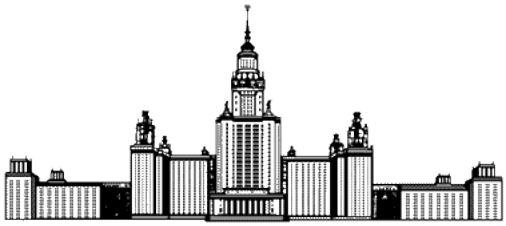


Московский государственный университет имени М. В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Козлов Владимир Дмитриевич

Алгоритмы оффлайн-распознавания рукописных цифр

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель
к. ф.–м. н, доцент
С. И. Гуров

Москва 2015

Оглавление

Аннотация

В работе рассматриваются основные методы оффлайн-распознавания рукописных символов. Предложен алгоритм оффлайн-распознавания рукописных цифр на основе метода опорных векторов, проведено тестирование разработанного алгоритма на базе рукописных цифр MNIST и на выборке, сформированной из рукописей Российского государственного архива литературы и искусства.

1 Введение

В настоящее время сканирование и сохранение в памяти компьютера текста с твердого носителя является решенной задачей. Это существенно облегчает задачу хранения рукописных и печатных текстов и предоставления доступа к ним различных пользователей. Однако полученный в результате сканирования текст хранится в памяти компьютера в виде изображения, чаще всего растрового, что делает работу с ним весьма сложной: затруднено ориентирование, практически невозможны редактирование, форматирование и поиск по тексту. Для решения этих задач необходимо провести процесс распознавания текста на изображении с созданием файла в том или ином текстовом формате. Перевод изображений рукописного, машинописного или печатного текста в текстовые данные называется распознаванием текста.

Задача распознавания машинописного текста носит название оптического распознавания символов (*optical character recognition*, *OCR*). В настоящее время существуют высокоточные системы для распознавания машинописных и рукопечатных текстов (например, ABBYY FineReader). Распознавание же рукописных текстов является гораздо более сложной и на данный момент не решённой задачей.

Задача распознавания рукописного текста носит название HWR (*handwriting recognition*). Существуют два различных класса задач HWR:

- онлайн-распознавание — распознавание текста ведётся параллельно с вводом текста;
- оффлайн-распознавание — распознавание текста ведётся на уже синтезированном изображении.

При онлайн-распознавании процесс формирования изображения текста и процесс его ввода в систему распознавания совмещены, что позволяет системе отслеживать процесс начертания символов. Это даёт возможность получать помимо графической информации ещё и информацию о структуре входных изображений, например, о направлении и скорости движения пера или о его нажиме при написании символа. На данный момент онлайн-системы распознавания широко используются на планшетных ПК.

В задаче оффлайн-распознавания, в отличие от предыдущей, системе доступна только графическая информация. Уже это делает её значительно труднее онлайновой.

Данная работа посвящена оффлайновому распознаванию рукописных цифр, что можно рассматривать как частный случай задачи распознавания рукописного текста.

Задача распознавания рукописных цифр является одной из классических задач распознавания образов и имеет значительную практическую ценность. Одним из первых практических применений методов распознавания рукописных цифр стало создание системы чтения ZIP-кодов почты США [1, 2]. Методы распознавания цифр применяются также для решения таких практических задач, как чтение банковских чеков [3], автоматизированное чтение анкет и др.

Цель работы — разработка алгоритма распознавания рукописных цифр, а также реализация его в виде приложения.

Задачи работы:

1) изучить существующие методы и алгоритмы оффлайн-распознавания рукописных символов и выявить основные факторы, оказывающие наибольшее влияние на точность распознавания;
2) предложить алгоритм оффлайн-распознавания рукописных цифр;
3) разработать программный продукт для оффлайн-распознавания рукописных цифр.

В настоящей работе использовались рукописи, предоставленные Российским государственным архивом литературы и искусства.

2 Постановка задачи

Оффлайн-распознавание заключается в распознавании уже сформированного изображения в текстовый формат. Отсутствие дополнительной информации о вводе текста вкупе с высокой вариабельностью распознаваемого объекта усложняет задачу оффлайнового распознавания текста в общем случае по сравнению с онлайновым. Неполный список проблем, типичных при распознавании рукописного текста оффлайн в общем случае, включает в себя:

- высокая вариативность начертания символов — по размеру, наклону, набору составных частей, связям между ними и др.;
- орфографические ошибки в тексте;
- специфические особенности начертания, не позволяющие уверенно разделять символы;
- пересечение элементов текста, наложение частей текста друг на друга;
- помарки, кляксы, исправления, дефекты носителя (бумаги), а также артефакты, возникающие при сканировании;
- непараллельность и неровность строк текста;
  и другие.

У задачи распознавания рукописных цифр есть несколько главных особенностей по сравнению с общей задачей распознавания текста. Во-первых, резко ограничен алфавит распознаваемых символов — рассматриваются только цифры от 0 до 9. Во-вторых, в строке все цифры имеют примерно одинаковую высоту. В-третьих, цифры, как правило, пишутся отдельно друг от друга, а их пересечения — скорее исключения. В совокупности эти особенности заметно упрощают этапы сегментации, извлечения признаков и классификации.

Общая постановка задачи. Дано изображение текста документа в произвольном графическом формате. Номер страницы документа подписан в правом верхнем углу страницы. Требуется найти этот номер страницы и распознать его.

