Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

Android based Mathematical Expression Evaluation from Images

Aanal Patel
IT Dept., LDRP-ITR,
Sector-15, Gandhinagar,
Gujarat, INDIA

Dhiren Pandit
Science & Humanities Dept., LDRP-ITR,
Sector-15, Gandhinagar,
Gujarat, INDIA

Neel Acharya
CE Dept., LDRP-ITR,
Sector-15, Gandhinagar, Gujarat, INDIA

***Abstract*—Current era is an era of automated systems and machine learning plays vital role in construction of automated systems. In machine learning, neural network is key tool. OCR is used for recognition of characters from printed text and mathematical tool of neural network is used for classification. Proposed system combines OCR and Mathematics, hence a system is designed that can compute mathematical expressions from images which contains equations. OCR systems can be used to convert the image of expression to a string of mathematical expressions and then combined with Mathematical libraries and compiler to obtain solution of such mathematical expressions**

***Keywords*— OCR, Expression, Android, Neural Networks, Evaluation, Image Processing, Character detection, Printed Character Recognition**

I. INTRODUCTION

Closed-Form Expression evaluation is one of the easiest task in mathematics. Its Computer application is also one such easy task. With built-in operators many mathematical simple equations can be easily solved using computers. However, with increase in mathematical complexity the solution of the equation becomes increasingly tougher task. This is due to need of more complex techniques of problem solution and more complex algorithmic procedures. Many such systems exist today that help us solve complex mathematical equations. With combination of such systems and an Optical Character Recognition system this problem can be transformed to a new domain of printed expression calculation.

II. M-EXPRESSION FOR MATHEMATICAL EXPRESSION EVALUATION

M-Expression is one such android and java based system under development that does the task of printed expression calculation. The front-end is an android application that allows a user to choose an image of printed expression which is then forwarded to the java based back-end that does the task of OCR and then obtains the expression in a form that the java code can calculate it. This is then further processes with a math library to compile and find errors if any. In case of no errors, the expression is evaluated and a response comprising of calculated answer is sent to the front-end. For the task of OCR, a neural network based approach has been taken that uses a java library to train the dataset of fonts which can be used to classify the test image. Along with a basic set of mathematical operations (like addition, subtraction, multiplication, division, power), relational operations (like equals, not-equals, less-than, greater-than, less-than or equals, greater-than or equals) Boolean operation (like and, or, not) are supported. Also trigonometric functions (like sine, cosine, ...) and other complex mathematical operations (like ceil, floor, log, ...) are supported.

A. Flow Diagram

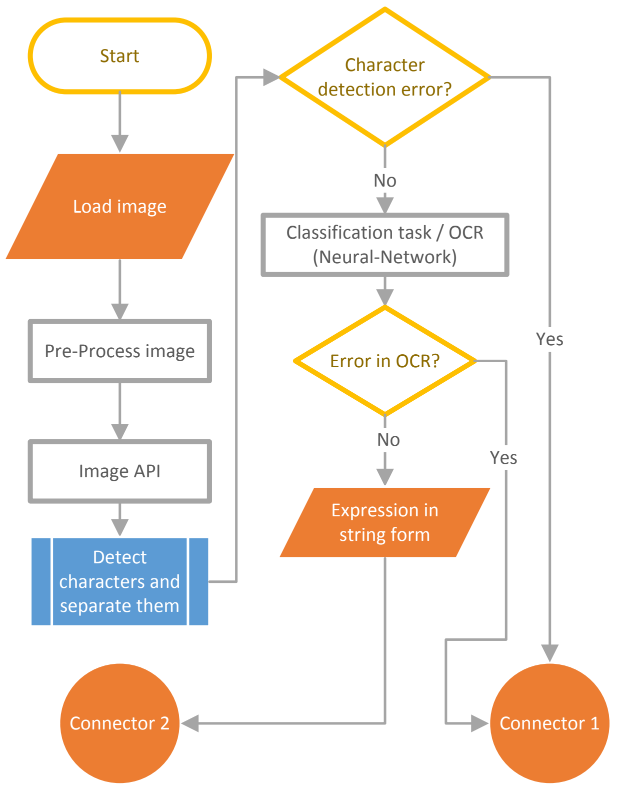

Start
Load image
Pre-Process image
Image API
Detect characters and separate them
Character detection error?
No
Yes
Classification task / OCR (Neural-Network)
Error in OCR?
No
Yes
Expression in string form
Connector 2
Connector 1

