

**Факултет по Математика и Информатика**

**Извличане на Информация и Откриване на Знания**

**К У Р С О В А Р А Б О Т А**

**SPELL CHECKER ЗА НЕВРОННА МРЕЖА - OCR**

**Изготвил: Борислав Красимиров Сираков**

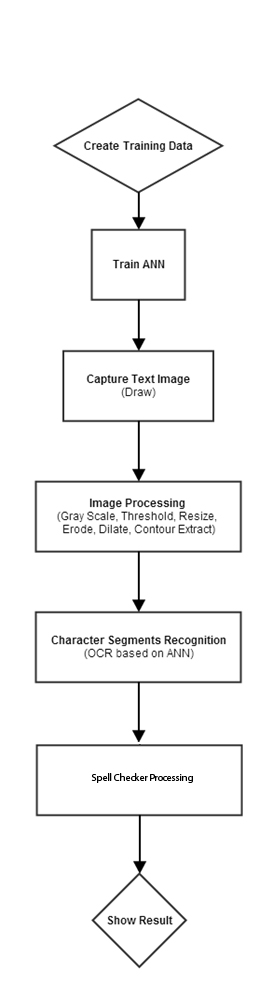
**Ф.Н. 24793,**

**План на курсовата работа**

1. **Схема на Процесите**
2. **Реализация**

* Генериране на тренировъчни данни
* Трениране на ANN
* Създаване на изображение за разпознаване
* Филтриране и сегментация на тестовия пример
* Процес на разпознаване на сегменти
* Представяне на резултат от разпознаването
* Валидиране на разпознатите думи – Spell Checker

1. **Използвани източници**
2. **Схема на Процесите**:



1. **Реализация**

* **Генериране на тренировъчни данни:**

В процеса на генериране на тренировъчните примери бяха взети три основни „Features“. Всяко изображение се разглежда, като квадратна матрица, в чийто клетки се съдържа RGB Color стойност. От тук можем да направим извода, че състоянията, в които се намират пикселите, ще бъдат първият обучаващ feature. Нека го означим с името **RGB Factor**.

Останалите два Feature – а се описват на базата на **хоризонтални и вертикални хистограми**. Преди да обясним тези два вида хистограми, нека най – напред дадем дефиниция на понятието ненулев пиксел. От тук насетне ще считаме, че това е пиксел притежаващ RGB стойност (0, 0, 0 – черен цвят).

От тук следва, че при първият вид(хоризонтален) хистограма се извършва преброяване на ненулевите пиксели, от съответен пореден ред, докато при другия вид(вертикален) се извършват същите операции, но по вертикалата. В резултат образуваме масив, съставен от пребрoените ненулеви пиксели (нормализирани, посредством делене на максималната открита стойност).

По този начин формираме краен едномерен масив, съставен от стойностите на вертикалната, хоризонталната хистограма и стойностите на пикселите на съответното зададено изображение. Трябва да отбележим също, че посочения image преминава най – напред през процес на нормализация, състояща се конвертиране в Gray Scale, Color Invert и Resize.

Посочените стъпки се прилагат върху цялото множество от изображения, а получените резултати запазваме в ArrayList.

След успешното приключване на тези операции, ArrayList се обхожда, с цел съхраняване на стойностите му във файл, като всеки “sample” се съхранява на нов ред, с първи елемент **клас** и останалите параметри разделени със запетая. Класа приема Integer стойност, съответно 0,…,9 за цифрите от 0 до 9,

Докато за буквите A…Z се използват 10,…,35.

* **Трениране на ANN**

Най – напред нека разгледаме структурата на нашата невронна мрежа:

**Структура:**



Броят на Input възлите е равен на броя на елементите получени при хоризонталната и вертикална хистограма, плюс описанието на thumbnails (поредица от всички пиксели 20x15). От тук получаваме краен брой възли, равен на 510. За Hidden нивото използваме 700, а за изходите – броя на класовете, т.е. 35. Възникна въпросът: Как ще представим target стойността?