Документ представляет собой рукописный текст, написанный на листе бумаги. Номер страницы представляет собой последовательность рукописных

цифр. Эту последовательность требуется выделить и по ней получить последовательность цифр — ASCII-символов.

Данное изображение последовательности рукописных цифр удовлетворяет следующим условиям:

1. Цифры темнее относительно фона.
2. На изображении нет других графических элементов, кроме распознаваемой последовательности цифр.
3. Цифры должны иметь примерно одинаковый размер и ориентацию, а также должны быть написаны в ряд.

3 Обзор существующих методов оффлайн-распознавания последовательностей рукописных символов

Распознавание рукописного текста проходит в несколько этапов [4]:

1. ***Предварительная обработка изображения*** *(preprocessing)*: на этом этапе происходит обработка изображения с целью повышения его качества и приведения его к виду, удобному для сегментации.
2. ***Сегментация*** *(segmentation)*: на этом этапе происходит выделение текста на изображении и его разделение на составные части. Обычно текст обрабатывается иерархически: сначала выделяются отдельные строки, затем отдельные слова, затем символы или части символов.
3. ***Извлечение признаков*** *(feature extraction)*: на этом этапе формируются признаковые описания выделенных на этапе сегментации частей.
4. ***Классификация*** *(classification)*: на этом этапе по признаковым описаниям, построенным на этапе извлечения признаков, система принимает решение о том, к какому заранее известному классу отнести выделенный на этапе сегментации элемент.
5. ***Обработка результатов*** *(postprocessing)*: на этом этапе происходит построение итогового текста по результатам классификации выделенных частей текста.

3.1 Предварительная обработка изображения

На этапе предварительной обработки производится *улучшение качества изображения* (*image enhancement*) — приведение изображения к виду, наиболее подходящему для дальнейшей автоматической работы с ним [5]. Улучшение качества изображения с целью дальнейшего распознавания текста включает в себя удаление дефектов изображения и отделение текста от фона [6].

Удаление дефектов. Удаление дефектов осуществляется стандартными методами обработки изображений. Наиболее часто для удаления шума используются фильтр Гаусса для подавления высокочастотного шума и медианный фильтр для удаления шума «соль и перец» [7].

Отделение текста от фона. Отделение текста от фона является частным случаем задачи выделения объекта на изображении. Задачу можно сформулировать следующим образом: по изображению текста A нужно построить бинарное изображение B того же размера такое, что

$$B(i,j) = \begin{cases} 1, & \text{если пиксель } (i,j) \text{ принадлежит тексту на } A, \\ 0, & \text{иначе.} \end{cases}$$

Такое преобразование изображение позволяет в дальнейшем использовать анализ связных компонент, контуров, скелетов и т.д.

Наиболее часто использующимся методом отделения текста от фона служит *пороговая бинаризация* (*threshold binarization*). Пусть дано изображение, $I(i,j)$ — яркость пикселя с координатами (i,j). Пороговой бинаризацией изображения называется попиксельное преобразование $f(i,j)$ такое, что

$$f(i,j) = \begin{cases} 1, & \text{если } I(i,j) \geq d, \\ 0, & \text{если } I(i,j) < d, \end{cases}$$

где d называется *порогом* бинаризации.

Обычно на гистограмме яркости изображения текста наблюдается два пика: высокий пик в области светлых пикселей (соответствует фону, то есть бумаге) и пик пониже в области тёмных (соответствует тексту). Поэтому задача поиска порогового значения яркости, т. е. такого, что пиксели с яркостью выше этого значения (фон) будут считаться чёрными, а ниже (текст) — белыми (такое «инвертирование» цвета делается в целях упрощения применения многих алгоритмов в дальнейшем), является задачей поиска оптимального значения между двумя пиками гистограммы. Для решения этой задачи существуют хорошо изученные методы, например, метод Оцу [8] и его вариации.

Результатом этапа предварительной обработки изображения является бинарное изображение белых цифр на чёрном фоне, соответствующих исходному изображению.

3.2 Сегментация

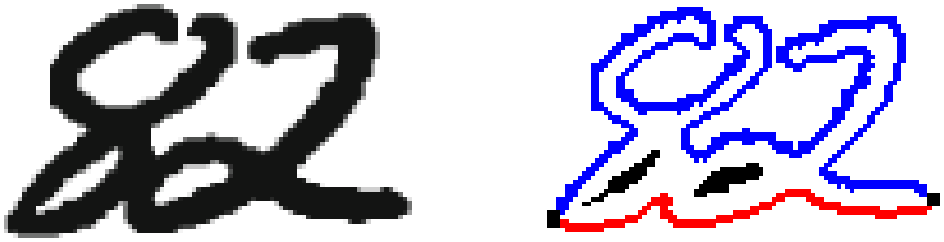
На этапе сегментации происходит выделение отдельных цифр на изображении, полученном на этапе предварительной обработки. Результатом этапа

сегментации должна быть последовательность изображений отдельных цифр, которые далее поступают на вход процедуре извлечения признаков. Такая стратегия, называемая *стратегией явной сегментации*, является более предпочтительной в задаче распознавания последовательностей рукописных цифр, чем неявная сегментация, при которой выделение цифр, извлечение признаков и классификация проходят параллельно [9].

