

*Софийски университет „Св. Климент Охридски”*

*Факултет по математика и информатика*

***Курсов проект***

***по Изкуствен интелект***

***на тема:***

*„OCR чрез невронни мрежи”*

***Преподаватели:***

доц. д-р Иван Койчев

ас. Гергана Лазарова

***Изготвили:***

*Николина Ефтимова* ф.№ 80403, IV курс, компютърни науки

*Дамяна Иванова,*, ф.№ 80404, IV курс, компютърни науки

*Борис Пелтеков* ф.№ 80420, IV курс, компютърни науки

*Александър Тодоров,*, ф.№ 80435, IV курс, компютърни науки

Съдържание:

1. Подготовка на изображението
2. Изготвяне на данни за обучение на невронните мрежи
3. Многослойните мрежи , използващи метода с обратното разпространение на грешката
4. **Конволюционна невронна мрежа**
5. Изготвяне на данни за сравняване на успеваемостта на невронните мрежи

## 

## 

## 1. Подготовка на изображението

Подготовката на изображението е има съществена роля за софтуерните продукти за разпознаване на символи. Най-важната и функция е отделянето на буквите от фона на картинката. В текущия проект са реализирани няколко метода за обработка на изображението. Всеки от тях е включен в един изпълним jar файл, като избора на метода става чрез command line параметри.

## 1.1. Филтри

Прилагането на филтри върху снимка позволява постигането на различни ефекти върху нея. Всички описани филтри, с изключение на grayscale, ползват изображения в сивата гама. Целта е да се работи само с един цветови канал (сив) вместо три (червено, зелено, синьо).

## 1.1.1 Grayscale

**параметри**: *-filter -grayscale <input\_file> <output\_file> [mode]*

Grayscale превръща снимката от цветна (RGB) в сива. Познати са няколко метода за преобразуването от единия формат в другия. Те се задават чрез параметъра mode. По подразбиране е избран методът HDTV. Поддържаните методи за mode са:

* -type-average Y = ⅓ \* R +⅓ \* G +⅓ \* B  
  Смята средноаритметичното на всички канали
* -type-pal Y = 0.299 \* R + 0.587 \* G + 0.114 \* B  
  Произлиза от стария стандарт за предаване на картина, реализиран в стандартите PAL, SECAM, NTSC, чрез един “цветен” и един черно-бял канал
* -type-hdtv Y = 0.2126 \* R + 0.7152 \* G + 0.0722 \* B  
  Произлиза от новия стандарт за предаване на картина (HDTV), който е по-близък до човешките възприятия за сиво.

Y - сив канал

R - червен канал

G - зелен канал

B - син канал

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| оригинална | average | hdtv | pal |

## 1.1.2 Blur

**параметри**: *-filter -blur <input\_file> <output\_file>*

Blur ефектът се използва за намаляване шума в изображението. За целта на всеки пиксел се присвоява нова стойност, която зависи от него и съседите му. Теглата на съседните пиксели се задават чрез матрица, а резултатът се смята като всеки пиксел се умножи по съответното му тегло и полученото число се раздели на сумата от теглата в матрицата. Например:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 1 |  | 126 | 255 | 227 |  |  |
| 1 | 4 | 2 | + | 255 | 255 | 255 | = | 205 |
| 1 | 2 | 1 |  | 78 | 75 | 209 |  |  |
| Тегла | | |  | Пиксели | | |  | Резултат |

Пикселите се взимат от входното изображение, а резултатът се записва в изходното изображение:

|  |  |
| --- | --- |
|  |  |
| преди | след |

## 1.1.3 Sharpen

**параметри**: *-filter -sharpen <input\_file> <output\_file>*

Sharpen ефектът се използва за увеличаване остротата на изображението. Принципът е същият както при Blur, но се използват други тегла. Например:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| -1 | -1 | -1 |  | 209 | 193 | 206 |  |  |
| -1 | 9 | -1 | + | 210 | 200 | 192 | = | 192 |
| -1 | -1 | -1 |  | 205 | 191 | 202 |  |  |
| Тегла | | |  | Пиксели | | |  | Резултат |

|  |  |
| --- | --- |
|  | ss |
| преди | след |

Ако резултатът е по-малък от 0, се присвоява 0, а в случай, че е по-голям от 255, се присвоява 255.

