

**DOKUZ EYLÜL UNIVERSITY**  
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**ONLINE HANDWRITTEN MATHEMATICAL  
EXPRESSION RECOGNITION**

by  
**Onur ÖZDEMİR**

**June, 2015**  
**İZMİR**

# **ONLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION**

**A Thesis Submitted to the  
Graduate School of Natural and Applied Sciences of Dokuz Eylül University  
In Partial Fulfillment of the Requirements for the Degree of Master of Science  
in Computer Engineering**

**by  
Onur ÖZDEMİR**

**June, 2015  
İZMİR**

## M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**ONLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION**” completed by **ONUR ÖZDEMİR** under supervision of **DR. ÖZLEM ÖZTÜRK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Dr. Özlem ÖZTÜRK

Supervisor



Yrd. Doç. Dr. Hatice DOĞAN

(Jury Member)



Yrd. Doç. Dr. Sen ÇAKIR

(Jury Member)



Prof. Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

## **ACKNOWLEDGMENTS**

I would like to thank my adviser Dr. Özlem ÖZTÜRK for her support throughout my master study. Her guidance and experiences have expanded my vision.

Onur ÖZDEMİR

# ONLINE HANDWRITTEN MATHEMATICAL EXPRESSION RECOGNITION

## ABSTRACT

Widespread usage of smartphones and tablets and their input and output units which are on the same surface, make softwares which support to automatic detection of handwritten symbols and drawings and allow to reorganization in creation of e-documents, more important.

Aim of this study is to develop a system which supports to recognition of online handwritten mathematical expression and solving equations and creation of its documents on touchscreen devices. The online handwritten mathematical expression recognition system is composed of five main modules: Symbol detection, symbol recognition, parsing, exporting and solving. In symbol detection, minimum spanning tree data structure is used so as to detect symbols from the strokes of expression. The detected symbols are sent to the symbol recognition module. Here, strokes of the detected symbols are converted to standart form by being preprocessed. Soon after features which is used for classification are extracted. Afterwards, symbols are recognized by using k-means clustering algorithm. All the recognized symbols are sent to parsing module in order to find hierarchical structures of the input expression according to predefined mathematical rules. Then parsed expression is converted to AsciiMathML and MathML which are standard mathematical markup language. Finally, text expressions are sent to solving module so as to determine the types of equations and solving process is performed.

**Keywords:** Online handwritten recognition, pattern recognition, machine learning

## EL YAZISI İLE YAZILAN MATEMATİKSEL İFADELERİ TANIMA

### ÖZ

Akıllı telefon ve tabletlerin yaygınlaşması ve bu cihazların girdi çıktı birimlerinin aynı yüzeyde olması e-belgelerin oluşturulmasında el yazısı sembol ve çizimlerin otomatik algılanmasına ve yeniden düzenlenmesine imkan sağlayan yazılımları önemli hale getirmiştir.

Bu çalışmada dokunmatik ekrana sahip cihazlarda el yazısı ile girilen matematiksel ifadelerin tanınmasını, çözülmesini ve belgelerinin oluşturulmasını sağlayan sistemin geliştirilmesi amaçlanmaktadır. Çevrimiçi elyazısı ile yazılan matematiksel ifadeleri tanıma sistemi beş ana kısımdan oluşmaktadır: Sembol tespit etme, sembol tanıma, ayrıştırma, dışa aktarma ve denklem çözme. Sembol tespit etme kısmında asgari tarama ağacı veri yapısı kullanıcı tarafından girilen ifadedeki vuruşların birlikte veya tek olarak sembol olup olmadığını tespit etmek için kullanılmıştır. Tespit edilen semboller sembol tanımlama kısmına gönderilir. Burada tespit edilen sembollerin vuruşları önışlemden geçerek standart forma getirilir. Sonrasında standart forma getirelen vuruşların sınıflandırma kısmında kullanılacak olan öznitelikleri çıkarılır. Daha sonra k-ortalama algoritması kullanılarak semboller tanımlanır. Girilen ifadedeki tüm semboller tanımlandıktan sonra ifade belirli matematiksel kurallara göre hiyerarşik yapılara ayrıştırılmak üzere ayrıştırma kısmına gönderilir. Ayrıştırılan ifade dışa aktarım kısmında standart matematiksel biçimlendirme dilleri olan AsciiMathML ve MathML' e çevrilir. Son olarak metin haline getirilen ifade denklem çözme kısmına gönderilir. Burada ifadenin hangi denklem tipine uyduğu tespit edilerek çözümü gerçekleştirilir.

**Anahtar kelimeler:** Çevrimiçi elyazısı tanıma, örüntü tanıma, makine öğrenmesi

## CONTENTS

	Page
M.Sc THESIS EXAMINATION RESULT FORM.....	ii
ACKNOWLEDGMENTS.....	iii
ABSTRACT.....	iv
ÖZ.....	v
LIST OF FIGURES.....	viii
LIST OF TABLES.....	x
<b>CHAPTER ONE – INTRODUCTION.....</b>	<b>1</b>
<b>CHAPTER TWO – ARCHITECTURE.....</b>	<b>4</b>
2.1 Symbol Detection.....	5
2.2 Symbol Recognition.....	6
2.2.1 Preprocessing.....	7
2.2.2 Feature Extracture.....	9
2.2.3 Recognition.....	10
2.3 Parsing.....	13
2.4 Exporting.....	16
2.5 Solving.....	18
<b>CHAPTER THREE – EVALUATION.....</b>	<b>22</b>
3.1 Data Collection.....	22
3.2 Training and Recognition.....	23
3.3 Data Collector Application.....	25
3.4 Math Solver Application.....	27
<b>CHAPTER FOUR – RESULTS.....</b>	<b>31</b>

<b>CHAPTER FIVE – CONCLUSION.....</b>	<b>38</b>
<b>REFERENCES.....</b>	<b>40</b>



## LIST OF FIGURES

	Page
Figure 1.1 The types of handwriting styles: (a) box discrete symbols (b) space discrete symbols (c) run-on discretely symbols (d) pure cursive handwriting.....	1
Figure 2.1 Block diagram of the online handwritten mathematical expression recognition system.....	4
Figure 2.2 Minumum spanning tree of strokes.....	6
Figure 2.3 Found symbols are shown in its bounding box.....	6
Figure 2.4 Shifting process of symbol 4.....	7
Figure 2.5 Shifting and scaling process of symbol 4.....	8
Figure 2.6 Some original symbols which is inputed by the user.....	8
Figure 2.7 Normalized and resampled symbols.....	9
Figure 2.8 K-nearest neighbors algorithm.....	11
Figure 2.9 A single model of a neuron.....	12
Figure 2.10 Some rules : a. Horizantal pairs b. Subexpresion c. Superscript d. Overscript.....	14
Figure 2.11 Bounding box of symbols and baselines belong to x and $\sqrt{\quad}$ .....	14
Figure 2.12 Expression tree of the mathematical expression given in Figure 2.11....	16
Figure 2.13 MathML output for the expression in Figure 2.11.....	18
Figure 2.14 Pseudo code for infix to postfix expression conversion algorithm.....	19
Figure 3.1 The flow of data collection steps.....	22
Figure 3.2 The flow of training steps.....	23
Figure 3.3 The flow of recognition steps.....	24
Figure 3.4 Screenshot of the Data Collector application.....	25
Figure 3.5 An example of a symbol map file. A row of file consists of symbol name, symbol and type.....	26
Figure 3.6 Screenshot of the Math Solver application for simultaneous linear equations.....	27
Figure 3.7 Convert file contents.....	29
Figure 3.8 Solve file contents.....	29

Figure 3.9 Screenshot of Math Solver application for complex expression which consists of horizontal pairs, superscrip and subexpression rules.....	30
Figure 3.10 Screenshot of the Math Solver application for calculation with no unknown and high degree equation.....	30

## LIST OF TABLES

	Page
Table 2.1 Symbol features obtained from the feature extraction submodule.....	9
Table 2.2 Supported MathML tag elements by the current system.....	17
Table 3.1 Descriptions for basic components of Data Collector application.....	26
Table 3.2 Descriptions for basic components of Math Solver application.....	28
Table 4.1 Recognition results of k-nearest neighbors for digits.....	31
Table 4.2 Recognition results of k-nearest neighbors for lower case letters (Latin alphabet).....	32
Table 4.3 Recognition results of k-nearest neighbors for Greek symbols.....	32
Table 4.4 Recognition results of k-nearest neighbors for operators.....	32
Table 4.5 Recognition results of neural networks for digits.....	33
Table 4.6 Recognition results of neural networks for lower case letters (Latin alphabet).....	34
Table 4.7 Recognition results of neural networks for Greek symbols.....	34
Table 4.8 Recognition results of neural networks for operators.....	35
Table 4.9 Accuracy rates of k-nearest neighbors algorithm for different k values.....	35
Table 4.10 Accuracy rates of neural networks algorithm for different number of hidden layer neurons.....	36

## CHAPTER ONE

### INTRODUCTION

Mathematical expression is the combination of numbers, letters and mathematical symbols that have the mathematical meaning described by the rules of mathematical operators (Z. Les & M. Les, 2015). The online handwritten mathematical expression recognition is about the recognition of combination of numbers, letters and mathematical symbols which are written by hand or stylus. This recognition is still difficult problem for computers in spite of being easy for humans. Since writing style varies from person to person. Besides symbols can be various in shape and size depending on type and speed of the handwriting. Different writing styles are given in Figure 1.1.

