обратного распространения ошибки.

В сети нет реализации сверточных нейронных сетей.

## Nengo

Nengo – еще одна система, написанная с помощью Java и Python [13, 16]. На ней было реализовано большое число прикладных задач управления и распознавания. Имеет встроенный графический интерфейс:



В Nengo так же, как и в рассмотренных выше, отсутствует реализация сверточных нейронных сетей.

## Encog

Система Encog является наиболее полной из рассматриваемых [13, 17]. Она включает в себя большое количество эффективно написанных Java классов. В отличие от прочих систем, в ней реализованы не только нейросетевые алгоритмы распознавания, но и кластеризация, генетические алгоритмы, скрытые марковские модели, байесовские сети и др.

Среди поддерживаемых архитектур нейронных сетей такие, как адалайн, машина Больцмана, нейронные сети обратного распространения, рекуррентные сети Эльмана, рекуррентные сети Джордана, самоорганизующиеся карты Кохонена и т.д. Доступны следующие функции активации нейронов: биполярная, соревновательная, функция Элиотта, функция Гаусса, гиперболический тангенс, линейная, синусоидальная, сигмоида.

В сравнении с другими система, Encog написана очень эффективно, поддерживается параллельное функционирование сетей на машинах с несколькими процессорами. Но Encog так же не поддерживает распознавание изображений – нет реализации сверточных нейронных сетей.

## Существующие реализации сверточных сетей

В интернете доступно некоторое количество свободных реализаций сверточных сетей. Но все они написаны в виде приложения для одной задачи. Реализации не позволяют добавлять новые архитектуры сетей, изменять характеристики обучения, оптимизировать структуру сети.

## Выводы из обзора

Рассмотрев большое количество систем для распознавания образов, был сделан вывод, что все они не подходят для реализации поставленных задач.

Системы широкого профиля, работающие с разными задачами и включающие в себя большое количество доступных архитектур нейронных сетей, не могут напрямую применяться к задаче распознавания изображений. Они не включают в себя реализацию сверточных сетей – слоев свертки и субдискретизации.

Известные же реализации сверточных сетей не могут быть использованы как системы, не обладают достаточной гибкостью. В большинстве случаев, они написаны для одной-двух выбранных задач.

# Описание решения

Для обучения распознавателя в общем случае необходимо подать на вход изображения, выбрать архитектуру сети, задать необходимые параметры и запустить сам процесс обучения.

# Варианты слоев нейронной сети

## Полносвязный слой

Самый простой и широко применимый слой в нейронной сети – полносвязный [1, 8]. Каждый нейрон в этом слое – персептрон с нелинейной функцией активации. В качестве функции активации принято использовать либо логистическую функцию , либо гиперболический тангенс . Оба варианта реализованы в системе и доступны.

На вход каждому из нейронов полносвязного слоя подаются выходы всех нейронов предыдущего слоя. Нейрон считает сумму своих входов, умноженных на веса, добавляет порог, и полученное значение пропускает через функцию активации. То, что получилось в итоге, подается на выход нейрона.

где – вход нейрона, – вес, сопоставленный этому входу, – аддитивный обучаемый порог.

Работа самой сети происходит «слева направо», от входного слоя к выходному – так перемещается сигнал.

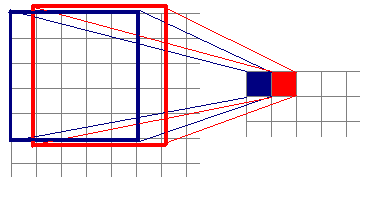
Для обучения полносвязного слоя используется алгоритм обратного распространения ошибки. Не будем приводить все необходимый формулы, остановимся лишь на основной сути.

Ошибка распространяется в обратную сторону, от выходного сигнала к входному. Вначале на каждом слое считается ошибка для предыдущего, потом корректируются веса, ведущие к нейронам слоя от предыдущего. Формула коррекции веса конкретной связи получается следующей:

Где - вес связи, - скорость обучения, – сумма ошибок, посчитанная для данного нейрона слоя, - выход нейрона, связанного с нейроном текущего слоя.

## Слой свертки

Слой свертки – способ уменьшить количество тренируемых параметров в слое нейронной сети. Слой состоит из нескольких карт, представляющих из себя матрицу нейронов, связанных с предыдущим слоем. Каждый из нейронов соединен с заданной прямоугольной областью. При создании слоя нужно задать размеры прямоугольной области, смещение области для соседних нейронов (обычно области располагаются внахлест), размеры карт, а также матрицу связей карт, которая показывает, какие карты текущего слоя связаны с какими картами предыдущего.