Fig. 1. Flow-chart part-1

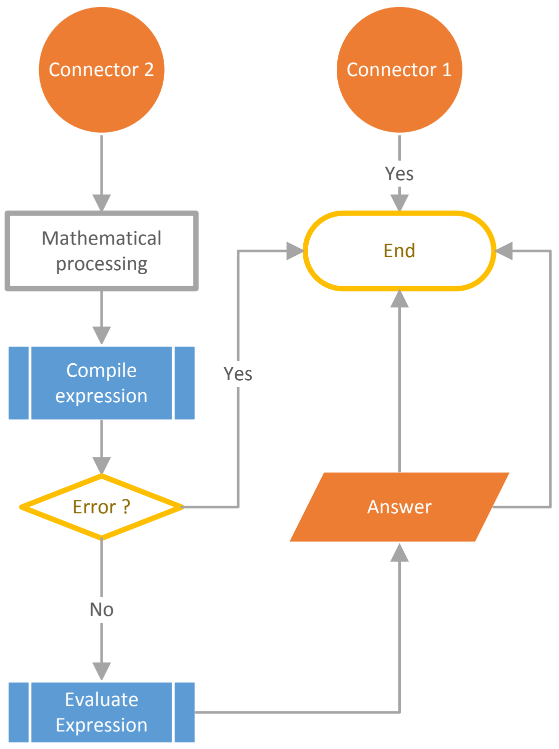


Fig. 2. Flow-chart part-2

B. *Preprocessing*

Once an image of mathematical expression is loaded in the system, the image is pre-processed. Pre-processing step includes enhancing the image and grey scale conversion. This step makes the image ready to be properly processed and removes chances of minor errors in the OCR process [1]. For this purpose, different filters are used. In this work following filters are used for better recognition rate.

1. *RGB to Grayscale (Dimension Reduction)*

Given input colored image has three dimensional pixel value (RGB). Hence matrix obtained for this input colored image is three dimensional matrix. It is quite difficult to perform image processing technique on three dimensional matrix, hence for suitable and smoother processing the image is converted in gray format. A gray image has two dimensional pixel value which lies between [0, 255] as per the gray value of a pixel [2].

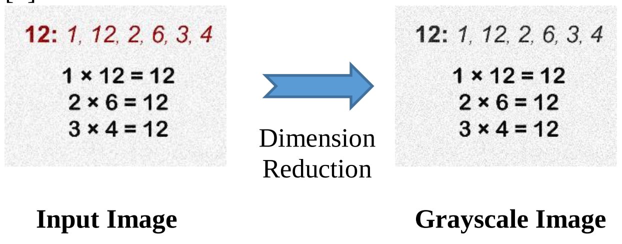


Fig. 3. Dimensional Reduction using RGB to Gray

i. *Gaussian Blur filter*

Once grayscale image is obtained, filters are applied for smoothing image. Smoothing, also called blurring, is a simple and frequently used image processing operation. Many reasons are there for use of this filter but here smoothing filters are used in order to reduce noise. There are many filters for smoothing but the most common type of filters are *linear*, in which an output pixel's value $(g(i,j))$ is determined as a weighted sum of input pixel values $(f(i+k,j+l))$. But most useful filter is Gaussian filter. Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array. Following is an image of 1D Gaussian kernel.

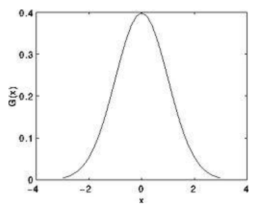


Fig. 4. 1 D Gaussian kernel

Assuming that an image is 1D, it can be noticed that the pixel located in the middle would have the biggest weight. The weight of its neighbors decreases as the spatial distance between them and the center pixel increases.
2D Gaussian can be represented as:

$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}+\frac{-(y-\mu_y)^2}{2\sigma_y^2}}$$

Where μ is the mean(peak) and σ represents the variance.