Grouping strokes into symbol is one of the recognition problems. It is called stroke segmentation or symbol detection problem. In case of box discrete symbols, strokes are already segmented by users when each symbol is written in a box. Spaced discrete symbols require symbol segmentation. In run-on discretely symbols, strokes can touch or overlap one another. Therefore, symbols cannot be easily separated one from another. Pure cursive handwriting requires advanced stroke segmentation. Since several symbols can be made one stroke (Tappert, Suen & Wakahara, 1990). The scope of this study, spaced discrete symbols are selected so as to narrow down the problem.

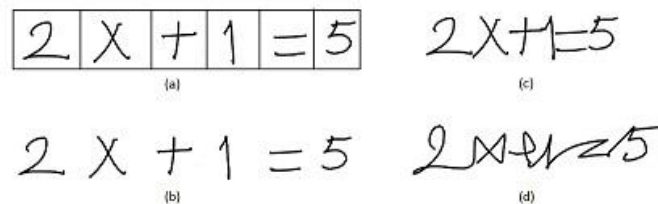


Figure 1.1 Types of handwriting styles: (a) box discrete symbols (b) spaced discrete symbols (c) run-on discretely symbols (d) pure cursive handwriting.

Handwritten recognition methods can be divided into two groups: Offline and online methods (Plamondon & Srihari, 2000). Offline methods convert handwriting

on paper to digital data, and then recognize the digital data. Online methods provide a means for the recognition of handwriting as the user performs a writing.

An advantage of online methods is that they obtain the temporal or dynamic information of the writing. This information consists of the number of strokes, the order of the strokes, the direction of the writing for each stroke, and the speed of the writing within each stroke (Tappert et al., 1990). Also, online methods allow users to instantly correct errors.

However, offline methods cannot obtain any temporal or dynamic information of the writing. And in some cases, handwriting on paper cannot be clear and readable. But offline methods do not need any special device and are used for all existing documents.

There have been many studies on the recognition of online mathematical expressions. Although there are many different methods for the online expression recognition, generally similar procedures are followed in such systems. These can be summarized in 4 steps: Preprocessing, partitioning, feature extraction and recognition.

Preprocessing generally consists of reduction of noise caused by hardware such as smoothing, thinning, etc. and normalization caused by the user such as baseline drift correction, size normalization, stroke length normalization, etc. (Tappert et al., 1990). In the partitioning step, expression is separated into symbols. There exist many methods for partitioning. One of the methods is that user indicates to finish a symbol writing by means of a sign. In some other methods, spaces between symbols and time information are used for partitioning. Another method requires symbols to be written into separate boxes by the user. Hence, partitioning problem is automatically solved. In feature extraction, a data space which only consists of key information of symbols is generated. In recognition, many algorithms are used for recognition of symbols such as artificial neural networks, k-nearest neighbors etc. (LeCun et al., 1995)

In this study, an online handwritten mathematical expression recognition (OHMER) system is developed for touchscreen devices. It supports the recognition of online handwritten mathematical expression and solution of the equations and conversion to a standard text expression.

OHMER system is composed of five modules: symbol detection, symbol recognition, parsing, exporting and solving.

The key components of OHMER system are symbols. A symbol is a sequence of strokes. A stroke is defined as a sequence of points. Symbol detection module determines which strokes are grouped into a symbol.

Symbol recognition module uses symbols obtained from symbol detection module. First, orientation and ordering of symbol strokes are standardized. For scale and translation invariance, size and coordinates are normalized. Then, symbol is resampled and sent to feature extraction. In this submodule, features of symbol are extracted. Features are x coordinates and y coordinates and rate change. Extracted features are sent to recognition. K-nearest neighbors algorithm is used to find the optimum symbol among the symbol set.

After all symbols are recognized, the input expression is ready in order to be analyzed relationship between parts of the expression. For this, predefined rules are used. Predefined rules are set of rules and describing how to parse the expression. According to these rules, input expression is parsed in the parsing module.

Then, expression tree which is generated in parsing module is converted text expressions such as AsciiMathML, MathML (Gray, 2007).

Finally, text expressions are sent to solving module so as to determine types of equations and solving process is performed.

## CHAPTER TWO

### ARCHITECTURE

Block diagram of OHMER system can be seen in Figure 2.1. Main modules of the system are listed below:

- Symbol Detection
- Symbol Recognition
- Parsing
- Exporting
- Solving

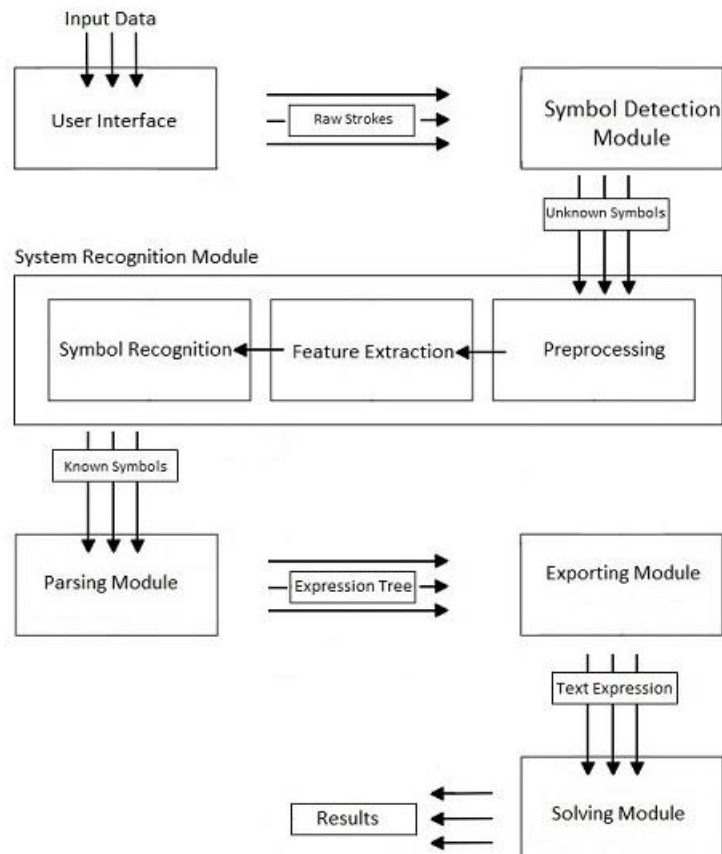


Figure 2.1 Block diagram of the online handwritten mathematical expression recognition system.

## 2.1 Symbol Detection

Purpose of the symbol detection module is to find unknown symbols which are not classified yet, from raw strokes. The system use minimum spanning tree approach (Matsakis, 1990) in order to combine strokes into a symbol.

First, connected and undirected a graph of strokes is constructed. In order to adapt strokes to the graph basic attributes of stroke are considered. These attributes are bounding box and location. Bounding box is the area defined by the minimum and maximum x, y coordinates of a stroke. Location holds the coordinates of the center of the bounding box.

Vertex of the connected and undirected graph corresponds to stroke location and edge weight is determined as the Euclidean distance between two strokes.

For each edge  $(u, v) \in G$ , set

$$weight(u, v) = dist \parallel u - v \parallel \quad (2.1)$$

where:

$u, v$ : Vertices of edge. They correspond to location of strokes.

$G$ : Graph of strokes

The system finds a minimum spanning tree from the constructed graph using Kruskal's algorithm. Kruskal's algorithm finds a minimum spanning tree for a connected weighted graph, i.e., it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized (Jayalakshmi, 2007).

Locations, bounding boxes and minimum spanning tree of strokes are shown in Figure 2.2.



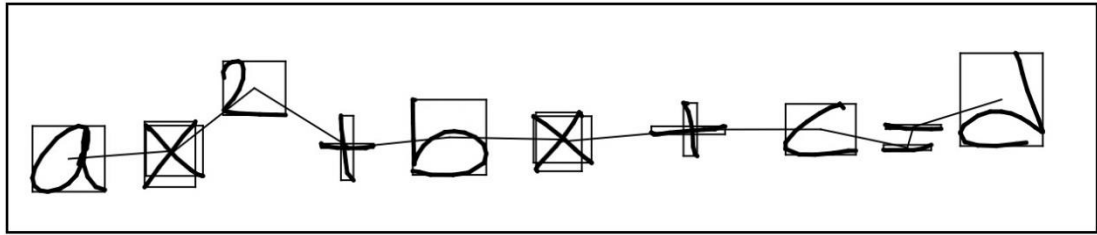


Figure 2.2 Minimum spanning tree of strokes.

After finding minimum spanning tree (MST), all bounding boxes of strokes which correspond to vertices of MST, are examined. If extended bounding boxes intersect each other, strokes corresponding to the intersected bounding boxes are grouped into a symbol.

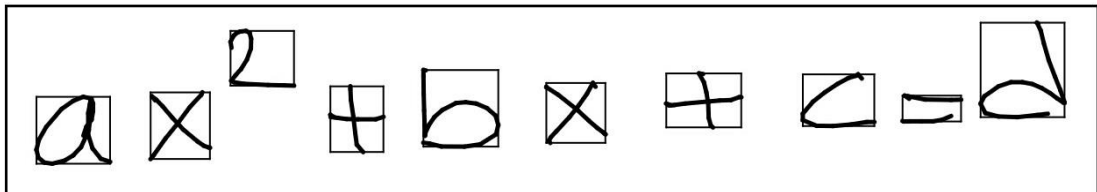


Figure 2.3 Found symbols are shown in its bounding box.

## 2.2 Symbol Recognition

In this module, the unknown symbols are recognized using their extracted features. Submodules of the symbol recognition are listed below:

- Preprocessing
- Feature Extraction
- Recognition

### 2.2.1 Preprocessing

Strokes of unknown symbols have to be standardized in order to increase accuracy of the recognition. Goal of this submodule is to remove unnecessary data and generate standard data space. The standard data space is one dimensional vector consists of stroke points and change rates.