Отличие от полносвязного слоя заключается в том, что у нейронов каждой карты общие веса. Получается, если каждый нейрон соединен с областью 5x5 предыдущего слоя, то у всей карты всего 26 обучаемых весов (25 весов – область и 1 вес – порог). В разных картах наборы обучаемых весов разные и никак не зависят друг от друга.

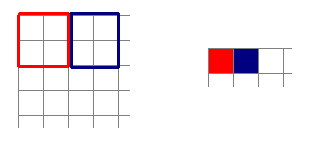
Для того, чтобы не хранить избыточную информацию, система хранит индекс веса вместо самого веса. Прямой проход в слое свертки происходит абсолютно таким же образом, как и в полносвязном слое – от входного слоя к выходному. При этому нужно учитывать, что веса у нейронов общие.

В самом обучении используется модификация алгоритма обратного распространения ошибки. Поскольку веса общие, то и коррекцию нужно проводить, используя данные об ошибках всех нейронов, пользующихся данным весом. Для этого мы сначала считаем сумму ошибок на всех нейронах, а потом, используя эту сумму как , обновляем значение веса. В качестве в формуле модификации веса алгоритма обратного распространения мы берем все выходные значения нейронов, к которым ведут связи, имеющие данный общий вес.

Сверточные слои нейронной сети отвечают за извлечение особенностей входного изображения и слоев, предшествующих им.

## Слой субдискретизации

Слой субдискретизации по своей структуре напоминает слой свертки. В нем так же, как и в слое свертки, каждый нейрон карты соединен с прямоугольной областью на предыдущем. Нейроны имеют нелинейную функцию активации – логистическую или гиперболический тангенс. Только, в отличие от слоя свертки, области соседних нейронов не перекрываются. В слое свертки, каждому нейрону области ведет собственная связь, имеющая вес. В слое субдискретизации происходит каждый нейрон усредняет выходы нейронов области, к которой он присоединен. Получается, что каждая карта имеет всего по два настраиваемых веса: мультипликативный (вес усреднения нейронов) и аддитивный (порог).



Вместо функции усреднения нейронов области может быть использована функция взятия максимума или функция взятия значения медианы. Оба этих варианта также внедрены в систему и могут быть использованы.

Чаще всего, при построении сети слои свертки и субдискретизации чередуются, чтобы обеспечить извлечение признаков при достаточно малом количестве тренируемых параметров.

# Методы оптимизации работы нейронной сети

## Стохастический и пакетный режимы обучения

Для обучения нейронной сети в течение одной эпохи могут быть использованы два различных подхода [1, 6]. В первом варианте мы подаем на вход нейронной сети примеры из обучающей выборки один за другим. После каждого примера мы обновляем веса сети. Такой подход называется стохастическим (онлайн) обучением. Во втором варианте мы подаем на вход нейронной сети целый пакет обучающих примеров, после чего обновляем веса сети. Внутри пакета мы накапливаем ошибку на весах сети, чтобы потом обновить их.

Рассмотрим стохастистический режим чуть более подробно. При подсчете ошибки на примере обучающей выборки нужно вычислить градиент функции ошибки на каждом слое нейронной сети. Используемая функция ошибки – среднеквадратическая.

где множество C включает в себя все нейроны выходного слоя сети, - ошибка для j-ого нейрона выходного слоя, разница между выходом этого нейрона и желаемым значением. Из-за того, что градиент мы вычисляем приближенно, в вычисления добавлется некоторый шум, который влияет на коррекцию весов. Но данный шум может быть полезен для обучения. Рассмотрим преимущества стохастического обучения перед пакетным:

* Стохастическое обучение в большинстве случаев гораздо быстрее пакетного
* Стохастическое обучения часто приводит к лучшим распознавателям.
* Стохастическое обучение можно использовать для отслеживания изменений

Представим случаей, в котором обучающая выборка размером 1000 состоит из 10 идентичных наборов по 100 примеров. Усреднение градиент по всей тысяче примеров в пакетном обучении дает такой же результат, как и вычисление градиента, основанное лишь на первой сотне примеров. Получается, что пакетное обучение вычисляет одно и то же значение 10 раз перед тем, как обновить веса нейронной сети. Стохастическое обучение, напротив, представит целую эпоху как 10 итераций (эпох) по обучающему набору длины 100. На практике примеры редко встречаются более одного раза в обучающей выборке, но все-таки попадаются кластеры очень похожих друг на друга примеров.

