

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего
профессионального образования
«Тверской государственный технический университет»
(ТвГТУ)

Кафедра Программное обеспечение вычислительной техники

К защите допустить:
Заведующий кафедрой

«____» _____ 2017 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
МАГИСТРА**

на тему: «Распознавание изображений CAPTCHA методами машинного обучения»

Направление: информационные технологии

Профиль: программная инженерия

Форма обучения: очная

Группа: 1106

Студент(ка): Коробейников Сердар Рустамович

_____ (подпись, дата)

Руководитель: К. т. н. доцент Мальков А. А.

_____ (подпись, дата)

МИНОБРНАУКИ РОССИИ

ТВЕРЬ

2017

РЕФЕРАТ

Текст 88 с., 3 ч., 24 рис., 0 табл., 74 источн., 2 прил.

Ключевые слова. Интеллектуальный анализ данных, машинное обучение, искусственные нейронные сети, обработка изображений, сегментация изображений, CAPTCHA, python.

Keywords. Data mining, machine learning, artificial neural networks, image processing, image segmentation, CAPTCHA, python.

В работе рассматриваются современные методы машинного обучения и интеллектуального анализа больших объемов данных в применении к задачам распознавания текстовой информации на графических изображениях. Нами последовательно анализируется метод искусственных нейронных сетей для автоматизации процесса распознавания и обсуждаются различные алгоритмы для предварительной обработки изображений (удаления шумов, сегментации и т.д.). Изученный подход применяется для автоматизированного распознавания рукописных цифр и CAPTCHA. В данной работе была создана нейросетевая архитектура и набор алгоритмов для предварительной обработки, сегментации и распознавания изображений, содержащих текстовую информацию. Для практической реализации построенного подхода было создано браузерное приложение и серверная часть, позволяющие производить распознавание CAPTCHA в автоматическом режиме. Результаты данной работы могут быть использованы для построения сервиса автоматизированного сбора информации, разработки рекомендаций по улучшению существующих алгоритмов построения CAPTCHA, а также для решения иных задач распознавания изображений, включая распознавание автомобильных номеров, различных объектов на биомедицинских изображениях и др.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	4
ВВЕДЕНИЕ	5
1 Методы машинного обучения для распознавания изображений	9
1.1 Введение	9
1.2 Методы машинного обучения и анализ больших объемов данных	10
1.3 Нейроны в искусственных нейронных сетях	15
1.4 Полносвязные искусственные нейронные сети	21
1.5 Обучение искусственных нейронных сетей	27
2 Машинное обучение для автоматического распознавания CAPTCHA	38
2.1 Введение	38
2.2 Методы распознавания CAPTCHA	39
2.3 Распознавание отдельных символов и сегментация CAPTCHA	44
2.4 Современные методы автоматического распознавания	52
3 Реализация метода автоматического распознавания CAPTCHA	59
3.1 Введение	59
3.2 Архитектура искусственной нейронной сети	60
3.3 Программный пакет для распознавания CAPTCHA	64
3.4 Результаты практического применения программного пакета	67
ЗАКЛЮЧЕНИЕ	70
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	73
ПРИЛОЖЕНИЕ А. Сингулярное разложение	79
ПРИЛОЖЕНИЕ Б. Результаты обучения на базе MNIST	81

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В данной работе применяются следующие определения, обозначения и сокращения.

Тензор – одномерный или многомерный упорядоченный массив вещественных или комплексных чисел.

Вектор – одномерный упорядоченный массив вещественных или комплексных чисел (одномерный тензор).

Матрица – двухмерный упорядоченный массив вещественных или комплексных чисел (двухмерный тензор).

$\mathbf{x} \in \mathbb{R}^N$ – вектор с числом элементов N .

$\mathbf{X} \in \mathbb{R}^{I \times J}$ – матрица размера $I \times J$.

$\mathbf{I}_J \in \mathbb{R}^{J \times J}$ – диагональная единичная матрица размера $J \times J$.

ИНС – искусственная нейронная сеть.

Графическое изображение – двумерный упорядоченный набор пикселей. В данной работе мы будем рассматривать только изображения, заданные в 256-цветной палитре, где 0 соответствует черному цвету, а 255 – белому цвету. Также мы будем использовать матричное представление изображения в виде $\mathbf{X} \in \mathbb{R}^{W \times H}$, где W – высота изображения, H – ширина, и при этом первый элемент матрицы всегда соответствует верхнему левому углу изображения.

Символ – буква английского/русского алфавита или цифра, если не указано иное.

CAPTCHA – Completely Automated Public Turing test to tell Computers and Humans Apart. В данной работе мы будем подразумевать под CAPTCHA графическое изображение, содержащее несколько упорядоченных символов (возможно с определенными искажениями), а также различные искажающие элементы.

ВВЕДЕНИЕ

В 21-ом веке эффективные методы машинного обучения и интеллектуального анализа больших объемов данных [1] приобрели особую важность практически для всех областей научных и прикладных знаний. Наибольшее распространение данные методы на сегодняшний день получили в контексте применения к проблеме машинного зрения, включающей в себя обработку изображений, распознавание объектов на изображениях, классификацию, кластеризацию и генерацию текстового описания, генерацию фотoreалистичных изображений и ряд иных задач. В данной работе мы рассматриваем задачу автоматизированного распознавания CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) [2], которая является частным случаем более общей проблемы распознавания текстовой информации на графических изображениях, соответственно методы, разрабатываемые при распознавании CAPTCHA, могут использоваться затем в более простых (в том смысле, что при создании CAPTCHA применяются искусственные методы зашумления изображений, препятствующие ее распознаванию) задачах распознавания надписей, текстов, и т.д. С другой стороны, разработка методов распознавания CAPTCHA представляет самостоятельный интерес, в частности, для анализа степени надежности уже существующих систем генерации CAPTCHA и формулировки рекомендаций для эффективной реализации надежных систем такого рода.

При разработке любой CAPTCHA, к ней предъявляются два, зачастую конкурирующих требования:

- сложность автоматизированного распознавания;
- простота распознавания человеком.

При этом для разработки CAPTCHA применяются различные методы размытия изображений, линейных и нелинейных трансформаций, частичное наложение отдельных символов, зашумление изображения отдельными пикселями или целыми линиями и т.д. Перечисленные методы в своей совокупности делают процесс автоматизированного распознавания CAPTCHA сложной задачей машинного обучения.

В качестве основной (конечной) цели работы рассматривается разработка набора алгоритмов и комплекса программного обеспечения для автоматизированного распознавания CAPTCHA методами машинного обучения. В соответствии с указанной целью определены следующие задачи исследования.

1. Изучение научно-технической базы по теме современных методов машинного обучения в контексте обработки и распознавания информации, представленной в графических изображениях.
2. Изучение современной литературы, посвященной подходам к автоматизированному распознаванию текстовой информации на графических изображениях и, в частности, автоматизированным методам распознавания CAPTCHA.

3. Отбор наиболее эффективных алгоритмов, а также при необходимости разработка собственных, для проведения процедуры автоматизированного распознавания изображений типа CAPTCHA, включающей: удаление шумов на изображении, сегментацию изображения, автоматизированное распознавание отдельных символов с помощью современных методов машинного обучения (искусственных нейронных сетей и т.д.).
4. Реализация комплекса программного обеспечения для автоматизированного распознавания CAPTCHA.
5. Тестирование и адаптация разработанного программного комплекса для ряда типов CAPTCHA, генерированных посредством специальных программных продуктов а также реальных CAPTCHA, используемых веб-сервисами.

Используемые в данной работе методы исследования включают в себя аналитические методы: методы фильтрации изображений, методы работы с многомерными массивами данных, методы машинного обучения (метод главных компонент, искусственные нейронные сети), а также аналитико-экспериментальные методы: вычислительный эксперимент в виде (имитационного) моделирования на ЭВМ объектов и задач исследования.

Достоверность полученных результатов определяется корректным применением использованных методов исследования. Она подтверждается совпадением результатов вычислительных экспериментов для тех данных, которые имеют аналоги в литературе, что позволяет сделать вывод об адекватности разработанных способов и моделей, а также эффективным (верным с заданной достоверностью) распознаванием ряда CAPTCHA, предварительно распознанных вручную, либо созданных самостоятельно на основе известного текста.

Результаты данной работы могут быть использованы для автоматизированного распознавания CAPTCHA для ряда «уместных» задач, для которых представляется оправданым использование автоматизированных систем распознавания — это могут быть, например, различные справочные системы (позволяющие установить наличие билетов в продаже, количество доступной для продажи продукции и т.д.), требующие ввода CAPTCHA для получения необходимой информации. Использование CAPTCHA в подобных системах затрудняет доступ к информации, препятствует проведению процедур по ее агрегированию, а также встраиванию в сторонние сервисы. Другое важное применение результатов — это возможность формулировки рекомендаций по эффективной разработке систем генерации CAPTCHA, устойчивых к алгоритмам автоматизированного распознавания. И, наконец, поскольку задача распознавания CAPTCHA является частным случаем задачи распознавания текстовых изображений, результаты данной работы могут быть обобщены в дальнейшем на такие актуальные задачи, как распознавание текстов, распознавание автомобильных номеров по фотоснимкам и записям камер видеонаблюдения и т.д.

В первом разделе мы проводим анализ наиболее распространенных методов машинного обучения и работы с большими объемами данных, и подробно рассматриваем формализм метода искусственных нейронных сетей (artificial neural network, далее мы будем использовать сокращение ИНС) [3, 4]. Метод ИНС является перспективным методом машинного обучения, позволяющим компьютеру выявлять сложные закономерности, фактически обучаться, на данных в цифровой форме. Данный метод изначально возник [5] как попытка математического и численного моделирования деятельности головного мозга (в частности, головного мозга человека). Классические полно связные ИНС состоят из отдельных нелинейных элементов – искусственных нейронов, являющихся простейшей моделью естественных (биологических) нейронов – основы мыслительного аппарата головного мозга. Каждый нейрон характеризуется набором параметров (весов), которые подстраиваются в процессе обучения для наилучшего соответствия выхода сети, требуемому (правильному) значению для данного набора входных данных. Так, например, для задачи распознавания цифр или букв алфавита на изображении, входными данными для сети будут значения интенсивности (для случая монохромного изображения) в каждом пикселе изображения, а желаемым выходом – правильно распознанное число или символ (естественно, в кодированной форме). Процесс обучения заключается в последовательном предъявлении изображений с известным содержанием (например, предварительно распознанным вручную) на вход сети и соответствующей модификации весов нейронов сети для получения наилучших (например, в среднем наиболее точных) ответов ИНС на всей тестовой выборке.

Направление ИНС на сегодняшний день является одним из наиболее активно развивающихся как в контексте разработки новых подходов и алгоритмов, так и в контексте использования ИНС во все большем числе практических приложений, большинство из которых связано с графическими изображениями (распознавание, классификация, улучшение, генерация), распознаванием речи и обработкой естественного языка. Быстрое развитие, в том числе, связано с активной поддержкой темы крупнейшими компаниями сферы информационных технологий, такими как, например, Google, Microsoft, Facebook.

Существенной и привлекательной особенностью метода ИНС является его (по крайней мере, принципиальная) универсальность – если в классических подходах (в том числе, машинного обучения) разрабатывается конкретный алгоритм, а затем компьютер производит понятные вычисления в соответствии с этим алгоритмом, то ИНС самостоятельно (используя, естественно, тестовый набор данных) обучаются решать предъявленные им проблемы, фактически генерируя алгоритм (заключающийся в наборе параметров весов обученной сети). Однако, стоит отметить, что с данной особенностью сопряжены также и определенные трудности на пути практического, в особенности, промышленного использования ИНС – имея обученную нейронную сеть, можно лишь использовать ответы (рекомендации) сети, при этом, практически невозможно понять «логику» ее работы и воспроизвести алгоритм вычислений. В этой связи, как было уже указано выше, на сегодняшний

день ИНС преимущественно применяются в приложениях, некритичных к ошибкам⁽¹⁾, а в областях, связанных с производством и медициной, ИНС пока находят лишь очень ограниченное применение и только в качестве рекомендательной системы, т.е. конечное решение всегда остается за человеком, специалистом соответствующего профиля⁽²⁾.

Далее во втором разделе работы мы рассматриваем основные подходы к задаче распознавания CAPTCHA, а также наиболее современные алгоритмы для повышения качества распознавания и ускорения процесса распознавания. Большинство современных CAPTCHA являются зашумленными графическими изображениями, содержащими несколько упорядоченных символов (букв определенного алфавита и/или арабских цифр). Процесс распознавания CAPTCHA предполагает предварительную фильтрацию изображения, его сегментацию на отдельные символы и последующее распознавание этих символов. Данные этапы также подробно рассматриваются во втором разделе работы.

Затем, в разделе 3 приводятся практические аспекты реализации программного комплекса для распознавания CAPTCHA, в частности, мы описываем использованные алгоритмы для предварительной обработки и сегментации CAPTCHA и выбранную архитектуру ИНС для распознавания отдельных символов. Также в данном разделе приводятся результаты применения созданного программного комплекса для задачи распознавания CAPTCHA.

Для практической реализации алгоритмов нами был выбран язык программирования python [6, 7, 8]. В качестве основных преимуществ python, послуживших причиной его выбора для данной работы, можно отметить такие как компактность и наглядность синтаксиса языка, мощная техническая поддержка, наличие крупнейшей свободной базы библиотек (пакетов) с реализованными алгоритмами практически во всех областях научной деятельности, простота в переносимости приложений и т.д.

Завершает данную работу Заключение, содержащее тезисное описание основных полученных результатов, выводы по проделанной работе и ряд рекомендаций для ее дальнейшего развития.

⁽¹⁾ Так например, если социальная сеть при добавлении пользователем новой фотографии верно распознала присутствующих на ней людей, указав ссылки на их профили в социальной сети, это будет полезной и привлекательной опцией, ведущей к росту популярности соответствующей социальной сети. При этом, неверный результат распознавания изображения не является критичным – в этом случае пользователь просто проигнорирует рекомендацию сети и воспользуется привычным ручным методом добавления ссылок на профили изображенных на фотографии людей.

⁽²⁾ ИНС, например, может указать врачу на высокую вероятность определенного диагноза по имеющимся данным обследования, однако дальнейшие медицинские исследования и окончательное принятие решения остаются за врачом.

1 Методы машинного обучения для распознавания изображений

1.1 Введение

В данной главе рассматриваются современные методы машинного обучения и интеллектуального анализа больших объемов данных, применимые к задаче распознавания изображений, содержащих текстовую информацию. В разделе 1.2 мы описываем наиболее распространенные на сегодняшний день методы машинного обучения, указываем их потенциальные возможности, известные недостатки и возможные области применения. В частности, кратко рассматриваются метод главных компонент, генетические алгоритмы, генетическое программирование и искусственные нейронные сети (ИНС).

Далее мы фокусируемся на наиболее перспективном, на наш взгляд, подходе – методе ИНС. Первые идеи в области метода ИНС появились в 40-х годах прошлого века при попытке создания математической модели деятельности головного мозга [5, 9, 10]. До 80-х годов данное направление оставалось по большей части теоретическим, и интерес в научном сообществе к ИНС быстро спадал, однако после выхода в 1986 году работы [11], где был фактически переоткрыт и существенно улучшен алгоритм обратного распространения ошибки, появились первые приложения, и к ИНС в полной мере вернулась былая популярность. Тем не менее, до конца первого десятилетия нашего века, данная парадигма все еще была далека от реального практического воплощения ввиду отсутствия больших размеченных объемов данных, необходимых для эффективного обучения сложных сетей, достаточно мощных вычислительных ресурсов, требуемых для быстрого обучения сетей и, собственно, достаточно устойчивых алгоритмов обучения, позволяющих проводить обучение сложных ИНС. Затем, отчасти из-за появления в открытом доступе больших объемов размеченных данных, отчасти из-за бурного роста доступных вычислительных мощностей (в первую очередь, графических процессоров), и отчасти из-за открытия внушительного набора эвристик (включая правильный выбор вида функции активации и функции стоимости, ряд подходов для решения проблемы переучивания сети и неустойчивости градиента и т.д.), некоторые из которых будут описаны в данной главе работы, стало возможным создание и эффективное обучение глубоких нейронных сетей, насчитывающих сотни тысяч нейронов и миллионы параметров.

В разделе 1.3 описывается принцип действия отдельного нейрона ИНС. Далее, в разделе 1.4 приводится систематическое описание парадигмы полносвязных ИНС, и в частности, рассматривается механизм распространения сигнала в сети. И наконец, в разделе 1.5 формулируются основные уравнения фундаментального метода обратного распространения ошибки для обучения ИНС на историческом наборе данных, а также обсуждаются наиболее популярные функции активации и функции стоимости.

1.2 Методы машинного обучения и анализ больших объемов данных

Один из наиболее популярных классических методов анализа данных – это **метод главных компонент⁽³⁾**, предложенный впервые в 1901-ом году Пирсоном [12] и затем переоткрытый и детально исследованный Хоттelingом [13]. Иногда данный метод называют преобразованием Кархунена-Лоэва или преобразованием Хоттelingа. Метод главных компонент позволяет понизить размерность пространства исходных признаков путем перехода в подпространство меньшей размерности при минимальности потери информативности (определенной как некоторый функционал специального вида). Данный метод выделяется простой логической конструкцией, используемой при его реализации, является широко распространенным и используемым в научных и технических приложениях, в том числе, в эконометрике, биоинформатике, алгоритмах обработки изображений и сжатия данных, а также в общественных науках.

Впервые метод главных компонент возник в своей исходной формулировке при решении Пирсоном задачи наилучшей аппроксимации конечного множества точек прямыми и плоскостями [12]. Таким образом, данный метод связан с получением наилучшей в некотором смысле проекции совокупности точек наблюдения в пространство меньшей размерности и основан на анализе матрицы расстояний. Затем Хоттelingом в работе [13] был предложен иной подход к построению главных компонент, основанный на выделении линейных комбинаций случайных величин, имеющих максимально возможную дисперсию, и опирающийся на ковариационную матрицу этих величин. Отметим, что в обоих случаях используется знание только матрицы вторых моментов рассматриваемых признаков.

В современной формулировке задача анализа главных компонент имеет, как минимум, четыре базовых версии:

1. аппроксимировать исходные многомерные данные некоторыми линейными многообразиями меньшей размерности;
2. для исходных многомерных данных найти подпространства меньшей размерности, в ортогональной проекции на которые разброс данных (то есть среднеквадратичное отклонение от среднего значения или дисперсия) максимальен;
3. для исходных многомерных данных найти подпространства меньшей размерности, в ортогональной проекции на которые среднеквадратичное расстояние между точками максимально;
4. для данной многомерной случайной величины построить такое ортогональное преобразование координат, в результате которого корреляции между отдельными координатами обратятся в нуль.

⁽³⁾ Для описания метода главных компонент автор частично использовал текст и основные результаты собственной выпускной квалификационной работы бакалавра «Разработка программного обеспечения для визуализации многомерных данных».

Первые три из приведенных выше версий оперируют конечными множествами данных, при этом, не используя никакой гипотезы о статистическом происхождении данных. С другой стороны, четвёртая версия фактически оперирует случайными величинами и конечные множества появляются в данном случае как конечные выборки из данного распределения. Соответственно, решение первых трёх из приведенных задач является приближением к разложению, получаемому из теоремы Кархунена — Лоэва (так называемого «истинного преобразования Кархунена — Лоэва» [14, 15]).

Программная реализация метода главных компонент предполагает первичную обработку исходных данных (вычисление выборочных средних, построение ковариационной матрицы и т.д.), нахождение спектра ковариационной матрицы и выбор оптимальной размерности в пространстве главных компонент. Алгоритм построения главных компонент в схематической форме можно описать следующим образом.

1. Получить матрицу входных данных \mathbf{X} размера $n \times p$, где p — это размерность пространства признаков, а n — количество измерений в выборке.

2. Вычислить вектор выборочных средних значений $\bar{\boldsymbol{\mu}}$ по формуле

$$\boldsymbol{\mu} \approx \bar{\boldsymbol{\mu}} = \left(\frac{1}{n} \sum_{k=0}^{n-1} \mathbf{X}[k, 0], \frac{1}{n} \sum_{k=0}^{n-1} \mathbf{X}[k, 1], \dots, \frac{1}{n} \sum_{k=0}^{n-1} \mathbf{X}[k, p-1] \right)^T.$$

3. Центрировать данные, то есть для каждой строки $\mathbf{X}[k, :]$ матрицы \mathbf{X} произвести преобразование $\mathbf{X}[k, :] \rightarrow \mathbf{X}[k, :] - \bar{\boldsymbol{\mu}}$, получив центрированную матрицу $\mathbf{X}^{(1)}$.
4. Вычислить сингулярное разложение⁽⁴⁾ для отмасштабированной преобразованной матрицы $\frac{1}{\sqrt{n}} \mathbf{X}^{(1)} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T$.
5. Используя матрицу \mathbf{V} , по формуле $\mathbf{z}' = \mathbf{V} \mathbf{x}$ получить преобразования исходных данных к главным компонентам.
6. Выбрать размерность p' подпространства в пространстве главных компонент и оставить в матрице \mathbf{V} первые p' столбцов, получив, соответственно, редуцированную матрицу \mathbf{V}' . Вычислить соответствующие преобразованные координаты по формуле $\mathbf{z}' = \mathbf{V}' \mathbf{x}$, где \mathbf{x} — это произвольная строка матрицы данных \mathbf{X} или новое измерение.

Полученный редуцированный базис позволяет в ряде случаев существенно понизить размерность входного пространства признаков, а также выделить наиболее существенные признаки, которые затем могут быть использованы для проведения классификации, кластеризации или распознавания. Более детально метод главных компонент, а также его практические приложения описываются, например, в обзорной статье [16].

⁽⁴⁾ Поскольку сингулярное разложение матрицы (SVD) будет также использовано далее в работе, то в Приложении 1 мы кратко приводим основные свойства данного разложения.

Отметим, что данный метод может применяться для задач распознавания объектов на графических изображениях. Так, например, в работе [17] модифицированный метод главных компонент применяется для распознавания человеческих лиц на изображениях. С использованием данного метода строится так называемое пространство лиц (face space) – множество простых базисных изображений, по линейной комбинации которых может быть построено изображение произвольного лица. Полученный базис может использоваться как для распознавания лиц, так и для решения задачи реконструкции лица по части изображения.

Существует также широкий ряд иных работ, где метод главных компонент применяется к различным задачам распознавания. Однако, данный метод на сегодняшний день объективно проигрывает современным алгоритмам машинного обучения, в частности нейросетевому подходу. Более того, с формальной точки зрения данный метод является не более чем модифицированным урезанным сингулярным разложением матрицы, а в области построения подобных урезанных (малопараметрических) разложений в последние годы произошел существенный скачок, и возникли новые более эффективные методы, которые будут кратко описаны далее в работе.

Чрезвычайно нестандартным для вычислительной математики подходом стали **генетические алгоритмы**, и тесно связанное с ними **генетическое программирование** [18, 19]. Если искусственные нейронные сети в своей исходной реализации являлись простейшей моделью нейронной сети головного мозга, то генетические алгоритмы заходят еще дальше, и являются собой предельно упрощенную математическую модель эволюции видов животного мира. В данном подходе в рассмотрение вводится популяция, состоящая из отдельных особей. Каждая из особей в случае генетического алгоритма – это некоторый упорядоченный набор байтов (возможно лишенный какого-либо смысла), а в случае генетического программирования – это кусок программного кода (возможно выполняющий бессмысленные операции, но с единственным ограничением на корректность с точки зрения компилятора используемого языка программирования).

В начале работы алгоритма (на начальном этапе эволюции) создается популяция из случайных особей. Под случайной особью в случае генетического алгоритма подразумевается случайный набор байтов, а в случае генетического программирования – кусок программного кода из случайных операторов и выражений (формально корректных с точки зрения синтаксиса используемого языка программирования и не приводящих к заикливанию при выполнении). Далее на каждом шаге работы алгоритма (на каждом витке эволюции) некоторые (случайно отобранные) особи популяции подвергаются мутациям (mutation) и кроссинговеру (crossover).

Мутация заключается в замене у заданной особи с некоторой вероятностью случайной части байтов или куска кода на новые случайно сгенерированные байты или кусок кода. При кроссинговере случайным образом отбираются две особи из популяции, и с некоторой вероятностью между ними происходит обмен случайными байтами или кус-

ками кода. В итоге, ввиду описанных процедур к концу витка эволюции многие особи популяции изменяются, и некоторые из них приобретают новые полезные с точки зрения их предназначения (качества решения рассматриваемой прикладной задачи) свойства. В конце витка производится соответствующая проверка каждой особи на качество решения рассматриваемой задачи, и затем все особи ранжируются по убыванию их качества.

Так например в случае задачи построения программного кода способного к игре в крестики-нолики, этап проверки может выглядеть следующим образом. Каждой особи (программе) «предлагается» сыграть с эталонной программой или с человеком, и по результату игры соответствующая особь получает +2 очка в случае победы, +1 очко в случае ничьи и -1 очко в случае поражения. Для каждой особи процесс игры может повторяться несколько раз, при этом набираемые очки суммируются. В итоге, особи, показавшие наихудший результат (учитывая специфику алгоритма многие, или даже большинство из них, будут просто случайным кодом, не имеющим никакой ценности), оказываются в конце списка, а особи с лучшим результатом (особи, соответствующие программному коду, случайно получившему модификации, способствующие улучшению качества игры) оказываются в начале списка.

Отметим, что очень красивой модификацией процедуры оценки качества особей является игра особей одной популяции друг с другом, в этом случае по сути отсутствует необходимость в учителе (эталонной программе или человеке), то есть система становится самообучающейся. Еще одним плюсом данной процедуры является возможность проведения большого числа раундов игры особей друг с другом, в отличие от случая, когда, например, игра происходит с человеком, и время одной партии становится лимитирующим фактором. С другой стороны, стоит отметить, что для предотвращения стагнации популяции в целом, все же необходимо периодически производить оценку особей по отношению к внешнему измерителю. Отметим, что похожий подход использовался при обучении глубокой нейронной сети [20], показавшей феноменальный результат при игре в Го. Для улучшения качества сети, периодически производились игры сети с самой собой, что позволяло ей отбрасывать заведомо неудачные стратегии (снижать соответствующие веса). Программа, использующая данную нейронную сеть приобрела широкую популярность в СМИ, так как ей впервые в истории удалось обыграть действующего чемпиона мира по игре Го, при том, что до недавних пор считалось практически невозможным реализовать достаточно серьезного компьютерного противника, способного на равных сражаться с профессионалами, ввиду чрезвычайно сложной структуры игры.

К полученному описанным выше образом списку особей, упорядоченному по убыванию их качества, применяется эволюционный отбор. То есть, в популяцию на следующий виток эволюции отбираются некоторые особи из текущего списка с вероятностью быстро убывающей к концу списка (для особи с плохими результатами таким образом вероятность перейти на следующий виток эволюции оказывается чрезвычайно малой, но не нулевой, по аналогии с реальной эволюцией видов, соответственно для особи в начале списка ве-

роятность перейти на следующий виток эволюции очень высока). Для сохранения «генетического разнообразия», к отобранным описанным образом особям на следующий виток эволюции добавляется также некоторый процент случайно сгенерированных особей (которые, напомним, являются последовательностью случайных бит или куском случайного программного кода).

Описанный метод генетических алгоритмов применялся, в частности, к задачам сегментации и классификации изображений [21]. Существует ряд иных работ, демонстрирующих перспективность данного метода, однако на сегодняшний день для него отсутствует достаточное теоретическое обоснование, и при всей неоспоримой привлекательности метода не существует надежных подходов для корректного выбора параметров расчета (вероятностей мутации и кроссинговера, числа особей в популяции и т.д.). В этой связи мы не будем углубляться в метод генетических алгоритмов, тем не менее подчеркивая, что есть высокая вероятность того, что данный подход, пройдя ряд эволюционных витков, станет мощным и надежным методом решения прикладных задач.

Наиболее перспективным методом машинного обучения и анализа больших объемов данных, на наш взгляд, является **нейросетевой подход**. В частности, на сегодняшний день методы искусственных нейронных сетей (ИНС) уже нашли применение в широком числе практических приложений, связанных с графическими изображениями:

- распознавание символов на изображении [22],
- распознавание объектов на изображении [23],
- классификация и кластеризация изображений [24, 25],
- генерация текстового описания к изображению [26],
- поиск изображений, похожих на данное [27],
- сжатие изображений [28, 29],
- устранение дефектов на изображении [30],
- восстановление целого изображения по его части [31],
- генерация фотoreалистичных изображений [32, 33].