One of the most important tasks in preprocessing is to perform transformation of strokes of the unknown symbols. There are four types of transformation: orientation, ordering, shifting and scaling.

In orientation process, strokes are put in standard directions. Then, strokes of the symbols are put into a standard ordering according to angles of strokes in ordering process (Matsakis, 1990).

In shifting process,  $(x, y)$  points of strokes are translated by amount  $dx$  and  $dy$ . Amount  $dx$  and  $dy$  are distance between top-left corner of symbol bounding box and origin.

For each  $(x, y)$  point  $\in$  stroke, set

$$\begin{aligned}x' &= x + dx \\ y' &= y + dy\end{aligned}\tag{2.2}$$

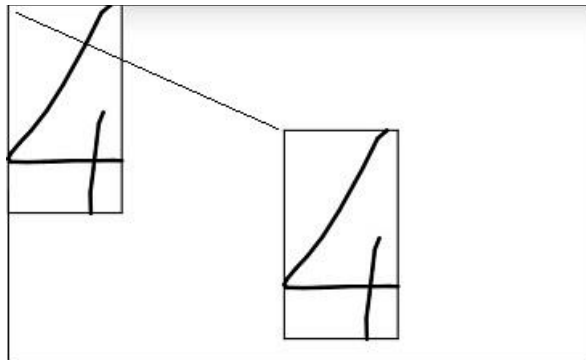


Figure 2.4 Shifting process of symbol 4.

In scaling process, symbol bounding box is scaled to unit bounding box. For this, (x, y) points of strokes are scaled by  $s_x$  and  $s_y$ .  $s_x$  and  $s_y$  are ratio of size bounding box and unit bounding box.

For each (x, y) point  $\in$  stroke, set

$$\begin{aligned} x' &= x * s_x \\ y' &= y * s_y \end{aligned} \quad (2.3)$$

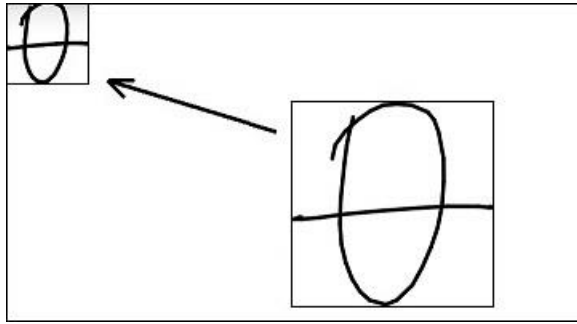


Figure 2.5 Shifting and scaling process of symbol 4.

Another important task in preprocessing submodule is to resample stroke points using linear interpolation. Hence, numbers of points in all symbols are equalized.

For point (x1, y1) and point (x2, y2)  $\in$  stroke, set the interpolant point (x, y)

$$\frac{y-y_1}{x-x_1} = \frac{y_2-y_1}{x_2-x_1} \quad (2.4)$$

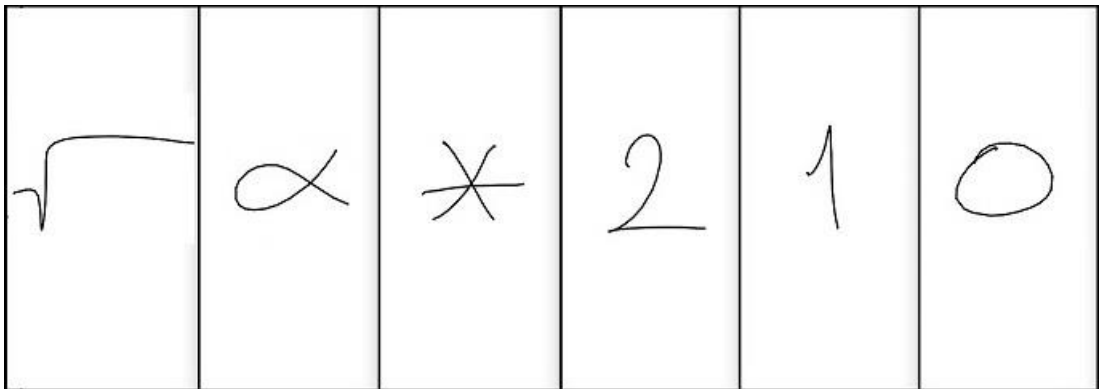


Figure 2.6 Some original symbols which is inputted by the user.

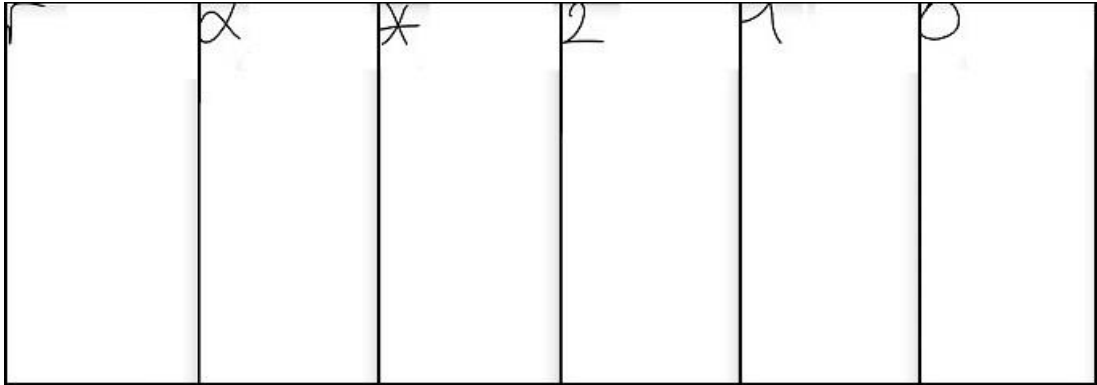


Figure 2.7 Normalized and resampled symbols.

### 2.2.2 Feature Extraction

Unknown symbols obtained via preprocessing module are used for feature extraction. Symbol features are extracted from the preprocessed unknown symbols. These features are obtained using stroke descriptors. Stroke descriptors are x, y coordinates and rate of change of coordinates.

x, y coordinates: x and y coordinates of strokes of symbols

Rate of change of coordinates: Second derivative of x and y coordinates. This feature is important for curved symbols.

Table 2.1 Symbol features obtained from the feature extraction submodule.

Feature	Description
x-coordinates	x-coordinates of strokes of symbol
y-coordinates	y-coordinates of strokes of symbol
change rate of x-coordinates	second derivation of x-coordinates
change rate of y-coordinates	second derivation of y-coordinates

### **2.2.3 Recognition**

In recognition submodule, k-nearest neighbors (KNN) and artificial neural networks algorithms are used for symbol classification.

First, k-nearest neighbor algorithm is implemented. The k-nearest neighbors (KNN) algorithm is a method to classify objects based on nearest training examples in the feature space. KNN is one of the simplest machine-learning algorithms. An object classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors. In order to identify neighbors, the objects are represented by position vectors in the multi-dimensional feature space. Usually the Euclidean distance is adopted, though other distance measures, such as the Manhattan distance, could in principle be used instead. (Yu, Lu, Lou, & Wang, 2010)

A basic example is given in Figure 2.8. A training set consists of two classes with five instances a piece, as indicated by of blue and red dots. And feature space is two-dimensional. Class can be assigned by a majority vote of the k nearest neighbors. The test sample (green dot) should be classified either to the first class of blue dots or to the second class of red dots. If  $k = 3$  (solid line circle) it is assigned to the second class because there are 2 red dots and only 1 blue dot inside the inner circle. If  $k = 5$  (dashed line circle) it is assigned to the first class (3 blue dots vs. 2 red dots inside the outer circle).

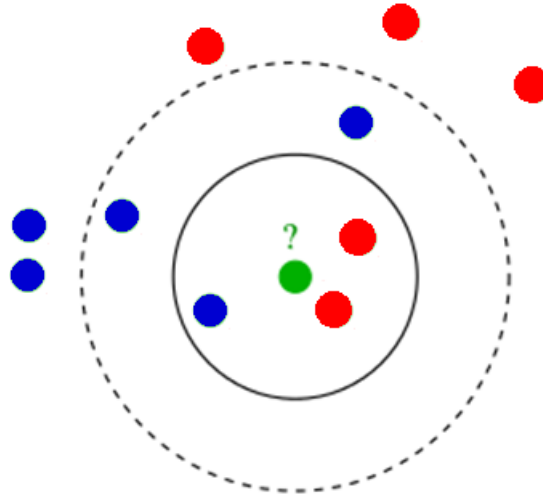


Figure 2.8 K-nearest neighbors algorithm.

Next, feed-forward artificial neural network is implemented. Feed-forward artificial neural network is a structured hierarchical layered network. Neurons are arranged in a layered configuration containing an input layer, one or two hidden layers, and an output layer. And the nodes in one layer are connected only to the nodes in the next layer (Dougherty, 2013).

Each of neurons has several input links (it takes the output values from several neurons in the previous layer as input) and several output links (it passes the response to several neurons in the next layer). A single model of a neuron is given in Figure 2.9.

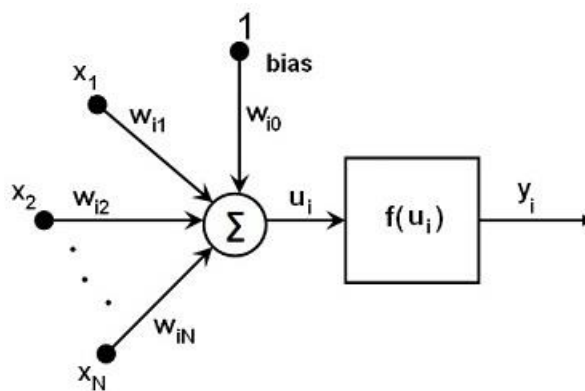


Figure 2.9 A single model of a neuron.

