ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ВОЛОГОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра Автоматики и Вычислительной Техники

Структуры и алгоритмы обработки данных

Пояснительная записка к курсовой работе

«Распознавание рукописных цифр

полным перебором через метод свёртки

с отсечениями»

Выполнил: Веселов Д.А.

Группа: ЭПО-21

Проверил: Андрианов И.А.

Вологда

2011

Содержание:

[1. Описание работы алгоритма 3](#_Toc294771205)

[1.1. Историческая справка 3](#_Toc294771206)

[1.2. Описание алгоритма 4](#_Toc294771207)

[1.3. Класс входных данных, для которых применим алгоритм 6](#_Toc294771208)

[1.4. Анализ временной и пространственной сложности алгоритма 6](#_Toc294771209)

[1.5. Сравнение с аналогами 7](#_Toc294771210)

[1.6. Примеры практических задач, где может использоваться данный алгоритм 7](#_Toc294771211)

[2. Разработка визуализатора 8](#_Toc294771212)

[2.1. Выбор средств разработки 8](#_Toc294771213)

[2.2. Проектирование интерфейса 8](#_Toc294771214)

[2.3. Разработка алгоритмов пошагового выполнения визуализации 10](#_Toc294771215)

[2.4. Особенности программной реализации 11](#_Toc294771216)

[2.5. Методика и результаты тестирования ПО 11](#_Toc294771217)

[3. Список литературы 12](#_Toc294771218)

# Описание работы алгоритма

# Историческая справка

В идею алгоритма заложена статья М. Ю. Сиротенко «Применение нейросетей в распознавании изображений» [1]. Из-за сложности реализации нейронных сетей и недостаточного количества хорошей литературы на русском языке было решено взять алгоритм полного перебора и усовершенствовать его для максимального ускорения распознания.

Метод ускорения алгоритма так же взят из статьи Сиротенко. В этой статье автор рассказывает о свёрточных нейронных сетях, впервые использованных для распознавания рукописных символов американским учёным французского происхождения Яном ЛеКуном, вдохновлённым работами нобелевских лауреатов в области медицины Torsten Nils Wiesel и David H. Hubel. Эти ученые исследовали зрительную кору головного мозга кошки и обнаружили несколько интересных фактов, которые потом взял на заметку ЛеКун.

Свёрточные сети основываются на математической операции, называемой «свёрткой» [2]. Суть этой операции в том, что каждый фрагмент изображения умножается на матрицу (ядро) свертки поэлементно и результат суммируется и записывается в аналогичную позицию выходного изображения. Основное свойство таких фильтров заключается в том, что значение их выхода тем больше чем больше фрагмент изображения похож на сам фильтр. Таким образом, изображение свернутое с неким ядром даст нам другое изображение, каждый пиксел которого будет означать степень похожести фрагмента изображения на фильтр. Иными словами это будет карта признаков.

Отсечение заведомо неправильных вариантов было найдено эмпирически.

# Описание алгоритма

На вход алгоритму подаётся четырёхмерная матрица обработанных рукописных цифр для обучения из базы MNIST и тестовое изображение для распознавания. Эта база содержит 60 000 обучающих пар (изображение — метка) и 10 000 тестовых (изображения без меток). Изображения нормализованы по размеру и отцентрованы. Размер каждой цифры не более 20х20, но вписаны они в квадрат размером 28х28. Пример первых 12 цифр из обучающего набора базы MNIST приведен на рисунке 1.

image

Рис.1. Первые 12 цифр из базы MNIST

Структура четырёхмерной булевой матрицы BdImgs[a][b][28][28]:  
a – 10 цифр арабского алфавита (0, 1, 2, и т.д. до 9)  
b – количество образов написания символа в базе  
последние 2 значения – двумерная матрица начертания текущего образа из выбранной цифры.

После загрузки базы в память происходит свёртка всех образов в матрицы размером 14х14 элементов. Свёртка происходит для квадрата 2х2 пиксела, ядром свёртки выбраны степени двойки из-за простоты кодирования и обратного декодирования. В таблице 1 приведена карта признаков для свёртки, закрашенные ячейки обозначают, что в текущей позиции есть пиксел.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Изображение | Свёртка | Изображение | Свёртка | Изображение | Свёртка |
|  | 0 |  | 6 |  | 12 |
|  | 1 |  | 7 |  | 13 |
|  | 2 |  | 8 |  | 14 |
|  | 3 |  | 9 |  | 15 |
|  | 4 |  | 10 |
|  | 5 |  | 11 |

Табл.1. Карта признаков для свёртки

Формула свёртки:  
Свёртка = Ячейка[0][0] \* 1 + Ячейка[0][1] \* 2 + Ячейка[1][0] \* 4 + Ячейка[1][1] \* 8

Обратные формулы свёртки:  
Ячейка[0][0] = (Свёртка & 1) != 0  
Ячейка[0][1] = (Свёртка & 2) != 0  
Ячейка[1][0] = (Свёртка & 4) != 0  
Ячейка[1][1] = (Свёртка & 8) != 0

Где «&» - обозначает логическое «И», а «!=» логическое «не равно».