Посредством едномерен масив с размер N, при който N – 1 елемента са със стойност 0, а останалия един елемент приема стойност 1. Позицията му зависи от съответния клас,т.е. получаваме че N е равно на броя класове =>

N = 35, а елемента със стойност 1, отбелязва кой е класа. Трябва да отбележим и факта, че в началото всички тегла приемат произволни стойности в диапазона [-0.05, 0.05].

Описана до тук структура се изгражда чрез метода setNetwork от класа Network.

**Алгоритъм:**

Използваме Backpropagation алгоритъм, със активираща функция sigmoid:

От тук следвам следните етапи:

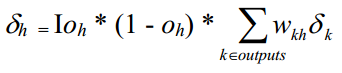
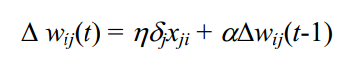
1. y = \sum_{i=1}^{n}w_ix_iSumming Function - сумата от произведението на тегла и вход на всички входни възли / ребра към съответен възел от Hidden Нивото.
2. Activation Function – Sigmoid

http://www.heatonresearch.com/images/article/1604390085/1604390085c5e1.jpg

1. За пресмятане на грешката използваме:

\frac{\partial E}{\partial w_i} = \frac{dE}{dy} \frac{dy}{d\mathrm{net}} \frac{\partial \mathrm{net}}{\partial w_i}

****\frac{\partial E}{\partial w_i} = (y - t) y (1 - y) x_i

1. ****За промяна на стойностите на ребрата използваме (momentum & rate)

Стъпки 1, 2 намират своята реализация във вече споменатия клас Network, метод getOutputs. 3 се изпълнява от calculateError, докато 4 е интегриран в updateWeights.

* **Филтриране и сегментация на тестовия пример**

След задаване на тестови пример (статична картинка с текст), при натискане на бутона Recognise се извършват няколко основни стъпки, водещи до сегментиране на посочения пример, т.е. при задаване на изображение с K на брой символа, в края на процеса се създават K + s (K, S >= 0 ) отделни изображения, всяко съдържащо по един символ. В K пазим необходимите ни сегментирани картинки, докато в S – trashy елементи, който трябва да филтрираме. В класа ImageProcessing( метод filterImage): е реализирана тази функционалност, която може да бъде разделена на следните стъпки:

1. Прилагане на medianBlur, с цел изчистване на ненужни артефакти.

Imgproc.*medianBlur*(image\_original, image\_gray, 5);



1. Конвертиране на изображението в Gray Scale.

Imgproc.*cvtColor*(image\_gray, image\_gray, Imgproc.*COLOR\_RGB2GRAY*); 

1. Прилагане на филтър threshold. Чрез него правим invert на цветовете и отстраняваме ненужните детайли. Приема се че няма да има загуба на нужните за нашите цели символи, поради контраста между тях и фона. В резултат отпадат ненужни пиксели.

Imgproc.*threshold*(image\_gray, image\_filter, 128, 255, 0); 

1. Прилагане на dilate, erode филтри, подобряват детайлите на символите.

Imgproc.*dilate*(image\_filter, image\_filter, Imgproc.*getStructuringElement*(Imgproc.*MORPH\_RECT*, **new** Size(2,2)), **new** Point(-1, -1), 2);



Imgproc.*erode*(image\_filter, image\_filter, Imgproc.*getStructuringElement*(Imgproc.*MORPH\_RECT*, **new** Size(2,2)), **new** Point(-1, -1), 8);



1. Накрая използваме метода findContours. Чрез него намираме всички сегменти от картинката, съдържащи отделните символи. По този начин извършваме сегментация на изображението. Но веднага се забелязва, че в резултата може да попаднат и ненужни контури. Има редица техники за тяхното отстраняване (SVM и др. класификатори), но за простота в нашата ситуация отхвърляме всички сегменти намиращи се извън определена

област на threshold. Този threshold формираме на базата на Image Ratio.