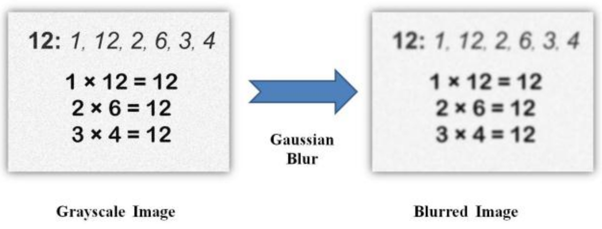


Fig. 5. Effect of Gaussian filter

ii. *Adaptive threshold*

Image binarization or thresholding is an important tool in image processing and computer vision, to extract the object pixels in an image from the background pixels. A number of methods have already been proposed for image thresholding. Bi-level image is used as a pre-processing unit in several applications. The use of binary images decreases computational load for the overall application. These applications include document analysis, optical character recognition system, scene matching, quality inspection of materials etc. The thresholding process computes the threshold value that differentiate object and background pixels. Under varying illumination and noise, the thresholding can become a challenging job. A number of factors contribute to complicate the thresholding scheme including ambient illumination, variance of gray levels within the object and the background, inadequate contrast, object shape and size non-commensurate with the scene. A wrong selection of threshold value may misinterpret the background pixel and can classify it as object and vice versa, resulting in overall degradation of system performance. In document analysis, thresholding is sensitive to noise, surrounding illumination, gray Ievel distribution, local shading effects, inadequate contrast, the presence of dense non-text components such as photographs, etc. There are a number of important performance requirements that need

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

to be considered while thresholding gray level images. These include:
a) Loss of features after thresholding input image should be zero or minimum. b) The features (objects) with similar relative gray levels should have same binary values in the processed output image. c) Thc effect of noise of minor gray level variations should be eliminated.

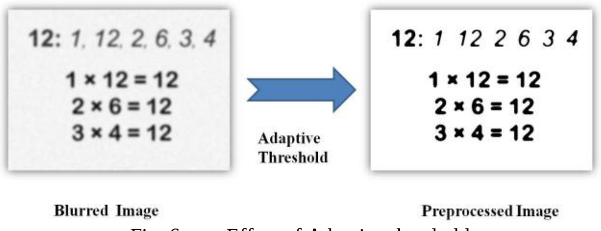


Fig. 6. Effect of Adaptive threshold

C. Character detection

Next the system will separate the image into individual character images. This is done by reading the pre-processed greyscale converted image pixel by pixel. A group of black pixels spaced by a minimum threshold is considered as a character in the image. This can be achieved by moving a variable sized window over the image pixel array and then adjusting the window till a suitable group of pixels with predefined threshold distance has been found [3][4].

D. Classification / OCR

After detecting each character, the next job is to classify the characters into the clusters of predefined characters. To be straight to the target with the arrow, this step involves OCR technique to detect the character. For this purpose, we use a properly trained neural network that does our job.

Neural network consists of n input neurons. Where n=(height)x(width) of individual character in the detection step [5][6][7].
The number of output neuron is m, where m=the number of total detectable characters.
There could be any amount of hidden neurons but it was experimentally found that 11 to 15 hidden neuron layers give optimal result. (IN our case it is 12 hidden layers) [8][9] [10].

Multilayer perceptrons can easily learn tougher patterns & complex decision boundaries using feed forward and back propagation algorithms.

For any input N = (N_1, N_2, N_3, …, N_n) are considered to be input layer. Every connection between 2 neutrons/perceptrons, there is an associated weight, W_{ab} (Weight of a connection from node an in a layer to node b in next layer).