The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term.

$$\begin{aligned} u_i &= \sum_j (w_{ij}^{n+1} * x_j) + x_{i,bias}^{n+1} \\ y_i &= f(u_i) \end{aligned} \quad (2.5)$$

The sum is transformed using the activation function  $f$  which is chosen sigmoid function for all neurons.

$$f(x) = (1 - e^{-x}) / (1 + e^{-x}) \quad (2.6)$$

The neural network training algorithm (Neural Networks, n.d.) is as follows:

1. Take the feature vector as input. The vector size is equal to the size of the input layer.
2. Pass values as input to the first hidden layer.
3. Compute outputs of the hidden layer using the weights and the activation functions.
4. Pass outputs further downstream until you compute the output layer.

The weights are computed by the training algorithm. The algorithm takes multiple input vectors with the corresponding output vectors, and iteratively adjusts the weights to enable the network to give the desired response to the provided input.

Finally, classification results are compared between k-nearest neighbors and artificial neural network algorithms. Both algorithms are tested on the same dataset. Dataset has 28 symbols and each symbol has 200 samples. %75 of dataset is used for training and %25 for test. Accuracy of k-nearest neighbors algorithm is 96.71 percent for  $k=2$ , while accuracy of neural network algorithm is 94.42 percent for 36 hidden layers. Both algorithms had shown similar performance regarding accuracy rate. But k-nearest neighbors is preferred because of simplicity.

## 2.3 Parsing

This module aims to build an expression tree from known symbols of the given mathematical expression. After the symbol recognition, the known symbols are sent to parsing module to obtain hierarchical structures of the expression.

All symbols have a bounding box. Bounding box is defined as the area enclosed by the minimum and maximum x, y coordinates. Symbols of the expression are analyzed as structural according to a set of rules. These rules are geometrical relationship between bounding box of symbols and describing how to parse the expression.

The geometrical relationships are used to determine the rules. The rules can be subexpressions, superscripts, horizontal pair, or etc. Some rules are given in Figure 2.10.

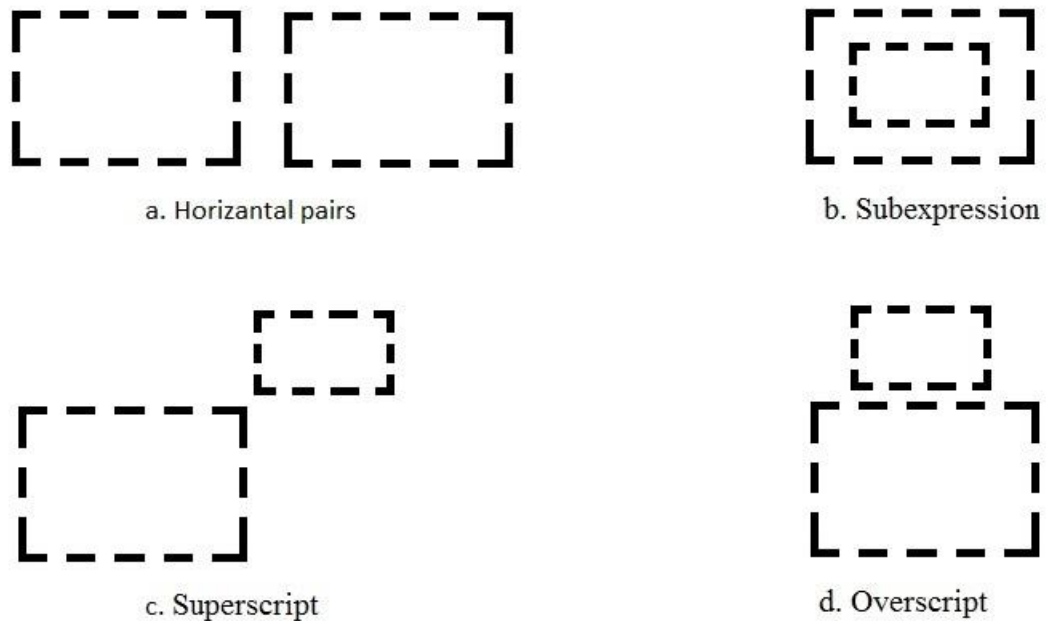


Figure 2.10 Some rules: a. Horizontal pairs b. Subexpression c. Superscript d. Over script.

OHMER system currently supports the horizontal pairs, subexpression and superscripts rules. Further rules could be employed in a similar fashion.



The system takes into account symbol location and baseline. Symbol location is expressed in terms of the coordinates of the center of the bounding box, whereas baseline represents the horizontal line of symbol. Examples of symbol bounding box and baseline are demonstrated in Figure 2.11. The system parses the expression and constructs an expression tree using these two attribute together with the supported rules. The expression tree is a tree of symbols that represents hierarchical structure of the expression.

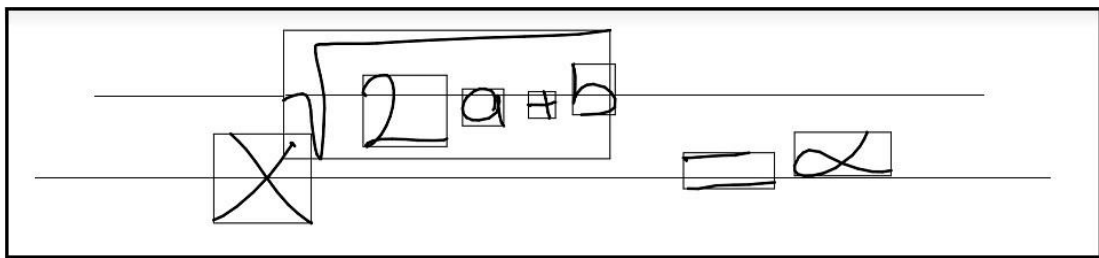


Figure 2.11 Bounding boxes of symbols and baselines belong to  $x$  and  $\sqrt{}$ .

The expression tree has three main parts: a root node, branches and nodes. The root node is the starting point of the tree. Both root node and other nodes contain symbols. Branches correspond to rules and each node has branches as many as the number of the rules.

The expression tree construction algorithm is given as follows:

Leftmost symbol of the expression is assigned to root node. The root node is then set as the current node.

Repeat until all symbols of the expression are assigned to other nodes:

1. A symbol list is populated according to subexpression rule of current node. If subexpression branch of current node is null and the symbol list is not empty, leftmost symbol of the symbol list is assigned to new node. And new node is added to subexpression branch of current node. The new node is then set as the current node. And go to loop condition.

2. A symbol list is populated according to superscript rule of current node. If superscript branch of current node is null and the symbol list is not empty, leftmost symbol of the symbol list is assigned to new node. And new node is added to superscript branch of current node. The new node is then set as the current node. And go to loop condition.
3. A symbol list is populated according to horizontal pairs rule of current node. If horizontal pairs branch of current node is null and the symbol list is not empty, leftmost symbol of the symbol list is assigned to new node. And new node is added to horizontal pairs branch of current node. The new node is then set as the current node. And go to loop condition.
4. If parent of current node is not null, the parent is then set as the current node. And go to loop condition.

Expression tree of the mathematical expression given in Figure 2.11 is demonstrated in Figure 2.12.