Выделение групп цифр. После отделения текста от фона на этапе предварительной обработки для выделения групп цифр становится возможным использование анализа связных компонент. Если рассматривать белые пиксели как вершины графа, а рёбрами соединять вершины, соответствующие соседним пикселям, становится возможным выделять компоненты связности, соответствующие одной или нескольким пересекающимся цифрам. Для выделения связных компонент существуют стандартные и широко известные алгоритмы поиска, например, поиск в глубину и поиск в ширину.

Разделение сцепленных цифр. После выделения компоненты связности на графе изображения требуется принять решение, является ли эта компонента одной или несколькими цифрами, и если несколькими, как следует их разделить. Обычно сегментация компоненты связности на цифры происходит в два этапа: сначала генерируются возможные способы сегментации компоненты связности, а потом из них выбирается наилучший. Построение возможных способов сегментации обычно проводится эвристическими методами. Выбором наилучшего метода сегментации может заниматься как подсистема сегментации (в этом случае на вход подсистеме извлечения признаков и классификации поступает только один набор входных данных), так и подсистема классификации (в этом случае подсистемам извлечения признаков и классификации поступает на вход несколько наборов входных данных, и решение о том, какая сегментация была выполнена правильно, принимается по результатам распознавания).

Для построения явной сегментации чаще всего используется два вида представления данных: контур и скелет. *Контуром объекта* называется множество всех пикселей, принадлежащих объекту и имеющих соседний пиксель, не при-



(a) Компонента связности (b) Контур

Рис. 1: Компонента связности, состоящая из двух пересекающихся цифр (1a), и её контур (1b). Синим выделен верхний контур, красным — нижний.

надлежащий объекту. Крайне правая и крайне левая точки контура делят его на верхний и нижний контуры (см. рис. 1). Область выпуклости контура называется *долиной*, а область вогнутости — *холмом*.

Методы сегментации на основе анализа контуров ранее выделенных компонент связности приводятся в работах [10, 11, 12]. В этих методах на контурах компонент связности, исходя из некоторых соображений, выбираются потенциальные точки разреза, через которые затем проводятся линии разреза, удовлетворяющие определённым условиям. В работе [10] возможные точки разреза ставятся там, где расстояние между верхним и нижним контурами превосходит порог, а наилучший разрез выбирается как кратчайший путь в графе с вершинами на возможных точках разреза. В работе [11] точки возможного разреза выбираются эвристически как точки, в которых контур делает наибольший поворот направо при обходе против часовой стрелки.

3.3 Извлечение признаков

Пусть $\mathbf{X}$ — множество рассматриваемых объектов, $\mathbf{Y}$ — конечное множество. Существует функция $\mathbf{y} : \mathbf{X} \to \mathbf{Y}$, значение которой известны только на конечной выборке $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. Требуется построить функцию $a : \mathbf{X} \to \mathbf{Y}$, близкую к $\mathbf{y}$. Функция a называется *классификатором*.

Признаком называется отображение $f : \mathbf{X} \to D_f$, где D_f — множество значений признака. Если заданы признаки $f_1, \ldots, f_n$, то вектор $(f_1(x), \ldots, f_n(x))$ называется *признаковым описанием* объекта $x \in \mathbf{X}$. Построение признакового описания для объекта x также носит название *извлечение признаков*.

Обычно классификаторы строят в виде $a(x) = b(\mathbf{f}(x))$, где $\mathbf{f}(x) = (f_1(x), \ldots, f_n(x))$ — некоторое признаковое описание объекта. Удачный выбор признакового описания для объекта позволяет резко повысить качество системы распознавания. Для задачи распознавания объектов на изображении вообще и распознавания цифр в частности было придумано огромное количество различных признаков. Принято делить признаки на 3 класса:

1. Статистические — признаки, хранящие информацию о распределении пикселей на изображении. Это могут быть средние значения пикселей, гистограммы яркости, профили проекции изображения.
2. Геометрические — признаки, хранящие информацию о геометрических свойствах объекта. Сюда относятся площадь и длина контура объекта, выпуклости, вогнутости и т.д.
3. Структурные — признаки, описывающие структуру объекта, т.е. наличие составных частей и связей между ними. Сюда может относиться наличие / отсутствие петель, дуг, прямых линий и связей между ними.

3.4 Классификация

Бустинг с произведением базовых классификаторов. Бустинг с произведением классификаторов [13] является модификацией алгоритма AdaBoost [14].

Пусть X — обучающая выборка, матрица $N \times D$, где каждая строка отвечает наблюдению, а каждый столбец — признаку. Далее, пусть Y — набор меток класса, матрица размера $N \times K$, где $Y_{ik} \in +1, -1$, $+1$ означает принадлежность к классу k. В случае многоклассовой задачи в каждой строке матрицы Y стоит ровно одна единица.