**1.3. Бинаризация - метод на Otsu**

Методът на Otsu се използва за автоматично определяне на границата между фон и букви. Първата стъпка от прилагането му е да се обърне изображението от цветната в сивата скала. Това позволява работата с по-малко стойности и само един канал за обработка.

Втората стъпка е създаването на хистограма по цвета на пикселите. Броят се всички пиксели с един и същи нюанс на сивото. За всеки нюанс се смята претеглената вътрешна разликата между двата класа след отсичането на фон и букви. Използва се формулата:

, където с t е означена стойността на нюанса, който се ползва за разделител между фон и букви, с и са означени теглата на двата класа, а с и са означени стандартните отклонения на двата класа.



Правилната стойност на сивото, която трябва да служи за граница на двата класа, е t-то, при което има най-малка стойност.



Отсу е доказал, че минимизирането на вътрешна разликата на класовете е същото като максимизирането на междукласовата разлика, която е описана с формулата:

, където



се смята от хистограмата, след като се фиксира t по формулата , където е стойността на нюанса на сивото i, е средностатистическата стойност на класа, сметната по форулата , където е стойността на хистограмата за нюанса i.



Пример:

|  |  |
| --- | --- |
|  |  |
| Преди бинаризацията | След бинаризацията |

Проблемът е, че алгоритъмът работи добре само ако фонът на изображението е хомогенен (бял). В такъв случай, за да се постигне добра бинаризация се използва локален Otsu. Изображението се разделя на сектори NxN и се прилага алгоритъмът за всеки от тях. Накрая резултатите се събират.

Пример:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1х1 |  | 2х2 |
|  | 6х6 |  | 23х23 |

Пример със снимана страница от ксерокопие:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1х1 |  | 3х3 |
|  | 8х8 |  | 29х29 |

Направена е и една модификация на алгоритъма. За по-добри резултати всички изходи от NxN до MxM (N < M) се наслагват един върху друг и на полученото изображение отново се пуска алгоритъмът.

Резултати:

|  |  |
| --- | --- |
|  |  |

Начин на използване:

**параметри**: *-binarize <input\_file> <output\_files\_prefix> [-start N] [-end M]*

*input\_file -* входен файл, цветен или в сивата скала

*output\_files\_prefix -* начало на изходните файлове

*-start N - м*оже да се изпусне, по подразбиране е 1, задава началото на големината на секторите

*-end M -* може да се изпусне, по подразбиране е N + 1, задава края на големината на секторите

Алгоритъмът прави всички сектори в интервала от N до M-1 като името на резултатите е *<output\_files\_prefix><i>.png и <output\_files\_prefix>all.png за резултата от наложените изображения, където N <= I < M*

## 1.2. Отделяне на букви

След като е направена бинаризацията и буквите са получили черен цвят, а фонът е останал бял, те трябва да бъдат отделени една от друга. Използвани са два метода:

**1.2.1. Хистограма**

**параметри**: *-wordcropper -histogram <input\_file> <output\_files\_prefix>*

Хистограмите, използвани за отделяне на текст, представляват натрупване на броя пиксели по хоризонтала или вертикала. Ако хистограмата се приложи по редове, получената графика показва къде в изображението има ред с букви и къде има междуредие. Когато има локален минимум, това показва, че е засечено междуредие.



След това редовете се отделят един по един и на всеки от тях се прави хистограма по колони. Падовете показват къде буквите се отделят една от друга.



Резултатът е:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |

**1.2.2. BFS**

Вторият метод използва търсене в ширина. Веднъж засечен черен пиксел започва обхождане на региона от черни пиксели, в който участва. По време на обхождането му се записват границите на минималния правоъгълник, който обхваща обхождания регион.