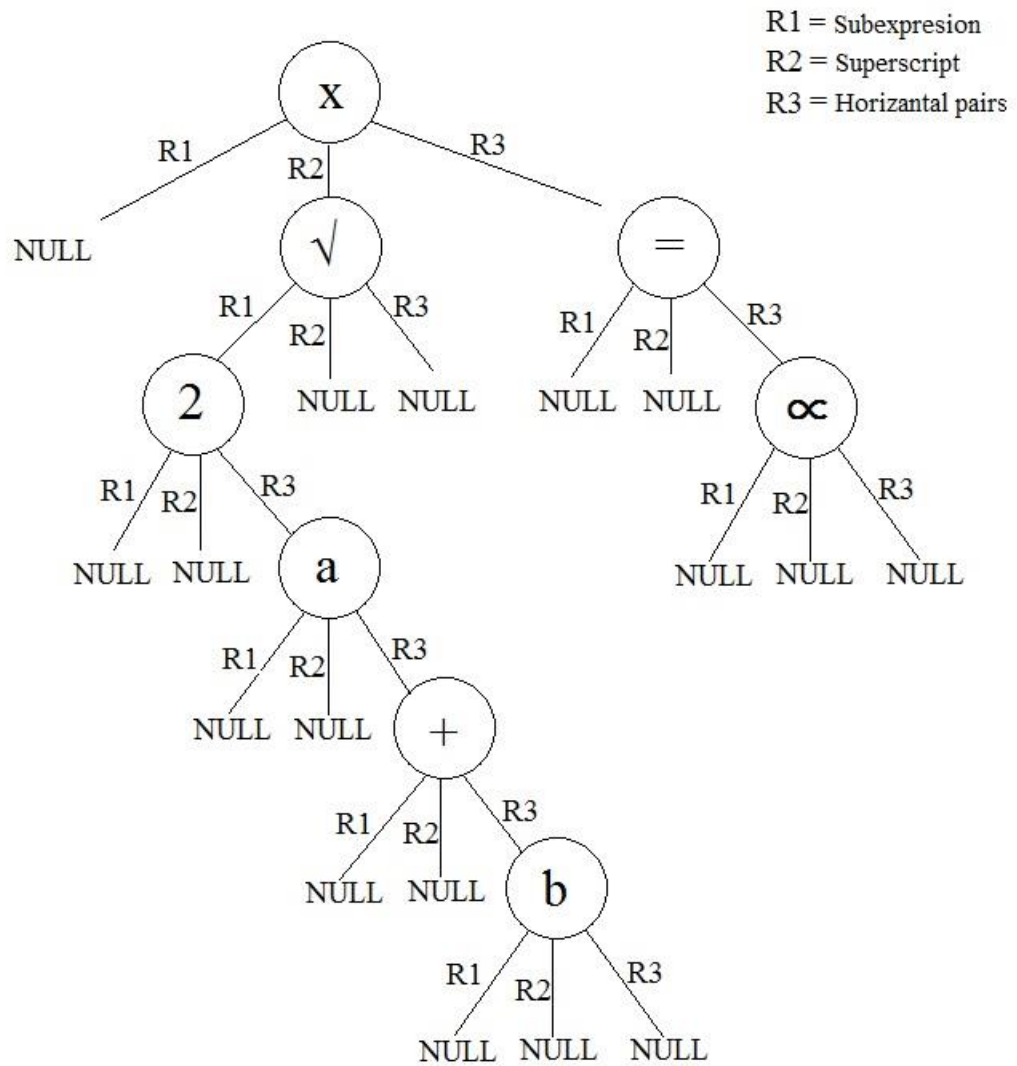


Figure 2.12 Expression tree of the mathematical expression given in Figure 2.11

## 2.4 Exporting

In exporting module, an equivalent text expression is generated according to the equation tree.

AsciiMathML is a linear mathematical markup language for displaying mathematical expression in web pages (Gray, 2007). It is generated to visit each node in the expression tree according to pre-order type of depth-first traversal.

Pre-order visiting steps:

1. Traverse the data part of root element (or current element)
2. Traverse the left subtree by recursively calling the pre-order function.
3. Traverse the middle subtree by recursively calling the pre-order function.
4. Traverse the right subtree by recursively calling the pre-order function.

Pre-order of the expression tree in Figure 2.12:  $x, \sqrt{\phantom{x}}, 2, a, +, b, =, \propto$

AsciiMathML output for the expression in Figure 2.11:

$$x^{(\text{sqrt}(2a + b))} = \propto \quad (2.7)$$

AsciiMathML output for the expression in Figure 2.3:

$$ax^2 + bx + c = d \quad (2.8)$$

MathML, a mathematical markup language, is generated to parsing the AsciiMathML expression. It is used for integrating mathematical expression into web pages (Wright, 2000).

MathML uses tags to describe the various parts of a mathematical expression. Variables, operators, radicals, and fractions are the fundamental elements of MathML.

Table 2.2 Supported MathML tag elements by the current system.

Tags	Description
<code>&lt;mi&gt;x&lt;/mi&gt;</code>	Tag elements for variables.
<code>&lt;mn&gt;2&lt;/mn&gt;</code>	Tag elements for numbers.
<code>&lt;mo&gt;+&lt;/mo&gt;</code>	Tag elements for operators.
<code>&lt;mrow&gt; &lt;/mrow&gt;</code>	Tag elements for group the expression.

Table 2.2 Supported MathML tag elements by the current system. (cont.)

<code>&lt;msup&gt; &lt;/msup&gt;</code>	Tag elements for create superscript.
<code>&lt;msqrt&gt; &lt;/msqrt&gt;</code>	Tag elements for create square root.

MathML output for the expression in Figure 2.11 can be seen in Figure 2.13.

```
<math xmlns='http://www.w3.org/1998/Math/MathML' display='block'>
  <msup>
    <mi>x</mi>
    <mrow>
      <msqrt>
        <mn>2</mn>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </msqrt>
    </mrow>
  </msup>
  <mo>=</mo>
  <mi>?</mi>
</math>
```

Figure 2.13 MathML output for the expression in Figure 2.11.

## 2.6 Solving

Generated text expression is sent to solving module so as to perform the following three types of calculations:

- Solve calculations with no unknown.
- High degree equations.
- Simultaneous linear equations.

If there happens to be no unknowns in the text expression, such as  $2+3$ ,  $5*2+4/3=?$ ,  $(2+3)*5$ , etc., First infix expression which is human readable is converted to postfix notation. The advantage of postfix notation is to be written expressions without needing for parenthesis and it is easier to evaluate in a computer. For instance, the expression  $(2+3)*4$  in postfix notation would be  $23+4*$ .

In order to convert infix to postfix expression, we need a stack for operators and a string to store postfix expression. Pseudo code for infix to postfix expression conversion algorithm is given in Figure 2.14.

Then postfix expression is scanned from left to right for evaluating. Each time a number is found, it is pushed to stack. If an operator is found, two numbers are popped from stack and a calculation is performed according to the operator and the result is pushed to stack. At the end, only the result of the expression remains on the stack.

```

FOR each item in the infix expression
  IF item is number THEN
    Add item to postfix string
  ELSE IF item is operator
    IF item is leftparanthes THEN
      Push item to stack
    ELSE IF item is rightparanthes THEN
      Pop temp_item from stack
      WHILE temp_item is not leftparanthes
        Add temp_item to postfix string
        Pop temp_item from stack
      ENDWHILE
    ELSE
      WHILE stack is not empty AND stack.peek is higher precedence then item THEN
        IF stack.peek is not leftparanthes AND stack.peek is not rightparanthes
          Pop temp_item from stack
          Add temp_item to postfix string
        END IF
      ENDWHILE
      Push item to stack
    END IF
  END IF
END FOR
WHILE stack is not empty
  Pop temp_item from stack
  IF temp_item is not leftparanthes AND temp_item is not rightparanthes
    Add temp_item to postfix string
  END IF
ENDWHILE

```

Figure 2.14 Pseudo code for infix to postfix expression conversion algorithm

If text expression contains second or more degree equation with one unknown such as  $x^2+4x=?$ ,  $4x^2+3x+5=2x^2+x+7$ , etc., the expression is considered as a polynomial equation. First, polynomial equations of form  $P(x) = Q(x)$  are converted to  $P(x) = 0$ , where  $P(x)$  and  $Q(x)$  are polynomials. Then  $P(x) = 0$  equation is solved. Special cases of such polynomial equations are quadratic, cubic, quartic, etc. Current

version of OHMER supports only quadratic equations. The system solves quadratic equations using quadratic formula.

$$P(x) = a_2x^2 + a_1x + a_0 = 0$$

$$x_{1,2} = \frac{-a_1 \mp \sqrt{a_1^2 - 4a_2a_0}}{2a_2} \quad (2.9)$$

If text expression contains first degree equations with one or more unknown, the system solves simultaneous linear equations using Cramer's rule.

In general, we may write simultaneous linear equations in the form:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= c_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= c_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= c_3 \\ &\vdots \\ &\vdots \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= c_n \end{aligned} \quad (2.10)$$

which in vector notation is

$$A \vec{x} = \vec{c} \quad (2.11)$$

where

x: A n-dimensional vector the elements of which represent the solution of equations.

c: The constant vector of the system of equations.

A: The matrix of the system's coefficients.

We can write the solution to these equations as

$$\vec{x} = A^{-1} \vec{c} \quad (2.12)$$

where

$A^{-1}$ : Inverse of the coefficient matrix.

Cramer's rule is probably the best known method that to find the solution without finding the inverse of the matrix (Collins, 2003). The system finds determinant of the matrix A so as to use this method.

Determinant of the matrix A:

$$DetA = \sum_{i=1}^n (-1)^{(i+j)} a_{ij} M_{ij}, \forall j \quad (2.13)$$

where

$M_{ij}$ : The determinant of the matrix A with the  $i$ th row and  $j$ th column removed.

General solution of equation is given by:

$$x_j = \frac{\begin{vmatrix} a_{11} & \dots & a_{1j-1} & c_1 & a_{1j+1} & \dots & a_{1n} \\ a_{21} & \dots & a_{2j-1} & c_2 & a_{2j+1} & \dots & a_{2n} \\ \dots & & & & & & \\ a_{n1} & \dots & a_{nj-1} & c_n & a_{nj+1} & \dots & a_{nn} \end{vmatrix}}{DetA} \quad (2.14)$$

This solution requires evaluating the determinant of the matrix A as well as an augmented matrix where the  $j$ th column has been replaced by the elements of the constant vector.



## CHAPTER THREE

### EVALUATION

#### 3.1 Data Collection

Data Collector application is developed for rapid data collection. The data were collected in set of 200 samples for each symbol using this application. The flow of data collection steps are illustrated Figure 3.1.

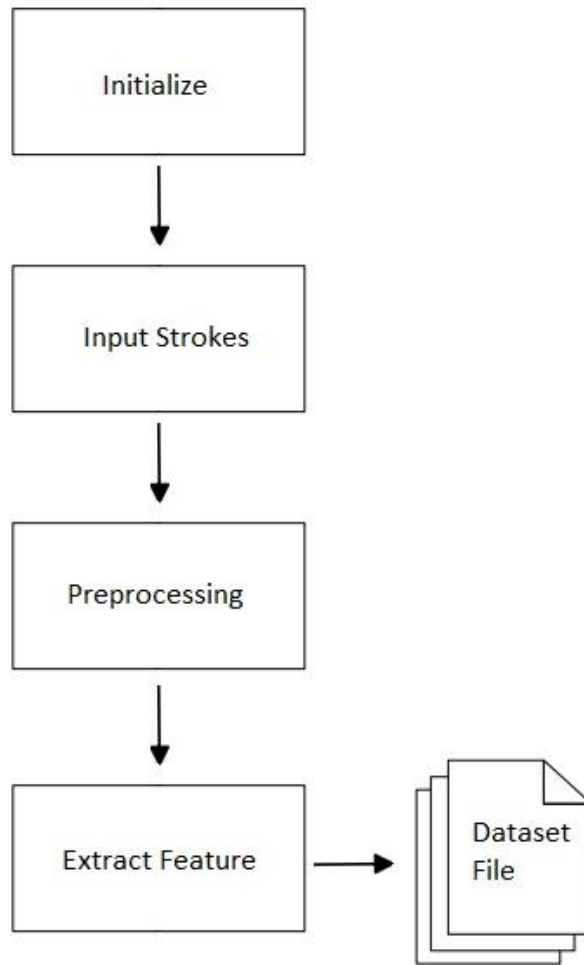


Figure 3.1 The flow of data collection steps.