Алгоритм AdaBoost строит классификатор $\mathbf{f} : \mathcal{X} \to \mathbb{R}^K$, минимизирующий экспоненциальную аппроксимацию взвешенной ошибки классификатора:

$$R_H(\mathbf{f}, W) = \sum_{i=1}^{N} \sum_{l=1}^{K} w_{il} I(y_{il} \neq \operatorname{sgn}(f_l(x_i)))$$

$$R_e(\mathbf{f}, W) = \sum_{i=1}^{N} \sum_{l=1}^{K} w_{il} \exp(-y_{il} f_l(x_i))$$

Обычно W ставятся равномерными по объектам:

$$W = \begin{cases} \frac{1}{2N}, & \text{если } y_{il} = 1 \\ \frac{1}{2N(K-1)}, & \text{если } y_{il} = -1 \end{cases}$$

AdaBoost итеративно строит классификатор $\mathbf{f}$ как сумму вида $\mathbf{f}(x) = \sum_{t=1}^{T} \mathbf{h}_t(x)$, где $\mathbf{h}_t(x) = \alpha_t \mathbf{v}_t \varphi_t(x)$, где $\alpha_t \in \mathbb{R}^+$, $\mathbf{v}_t \in \{+1, -1\}^K$, $\varphi_t(x) : \mathcal{X} \to \{+1, -1\}$ — классификатор из некоторого семейства базовых классификаторов. На каждой итерации t алгоритм подбирает α, $\mathbf{v}$, $\phi(x)$, минимизирующие взвешенную ошибку $E(\mathbf{h}, W^t) = \sum_{i=1}^{N} \sum_{l=1}^{K} w_{il}^t \exp(-y_{il} h_l(x_i))$, после чего пересчитывает веса $w_{il}^{t+1} = w_{il}^t \exp(-y_{il} h_l(x_i))$ и нормирует их.

В данном методе предлагается рассматривать $\mathbf{h}(x) = \alpha \prod_{j=1}^{m} \mathbf{v}_j \varphi_j(x)$ (m — параметр алгоритма). Пусть имеется процедура поиска $\alpha, \mathbf{v}, \varphi(x)$ для исходной задачи бустинга. Тогда на каждой итерации по t мы запускаем итеративный процесс: на каждой итерации мы сохраняем уже полученный результат $\alpha^* = \alpha, \mathbf{v}^* = \prod_{j=1}^{m} \mathbf{v}_j, \varphi(\cdot) = \prod_{j=1}^{m} \varphi_j(\cdot)$, фиксируем все классификаторы, кроме одного (j-го), и пытаемся улучшить его. Для этого j-й классификатор обучается с весами W^t на выборке X с метками классов $y'_{il} = \operatorname{sgn}\left(y_{il} \frac{v_l^* \varphi^*(x_i)}{v_{jl} \varphi_j(x_i)}\right)$. После этого по нему получаются новые значения α и $\mathbf{v}_j$. Если для новых значений взвешенная экспоненциальная ошибка больше, чем для оптимальных, итерационный процесс останавливается, иначе на следующей итерации оптимизируется следующий классификатор.

Результаты, полученные при тестировании алгоритма (использовалась реализация [15], обучение проходило на 10000 объектах выборки MNIST, $T = 5000$ итераций, $m = 3$) на собственных данных, показаны на таблице 1.

Таблица 1: Результаты тестирования AdaBoost с произведением базовых классификаторов на собственных данных

Класс	True Positive Rate
0	12.5%
1	94.4%
2	90%
3	75%
4	55.6%
5	88.2%
6	54.6%
7	14.3%
8	60%
9	0%

Согласно [13], на выборке MNIST достигается ошибка на тесте 1.26% при $m = 3$ и $T = 100000$. В моём тестировании суммарная ошибка составила 37% в связи с недобучением (всего 5000 итераций). Эволюция ошибки на тесте в зависимости от числа итераций показана на рис. 2.

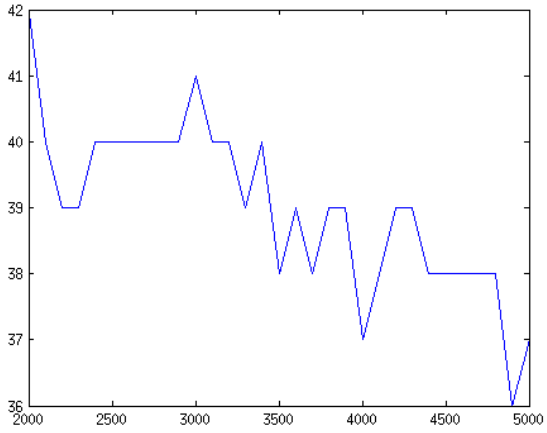


Рис. 2: Эволюция ошибки на тестовой выборке (%).

Свёрточная сеть LeNet 1. Свёрточные нейронные сети представляют собой особый класс нейронных сетей, в которых посредством ограничений на веса нейронов попеременно реализуются операции *свёртки* (*convolution*) и

подразбиения (*subsampling*).

В [16] предлагается конструкция нейронной сети из 6 слоёв: входной, принимающий изображения 28x28 пикселей со значениями от -1 до 1, выходной из 10 элементов и 4 скрытых слоя. Эта конструкция носит название LeNet 1 [17].

Первый скрытый слой состоит из 4 карт признаков — изображений 24x24, получающихся из входного изображения путём применения к нему операции свёртки с фильтрами 5x5, веса которых настраиваются методом обратного распространения ошибки [18], с последующим применением к результату свёртки функции активации. С точки зрения нейронной сети это означает, что у каждого нейрона первого скрытого слоя 26 входов (включая константный), причём для нейронов, относящихся к одной карте признаков, веса, соответствующие одинаковой позиции входного пикселя относительно выходного, должны быть одинаковыми.