Нелинейные сети часто имеют множество локальных минимумов различной глубины. Задача обучения заключается в попадании сети в один из минимумов. Пакетное обучение приведет к минимуму, в окрестностях которого изначально расположены веса. В стохастическом обучении, шум, появляющийся при коррекции весов, приводит к тому, что сеть прыгает от одного локального минимума к другому, возможно более глубокому.

Рассмотрим теперь преимущества пакетного режима обучения перед стохастическим:

* Условия сходимости хорошо изучены
* Большое количество техник ускорения обучения работают только с пакетным режимом
* Теоретический анализ динамики изменения весов и скорости сходимости проще

Эти преимущества проистекают из того же фактора шума, который присутствует в стохастическом обучении. Этот шум можно убирать различными способами. Несмотря на некоторые преимущества пакетного режима, стохастический метод обучения используется гораздо чаще, в особенности в тех задачах, в которых обучающая выборка велика. В системе реализованы оба метода обучения.

## Перемешивание обучающих примеров

Нейронные сети обучаются быстрее всего на наиболее непредсказуемых примерах из обучающей выборки [2]. Поэтому лучше всего выбирать на каждой итерации наименее знакомый системе пример. Заметим, что это относится только к стохастическому обучению, так как для пакетного порядок, в котором предъявляются примеры, неважен. Конечно, нет простого способа понять, какие обучающие примеры несут в себе большее количество информации, но существует возможность, которая использует ту же идею – выбирать в качестве последующего обучающего примера тот, который принадлежит к другому классу. Обучающие примеры из одного класса чаще всего содержат похожую информацию.

Для использования данного принципа в системе используется перемешивание обучающих примеров перед подачей их нейронной сети. В перемешанном наборе вероятность того, что подряд идущие примеры содержат похожую информацию, резко уменьшается.

## Нормализация входных данных

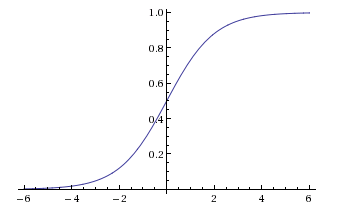
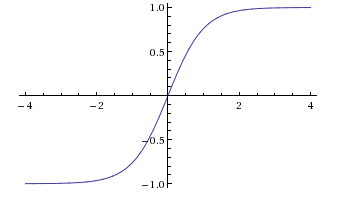
Сходимость весов нейронной сети во время обучения ускоряется, если среднее значение по всей обучающей выборке каждой входной переменной близко к нулю [5]. Представим экстремальный случай, когда все входы сети положительны. Веса конкретного нейрона первого скрытого слоя обновляются на значение, пропорциональное , где - скаляр ошибки в этой вершине, а - входной вектор. Когда все компоненты входного вектора положительны, все изменения весов, связанных с нейроном, будут иметь одинаковый знак. Как результат, эти веса смогут либо все увеличиваться, либо все уменьшаться вместе на данном примере из обучающей выборки. Таким образом, если вектор весов должен сменить направление, он сможет это делать только зигзагообразно, что неэффективно и потому очень медленно.

В приведенном выше примере, все входные значения положительны. В общем случае, любой сдвиг среднего значения от нуля сместит изменения весов в определенном направлении и, в связи с этим, замедлит обучение. Поэтому хорошо сказывается на производительности смещение значений всех входов так, чтобы среднее значение было максимально близко к нулю. Эту эвристику следует применить ко всем слоям нейронной сети, так как выходы какдого из слоев являются входами для последующего слоя. Эту трудность можно решать путем выбора функции активации, которая будет трансформировать входные значения. Выбор функции активации описан в следующем разделе.

Перед обучением распознавателей в системе происходит нормализация входных значений для обучающих и проверочных изображений.

## Сигмоидальная функция

Нелинейные функции активации – то, что дает нейронным сетям их нелинейные возможности. Одна из наиболее часто распространенных форм функций активации – сигмоидальная функция, которая является монотонно возрастающей и имеет некоторые конечные асимптоты при приближении к . Самые популярные примеры – стандартная логистическая функция и гиперболический тангенс [1, 7].