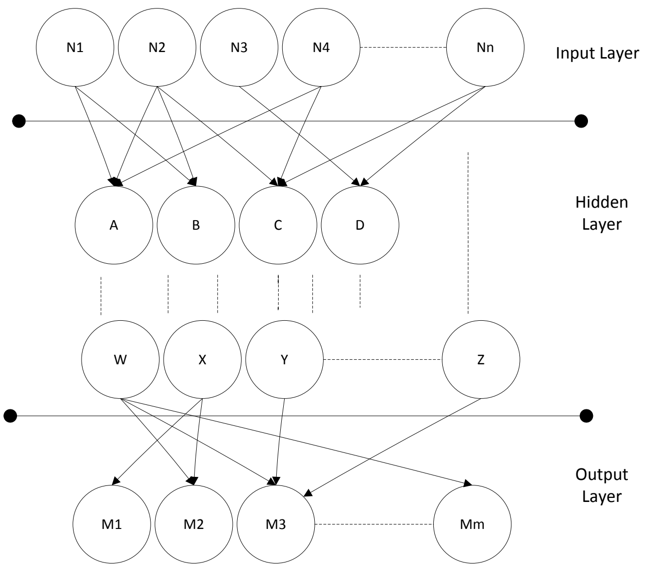


Fig. 7. Neural network: Input layer, Hidden layer, Output layer

Similarly, there are output neurons M = (M_1, M_2, M_3, …, M_m) that provide output of the network. In our case the input values, (N_1, N_2, N_3, …, N_n) consists of the pixels of the input character detected. The output layer in our case consists of m node (neurons) each representing a character and the output at any output node (neurons) is either 1 or 0, 1 at the place of character detected & 0 for the rest of outputs. This outputs are ideal & may deviate slightly in practical world.
There is a connection between input node (neuron) & output node (neuron) via one or more hidden nodes (neurons).
The value of any node is the weighted sum of all the input values to the node.

Back propagation algorithm:

```
• Initialization network weights
• Until the termination:
{
For each training example:
{
Propagate the input forward to the network and
compute the observed outputs.

Propagate the errors backward as follows:
For each network output unit o calculate its error
term
$Eo = Vo (1 - Vo) (To - Vo)$
Where, $V_o$ is the value of node k
For each hidden unit calculate its error term,

$Eh = Vh\ (1 - Vh) \sum_{outputs}^{k} W\, kh\, Ek$

Where, Vk is the value of node k

Finally, update each weight

$Wij = Wji + \Delta Wji$

Where, $\Delta Wji = -\eta . Ejxji$

}

}
```

Important parameters in the above algorithm:

Two most important parameters in BACKPROPAGATION are the learning rate and the momentum.

- Learning Rate: in the above algorithm the weights of the nodes are updated by:
  $$\Delta Wji = -\eta . Ejxji \quad ,$$
  this parameter ΔWji is called the learning rate.

- Momentum**:** The weight update part of the equation can be modified in a way that the update in nth iteration also affected by the previous iteration in multiples of what is called a momentum factor. By adding this term to the formula, the update rule will be: $\Delta Wji = -\eta . Ejxji + \alpha \Delta Wji\ (n - 1)$. Momentum takes values in the range $0 \leq \alpha < 1$.

E. Mathematical-Processing (Compiling + Evaluation)

For compiling a mathematical expression in we use a classical calculator stack approach that pushes in the operators and operands one by one into a stack and once everything is in the stack, the stack is evaluated one operator at a time by pushing and popping as per the precedence and associativity of the operator. In case of any error, the process is stopped and error is returned.

III. STANDARD DATASET FOR EXPERIMENT

For testing of the above mentioned procedure, a dataset consisting of 7 different fonts was used.

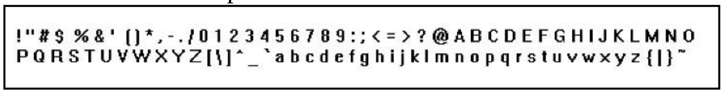
All the mentioned fonts were tested for characters in ASCII sequence of 32 (space) to 126 (~)

Fonts:

TABLE I. . FONTS USED FOR TESTING

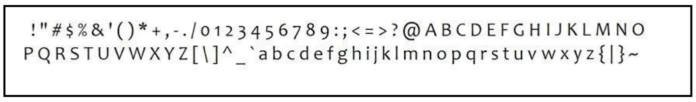
Shruti	Candara
Times New Roman	Carlito
Consolas	System
Book Antiqua	

Below are a few samples of test dataset



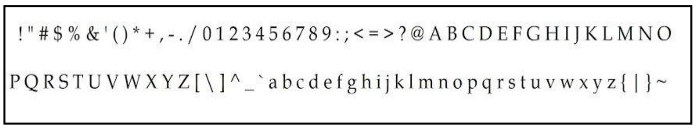
```
!"#$ %&' ()*,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

System Font