В данной работе для решения практической задачи автоматического распознавания CAPTCHA мы остановимся именно на нейросетевом подходе. Далее в этой главе мы соответственно рассмотрим более подробно данный подход и его основные положения, при этом, сосредоточив основное внимание на полносвязных ИНС прямого распространения (в том числе, глубоких ИНС), оставляя в стороне массу альтернативных реализаций (вариационные автокодировщики, сети Кохонена, ограниченные машины Больцмана и т.д.).

1.3 Нейроны в искусственных нейронных сетях

Для последовательного анализа метода искусственных нейронных сетей (ИНС) мы начнем рассмотрение с основного элемента ИНС – искусственного нейрона или, иначе мы рассмотрим в данном разделе ИНС, состоящую из всего одного нейрона. Отметим, что для искусственных нейронов всех типов часто используют единый термин персептрон (perceptron), восходящий к оригинальной работе Розенблатта [34], являющегося одним из основоположников парадигмы ИНС. Однако изначально (в частности, в оригинальной работе) под персептроном понимался частный случай искусственного нейрона, с бинарным выходом (нейрон на выходе дает 1, в случае если суммарный входной сигнал с учетом весов входов нейрона превышает заданное пороговое значение, и 0 – в противном случае), поэтому далее мы будем использовать более обширный термин «искусственный нейрон», опуская слово «искусственный» в тех местах, где это не приводит к неоднозначности.

ИНС используются для выявления закономерностей и взаимозависимостей в наборах данных, и в частности для приближения зависимости некоторой (выходной) характеристики явления или процесса от набора (обычно неполного) свойств конкретного явления или условий протекания конкретного процесса. Например, пусть для некоторого человека нам известны результаты сданных им выпускных экзаменов: x_0 баллов по математике, x_1 баллов по программированию, ..., x_{n-1} баллов по русскому языку. В данном случае, n – это количество сданных экзаменов, а величины x_0, x_1, \dots, x_{n-1} – это свойства наблюдаемого процесса. Для рассматриваемого примера эти свойства являются натуральными числами, ограниченными максимальным баллом, который можно получить за экзамен, однако для сохранения общности мы будем рассматривать их как набор произвольных вещественных чисел, то есть $x_k \in \mathbb{R}$ при $k = 0, 1, \dots, n - 1$. Для компактности математических выкладок объединим все величины x_0, x_1, \dots, x_{n-1} в вектор⁽⁵⁾ наблюдения \mathbf{x} длины n .

Одной из возможных задач машинного обучения для рассматриваемого примера может быть предсказание стартовой зарплаты y выпускника по результатам сдачи экзаменов, описываемых вектором \mathbf{x} длины n , где, как было указано выше, каждый элемент вектора – это количество баллов, полученных за соответствующий экзамен. Обозначим данную связь в виде функции⁽⁶⁾ f и запишем соответственно $y = f(\mathbf{x})$. Простейшая (не

⁽⁵⁾ Здесь и далее для задания элементов векторов и матриц мы будем использовать нумерацию, начинающуюся с нуля для удобства последующей программной реализации. Так для вектора \mathbf{x} длины n , первый элемент мы будем обозначать как x_0 или $\mathbf{x}[0]$, а последний элемент тогда запишется как x_{n-1} или $\mathbf{x}[n-1]$. Также отметим, что для скалярных величин (чисел) мы будем использовать в работе строчные символы (например, a), для векторов мы будем использовать строчные жирные символы (например, \mathbf{x}), а для матриц – прописные жирные символы (например, \mathbf{A}). В рамках описанной системы обозначений, x_0 – это число (являющееся по смыслу первым элементом вектора), а $\mathbf{x}[0]$ – это вектор, из которого взят первый его элемент.

⁽⁶⁾ Очевидно, что более строго данная связь должна задаваться некоторой вероятностной функцией, так как, например, для рассматриваемого случая, результаты экзамена не являются полным набором переменных, определяющих итоговую зарплату. Однако для упрощения построений мы опустим данный момент, при этом всюду неявно его подразумевая далее в тексте.

тривиальная) функция – это линейная функция

$$y = f(\mathbf{x}) = w_0x_0 + w_1x_1 + \dots + w_{n-1}x_{n-1} + b,$$

где $w_0, w_1, \dots, w_{n-1}, b \in \mathbb{R}$. Подобная запись фактически означает, что зарплата y определяется суммой оценок за экзамены, причем каждому предмету приписывается свой вес w_k при $k = 0, 1, \dots, n - 1$ (так например, если подразумевается работа по технической специальности, то оценкам по математике и программированию может приписываться большой вес $w_0 = w_1 = 1$, а оценке по русскому языку – малый вес $w_{n-1} = 10^{-1}$, тогда в приведенной сумме слагаемое, соответствующее результату экзамена по русскому языку будет вносить малый вклад в результат). В вышеприведенной формуле также была добавлена константа b , которая может иметь смысл базовой части зарплаты, не зависящей от уровня квалификации сотрудника.

Отметим, что если ввести вектор весов \mathbf{w} длины n (такой что $\mathbf{w}[k] = w_k$ при $k = 0, 1, \dots, n - 1$) и использовать вектор наблюдения \mathbf{x} , то последнее выражение можно переписать более компактно с использованием скалярного произведения векторов

$$y = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b,$$

где надстрочным символом T обозначена операция транспонирования.

В реальных приложениях конечно вид функции f неизвестен, и задачей машинного обучения, как раз является построение эффективного приближения этой функции в виде композиции некоторых простых функций. Простейшим вариантом, естественно, оказывается линейная модель (линейная регрессия), то есть предположение, что искомая функция f может быть с достаточной точностью аппроксимирована линейной функцией, соответствующей приведенной выше формуле. При этом входящие в приближение параметры (вектор весов \mathbf{w} и константа b для рассматриваемого случая) должны определяться, исходя из известного набора исторических данных, то есть для рассматриваемого примера для M выпускников, устроившихся на работу, должны быть указаны их оценки за экзамены $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{M-1}$ (где \mathbf{x}_q – это вектор, являющийся набором баллов, полученных q -ым выпускником за каждый из n предметов⁽⁷⁾) и соответствующие реальные стартовые зарплаты y_0, y_1, \dots, y_{M-1} (где y_q – это зарплата в рублях или в валюте q -ого выпускника).

В контексте подхода ИНС обычно выбирается более простая величина в качестве предсказываемой. Например, будем под y понимать величину, характеризующую факт трудоустройства (или факт получения «престижной» работы) выпускника в течение года по окончании учебного заведения, и для возможности математических выкладок положим $y = 1$, если человек получил работу и $y = 0$, если найти работу не удалось. Предсказанием

⁽⁷⁾ Отметим, что \mathbf{x}_q при $q = 0, 1, \dots, M-1$ – это именно вектор, а не элемент вектора, что подчеркивается использованным жирным шрифтом. При этом, величина $\mathbf{x}_q[k]$ при $q = 0, 1, \dots, M-1$ и $k = 0, 1, \dots, n-1$ – это оценка q -ого выпускника за $(k-1)$ -ый предмет.

ИНС (в нашем случае одного нейрона) тогда будет вещественное число a , лежащее в отрезке $[0, 1]$, причем в случае, если a больше или равно 0.5, считаем что ИНС предсказывает удачу в поиске работы, а в противном случае – неудачу.

Как и выше, рассмотрим простейшую линейную модель зависимости от вектора наблюдения или входного вектора \mathbf{x} (input), и назовем взвешенным выходом нейрона (weighted output) величину

$$z = w_0x_0 + w_1x_1 + \dots + w_{n-1}x_{n-1} + b = \mathbf{w}^T\mathbf{x} + b.$$

Элементы вектора \mathbf{w} будем соответственно называть весами нейрона (weights), а число b – смещением (bias), при этом, очевидно, чем больше абсолютное значение конкретного веса, тем большую значимость имеет соответствующая наблюдаемая переменная. Для получения предсказания нейрона в виде числа a (output), лежащего в отрезке $[0, 1]$, подействуем некоторой функцией σ с областью значений в отрезке от нуля до единицы, называемой функцией активации (activation function), которая может, например, представлять из себя сигмоидальную (логистическую) функцию вида

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

на взвешенный выход нейрона z

$$a = \sigma(z) = \sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x} - b}}.$$

Полученная величина a и будет являться предсказанием ИНС (в нашем случае, состоящей из всего одного нейрона).

При создании (программной реализации) искусственных нейронов их веса и смещения обычно инициализируются случайными величинами, и естественно предсказанное нейроном значение a для заданного набора входных данных \mathbf{x} (оценок конкретного выпускника за экзамены в нашем примере) не имеет ни малейшего отношения к правильному значению y (факту трудоустройства). Важнейшим свойством отдельных нейронов, и ИНС в целом, является возможность обучения по историческим данным⁽⁸⁾. Более подробно процесс обучения будет рассмотрен далее в работе после описания ИНС, однако уже сейчас мы приведем его схематическое описание.

Пусть в нашем распоряжении имеется набор из M исторических данных – входные значения $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_{M-1}$ и соответствующие известные правильные значения выходных параметров y_0, y_1, \dots, y_{M-1} (в нашем случае – это оценки за экзамены и реальный результат в виде получения/неполучения работы для каждого из M выпускников, например из

⁽⁸⁾ Более строго, существует несколько подходов к обучению ИНС. В данной работе (в соответствии с ее основной целью – распознаванием символов на графических изображениях) мы рассматриваем только вариант обучения по размеченной выборке исторических данных, называемый также обучением с учителем (supervised learning).

нескольких предыдущих выпусков). Для каждого элемента исторического набора данных q ($q = 0, 1, \dots, M - 1$), то есть для каждого выпускника, подадим на вход нейрона вектор оценок \mathbf{x}_q , получим соответствующий выход нейрона a_q и оценим полную ошибку на всем наборе исторических данных. Оценить ошибку можно с использованием функции стоимости (cost function), которая, например, может иметь вид:

$$C = \frac{1}{2M} \sum_{q=0}^{m-1} (y_q - a_q)^2.$$

Смысл данной величины очевиден. Если для каждого элемента исторического набора данных нейрон дает верный результат ($a_q = y_q$), то функция стоимости равна нулю. Если для некоторых элементов набора ошибка ненулевая, то функция стоимости будет положительной, и ее величина будет тем больше, чем больше количество ошибок и величина каждой из ошибок. При некотором выборе весов и смещения нейрона функция ошибки для заданного набора исторических данных достигнет своего минимального значения (в идеальном случае она станет равной нулю, но на практике это практически недостижимо). При минимальности (или близости к минимуму) функции стоимости говорят, что нейрон (или ИНС, если речь идет про целую сеть) обучен, и если набор исторических данных обладает достаточной репрезентативностью, то при предъявлении затем нейрону нового входного вектора (не входящего в исторические данные) выходное значение нейрона или сети будет близким или даже совпадет с правильным значением.

Под репрезентативностью набора данных обычно подразумевают, что данный набор охватывает все возможные значения входных параметров или, по крайней мере, достаточно равномерно распределен по их области определения. Например, для рассматриваемого нами примера оценок за экзамены, выборка в которой будут присутствовать только отличники или, наоборот, только выпускники с низкой успеваемостью, очевидно, не будет обладать свойством репрезентативности. Так ИНС (или отдельный нейрон), обучившаяся только по историческим данным по ученикам с низкой успеваемостью, безусловно, не сможет дать корректное предсказание для отличника и наоборот.

Отметим, что обычно при работе с ИНС (и со многими другими подходами машинного обучения) набор исторических данных разбивается на две части. Первая большая часть называется обучающим набором данных (training data) и используется для обучения сети в соответствии с кратко описанным выше подходом минимизации функции стоимости (подстройки весов сети). Вторая часть, называемая тестовым набором данных (test data), используется для оценки корректности работы обученной сети (вычисляется доля правильных предсказаний сети на тестовом наборе данных, и полученная величина используется в качестве показателя точности сети). Так же стоит отметить, что во многих приложениях исторические данные разбиваются даже на три части. Две части, как было указано, являются обучающим и тестовым набором, а третья часть, называемая

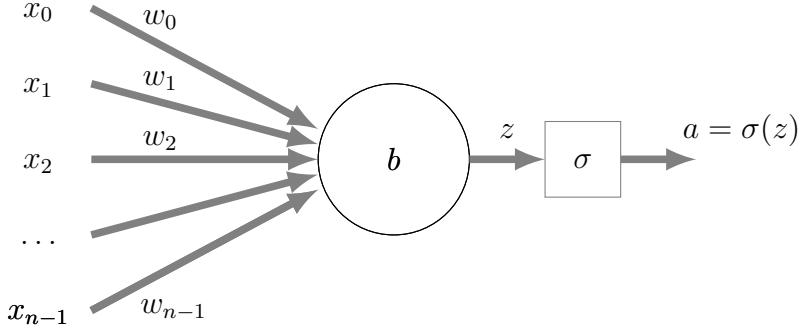


Рис. 1: схематическое изображение искусственного нейрона (для наглядности функция активации вынесена отдельно) и используемые в данной работе обозначения. Входной сигнал нейрона, имеющего n входов, обозначается вектором \mathbf{x} длины n с элементами x_0, x_1, \dots, x_{n-1} . Совокупность весов входов нейрона обозначается как вектор \mathbf{w} длины n с элементами w_0, w_1, \dots, w_{n-1} , смещение нейрона обозначено символом b , взвешенный выход как z ($z = \mathbf{w}^T \mathbf{x} + b$), а функция активации как σ . Результирующий выход нейрона после действия функции активации на взвешенный выход тогда записывается как $a = \sigma(z)$.

валидационным набором данных (validation data), используется для подбора оптимальных значений гиперпараметров сети. Под гиперпараметрами сети понимаются, по сути, все параметры, отличные от подстраиваемых в процессе обучения сети (весов и смещений нейронов). Так гиперпараметрами полносвязной ИНС являются число слоев сети, количество нейронов в каждом из слоев, вид функции активации, вид функции стоимости и т.п. Процесс обучения сети по обучающему набору данных происходит при фиксированных значениях гиперпараметров. Для настройки оптимальных значений гиперпараметров формируется ряд сетей с разными их значениями, полученные сети обучаются на одном и том же обучающем наборе данных, затем на валидационном наборе данных оценивается качество (точность) каждой из сетей, на основе чего делается заключение об оптимальных значениях гиперпараметров.

На рисунке 1 приводится схематическое изображение одного нейрона ИНС с использованием введенных выше обозначений. Нейрон имеет n входов, каждый из которых соответствует одной из наблюдаемых переменных (в нашем случае, полученным баллам за один из экзаменов), и каждому входу приписывается соответствующий вес. Вектор весов нейрона \mathbf{w} , а также смещение b образуют набор параметров нейрона, и задачей обучения сети является фактически выбор их оптимальных значений. Также нейрон характеризуется используемой функцией активации σ (гиперпараметр).

Принцип работы нейрона соответственно можно описать следующим образом.

1. На вход нейрона подается вектор наблюдаемых переменных \mathbf{x} длины n .
2. С использованием текущих значений весов и смещения нейрона (задаваемых вектором \mathbf{w} длины n и числом b), вычисляется взвешенный выход нейрона $z = \mathbf{w}^T \mathbf{x} + b$.
3. Используемая в данном нейроне функция активации σ действует на взвешенный

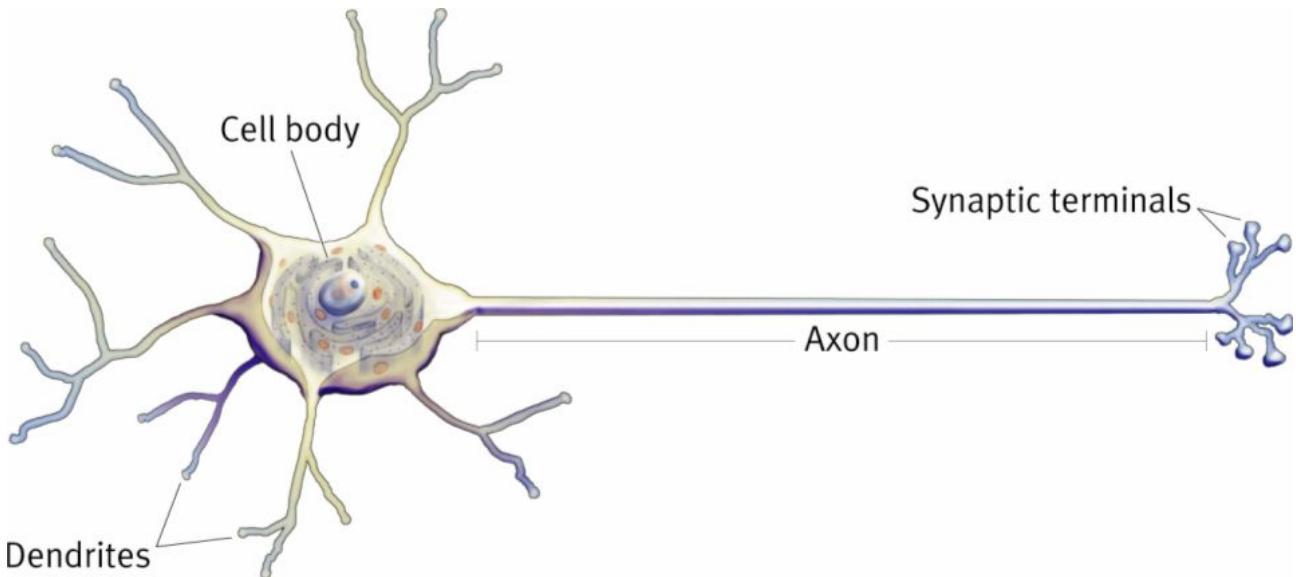


Рис. 2: Схематическое изображение нейрона головного мозга.

выход нейрона, и соответственно получаем окончательный выход нейрона $a = \sigma(z)$ (число, в большинстве реализаций лежащее в отрезке $[0, 1]$), который является предсказанием значения исследуемой величины.

На рисунке 2⁽⁹⁾ изображены основные элементы реального нейрона головного мозга, ставшего прототипом для модели искусственного нейрона в ИНС. В сильно упрощенной форме принцип его действия следующий: дендриты (dendrites) передают телу нейрона (cell body) сигналы от соседних нейронов (вектор входных параметров \mathbf{x} в модели искусственного нейрона) в виде физико-химических импульсов, причем пропускные способности дендритов различаются и могут меняться с течением времени (веса \mathbf{w} в модели искусственного нейрона). Если суммарная величина сигнала (взвешенный выход нейрона z в модели искусственного нейрона) превышает некоторое пороговое значение (роль порогового значения в модели искусственного нейрона играет величина b , однако для удобства математического представления в формулах выше она была учтена в выражении для взвешенного выхода нейрона, соответственно, в данном случае для продолжения аналогии, в модели искусственного нейрона в качестве порогового значения можно указать нуль), то активизируется выход аксона (axon) нейрона (имеется определенная аналогия с функцией активации σ в модели искусственного нейрона), и сигнал передается далее по нейронной сети мозга. При этом, если на некоторый дендрит сигнал от соседних нейронов поступает редко, то происходит снижение его проводящих свойств, а в противном случае – повышение, что является аналогией для классического алгоритма обучения искусственных нейронов и ИНС, который будет рассмотрен далее в работе.

⁽⁹⁾ Данное изображение заимствовано с веб-ресурса <http://bio3520.nicerweb.com/Locked/chap/ch03/neuron.html>.

1.4 Полносвязные искусственные нейронные сети

В данном разделе мы последовательно сформулируем принцип действия полносвязной ИНС, составленной из искусственных нейронов, которые были описаны в предыдущем разделе. Приводимые выкладки будут использованы нами далее в работе при программной реализации ИНС данного типа для решения задачи распознавания символов на графических изображениях, а именно для автоматического распознавания САРТСНА.

На рисунке 3⁽¹⁰⁾ схематически изображена полносвязная нейронная сеть, которая может быть использована для задачи распознавания цифры, представленной на графическом монохромном изображении. На рисунке 4⁽¹¹⁾ в качестве примера приведен набор из рукописных цифр, каждую из которых (после сегментации исходного изображения) можно распознать посредством ИНС. В рассматриваемом примере монохромное изображение размера 28×28 пикселей представляется в виде входного вектора (input vector) \mathbf{x} длины 748 (для этого необходимо матрицу значений интенсивностей пикселей развернуть в вектор, производя последовательный обход строка за строкой, либо столбец за столбцом), в качестве значений содержащего интенсивности соответствующих пикселей. Данный вектор подается на входной слой (input layer) ИНС, содержащий аналогичное число (748) входных нейронов. Входной слой по своей сути является формальным – в нем не выполняется никаких преобразований сигнала, а основное его предназначение – это ввод сигнала в ИНС и передача его на следующий, внутренний слой (hidden layer) ИНС (то есть фактически входной слой составлен из обычных приемников сигнала, или формально – из нейронов с единичным весом, нулевым смещением и тождественной функцией активации).

Каждый нейрон внутреннего слоя ИНС устроен в точности так, как было подробно описано в предыдущем разделе и изображено на рисунке 1. Соответственно, на внутреннем слое происходит преобразование сигнала каждым нейроном слоя, а полученный промежуточный выходной сигнал передается на выходной слой (output layer), где также происходит дополнительное преобразование сигнала нейронами выходного слоя. В простейшем случае выходной слой состоит из таких же нейронов, как и внутренний слой, и после преобразования ими сигнала, полученный результирующий сигнал (output vector), собственно и является результатом работы сети. Для случая распознавания цифр, выходной слой обычно имеет 10 нейронов, соответственно длина выходного вектора \mathbf{y} равна 10, а k -ый ($k = 0, 1, \dots, 9$) его элемент соответствует вероятности того, что на входном изображении представлена цифра k . В рассматриваемом случае в качестве итогового ответа сети можно рассматривать $v = \text{argmax}(\mathbf{y})$, т.е. номер элемента выходного вектора, имеющего наибольшее значение. Этот номер и будет предсказанным (распознанным) сетью числом

⁽¹⁰⁾ Данное изображение заимствовано с веб-ресурса <https://people.cs.pitt.edu/~xianeizhang/notes/NN/NN.html>.

⁽¹¹⁾ Данное изображение заимствовано с веб-ресурса <http://neuralnetworksanddeeplearning.com/chap1.html>.

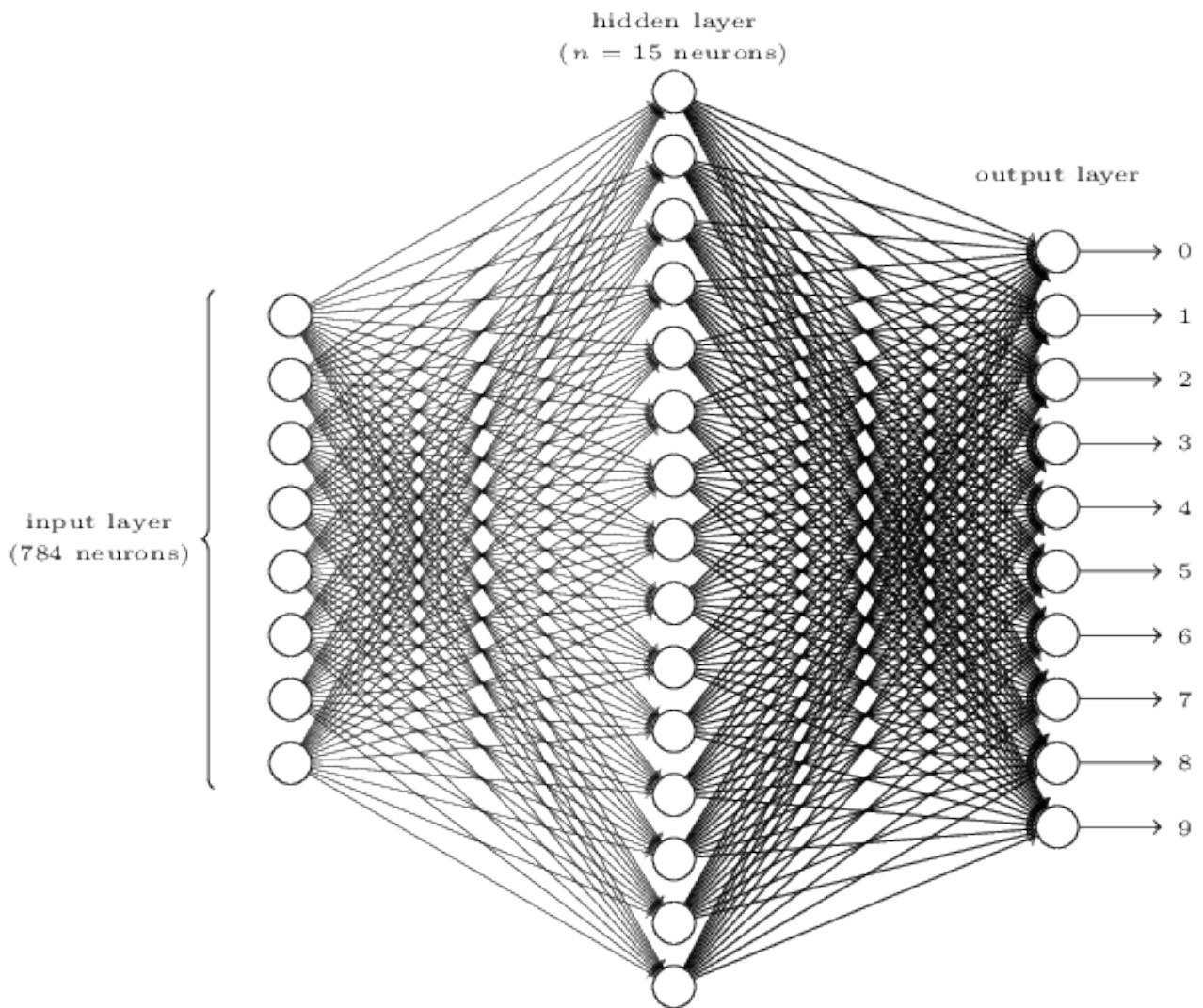


Рис. 3: Схематическое изображение полносвязной ИНС. Входной вектор \mathbf{x} сети для данного примера имеет длину 748 (и соответственно столько же нейронов находится во входном слое ИНС), что соответствует интенсивностям пикселей монохромного изображения размера 28×28 . Число нейронов выходного слоя (и соответственно, размерность выходного вектора \mathbf{y}) равно 10, что соответствует задаче распознавания цифры, представленной на входном изображении. Внутренний слой изображенной ИНС состоит из 15 нейронов (количество нейронов внутреннего слоя – это один из гиперпараметров сети).

на изображении.

Поскольку в приведенной модели ИНС каждый нейрон внутреннего слоя связан с каждым нейроном входного слоя, а каждый нейрон выходного слоя связан с каждым нейроном промежуточного слоя, то такие сети называют полносвязными ИНС (при наличии нескольких внутренних слоев соответствующие сети обычно называют глубокими или глубинными полносвязными ИНС). Отметим, что, например, каждый нейрон выходного слоя имеет число входов, равное числу нейронов внутреннего слоя, при этом входным вектором значений для каждого из нейронов выходного слоя является вектор, составленный из выходов всех нейронов внутреннего слоя (a_0, a_1, \dots, a_{14} для примера ИНС с 15-ю нейронами во внутреннем слое, изображенной на рисунке 3).

Теперь мы введем систему обозначений, которую будем использовать далее в работе, для рассматриваемой модели полносвязной ИНС.