Резултатът е:

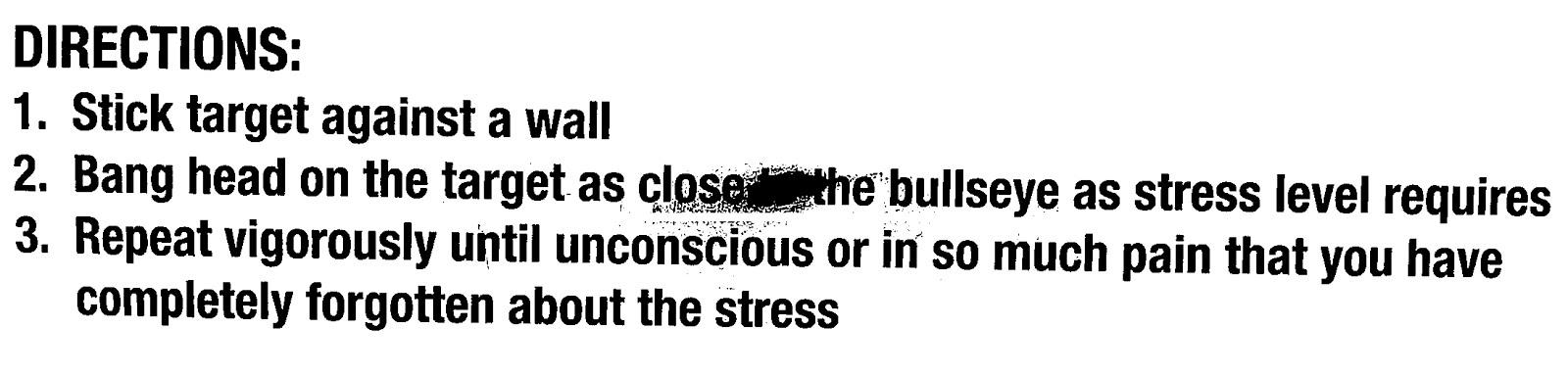
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |

Така получените правоъгълници обхващат всяка буква една по една. Изходните изображения се свиват или разпъват до 29х29 пиксела и след това биват подавани като вход на невронните мрежи за разпознаване.

## Предимства и недостатъци.

Нека да разгледаме едно не толкова добро изображение:

При използването на хистограми се получава следният резултат:



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

При използването на BFS се получава следният резултат:



Както се вижда, методът с хистограмите е успешен, когато няма шум между редовете и те са хоризонтални. В противен случай се стига до смесването им. Методът на BFS не смесва редовете, но може да раздели на две буква, състояща се от две части. Също така, ако има достатъчно голям шум, може да го счете за символ.

**1.4. Преобразуване от снимка в матрица, подходяща за невронните мрежи**

**параметри**: -*datamatrix <input\_file> <output\_matrix\_file>*

Този метод взима пикселите на входното изображение и записва техните стойности на сивото в изходния файл. На всеки ред от изходния файл съответства един ред от изображението. Пикселите са разделени с една или повече шпации (“ ”). Записаните стойности са обратни на реалните (255 - реална\_стойност), тъй като на черния цвят съответства код 0, а за невронните мрежи това е най-същественият цвят (255) и съответно кодът на бялото е 255, а за невронните мрежи трябва да е 0.

Пример:

|  |  |
| --- | --- |
|  | 0 0 0 0 0 0 24 163 137 14 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 44 242 196 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 1 1 1 0 46 255 199 29 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0  59 93 97 97 97 96 126 255 227 115 96 97 97 97 97 97 97 97 97 97 97 97 97 94 53 3 0 0 0  154 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 127 10 0 0 0  19 37 40 40 40 38 78 255 209 65 39 40 40 40 40 40 40 40 40 40 40 40 40 38 17 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 45 255 199 29 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 39 221 217 39 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 18 160 255 92 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 31 21  0 0 0 0 0 0 1 75 255 209 75 10 0 0 0 0 0 0 0 0 0 0 2 10 29 67 129 209 154  0 0 0 0 0 0 0 9 92 221 233 130 64 33 19 14 13 15 20 29 44 67 100 142 199 255 233 147 67  0 0 0 0 0 0 0 0 6 55 136 217 255 209 174 157 154 160 176 202 242 255 242 191 138 90 46 14 1  0 0 0 0 0 0 0 0 0 0 13 49 94 140 184 227 255 227 191 157 124 90 58 31 11 1 0 0 0 |