На втором слое происходит усреднение и подразбиение результатов, полученных на первом слое. Второй слой состоит из 4 карт размера 12x12. Каждая карта первого слоя разбивается на непересекающиеся квадраты 2x2, и значения внутри них суммируются с некоторым весом. С точки зрения нейронной сети это значит, что каждый нейрон второго слоя связан с 4 нейронами первого слоя, каждый нейрон первого связан только с одним нейроном второго и все веса нейронов второго слоя должны быть равны.

На третьем слое вновь производится «свёртка» ядром размера 5x5 результатов, полученных на втором слое. Третий слой состоит из 12 карт размера 8x8, причём часть карт связана с 2 картами второго слоя (и для них выполняется суммирование свёрток), а часть — с одной. Ограничения, накладываемые на нейроны третьего слоя, похожи на те, что накладываются на нейроны первого.

На четвёртом слое вновь происходит усреднение и подразбиение результатов, полученных на третьем слое. Четвёртый слой состоит из 12 карт признаков размера 4x4, и его ограничения аналогичны ограничениям второго слоя.

Пятый слой состоит из 10 нейронов, полностью соединённых с нейронами 4 слоя без ограничений. Согласно [17], такая сеть достигает ошибки 1.7% на тесте MNIST при обучении на полной обучающей выборке.

SVM с RBF-ядром. Пусть x — вектор признаков объекта, $Y = \{-1, 1\}$. Линейным классификатором называется функция вида $a(x) = \text{sgn}(w^T x + b)$.

Классический метод опорных векторов — это метод обучения линейного классификатора. Пусть дана обучающая выборка X — матрица $N \times D$, где N — количество объектов, D — размернсть признакового пространства, Y — набор меток класса, вектор длины N с элементами $\{1, -1\}$. Метод опорных векторов подбирает параметры w, b линейного классификатора как решение задачи оптимизации:

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i \to min_{w,b,\xi} \\ y_i(< w, x_i > +b) \geq 1 - \xi_i, 1 \leq i \leq N \\ \xi_i \geq 0, 1 \leq i \leq N \end{cases}$$

Коэффициент C — параметр регуляризации.

Выборка может не быть линейно разделима в текущем признаковом пространстве, но может быть разделима в пространстве более высокой размерности. На этом основано использование *ядер*: в задаче оптимизации скалярное произведение заменяется на функцию, являющуюся скалярным произведением в пространстве признаков более высокой размерности. Такой функцией, например, является радиальная базисная функция Гаусса: $k(x, y) = \exp\left(-\gamma\frac{\|x-y\|^2}{2}\right)$.

Обобщение метода на случай многоклассовой классификации происходит с использованием стратегии «все против всех»: если требуется провести классификацию с k метками класса, то для каждой пары классов строится свой классификатор, объект при классификации получает метку класса по общему решению классификаторов.

Использовалась реализация из библиотеки LIBSVM [19] с параметрами $C = 64$ и $\gamma = 0.03125$. Согласно данным на странице MNIST [20], SVM с RBF-ядром достигает значения ошибки в 1.4% на тестовых данных. На моих данных ошибка составила 46%. Данные по тестированию даны в таблице 2.

Таблица 2: Результаты тестирования SVM с RBF-ядром на собственных данных

Класс	True Positive Rate
0	12.5%
1	94.4%
2	80%
3	62.5%
4	55.6%
5	58.8%
6	27.3%
7	14.3%
8	80%
9	0%

4 Разработка алгоритма оффлайн-распознавания рукописных цифр. Основные результаты работы.

На вход алгоритму распознавания подаётся изображение цифры, вырезанной из изображения документа. Выходом служит цифра в текстовом виде. Считается, что на изображении нет других графических объектов. Распознавание цифры проходит в четыре этапа:

1) получение и предварительная обработка изображения;
2) выделение цифры на изображении;
3) извлечение признаков из выделенной цифры;
4) классификация цифры.

Процедура распознавания реализовывалась в программной системе Matlab.

4.1 Описание алгоритма

Предварительная обработка изображения. На вход процедуре поступает путь к файлу изображения. Изображение может иметь любой из стандартных графических форматов.

Указанное изображение считывается программой в формате стандартного цветного изображения с тремя каналами цвета. Затем изображение приводится к формату «уровни серого». Строится изображение Y того же размера, что и исходное, в котором значение каждого пикселя вычисляемая по стандартной формуле:

$$Y(i,j) = 0.2989R(i,j) + 0.5870G(i,j) + 0.1140B(i,j),$$

где $Y(i,j)$ — уровень яркости пикселя (i,j), $R(i,j)$, $G(i,j)$, $B(i,j)$ — уровни красного, зелёного и синего для того же пикселя.

Полученное изображение в формате «уровни серого» затем подвергается пороговой бинаризации по методу Оцу. Строится бинарное изображение I того же размера, что и Y, в котором значение каждого пикселя вычисляется

по следующей формуле:

$$I(i,j) = \begin{cases} 1, & \text{если } Y(i,j) \leq d, \\ 0, & \text{если } I(i,j) > d. \end{cases}$$

В результате такого преобразования более тёмные пиксели (текст) получают значение 1, более светлые (фон) — значение 0. Порог d подбирается методом Оцу как решение задачи оптимизации:

$$d = \arg\min_t \sigma_\omega^2(t)$$

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

где t^* — порог бинаризации, $\omega_i(t), i = 1, 2$ — вероятности классов, разделённых порогом t, $\sigma_i^2(t), i = 1, 2$ — внутриклассовые дисперсии.