- Пусть L – это число слоев ИНС⁽¹²⁾ (в приведенном выше примере $L = 3$). При этом, для обозначения конкретного слоя сети будем использовать символ l с нумерацией, начинающейся с нуля (соответственно, $l = 0, 1, \dots, L - 1$).
- Количество нейронов в l -ом ($l = 0, 1, \dots, L - 1$) слое будем обозначать как $N^{(l)}$ (в приведенном выше примере $N^{(0)} = 784$, $N^{(1)} = 15$, $N^{(2)} = 10$). При этом, для обозначения конкретного нейрона будем обычно использовать символ i или j с нумерацией, начинающейся с нуля (соответственно, $i = 0, 1, \dots, N^{(l)} - 1$ для l -ого слоя при $l = 0, 1, \dots, L - 1$).
- Вектор, подаваемый на вход ИНС (например, вектор, состоящий из определенным образом упорядоченных значений интенсивности пикселей анализируемого изображения) будем обозначать как \mathbf{x} и называть входом ИНС или входным вектором ИНС (вектором, поданным на вход сети)⁽¹³⁾. Отметим, что длина вектора \mathbf{x} и число нейронов входного слоя $N^{(0)}$ должны в точности совпадать.
- Смещение j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 0, 1, \dots, L - 1$) будем обозначать как $b_j^{(l)}$, а соответствующие упорядоченные значения для всех нейронов l -ого слоя объединим в вектор $\mathbf{b}^{(l)}$ длины $N^{(l)}$.
- Для j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 1, 2, \dots, L - 1$, при этом входной слой не рассматривается) определим i -ый вес или вес его i -ого входа как $w_{ji}^{(l)}$ (подчеркнем, что i -ый вес является выходом i -ого нейрона ($l - 1$)-ого слоя, соответственно $i = 0, 1, \dots, N^{(l-1)} - 1$). Веса всех нейронов l -ого слоя сети ($l = 1, 2, \dots, L - 1$) для удобства математических выкладок соберем в матрицу $\mathbf{W}^{(l)}$ размера $N^{(l)} \times N^{(l-1)}$, в которой $\mathbf{W}^{(l)}[j, i] = w_{ji}^{(l)}$, то есть элемент матрицы на пересечении j -ой строки и i -ого столбца – это i -ый вес j -ого нейрона l -ого слоя.
- Взвешенный выход j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 0, 1, \dots, L - 1$) будем обозначать как $z_j^{(l)}$, а соответствующие упорядоченные значения для всех нейронов l -ого слоя объединим в вектор $\mathbf{z}^{(l)}$ длины $N^{(l)}$.
- Функцию активации нейронов l -ого слоя ($l = 0, 1, \dots, L - 1$) будем обозначать как $\sigma^{(l)}$ (все нейроны в одном слое всегда имеют одну и ту же функцию активации), а

⁽¹²⁾ Формально при количестве слоев $L > 3$, то есть при наличии более чем одного внутреннего слоя, ИНС является глубокой (deep). Однако обычно данный термин применяют для сетей с числом слоев значительно больше 3-х, например, 10 слоев.

⁽¹³⁾ Мы не вводим специальных обозначений для входных векторов нейронов внутренних слоев и нейронов выходного слоя, поскольку входными векторами нейронов l -ого слоя ($l = 1, 2, \dots, L - 1$) являются выходные векторы предыдущего, ($l - 1$)-ого слоя.



Рис. 4: Пример изображения с рукописными цифрами, каждая из которых после сегментации исходного изображения на отдельные символы может быть распознана посредством ИНС.

для случая использования одинаковой функции активации для всех слоев сети будем использовать обозначение σ , то есть опускать индекс, соответствующий номеру слоя.

- Выход j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 0, 1, \dots, L - 1$) будем обозначать как $a_j^{(l)}$, а соответствующие упорядоченные значения для всех нейронов l -ого слоя объединим в вектор $\mathbf{a}^{(l)}$ длины $N^{(l)}$. Выход последнего слоя (выходного слоя) $\mathbf{a}^{(L-1)}$ является результатом работы (предсказанием) ИНС и называется выходом ИНС или выходным вектором ИНС.
- В случае, если для данного входа ИНС (вектора \mathbf{x}) известно правильное значение моделируемого свойства (например, цифра, представленная на изображении), то выход сети, соответствующий этому правильному значению, будем обозначать \mathbf{y} и называть желаемым выходом сети⁽¹⁴⁾.
- Функцию $C_{\mathbf{x}}$, зависящую от выхода ИНС $\mathbf{a}^{(L-1)}$ (а значит от архитектуры сети, в частности, от значений весов и смещений всех нейронов сети) и желаемого выхода ИНС \mathbf{y} для данного значения входа ИНС \mathbf{x} будем называть функцией стоимости отдельного измерения.
- Если имеется обучающая выборка размера M , то есть, M пар входных векторов \mathbf{x}_q и соответствующих желаемых выходов \mathbf{y}_q ($q = 0, 1, \dots, M - 1$), то функцией стоимости будем называть функцию вида

$$C = \frac{1}{M} \sum_{q=0}^{M-1} C_{\mathbf{x}_q},$$

где $C_{\mathbf{x}_q}$ – это функция стоимости отдельного измерения, соответствующего q -ому элементу выборки.

⁽¹⁴⁾ Например, если известно, что на изображении представлен нуль, то желаемым выходом ИНС с десятью нейронами в выходном слое будет вектор длины 10, в котором первый элемент равен единице, а остальные являются нулями. Очевидно, данный вектор будет соответствовать предсказанию сети, отвечающему стопроцентной вероятности того, что на изображении представлен нуль.

Отметим, что смещения и функция активации для нейронов входного слоя ввиду формальной его структуры не имеют прямого смысла, однако для удобства будем полагать, что вектор смещений нейронов входного слоя – это нулевой вектор, а функция активации – это тождественная функция. Взвешенный выход и (итоговый) выход входного слоя соответственно совпадают со входным вектором сети. При этом, очевидно, что введенная выше матрица весов явно учитывает формальный характер входного слоя ИНС.

Учитывая модель искусственного нейрона, представленную в предыдущем разделе, можем записать для взвешенного выхода j -ого нейрона l -ого слоя

$$z_j^{(l)} = \sum_{i=0}^{N^{(l-1)}-1} w_{ji}^{(l)} a_i^{(l-1)} + b_j^{(l)}, \quad j = 0, 1, \dots, N^{(l)} - 1, \quad l = 1, 2, \dots, L - 1, \quad (1)$$

а для взвешенного выхода первого слоя ($l = 0$) с учетом его формального характера

$$z_j^{(0)} = x_j, \quad j = 0, 1, \dots, N^{(0)} - 1. \quad (2)$$

Отметим, что в формуле (1) учтено, что i -ый вход любого нейрона l -ого слоя является выходом i -ого нейрона $(l - 1)$ -ого слоя.

Формулы (1) и (2) могут быть переписаны более компактно в векторном виде

$$\mathbf{z}^{(0)} = \mathbf{x}, \quad \mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad l = 1, 2, \dots, L - 1, \quad (3)$$

где использована матрица весов l -ого слоя, введенная выше, и соответствующее суммирование заменено на эквивалентную операцию матрично-векторного произведения.

Выход (итоговый) нейронов l -ого слоя можем получить, подействовав функцией активации на соответствующий взвешенный выход. В векторной форме можем записать⁽¹⁵⁾

$$\mathbf{a}^{(0)} = \mathbf{x}, \quad \mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{z}^{(l)}) = \sigma^{(l)}(\mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, 2, \dots, L - 1. \quad (4)$$

Таким образом, математически процесс распространения сигнала в ИНС описывается следующим образом.

1. На вход сети подается входной вектор \mathbf{x} , имеющий длину, равную числу нейронов первого (входного) слоя.
2. Взвешенный выход и выход (итоговый) входного слоя сети задаются как

$$\mathbf{z}^{(0)} = \mathbf{x}, \quad \mathbf{a}^{(0)} = \mathbf{x}.$$

⁽¹⁵⁾ В данной формуле функция активации действует на вектор, соответственно результат является также вектором, каждый элемент которого – это результат действия функции активации на соответствующий элемент векторного аргумента.

3. Для каждого из последующих слоев сети (в цикле) вычисляются взвешенный выход и выход (итоговый) на основе выхода предыдущего слоя

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{z}^{(l)}), \quad l = 1, 2, \dots, L-1.$$

4. Выход последнего слоя (выходного слоя) $\mathbf{a}^{(L-1)}$ является результатом работы (предсказанием) ИНС или выходным вектором ИНС.
5. Если для текущего входного вектора \mathbf{x} известен желаемый выход сети \mathbf{y} (элемент данных обучающей выборки), то может быть также вычислена функция стоимости отдельного измерения C_x , которая затем используется для обучения (подстройки весов и смещений) ИНС.

Например, для задачи распознавания цифры на изображении, входной вектор \mathbf{x} – это интенсивности пикселей изображения, а результат работы сети – это вектор $\mathbf{a}^{(L-1)}$, максимальный элемент которого может считаться предсказываемым числом на изображении. Отметим, что если известно реально представленное на изображении число (то есть, рассматривается элемент обучающей выборки), то составляется соответствующий вектор желаемого выхода \mathbf{y} длины 10 с единственным ненулевым (и равным единице) элементом в позиции, соответствующей числу, представленному на изображении. С использованием выходного вектора ИНС $\mathbf{a}^{(L-1)}$ и желаемого выходного вектора \mathbf{y} , может быть вычислена функция стоимости (отдельного измерения) C_x для данного элемента данных, с использованием которой может быть проведено обучение сети, то есть, коррекция значений весов и смещений всех или некоторых нейронов сети. Одним из популярных выборов функции стоимости является, например, квадратичная функция вида

$$C_x = \frac{1}{2} \|\mathbf{y} - \mathbf{a}^{(L-1)}\|^2 = \frac{1}{2} \sum_{j=0}^{N^{(L-1)}-1} (\mathbf{y}[j] - \mathbf{a}^{(L-1)}[j])^2. \quad (5)$$

1.5 Обучение искусственных нейронных сетей

Описанный в предыдущем разделе процесс распространения сигнала в полносвязной ИНС позволяет получить предсказание сети для данного входного вектора и заданных параметров и гиперпараметров сети. Параметрами сети являются веса и смещения каждого из ее нейронов. Соответственно, гиперпараметры – это количество слоев сети, число нейронов в каждом из слоев, вид функции активации для каждого из слоев, вид функции стоимости, а также вводимые далее в данном разделе параметр скорости обучения, размер подвыборки для обучения методом стохастического градиентного спуска и количество эпох обучения.

При создании ИНС ее параметры инициализируются случайными значениями (с использованием определенных эвристик выбора параметров соответствующих случайных распределений), и предсказание такой (необученной) сети не будет соответствовать правильным (желаемым) значениям. Таким образом, основной задачей является обучение ИНС, то есть выбор по определенной процедуре таких значений ее параметров (а также, в идеале, и гиперпараметров), что предсказания сети с большой вероятностью являются верными.

Как уже указывалось ранее в работе, для обучения сети необходим набор размеченных данных, то есть M -пар вида входной вектор / желаемый выходной вектор (при распознавании цифр на изображении каждый входной вектор – это упорядоченные интенсивности пикселей изображения, а соответствующий выходной вектор – это вектор длины 10, все элементы которого нулевые, кроме элемента, номер которого совпадает с представленной на изображении цифрой). Набор размеченных данных обычно разбивается на три части: обучающий набор данных (training data) с числом элементов $M^{(trn)}$, валидационный набор данных (validation data) с числом элементов $M^{(vld)}$ и тестовый набор данных (test data) с числом элементов $M^{(tst)}$. При этом, очевидно, выполняется

$$M^{(trn)} + M^{(vld)} + M^{(tst)} = M.$$

В простейшем случае, для каждого элемента обучающего набора данных, то есть пары

$$(\mathbf{x}_q, y_q), \quad q = 0, 1, \dots, M^{(trn)} - 1,$$

где \mathbf{x}_q – это входной вектор и y_q – это желаемый выход ИНС, вычисляется выход ИНС $\mathbf{a}^{(L-1)}$, а затем функция стоимости отдельного измерения $C_{\mathbf{x}_q}$, которая, очевидно, зависит от текущих значений весов и смещений нейронов сети.

Вычислив соответствующие функции стоимости отдельных измерений для всех эле-

ментов обучающего набора, можно построить итоговую функцию стоимости

$$C = \frac{1}{M^{(trn)}} \sum_{q=0}^{M^{(trn)}-1} C_{\mathbf{x}_q}, \quad (6)$$

величина которой пропорциональна ошибке сети. Процесс обучения (настройки параметров сети) соответственно заключается в выборе таких значений весов и смещений нейронов ИНС, при которых функция стоимости (6) достигнет минимума. То есть, фактически функция стоимости из формулы (6) должна быть рассмотрена как функция весов и смещений всех нейронов сети, и для такой функции многомерного аргумента должен быть найден локальный или глобальный минимум. Соответствующие значения весов и смещений, обеспечивающие этот минимум, и будут оптимальными значениями параметров сети, при которых она дает максимальную точность.

Явная минимизация функции стоимости является для большинства практически значимых случаев невозможной, поэтому обычно используют итерационный метод градиентного спуска (gradient descent) [35]. Данный метод применим для функций многих аргументов. Для пояснения сути метода, рассмотрим некоторую скалярную функцию $f = f(\mathbf{x})$ от N -мерного (векторного) аргумента $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]^T$. Запишем приращение функции в некоторой многомерной точке \mathbf{x} (формулу Тейлора) с точностью первого порядка

$$f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + \frac{\partial f}{\partial x_0} \Delta x_0 + \frac{\partial f}{\partial x_1} \Delta x_1 + \dots + \frac{\partial f}{\partial x_{N-1}} \Delta x_{N-1}, \quad (7)$$

где $\Delta\mathbf{x} = (x_0, x_1, \dots, x_{N-1})^T$ – это малое приращение многомерного аргумента. Вводя вектор градиента функции

$$\nabla f = \frac{\partial f}{\partial \mathbf{x}} = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_{N-1}} \right)^T,$$

и приращение функции

$$\Delta f = f(\mathbf{x} + \Delta\mathbf{x}) - f(\mathbf{x}),$$

можем переписать формулу (7) как

$$\Delta f \approx (\nabla f)^T \Delta\mathbf{x}. \quad (8)$$

Данная формула отражает практически очевидный факт: приращение функции приблизительно пропорционально приращению аргумента, причем коэффициент пропорциональности – это градиент функции в соответствующей точке (в случае одномерного аргумента, градиент заменился бы на соответствующую производную). Отметим, что для корректности формулы (8) необходима непрерывность вторых частных производных функции (только в этом случае правомерно разложение (7)), а также малость приращения аргу-

мента $\Delta \mathbf{x}$.

Формула (8) лежит в основе метода градиентного спуска. Зафиксируем некоторое значение многомерного аргумента функции \mathbf{x} (например, возьмем некоторое случайное значение из области определения) и вычислим градиент функции в данной точке $\nabla f(\mathbf{x})$. Дадим теперь приращение аргументу функции в направлении противоположном градиенту

$$\Delta \mathbf{x} = -\eta \nabla f(\mathbf{x}), \quad (9)$$

где положительная величина η называется скоростью обучения (learning rate). Отметим, что величина η должна быть достаточно малой для выполнения условия малости приращения аргумента и соответственно применимости разложения (7).

Объединяя формулы (8) и (9), получим для приращения функции в точке \mathbf{x}

$$\Delta f \approx -\eta (\nabla f)^T \nabla f. \quad (10)$$

Очевидно, что приращение функции согласно формуле (10) (при выполнении описанных выше условий ее применимости) является отрицательным (или, в редких случаях, нулевым). Действительно, в правой части данной формулы стоит скалярный квадрат вектора градиента, который является неотрицательным; скорость обучения, которая положительная по определению; соответственно знак «-» определяет итоговый знак выражения.

Из вышеприведенных соображений напрямую следует компактная формулировка метода градиентного спуска. Для минимизации функции многих переменных следует взять некоторую случайную точку (значение многомерной переменной), вычислить градиент функции в данной точке и изменить значение аргумента на величину, определяемую формулой (9) (то есть, в направлении, противоположном направлению градиента). Затем для нового значения аргумента следует вновь вычислить градиент и вновь обновить значение аргумента. Данный шаг повторяется в цикле до выполнения некоторого зафиксированного условия сходимости, например, достаточной малости изменения функции на очередном шаге или достижения максимального числа шагов. При этом, при выполнении условий на вторые производные функции и на малость скорости обучения, гарантируется, что на каждом шаге значение функции будет становиться все ближе к минимальному, поскольку на каждом шаге значение функции убывает в соответствии с (10).

Особо подчеркнем, что с одной стороны, чем меньше параметр скорости обучения η , тем устойчивее и точнее метод градиентного спуска, с другой стороны, чем большие значение данного параметра, тем быстрее работает алгоритм. Таким образом, при практической реализации ИНС особенно важным является удачный выбор значения параметра скорости обучения (который, фактически является гиперпараметром сети), обеспечивающий одновременно и достаточно точное, и достаточно быстрое обучение сети. Также отметим, что в нейросетевом подходе каждый шаг обучения (точнее, каждый полный проход обучающего

набора данных) называют эпохой (epoch) обучения, при этом для задаваемого числа эпох обучения, которое соответственно также является гиперпараметром сети, будем использовать обозначение $M^{(ep)}$.

Описанный метод градиентного спуска может быть, в принципе, применен для обучения ИНС. Мы фиксируем некоторые случайные значения весов и смещений нейронов сети. Для каждого из элементов обучающей выборки мы вычисляем выход сети и соответствующую функцию стоимости отдельного измерения, а затем вычисляем функцию стоимости по формуле (6). Далее находим градиент функции стоимости и изменяем веса и смещения сети на малую величину, в направлении, противоположном градиенту с некоторой скоростью обучения η . Для обновленных значений весов и смещений (на очередной эпохе обучения) мы вновь вычисляем функции стоимости отдельных измерений, затем новое значение функции стоимости и ее градиента, и вновь обновляем веса и смещения. Данный процесс повторяется до тех пор, пока не будут получены веса и смещения сети, при которых функция стоимости достаточно близка к нулю, то есть, пока сеть не научится давать правильные предсказания на обучающем наборе данных или пока не будет достигнуто максимальное число эпох обучения $M^{(ep)}$.

Однако практическая реализация приведенной схемы обучения на основе метода градиентного спуска практически нереальна, поскольку для каждого шага градиентного спуска необходимо заново вычислять выход сети для каждого из элементов обучающего набора, а при крупных обучающих выборках (например, $M^{(trn)} \approx 10^6$), которые применяются в реальных приложениях для обеспечения репрезентативности, это становится вычислительно сложным.

Используемое на сегодняшний день решение данной проблемы – это замена классического метода градиентного спуска на его стохастический вариант – стохастический метод градиентного спуска (stochastic gradient descent) [36]. Основная идея здесь довольно простая: мы проводим минимизацию функции стоимости согласно описанному выше методу градиентного спуска, однако на каждом его шаге (на каждой эпохе обучения) для построения функции стоимости мы вычисляем функции стоимости отдельных измерений не для всех элементов обучающей выборки, а только для нескольких $M^{(mb)}$ случайно отобранных элементов ($M^{(mb)} \ll M^{(trn)}$), называемых подвыборкой (mini-batch). Соответственно, формула (6) для функции стоимости заменится на

$$C \approx \frac{1}{M^{(mb)}} \sum_{q=0}^{M^{(mb)}-1} C_{x_{r(q)}}, \quad (11)$$

где $r(q)$ – это случайное целое неотрицательное число, такое что $0 \leq r(q) < M^{(trn)}$. То есть, из всего тестового набора данных размера $M^{(trn)}$ отбирается $M^{(mb)}$ случайных элементов, по которым строится приближение функции стоимости, причем на каждой эпохе обучения отбор случайных элементов производится заново. В практических приложениях метод

стохастического градиентного спуска в большинстве случаев сходится к минимуму, при этом вычислительная сложность относительно обычного градиентного спуска существенно снижается.

Отметим, что крайним случаем метода стохастического градиентного спуска является построение подвыборок размера 1, то есть $M^{(mb)} = 1$. В данном случае на каждой эпохе обучения отбирается только один случайный элемент из обучающего набора данных, и соответственно функция стоимости (11) совпадает с функцией стоимости отдельного измерения. Вычисленная таким образом функция стоимости используется для обновления весов и смещений нейронов сети. Поскольку в данном варианте метода фактически не требуется знания всего массива обучающей выборки единовременно (сети просто подается на вход один за другим обучающий примеры, и на каждом примере сеть адаптирует параметры), то такое обучение часто называют обучением на лету. Например, сети может быть последовательно передано 10 обучающих примеров, на каждом из которых она «немного обучится», далее сеть используется для работы (предсказаний), а затем, например, сети передается еще один обучающий пример, на котором она еще «немного обучится». Даный подход является очень привлекательным для практических приложений, однако он не всегда приводит к качественному обучению ИНС.

Последним шагом для реализации практического метода обучения ИНС является построение эффективного метода вычисления градиента функции стоимости для возможности использования стохастического метода градиентного спуска, описанного выше. Отметим, что существует очевидный метод построения градиента функции на основе конечных разностей. Так для функции $f = f(\mathbf{x})$ от N -мерного (векторного) аргумента $\mathbf{x} = [x_0, x_1, \dots, x_{N-1}]^T$, приводимой выше в качестве примера, градиент в точке \mathbf{x} может быть вычислен следующим образом. Мы даем малое приращение h первой координате вектора $\Delta_0\mathbf{x} = [x_0 + h, x_1, \dots, x_{N-1}]^T$ и вычисляем

$$\frac{\partial f}{\partial x_0} \approx \frac{f(\mathbf{x} + \Delta_0\mathbf{x}) - f(\mathbf{x})}{h},$$

затем мы даем приращение только второй координате $\Delta_1\mathbf{x} = [x_0, x_1 + h, \dots, x_{N-1}]^T$ и вычисляем

$$\frac{\partial f}{\partial x_1} \approx \frac{f(\mathbf{x} + \Delta_1\mathbf{x}) - f(\mathbf{x})}{h},$$

и аналогично для остальных координат. Существенной проблемой данного метода является чрезвычайно высокая вычислительная сложность: для каждого элемента обучающей выборки необходимо произвести вычисления по явным формулам, представленным выше, для каждого веса и каждого смещения для всех нейронов сети.

Данная проблема была решена только в конце 80-х годов прошлого века, когда в работе [11] был предложен (частично переоткрытый) алгоритм обратного распространения ошибки (backpropagation) в ИНС, позволяющий быстро вычислять частные производные

функции стоимости по весам и смещениям нейронов сети.

Далее мы сформулируем алгоритм обратного распространения ошибки для полносвязанной ИНС, представленной в предыдущем разделе, с использованием соответствующих обозначений, подробно описанных там же. Отметим, что функция стоимости C на обучающей выборке – это среднее по выборке от функции стоимости отдельного измерения C_{x_q} при $q = 0, 1, \dots, M^{(trn)} - 1$ (в соответствии с формулой (6) или приближенной формулой (11)). При этом функция стоимости отдельного измерения C_{x_q} (определенная, например, по формуле (5) для случая выбора квадратичной функции стоимости) является мерой отклонения выхода ИНС от желаемого (правильного) значения. Для заданного элемента обучающей выборки и заданной архитектуры сети C_{x_q} зависит только от весов и смещений всех нейронов ИНС, то есть

$$C_{x_q} = C_{x_q}(\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}),$$

где $\mathbf{W}^{(l)}$ ($l = 1, 2, \dots, L - 1$) – это матрица весов l -ого слоя, а $\mathbf{b}^{(l)}$ ($l = 1, 2, \dots, L - 1$) – это вектор смещений l -ого слоя (для компактности записи мы объединили веса и смещения нейронов каждого слоя в соответствующие матрицы весов и векторы смещений). Соответственно для обучения ИНС по методу (стохастического) градиентного спуска нам необходимы выражения для частных производных функции стоимости по всем весам

$$\frac{\partial C_{x_q}}{\partial w_{ji}^{(l)}}, \quad j = 0, 1, \dots, N^{(l)} - 1, \quad i = 0, 1, \dots, N^{(l-1)} - 1, \quad l = 1, 2, \dots, L - 1,$$

и по всем смещениям

$$\frac{\partial C_{x_q}}{\partial b_j^{(l)}}, \quad j = 0, 1, \dots, N^{(l)} - 1, \quad l = 1, 2, \dots, L - 1,$$

нейронов ИНС (для входного слоя, соответствующего $l = 0$, веса отсутствуют, а смещения являются нулевыми).

Дополнительно для удобства выкладок мы введем также формальную величину ошибки нейрона⁽¹⁶⁾ по формуле

$$\delta_j^{(l)} = \frac{\partial C_x}{\partial z_j^{(l)}}, \quad j = 0, 1, \dots, N^{(l)} - 1, \quad l = 1, 2, \dots, L - 1,$$

где C_x – это функция стоимости ИНС на конкретном элементе обучающей выборки, а $z_j^{(l)}$ – это взвешенный выход j -го нейрона l -ого слоя. Ошибки нейронов одного слоя можно

⁽¹⁶⁾ Производная здесь понимается в смысле производной по сложной функции: функция стоимости зависит от весов и смещений нейронов ИНС, при этом взвешенные выходы нейронов, по которым производится дифференцирование, также зависят от весов и смещений.

объединить в соответствующий вектор ошибки нейронов слоя следующим образом

$$\boldsymbol{\delta}^{(l)} = \frac{\partial C_{\mathbf{x}}}{\partial \mathbf{z}^{(l)}}, \quad l = 1, 2, \dots, L - 1,$$

где производная по вектору обозначает вектор, каждая из компонент которого – это соответствующая частная производная числителя по соответствующей компоненте вектора в знаменателе.

Теорема 1. Для полносвязной искусственной нейронной сети, имеющей L слоев, состоящих из $N^{(0)}, N^{(1)}, \dots, N^{(L-1)}$ нейронов соответственно, частная производная функции стоимости отдельного измерения $C_{\mathbf{x}}$ по смещению j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 1, 2, \dots, L - 1$) может быть записана как

$$\frac{\partial C_{\mathbf{x}}}{\partial b_j^{(l)}} = \delta_j^{(l)}, \quad (12)$$

а частная производная по i -ому весу ($i = 0, 1, \dots, N^{(l-1)} - 1$) j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 1, 2, \dots, L - 1$) как

$$\frac{\partial C_{\mathbf{x}}}{\partial w_{ji}^{(l)}} = a_i^{(l-1)} \delta_j^{(l)}, \quad (13)$$

где $\delta_j^{(l)}$ – это величина ошибки j -ого нейрона ($j = 0, 1, \dots, N^{(l)} - 1$) l -ого слоя ($l = 1, 2, \dots, L - 1$), которая может быть определена по следующим векторным рекуррентным формулам

$$\begin{aligned} \boldsymbol{\delta}^{(L-1)} &= \frac{\partial C_{\mathbf{x}}}{\partial \mathbf{a}^{(L-1)}} \odot \sigma'(\mathbf{z}^{(L-1)}), \\ \boldsymbol{\delta}^{(l)} &= \left((\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \right) \odot \sigma'(\mathbf{z}^{(l)}), \quad l = L - 2, L - 3, \dots, 1, \end{aligned} \quad (14)$$

где « \odot » – это операция поэлементного (Адамарова) перемножения векторов, штраф у функции активации⁽¹⁷⁾ соответствует полной производной по ее аргументу, а оставльные величины соответствуют системе обозначений, введенной в разделе 1.4.

Доказательство. Докажем истинность первой формулы в (14). Согласно определению ошибки нейронов, а также правилу дифференцирования сложной функции имеем

$$\delta_j^{(L-1)} = \frac{\partial C_{\mathbf{x}}}{\partial z_j^{(L-1)}} = \sum_{i=0}^{N^{(L-1)}} \frac{\partial C_{\mathbf{x}}}{\partial a_i^{(L-1)}} \frac{\partial a_i^{(L-1)}}{\partial z_j^{(L-1)}}, \quad j = 0, 1, \dots, N^{(L-1)} - 1.$$

Поскольку $a_i^{(L-1)} = \sigma(z_i^{(L-1)})$, то только одно слагаемое в последней формуле отлично от

⁽¹⁷⁾ Для компактности формул мы опускаем номер слоя (l) у функций активации.

нуля, и

$$\delta_j^{(L-1)} = \frac{\partial C_x}{\partial a_j^{(L-1)}} \frac{\partial a_j^{(L-1)}}{\partial z_j^{(L-1)}} = \frac{\partial C_x}{\partial a_j^{(L-1)}} \sigma'(z_j^{(L-1)}).$$

Соответственно в векторной форме можем переписать как

$$\boldsymbol{\delta}_j^{(L-1)} = \frac{\partial C_x}{\partial \mathbf{a}^{(L-1)}} \sigma'(\mathbf{z}^{(L-1)}),$$

то есть получаем в точности первую формулу из (14).