## 2. Изготвяне на данни за обучение на невронните мрежи

За обучението на невронните мрежи е необходимо да се подготви достатъчно голям и разнообразен training set. Решихме, че подходящ формат на входните данни е матрица 29x29 пиксела със стойности от 0 (най-незначими) до 255(най-значими). Символите, които избрахме да включим в training set-а са цифрите от “0” до “9”, големите латински букви от “A” до “Z” и малките латински букви от “a” до “z”. За шрифтове използвахме едни от най-распространените: **Georgia, Times-New-Roman, Helvetica, Arial, Comic-Sans-MS, Tahoma, Trebuchet-MS, Verdana, Courier-New, Lucida-Console, Palatino-Linotype и Lucida-Sans-Unicode.**

За да генерираме за всеки шрифт всички символи, използвахме командния редактор **ImageMagick®.** Командата, която използвахме е:

*convert -fill "#000000" -font <font> -pointsize 350 label:"<symbol>" -trim -resize "29x29!" -type BiLevel <folder>/<symbol>/<font>.png*

Като <font> трябва да се замести с един от шрифтовете, <symbol> с един от символите [0-9A-Za-z] и <folder> с папката, в която да бъде сложен изходния файл. Другите параметри специфицират, че цветът на буквата трябва да е черен, буквата трябва да е изрязана (да няма бял фон отстрани), изходният формат трябва да е картинка 29х29 пиксела с два варианта на сивото (черно и бяло). Резултатът за буквата “A” е:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| arial | comic sans ms | courier new | georgia | helvetica | lucida console | lucida sans unicode | palatino linotype | tahoma | times new roman | trebuchet ms | verdana |

Така получените картинки бяха конвертирани към файл, удобен за обучение на невронните мрежи.

Файлът може да бъде свален от: <https://github.com/gibona/fmi-ocr-neural-network/blob/master/training_data/source_data/all.txt>

Форматът му е следният:

1. На първия ред се намира **N** - броят матрици, включени във файла
2. Следват N пъти по 30 реда, които описват всеки пример от training set-a:
   1. Първите 29 реда съдържат по 29 числа, представляващи матрицата за примера
   2. На следващия ред се намира символ, отговарящ на съдържанието на матрицата

**3. Невронни мрежи**

Ефективността на невронните мрежи се дължи на големия брой елементарни изпълнителни единици, известни като неврони. Нека разгледаме биологичен и изкуствен неврон.

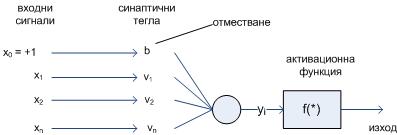
* + 1. **Биологичен неврон**

Биологичната нервна клетка се състои от тяло, входни връзки - дендрити, единствена изходна връзка - аксон. Местата където нервните клетки образуват връзки се наричат синапси.

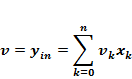
Предаването на сигнала става като във всеки един момент от времето, клетката приема сигнали (по дендритите) от други клетки. В зависимост от тези сигнали тялото променя своя електрическия потенциал. Ако той достигне стойност над определен праг се генерира електрически импулс, който се разпространява по аксона достигайки до други клетки, с които има образувани синапси. Импулса, разпространяващ се по аксона има винаги една и съща амплитуда, което прилича на двоично предаване на информация. Невробиолозите смятат, че информацията се предава посредством честотата на импулсите, т.е. чрез честотно кодиране.

* + 1. **Изкуствен неврон**

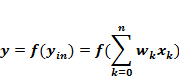
Структурата на изкуствен неврон е аналогична на тази на биологичната нервна клетка.



## Всеки неврон има много входове, които са модел на дендритите, и един изход, които е модел на аксона. На входовете x1, x2, …, xn постъпват сигналите към неврона. Те могат да са външни сигнали или сигнали от изходите на други неврони. С всеки изход е свързано тегло(синаптично тегло), което моделира силата на връзката между невроните при предаване на сигнала посредством дендрита свързан със съответния синапс. За удобство и повече свобода, при модела на неврона се въвежда и сигнал с постоянна стойност b, наречен още отместване (bias). Входните сигнали в тялото на неврона се генерират със суматора, чийто изходен сигнал се определя чрез формулата:

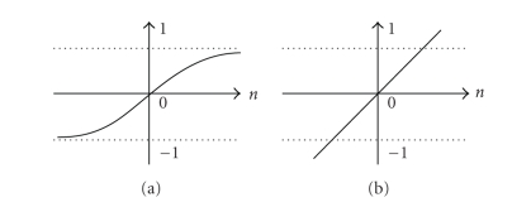


## Тялото на неврона се моделира от суматора и активационната функция, посредством която се изчислява изходния сигнал на неврона. В общия случай активационната функция не е линейна, а изходният сигнал се изразява по следния начин:



## Активационните функции могат да бъдат:

## линейни

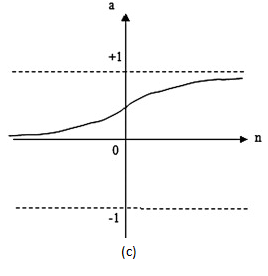


## (b)



## сигмуидални

## (a)

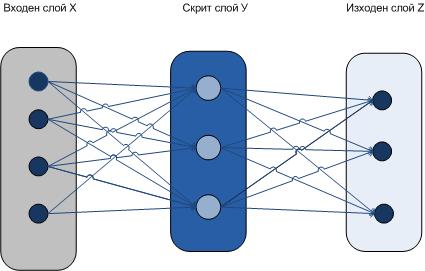


(c)



## 3. Multilayer Feedforward Neural Network

## Многослойните мрежи , използващи метода с обратното разпространение на грешката, могат да бъдат двуслойни, трислойни, и т.н. Но най – често използваните многослойни невронни мрежи са двуслойните. Те имат следната архитектура:



Мрежата се състои от входен слой X, чрез които се подават входните сигнали

Х [x0, x1, x2, …, xn]T към мрежата, от изходен слой Z , като векторът Z [z1, z2, …, zm]T , образуван от изходните сигнали на невроните от изходния слой е изходния вектор на мрежата, и от скрит слой Y , изходните сигнали на който са вектора Y[y1, y2, …, yp]T. Ако мрежата е трислойна, четирислойна или повече, то тогава броят на скритите слоеве е съответно два, три и т.н. Ще представим невронни мрежи с един скрит слой, тъй като мрежите с повече скрити слоеве са абсолютно аналогични.

Теглата между входния и скрития слой и между скрития и изходния слой сформират матриците:



При инициализирането на мрежата на теглата се присвояват случайни величини в произволен интервал, като теглата могат да бъдат както положителни, така и отрицателни – обикновено се използва интервал [-a, a], a є (0 , 1].

Входният сигнал на неврон (j-ти неврон) от скрития слой се изразява чрез претеглената сума на входните сигнали на неврона:



където ***yj*** е изходния сигнал на неврона, а *f* е активационната функция. Обикновено активационната функция е сигмоидална, но в зависимост от решаваната задача тя може да бъде сигмоидална, линейна или друга.

Обучението на невронните мрежи се извършва върху обучаваща извадка с N желани входно – изходни набори от данни за мрежата - x(i), r(i), където

x(i) = [x0(i), x1(i), … xn(i)]Т , x0(i) = +1, и r(i) = [r0(i), … rm(i)]Т i = 1 ÷ N, са съответно входни сигнали и желан изходен вектор. След изчерпване на елементите от обучителното множество, процедурата продължава да бъде изпълнявана върху същото множество докато не бъде удоволетворен съответния критерий за спиране. По този начин обучението на мрежата може да трае няколко епохи, като една епоха е един път изпълнението на алгоритъма върху тренировъчните данни. Критерият за спиране може да бъде стойността на целевата функция ( функция на грешката), към която се стремим. Целта на обучението е да бъдат настроени теглата vij , *i = 0 ÷ n, j = 0 ÷ p ,*между входния и скрития слой , и теглата wij , *i = 0 ÷ p, j = 0 ÷ m,* между скрития и изходния слой, така че векторът от изходния слой да се доближава максимално до желания резултат.

При изчисляване на изходните сигнали на невроните съответно от скрития и изходния слой се използват следните зависимости:

, където *, j = 0 ÷ p*



, където , *k = 0 ÷ m*



Подходът, който използваме за минимизиране на грешката е градиентен подход, т.е. правим стъпки обратни на градиента на функцията(градиента е вектор, образуван от частните производни на функцията по всичките й променливи, той определя в коя посока функцията расте най-бързо => стъпките са в посока обратна на градиента, тъй като ние искам да я минимизираме). Този подход е от типа методи на най-малките квадрати, тъй като целевата функция, която минимизираме е образувана от квадратите на грешките за всеки един от невроните в изходния слой.