После того, как получен массив свёрток всех образов производится свёртка символа для распознания. Далее происходит обход по всем образам из базы с отсечением варианта, когда в текущем положении свёртка изображения для распознания равна 0. Когда свёртки символа для распознания и образа равны к так называемой «вероятности» символа прибавляется 1.

После полного обхода всех образов выбирается максимальная «вероятность» - это и будет распознанным символом.

К сожалению, этот алгоритм груб и не точен, в отличии от свёрточных нейронных сетей, но зато он позволяет очень быстро узнать, с 70% точностью в среднем, какой символ для распознавания подан на вход. Так же в нём не используются повороты и искажения обучающих образов, т.к. в базе каждого символа насчитывается по несколько тысяч различных цифр.

# Класс входных данных, для которых применим алгоритм

Алгоритм применим к тестовым изображениям из базы MNIST и собственным рукописным символам размером до 20х20 пикселей. Ради эксперимента было опробовано изменение положения всех символов в обучающей базе к левому верхнему краю, чтобы сверху и слева не было пустых рядов и строк пикселей, но это почему-то привело к резкому ухудшению результата распознавания. Процент ошибок возрос до 50%. Так же если отцентровать все изображения из обучающей и тестовой выборки, то процент ошибок увеличится.

# Анализ временной и пространственной сложности алгоритма

Анализ временной составляющей производился замером времени непосредственно в программе. Загрузка базы обучающих образов составляет примерно 5-6 с, а распознавание символа менее 1 с.

Сложность алгоритма зависит от количества обучающих образов в базе и от количества свёрток у изображения для распознавания. На данный момент в базе MNIST содержится 59 990 образов, максимальное количество свёрток у изображения для распознавания 196, но оно никогда не достигается и действительное количество гораздо меньше. По этим данным можно судить, что при распознавании символа производится до 11 758 040 итераций цикла, если не учитывать отсечения, которые дают очень большое ускорение работы алгоритма.

# Сравнение с аналогами

Аналогов программы множество, но из бесплатных и работающих с базой MNIST удалось опробовать только «Neural Network for Recognition of Handwritten Digits» [3]. Она работает на свёрточных нейронных сетях и в разы превосходит выбранный алгоритм после полного обучения сети. Процент ошибок в ней составляет менее 1%.

Так же есть проприетарная коммерческая программа «ABBYY FineReader». Она работает не только с распознаванием цифр, но и с другими печатными символами различных шрифтов и языков.

# Примеры практических задач, где может использоваться данный алгоритм

Алгоритм может использоваться как часть системы оптического распознавания символов или для обучения студентов в курсе «распознавания изображений» так как данный алгоритм прост для понимания начинающим.

# Разработка визуализатора

# Выбор средств разработки

Вначале курсовая работа была выполнена в виде консольного приложения в среде разработки Microsoft Visual C++ 2010 Express, затем перенесена в виде приложения для Windows в Borland C++Builder 6.

Для работы с изображениями выбрана библиотека FreeImage от Hervé Drolon. Она была выбрана из-за простоты работы с различными типами изображений и небольшого размера, в сравнении с такими «гигантами» как OpenCV и ImageMagick.

Так же в самом начале была написана небольшая утилита для обработки BMP изображений в булевые массивы для ускорения загрузки в программу (10-15 с обработка изображений и занесение их в память против 5-8 с загрузки обработанных массивов из файлов).

# Проектирование интерфейса

Интерфейс программы выполнен в «минималистическом» стиле с добавлением нестандартных иконок для кнопок. На рисунке 2 показано окно работы программы после загрузки.