First, the system is initialized, and allows users to input strokes. Input strokes are preprocessed and features are extracted. Then features are stored in dataset file. The dataset file is text file that contains a list of features.

### 3.2 Training and Recognition

The flow of training steps are illustrated Figure 3.2.

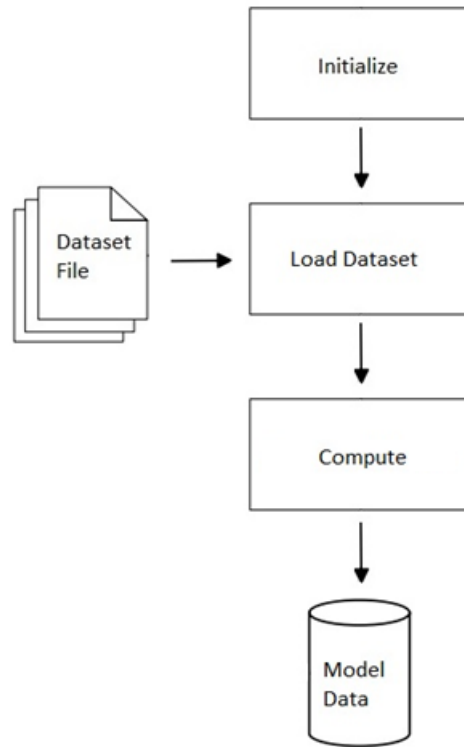


Figure 3.2 The flow of training steps.

The dataset file is loaded after the system is initialized. Then model data is generated from dataset which is computed according to k-nearest neighbors algorithm.

The flow of recognition steps are illustrated Figure 3.3.

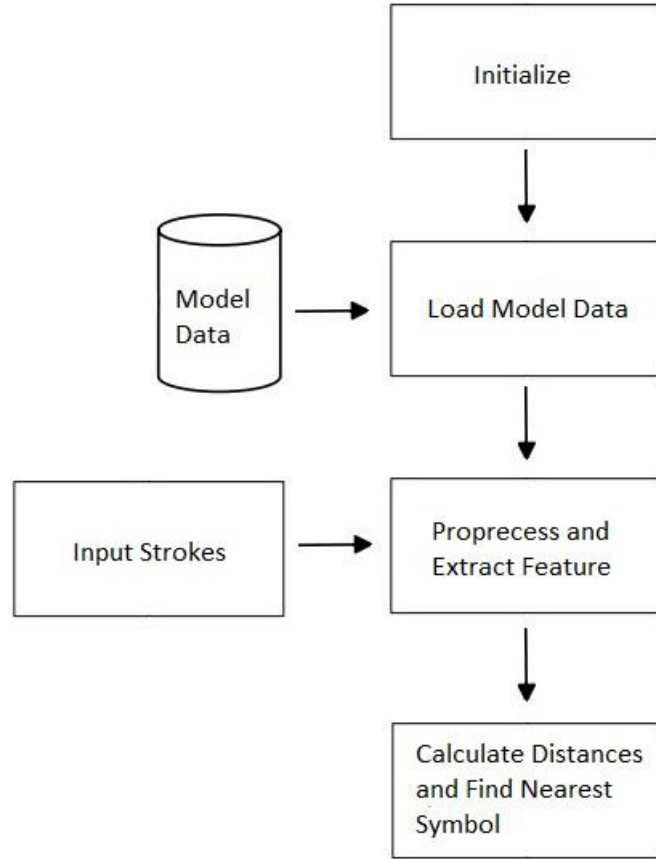


Figure 3.3 The flow of recognition steps.

The model data file is loaded after the system is initialized. Interface of the system allows users to input strokes. Input strokes are preprocessed and features are extracted. Distances are calculated. Then, nearest symbol is found.

### 3.3 Data Collector Application

Data Collector android application is developed. It provides a graphical user interface for collecting data, and it is used for rapid data collection. A screenshot of the application is shown in Figure 3.4.

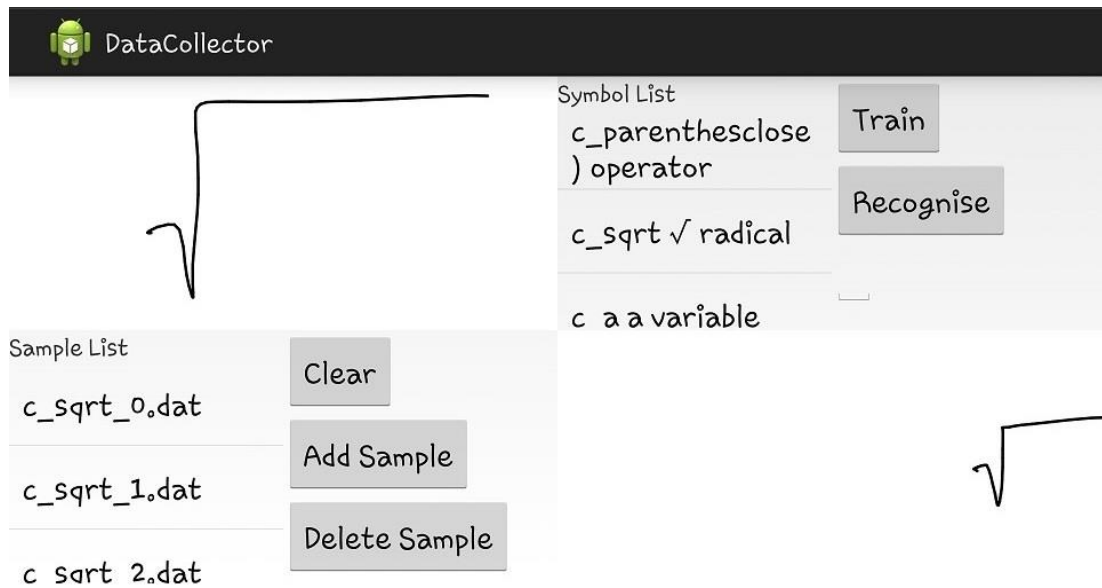


Figure 3.4 Screenshot of the Data Collector application.

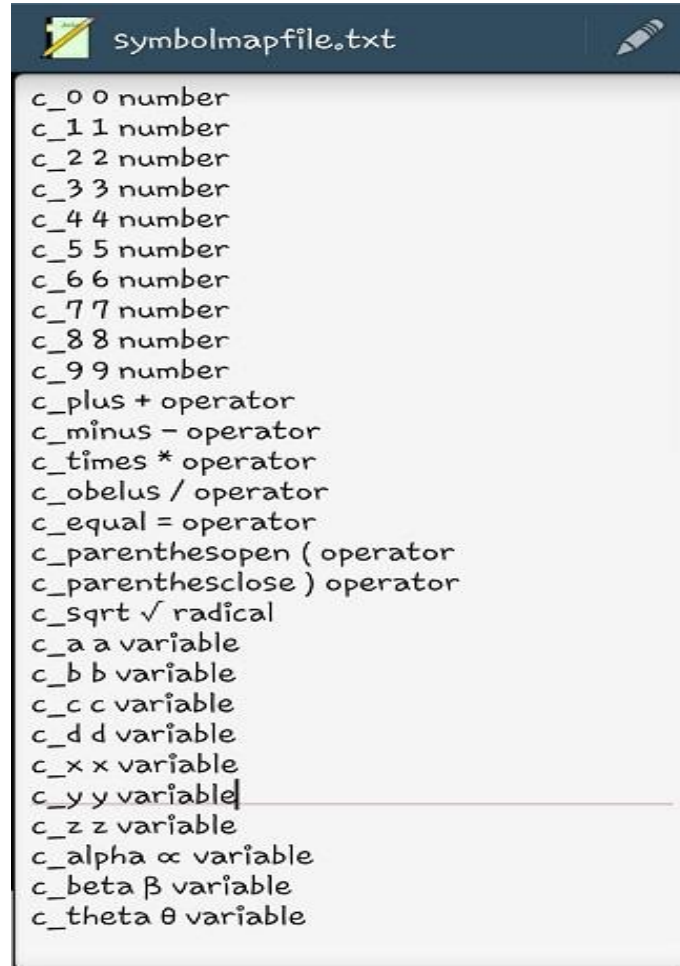
The application allows the user to draw samples of each symbol using stylus or finger. It creates dataset file and builds model data using k-nearest neighbors algorithm. It also supports to recognize a symbol according to model data for testing.

The user interface of application contains the components given in Table 3.1.

Table 3.1 Descriptions for basic components of Data Collector application.

Component	Description
Draw Area	Allows the user to draw symbols for data collection.
Display Area	Displays a sample which is selected.
Symbol List	List of the name of symbols.
Sample List	List of file names of collected sample.
Clear	Clear the draw area.
Add Sample	Create a sample file for drawing symbol.
Delete Sample	Delete a sample file which is selected.
Train	Allows the user to generate dataset and model data from raw data.
Recognize	Allows the user to test model data.