Нека се намираме на стъпка t от обучението на мрежата, където x(t) = [x0(i), x1(t), … xn(t)]Т и r(t) = [r0(t), … rm (t)]Т са съответно входните сигнали и очаквания резултат. Тогава целевата функция (ф-цията на грешката) има следния вид:



Тъй като изходните сигнали на невроните от изходния слой зависят и от теглата между входния и скрития слой, и от теглата между скрития и изходния слой, то E е функция на елементите на матриците V и W (E = E(V,W)). Тогава за да намерим компонентите на градиента на целевата функция трябва да намерим първите производни на функцията спрямо теглата и на двете матрици.

За намиране на частните производни ще използваме малки и големи букви, като с големи букви ще отбелязваме теглото, по което диференцираме, а с малки ще отбелязваме индекси при сумиранията.

Частните производни на средноквадратичната грешка по отношение на всяко от теглата между изходния и скрития (W) слой има вида:



като последното равенство следва от това, че и само съдържа .



Нека с δк бележим мащабираната с грешка на *k*-ти неврон от изходния слой, а с ∆wJK бележим донастройката на теглото wjk., т.е



и ∆ ,



където с α се определя стъпката на промяната. Ако α е с голяма стойност, то процедурата по настройката на теглата може да не приключи, а ако α е много малка, то тогава процедурата ще е много бавна.

При горните означения новите стойности на теглата от изходния слой са следните:

*, K = 0 ÷ m, J = 0 ÷ p*



Частните производни на средноквадратичната грешка по отношение на всяко от теглата между входния и скрития слой има вида:



От грешката за неврон от изходния слой =>



*,*



тъй като и само *съдържа .*



*(тъй като )*



Нека с δJ бележим мащабираната с грешка на изходния сигнал на  *j*-ти неврон от скрития слой, а с ∆vIJ бележим донастройката на теглото vIJ., т.е



и



При горните означения новите стойности на теглата от скрития слой са следните:

*, I = 0 ÷ n, J = 0 ÷ p*



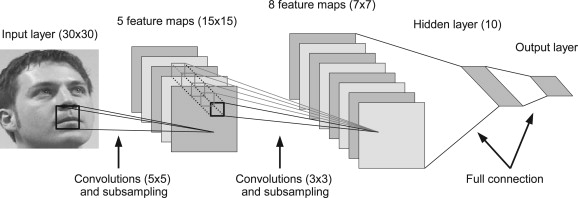
Това разпределение на сигнала и разпространение на грешката продължава докато не се достигне критерия за спиране или докато някакво друго условие не е изпълнено. След което обучението на мрежата е приключило. За проверка на генерализиращите свойства на мрежата са използвани данни различни от тези за обучение.

**4. Конволюционна невронна мрежа**

**Обосновка**

В основата на конволюционните невронни мрежи стои идеята, че за OCR би било по-добре да не се разчита на невронна мрежа с обща структура, а по-скоро на специализирана такава. Трябва да се лавира внимателно между възможностите за самообучение на мрежата и нейната специализираност /нагоденост/ към даден специфичен проблем. Едната крайност са ръчно написани алгоритми за откриване на специфични „белези” в изследваното изображение и чак на по-късен етап се разчита на ИИ. Другата е да се използват невронни мрежи без изразена специфична структура /както първата мрежа в нашия проект/. Разчита се мрежата да се „нагоди” към проблема чрез подходящо /и дълго/ трениране.

Нека вземем предвид силно изразената локалност на данните /пикселите/ при OCR. От много голямо значение за крайното възприятие са пиксели, които са близо един до друг. Те формират „белези” – чупки, ръбове, кръстчета, ъгълчета и други. Тези белези от своя страна в съвкупност формират композиции, които вече са лесноразпознаваеми като символи. Целта в крайна сметка е да се вземат предвид релативните разстояния и ъгли между белезите, а не абсолютните, тъй като абсолютните могат да варират много при един и същи символ. Образно казано конволюционната невронна мрежа може да се разглежда като съставена от две половини – първата е „търсач” и „композитор” на белези, а втората е общ невронен класификатор – т.е. един или няколко слоя неврони, свързани всеки със всеки.