Сигмоидальные функции, симметричные относительно начала координат предпочтительнее по той причине, что входы нейронной сети должны быть нормализованы. Симметричные сигмоидальные функции с большей вероятностью дадут выход (который является входом для следующего уровня), который в среднем близок к нулю. В противоположность, например, стандартной логистической функции, чьи выходы всегда положительны, а потому и среднее значение выхода больше нуля.

Нейронные сети с симметричными сигмоидальными функциями, как гиперболический тангенс, часто обучаются быстрее, чем стандартная логистическая функция.

Рекомендуемая сигмоида - . Так как функция гиперболического тангенса вычислительно затратна, можно использовать приближение полиномиальной дробью.

Иногда бывает полезно добавить небольшой линейный член, например для того, чтобы избегать плоских мест на поверхности функции ошибки.

Константы в рекомендованной сигмоидальной функции подобраны таким образом, чтобы, при использовании с нормализованными входными данными, дисперсия выходов была также близка к 1. Эта сигмоидальная функция имеет свойства , вторая производная достигает максимума при и эффективный прирост близок к 1.

В системе реализованы как рекомендуемая сигмоидальная, так и стандартная логистическая функции активации. Более того, система позволяет добавлять собственные функции активации, реализовав нужный интерфейс.

## Выбор скорости обучения

Существует большое количество предложенных в литературе методов, позволяющих автоматически подстраивать скорости обучения [4]. Большинство таких схем уменьшают скорость обучения, когда вектор весов «осцилирует», и увеличивают, когда веса меняются в относительно стабильном направлении. Главная проблема данных методов заключается в том, что они не применимы для стохастического градиента или обучения онлайн из-за постоянных колебаний вектора весов.

В общем случае, выбор разных скоростей обучения для разных весов может улучшить сходимость. Главная философия метода, используемого в системе, заключается в том, чтобы все веса нейронной сети сходились примерно одинаково.

Основываясь на кривизне поверхности функции ошибки, некоторые веса могут потребовать малую скорость обучения, для избежания дивергенции, а другие могут потребовать большую скорость обучения, чтобы сходится за приемлимое время. В соответствии с этим, скорости обучения в нижних слоях нейронной сети должны быть в целом больше, чем в верхних слоях. Это подтверждается тем фактом, что в большинстве архитектур нейронных сетей вторая производная функции стоимости, по отношению к весам нижних слоев, меньше, чем в верхних слоях.

Все эти улучшения позволяют выделить следующие правила, улучшающие обучение нейронной сети:

* Каждому весу нейронной сети следует сопоставить свою скорость обучения.
* Скорость обучения должна быть пропорциональна корню квадратному из количества входов нейрона.
* Скорости обучения в нижних слоях нейронной сети должны быть больше, чем в верхних.

В системе скорость обучения для весов выбирается с помощью алгоритма, высчитывающего Гессиан (матрицу вторых частных производных) для каждого из весов.

# Структура системы

Классы, написанные на языке Java, которые составляют систему, делятся на несколько частей.

В пакете layers лежат классы, описывающие различные слои, реализованные в системе. Все они реализуют интерфейс Layer, включающий в себя основные методы слоя нейронной сети. Пакет functions содержит реализацию различных функций активации для нейронов сети. Пакет network включает в себя классы, реализующие саму сеть, связи между нейронами, карты признаков и т.д. В пакете dataprovider представлены классы, которые нужны для предобработки изображений и для подачи их на вход нейронной сети. Пакет test содержит класс, в котором реализовано построение нейронного распознавания для тестирования системы.

На вход нейронной сети подается вектор вещественных чисел, задающих нормализованные цвета пикселей изображения. Размер и характеристики входа распознавателя настраиваются при описании сети. На выходе мы получаем вектор, длина которого равна количеству классов распознаваемых изображений. 1 на месте в выходном векторе показывает принадлежность изображения к классу.

После задания топологии нейронной сети, параметров обучения и количества эпох, работа системы происходит в автоматическом режиме.

# Тестирование системы

Для тестирования разработанной системы была выбрана база изображений рукописных цифр MNIST [3].

## Описание базы MNIST

База содержит в себе 60000 обучающих и 10000 тестовых изображений. Каждое изображение разбито на два файла – собственно изображение и метка к нему (цифра от 0 до 9, правильный класс изображения). Размер изображения каждой цифры не более 20x20, но они вписаны в квадрат размером 28x28. Вот пример первых 12 цифр обучающей выборки цифр:

image

Каждый пиксель изображения кодируется числом от 0 до 255, показывающим значение пикселя в градациях серого. 0 соответствует черному цвету, а 255 – белому.