The application uses symbol map file. This file stores all symbols' information. New symbols can be added to the system or existing symbols can be updated by editing the symbol map file. An example of a symbol map file is given in Figure 3.5



```
c_0 0 number
c_1 1 number
c_2 2 number
c_3 3 number
c_4 4 number
c_5 5 number
c_6 6 number
c_7 7 number
c_8 8 number
c_9 9 number
c_plus + operator
c_minus - operator
c_times * operator
c_obelus / operator
c_equal = operator
c_parenthesopen ( operator
c_parenthesclose ) operator
c_sqrt √ radical
c_a a variable
c_b b variable
c_c c variable
c_d d variable
c_x x variable
c_y y variable
c_z z variable
c_alpha α variable
c_beta β variable
c_theta θ variable
```

Figure 3.5 An example of a symbol map file. A row of file consists of symbol name, symbol and type.

### 3.4 Math Solver Application

Math Solver android application is developed. It provides a graphical user interface for online handwritten mathematical expression recognition system. It supports Turkish and English language and multiline for entering multiple equations. OpenCV Machine Learning Library (MLL) is used for the k-nearest neighbors algorithm implementation in the application. The current version can recognize spaced discrete handwritten symbols. Besides, current version also supports

horizontal pairs, subexpression, superscript rules and performs calculations. Screenshots of the Math Solver are shown in Figure 3.6, 3.9 and 3.10.

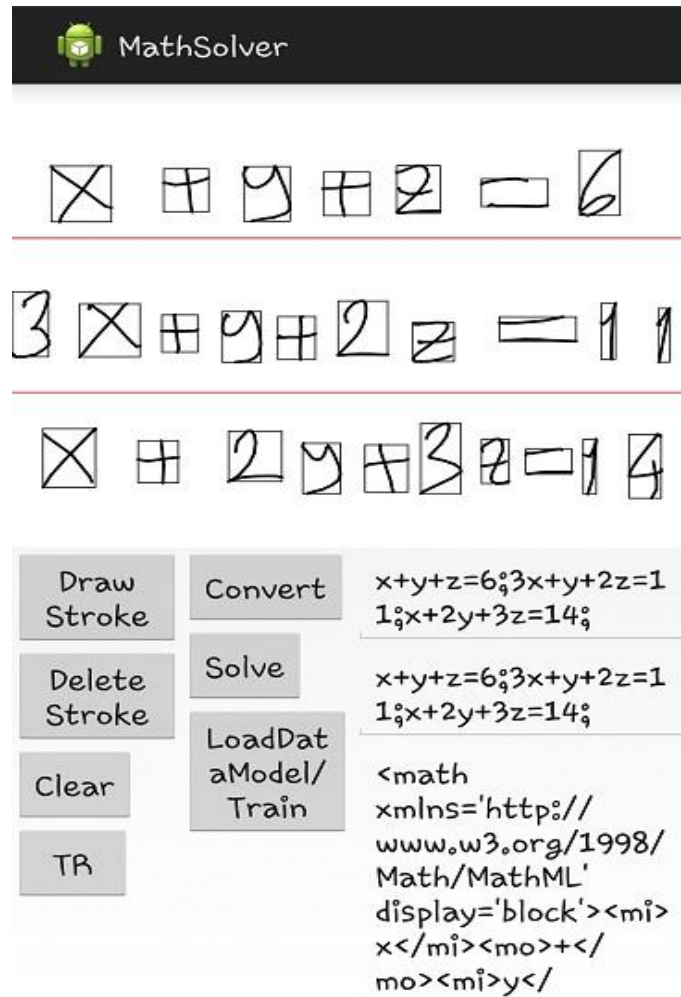


Figure 3.6 Screenshot of the Math Solver application for simultaneous linear equations.

Table 3.2 Descriptions for basic components of Math Solver application.

Component	Description
Draw Area	Allows the user to draw mathematical expressions.
Raw Data Field	Displays the raw text expressions.
AsciiMathML Field	Displays the AsciiMathML text expressions.
MathML Field	Displays the MathML text expression.

Table 3.2 Descriptions for basic components of Math Solver application. (cont.)

Draw Stroke	Enables pen mode which allows the user to draw strokes.
Delete Stroke	Enables eraser mode which allows the user to remove strokes.
Clear	Clear the drawing area.
TR/EN	Change language.
Convert	Converts input expressions which are in draw area to text expressions and creates convert file.
Solve	Solves input expressions which are converted to text and creates solve file.
Load DataModel/Train	Loads model data which is generated in recognition module.

The application produces a convert file. A convert file stores text expressions corresponding to the input expressions drawn in the application area. Convert file of the input expression given in Figure 3.6 is demonstrated in Figure 3.7.

The application also produces a solve file. Solve file stores results of converted input expressions. Solve file of the same expression is given in Figure 3.8.


 convert.txt  
CONVERT RESULTS  
Inputs:  
Raw  
 $x+y+z=6$ ; $3x+y+2z=11$ ; $x+2y+3z=14$ ;  
Ascii  
 $x+y+z=6$ ; $3x+y+2z=11$ ; $x+2y+3z=14$ ;  
MathML  
$$\begin{matrix} x \\ + \\ y \\ + \\ z \\ = \\ 6 \\ 3x \\ + \\ y \\ + \\ 2z \\ = \\ 11 \\ x \\ + \\ 2y \\ + \\ 3z \\ = \\ 14 \end{matrix}$$

Figure 3.7 Convert file contents.


 result.txt  
SOLVE RESULTS  
Inputs:  
 $x+y+z=6$   
 $3x+y+2z=11$   
 $x+2y+3z=14$   
Outputs:  
 $z=3.0$   
 $y=2.0$   
 $x=1.0$   
|

Figure 3.8 Solve file contents.



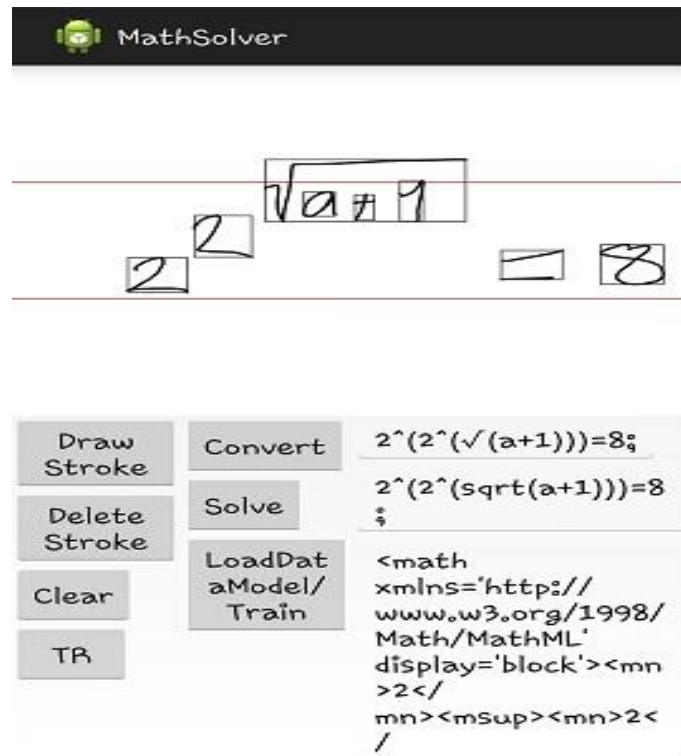


Figure 3.9 Screenshot of Math Solver application for complex expression which consists of horizontal pairs, superscript and subexpression rules.

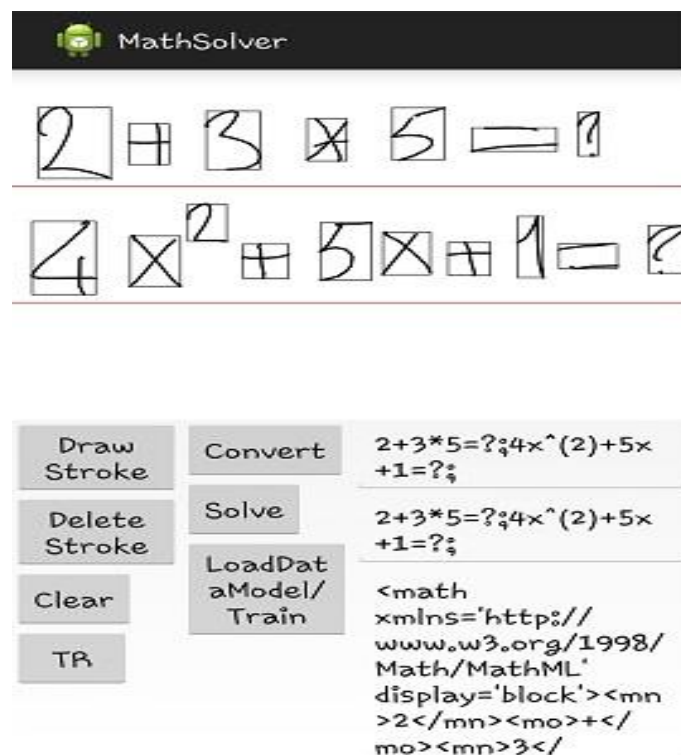


Figure 3.10 Screenshot of the Math solver application for calculation with no unknown and high degree equation.