```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Canadra Font



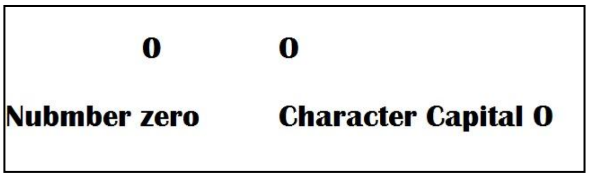
```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

Book-Antiqua Font

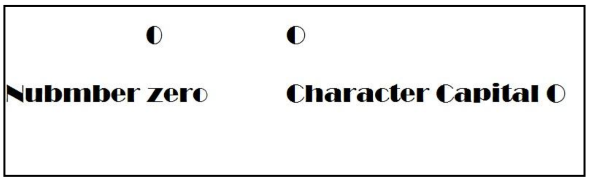
Fig. 8. Image used for testing

With the setup as mentioned above and 7 datasets, the results were quite impressive. Efficiency pars the OCR process at approximately 91% and the Mathematical evaluation is quite effective to calculate all the operators thrown at it.

Also it is important to note that for certain fonts with similar looking characters, the error rate was high. For Example, for fonts Broadway & Britannic O (capital O), & 0(Zero) looked almost same, so were erroneous.


0
0
Nubmber zero
Character Capital O

Britannic Font confusion for Zero & ‘O’


0
O
Nubmber zero
Character Capital O

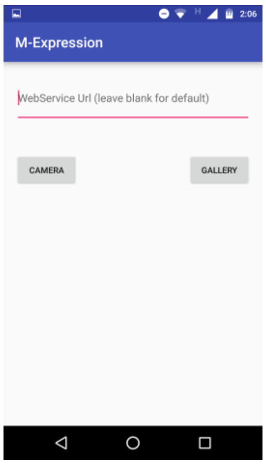
Broadway Font confusion for Zero & ‘O’

Fig. 9. Confusion for Zero & ‘O’ in different fonts

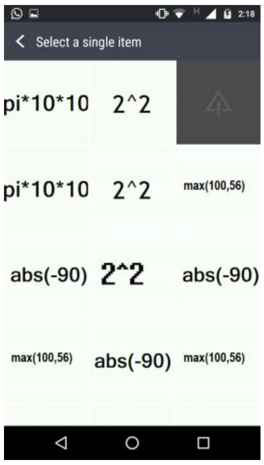
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 08, August-2016

Procedure for the app:

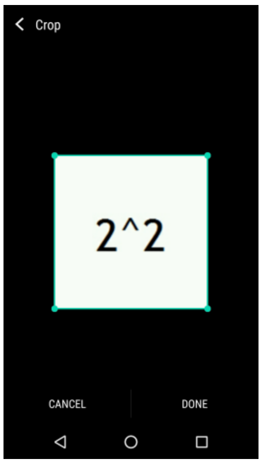
- Step 1 + 2 + 3 + 4:



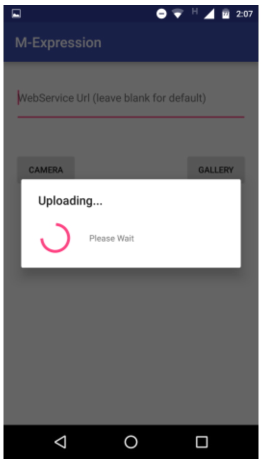
Step 1: Open the application



Step 2: Select the expression image (capture via camera or browse in the gallery)



Step 3: Crop the image to select proper expression



Step 4: Wait for the processing

Fig. 10. App screenshots (1)

- Step 5: Check output

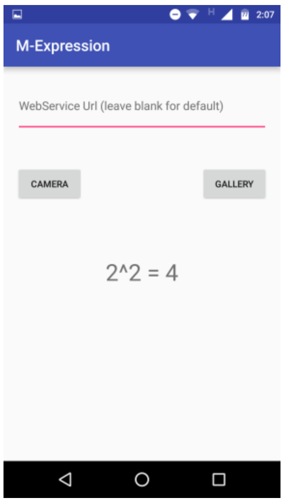
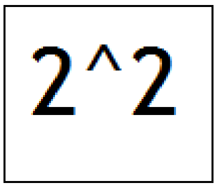
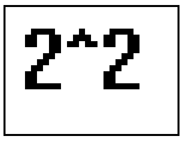
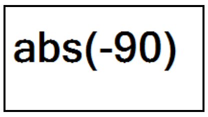
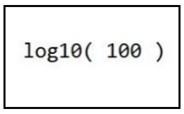
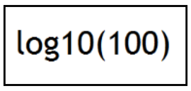


Fig. 11. App screenshots (2)

TABLE II. . EXPERIMENTAL RESULTS: INPUT IMAGE AND EVALUATED EXPRESSIONS

Image	Detected Expression	Evaluated Expression
	2^2	4
	2^2	4
	abs(-90)	90
	loglo(100)	Error: no operator loglo
	log10(100)	2

max(100,56)	max(100,56)	100
PI*10*10	PI*10*10	314
pi*10*10	Pi*10*10	314