Теперь докажем рекуррентную формулу (вторую формулу) из (14). Вновь воспользуемся определением ошибки нейронов и правилом дифференцирования сложной функции с учетом того, что взвешенный выход любого нейрона $(l+1)$ -ого слоя определяется (является функцией) взвешенных выходов всех нейронов l -ого слоя

$$\delta_j^{(l)} = \frac{\partial C_x}{\partial z_j^{(l)}} = \sum_{i=0}^{N^{(l+1)}} \frac{\partial C_x}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}}, \quad j = 0, 1, \dots, N^{(l)} - 1, \quad l = 1, 2, \dots, L - 2. \quad (15)$$

Производная от функции стоимости под знаком суммы – это ошибка соответствующего нейрона $(l+1)$ -ого слоя. Учитывая также, что по определению взвешенного выхода

$$z_i^{(l+1)} = \sum_{k=0}^{N^{(l)}-1} w_{ik}^{(l+1)} a_k^{(l)} + b_i^{(l+1)} = \sum_{k=0}^{N^{(l)}-1} w_{ik}^{(l+1)} \sigma(z_k^{(l)}) + b_i^{(l+1)},$$

то есть

$$\frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}} = w_{ij}^{(l+1)} \sigma'(z_j^{(l)}),$$

где штрих у функции активации означает полную производную по аргументу, можем переписать (15) в следующей форме

$$\delta_j^{(l)} = \sum_{i=0}^{N^{(l+1)}-1} \delta_i^{(l+1)} w_{ij}^{(l+1)} \sigma'(z_j^{(l)}) = \left(\sum_{i=0}^{N^{(l+1)}-1} w_{ij}^{(l+1)} \delta_i^{(l+1)} \right) \sigma'(z_j^{(l)}).$$

Поскольку выражение в скобках – это соответствующий элемент матрично-векторного произведения для матрицы весов $(l+1)$ -ого слоя и вектора ошибок нейронов $(l+1)$ -ого слоя, то, в итоге, можем переписать данную формулу в векторной форме как

$$\boldsymbol{\delta}^{(l)} = \left((\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \right) \odot \sigma'(\mathbf{z}^{(l)}),$$

что в точности соответствует второй формуле из (14).

Для доказательства формулы (12) перепишем производную по смещению нейрона через производную по соответствующему взвешенному выходу, используя правило диффе-

ренцирования сложной функции

$$\frac{\partial C_x}{\partial b_j^{(l)}} = \frac{\partial C_x}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial b_j^{(l)}},$$

$$j = 0, 1, \dots, N^{(l)} - 1, \quad l = 1, 2, \dots, L - 1,$$

и учитывая определение ошибки нейрона, а также то что по определению

$$z_i^{(l)} = \sum_{k=0}^{N^{(l)}-1} w_{ik}^{(l)} a_k^{(l-1)} + b_i^{(l)}, \quad (16)$$

получаем

$$\frac{\partial C_x}{\partial b_j^{(l)}} = \delta_j^{(l)} \times 1,$$

то есть, искомую формулу (12).

И наконец, для доказательства формулы (13) мы, по аналогии, переписываем частную производную по весу нейрона через частную производную по соответствующему взвешенному выходу, используя правило дифференцирования сложной функции

$$\frac{\partial C_x}{\partial w_{ji}^{(l)}} = \frac{\partial C_x}{\partial z_j^{(l)}} \frac{\partial z_j^{(l)}}{\partial w_{ji}^{(l)}},$$

$$j = 0, 1, \dots, N^{(l)} - 1, \quad i = 0, 1, \dots, N^{(l-1)} - 1, \quad l = 1, 2, \dots, L - 1,$$

затем, как и выше, мы учитываем определение ошибки нейрона и выражение (16), и, в итоге, получаем

$$\frac{\partial C_x}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} a_i^{(l-1)},$$

то есть, формулу (13). \square

Отметим, что при получении предсказания сети (выходного вектора) по заданному входному вектору (алгоритм прямого распространения сигнала по ИНС), как было описано в предыдущем разделе, используется «прямая» рекуррентная формула, то есть, последовательно обходятся слои сети, начиная с первого, и выходной вектор одного слоя является входным вектором для следующего слоя. При этом, согласно приведенной выше теореме, для получения частных производных функции стоимости используется «обратная» рекуррентная формула, то есть слои сети обходятся в цикле, начиная с последнего, и для каждого из них вычисляется вектор ошибки нейронов по формуле (14). В этой связи данный алгоритм получил название «алгоритм обратного распространения ошибки».

Доказанная теорема является эффективным инструментом обучения ИНС по заданным обучающим наборам данных в соответствии с методом стохастического градиентного спуска. С использованием данной теоремы, а также описания процесса распространения

сигнала в ИНС, представленного в предыдущем разделе, мы можем теперь схематически описать процесс обучения ИНС, который будет использован нами далее в работе для создания автоматизированной системы распознавания САРТСНА на основе известных данных по уже распознанным изображениям.

- На вход сети подается входной вектор \mathbf{x} , имеющий длину, равную числу нейронов первого (входного) слоя. Взвешенный выход и выход (итоговый) входного слоя сети задаются как

$$\mathbf{z}^{(0)} = \mathbf{x}, \quad \mathbf{a}^{(0)} = \mathbf{x}.$$

- Для каждого из последующих слоев сети вычисляем и сохраняем в памяти взвешенный выход и выход по рекуррентным формулам

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)} \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)}), \quad l = 1, 2, \dots, L - 1.$$

При этом, выход последнего слоя (выходного слоя) $\mathbf{a}^{(L-1)}$ является результатом работы (предсказанием) ИНС или выходным вектором ИНС.

- Используя явную форму (производных) функции стоимости отдельного измерения и функции активации, а также значение выходного вектора сети для текущего входного вектора, вычисляем ошибку нейронов последнего слоя по формуле

$$\boldsymbol{\delta}^{(L-1)} = \frac{\partial C_x}{\partial \mathbf{a}^{(L-1)}} \odot \sigma'(\mathbf{z}^{(L-1)}).$$

- Обходим слои в обратном направлении и вычисляем по рекуррентным формулам ошибку нейронов соответствующего слоя, используя явный вид функции активации

$$\boldsymbol{\delta}^{(l)} = \left((\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \right) \odot \sigma'(\mathbf{z}^{(l)}), \quad l = L - 2, L - 3, \dots, 1.$$

- Вычисляем частные производные функции стоимости отдельного измерения по весам и смещениям всех нейронов сети в соответствии с методом обратного распространения ошибок

$$\begin{aligned} \frac{\partial C_x}{\partial b_j^{(l)}} &= \delta_j^{(l)}, \\ \frac{\partial C_x}{\partial w_{ji}^{(l)}} &= a_i^{(l-1)} \delta_j^{(l)}, \end{aligned}$$

для всех $j = 0, 1, \dots, N^{(l)} - 1, i = 0, 1, \dots, N^{(l-1)} - 1, l = 1, 2, \dots, L - 1$.

- В соответствии с методом стохастического градиентного спуска, повторяем шаги 1-5 $M^{(mb)}$ раз (собирая подвыборку), сохраняя значения частных производных функ-

ции стоимости отдельного измерения по весам и смещениям всех нейронов сети для каждого из измерений, и вычисляем приближенное значение (итоговой) функции стоимости

$$C \approx \frac{1}{M^{(mb)}} \sum_{q=0}^{M^{(mb)}-1} C_{\mathbf{x}_{r(q)}},$$

а также ее частных производных по весам и смещениям всех нейронов сети ($j = 0, 1, \dots, N^{(l)}, i = 0, 1, \dots, N^{(l-1)}$ и $l = 1, 2, \dots, L - 1$)

$$\begin{aligned} \frac{\partial C}{\partial b_j^{(l)}} &\approx \frac{1}{M^{(mb)}} \sum_{q=0}^{M^{(mb)}-1} \frac{\partial C_{\mathbf{x}_{r(q)}}}{\partial b_j^{(l)}}, \\ \frac{\partial C}{\partial w_{ji}^{(l)}} &\approx \frac{1}{M^{(mb)}} \sum_{q=0}^{M^{(mb)}-1} \frac{\partial C_{\mathbf{x}_{r(q)}}}{\partial w_{ji}^{(l)}}, \end{aligned}$$

где $r(q)$ ($0 \leq r(q) < M^{(trn)}$) – это номера случайно отбираемых элементов тестовой выборки, а $\mathbf{x}_{r(q)}$ – это соответствующие входные векторы.

7. Обновляем веса и смещения всех нейронов сети в соответствии с методом градиентного спуска

$$\begin{aligned} b_j^{(l)} &\rightarrow b_j^{(l)} - \eta \frac{\partial C}{\partial b_j^{(l)}}, \\ w_{ji}^{(l)} &\rightarrow w_{ji}^{(l)} - \eta \frac{\partial C}{\partial w_{ji}^{(l)}}, \end{aligned}$$

для всех $j = 0, 1, \dots, N^{(l)}, i = 0, 1, \dots, N^{(l-1)}$ и $l = 1, 2, \dots, L - 1$.

8. Повторяем шаги 1-7 $\frac{M^{(trn)}}{M^{(mb)}}$ -раз (полный проход обучающего набора данных).
9. Повторяем шаги 1-8 $M^{(ep)}$ -раз, где $M^{(ep)}$ – это полное число эпох обучения, и сохраняем веса и смещения полученной обученной ИНС.

В заключение раздела отметим, что нами был подробно рассмотрен только один тип ИНС – полносвязная сеть, наиболее простой в реализации и наименее ресурсоемкий. В следующем разделе, посвященном анализу методов распознавания САРТСНА, нами будет также рассмотрен ряд специализированных решений, основанных на подходе ИНС, а в разделе 3 работы мы построим полносвязную ИНС для задачи распознавания САРТСНА, а также более детально проанализируем наилучший выбор гиперпараметров сети (в частности, функции активации и функции стоимости) и практические аспекты обучения ИНС. Также стоит отметить, что на сегодняшний день парадигма ИНС остается бурно развивающимся направлением, и регулярно находят новые приложения, в особенности, в области автоматизированной работы с изображениями (распознавание, классификация, сжатие, улучшение, генерация и т.д.).

2 Машинное обучение для автоматического распознавания CAPTCHA

2.1 Введение

Данная глава посвящена анализу методов машинного обучения для распознавания конкретного типа изображений, а именно, основного объекта изучения в данной работе – CAPTCHA. Нами будут рассмотрены наиболее распространенные подходы, из числа представленных в научной литературе, для первичной обработки, сегментации CAPTCHA, распознавания отдельных символов, получаемых после сегментации, а также ряд специализированных современных методов для задачи распознавания текстовых изображений и методов для ускорения и снижения потребляемой памяти в задачах распознавания.

В разделе 2.2 приводится обзор научно-технической литературы, посвященной задаче распознавания CAPTCHA, а также анализируются основные возникающие здесь сложности. Затем в разделе 2.3 рассматриваются современные подходы распознавания символьных изображений (буква алфавита или цифра), в частности мы описываем методы прореживания пикселей (thinning) и скелетирования (skeletonization) фигур, а также возможные методы выделения важных признаков (features) у символов. Также в разделе рассматривается процесс первичной подготовки CAPTCHA (удаление лишних цветов, искажений текста, нетекстовой информации и др.) и описываются возможные подходы к задаче сегментации CAPTCHA, то есть разделения текста, представленного на изображении, на отдельные символы.

И наконец, в разделе 2.4 описываются наиболее современные методы распознавания текстовой информации на графических изображениях и, в частности, задачи распознавания CAPTCHA. Нами рассматриваются сверточные искусственные нейронные сети и глубокие искусственные нейронные сети, а также в данном разделе рассматривается наиболее современный, на наш взгляд, подход, получивший свое развитие в последние 2-3 года – так называемый метод тензоризации глубоких искусственных нейронных сетей. В основе данного подхода лежит идея представления параметров искусственной нейронной сети в малопараметрических форматах и соответствующая адаптация алгоритмов прямого и обратного распространения сигнала в сети для значительного снижения потребляемой сетью памяти, а также ускорения обучения и функционирования сети.

2.2 Методы распознавания CAPTCHA

На сегодняшний день существует большое число научных работ, посвященных теме автоматического распознавания текстовой информации на изображениях CAPTCHA. Даные исследования имеют следующие цели:

- развитие существующих и создание новых методов машинного обучения и интеллектуального анализа данных на примере конкретной (сложной) задачи распознавания CAPTCHA;
- выявление общих закономерностей в задачах распознавания текстовой информации на графических изображениях на примере задачи распознавания CAPTCHA для последующего их применения в других приложениях (распознавание текстов, автомобильных и дорожных знаков и т.д.);
- создание автоматических систем «взлома» конкретных типов CAPTCHA для выявления слабых мест и выработки рекомендаций по улучшению алгоритмов формирования CAPTCHA.

В литературе наиболее распространены три условных подхода к задаче распознавания CAPTCHA. В рамках первого подхода строится некоторый сложный классификатор практически без учета специфики задачи (в большинстве случаев – это нейросетевой классификатор), который обучается на крупной выборке предварительно распознанных вручную CAPTCHA. Второй подход предполагает выявление конкретных характерных признаков (feature extraction) или эвристик, позволяющих распознать конкретную CAPTCHA с использованием простейшего классификатора (например, по методу ближайших соседей). Третий подход комбинирует первый два: предварительно выявленные характерные признаки используются как входной вектор для обучения нейросетевого классификатора.

Отметим, что при практической реализации любого из подходов отдельную сложность представляет предварительная очистка изображения от шумов и помех, а также сегментация, то есть разделение текста, представленного на изображении, на отдельные символы. Также в ряде работ осуществляется дополнительный анализ результата распознавания, например, если заранее известно, что на CAPTCHA представлено слово из определенного словаря: в этом случае могут быть проверены несколько наиболее вероятных предсказаний классификатора и выбрано то из них, которое содержится в словаре. Таким образом, процесс распознавания CAPTCHA можно упрощенно разделить на следующие этапы:

1. предварительная обработка изображения, включая удаление шумовых («мусорных») линий и групп пикселей, бинаризацию изображения, а также в ряде случаев скелетирование (утоньшение) линий;
2. сегментация изображения на отдельные символы с использованием определенных эвристик или вспомогательного классификатора;

verification code	binarization result	recognition result
WA_R	WA_R	WA_R
NDS_d	NDS_d	NDS_
SnMFbb	SnMFbb	SnMFbb
NWVCa	NWVCa	NWVCa
VdWXd	VdWXd	VdWXd
4NCDW4	4NCDW4	4NCDW4

Рис. 5: Пример автоматически распознанных CAPTCHA из работы [37].



Рис. 6: Примеры CAPTCHA, анализируемых в работе [38].

3. выявление характерных признаков и эвристик, облегчающих процесс распознавания отдельных символов (в качестве характерных признаков могут выступать количество внутренних областей у символа, число самопересечений и т.д.);
4. формирование обучающей размеченной выборки изображений и обучение классификатора на данной выборке;
5. пост-анализ результата работы классификатора, включая проверку по корректности длины итогового распознанного символа и принадлежность распознанного текста к определенной словарной базе.

В работе 2017 г. [37] для распознавания буквенно-цифровых CAPTCHA конкретного типа после бинаризации изображения используется метод выделения особенностей формы для каждой точки изображения (relative shape context and point mode). На рисунке 5 приводится результат распознавания нескольких CAPTCHA из работы. Как можно видеть, некоторые изображения распознаются существенно неверно, однако для ряда изображений метод дает корректный результат.

В работе 2016 г. [38] проводится распознавание текстовой CAPTCHA, подобных изображенными на рисунке 6 (tmall, JCAPTCHA, Microsoft CAPTCHA). Особое внимание в

The type of CAPTCHA	Accuracy rate	
	TM/OCR	CNN
6 减 2 等于 ?	288/300=96.0%	296/300=98.6%
0 乘 7 等于	264/300=88.0%	289/300=96.3%
九乘以八等于几	244/300=81.3%	273/300=91.2%
伍加肆 等于	241/300=80.3%	260/300=86.7%
肆减零等 于	235/300=78.3%	255/300=85%
叁加贰等 于	215/300=71.6%	263/300=87.6%
Y Tj eej	166/300=55.3%	192/300=64%
伍乘玖等于	162/300=54.0%	195/300=65%
8 乘 次 多 ?	100/300=33.3%	162/300=54%
0 乘 3 等 于 ?	105/300=35.0%	166/300=55.3%
捌乘壹等 于	192/300=64%	200/300=66.6%

Рис. 7: Результаты распознавания различных CAPTCHA из работы [39].

работе уделяется сегментации изображения. Для корректной сегментации на отдельные символы используется скользящее окно переменной ширины: выделенная область изображения отправляется на анализ нейросетевому классификатору (полносвязная ИНС), который был предварительно обучен на распознавание отдельных символов, затем в качестве места расположения символа выбора то положение окна, при котором классификатор дал наибольшую уверенность. Отдельные символы сегментированного таким образом изображения затем распознаются тем же нейросетевым классификатором. Заявленная в работе точность сегментации 58.25%, а точность распознавания отдельных символов 95%.

CAPTCHA	Success Recognition Rate SRR (%)	Segmentation Success Rate SSR (%)	Overall Precision OP(%)	No. of images in a dataset
Taobao	96.5	59.25	51.3	1000
MSN	91.25	51.5	27.1	500
eBay	97.5	62	53.2	1000

Рис. 8: Результаты распознавания различных CAPTCHA из работы [40].

В работе 2017 г. [39] для распознавания нескольких типов CAPTCHA, содержащих цифры и арабские символы, используется сверточная нейронная сеть. Для сегментации изображения на отдельные символы в работе предлагается использование простейшего метода анализа гистограммы распределения пикселей изображения по горизонтали: в местах, где значение оказывается меньше порогового предполагается окончание очередного символа. Результаты распознавания приведены на рисунке 7. Как можно видеть для ряда классов CAPTCHA точность автоматического распознавания является очень высокой.

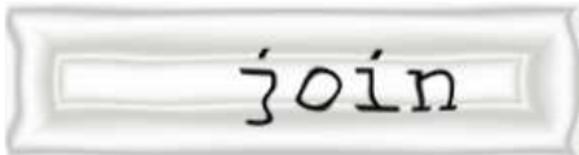
В работе 2016 г. [40] рассматривается задача распознавания буквенно-цифровых текстов на CAPTCHA с существенно перекрывающимися символами. На рисунке 8 мы отображаем полученные в работе результаты.

Отметим также очень активно цитируемую работу 2003 г. [41], в которой для сегментации и распознавания текстовых CAPTCHA, составленных из слов английского алфавита (см, рисунок 9) используется набор элегантных методов, основанных на ряде эвристик. В частности, в данной работе был разработан мощный фреймворк для выделения отдельных символов на основе анализа информации о формах и взаинных расположений закрашенных пикселей бинаризованного изображения.

Таким образом, резюмируя, можно видеть, что направление автоматического распознавания («взлома») CAPTCHA является активно развивающимся, однако предлагаемые в научных публикациях алгоритмы являются узко направленными и способными относительно эффективно работать только на конкретных классах CAPTCHA.



(a) collar



(c) join



(e) flag



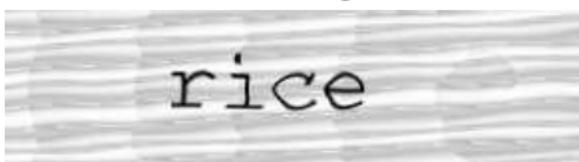
(g) smile



(i) horse



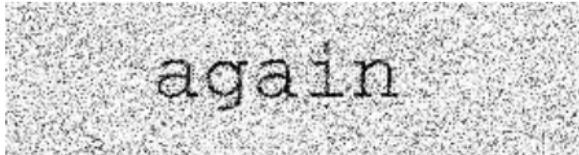
(k) weight



(m) rice



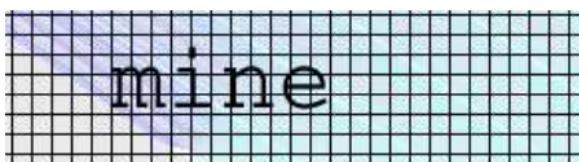
(b) here



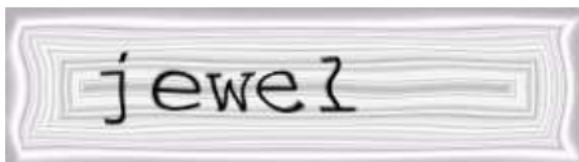
(d) again



(f) canvas



(h) line



(j) jewel



(l) sound



(n) spade

Рис. 9: Результаты распознавания различных CAPTCHA из работы [41].

2.3 Распознавание отдельных символов и сегментация САРТСНА

Программное распознавание символов является сложной задачей машинного обучения, в частности, из-за огромного числа вариантов начертания каждого конкретного символа. Простейшим (наивным) подходом распознавания символов является простое угадывание, при этом, как ни странно, данный метод имеет определенную точность, например для распознавания цифр точность на достаточно большой и равномерной тестовой выборке составит 10%, что может оказаться удовлетворительным для определенных задач. Еще один наивный подход – это вычисление суммарной интенсивности пикселей изображений обучающей выборки и последующее сравнение данной величины для тестовых данных с соответствующими шаблонами. В ряде случаев такой подход может давать достаточно высокую точность, очень сильно зависящую от единообразия начертания символов.

В основе современных подходов распознавания символов лежит идея выделения важных признаков (features; feature extraction) каждого из возможных символов, и последующего обучения классификатора на соответствующей обучающей выборке. На сегодняшний день существует множество алгоритмов выделения важных признаков у символов и масса моделей классификации, включая методы k-ближайших соседей, главных компонент, радиальных базисных функций, полно связанных и сверточных ИНС (в том числе, глубоких).

В качестве примера мы рассмотрим задачу классификации цифр. На рисунке 10⁽¹⁸⁾ приводится несколько изображений из базы MNIST [42], содержащей 70000 распознанных рукописных цифр (от нуля до девяти включительно), написанных различными людьми, в формате изображений размера 28×28 в градациях серого цвета. Обычно данный набор разбивается на три части: обучающий набор (50000 изображений), валидационный набор (10000 изображений) и тестовый набор (10000 изображений). Соответственно, обучающий набор используется для настройки параметров модели (классификатора), на валидационном наборе уточняется выбор гиперпараметров модели, а тестовый набор используется для проверки качества обученной модели.

Простейшим вариантом выделения важных признаков является использование исходных интенсивностей всех пикселей изображения, соответственно, в данном случае мы имеем 784 (28×28) признака для каждого изображения. В качестве классификатора может быть использована полно связная ИНС, подробно рассмотренная нами в предыдущем разделе работы, при этом точность классификации цифр может достигать более 96% (см. например [43]). В такой ИНС входной слой будет иметь 784 нейрона, а выходной слой – 10 нейронов, а результатом распознавания будет считаться номер выхода ИНС (или, эквивалентно, номер элемента выходного вектора ИНС), дающего наибольший сигнал. Описанный подход является наиболее простым в практическом реализации, так как он не предполагает никакой предварительной обработки изображений, однако при этом суще-

⁽¹⁸⁾ Данное изображение заимствовано с веб-ресурса <http://neuralnetworksanddeeplearning.com/chap1.html>.

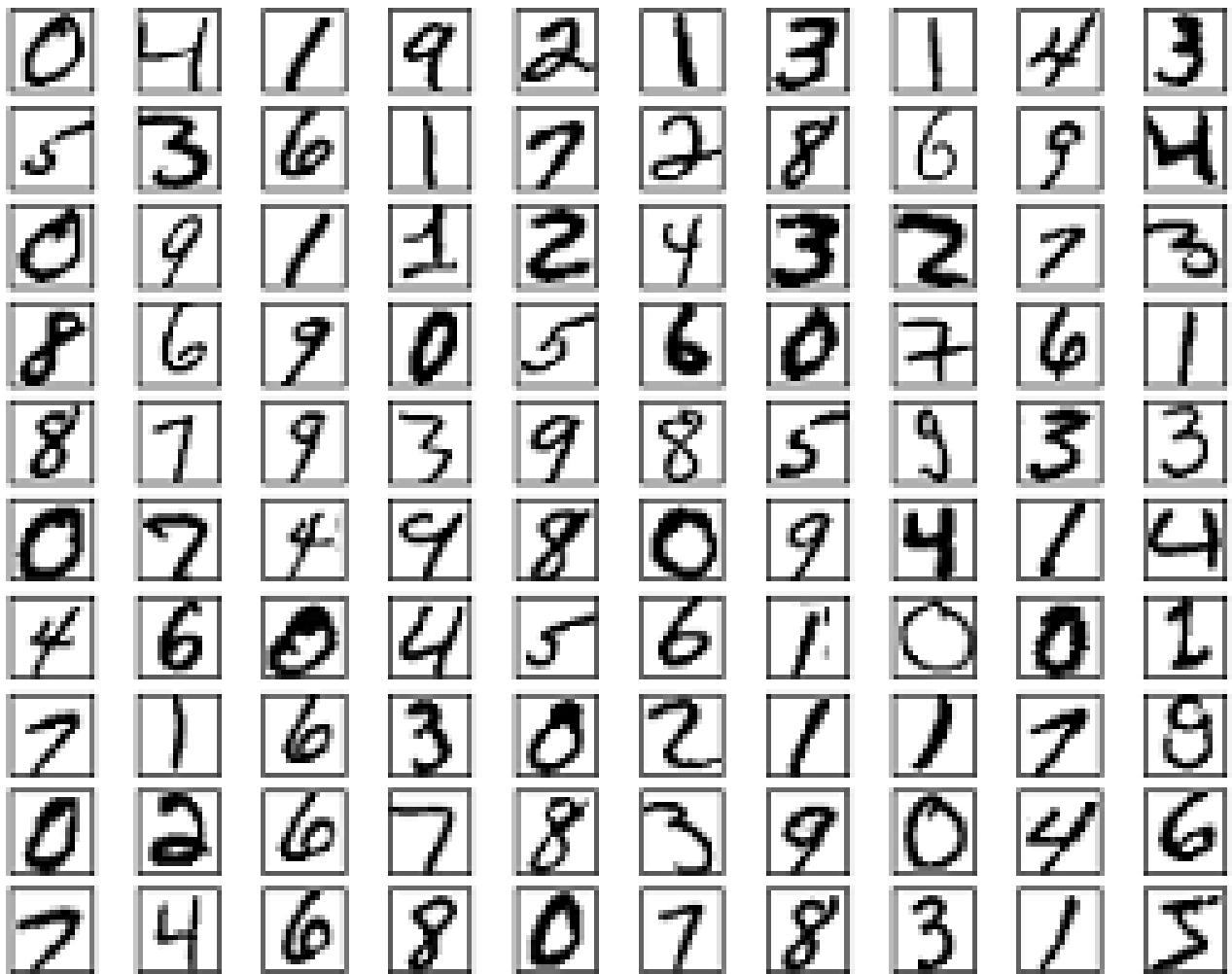


Рис. 10: Пример рукописных цифр, которые могут быть распознаны посредством ИНС.

ственным является как время обучения/работы модели, так и возможная ошибка результатов.

Более перспективным представляется подход, при котором производится предварительное выделение важных признаков из изображения по определенным алгоритмам, при этом число итоговых признаков может быть существенно меньше, чем количество пикселей в исходном изображении. Классическим подходом здесь является метод главных компонент [16], основанный на сингулярном разложении матрицы признаков. Элементы обучающего набора данных в данном методе группируются в матрицу (в рассматриваемом примере размера 50000×784), для которой вычисляется усеченное сингулярное разложение. Параметр обрезания сингулярного разложения соответственно становится параметром, определяющим степень сжатия исходных признаков, а данный подход позволяет выделять важные признаки без учета детальной структуры изображения.