## Предобработка изображений

Как уже было сказано в части, посвященной методам оптимизации, необходимо нормализовать параметры, чтобы в среднем значение входа было близким к единице. Для этого производится линейное преобразование, проецирующее интервал [0, 255] в интервал [-1, 1].

Распознаватель должен быть инвариантным к различным незначительным искажениям входных обучающих и тестовых изображений. Небольшой поворот, шум или масштабирование не должны сказываться на качестве распознавания. Для того, чтобы построенный распознаватель удовлетворял данным критериям, в процессе предобработки к изображениям цифр применяются эластичные (нелинейные) искажения.

Линейные аффинные искажения получаются легко, путем применения масштабирования, сдвига и поворота к обучающим изображениям. Но еще лучшего качества распознавания изображений получается добиться, используя нелинейные искажения, которые формируются следующим образом. Сначала создается гауссово ядро посредством построения матрицы , значение элемента (x, y) которой вычисляется по формуле:

,

где – размер ядра. Чаще всего это ядро выбирают нечетным и пропорциональным радиусу так, чтобы в это ядро попадали значения функции Гаусса вплоть до уменьшения от максимального значения в пределах 0,1-0,3.

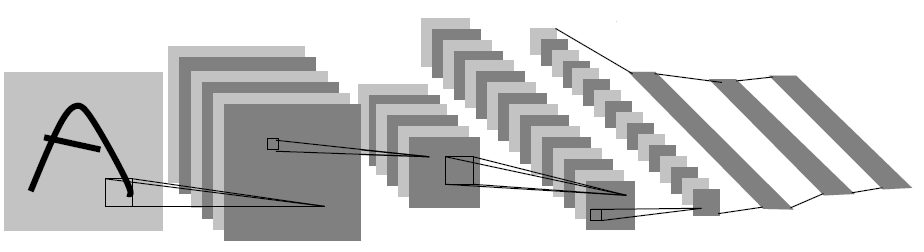
Далее генерируются две матрицы размером (по размеру входного изображения). Значения элементов матриц выбираются как случайные величины, равномерно распределенные на отрезке [-1, 1]. К этим матрицам применяется размытие изображения с использованием ранее сформированного ядра. Далее ищется новое значение для каждого пикселя входного изображения с применением билинейной интерполяции и сформированных матриц. Так выглядит пример применения эластичных искажений к изображению из обучающей выборки базы MNIST:



При обучении нейронной сети эластичные искажения применяются на каждой эпохе. Перед тем, как подавать изображения обучающей выборки на вход распознавателя, к каждому применяется искажение (и они подаются в случайном порядке). Таким образом, получается, что в каждой эпохе набор изображений не похож на изображения других эпох.

## Описание решения и топология сети

Для данной задачи распознавания строится и обучается нейронная сеть. Сеть состоит из семи слоев, каждый из которых содержит обучаемые веса.



Вход нейронной сети – изображение 32x32 пикселя. Мы помещаем входное изображение размером 28x28 в квадрат 32x32 для повышения качества обучения сети. Все значения цвета входных пикселей нормализованы так, что цвет фона (белый) имеет значение -0.1, а цвет цифры (черный) – значение 1.175. Благодаря этому, среднее входное значение становится примерно 0, а дисперсия приближается к 1, что ускоряет обучение нейронной сети.

Первый слой – слой свертки с шестью картами признаков. Каждый нейрон на каждой из карт соединен с областью 5x5 на входном изображении. Размер одной карты – 28x28. Первый слой содержит 156 тренируемых весов и 122304 соединений.

Второй слой – слой субдискретизации с шестью картами признаков размером 14x14. Каждый нейрон на картах соединен с областью 2x2 на соответствующей карте из первого слоя. Значения четырех входов нейрона складываются, умножаются на обучаемый коэффициент и к ним добавляется обучаемый порог. Результат прогоняется через гиперболический тангенс. Рецептивные поля 2x2 не накладываются друг на друга, поэтому во втором слое, в каждой из карт, в два раза меньше строк и столбцов, чем в картах первого слоя. Второй слой содержит 12 обучаемых параметров и 5880 соединений.