Imgproc.*findContours*(image\_filter, contours, contours\_info, Imgproc.*RETR\_LIST*, Imgproc.*CHAIN\_APPROX\_SIMPLE*, **new** Point(0, 0));

В края на пета стъпка получаваме списък със сегменти, които трябва да преминат през процеса на разпознаване.

1. Групиране на букви( образуване на думи) – в тази стъпка се осъществява формирането на думи, спрямо групиране на отделните символи. За целта се използват следните съображения:
   * + Извършва се предварително групиране на символите спрямо реда на който се намират. По този начин можем да разчитаме текст на няколко реда.
     + Извършва се същественото формиране на думи. За всеки от вече формираните редове от символи се изчислява най – често срещаната дистанция между символи. Определяме нея за дистанция между букви от една дума, а най – голямото разстояние - за дистанцията между две думи (при формирането на тези дистанции се добавя и threshold). По този начин можем да извършим финално групиране на буквите в думи. Получаваме ArrayList от думи, които трябва да бъде разпознати.

* **Процес на разпознаване на сегменти**

Преди дадено изображение/сегмент да постъпи за разпознаване в невронната мрежа, то най – напред трябва да премине през процес на извличане на вече споменатите Features (**RGB Factor**, **хоризонтални и вертикални хистограми**). Реализацията на тази функционалност е интегрирана в класа ImageProcessing, метод exportORCData.

В резултат получаваме едномерен масив с дължина 510. На първо място в него поместваме елементите от хоризонталната хистограма, следвани от тези от вертикалната такава, и накрая поредица от пикселите на съответния image (само първата координата от RGB модела), който преди това resize – ваме до размер 20x15. Получения резултат подаваме на ансамбъл от тренирани невронни мрежи (множество от невронни мрежи), като всяка от тях разпознава дадения символ. Накрая на базата на Bagging алгоритъма се осъществява взимане на финално решение. Всяка невронна мрежа има равни права върху финалното решение – мнозинството печели.

* **Представяне на резултат от разпознаването**

Резултата се визуализира в специално предвиден текстов панел. За всяка от думите, се извършва Spell Checking, като тя се оцветява в червено ако съществува причина за некоректност.

* **Spell Checking Process**

Потребителят може сам да извърши проверка за грешни думи, за целта трябва да маркира текст от съответното поле (Input) и да натисне специфичен бутон Is Correct? В резултат получава най – близките възможни форми/кандидати на съответната дума. Остава да се отговори на техническия въпрос: как е реализиран дадения Spell Checker?

За целта се извършва конструиране на специфичен Index – Bigram. За формирането му се използва речник с 110 000 английски думи. Всяка от думите се разбива на срички(2 букви, добавя се символ $ за начало и край на думата), като ги добавяме в HashMap. Всяка от тях сочи(Pointer) към множество то думи от речника (в които се съдържа). При проверка за грешка на дума, се извършва специфично търсене:

1. Търсената дума се разбива на срички.
2. За всяка от тях се извличат думите към които сочат.
3. Получаваме едно множество от думи – кандидати, като за всяка от тях трябва да пресметнем тяхната „отдалеченост“ от оригинално въведената за корекция дума. Тази близост се пресмята посредством дистанция на Levenshtein. В резултат получаваме множество от кандидати, сортирани по тяхната близост до оригинала. Визуализираме само първите 6 по близост думи.
4. **Използвани източници**
   1. **Libraries:**

* OpenCV (JavaCV) Library – чрез нея се извършват всички филтрирания на изображения.
  1. **Books and online Resources**
* **Coursera.org** - Neural network Course – основни концепции и понятия на невронните мрежи.
* IMPROVED BACKPROPAGATION LEARNING IN NEURAL NETWORKS WITH WINDOWED MOMENTUM Ernest Istook, Tony Martinez – формули за по – бързо „учене“ (използване на momentum и speed rate)
* A brief introduction to Neural network David Kriesel.
* <http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/> тестови картинки (png, jpg), на базата на които изграждаме тестови файл с Image Features.
* Introduction to Information Retrieval – Stanford
* SIL organization (Word Dictionary)