В ходе разработки и реализации алгоритма было установлено, что применение других методов обработки изображений не повышает качество распознавания.

Выделение цифры на изображении. Для выделения цифры на изображении в данной работе используется подход, основанный на анализе связных компонент.

По изображению I, полученному в результате бинаризации, строится граф $\Gamma = (V, E)$:

- множество V — белые пиксели изображения I;
- множество E — соседние пиксели в восьмикратной схеме соседства.

Для выделения компонент связности на таком графе используется алгоритм обхода в ширину [21]. В процессе обхода компоненты связности ищутся координаты её самого правого, самого левого, самого нижнего и самого верхнего пикселя. По этим координатам строится минимальный прямоугольник со сторонами, параллельными сторонам исходного изображения, ограничивающий компоненту связности.

Далее выполняется объединение компонент связности, идущих друг за другом: если расстояние между нижней границей одной компоненты связности и верхней границей другой компоненты связности меньше порога, то

эти компоненты связности рассматриваются как одна (т.е. ограничивающие их прямоугольники объединяются). Это делается для того, чтобы соединить верхний штрих цифры "5"и остальную её часть.

Далее из всех ограничивающих прямоугольников выбирается тот, у которого наибольшая ширина, и ограниченное им изображение рассматривается как искомая цифра.

Извлечение признаков. Для выделенных на предыдущем этапе изображений формируется вектор признаков. Изображение цифры достраивается до квадрата так, чтобы центр масс исходного изображения находился в центре этого квадрата. Центр масс имеет координаты:

$$x = \left[\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j) i}{\sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j)} \right],$$

$$y = \left[\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j) j}{\sum_{i=1}^{n} \sum_{j=1}^{m} I(i,j)} \right],$$

где m, n — размеры изображения. После этого для преобразованного изображении производится построение признакового описания цифры. Для этого используются следующие техники:

1. Диагональные признаки [22]. Изображение цифры приводится к размеру 90x60 пикселей с билинейной интерполяцией, после чего делится на 54 квадрата размера 10x10. Далее для каждого квадрата берутся 19 побочных диагоналей, для каждой диагонали берётся среднее значение пикселей, лежащих на ней, после чего берётся среднее этих величин (всего 19). Таким образом, каждый квадрат описывается одним числом, и в сумме получается 54 признака.
2. Среднее значение пикселей по квадратам, полученным в описанном выше методе. Для каждого квадрата в схеме, описанной выше, считается среднее значение (сумма всех пикселей, делённая на 100). Это даёт ещё 54 признака.
3. Два признака Хаара [23] на изображении 90x60. Выбирается правая или левая половина изображения, после чего из верхней четверти вычитается нижняя и берётся среднее значение разности.

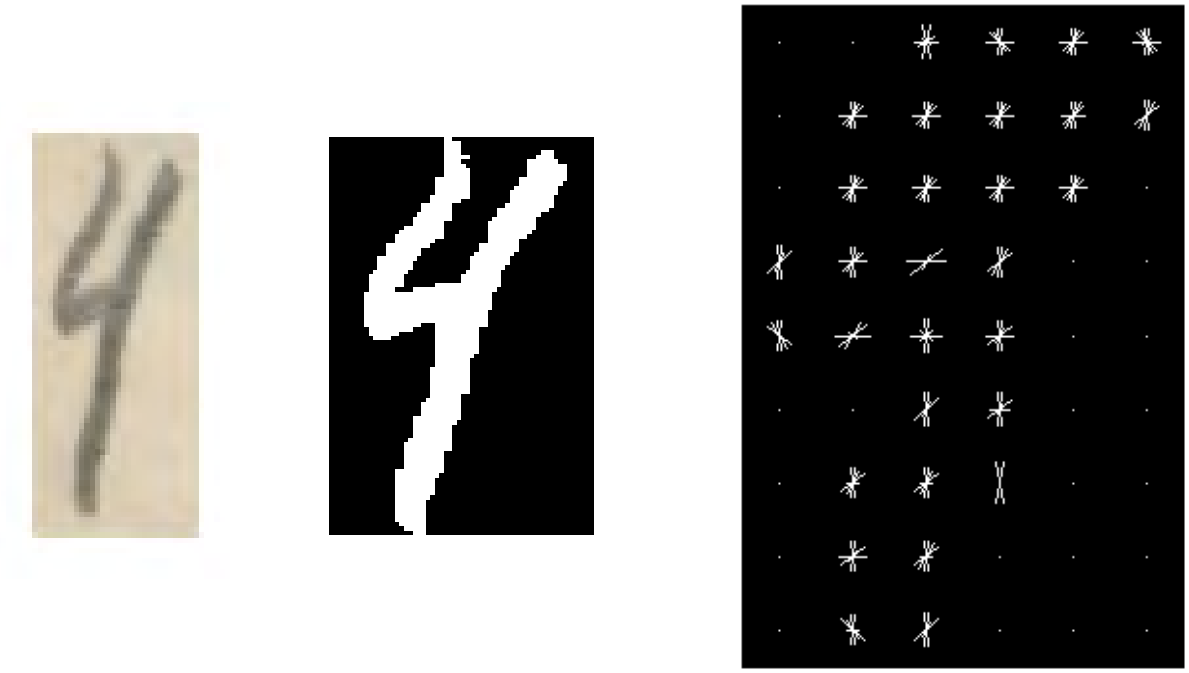


Рис. 3: Цифра, преобразованная цифра, гистограмма ориентированных градиентов.