## CHAPTER FOUR

### RESULTS

K-nearest neighbors and neural networks algorithms are used in the system. Results of both algorithms are compared in order to determine the classification algorithm. Both algorithms are tested on the same dataset. Dataset has 28 symbols and each symbol has 200 samples. %75 of dataset is set for training and %25 for testing. Accuracy of k-nearest neighbors algorithm is 96.71 percent, while accuracy of neural networks algorithm is 94.42 percent. All symbols except the ones that are similar in shape are recognized with high accuracy. Recognition results of k-nearest neighbors and neural networks are shown in the following tables.

Table 4.1 Recognition results of k-nearest neighbors for digits.

Symbol Class	Result				Accuracy Rate
	True	False	False	False	
0	0				%100
	50				
1	1	0	2	y	%86
	43	5	1	1	
2	2				%100
	50				
3	3				%100
	50				
4	4				%100
	50				
5	5				%100
	50				
6	6	4	8		%88
	44	1	5		
7	7				%100
	50				
8	8				%100
	50				
9	9	y			%96
	48	2			

Table 4.2 Recognition results of k-nearest neighbors for lower case letters (Latin alphabet).

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
a	a	0			%98
	49	2			
b	b				%100
	50				
c	c	0	(		%92
	46	1	3		
d	d				%100
	50				
x	x	$\alpha$	$\theta$	*	%94
	47	1	1	1	
y	y				%100
	50				
z	z				%100
	50				

Table 4.3 Recognition results of k-nearest neighbors for Greek symbols.

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
$\alpha$	$\alpha$				%100
	50				
$\beta$	$\beta$	5			%98
	49	1			
$\theta$	$\theta$				%100
	50				

Table 4.4 Recognition results of k-nearest neighbors for operators.

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
*	*				%100
	50				
/	/	2	-		%96
	48	1	1		

Table 4.4 Recognition results of k-nearest neighbors for operators. (cont.)

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
+	+	y			%98
	49	1			
-	-	0	/		%68
	34	1	15		
$\sqrt{\quad}$	$\sqrt{\quad}$				%100
	50				
(	(	0			%96
	48	2			
)	)				%100
	50				
=	=				%100
	50				

Table 4.5 Recognition results of neural networks for digits.

Symbol Class	Result				Accuracy Rate
	True	False	False	False	
0	0	2			%98
	49	1			
1	1	0	y		%78
	39	5	6		
2	2				%100
	50				
3	3				%100
	50				
4	4				%100
	50				
5	5				%100
	50				
6	6	0	8	$\sqrt{\quad}$	%90
	45	1	2	2	
7	7				%100
	50				

Table 4.5 Recognition results of neural networks for digits. (cont.)

Symbol Class	Result				Accuracy Rate
	True	False	False	False	
8	8	1	6		%96
	48	1	1		
9	9	3	y		%92
	46	1	3		

Table 4.6 Recognition results of neural networks for lower case letters (Latin alphabet).

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
a	a	0			%96
	48	2			
b	b				%100
	50				
c	c	(			%88
	44	6			
d	d				%100
	50				
x	x	1	$\beta$	$\theta$	%94
	47	1	1	1	
y	y	)			%98
	49	1			
z	z				%100
	50				

Table 4.7 Recognition results of neural networks for Greek symbols.

Symbol Class	Result				Accuracy Rate %
$\alpha$	$\alpha$				%100
	50				
$\beta$	$\beta$	z			%98
	49	1			
$\theta$	$\theta$				%100
	50				

Table 4.8 Recognition results of neural networks for operators.

Symbol Class	Result				Accuracy Rate %
	True	False	False	False	
*	*				%100
	50				
/	/	2	9		%96
	48	1	1		
+	+	z			%98
	49	1			
-	-	√	/	)	%26
	13	1	34	2	
√	√				%100
	50				
(	(	0			%96
	48	2			
)	)				%100
	50				
=	=				%100
	50				

According to recognition results, some symbols (such as -, /, c, ( ) which are very similar in handwriting expressions, are recognized with low accuracy rates. Those result rates can be increased by selecting better features and collecting more samples. In case of false symbol recognition, user interface of the system allows the user to correct errors by stroke deletion and re-drawing.

Accuracy rates of k-nearest neighbors algorithm are given in Table 4.9 for different k values.

Table 4.9 Accuracy rates of k-nearest neighbors algorithm for different k values.

k	Result		Accuracy Rate %
	True	False	
50	1270	130	%90.71
30	1306	94	%93.28
20	1319	81	%94.21
15	1331	69	%95.07

Table 4.9 Accuracy rates of k-nearest neighbors algorithm for different k values. (cont.)

<b>k</b>	<b>Result</b>		<b>Accuracy Rate %</b>
14	1329	71	%94.92
13	1333	67	%95.21
12	1331	69	%95.07
11	1335	65	%95.35
10	1334	66	%95.28
9	1340	60	%95.71
8	1340	60	%95.71
7	1344	56	%96.00
6	1350	50	%96.42
5	1350	50	%96.42
4	1351	49	%96.50
3	1351	49	%96.50
2	1354	46	%96.71
1	1353	47	%96.64

Accuracy rates of neural networks algorithm are given in Table 4.10 for different number of hidden layer neurons.

Table 4.10 Accuracy rates of neural networks algorithm for different number of hidden layer neurons.

<b>Number of hidden layer neurons</b>	<b>Result</b>		<b>Accuracy Rate %</b>
	<b>True</b>	<b>False</b>	
8	461	939	%31.92
10	482	918	%34.42
16	498	902	%35.57
20	527	873	%37.64
24	797	603	%56.92
32	937	463	%66.92
35	1282	118	%91.57
36	1322	78	%94.42
37	1144	256	%81.71
38	899	501	%64.21
40	934	466	%66.71
48	1000	400	%71.42
72	1046	354	%74.71
76	1197	203	%85.50
144	1305	95	%93.21

Current system successfully supports three mathematical rules: horizontal pairs, subexpression and superscript. Other rules could be easily incorporated in to the system in order to parse complex expressions such as fractions.

100 equations which consist of 30 calculations with no unknown, 30 high degree equations, 40 simultaneous linear equations, are entered in the system. The system successfully solves all equations that are successfully recognized.



## **CHAPTER FIVE**

### **CONCLUSION**

An online handwritten mathematical expression recognition system is developed within the scope of this study. Symbol detection, symbol recognition, parsing, exporting and solving techniques are introduced for the online recognition system.

The first module of the system architecture is symbol detection. Minimum spanning tree approach is applied to group raw strokes into symbols. Thus, raw strokes are converted to unknown symbols using distance between strokes as threshold.

The second module of the system architecture is symbol recognition which consists of three submodules. Unknown symbols are sent to preprocessing for standardization of strokes. For this, normalization processes is applied to strokes of unknown symbols such as stroke shifting, scaling, etc. Then standardized unknown symbols are sent the feature extraction. Here, some representative data about strokes are obtained. Finally, the obtained features are computed based on k-nearest neighbors algorithm and nearest symbols are found in recognition.

The third module of the system architecture is parsing. The aim of this module is to construct an expression tree according to predefined rules which determine geometrical relationships of symbols.

The fourth module of the system architecture is exporting. Math text expressions are generated according to the expression tree structure in this module.

The last module of the system architecture is solving. This module evaluates math text expressions and performs calculations.

Data Collector and Math Solver applications are developed on Android operating system in order to implement the overall system. Math Solver application allows the

user to divide draw area into lines so as to enter system of linear equations. It also supports Turkish and English languages.

The applications are tested on Samsung N9000 smartphone. The smartphone configuration is 1.9 ghz quad core processor, 3 gb ram and 5.7" 1920 x 1080 pixels display with 386 ppi resolution and s-pen stylus. It is well enough for applications of online mathematical expression recognition system.

All symbols except the ones that are similar in shape are recognized with high accuracy. Overall accuracy of the system is 96.71 percent. The system could consider other information about symbols in order to increase the rate of recognition. This information could be the ratio of symbol size relative to each other and symbol position relative to baseline.

If smartphone and tablets are used more effectively in education, problems regarding mathematical expressions recognition will be made more attractive for researchers.

## REFERENCES

- Collins, G. W. (2003). *Fundamental numerical methods and data analysis*. Case Western Reserve University.
- Dougherty, G. (2013). *Pattern recognition and classification: An introduction*. New York: Springer Science Business Media
- Gray, J. (2007) ASCHMathML: Now everyone can type MathML. *MSOR Connections*, 7 (3), 26-30.
- Jayalakshmi, N. (2007). *Data structure using C++*. New Delhi: Firewall Media.
- LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., et al. (1995). *Comparison of learning algorithms for handwritten digit recognition*. Bell Laboratories, Holmdel, NJ 07733, USA.
- Les, Z., & Les, M. (2015). *Shape understanding system: Machine understanding and human understanding*. Switzerland: Springer International.
- Matsakis, N. (1999). *Recognition of handwritten mathematical expressions*. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA.
- Neural Networks* (n.d.). Retrieved May 17, 2015, from [http://docs.opencv.org/2.4.9/modules/ml/doc/neural\\_networks.html](http://docs.opencv.org/2.4.9/modules/ml/doc/neural_networks.html)
- Plamondon, R., & Srihari, S.N. (2000). Online and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (1), 63 – 84.

- Tapia, E., & Rojas, R. (2004). *Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance*. Freie Universität Berlin, Institut für Informatik.
- Tappert, C.C., Suen, C.Y., & Wakahara, T. (1990). The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 (8), 787 – 808.
- Wright, F. J. (2000). Interactive mathematics via the web using MathML. *SIGSAM Bulletin*, 34 (2), 49 – 57.
- Yu, F., Lu, Z., Lou, H., & Wang, P. (2010). *Three-dimensional model analysis and processing*. Hangzhou: Zhejiang University Press.