В първата половина се редуват слоеве за конволюция и семплиране. В първия слой за конволюция се намират известен брой карти на белезите. Всяка карта отговаря за точно един белег. Това се постига като невроните в тази карта споделят общи тегла на връзките към входния слой т.е. прилагат се едни и същи тегла, но на различни пиксели. Така при подходящо подбрани тегла в дадена карта на белезите ще се разграничават лесно /т.е. чисто пространствено/ точно еднаквите белези. На следващ етап се взимат семпли, на пример всеки четен пиксел по х и по у. По този начин се скъсяват разстоянията между белезите и се изхвърлят ненужни или повтарящи се данни. Скъсяването е необходимо, за да може белезите да бъдат „хванати” от конволюцията на следващия слой. Като цяло идеята може да се изрази в трансформацията на едно сравнително високорезолюционното изображение в серия от по-нискорезолюционни, но с диференцирани „белези” помежду им.

**Имплементация**

Използваната идея е взаимствана от мрежата LeNet-5 на Dr. Yann LeCun. Тя е широкоизползвана за OCR на ръкописен текст. В проекта ни е реализирана модифицирана нейна версия, в която слоевете за конволюция и събсемплинг са смесени, но пък е запазена идеята за споделените тегла в рамките на всяка карта от белези. Също така всеки неврон получава не само претеглен вход от предшестващите го неврони, а и адитивна част – отклонение, което също подлежи на обучение. Използваната активираща функция е хиперболичен тангенс **tanh(x)**. Тя е избрана поради редица причини. Мрежата работи със сигнали от -1 до +1, а функцията клони точно към тези стойности в безкрайностите. Също така производната се смята лесно, дори може да бъде изразена чрез стойността на функцията в същата точка – не е нужен отделно аргумента. Тестването също показа, че хиперболичния тангенс се справя по-добре от скалиран и транслиран сигмоид. Използваната функция за грешка е сумата от квадратите на разликите. Използва се стандартен backpropagation на базата на алгоритъма Gradient descent. Имплементацията включва гъвкаво избиране на броя на картите във всеки слой както и на броя „силно”-свързани върхове в последните слоеве.

Ще използваме следните означения:

 - стойност на изхода на i-ти неврон в n-ти слой

 - тегло на реброто свързващо i-ти неврон в n-ти слой и j-ти неврон в (n-1)-ви слой

 - адитивно отклонение за i-ти неврон в n-ти слой

- активационна функция



Нека изчислим производната на активационната функция /тя ще ни трябва по-късно/



Определяме и функцията за оценка на грешката в n-ти слой

където  е желаната стойност в i-ти неврон на последния n-ти слой

Идеята на **Gradient descent** алгоритъма е да намерим частните производни на относно всички тегла . След това модифицираме тези тегла в посока обратна на частната производна умножена по коефициент  който изразява скоростта на учене на мрежата. Тъй като се предполага, че читателят е запознат със стандартния backpropagation алгоритъм, тук ще обсъдим само специфичните моменти при конволюционните мрежи. А това са точно споделените тегла. За разлика от стандартните невронни мрежи, в които на всяко ребро съответства по едно тегло, в конволюционните слоеве на LeNet-5 сравнително голям брой ребра споделят едно тегло. Когато взимаме частната производна на грешката относно тегло, трябва да вземем предвид всички двойки неврони, който се „свързват” с него.





В резултат получаваме формула за определяне на частните производни в последния слой.



А също така и междинните слоеве, като k-тия зависи от (k+1)-вия.



Накрая променяме теглата по следния начин:



Например  е добра начална стойност.