4. Площадь области фона, находящейся внутри цифры, делённая на размер изображения. Для поиска этих областей используется алгоритм обхода в ширину на исходном (не преобразованном к размеру 90x60) изображении. Для изображения строится граф $\Gamma = (V, E)$: V — все чёрные пиксели изображения, E — пиксели, соседние в четырёхкратной системе соседства. Алгоритм обхода графа в ширину запускается из всех точек, лежащих на границе изображения. Точки, до которых не удалось дойти, лежат в области, ограниченной цифрой.
5. Гистограмма ориентированных градиентов [24] для изображения, приведённого к размеру 90x60. Использовались клетки размера 10x10 с 1 клеткой в блоке и 9 возможными направлениями. Пример приведён на рис. 3.

В результате каждая выделенная цифра описывается вектором из 597 элементов.

Распознавание цифр. Для классификации изображений использовался классификатор SVM с RBF-ядром [25] в реализации библиотеки LIBSVM [19].

Обучение классификатора происходило на 10000 объектах выборки MNIST [20] с извлечением указанного выше набора признаков. Методом кросс-валидации были подобраны параметры $C = 30$, $\gamma = 0.026$. Построенный классификатор тестировался на 5000 изображениях цифр из базы MNIST и на 100 изображениях цифр, вырезанных из рукописей, предоставленных РГАЛИ. На изображениях из базы MNIST была достигнута точность 97.4%.

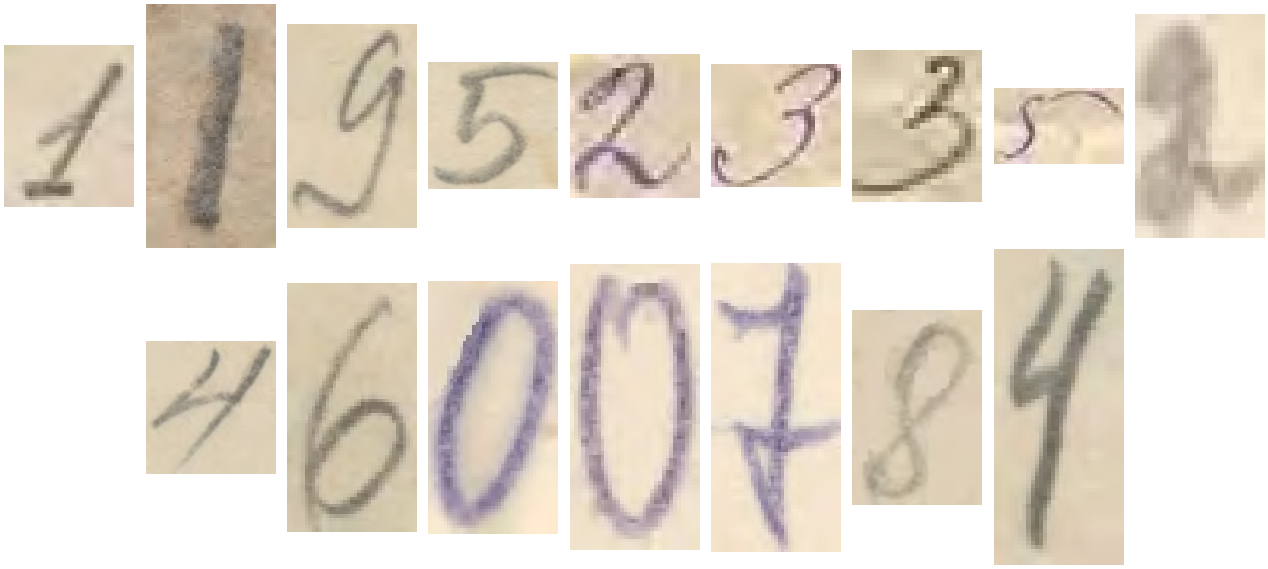
4.2 Тестирование алгоритма

Алгоритм распознавания тестировался с использованием выборки из 100 цифр, вырезанных из изображений рукописей Российского государственного архива литературы и искусства. Была достигнута точность 81%. Точность распознавания для каждой цифры приведена в таблице 3. Время работы на одном изображении составляет ≈ 0.3 секунды, включая считывание изображения, выделение цифры, извлечение признаков и классификацию. Эти результаты превосходят результат, полученный в работе [2] (лучшая достигнутая точность — 76.2%).

Таблица 3: Результаты тестирования SVM с RBF-ядром на собственных данных с извлечением признаков

Класс	True Positive Rate
0	75%
1	100%
2	90%
3	100%
4	88.9%
5	94.1%
6	72.7%
7	28.6%
8	100%
9	14.3%

Пример входных данных:



Результат классификации: 1, 1, 1, 5, 2, 3, 3, 5, 2, 4, 6, 0, 0, 1, 8, 1.