Большую эффективность однако могут иметь методы, основанные на «ручном» выделении существенных признаков. Так например, рассматривая изображения, представленные на рисунке 10, можно увидеть, что характерным признаком нуля является отсутствие концевых пикселей (пикселей, соответствующих концу линии), в то время как для цифр

3 и 4 есть три таких пикселя на изображении. Или, например, для цифры 8 (и иногда для цифры 2) имеется пиксель, соответствующий пересечению двух линий. Другими важными признаками могут быть также наличие и количество углов, внутренних областей (областей фона, окруженных цветными пикселями, как, например, в цифрах нуль и восемь) и т.д. Подобные эвристики могут быть использованы в эффективном алгоритме выделения существенных признаков, которые затем будут использованы для обучения классификатора. При этом правила положенные в основу действия классификатора, могут быть как явными, например, для модели решающих деревьев (упрощенный пример здесь может быть таким: «если на изображении отсутствуют концевые пиксели, а также есть область фона, ограниченная цветными пикселями, значит на изображении представлена цифра 0»), так и неявными, например, для модели ИНС, где в процессе последовательного «изучения» экземпляров обучающей выборки сеть самостоятельно формулирует решающие правила, представленные в форме весов и смещений сети. Отметим, что как уже отмечалось ранее в работе, принципиальной слабой стороной нейросетевого подхода является отсутствие возможности увидеть конкретные правила, которыми руководствуется ИНС при принятии решения. Однако на сегодняшний день уже появился ряд работ, посвященных визуализации структуры ИНС (см., например [44]).

Описанный выше метод «ручного» выделения существенных признаков, тем не менее, не может быть напрямую применен к изображениям, в частности, ввиду их зашумленности. Так, например, на рисунке 10 у некоторых изображений присутствуют паразитные (мусорные) пиксели вне изображения, у некоторых четверок есть лишние выступы справа и т.д. Возможное решение – это предварительная фильтрация изображения и прореживание пикселей (thinning) или скелетирование (skeletonization) форм, представленных на изображении.

Первым шагом в алгоритме фильтрации изображения часто бывает процесс бинаризации, заключающийся в переходе к двухцветовой (бинарной) палитре из нулей и единиц, где нуль соответствует фону, а единица – изображению. Бинаризация обычно выполняется по некоторому пороговому значению (в предположении, что фон изображения имеет нулевую интенсивность пикселей): если интенсивность некоторого пикселя не превосходит заданное пороговое значение, то такой пиксель удаляется (делается фоновым), при этом простейшим вариантом является использование нулевого порогового значения. Бинаризованное изображение может быть представлено соответствующей матрицей, состоящей только из нулей и единиц. Например, для цифры 4 матрица может иметь вид (для иллю-

страции, на изображении представлено также несколько шумовых пикселей)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Для очистки бинаризованного изображения от мусорных (шумовых) пикселей может быть использован алгоритм удаления пикселей по девятиточечному шаблону [45]. В данному алгоритме последовательно обходятся все пиксели изображения и соответствующий нефоновый пиксель (равный 1), удаляется если он соответствует одному из следующих шаблонов (центральный пиксель в представленных шаблонах соответствует текущему анализируемому нефоновому пикселию)

$$\begin{aligned} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\ & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \\ & \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \\ & \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Приведенные шаблоны имеют простой смысл. Так, например, первый шаблон соответствует ситуации, когда данный пиксель окружен исключительно фоном. На втором шаблоне отражена ситуация, когда данный пиксель имеет в своем окружении ровно один нефоновый пиксель, который располагается справа от него. Последний шаблон отвечает случаю, когда данный пиксель имеет ровно три нефоновых пикселя в своем окружении, причем расположены они справа вверху от него.

После проведения бинаризации и очистки изображения от шума, может быть проведена

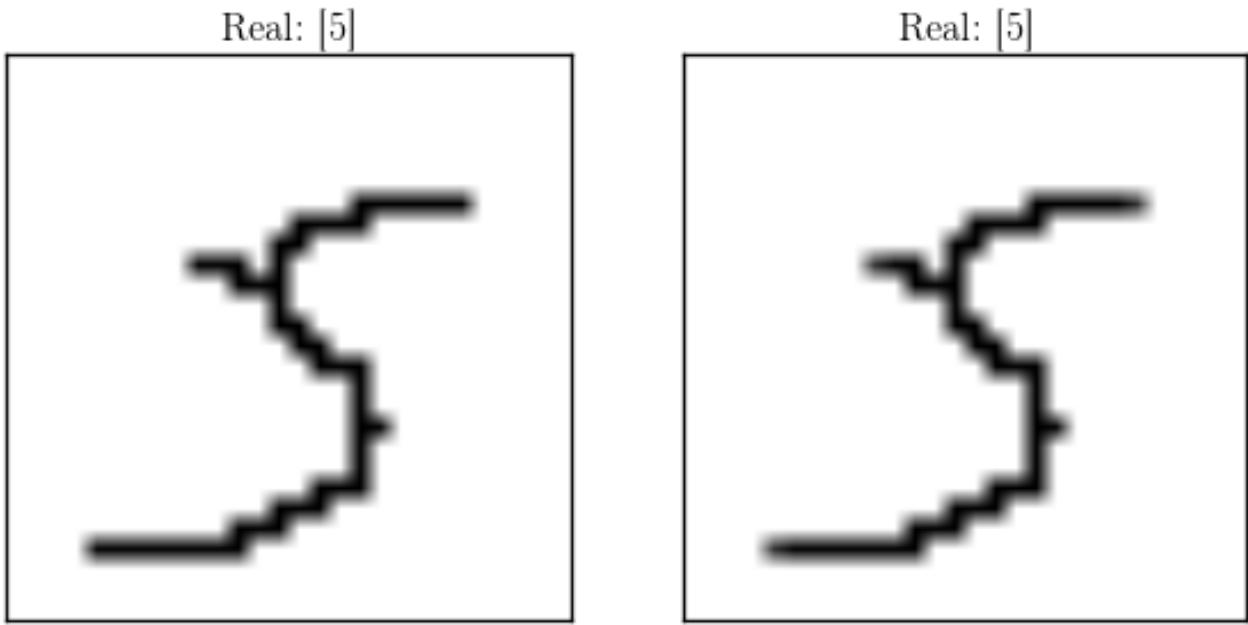


Рис. 11: Результат прореживания бинаризованного символа 5 из базы MNIST.

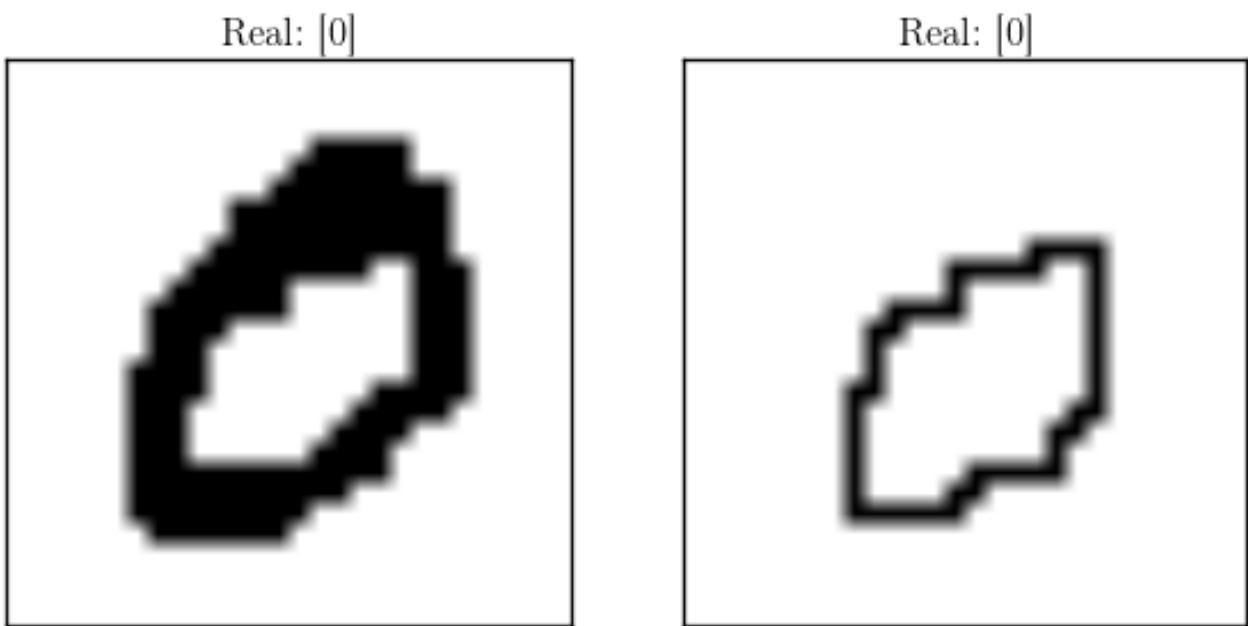


Рис. 12: Результат прореживания бинаризованного символа 0 из базы MNIST.

также полезная для последующего выделения ключевых признаков процедура прореживания пикселей. Суть данного метода заключается в последовательном удалении нефоновых пикселей для преобразования исходного изображения в модифицированное изображение, где все или почти все линии имеет единичную толщину, при сохранении основной информации об изображении. В обзорных статьях [46] и [47] рассматриваются основные существующие алгоритмы прореживания. Мы более подробно остановимся на алгоритме из работы [48] для последующей реализации данного метода в разрабатываемом нами программном пакете для распознавания CAPTCHA.

В рассматриваемой реализации процедуры прореживания, как и в описанном выше методе фильтрации, используется девятиточечный шаблон для каждого из пикселей изображений. При этом вводится следующая нумерация пикселей

$$\begin{bmatrix} P_9 & P_2 & P_3 \\ P_8 & P_1 & P_4 \\ P_7 & P_6 & P_5 \end{bmatrix},$$

где символ P_1 соответствует текущему анализируемому пикслю, а символы P_2, P_3, \dots, P_9 соответствуют его соседям. Также вводятся две функции: функция $B(P_1)$, равная количеству нефоновых пикселей, соответствующих рассматриваемому пикслю P_1

$$B(P_1) = P_2 + P_3 + \dots + P_9,$$

и функция $A(P_1)$, равная количеству пар вида 0, 1 в окружении рассматриваемого пикселя.

Так, например, для случая

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

имеем $B(P_1) = 3$ и $A(P_1) = 2$.

Соответствующий алгоритм прореживания формулируется следующим образом. Осуществляется двукратный проход всех пикселей изображения, исключая граничные. Для каждого обнаруженного нефононового пикселя проверяется выполнение четырех условий:

1. $2 \leq B(P_1) \leq 6$,
2. $A(P_1) = 1$,
3. $P_2 * P_4 * P_6 = 0$,
4. $P_4 * P_6 * P_8 = 0$,

и в случае выполнения каждого из условий, соответствующий пиксель P_1 удаляется (делается фоновым). На втором проходе проверяется совместное выполнение следующих условий:

1. $2 \leq B(P_1) \leq 6$,
2. $A(P_1) = 1$,
3. $P_2 * P_4 * P_8 = 0$,
4. $P_2 * P_6 * P_8 = 0$,

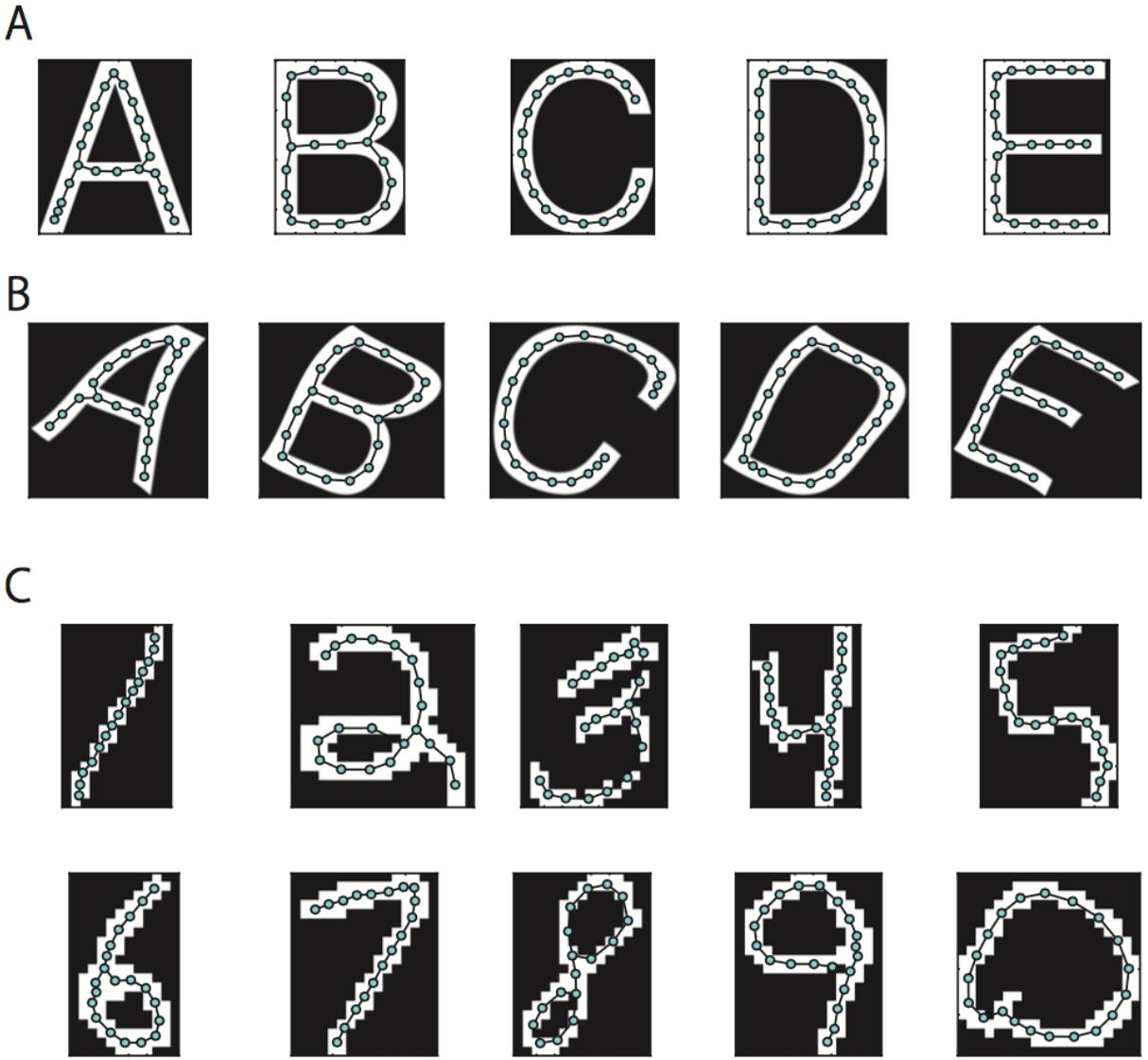


Рис. 13: Пример скелетирования символов (из работы [49]), соответствующих буквам латинского алфавита (А), наклонным буквам латинского алфавита (В) и цифрам (С). Цепочки на изображениях соответствуют результатам скелетирования.

и в случае выполнения каждого из них, соответствующий пиксель P_1 удаляется (делается фоновым). Отметим, что описанный алгоритм на первом проходе удаляет граничные пиксели слева внизу, а на втором проходе – справа вверху. Нами был программно реализован данный алгоритм, и на рисунках 11 и 12 приводятся результаты прореживания для двух очищенных от шума и бинаризованных изображений из базы MNIST, содержащих цифры 5 и 0 соответственно.

Вместо процедуры прореживания бинаризованного изображения, описанной выше, может быть применен альтернативный подход, называемый скелетированием изображений. Скелетирование изображений также позволяет упростить последующий процесс извлечения существенных признаков изображения. Суть данного метода заключается в замене исходной фигуры на изображении на однопиксельные линии, задающие общий контур фигуры. Отметим работы [50] и [49] где для скелетирования объектов, представленных

на изображении, используются современные методы машинного обучения такие как распространяющийся нейронный газ [51] (growing neural gas, GNG) и метод относительных соседей на графе [52] (relative neighborhood graph, RNG). Также отметим работу [53], где для скелетирования предложен быстрый метод, использующий свойства симметрии объекта, и работу [54], где для подавления артефактов скелетирования используются методы анализа точек регулярности и сингулярности формы. В качестве примера, на рисунке 13 приводятся результаты процедуры скелетирования из работы [49], примененной к символам латинского алфавита и к цифрам. Как можно видеть, в результате скелетирования образуются однопиксельные кривые, сохраняющие общую информацию о форме исходного изображения.

В итоге, после успешного проведения всех описанных процедур, мы имеем очищенное от шума, бинаризованное изображение, на котором все линии имеют толщину не более одного пикселя.

2.4 Современные методы автоматического распознавания

В данном разделе мы рассмотрим наиболее современные методы машинного обучения в контексте задачи распознавания САРТСНА. Как уже неоднократно указывалось в работе, на сегодняшний день наиболее активно развивающимся направлением является нейросетевой подход. Создано большое число новых архитектур ИНС, постоянно публикуются работы с новыми эвристиками выбора функций активации и функции стоимости. В этом контексте, подробно рассмотренная в первом разделе работы полносвязная ИНС является лишь частным (и не всегда лучшим) случаем ИНС. Однако полносвязные сети продолжают активно использоваться исследователями и инженерами.

При этом, естественным этапом в развитии метода ИНС стало увеличение их глубины, то есть последовательное расположение нескольких полносвязных слоев и иных типов слоев (с полным числом слоев даже более двадцати на сегодняшний день) для возможности иерархического анализа и выявления закономерностей во входных данных. Такие ИНС называют глубокими (в литературе используется также термин «глубинные») сетями, при этом отсутствует формальное ограничение на число слоев, начиная с которого сеть можно считать глубокой. Так, в качестве примера, на рисунке 14⁽¹⁹⁾ схематически изображена полносвязная нейронная сеть с третями скрытыми слоями, которую можно считать глубокой ИНС. Принцип действия такой сети не имеет принципиальных отличий от механизма работы полносвязной ИНС с одним скрытым слоем, описанной в первом разделе работы. Входной слой передает без изменений входной вектор ИНС на первый внутренний слой. Нейроны первого внутреннего слоя осуществляют преобразование сигнала в соответствии с текущими значениями весов и смещений. Выходной сигнал затем подается на вход второго внутреннего слоя, где осуществляются аналогичные преобразования. Затем аналогично сигнал обрабатывается третьим внутренним слоем и поступает на выходной слой сети. После преобразования сигнала выходным слоем, получаем выходной вектор сети, являющийся ее предсказанием.

Глубокие нейронные сети позволяют существенно повысить качество предсказаний сети. На сегодняшний день отсутствует полное обоснование причин их эффективности, кроме интуитивного соображения, заключающегося в следующем факте. Человеческому мозгу свойственна иерархичность восприятия. Когда мы видим некоторый объект, то изначально светочувствительными элементами глаза воспринимаются отдельные точки изображения, затем нервной системой глаза производится первичная обработка сигнала: выделяются некоторые простые элементы (линии, углы). Затем сигнал передается в мозг человека, где происходит дальнейшая последовательная обработка с введением все более сложных сущностей. Аналогично глубокая ИНС при распознавании, например, лица человека на изображении может на первом внутреннем слое производить идентификацию

⁽¹⁹⁾ Данное изображение заимствовано с веб-ресурса <http://searchnetworking.techtarget.com/definition/neural-network>.

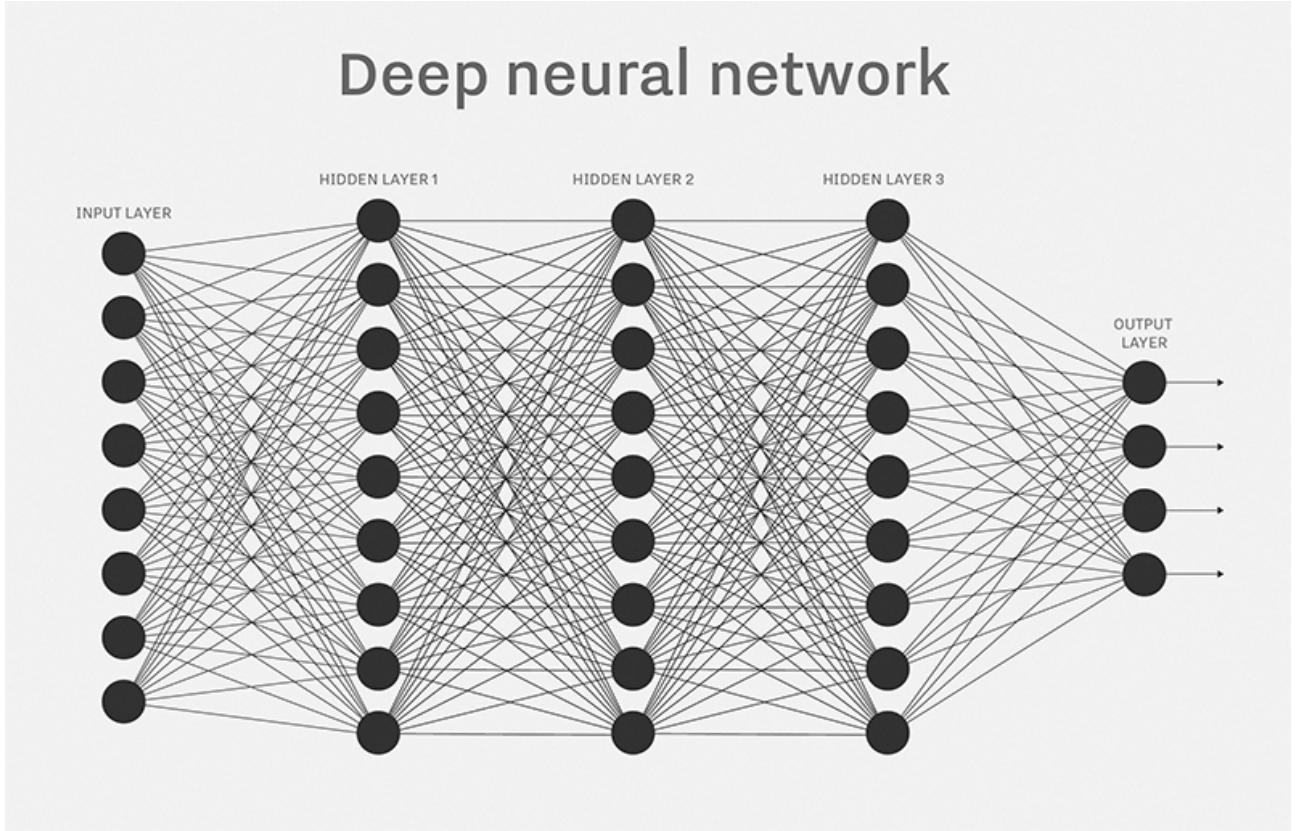


Рис. 14: Схематическое изображение полносвязной глубокой нейронной сети с тремя скрытыми слоями. Входной вектор \mathbf{x} сети для данного примера имеет длину 8, а выходной вектор \mathbf{y} – длину 4.

некоторых простых форм (овалов, линий, углов), на следующем слое выявленные сущности могут усложняться (при наличии сигналов от нейронов предыдущего слоя, ответственных, например, за выявление овалов, может приниматься решение о наличии глаза на изображении) и далее от слоя к слою происходит дальнейшее усложнение сущностей, которое в итоге приводит к ответу сети о наличии или отсутствии лица на изображении.

Описанная иерархичность обработки входной информации, по всей видимости, и обеспечивает потрясающие результаты, которые глубокие ИНС демонстрируют в задачах распознавания и классификации. Так например, глубокие сети с более чем тридцатью слоями используются в настоящее время для классификации популярного набора данных ImageNet [55]. Данная база содержит более 16 миллионов полноцветных изображений, разбитых на 20 тысяч категорий. Задача, соответственно заключается в обучении сети предсказывать категорию заданного изображения. Глубокие ИНС на сегодняшний день решают данную несоизмеримо лучше человека (см., например, [56] и [24]), верно определяя топ-5 категорий с точностью более 95%. Отметим, что для описанной базы данных наиболее распространенная оценка качества сети – это именно топ-5 категорий; рассматривается 5 предсказаний категории, которые сеть считает наиболее вероятными, и если одно из них оказывается совпадающим с правильным, то такое предсказание считается верным. Такой критерий оценки связан с тем фактом, что для огромного числа изображений

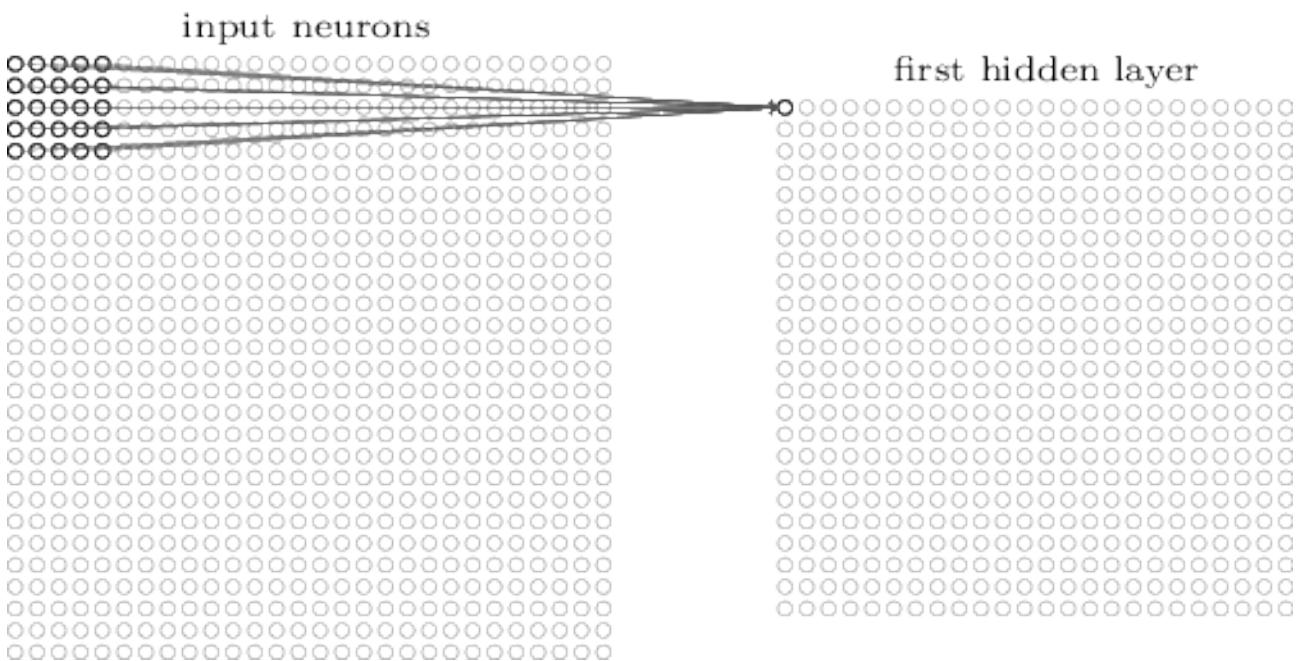


Рис. 15: Схематическое изображение сверточного слоя ИНС.

практически невозможно (в том числе, человеку) точно определить категорию, например, для фотографии, на которой изображена собака и кошка, трудно выбрать между категориями «собаки» и «кошки». Добавим, что глубокие ИНС нашли на сегодняшний день ряд применений и в задачах распознавания речи на основе обработки отдельных звуков [57], а также на основе специальных словарей [58].

Еще одно важное направление современного развития нейросетевого подхода – это сверточные ИНС (convolutional neural network) [59, 60]. Данный подход был разработан в последнее десятилетие и является естественным обобщением метода классических ИНС, адаптированным для выявления закономерностей и работы с двухмерными и трехмерными входными структурами данных, в частности, для работы с графическими изображениями.

Основная идея здесь следующая. Как описывалось в предыдущем разделе, для использования полносвязной ИНС в задаче распознавания изображений, необходимо векторизовать матрицу интенсивностей пикселей, то есть преобразовать ее в вектор, склеивая последовательно строки или столбцы исходной матрицы. При этом неизбежно теряется часть информации связанной с двумерной структурой изображения, например, те пиксели, которые располагались рядом на изображении, в векторе уже могут оказаться на значительном удалении. Для разрешения подобных проблем был предложен новый тип слоев ИНС - сверточный слой. Данный слой также состоит из нейронов, однако они являются упорядоченными на плоскости в соответствии с геометрией исходного изображения. В качестве примера на рисунке 15⁽²⁰⁾ схематически изображен входной слой сверточной

⁽²⁰⁾ Данное изображение, а также изображения на рисунках 16 и 17 заимствованы с веб-ресурса <http://neuralnetworksanddeeplearning.com/chap6.html>.