Нека уточним уточним топологията на конкретната имплементация:

* Входен слой – 29х29 неврона, по един за всеки входен пиксел.
* Първи конволюционен слой – **A** на брой карти с белези, всяка от тях съдържа 13х13 неврона. Всеки неврон отговаря за конволюция с ядро 5х5 около съответния и пиксел във входните данни /взимат се само центрове с нечетни координати, за да се прави subsampling/. В крайна сметка всеки неврон е свързан с 25 + 1 (за bias) ребра. В рамките на една карта с белези се споделят общо 25+1 тегла.
* Втори конволюционен слой – **B** на брой карти с белези, всяка от тях 5х5. Пак се прави конволюция с ядро 5х5 и с нечетни центрове относно предния слой. Тъй като на предишния слой има **A** карти следователно към всеки неврон ще има **A\*5\*5+1** ребра. Разбира се, всички тегла в дадена карта се споделят.
* Силносвързан слой – **C** на брой неврона, всеки от който е свързан със всички неврони от предишния слой. Няма споделени тегла.
* Изходен слой – **D** (обикновено 62) неврона, всеки свързан с всички неврони от предишния слой. Няма споделени тегла.

**Резултати**

Алгоритъмът се справя отлично дори и в „шумна” среда. Все пак дизайна му е предназначен за ръкописни цифри, а в нашия проект /макар и леко „окастрен”/ работи за печатни букви и цифри с не чак толкова вариации и шум. Само за цифри сходимостта е невероятно бърза – в рамките на 15 епохи трениране се достига до 100% разпознаване, а когато се прибавят големи и малки букви са нужни между 100 и 200 епохи за над 98% точност. Разбира се, изисква се и по-„широка” LeNet – т.е. такава с повече неврони.

Примерна крива на обучение за шрифт **Verdana** и параметри на мрежата **A=20 B=80 C=200 D=62**.



**Бъдеща работа**

1. Поради понякога бавната сходимост трябва да се реализира оптимизационен метод от 2ри ред – например  методът на Levenberg–Marquardt или на Gauss-Newton.
2. Реализиране на LeNet-5 в пълния и вид.
3. Експериментиране с различно мащабиран хиперболичен тангенс за активираща функция.
4. Експериментиране с различна топология на мрежата.
5. Multithreading – поради многото паралелизъм в алгоритъма.
6. Създаване на инфраструктура за обучение и тестване с две отделни множества и динамична статистическа промяна на скоростта на учене на мрежата с цел по-бърза сходимост.

**Използвани материали**

* + Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, ["Gradient-Based Learning Applied to Document Recognition,"](http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf)Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. [46 pages]
  + Y. LeCun, L. Bottou, G. Orr, and K. Muller, ["Efficient BackProp,"](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)in Neural Networks: Tricks of the trade, (G. Orr and Muller K., eds.), 1998. [44 pages]
  + Patrice Y. Simard, Dave Steinkraus, John Platt, ["Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis,"](http://research.microsoft.com/~patrice/PDF/fugu9.pdf) International Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society, Los Alamitos, pp. 958-962, 2003.



* + <http://www.codeproject.com/Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi>
  + <http://yann.lecun.com/exdb/lenet/>

**5**. **Изготвяне на данни за сравняване на успеваемостта на невронните мрежи**

Също както в точка (2) се използва командния редактор **ImageMagick®,** но с променена команда:

*convert -fill "#000000" -font <font> -pointsize 350 label:"<symbol>" -trim -resize "29x29!" -type Grayscale -depth 8 <folder>/<symbol>/<font>.png*

Целта е да се създадат картинки, в които да участват всички видове сиво и да се доближат максимално до изходния формат от модула за подготовка на изображението. Заради това вместо цветовете черно и бяло вече се включват и всички нюанси на сивото. За да тестваме как се справят невронните мрежи се включват още два шрифта: Impact и Rabbit-on-the-moon. Impact, включва по-дебели букви, докато при Rabbit-on-the-moon буквите са по-раздвижени и нестандартни

|  |
| --- |
|  |
| Цифри на Rabbit-on-the-moon |

|  |
| --- |
|  |
| Цифри на Impact |

Използвана литература:

<http://en.wikipedia.org/wiki/Grayscale>

<http://lodev.org/cgtutor/filtering.html>

<http://en.wikipedia.org/wiki/Otsu's_method>

<http://www.cs.cmu.edu/~chuck/lennapg/>

<http://www.dafont.com/rabbit-on-the-moon.font>

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

<https://github.com/gibona/fmi-ocr-neural-network>