Третий слой – слой свертки, состоящий из 16 карт признаков. Каждый нейрон соединен с областью 5x5, находящейся в одинаковом месте на подмножестве из карт признаков второго слоя. Таблица связей между картами третьего и второго слоев:



Почему мы не соединяем каждую карту третьего слоя с каждой картой второго? На это есть две причины. Во-первых, неполносвязная схема соединений позволяет сохранить количество связей в разумных пределах. Что более важно, это позволяет избежать симметрии в сети. Разные карты признаков вынуждены извлекать различные признаки из предыдущего слоя, так как соединены с разными наборами карт. Первые шесть карт получают информацию из всех последовательностей карт длины 3. Следующие шесть карт – из последовательностей длины 4. Следующие три – из некоторых непоследовательных наборов длины 4. Последняя карта, наконец, соединена со всеми картами признаков второго слоя. Третий слой содержит 1516 обучаемых параметров и 151600 соединений.

Четвертый слой – слой субдискретизации с 16 картами признаков размером 5x5. Каждый нейрон в каждой карте признаков соединен с областью 2x2 на соответствующей карте третьего слоя, таким же образом, как и соединения между вторым и первым слоями. Четвертый слой содержит 32 обучаемых параметра и 2000 соединений.

Пятый слой – слой свертки с 120 картами признаков. Каждый нейрон соединен с областью 5x5 на всех шестнадцати картах признаков четвертого слоя. Поскольку размер карт четвертого слоя также 5x5, размер каждой из карт пятого слоя – 1x1. Достигается полная связность между пятым и четвертым слоями. Пятый слой содержит 48120 обучаемых весов.

Шестой слой – полносвязный, состоит из 84 нейронов. Содержит 10164 параметра.

Последний, седьмой слой, состоит из 10 нейронов, соединенных со всеми нейронами предыдущего слоя. Это и есть выходной слой.

Обучение сети происходит в течение 28 эпох, на каждой из которых на вход подается вся обучающая выборка из 60000 изображений. В начале эпохи к каждому изображения обучающей выборки применяется эластическое искажение. Обучающие примеры подаются в случайно перемешанном порядке.

Получающие распознаватели делают ошибку на менее чем 1% тестовых примеров (<100 изображений). Конкретное число ошибок зависит от первоначального случайного задания весов сети.

# Заключение

В результате данной дипломной работы:

* Проведено исследование существующих подходов к задаче распознавания изображений
* Изучены современные нейросетевые технологии
* Разработана и реализована система для построения нейросетевых распознавателей на языке Java
* Изучены и встроены в систему алгоритмы ускорения обучения и повышения качества распознавания
* Проведено тестирование распознавателя, построенного с помощью системы, на реальных данных

## Продолжение исследования

В дальнейшем, систему можно улучшать следующим образом:

* Расширить список поддерживаемых нейронных слоев и функций активации
* Внедрить поддержку параллельных вычислений на многопроцессорных системах, описанную в [9, 10, 11]
* Провести тестирование на других прикладных задачах классификации изображений, которые представлены в [12]

# Список литературы

1. Хайкин, С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1104 с.
2. LeCun, Y. Scaling learning algorithms towards AI / Y. LeCun, Y. Bengio – MIT Press, 2007.
3. LeCun, Y. The MNIST database of handwritten digits – <http://yann.lecun.com/exdb/mnist>
4. LeCun, Y.Efficient BackProp in Neural Networks: Tricks of the trade / Y. LeCun, L. Bottou, G. Orr, K. Muller – Springer, 1998. – 44 p.
5. Bishop, C.M. Neural Networks for Pattern Recognition – Oxford University Press, 1995. – 498 p.
6. Yoshua Bengio, Y. Le Cun, and D. Henderson. Globally trained handwritten word recognizer using spatial representation, space displacement neural networks and hidden Markov models. In Advances in Neural Information Processing Systems 6, San Mateo CA, 1994. Morgan Kaufmann.
7. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Neural Information Processing Systems, 2007.
8. D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. Neural Computation, 22(12):3207–3220, 2010.
9. D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In International Conference on Document Analysis and Recognition, pages 1250–1254, 2011.
10. D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In International Joint Conference on Artificial Intelligence, pages 1237–1242, 2011.
11. A. Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, Computer Science Department, University of Toronto, 2009.
12. C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. Chinese Handwriting Recognition Contest. In Chinese Conference on Pattern Recognition, 2010.
13. Comparison of Neural Network Simulators – <http://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators>
14. Simbrain – <http://www.simbrain.net/>
15. Neuroph – <http://neuroph.sourceforge.net/>
16. Nengo – <http://nengo.ca/>
17. Encog – <http://www.heatonresearch.com/encog>