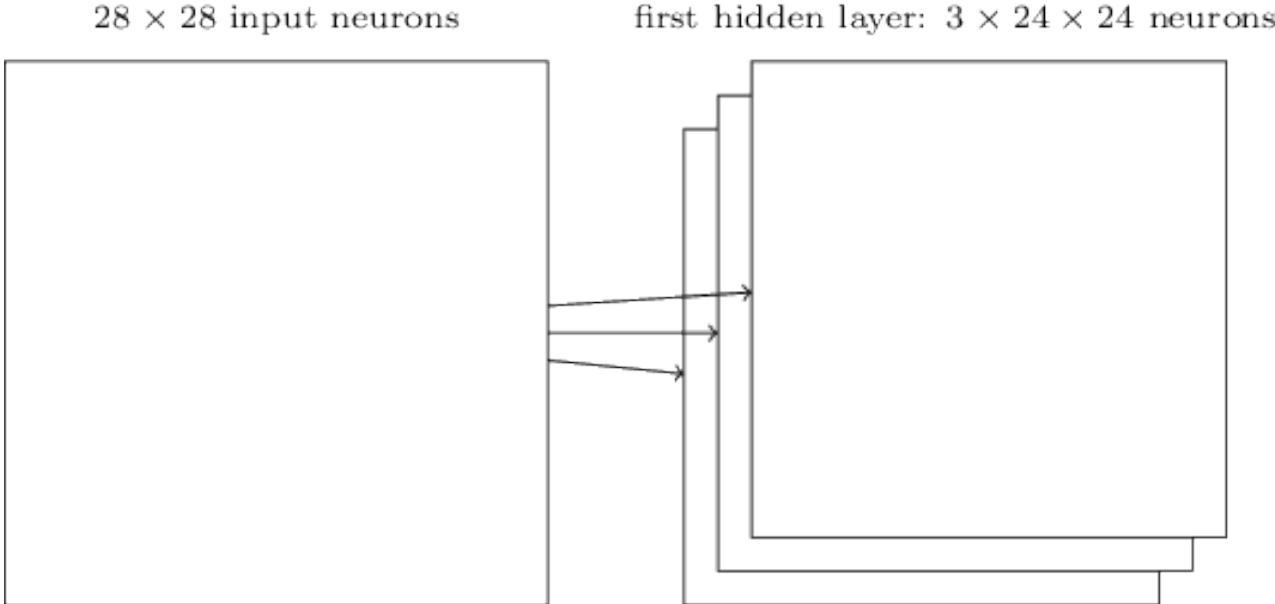


Рис. 16: Схематическое изображение совокупности трех сверточных слоев ИНС.

сети размера 28×28 (он может, например, соответствовать изображению размера 28×28) и следующий за ним сверточный слой. Входной слой (input neurons на рисунке) соответствует значениям интенсивности пикселей изображения с учетом геометрии изображения. Для получения сигнала сверточного слоя, находящегося за ним, выполняется следующая операция. Выбирается некоторый размер окна (например, 5×5 пикселей, как изображено на рисунке), и выбираются соответствующие пиксели (нейроны), попадающие в данное окно. Изначально (как изображено на рисунке) окно располагается в верхнем левом углу изображения. Для всех нейронов, попавших в окно, вычисляется свертка по формуле

$$\sigma \left(b + \sum_{l=0}^4 \sum_{m=0}^4 w_{l,m} a_{i+l, j+m} \right), \quad (17)$$

значение которой будет выходом (i, j) -ого нейрона сверточного слоя (для начального положения окна имеем соответственно $i = 0$ и $j = 0$). Затем происходит сдвиг окна на один пиксель вправо и вновь вычисляется выход (i, j) -ого нейрона сверточного слоя (теперь $i = 0$ и $j = 1$) по формуле (17). Данный процесс продолжается до достижения правой границы изображения, затем происходит возврат к левой границе изображения со сдвигом на один пиксель вниз (что соответствует $i = 1$ и $j = 0$).

Поскольку на каждом шаге окно размера 5 сдвигается на один пиксель, и при этом размер входного слоя для рассматриваемого примера 28×28 , то в итоге сверточный слой, в соответствии с описанным алгоритмом, будет содержать 24×24 нейронов (так всего возможно $28 - 5 + 1 = 24$ положений окна по каждому из направлений). При этом, в соответствии с формулой (17), каждому положению окна соответствует одни и те же значения весов $w_{l,m}$ и смещения b , таким образом, сверточный слой имеет всего 26 параметров.

Смысл введения сверточного слоя в соответствии с формулой (17) следующий. Опи-

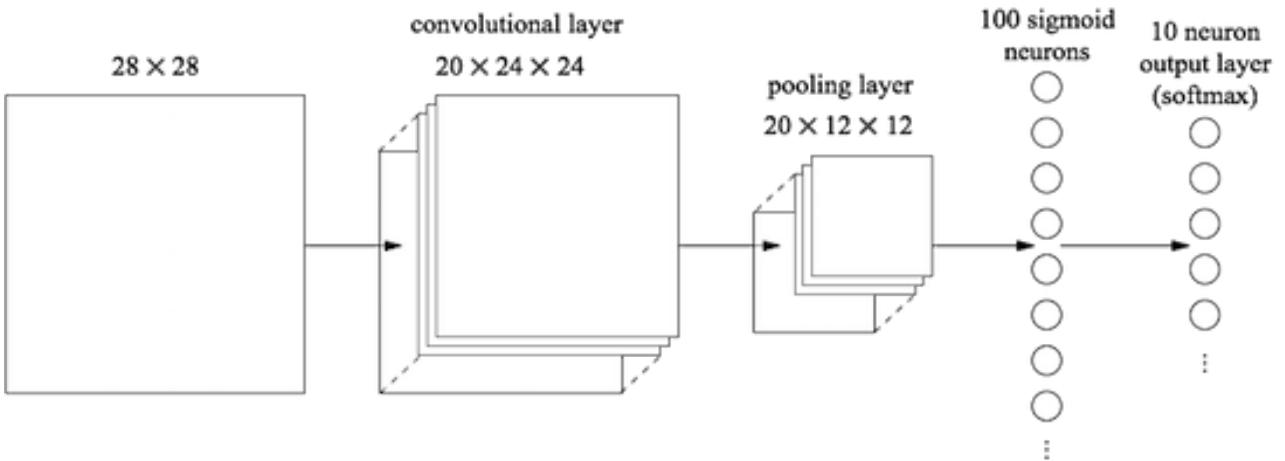


Рис. 17: Схематическое изображение ИНС, содержащей сверточные слои.

санный обход входного слоя с помощью небольшого окна и малым сдвигом соответствует поиску некоторой (одной) особенности на изображении, например, это может быть поиск острого угла на картинке. В данном контексте становится понятно, почему для каждого положения окна используются одни и те же веса и смещения – конкретный сверточный слой нацелен на выявление конкретной особенности на изображении (в любом месте изображения).

Поскольку один сверточный слой, как было описано выше, способен выявить только одну особенность, то логичным представляется параллельное использование нескольких сверточных слоев (корректнее – подслоев, так все эти сверточные подслои располагаются в одном месте сети, как бы расширяя ее в дополнительном измерении), каждый из которых характеризуется своими весами и смещениями и нацелен на выявление некоторой особенности изображения. На рисунке 16 приведен пример части ИНС со входным слоем и первым внутренним слоем, состоящим из трех сверточных подслоев.

В реальных приложениях сверточных подслоев в одном слое может быть существенно больше, чем три – их число должно соответствовать количеству важных особенностей изображения. Так в качестве примера, на рисунке 17 изображена глубокая ИНС, содержащая сверточный слой, который, в свою очередь, состоит из 20 подслоев, каждый из которых выделяет одну характерную особенность на изображении.

На сегодняшний день ИНС, в том числе, со сверточными слоями, применяются в широком спектре приложений. При этом сложность архитектуры современных ИНС достигла потрясающих масштабов. На рисунке 18 изображена сверточная глубокая ИНС, использованная для классификации более чем миллиона изображений в высоком разрешении на 1000 различных заданных классов в рамках популярного конкурса машинного обучения ImageNet, описанного выше. Сеть имеет 650000 нейронов, включая пять сверточных слоев и три полносвязных слоя, при этом сеть имеет более 60 миллионов параметров. Приведенный пример взят из работы [59] 2010 года, и на сегодняшний день, безусловно, создаются и

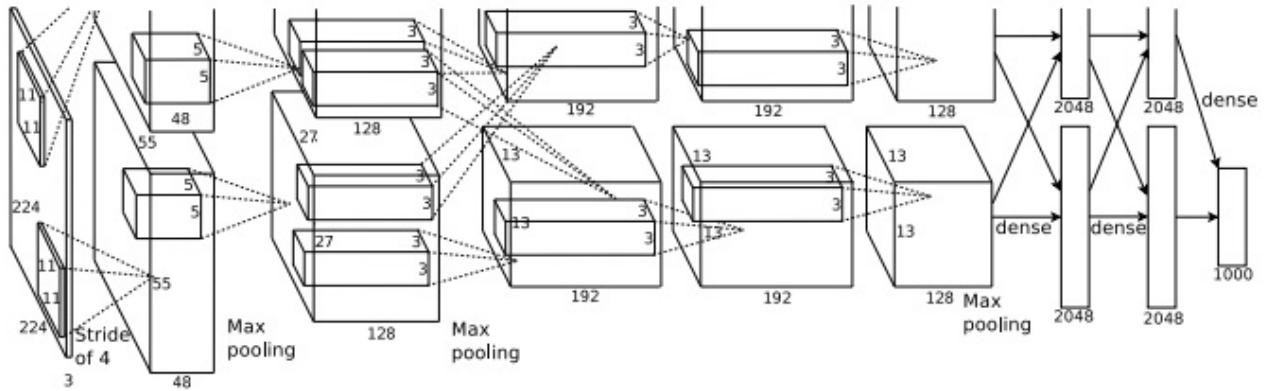


Рис. 18: Пример реальной глубокой нейронной (из работы [59]) для классификации изображений.

успешно обучаются существенно более сложные ИНС с еще большим числом параметров.

Чрезвычайно высокая сложность и требовательность к вычислительным ресурсам современных глубоких нейросетевых структур обуславливает необходимость разработки новых эффективных подходов для обучения и функционирования сети. Отметим, что обучение и функционирование являются двумя различающимися аспектами работы с современными сетями. Так первичное обучение глубоких сетей обычно осуществляется с использованием суперкомьютеров и вычислительных кластеров на мощных графических процессорах, например, это может быть обучение сети, предсказывающей пробки на вычислительных системах Яндекса. Обученная таким образом сеть далее может использоваться конкретными пользователями для получения соответствующих прогнозов (фаза функционирования), при этом запросы во многих случаях осуществляются с маломощных мобильных устройств (смартфоны, планшеты и т.д.).

На обоих приведенных этапах ИНС потребляет массу вычислительных ресурсов. При этом обучение современных сетей может занимать недели и месяцы, а функционирование обученной сети на мобильном устройстве практически невозможно, поэтому для использования сети необходим постоянный доступ к сети Интернет: мобильное устройство делает запрос к серверу, на котором размещена сеть, делается предсказание, и результат отправляется обратно по сети. Однако такая модель имеет ряд ограничений, в частности, необходимость наличия Интернет-соединения. Яркий иллюстративный пример здесь – это наблюдение за тем как «глушеют» электронные помощники с искусственным интеллектом (например, Siri) при потери доступа к сети Интернет. Очевидное решение здесь – это размещение обученных ИНС непосредственно на мобильных устройствах, однако для этого необходимы новые алгоритмы, позволяющие компактно представлять массивы весов сети и вычислительно эффективно проводить процедуру прямого распространения сигнала в сети.

Наиболее современный на сегодняшний день подход в рамках данного направления, на наш взгляд – это методы тензоризации ИНС [61, 62, 63, 64, 65, 66, 67]. Данное направ-

ление начало активно развиваться только лишь в последние 2-3 года, и основано на идее малопараметрических тензорных аппроксимаций.

Как было показано в работах [68, 69, 70] во многих практически значимых случаях многомерные массивы (тензоры) могут быть представлены в компактной (малопараметрической или малоранговой) форме в рамках, так называемого, разложения тензорного поезда (tensor train decomposition). Пусть \mathcal{X} – это многомерный массив, имеющий d измерений (при $d = 2$ – это матрица, а при $d = 1$ – вектор), с числом элементов n вдоль каждого из измерений. Как можно видеть, полное число элементов такого массива n^d , и уже при сравнительно небольших значениях d и n такие массивы становятся невозможным хранить в памяти компьютеров (например, для четырехмерного массива с 1000 элементов вдоль каждого измерения мы имеем полное число элементов 10^{12}) – так называемая проблема проклятия размерности. Однако практически любой массив данных имеет некоторую внутреннюю структуру, при учете которой он может быть представлен в значительно более компактной форме. В рамках разложения тензорного поезда многомерный массив \mathcal{X} путем многократного вычисления сингулярного разложения (SVD, см. Приложение 1) матриц-разверток, представляется в виде

$$\mathcal{X} = \sum_{\alpha_1=1}^{r_1} \sum_{\alpha_2=1}^{r_2} \dots \sum_{\alpha_{d-1}=1}^{r_{d-1}} \mathcal{G}_1(1, i_1, \alpha_1) \mathcal{G}_2(\alpha_1, i_2, \alpha_2) \dots \mathcal{G}_d(\alpha_{d-1}, i_d, 1),$$

где r_1, r_2, \dots, r_{d-1} – это ранги разложения, а трехмерные массивы $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_d$ – это, так называемые, ядра разложения. Как можно видеть из приведенной формулы, полное число параметров в таком разложении не превосходит величины

$$d \times n \times \max(r_k)^2,$$

то есть линейно зависит от размерности и числа элементов вдоль каждого измерения при ограниченности рангов разложения. В указанных работах было доказано, что ранги действительно являются ограниченными для широкого класса многомерных массивов, в частности, полученных из линейных комбинаций полиномиальных и тригонометрических функций и т.д.

Аналогичное разложение может быть применено также к матрицам и векторам, если они формально представлены в виде многомерных массивов путем процедуры тензоризации [71]. Для получаемых объектов потребляемая память и вычислительная сложность становится логарифмической по числу элементов, что позволяет в ряде случаев в тысячи раз снижать потребление памяти и вычислительных ресурсов.

В работе [61] описанное разложение было применено для компактного представления полносвязного слоя ИНС, что позволило на порядок ускорить работу сети, а, например, в работе [62], подобное разложение было использовано для представления сверточного слоя сети, что также в десятки раз улучшило ее производительность.

3 Реализация метода автоматического распознавания CAPTCHA

3.1 Введение

В данной, заключительной, главе мы рассматриваем практические аспекты реализации программного комплекса для распознавания текстов на графических изображениях, в том числе для распознавания CAPTCHA. В основе программного комплекса лежит полносвязная искусственная нейронная сеть (ИНС), принцип действия которой был рассмотрен в первой главе работы. Для предварительной обработки изображения и сегментации его на отдельные символы мы используем методы, описанные во второй главе работы.

В разделе 3.2 мы более подробно рассмотрим выбранную архитектуру ИНС и сформулируем алгоритмы обучения и функционирования, на основе которых будет разработана программная реализация.

Далее в разделе 3.3 мы детально описываем созданный программный пакет CapSolver для распознавания текста на графических изображениях, включая CAPTCHA, содержащий в качестве основных компонент модуль, реализующий полносвязную ИНС; модуль, реализующий загрузку, обработку и отображение изображений; браузерное расширение (плагин) для автоматического обнаружения поддерживаемой CAPTCHA на веб-странице, ее отправки на локальный сервер и автоматической вставки результата распознавания в соответствующее поле веб-страницы; локальный сервер и обработчики клиент-серверных сообщений и ряд вспомогательных модулей. Браузерное расширение написано на языках html, css, javascript, а также использует специфическое API браузера Google Chrome. Остальная часть кода создана с использованием языка программирования python 2.7 и ряда вспомогательных библиотек.

И наконец в разделе 3.4 мы приводим результаты практического применения программного пакета и особенности соответствующих наборов обучающих данных для конкретных задач распознавания, включая распознавание рукописных цифр из базы MNIST (60000 изображений в размеченной базе) и самостоятельно размеченной автором работы циферной CAPTCHA иностранного новостного портала.

Алгоритм 1 Алгоритм распространения сигнала в ИНС (forward).

Входные данные:

число слоев сети L ;
количества нейронов в слоях $N^{(0)}, N^{(1)}, \dots, N^{(L-1)}$;
матрицы весов нейронов $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$;
векторы смещений нейронов $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$;
функции активации для каждого из слоев $\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(L-1)}$;
входной вектор \mathbf{x} размера $N^{(0)}$.

- 1: Задать векторы $\mathbf{z}^{(0)} = \mathbf{x}$ и $\mathbf{a}^{(0)} = \mathbf{x}$.
- 2: **for** $l = 1, 2, \dots, L - 1$ **do**
- 3: Вычислить вектор $\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$.
- 4: Вычислить вектор $\mathbf{a}^{(l)} = \sigma^{(l)}(\mathbf{z}^{(l)})$.
- 5: **end for**

Результат: выходные векторы слоев $\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \dots, \mathbf{a}^{(L-1)}$
($\mathbf{a}^{(0)}$ совпадает со входом сети, $\mathbf{a}^{(L-1)}$ является выходным вектором сети).

3.2 Архитектура искусственной нейронной сети

Основным этапом при решении задачи автоматизированного распознавания символов и САРТСНА является создание и обучение классификатора, позволяющего распознавать конкретные символы в режиме онлайн на основе результатов предварительного обучения в режиме оффлайн на размеченной обучающей выборке данных. В случае распознавания текстовой информации на графических изображениях элементами обучающей выборки являются пары \mathbf{x}, \mathbf{y} , где \mathbf{x} – это определенным образом упорядоченный массив интенсивностей отдельных пикселей изображения, а \mathbf{y} – это желаемый выход классификатора, то есть правильное значение символа, представленного на изображении.

Ранее в данной работе были описаны преимущества нейросетевого подхода для решения задач классификации. Для автоматизации процесса распознавания САРТСНА нами была реализована полносвязная ИНС, принцип действия которой подробно описывается в разделе 1 данной работы. Прямое распространение сигнала в сети может быть реализовано в форме функции forward, представленной в виде псевдокода в алгоритме 1. Данная функция принимает текущие значения параметров сети (весов и смещений всех нейронов), а также использует гиперпараметры сети (число слоев, количества нейронов в слоях, функции активации для каждого из слоев). По заданному входному вектору функция вычисляет и возвращает выходы всех слоев сети, при этом выход последнего слоя сети является выходом (предсказанием) сети, а выходы остальных слоев необходимы для работы алгоритма обучения сети.

В алгоритме 2 представлен псевдокод функции backward, осуществляющей одну эпоху обучения ИНС с использованием метода стохастического градиентного спуска и метода обратного распространения ошибок, описанных в разделе 1 данной работы.

При программной реализации алгоритма для каждого из слоев необходимы две функции: одна вычисляет значение функции активации, а вторая – ее производную по задан-

Алгоритм 2 Алгоритм одной эпохи обучения ИНС (backward).

Входные данные:

- число слоев сети L ;
- количества нейронов в слоях $N^{(0)}, N^{(1)}, \dots, N^{(L-1)}$;
- матрицы весов нейронов $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$;
- векторы смещений нейронов $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$;
- функции активации для каждого из слоев $\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(L-1)}$;
- функция стоимости одного измерения C_x ;
- размер подвыборки $M^{(mb)}$;
- параметр скорости обучения η ;
- набор обучающих данных $(\mathbf{x}_q, \mathbf{y}_q)$ ($q = 0, 1, \dots, M^{(trn)} - 1$).

- 1: Задать матрицы $\Delta\mathbf{W}^{(1)}, \Delta\mathbf{W}^{(2)}, \dots, \Delta\mathbf{W}^{(L-1)}$, заполненные нулями, и имеющие формы, совпадающие с $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$ соответственно.
- 2: Задать векторы $\Delta\mathbf{b}^{(1)}, \Delta\mathbf{b}^{(2)}, \dots, \Delta\mathbf{b}^{(L-1)}$, заполненные нулями, и имеющие размер, совпадающий с $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$ соответственно.
- 3: **for** $q = 0, 1, \dots, M^{(mb)} - 1$ **do**
- 4: Используя равномерное случайное распределение, получить случайное число $r(q)$ ($0 \leq r(q) < M^{(mb)}$) и выбрать соответствующий элемент $(\mathbf{x}_{r(q)}, \mathbf{y}_{r(q)})$ из обучающей выборки данных.
- 5: Вычислить выходные векторы слоев сети $\mathbf{a}^{(0)}, \mathbf{a}^{(1)}, \dots, \mathbf{a}^{(L-1)}$, соответствующие входному вектору сети $\mathbf{x}_{r(q)}$, используя функцию forward из алгоритма 1.
- 6: Вычислить вектор ошибки последнего слоя $\boldsymbol{\delta}^{(L-1)} = \frac{\partial C_x}{\partial \mathbf{a}^{(L-1)}} \odot \sigma'(\mathbf{z}^{(L-1)})$.
- 7: **for** $l = L - 2, L - 3, \dots, 1$ **do**
- 8: Вычислить вектор ошибки l -ого слоя $\boldsymbol{\delta}^{(l)} = \left((\mathbf{W}^{(l+1)})^T \boldsymbol{\delta}^{(l+1)} \right) \odot \sigma'(\mathbf{z}^{(l)})$.
- 9: Обновить значение $\Delta\mathbf{b}^{(l)} = \Delta\mathbf{b}^{(l)} + \boldsymbol{\delta}^{(l)}$.
- 10: Обновить значение $\Delta\mathbf{W}^{(l)} = \Delta\mathbf{W}^{(l)} + \boldsymbol{\delta}^{(l)} (\mathbf{a}^{(l-1)})^T$.
- 11: **end for**
- 12: **end for**
- 13: **for** $l = 1, 2, \dots, L - 1$ **do**
- 14: Обновить значение $\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \frac{\eta}{M^{(mb)}} \Delta\mathbf{b}^{(l)}$.
- 15: Обновить значение $\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \frac{\eta}{M^{(mb)}} \Delta\mathbf{W}^{(l)}$.
- 16: **end for**

Результат: обновленные матрицы весов нейронов $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$ и векторы смещений нейронов $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$.

ному взвешенному выходу нейрона (или всего слоя).

Фактически для работы алгоритма необходима только частная производная функции стоимости по выходному вектору сети $\frac{\partial C_x}{\partial \mathbf{a}^{(L-1)}}$, соответственно, при программной реализации алгоритма необходима функция `cost_function_der_a(a, y)`, которая по заданному выходному вектору сети и по желаемому выходу (который известен для данных из обучающей выборки), вычисляет соответствующую частную производную.

В соответствии с представленным в разделе 1 алгоритмом обучения ИНС, гиперпараметрами сети являются:

- число слоев (L),

Алгоритм 3 Алгоритм обучения ИНС (learning).

Входные данные:

число слоев сети L ;
количества нейронов в слоях $N^{(0)}, N^{(1)}, \dots, N^{(L-1)}$;
матрицы весов нейронов $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$;
векторы смещений нейронов $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$;
функции активации для каждого из слоев $\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(L-1)}$;
функция стоимости одного измерения C_x ;
размер подвыборки $M^{(mb)}$;
параметр скорости обучения η ;
набор обучающих данных $(\mathbf{x}_q, \mathbf{y}_q)$ ($q = 0, 1, \dots, M^{(trn)} - 1$);
набор тестовых данных $(\mathbf{x}_t, \mathbf{y}_t)$ ($t = 0, 1, \dots, M^{(tst)} - 1$);
максимальное количество эпох обучения $M^{(ep)}$;

- 1: Создать пустой список acc для значений точности сети на каждой из эпох.
- 2: **for** $e = 0, 1, \dots, M^{(ep)} - 1$ **do**
- 3: Провести одну эпоху обучения на обучающем наборе данных, используя функцию backward из алгоритма 2.
- 4: Создать и вещественную переменную $count$ и задать $count = 0$.
- 5: **for** $t = 0, 1, \dots, M^{(tst)} - 1$ **do**
- 6: Вычислить выходной вектор сети $\mathbf{a}^{(L-1)}$, соответствующий входному вектору сети \mathbf{x}_t из тестового набора данных, используя функцию forward из алгоритма 1.
- 7: **if** $\text{then } \mathbf{a}^{(L-1)}$ равно \mathbf{y}_t $count = count + 1$
- 8: **end if**
- 9: **end for**
- 10: Добавить в список acc процент правильных ответов сети на данной эпохе $\frac{count}{M^{(tst)}}$.
- 11: **end for**

Результат: Матрицы весов нейронов $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$ и векторы смещений нейронов $\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$ обученной сети, а также список точностей на каждой из эпох обучения acc .

- количество нейронов в каждом из внутренних слоев⁽²¹⁾ ($N^{(1)}, N^{(2)}, \dots, N^{(L-2)}$),
- метод выбора начальных (случайных) значений весов нейронов ($\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(L-1)}$) и смещений нейронов ($\mathbf{b}^{(1)}, \mathbf{b}^{(2)}, \dots, \mathbf{b}^{(L-1)}$),
- вид функции активации каждого из слоев, кроме входного ($\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(N-1)}$),
- вид функции стоимости отдельного измерения (C_x),
- размер подвыборки ($M^{(mb)}$),
- параметр скорости обучения (η),
- количество эпох обучения ($M^{(ep)}$).

⁽²¹⁾ Количество нейронов во входном слое ($N^{(0)}$) определяется фиксированной размерностью входного вектора, а количество нейронов в выходном слое ($N^{(L-1)}$) – фиксированной размерностью выходного вектора.

Отметим, что в данной работе нами были выбраны популярные и широко используемые сигмоидальная (логистическая) функция активации вида

$$\sigma(z) = \frac{1}{1 + e^{-z}},$$

и квадратичная функция стоимости вида

$$C_x = \frac{1}{2} \|\mathbf{y} - \mathbf{a}^{(L-1)}\|^2 = \frac{1}{2} \sum_{j=0}^{N^{(L-1)}-1} (\mathbf{y}[j] - \mathbf{a}^{(L-1)}[j])^2. \quad (18)$$

3.3 Программный пакет для распознавания САРТСНА

Для автоматического распознавания САРТСНА был создан программный пакет CapSolver, включающий в себя следующие основные компоненты:

- модуль, реализующий полносвязную ИНС, включая процесс обучения и функционирования (папка *neural_network*);
- модуль, реализующий загрузку изображений (из тестовой базы, из файла и из сети Интернет), их обработку и отображение (папка *image*);
- браузерное расширение (плагин) для автоматического обнаружения поддерживаемой САРТСНА на веб-странице, ее отправки на локальный сервер и автоматической вставки результата распознавания в соответствующее поле веб-страницы (папка *cap_plugin*);
- локальный сервер, осуществляющий связь с браузерным расширением (файл *server*);
- обработчик клиент-серверных сообщений (файл *message*);
- главный обработчик, взаимодействующий с локальным сервером и запускающий процесс распознавания САРТСНА по запросу браузерного расширения (файл *manager*);
- рабочие ipython-блокноты для локального обучения, сохранения и тестирования ИНС для распознавания цифр из базы MNIST и распознавания цифровой САРТСНА (файл *cap_recognition*);
- различные вспомогательные функции (файл *utils*).