5 Выводы

Оффлайн-распознавание рукописных цифр является частью общей задачи распознавания рукописного текста, в том числе исторических рукописей.

1. В настоящей работе: рассмотрены алгоритмы распознавания рукописных цифр (бустинг, свёрточные сети, метод опорных векторов), проведено тестирование алгоритмов на тестовой выборке, состоящей из цифр, вырезанных из реальных документов РГАЛИ.
2. Разработан алгоритм оффлайн-распознавания рукописных цифр на основе метода опорных векторов, произведено его тестирование на выборке MNIST и на собственном наборе данных — выборке цифр, вырезанных из рукописей Российского государственного архива литературы и искусства. Точность распознавания составила: на выборке MNIST — 97.4%, на собственных данных — 81%. Это превосходит точность распознавания в работе [2].
3. Разработаны и протестированы алгоритмы предварительной обработки изображения, поиска цифры и её распознавания на реальных документах.

Список литературы

[1] Yann LeCun и др. «Backpropagation applied to handwritten zip code recognition». В: *Neural computation* 1.4 (1989), с. 541—551.

[2] Ofer Matan и др. «Reading handwritten digits: A ZIP code recognition system». В: *Computer* 25.7 (1992), с. 59—63.

[3] Filipe Coelho и др. «Automatic System for the Recognition of Amounts in Handwritten Cheques.» В: *SIGMAP*. Citeseer. 2008, с. 320—324.

[4] Mansi Shah и Gordhan B Jethava. «A literature review on hand written character recognition». В: *Indian Streams Research Journal* 3.2 (2013), с. 1—19.

[5] SS Bedi и Rati Khandelwal. «Various image enhancement techniques-a critical review». В: *International Journal of Advanced Research in Computer and Communication Engineering* 2.3 (2013).

[6] Réjean Plamondon и Sargur N Srihari. «Online and off-line handwriting recognition: a comprehensive survey». В: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.1 (2000), с. 63—84.

[7] Л Шапиро и Дж Стокман. «Компьютерное зрение». В: *М.: Бином. Лаборатория знаний* 752 (2006), с. 8.

[8] Nobuyuki Otsu. «A threshold selection method from gray-level histograms». В: *Automatica* 11.285-296 (1975), с. 23—27.

[9] Felipe C Ribas и др. «Handwritten digit segmentation: a comparative study». В: *International Journal on Document Analysis and Recognition (IJDAR)* 16.2 (2013), с. 127—137.

[10] Hiromichi Fujisawa, Yasuaki Nakano и Kiyomichi Kurino. «Segmentation methods for character recognition: from segmentation to document structure analysis». В: *Proceedings of the IEEE* 80.7 (1992), с. 1079—1092.

[11] Zhixin Shi и Venu Govindaraju. «Segmentation and recognition of connected handwritten numeral strings». В: *Pattern Recognition* 30.9 (1997), с. 1501—1504.

[12] R Fenrich и S Krishnamoorthy. «Segmenting diverse quality handwritten digit strings in near real-time». В: *Procedings of the 4th Advanced Technology Conf.* 1990, с. 523—537.

[13] Balázs Kégl и Róbert Busa-Fekete. «Boosting products of base classifiers». В: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM. 2009, с. 497—504.

[14] Yoav Freund и Robert E Schapire. «A desicion-theoretic generalization of on-line learning and an application to boosting». В: *Computational learning theory*. Springer. 1995, с. 23—37.

[15] Djalel Benbouzid и др. «MultiBoost: a multi-purpose boosting package». В: *The Journal of Machine Learning Research* 13.1 (2012), с. 549—553.

[16] B Boser Le Cun и др. «Handwritten digit recognition with a back-propagation network». В: *Advances in neural information processing systems*. Citeseer. 1990.

[17] Yann LeCun и др. «Comparison of learning algorithms for handwritten digit recognition». В: *International conference on artificial neural networks*. Т. 60. 1995.

[18] David E Rumelhart, Geoffrey E Hinton и Ronald J Williams. *Learning internal representations by error propagation*. Тех. отч. DTIC Document, 1985.

[19] Chih-Chung Chang и Chih-Jen Lin. «LIBSVM: A library for support vector machines». В: *ACM Transactions on Intelligent Systems and Technology* 2 (3 2011). Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 27:1—27:27.

[20] Yann LeCun и Corinna Cortes. *The MNIST database of handwritten digits*. 1998.

[21] Википедия. *Поиск в ширину — Википедия, свободная энциклопедия*. 2015. URL: `http://ru.wikipedia.org/?oldid=67694205` (дата обр. 19.05.2015).

[22] J Pradeep, E Srinivasan и S Himavathi. «Diagonal based feature extraction for handwritten alphabets recognition system using neural network». В: *arXiv preprint arXiv:1103.0365* (2011).

[23] Википедия. *Признаки Хаара — Википедия, свободная энциклопедия*. 2015. URL: `http://ru.wikipedia.org/?oldid=67915229` (дата обр. 04.05.2015).

[24] Navneet Dalal и Bill Triggs. «Histograms of oriented gradients for human detection». В: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Т. 1. IEEE. 2005, с. 886—893.

[25] Corinna Cortes и Vladimir Vapnik. «Support-vector networks». В: *Machine learning* 20.3 (1995), с. 273—297.