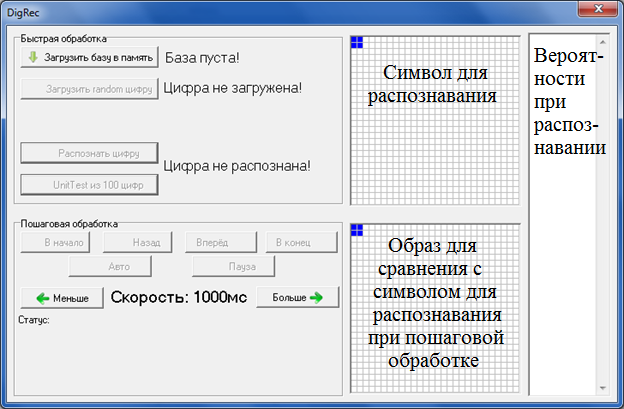


Рис 2. Окно программы

После загрузки перед пользователем появляется окно, в котором можно нажать кнопки «Загрузить базу в память» для загрузки массивов из файлов, «Загрузить random цифру» для загрузки символа для распознавания, «Распознать цифру» и кнопкой «UnitTest из 100 цифр» для проведения тестирования работы алгоритма.

Так же в программе доступна пошаговая обработка алгоритма с выбором автоматической скорости для распознавания. К сожалению, из-за отрисовки образа для распознания работает она крайне медленно даже при скорости 1 мс. На рисунке 3 показано окно работы программы в пошаговом режиме.

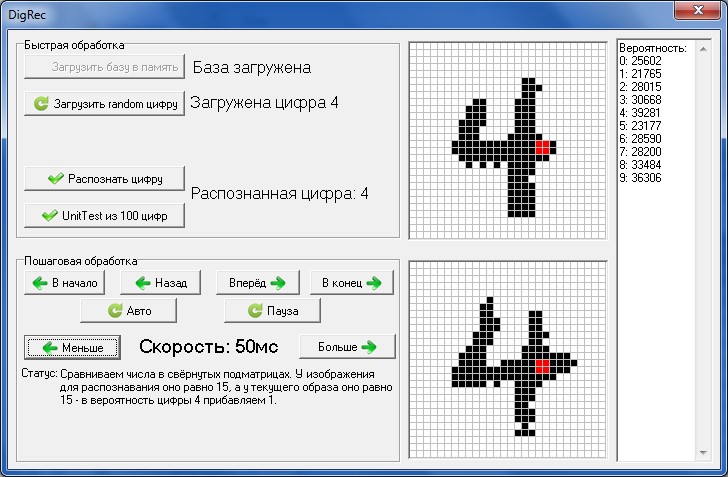


Рис. 3. Пошаговая обработка.

# Разработка алгоритмов пошагового выполнения визуализации

Пошаговая обработка была спроектирована с использованием логики «конечных автоматов» [4], т.е. в каждый момент времени в памяти хранится 3 состояния:  
- нулевое;  
- текущее;  
- конечное.

При нажатии на кнопки «В начало», «Назад», «Вперёд» и «В конец» происходит изменение текущего состояния. Конечное меняется лишь при начале работы пошагового выполнения. Нулевое состояние не меняется и служит границей для определения начала алгоритма при многократном нажатии на кнопку «Назад».

# Особенности программной реализации

Особенностями программной реализации является то, что при пошаговом выполнении можно смотреть на визуальный результат с отображением свёрток для изображения для распознания и сравниваемым образом. Так же особенностью является то, что можно узнать вероятности цифр при распознавании.

# Методика и результаты тестирования ПО

Для тестирования работы программы был написан UnitTest или так называемое «модульное тестирование» [5]. Этот тест отображает количество ошибок при распознавании символа в выборке из 100 случайных изображений. Данный метод помог найти регрессию при использовании смещения изображений в левый верхний угол.

Результат тестирования: 20-35% ошибок при распознавании символа из 100 случайных. Около 5% из которых составляют символы, которые сложно распознать даже человеку.

Я считаю, что результат работы ПО удовлетворительный.

# Список литературы

1. **М.Ю., Сиротенко.** Применение нейросетей в распознавании изображений. *Хабрахабр (Искусственный интеллект).* [В Интернете] 2011 г. http://habrahabr.ru/blogs/artificial\_intelligence/74326/.

2. Свёртка (математический анализ). *Википедия.* [В Интернете] http://ru.wikipedia.org/wiki/Свёртка\_(математический\_анализ).

3. **O'Neill, Mike.** Neural Network for Recognition of Handwritten Digits. [В Интернете] 2006 г. http://www.codeproject.com/KB/library/NeuralNetRecognition.aspx.

4. Конечный автомат. *Википедия.* [В Интернете] http://ru.wikipedia.org/wiki/Конечный\_автомат.

5. Модульное тестирование. *Википедия.* [В Интернете] http://ru.wikipedia.org/wiki/Модульное\_тестирование.