На рисунке 20 схематически представлен принцип работы программного комплекса. При загрузке в веб браузере страницы сайта, содержащей поддерживаемую САРТСНА, в автоматическом режиме происходит ее обнаружение, и на локальный сервер посредством технологии веб сокетов отправляется ссылка на изображение с указанием его типа. Сервер при получении очередного сообщения передает его обработчику, который в свою очередь передает изображение модулю обработки изображения. Здесь происходит удаление шумов, бинаризация и сегментация изображения на отдельные символы. Далее символы последовательно передаются на распознавание предварительно обученной искусственной нейронной сети. Результаты распознавания собираются обработчиком (менеджером) и далее при помощи сервера отправляются в плагин (браузерное расширение) по сети. Полученный результат распознавания плагин автоматически вставляет в соответствующее поле веб страницы, предназначенное для ответа на заданную САРТСНА. При этом в текущей реализации время необходимое на распознавание символов и на совершение передачи на сервер и обратно в сумме составляет не более секунды. Для возможности работы плагины

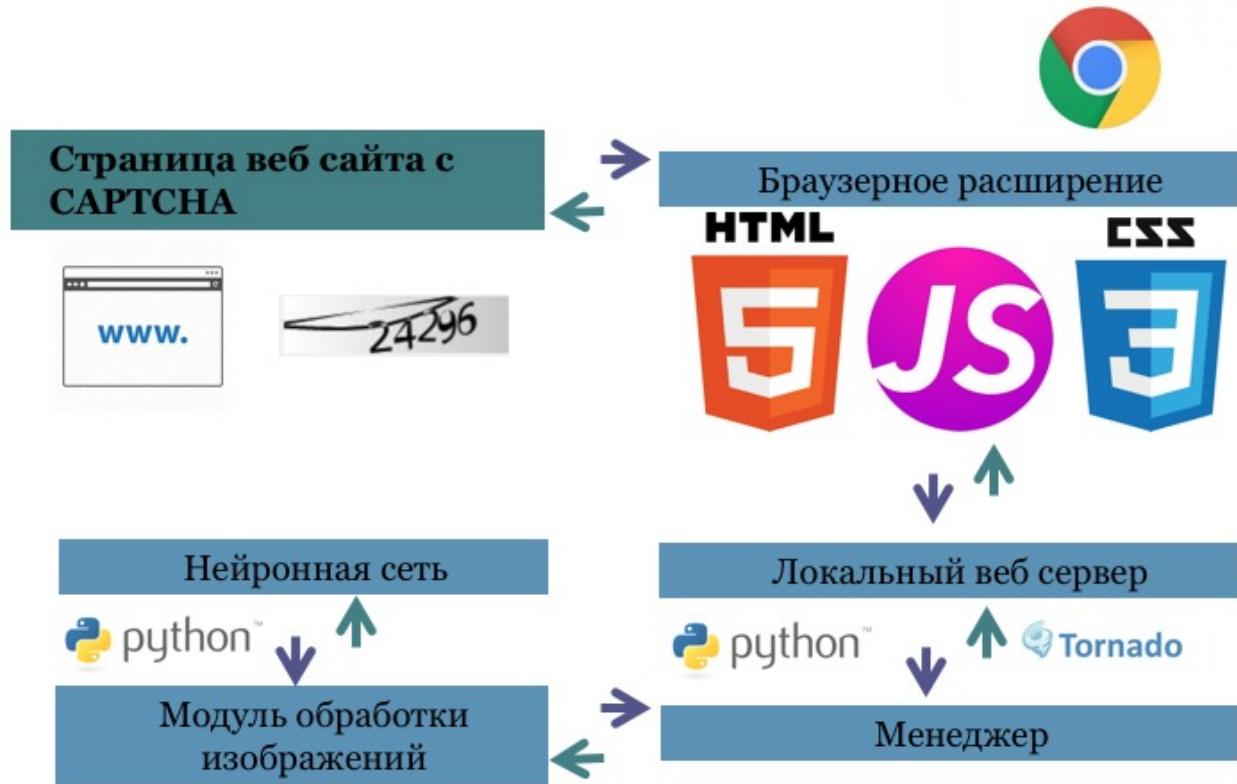


Рис. 19: Схематическое изображение логики работы разработанного программного комплекса CapSolver.

и взаимодействия с локальным сервером при использовании протокола https на целевой странице, был реализован самоподписанный сетевой сертификат.

На всех этапах работы программного комплекса организован контроль ошибок. При этом роль управляющего центра в текущей реализации играет сервер: при возникновении исключений на стороне браузерного расширения, на сервер отправляется соответствующее сообщение с описанием ошибки, и далее сервер принимает решение о продолжении работы либо об окончании текущего сокет-соединения. При невозможности отправки сообщения на сервер, браузерное расширение отправляет сообщения в интерактивную консоль браузера.

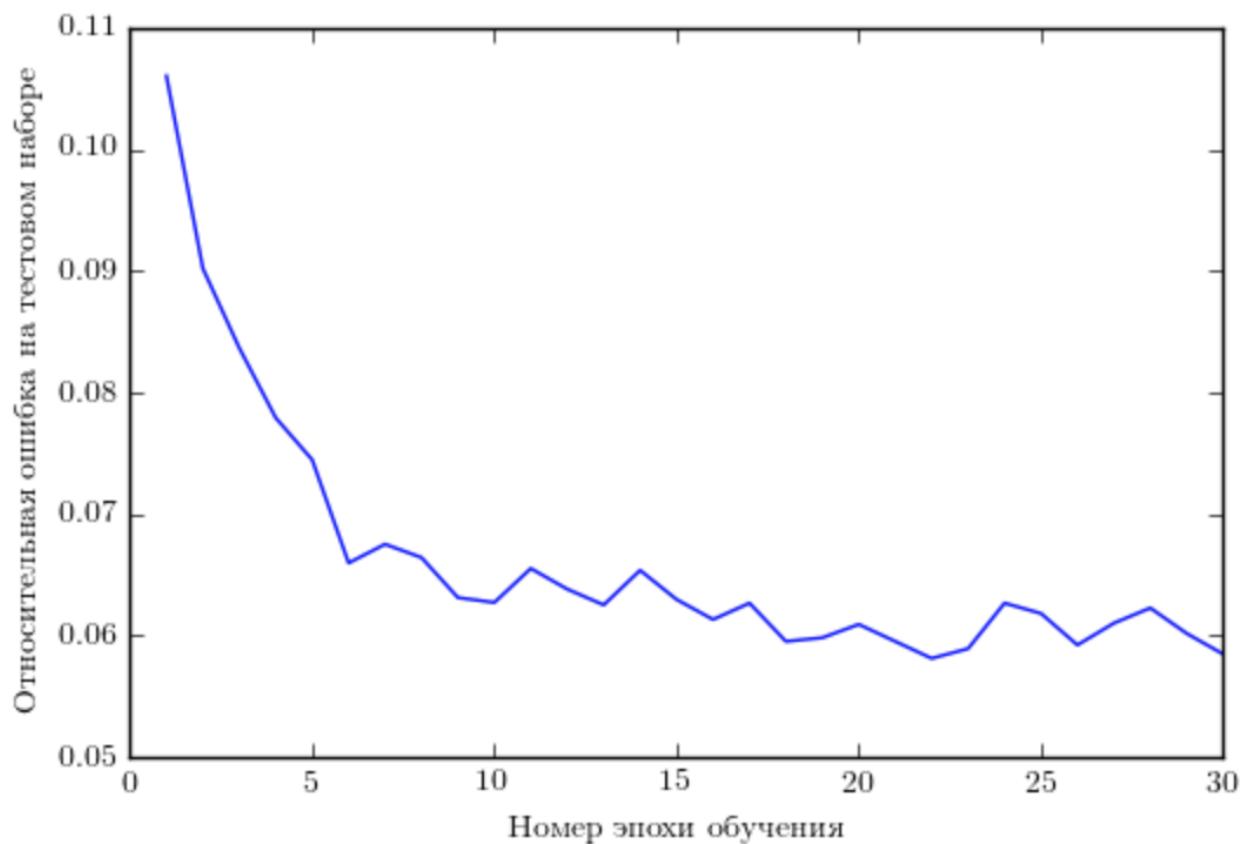


Рис. 20: Относительная ошибка полносвязной ИНС при обучении на выборке из базы MNIST.

Форма регистрации на сайте. Поля для ввода логина и пароля скрыты. Поле для ввода CAPTCHA содержит значение **8093**. Кнопка **Зарегистрироваться** имеет зеленый фон. Важное уведомление о согласии с правилами сайта расположено внизу страницы.

Логин *

Пароль *

Пароль *

8093

8093

Зарегистрироваться

Я заявляю предприятие, я принимаю и несу ответственность за выполнение [Правил сайта](#)

Рис. 21: Пример поля для ввода CAPTCHA, к которой применялось автоматическое распознавание.

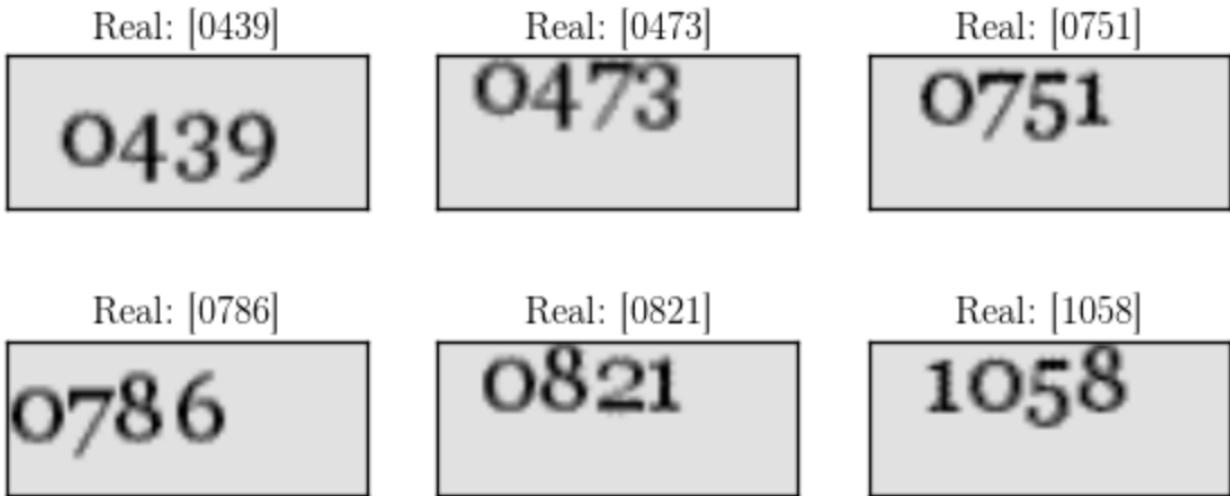


Рис. 22: Пример распознанных вручную CAPTCHA.

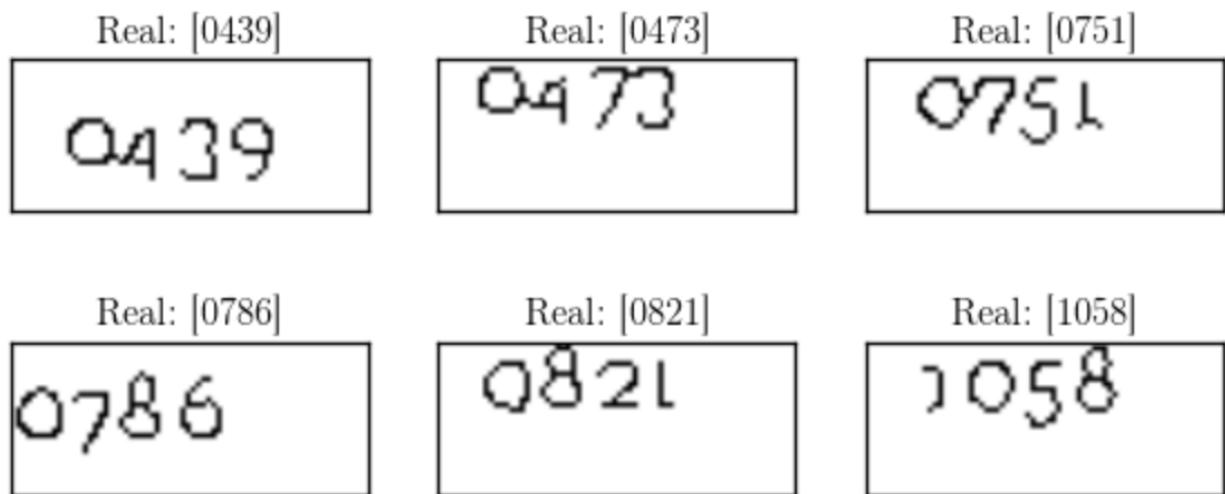


Рис. 23: Пример CAPTCHA уточненных до одного пикселя.

3.4 Результаты практического применения программного пакета

Для тестирования созданного программного комплекса нами было рассмотрено два класса задач. Разработанная архитектура полносвязной искусственной нейронной сети и соответствующих алгоритмов очистки изображений от шума были протестированы на базе данных из предварительно распознанных рукописных символов MNIST. Данная база содержит 7000 распознанных изображений рукописных цифр, написанных различными людьми. В первой части работы нами уже обсуждалась данная база и приводились примеры соответствующих изображений, поэтому здесь мы данное рассмотрение опускаем.

Нейронная сеть обучалась на 50000 тестовых изображениях, для тестирования результатов обучения использовалось 10000 изображений на этапе валидации, и затем 20000 изображений. Для оптимизации выбора гиперпараметров сети нами были проведена ее валидация на наборе из 10000 размеченных изображений. Итоговый выбор, комбинирующий удовлетворительную точность предсказаний при адекватном времени обучения (на

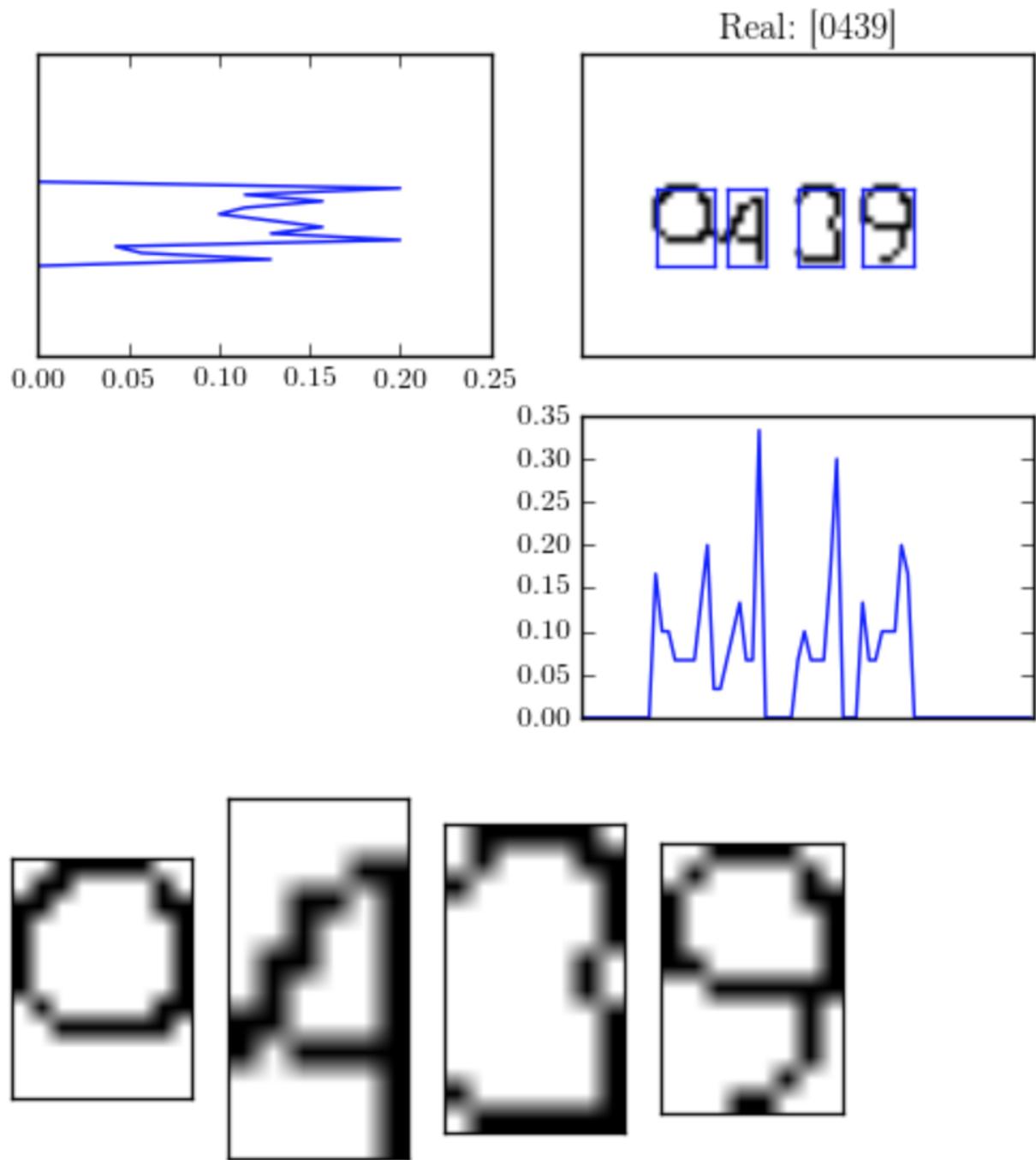


Рис. 24: Пример сегментированной CAPTCHA.

настольном компьютере с использование CPU) был: 3.0 для параметра скорости обучения, 1 внутренний слой сети с 30 нейронами, размер подвыборки для метода стохастического градиентного спуска был выбран равным 10, а обучение происходило на протяжении 30 эпох.

На рисунке 20 представлен график зависимости относительной доли ошибок сети на тестовом наборе данных в зависимости от текущей эпохи обучения. В Приложении Б данной работы более подробно представлена практическая реализации обучения сети.

Для тестирования работы программного комплекса с реальными изображениями CAPTCHA,

нами был выбран иностранный новостной ресурс <https://www.0629.com.ua/catalog/add>, требующий ввода циферной CAPTCHA при регистрации либо отправке комментария к новости. На рисунке 21 приводится вид данной CAPTCHA, а на рисунке 22 несколько примеров распознанных вручную CAPTCHA.

Обучение нейросетевого классификатора производилось по самостоятельно размеченной выборке из 1000 символов. Для успешного автоматического распознавания данного вида CAPTCHA необходимо произвести последовательно:

1. определить фоновые и текстовые наборы цветов и произвести бинаризацию изображения;
2. произвести утоньшение (скелетонизацию изображения) с использованием алгоритмов из раздела 2 данной работы (Zhang-Suen thinning algorithm), см. рис. 23;
3. произвести сегментацию полученного изображения, см. рис. 24 - для данного типа CAPTCHA нами использовался простой алгоритм обрезания по порогу гистограммы, дающий точность около 85%;
4. полученные в результате сегментации изображения распознаются полно связной искусственной нейронной сетью по аналогии с изображениями из базы MNIST, при этом мы имеем точность около 80% для отдельного символа.

ЗАКЛЮЧЕНИЕ

В работе нами были рассмотрены современные методы машинного обучения и интеллектуального анализа данных в контексте применения к задачам автоматического распознавания текстовой информации на графических изображениях. В качестве основного метода нами был выбран нейросетевой подход, а в качестве задачи распознавания рассматривалось автоматическое распознавание («взлом») САРТСНА.

Как было продемонстрировано в работе, наиболее перспективным на сегодняшний день направлением в области машинного обучения является метод искусственных нейронных сетей. Регулярно публикуется большое число работ, посвященных новым архитектурам сетей и модификациям алгоритмов обучения. Широкое распространение, особенно в контексте задачи обработки графических изображений, за последние несколько лет получили глубокие сверточные нейронные сети, которые были описаны в данной работе.

Одним из основных преимуществ метода искусственных нейронных сетей является практически полная универсальность, то есть единый подход к задачам из различных предметных областей: сбор размеченной обучающей выборки данных, выбор архитектуры и других гиперпараметров сети и, собственно, обучение сети в автоматическом режиме. Появление больших объемов размеченных данных и мощных вычислительных средств обусловило возможность реального применения нейросетевого подхода в практических приложениях, включая задачи обработки изображений и текста, автоматизированные системы поддержки принятия решений в технических и медицинских приложениях и т.д.

Современные глубокие искусственные нейронные сети насчитывают миллионы параметров и требуют для своего обучения недели и месяцы. Дальнейшее развитие метода искусственных нейронных сетей, как обсуждалось в работе, связано с созданием новых архитектур и алгоритмов обучения, позволяющих снизить вычислительную сложность и потребляемую память при обучении и функционировании сети. Многообещающим подходом является, в частности, метод тензоризации сетей, в котором матрицы весов нейронных слоев представляются в малопараметрическом (сжатом) формате. Подобные подходы позволяют использовать обученные сети на мобильных и настольных устройствах без необходимости обращения к серверу, что приведет к еще большему распространению интеллектуальных программных систем (персональные помощники такие как Siri, локальные приложения для обработки изображений, системы для построения оптимальных маршрутов и др.).

Нейросетевой подход в применении к задачам распознавания САРТСНА позволяет получать удовлетворительные результаты, как следует из анализа литературы, проведенного в работе. Так для ряда типов САРТСНА успешность распознавания может доходить до 90%. С научной точки зрения, основная цель при построении подобных автоматических систем распознавания – это выявление узких мест существующих алгоритмов распознавания и выработка рекомендаций по их улучшению. По результатам проведенного

в данной работе анализа литературы и практической реализации автоматической системы распознавания можно отметить, что при реализации CAPTCHA для повышения ее взломоустойчивости необходимо

- в качестве текста использовать сочетания из цифровых и буквенных (в обоих регистрах) символов;
- избегать использования в качестве текста существующих словарных слов;
- использовать трансформации символов, в частности, повороты, неоднородные иска-
жения и смещения по вертикали;
- реализовывать слабо повторяющееся расположение символов по горизонтали, а так-
же использовать тексты переменной длины;
- использовать шумовые линии, пересекающие текст CAPTCHA, а также выходящие
за пределы текста.

Отметим, что в задаче разработки CAPTCHA помимо требования устойчивости к ав-
томатизированному взлому, возникает второе, не менее важное требование – возможность
быстрого распознавания CAPTCHA человеком. То есть CAPTCHA должна легко разга-
дываться человеком, но иметь высокую сложность для автоматического распознавания
роботом. И в этом контексте, сформулированные выше требования по взломоустойчиво-
сти часто не могут быть применены в полной мере ввиду усложнения итоговой CAPTCHA
для восприятия человеком. На сегодняшний день начинают получать распространение
системы аналогичные по задаче CAPTCHA, но построенные на основе интеллектуаль-
ного анализа (с использованием нейросетевого подхода) истории поисковых запросов и
иной информации о конкретном пользователе для принятия окончательного решения че-
ловек/робот. Однако пока что такие системы не являются достаточно надежными и в
большинстве случаев все равно требуют разгадывания CAPTCHA (при сомнениях в «че-
ловечности» посетителя веб-ресурса).

На основе нейросетевого подхода в работе был разработан программный комплекс
для автоматического распознавания CAPTCHA. В данном комплексе были использова-
ны полносвязные искусственные нейронные сети и ряд алгоритмов для предварительной
обработки и сегментации изображений с текстовой информацией. Для тестирования ком-
плекса нами была рассмотрена задача распознавания рукописных цифр из популярной
базы MNIST, содержащей 60000 изображений цифр, написанных различными людьми.
Точность распознавания на тестовой выборке из 10000 изображений составила 95%.

Верификация комплекса проводилась на задаче автоматического распознаванияци-
фровой CAPTCHA. Точность сегментации изображения на отдельные символы в данном
случае составила 30%, а точность распознавания отдельного символа 70%. Для возмож-
ности практического использования данного комплекса было разработано браузерное рас-
ширение (plugin), которое автоматически обнаруживает поддерживаемую CAPTCHA на

веб-странице, отправляет изображение на локальный сервер и автоматически вводит полученный результат распознавания в соответствующее поле. Отметим, что несмотря на недостаточно высокую точность распознавания в текущей версии программного комплекса, он может быть удобным для ускорения процесса разгадывания CAPTCHA человеком: пользователю достаточно просто при необходимости подкорректировать результат автоматического распознавания, то есть комплекс в данном случае выступает как система типа «второе мнение».

Проделанная работа имеет ряд естественных направлений дальнейшего развития, включая:

1. улучшение используемой нейросетевой архитектуры, в частности, переход к глубоким сверточным искусственным нейронным сетям;
2. развитие алгоритмов предобработки и сегментации изображений;
3. обучение комплекса распознаванию различных типов распространенных CAPTCHA;
4. развитие созданного браузерного расширения, в частности, дополнение его развитым пользовательским интерфейсом, добавление системы аутентификации пользователя и т.д.;
5. применение разработанного подхода к распознаванию иных типов графических изображений, включая тексты, автомобильные номера, дорожные знаки и др.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.
- [2] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 294–311. Springer, 2003.
- [3] Donald Olding Hebb. The organization of behavior: A neuropsychological theory. Psychology Press, 2005.
- [4] Саймон Хайкин. Нейронные сети: полный курс, 2-е издание. Издательский дом Вильямс, 2008.
- [5] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943.
- [6] Guido Van Rossum and Fred L Drake. The python language reference manual. Network Theory Ltd., 2011.
- [7] Dave Kuhlman. A Python Book: Beginning Python, Advanced Python, and Python Exercises. Dave Kuhlman, 2009.
- [8] Vern Ceder. The quick python book. Manning Publications Co., 2010.
- [9] N Vinner. Cybernetics, 1948.
- [10] Donald Olding Hebb. The organization of behavior: A neuropsychological approach. John Wiley & Sons, 1949.
- [11] DRGHR Williams and GE Hinton. Learning representations by back-propagating errors. Nature, 323(6088):533–538, 1986.
- [12] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559–572, 1901.
- [13] Harold Hotelling. Analysis of a complex of statistical variables into principal components. Journal of educational psychology, 24(6):417, 1933.
- [14] K Karhunen. Über lineare methoden in der wahrscheinlichkeitsrechnung, vol. 37. Annales AcademiæScientiarum Fennicæ, Ser. A. I, 1947.
- [15] M Loeve. Fonctions aldatories de seconde ordre. Hermann, Paris, 1948.

- [16] Sasan Karamizadeh, Shahidan M Abdullah, Azizah A Manaf, Mazdak Zamani, and Alireza Hooman. An overview of principal component analysis. *Journal of Signal and Information Processing*, 4(03):173, 2013.
- [17] Weihong Deng, Jian Hu, Jiwen Lu, and Jun Guo. Transform-invariant pca: A unified approach to fully automatic facealignment, representation, and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1275–1284, 2014.
- [18] John R Koza. Genetic programming ii: Automatic discovery of reusable subprograms. *Cambridge, MA, USA*, 1994.
- [19] Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. *Genetic programming: an introduction*, volume 1. Morgan Kaufmann San Francisco, 1998.
- [20] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [21] Mantas Paulinas and Andrius Ušinskas. A survey of genetic algorithms applications for image enhancement and segmentation. *Information Technology and control*, 36(3), 2015.
- [22] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.
- [23] Henry A Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on pattern analysis and machine intelligence*, 20(1):23–38, 1998.
- [24] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.
- [25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [26] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.
- [27] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010.

- [28] Robert D Dony and Simon Haykin. Neural network approaches to image compression. *Proceedings of the IEEE*, 83(2):288–303, 1995.
- [29] John G Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988.
- [30] Viren Jain and Sebastian Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009.
- [31] WQ Yang and Lihui Peng. Image reconstruction algorithms for electrical capacitance tomography. *Measurement science and technology*, 14(1):R1, 2002.
- [32] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. [arXiv preprint arXiv:1502.04623](https://arxiv.org/abs/1502.04623), 2015.
- [33] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *Int. Conf. on Machine Learning (ICML)*, 2016.
- [34] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [35] Roger Fletcher and Michael JD Powell. A rapidly convergent descent method for minimization. *The computer journal*, 6(2):163–168, 1963.
- [36] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT’2010*, pages 177–186. Springer, 2010.
- [37] Gaihuan An and Wanjun Yu. Captcha recognition algorithm based on the relative shape context and point pattern matching. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2017 9th International Conference on*, pages 168–172. IEEE, 2017.
- [38] Rafaqat Hussain, Hui Gao, Riaz Ahmed Shaikh, and Shazia Parveen Soomro. Recognition based segmentation of connected characters in text based captchas. In *Communication Software and Networks (ICCSN), 2016 8th IEEE International Conference on*, pages 673–676. IEEE, 2016.
- [39] Ye Wang, Yuanjiang Huang, Wu Zheng, Zhi Zhou, Debin Liu, and Mi Lu. Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based captcha. In *Industrial Technology (ICIT), 2017 IEEE International Conference on*, pages 980–985. IEEE, 2017.