sin(90)	sin(90)	1
tan[90]	tan(90)	16331239350 000000 ≈ infinite
tanh[90]	tanh(90)	1

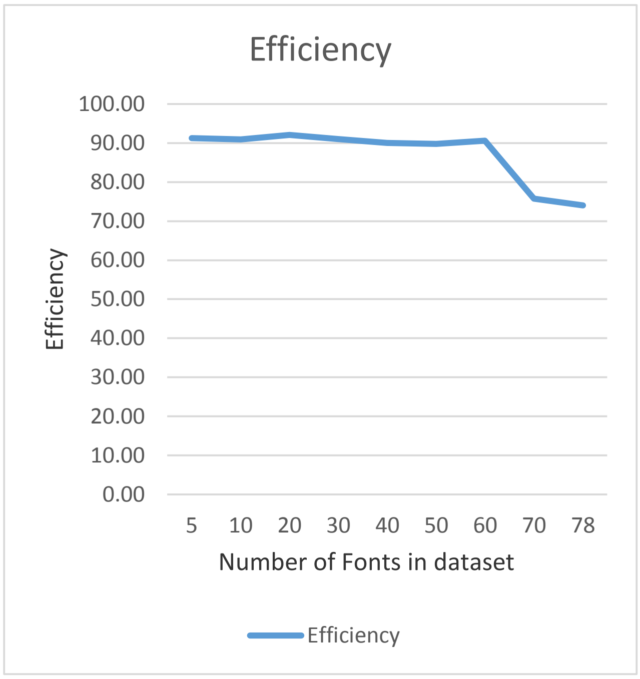


Fig. 12. Experimental results: Input image and Evaluated Expression

With increase in the fonts in the dataset, we are able to achieve a stable efficiency rate of about 90% to 92%. With increase in font for a certain amount, the efficiency went on to plateau and further more increase led to decline of efficiency due to ambiguity in classification.

REFERENCES

[1] P. P. Roy, J. Llados, U. Pal, “Text/graphics separation in color maps”, in: Computing: Theory and Applications, 2007. ICCTA' 07. International Conference on, IEEE, 2007, pp. 545-551.

[2] Li Peng, Junhua Li ,“A Facial Expression Recognition Method Based on Quantum Neural Networks”, Publication: ISKE-2007, October 2007

[3] R. Singh, C. Yadav, P. Verma, V. Yadav, “Optical character recognition (ocr) for printed devnagari script using artifcial neural network”, international Journal of Computer Science & Communication 1 (1) (2010) 91-95.

[4] M. S. Uddin, T. Rahman, U. S. Busra, M. Sultana, “Automated extraction of text from images using morphology based approach”, proceeding of international journal of electronics & informatics 1 (1).

[5] F. Gounther, S. Fritsch, “neural-net: Training of neural networks”, The R journal 2 (1) (2010) 30-38.

[6] C. M. Bishop, “Neural networks for pattern recognition”, Oxford university press, 1995.

[7] S. B. Maind, P. Wankar, “Research paper on basic of artificial neural network”, International Journal on Recent and Innovation Trends in Computing and Communication 2 (1) (2014) 96-100.

[8] C. Peterson, T. Rognvaldsson, L. Lonnblad, “Jetnet 3.0a versatile artifcial neural network package”, Computer Physics Communications 81 (1) (1994) 185-220.

[9] K. Hornik, “Approximation capabilities of multilayer feedforward networks”, Neural networks 4 (2) (1991) 251-257.

[10] R. Lippmann, “An introduction to computing with neural nets”, IEEE Assp magazine 4 (2) (1987) 4-22.