- [40] Rafaqat Hussain, Hui Gao, and Riaz Ahmed Shaikh. Segmentation of connected characters in text-based captchas for intelligent character recognition. *Multimedia Tools and Applications*, pages 1–15, 2016.
- [41] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2003.
- [42] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.
- [43] Devinder Singh and Baljit Singh Khehra. Digit recognition system using back propagation neural network. *International Journal of Computer Science and Communication*, 2(1):197–205, 2011.
- [44] <https://www.graphcore.ai/blog/what-does-machine-learning-look-like>.
- [45] U Ravi Babu, Aneel Kumar Chinthia, and Y Venkateswarlu. Handwritten digit recognition using structural, statistical features and k-nearest neighbor classifier. *International Journal of Information Engineering and Electronic Business*, 6(1):62, 2014.
- [46] Louisa Lam, Seong-Whan Lee, and Ching Y Suen. Thinning methodologies-a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 14(9):869–885, 1992.
- [47] MATÚŠ GRAMBLÍČKA and JOZEF VASKÝ. Comparison of thinning algorithms for vectorization of engineering drawings. *Journal of Theoretical and Applied Information Technology*, 94(2), 2016.
- [48] TY Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.
- [49] Kazuhisa Fujita. Extract an essential skeleton of a character as a graph from a character image. *arXiv preprint arXiv:1506.05068*, 2015.
- [50] Iping Supriana and Mastur Jaelani. Handwriting recognition using river-lake skeletonization with combination of structural elements similarity classification. In *Electrical Engineering and Computer Science (ICEECS), 2014 International Conference on*, pages 47–52. IEEE, 2014.
- [51] Bernd Fritzke et al. A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7:625–632, 1995.
- [52] Godfried T Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980.

- [53] Ju Jia Zou. A fast skeletonization method. In *Proc. 5th Digital Image Computing Techniques and Applications*, 2003.
- [54] Ju Jia Zou and Hong Yan. Skeletonization of ribbon-like shapes based on regularity and singularity analyses. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(3):401–407, 2001.
- [55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.
- [56] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.
- [57] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.
- [58] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.
- [59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
- [61] Alexander Novikov, Dmitrii Podoprikin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.
- [62] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.
- [63] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

- [64] Dong Yu, Li Deng, and Frank Seide. Large vocabulary speech recognition using deep tensor neural networks. In INTERSPEECH, pages 6–9, 2012.
- [65] Dong Yu, Li Deng, and Frank Seide. The deep tensor neural network with applications to large vocabulary speech recognition. IEEE Transactions on Audio, Speech, and Language Processing, 21(2):388–396, 2013.
- [66] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:1511.06530, 2015.
- [67] Majid Janzamin, Hanie Sedghi, and Anima Anandkumar. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. arXiv preprint arXiv:1506.08473, 2015.
- [68] Ivan Oseledets and Eugene Tyrtyshnikov. Tt-cross approximation for multidimensional arrays. Linear Algebra and its Applications, 432(1):70–88, 2010.
- [69] Ivan V Oseledets. Tensor-train decomposition. SIAM Journal on Scientific Computing, 33(5):2295–2317, 2011.
- [70] Dmitry Savostyanov and Ivan Oseledets. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. In Multidimensional (nD) Systems (nDs), 2011 7th International Workshop on, pages 1–8. IEEE, 2011.
- [71] Ivan V Oseledets. Approximation of $2^d \times 2^d$ matrices using tensor decomposition. SIAM Journal on Matrix Analysis and Applications, 31(4):2130–2145, 2010.
- [72] Тыртышников Е. Е. Матричный анализ и линейная алгебра. Физматлит М., 2007.
- [73] Gene Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis, 2(2):205–224, 1965.
- [74] Lapack library. <http://www.netlib.org/lapack/>.

ПРИЛОЖЕНИЕ А. Сингулярное разложение

Для вычисления собственных чисел и собственных векторов (спектра) симметричной матрицы [72], а также для построения малорангового (малопараметрического) приближения матрицы может быть использован метод сингулярного разложения (singular value decomposition, SVD) [73].

Для произвольной матрицы \mathbf{A} размера $m \times n$, заданной на поле вещественных или комплексных чисел, рассмотрим эрмитово сопряженную с ней матрицу \mathbf{A}^* , которая может быть получена из исходной матрицы путем ее транспонирования и замены всех элементов на комплексно сопряженные. Тогда матричное произведение $\mathbf{A}^* \mathbf{A}$ оказывается эрмитовой ($(\mathbf{A}^* \mathbf{A})^* = \mathbf{A}^* \mathbf{A}$) неотрицательно определенной ($\mathbf{x}^T \mathbf{A}^* \mathbf{A} \mathbf{x} \geq 0, \forall \mathbf{x} \in R^n$) матрицей размера $n \times n$. Как известно, все собственные значения такой матрицы неотрицательны, а собственные векторы образуют ортонормированный базис.

Сингулярными числами матрицы \mathbf{A} называют квадратные корни из собственных значений матрицы $\mathbf{A}^* \mathbf{A}$, отсортированных по неубыванию и обозначают σ_i или $\sigma_i(\mathbf{A})$, где $i = 1, \dots, n$. Имеем, соответственно,

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0.$$

В соответствии с данным представлением, матрица \mathbf{A} имеет ровно r ненулевых сингулярных чисел (r может в некоторых случаях совпадать с n).

Пусть $\mathbf{u}_1, \dots, \mathbf{u}_n$ - ортонормированный базис, состоящий из собственных векторов матрицы $\mathbf{A}^* \mathbf{A}$, причем, нумерация выбрана так, что

$$\begin{aligned} \mathbf{A}^* \mathbf{A} \mathbf{u}_i &= \sigma_i^2 \mathbf{u}_i, \quad 1 \leq i \leq r, \\ \mathbf{A}^* \mathbf{A} \mathbf{u}_i &= 0, \quad r + 1 \leq i \leq n. \end{aligned} \tag{19}$$

Отметим, что при $i \geq r + 1$ выполняется:

$$\mathbf{A}^* \mathbf{A} \mathbf{u}_i = 0 \longrightarrow \mathbf{u}_i^* \mathbf{A}^* \mathbf{A} \mathbf{u}_i = 0 \longrightarrow \mathbf{A} \mathbf{u}_i = 0. \tag{20}$$

Введем систему векторов

$$\mathbf{v}_i = \frac{\mathbf{A} \mathbf{u}_i}{\sigma_i}, \quad i = 1, \dots, r. \tag{21}$$

Для такой системы векторов выполняется:

$$\begin{aligned} \mathbf{v}_i^T \mathbf{v}_j &= 0, \quad 1 \leq i, j \leq r, \quad i \neq j, \\ \mathbf{v}_i^T \mathbf{v}_i &= 1, \quad 1 \leq i \leq r. \end{aligned} \tag{22}$$

Система из r векторов $\mathbf{v}_i, i = 1, \dots, r$ может в этом случае быть дополнена векторами $\mathbf{v}_{r+1}, \dots, \mathbf{v}_m$ до ортонормированного базиса в m -мерном пространстве.

Соответственно, с учетом 20 и 21 получаем:

$$\begin{aligned} A\mathbf{u}_i &= \mathbf{v}_i \times \sigma_i, \quad i = 1, \dots, r, \\ A\mathbf{u}_i &= \mathbf{v}_i \times 0, \quad i = r + 1, \dots, n. \end{aligned} \tag{23}$$

В матричной форме последнее соотношение может быть записано как

$$AU = VS, \tag{24}$$

где эрмитовы (в вещественном случае - ортогональные) матрицы U и V ($U^* = U^{-1}$ и $V^* = V^{-1}$) получены объединением столбцов, соответствующих системам векторов \mathbf{u}_i и \mathbf{v}_i , а S -диагональная матрица из сингулярных векторов, имеющая тот же размер, что и матрица A :

$$\begin{aligned} U_{n \times n} &= [\mathbf{u}_1 \cdots \mathbf{u}_n], \\ S_{m \times n} &= \text{diag}[\sigma_1 \cdots \sigma_n], \\ V_{m \times m} &= [\mathbf{v}_1 \cdots \mathbf{v}_m]. \end{aligned} \tag{25}$$

Соотношение 24 может быть переписано с учетом ортогональности матрицы U :

$$A = VSU^*. \tag{26}$$

Последнее соотношение обычно называют сингулярным разложением матрицы A , а столбцы матриц U и V называют левыми и правыми сингулярными векторами (а в совокупности - базисами) соответственно. Сингулярное разложение позволяет получать очень удобные представления для матриц и применяется в огромном количестве приложений линейной алгебры и вычислительной математики. Наиболее эффективный на сегодняшний день алгоритм вычисления сингулярного разложения предложен в работе [73] и реализован в стандартной библиотеке с операциями линейной алгебры LAPACK [74].

ПРИЛОЖЕНИЕ Б. Результаты обучения на базе MNIST

В данном Приложении мы приводим детали и результаты обучения полносвязной искусственной нейронной сети для распознавания рукописных символов из базы MNIST. Подробные комментарии по данной задаче сформулированы в разделе 3 работы.

Обучение нейронной сети для распознавания рукописных цифр на изображениях из базы MNIST

Импорт основных модулей

- Импортируется стандартный класс **numpy** для работы с векторами и матрицами
- Импортируется стандартный класс **time** для замера времени работы скриптов
- Выставляется путь к папке с кодом
- Импортируется класс **Img** для хранения и обработки изображения
- Импортируется класс **Img** для хранения массива всех изображений
- Импортируется класс **LayerFC**, представляющий один полно связанный слой нейронной сети
- Импортируется класс **NeuralNetwork**, представляющий нейронную сеть и функцию **nn_load** для загрузки параметров сохраненной на диске сети

```
In [2]: import numpy as np
import time
import sys
sys.path.append('../')

from cap_solver.image.img import Img
from cap_solver.image.imgs import Img
from cap_solver.neural_network.layer_fc import LayerFC
from cap_solver.neural_network.neural_network import NeuralNetwork,
nn_load
```

Функция, переводящая символ $(0, 1, \dots, 9)$ на изображении в вектор длины 10

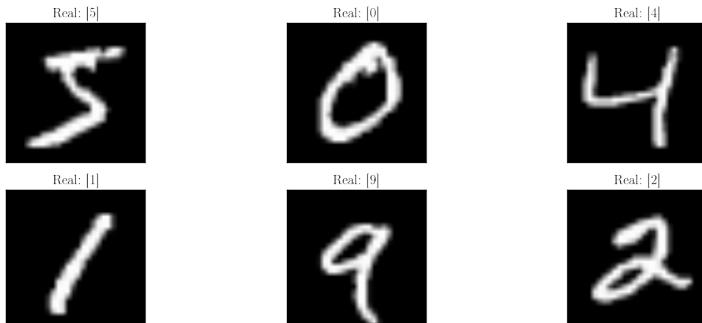
```
In [3]: def fsmb2vec(s):
    ''' Symbol may contain 0-9 numbers. '''
    v = np.zeros((10, 1))
    v[int(s)] = 1.
    return v
```

Загрузка изображений

- Инициализация класса, хранящего изображения и выставление опции печати промежуточной информации
- Распаковка изображений из архива (база mnist из 70000 распознанных рукописных цифр)
- Отрисовка шести первых изображений для примера

```
In [4]: Ims = Img(fsmb2vec=fsmb2vec, verb=True)
Ims.load('./data/mnist/mnist.pkl.gz', dtype='mnist')
Ims.show(n=6, c=3, figsize=(13, 5), figsize_sub=(4, 4))
```

```
Total time (sec.):      3.9
Time per img (sec.):    0.0001
Total number of img:    70000
```



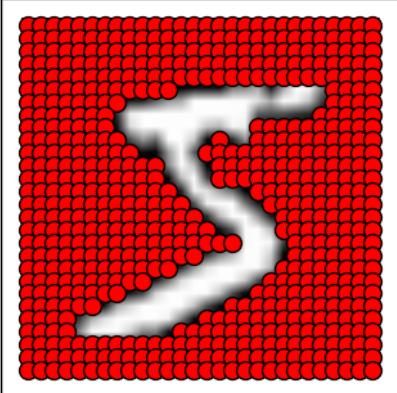
Анализ изображений (на примере первого изображения в базе)

- Вычисление и отображение (в виде списка из пар: номер цвета - количество пикселей) гистограммы цветов (функция calc_hist)
- Явное отображение (в виде красных кругов) первых двух по распространенности на изображении цветов (функция show_colors)

```
In [5]: hist = Ims[0].calc_hist(present=True)
for h in hist[:2]:
    print 'Color: %-3d | Count: %-d' %tuple(h)
Ims[0].show_colors(colors=h, figsize=(4, 4))
```

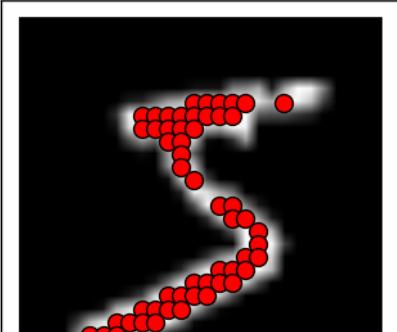
C: 0->	618	C:253->	54	C: 18->	5	C:
2->	3	C: 11->	3	C:136->	2	C:
C:154->	3	C: 1->	2	C:190->	2	C:
16->	2	C: 39->	2	C:241->	2	C:
C:172->	2	C:182->	2	C: 81->	2	C:
64->	2	C:195->	2	C: 219->	2	C:
C:198->	2	C: 80->	2	C: 225->	2	C:
82->	2	C:219->	2	C: 249->	2	C:2
C: 93->	2	C:225->	2	C:132->	1	C:1
47->	2	C:249->	2	C:133->	1	C:1
C: 3->	1	C: 9->	1	C:148->	1	C:1
C:130->	1	C: 14->	1	C:160->	1	C:
71->	1	C:150->	1	C: 27->	1	C:
C: 23->	1	C: 24->	1	C: 25->	1	C:
26->	1	C: 27->	1	C:160->	1	C:
C:156->	1	C: 30->	1	C:221->	1	C:
35->	1	C: 36->	1	C: 43->	1	C:
C: 70->	1	C:166->	1	C:175->	1	C:
70->	1	C: 43->	1	C:183->	1	C:
C: 45->	1	C: 46->	1	C:186->	1	C:187->
49->	1	C:183->	1	C:201->	1	C:
C: 56->	1	C:186->	1	C:205->	1	C:2
66->	1	C:201->	1	C:139->	1	C:
C:119->	1	C:205->	1	C: 55->	1	C:90->
07->	1	C:139->	1	C:226->	1	C: 229->
C:213->	1	C: 55->	1	C:212->	1	C:1
94->	1	C:226->	1	C:108->	1	C:242->
C:251->	1	C:212->	1	C:240->	1	C:2
07->	1	C:108->	1	C:114->	1	C:252->
C:238->	1	C:240->	1	C:255->	1	C:1
44->	1	C:114->	1	C:127->	1	C:
C:250->	1	C:255->	1	Color: 0 Count: 618		

Real: [5]



Color: 253 | Count: 54

Real: [5]



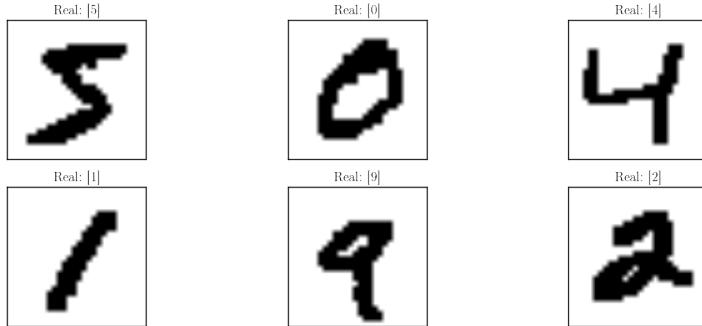


Бинаризация изображений

- На основе анализа гистограммы, наиболее представленный на изображении цвет считаем фоновым
- Все остальные цвета считаем соответствующими тексту
- Бинаризация: переводим цвет фона в нуль, а цвет текста в единицу (функция binarization)
- Отрисовка нескольких преобразованных изображений для примера

```
In [6]: _t = time.time()
for Im in Ims:
    Im.binarization(colors_bg=[0])
print 'Total time (sec.): %6.1f'%(time.time() - _t)
print 'Time per img (sec.): %9.4f'%((time.time() - _t)/Ims.len_)
Ims.show(n=6, r=2, figsize=(13, 5), figsize_sub=(4, 4))

Total time (sec.): 123.5
Time per img (sec.): 0.0018
```



Подготовка данных для обучения сети

- Инициализация класса, представляющего данные
- Преобразование подготовленных изображений в матрицу входных и выходных векторов
- Разделение изображений на набор обучающих данных (50000), валидационных данных (0) и тестовых данных (20000)

```
In [7]: _t = time.time()
X_all = Ims.get_matrix(var='arr')
Y_all = Ims.get_matrix(var='smb_real')
print X_all.shape
print Y_all.shape
N_trn = 50000
X_trn, Y_trn = X_all[:, :N_trn], Y_all[:, :N_trn]
X_tst, Y_tst = X_all[:, N_trn:], Y_all[:, N_trn:]
print 'Total time (sec.): %6.1f'%(time.time() - _t)
print 'Time per img (sec.): %9.4f'%((time.time() - _t)/Ims.len_)

(784, 70000)
(10, 70000)
Total time (sec.): 1.9
Time per img (sec.): 0.0000
```

Создание нейронной сети

- Инициализируется экземпляр класса нейронной сети и выставляется опция печати промежуточной информации
- Создаются три слоя с числами нейронов 784 (входной слой), 30 (внутренний слой), 10 (выходной слой)
- Гиперпараметр числа элементов в подвыборке выбран равным 10, а гиперпараметр скорости обучения выбран равным 3 (на основе ряда текстовых расчетов по выбору оптимального значения гиперпараметров)

```
In [8]: NN = NeuralNetwork(verb=True)
NN.add_layers(LayerFC(None, 784))
NN.add_layers(LayerFC(784 , 30))
NN.add_layers(LayerFC( 30 , 10))
NN.set_params(mb_size=10, eta=3.)
```

Обучение нейронной сети

- Выбрано 30 эпох для обучения
- Выходная информация на каждой эпохе обучения:
 - Номер эпохи
 - Время в секундах, затраченное на эпоху
 - Полное количество итераций обратного распространения в сети (с момента создания сети)
 - Количество данных в тестовом наборе
 - Качество сети после соответствующей эпохи обучения на тестовом наборе данных (относительная доля неправильных результатов предсказания)

```
In [9]: NN.learning(X_trn, Y_trn, X_tst, Y_tst, epochs=30)

Epoch #  1: T=    9.71; m=    50000; n_check=    20000; e_check=0.
106050
Epoch #  2: T=    7.81; m=   100000; n_check=    20000; e_check=0.
090200
Epoch #  3: T=    7.90; m=   150000; n_check=    20000; e_check=0.
083650
Epoch #  4: T=    8.70; m=   200000; n_check=    20000; e_check=0.
077900
Epoch #  5: T=    6.44; m=   250000; n_check=    20000; e_check=0.
074450
Epoch #  6: T=    6.51; m=   300000; n_check=    20000; e_check=0.
065950
Epoch #  7: T=    6.76; m=   350000; n_check=    20000; e_check=0.
067500
Epoch #  8: T=    7.67; m=   400000; n_check=    20000; e_check=0.
066400
Epoch #  9: T=    7.30; m=   450000; n_check=    20000; e_check=0.
063100

Epoch # 10: T=    7.39; m=   500000; n_check=    20000; e_check=0.
062700
Epoch # 11: T=    6.64; m=   550000; n_check=    20000; e_check=0.
065500
Epoch # 12: T=    6.99; m=   600000; n_check=    20000; e_check=0.
063800
Epoch # 13: T=    6.80; m=   650000; n_check=    20000; e_check=0.
062500
Epoch # 14: T=    8.26; m=   700000; n_check=    20000; e_check=0.
065350
Epoch # 15: T=    8.02; m=   750000; n_check=    20000; e_check=0.
062950
Epoch # 16: T=    7.40; m=   800000; n_check=    20000; e_check=0.
061300
Epoch # 17: T=    7.95; m=   850000; n_check=    20000; e_check=0.
062650
Epoch # 18: T=    6.88; m=   900000; n_check=    20000; e_check=0.
059500
Epoch # 19: T=    7.72; m=   950000; n_check=    20000; e_check=0.
059800
Epoch # 20: T=    6.94; m=  1000000; n_check=    20000; e_check=0.
060900
Epoch # 21: T=    7.96; m=  1050000; n_check=    20000; e_check=0.
059500
Epoch # 22: T=    8.90; m=  1100000; n_check=    20000; e_check=0.
058100
Epoch # 23: T=    8.78; m=  1150000; n_check=    20000; e_check=0.
058900
Epoch # 24: T=    7.82; m=  1200000; n_check=    20000; e_check=0.
062650
Epoch # 25: T=    9.76; m=  1250000; n_check=    20000; e_check=0.
061800
Epoch # 26: T=    8.44; m=  1300000; n_check=    20000; e_check=0.
059200
Epoch # 27: T=    6.83; m=  1350000; n_check=    20000; e_check=0.
061000
Epoch # 28: T=    6.27; m=  1400000; n_check=    20000; e_check=0.
062250
```

```

Epoch # 29: T= 8.46; m= 1450000; n_check= 20000; e_check=0.
060150
Epoch # 30: T= 9.92; m= 1500000; n_check= 20000; e_check=0.
058450

```

Комментарий: как следует из результатов, нейронная сеть после 30 эпох обучения дает менее 6% ошибок на тестовой выборке размера 20000 (качество предсказания может быть существенно улучшено путем лучшего выбора гиперпараметров сети).

Добавим еще один слой из 30 нейронов к сети и соответствующим образом модифицируем параметр скорости обучения и максимальное число эпох обучения

```

In [20]: NN = NeuralNetwork(verb=True)
NN.add_layers(LayerFC(None, 784))
NN.add_layers(LayerFC(784 , 30))
NN.add_layers(LayerFC( 30 , 30))
NN.add_layers(LayerFC( 30 , 10))
NN.set_params(mb_size=10, eta=1.5)
NN.learning(X_trn, Y_trn, X_tst, Y_tst, epochs=50)

Epoch # 1: T= 9.30; m= 50000; n_check= 20000; e_check=0.
122550
Epoch # 2: T= 12.27; m= 100000; n_check= 20000; e_check=0.
096000
Epoch # 3: T= 8.18; m= 150000; n_check= 20000; e_check=0.
091200
Epoch # 4: T= 7.97; m= 200000; n_check= 20000; e_check=0.
079100
Epoch # 5: T= 7.66; m= 250000; n_check= 20000; e_check=0.
076500
Epoch # 6: T= 10.33; m= 300000; n_check= 20000; e_check=0.
072750
Epoch # 7: T= 8.65; m= 350000; n_check= 20000; e_check=0.
072650
Epoch # 8: T= 7.59; m= 400000; n_check= 20000; e_check=0.
071700
Epoch # 9: T= 10.79; m= 450000; n_check= 20000; e_check=0.
071350
Epoch # 10: T= 7.56; m= 500000; n_check= 20000; e_check=0.
066400
Epoch # 11: T= 7.86; m= 550000; n_check= 20000; e_check=0.
067650
Epoch # 12: T= 7.54; m= 600000; n_check= 20000; e_check=0.
064750
Epoch # 13: T= 7.54; m= 650000; n_check= 20000; e_check=0.
061650
Epoch # 14: T= 7.74; m= 700000; n_check= 20000; e_check=0.
060700
Epoch # 15: T= 9.88; m= 750000; n_check= 20000; e_check=0.
061250
Epoch # 16: T= 9.33; m= 800000; n_check= 20000; e_check=0.
061150
Epoch # 17: T= 7.73; m= 850000; n_check= 20000; e_check=0.
061400
Epoch # 18: T= 7.69; m= 900000; n_check= 20000; e_check=0.
061150
Epoch # 19: T= 7.68; m= 950000; n_check= 20000; e_check=0.
060800
Epoch # 20: T= 7.54; m= 1000000; n_check= 20000; e_check=0.
059900
Epoch # 21: T= 7.55; m= 1050000; n_check= 20000; e_check=0.
061000
Epoch # 22: T= 7.55; m= 1100000; n_check= 20000; e_check=0.
062400
Epoch # 23: T= 7.55; m= 1150000; n_check= 20000; e_check=0.
058700
Epoch # 24: T= 7.58; m= 1200000; n_check= 20000; e_check=0.
059700
Epoch # 25: T= 9.09; m= 1250000; n_check= 20000; e_check=0.
056950
Epoch # 26: T= 8.40; m= 1300000; n_check= 20000; e_check=0.
057300
Epoch # 27: T= 7.85; m= 1350000; n_check= 20000; e_check=0.
058850
Epoch # 28: T= 7.52; m= 1400000; n_check= 20000; e_check=0.
057650
Epoch # 29: T= 7.54; m= 1450000; n_check= 20000; e_check=0.
058600
Epoch # 30: T= 7.78; m= 1500000; n_check= 20000; e_check=0.
058100
Epoch # 31: T= 7.56; m= 1550000; n_check= 20000; e_check=0.
059650
Epoch # 32: T= 8.04; m= 1600000; n_check= 20000; e_check=0.
057700
Epoch # 33: T= 8.95; m= 1650000; n_check= 20000; e_check=0.
057150
Epoch # 34: T= 9.23; m= 1700000; n_check= 20000; e_check=0.
056550

```

```
Epoch # 35: T= 9.57; m= 1750000; n_check= 20000; e_check=0.  
056550  
Epoch # 36: T= 8.76; m= 1800000; n_check= 20000; e_check=0.  
056750  
Epoch # 37: T= 9.53; m= 1850000; n_check= 20000; e_check=0.  
056600  
Epoch # 38: T= 8.09; m= 1900000; n_check= 20000; e_check=0.  
059150  
Epoch # 39: T= 8.55; m= 1950000; n_check= 20000; e_check=0.  
058650  
Epoch # 40: T= 8.83; m= 2000000; n_check= 20000; e_check=0.  
057800  
Epoch # 41: T= 9.11; m= 2050000; n_check= 20000; e_check=0.  
057650  
Epoch # 42: T= 10.49; m= 2100000; n_check= 20000; e_check=0.  
056900  
Epoch # 43: T= 11.11; m= 2150000; n_check= 20000; e_check=0.  
055750  
Epoch # 44: T= 11.01; m= 2200000; n_check= 20000; e_check=0.  
056300  
Epoch # 45: T= 10.49; m= 2250000; n_check= 20000; e_check=0.  
055550  
Epoch # 46: T= 10.89; m= 2300000; n_check= 20000; e_check=0.  
057500  
Epoch # 47: T= 10.48; m= 2350000; n_check= 20000; e_check=0.  
057800  
Epoch # 48: T= 10.40; m= 2400000; n_check= 20000; e_check=0.  
056350  
Epoch # 49: T= 10.05; m= 2450000; n_check= 20000; e_check=0.  
055850  
Epoch # 50: T= 10.56; m= 2500000; n_check= 20000; e_check=0.  
057650
```

Комментарий: как видим, точность существенно не увеличилась.

Проверка качества предсказания на конкретном примере

- Выбираем одно изображение (последнее изображение из набора) для примера и получаем соответствующие интенсивности пикселей (x)
- Запускаем расчет посредством обученной нейронной сети и выводим результат (предсказание сети)
- Отрисовываем изображение

```
In [13]: Im = Ims[-1]  
x = Im.get_vector('arr')  
a = NN.forward(x)  
print 'The answer of the neural network is ', np.argmax(a)  
Im.show(figsize=(4, 4))
```

The answer of the neural network is 6

Real: [6]



Комментарий: обученная нейронная сеть выдала правильное предсказание (цифра 6).

Сохраняем обученную нейронную сеть в файл для возможности последующего использования

```
In [14]: NN.save('./nn_saved/nn_mnist.p')
```

Загружаем (для проверки корректности опции сохранения) обученную нейронную сеть из файла и демонстрируем первый ошибочный результат распознавания

```
In [21]: NN = nn_load('./nn_saved/nn_mnist.p')

for i, Im in enumerate(ImS):
    x = Im.get_vector('arr')
    a = NN.forward(x)
    a = np.argmax(a)
    Im.smb_calc = unicode(a)
    if not Im.symb_is_corr:
        print 'NN result is "%s" (real value is "%s"). Image number
is %d'%(Im.smb_calc, Im.smb_real, i+1)
        break
Im.show(figsize=(4, 4))
```

NN result is "3" (real value is "9"). Image number is 49

Real: [9] | Pred: [3]



In []: