ISSN:0975-9646

Optimal Neural Network Based Classifier Using Optical Character Recognition Engine for Tamil Language

Dr.N Venkateswaran[1] Dr.PS Jagadeesh Kumar[2*]
*Corresponding author

Abstract-In this paper, a neural network based classifier using optical character recognition engine for Tamil language is proposed. At the first level, features derived at each sample point of the preprocessed character are used to construct a subspace using Optical Character Recognition (OCR) software. Recognition of the test sample is performed using a neural network based classifier. Based on the analysis of the proposed method, it was identified that Tamil character recognition was optimal and the implementation reduces the coding complexity to a greater extent. The proposed method can be used to recognize any language characters but in this paper only Tamil characters were tested for recognition. As a future enhancement, implementation of speech processing can be used to identify the classified characters to visually impaired persons for ease. The same technique can be used to resolve confusions between triplets and quadruples of similar characters. The same recognition could be carried out not only for individual characters but also for Tamil words.

General Terms
Optical Character Recognition, Neural Network, Tamil language

Keywords
Feature matrix, Character grid, Character Switching, Nearest neighbor classifier, Neural network based classifier

1. INTRODUCTION

Tamil is a popular classical language spoken by a significant population in south East Asian countries. There are 156 distinct symbols in Tamil. For the recognition of Tamil characters, earlier methods use class specific subspaces and employed elastic matching schemes as in hidden markov models for recognition of Tamil characters. In this paper, a neural network based classifier using optical character recognition engine is proposed. For the first level of classification, Optical character recognition [1] algorithm for the extraction of features from Tamil characters in a subspace was implemented. For the classification of a test character, we employ a neural network based classifier. It is an established fact that one way of assessing the performance of any given classifier depends on how well it can perform on an unknown test sample [2]. The nearest neighbor classifier in the first stage fails to capture finer nuances between certain structural shapes that form the basic cues in making certain characters distinct. To further improve the classification accuracy of the system, it becomes imperative to design a robust, neural network classification scheme to distinguish between visually similar misclassified characters.

2. NEURAL NETWORK

2.1 Basics of Neural Networks

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in Fig.1. Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs. Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The delta rule is often utilized by the most common class of ANNs called 'back propagation neural networks' (BPNNs).

Back propagation is an abbreviation for the backwards propagation of error. With the delta rule, as with other types of back propagation, 'learning' is a supervised process that occurs with each cycle or 'epoch' (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backwards error propagation of weight adjustments.

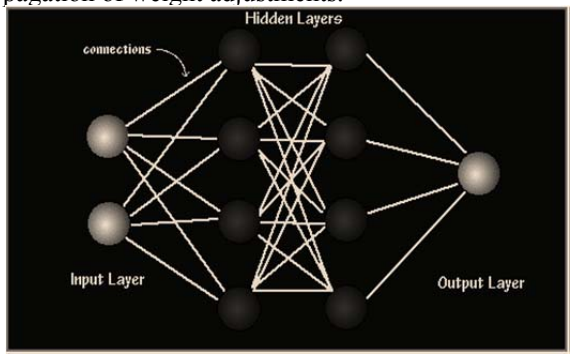

Fig 1: Layers in Neural Network

More simply, when a neural network is initially presented with a pattern it makes a random 'guess' as to what it might be. It then sees how far its answer was from the actual one and makes an appropriate adjustment to its connection weights. Back propagation performs a gradient descent within the solution's vector space towards a 'global minimum' along the steepest vector of the error surface. The global minimum is that theoretical solution with the lowest possible error.

The error surface itself is a hyper paraboloid but is seldom 'smooth'. Indeed, in most problems, the solution space is quite irregular with numerous 'pits' and 'hills' which may cause the network to settle down in a 'local minimum' which is not the best overall solution. Since the nature of the error space cannot be known a prioi, neural network analysis often requires a large number of individual runs to determine the best solution. Most learning rules have built-in mathematical terms to assist in this process which control the 'speed' (Beta-coefficient) and the 'momentum' of the learning. The speed of learning is actually the rate of convergence between the current solution and the global minimum. Momentum helps the network to overcome obstacles (local minima) in the error surface and settle down at or near the global minimum [3, 7].

Once a neural network is 'trained' to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no back propagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation. It is also possible to over-train a neural network, which means that the network has been trained exactly to respond to only one type of input; which is much like rote memorization. If this should happen then learning can no longer occur and the network is referred to as having been "grand mothered" in neural network jargon. In real-world applications this situation is not very useful since one would need a separate grand mothered network for each new kind of input.

2.2 Advantages and Disadvantages

There are many advantages and limitations to neural network analysis and to discuss this subject properly, should be looked at each individual type of network, which isn't necessary for this general discussion.

Depending on the nature of the application and the strength of the internal data patterns you can generally expect a network to train quite well. This applies to problems where the relationships may be quite dynamic or non-linear. ANNs provide an analytical alternative to conventional techniques which are often limited by strict assumptions of normality, linearity, variable independence etc. Because an ANN can capture many kinds of relationships it allows the user to quickly and relatively easily model phenomena which otherwise may have been very difficult or impossible to explain otherwise.

In reference to back propagation networks however, there are some specific issues potential users should be aware of. Back propagation neural networks (and many other types of networks) are in a sense the ultimate 'black boxes'. Apart from defining the general architecture of a network and perhaps initially seeding it with a random numbers, the user has no other role than to feed it input and watch it train and await the output. In fact, it has been said that with back propagation, "you almost don't know what you're doing". Some software freely available software packages do allow the user to sample the networks progresses at regular time intervals, but the learning itself progresses on its own. The final product of this activity is a trained network that provides no equations or coefficients defining a relationship (as in regression) beyond its own internal mathematics. The network 'IS' the final equation of the relationship.

Back propagation networks also tend to be slower to train than other types of networks and sometimes require thousands of epochs. If run on a truly parallel computer system this issue is not really a problem, but if the BPNN is being simulated on standard serial machine (i.e. a single SPARC, Mac or PC) training can take some time. This is because the machines CPU must compute the function of each node and connection separately, which can be problematic in very large networks with a large amount of data. However, the speed of most current machines is such that this is typically not much of an issue.

3. OPTICAL CHARACTER RECOGNITION

The mechanical or electronic conversion of typewritten or printed text into machine-encoded text is called Optical Character Recognition (OCR). It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitizing printed texts so that it can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision [2].

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components. Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind. In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code. Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters. In the late 1920s and into the 1930s Emanuel Goldberg developed what he called a "Statistical Machine" for searching microfilm archives using an optical code recognition system. In 1931 he was granted USA Patent number 1,838,389 for the invention. The patent was acquired by IBM.

4. TAMIL LANGUAGE

Tamil belongs to the southern branch of the Dravidian languages, a family of around 26 languages native to the Indian subcontinent. It is also classified as being part of a Tamil language family, which alongside Tamil proper,

also includes the languages of about 35 ethno-linguistic groups such as the Irula and Yerukula languages. The closest major relative of Tamil is Malayalam; the two began diverging around the 9th century. Although many of the differences between Tamil and Malayalam demonstrate a pre-historic split of the western dialect, the process of separation into a distinct language, Malayalam, was not completed until sometime in the 13th or 14th century. According to Hindu legend, Tamil, or in personification form Tamil Tāy (Mother Tamil), was created by Shiva. Shiva's Son, Lord Murugan, known as Lord Kartikeya in other Indian languages, and the sage Agastya brought it to people [4].

Tamil phonology is characterized by the presence of retroflex consonants and multiple rhotics. Tamil does not distinguish phonologically between voiced and unvoiced consonants; phonetically, voice is assigned depending on a consonant's position in a word. Tamil phonology permits few consonant clusters, which can never be word initial. Native grammarians classify Tamil phonemes into vowels, consonants, and a "secondary character", the āytam. The vocabulary of Tamil is mainly Dravidian. A strong sense of linguistic purism is found in Modern Tamil, which opposes the use of foreign loanwords. Nonetheless, a number of words used in classical and modern Tamil are loanwords from the languages of neighboring groups. In more modern times, Tamil has imported words from Urdu and Marathi, reflecting groups that have influenced the Tamil area at various points of time, and from neighboring languages such as Telugu, Kannada, and Sinhala. During the modern period, words have also been adapted from European languages, such as Portuguese, French, and English.

The strongest impact of purism in Tamil has been on words taken from Sanskrit. During its history, Tamil, along with other languages like Telugu, Kannada, Malayalam etc., was influenced by Sanskrit in terms of vocabulary, grammar and literary styles, reflecting the trend of Sanskritisation in the Tamil country. Tamil vocabulary never became quite as heavily sanskritised as that of the other Dravidian languages, and unlike in those languages, it was and remains possible to express complex ideas (including in science, art, religion and law) without the use of Sanskrit loan words. In addition, Sanskritisation was actively resisted by a number of authors of the late medieval period, culminating in the 20th century in a movement called tan̲it tamil̲ iyakkam (meaning "pure Tamil movement"), led by Parithimaar Kalaignar and Maraimalai Adigal, which sought to remove the accumulated influence of Sanskrit on Tamil. As a result of this, Tamil in formal documents, literature and public speeches has seen a marked decline in the use Sanskrit loan words in the past few decades, under some estimates having fallen from 40–50% to about 20%. As a result, the Prakrit and Sanskrit loan words used in modern Tamil are, unlike in some other Dravidian languages, restricted mainly to some spiritual terminology and abstract nouns. In the 20th century, institutions and learned bodies have, with government support, generated technical dictionaries for Tamil containing neologisms and words derived from Tamil roots to replace loan words from English and other languages [5].

5. IMPLEMENTATION

This paper is implemented with the help of MATLAB Simulink environment. The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB is a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. Using the MATLAB product, one can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN. MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology. It include features like high-level language for technical computing, development environment for managing code, files, and data, interactive tools for iterative exploration, design, and problem solving, mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, and numerical integration, 2-D and 3-D graphics functions for visualizing data and tools for building custom graphical user interfaces.

5.1 Raw input of Tamil character

As shown in the Fig.2 system design, the Tamil character to be recognized is given as input. Initially the Tamil character is converted to grayscale and the grayscale image is converted into binary image in-order to make it convenient for pre-processing.

5.2 Pre-processing the input

Most of the recognition and classification techniques require the data to be in a predefined type and format which satisfy several requirements like size, quality, and invariance. These requirements are generally not met in the case of regional languages, because of many factors such as noise during digitalizing, irregularity, and styles variations. Preprocessing is performed to overcome these problems by performing smoothing, point clustering and dehooking. First smoothing is performed using low pass filter algorithm to reduce noise and remove imperfection caused by acquisition device. Point clustering is then performed in order to eliminate redundant points by averaging the neighboring points. Dehooking is the final preprocessing procedure to eliminate the part of the stroke which contains hooks which are commonly encountered at the strokes ends.

5.3 Formation of feature matrix

The input characters are sampled to 60 points and feature matrix is formed. The matrix is then transformed into 8 subspaces using OCR. There are two basic types of core OCR algorithm, which may produce a ranked list of candidate characters. Matrix matching involves comparing an image to a stored glyph on a pixel-by-pixel basis; it is

also known as pattern matching, pattern recognition, or image correlation. This relies on the input glyph being correctly isolated from the rest of the image, and on the stored glyph being in a similar font and at the same scale. This technique works best with typewritten text and does not work well when new fonts are encountered. This is the technique the early physical photocell-based OCR implemented, rather directly.

Feature extraction decomposes glyphs into "features" like lines, closed loops, line direction, and line intersections. These are compared with an abstract vector-like representation of a character, which might reduce to one or more glyph prototypes.

General techniques of feature detection in computer vision are applicable to this type of OCR, which is commonly seen in "intelligent" handwriting recognition and indeed most modern OCR software. Nearest neighbour classifiers such as the k-nearest neighbours algorithm are used to compare image features with stored glyph features and choose the nearest match. Software such as Cuneiform and Tesseract use a two-pass approach to character recognition. The second pass is known as "adaptive recognition" and uses the letter shapes recognized with high confidence on the first pass to recognize better the remaining letters on the second pass. This is advantageous for unusual fonts or low-quality scans where the font is distorted (e.g. blurred or faded).

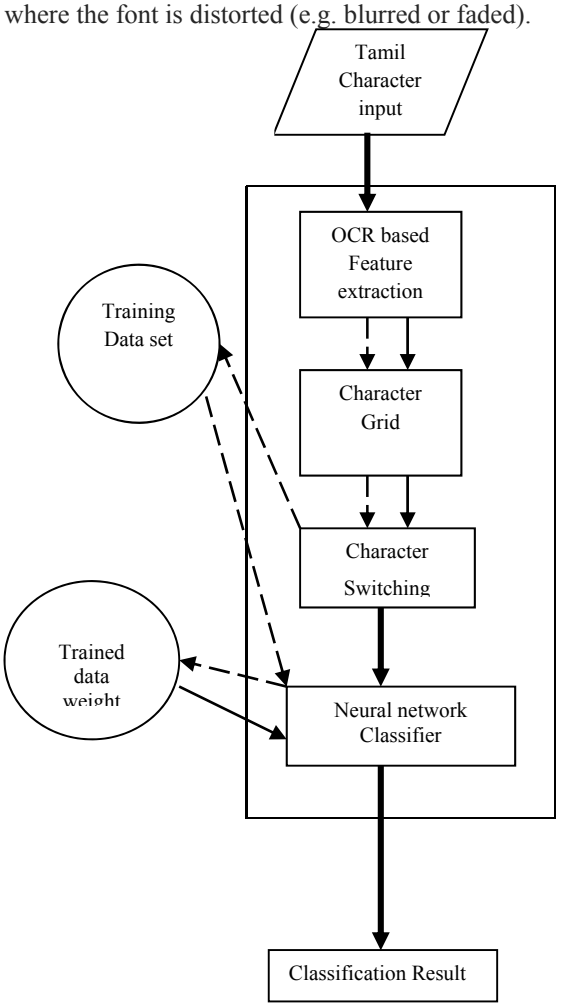

Fig 2: System Design

5.4 Character grid

In the proposed system design, feature extraction consists of three steps: extreme coordinates measurement, grabbing character into grid, and character digitization. The Tamil character is captured by its extreme coordinates from left /right and top/bottom and is subdivided into a rectangular grid of specific rows and columns. The algorithm automatically adjusts the size of grid and its constituents according to the dimensions of the character. Then it searches the presence of character pixels in every box of the grid. The boxes found with character pixels are considered “on” and the rest are marked “off” as shown in Fig.6.

5.5 Character Switching

A binary string of each character is formed locating the “on” and “off” boxes (named as character switching) and presented to the neural network input for training and recognition purposes. The total number of grid boxes represented the number of binary inputs. A 14x8 grid thus resulted in 112 inputs to the recognition model. An equivalent statement would be that a 14x8 grid provided a 112 dimensional input feature vector. The developed software contains a display of this phenomenon by filling up the intersected squares as shown in Fig.7.

5.6 Neural network based classifier

The work flow for the neural network design process has seven steps:

a) Collect data
b) Create the network
c) Configure the network
d) Initialize the weights and biases
e) Train the network
f) Validate the network
g) Use the network

After a neural network has been created, it needs to be configured and then trained. Configuration involves arranging the network so that it is optimal with classifying the Tamil characters, as defined by sample data [3].

After the network has been configured, the adjustable network parameters (called weights and biases) need to be tuned, so that the network performance is optimized. This tuning process is referred to as training the network. Configuration and training require that the network be provided with example data. For many types of neural networks, the weight function is a product of a weight times the input, but other weight functions (e.g., the distance between the weight and the input, $|w - p|$) are sometimes used. The most common net input function is the summation of the weighted inputs with the bias, but other operations, such as multiplication, can be used [6, 7].

5.7 Test data and output

5.7.1 Unit testing

Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software

design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

5.7.2 Functional tests

Functional test cases involved exercising the code with nominal input values for which the expected results are known, as well as boundary values and special values, such as logically related inputs, files of identical elements, and empty files. Three types of tests in Functional test:

a) Performance Test
b) Stress Test
c) Structured Test

5.7.3 Performance test

It determines the amount of execution time spent in various parts of the unit, program throughput, and response time and device utilization by the program unit.

5.7.4 Stress test

Stress Test is those test designed to intentionally break the unit. A Great deal can be learned about the strength and limitations of a program by examining the manner in which a programmer in which a program unit breaks.

5.7.5 Structured test

Structure Tests are concerned with exercising the internal logic of a program and traversing particular execution paths. The way in which White-Box test strategy was employed to ensure that the test cases could Guarantee that all independent paths within a module have been have been exercised at least once.

a) Exercise all logical decisions on their true or false sides.
b) Execute all loops at their boundaries and within their operational bounds.
c) Exercise internal data structures to assure their validity.
d) Checking attributes for their correctness.
e) Handling end of file condition, I/O errors, buffer problems and textual errors in output information.

5.7.6 Integration testing

Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected. The major error that was faced during the project is linking error. When all the modules are combined the link is not set properly with all support files. Then we checked out for interconnection and the links. Errors are localized to the new module and its interconnections. The product development can be staged, and modules integrated in as they complete unit testing. Testing is completed when the last module is integrated and tested.

5.8 Testing strategies

5.8.1 Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consists of several key activities and steps for run program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing.

The software testing process commences once the program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Otherwise the program or the project is not said to be complete. Software testing is the critical element of software quality assurance and represents the ultimate the review of specification design and coding. Testing is the process of executing the program with the intent of finding the error. A good test case design is one that as a probability of finding an yet undiscovered error. A successful test is one that uncovers a yet undiscovered error. Any engineering product can be tested in one of the two ways:

5.8.1.1 White box testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases. Basis path testing is a white box testing. Basis path testing:

a) Flow graph notation
b) Cyclometric complexity
c) Deriving test cases
d) Graph matrices Control

5.8.1.2 Black box testing

In this testing by knowing the internal operation of a product, test can be conducted to ensure that “all gears mesh”, that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software. The steps involved in black box test case design are:

a) Graph based testing methods
b) Equivalence partitioning
c) Boundary value analysis
d) Comparison testing

5.8.2 Software testing strategies:

A software testing strategy provides a road map for the software developer. Testing is a set activity that can be planned in advance and conducted systematically. For this reason a template for software testing a set of steps into which we can place specific test case design methods should be strategy should have the following characteristics:

a) Testing begins at the module level and works "outward" toward the integration of the entire computer based system.
b) Different testing techniques are appropriate at different points in time.
c) The developer of the software and an independent test group conducts testing.
d) Testing and Debugging are different activities but debugging must be accommodated in any testing strategy.

5.8.2.1 Integration testing:

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when we put them together. The problem of course, is "putting them together"- interfacing. There may be the chances of data lost across on another's sub functions, when combined may not produce the desired major function; individually acceptable impression may be magnified to unacceptable levels; global data structures can present problems [8].

5.8.2.2 Program testing:

The logical and syntax errors have been pointed out by program testing. A syntax error is an error in a program statement that in violates one or more rules of the language in which it is written. An improperly defined field dimension or omitted keywords are common syntax error. These errors are shown through error messages generated by the computer. A logic error on the other hand deals with the incorrect data fields, out-off-range items and invalid combinations. Since the compiler s will not deduct logical error, the programmer must examine the output. Condition testing exercises the logical conditions contained in a module. The possible types of elements in a condition include a Boolean operator, Boolean variable, a pair of Boolean parentheses, a relational operator or an arithmetic expression. Condition testing method focuses on testing each condition in the program. The purpose of condition test is to deduct not only errors in the condition of a program but also other a errors in the program.

5.8.2.3 Security testing:

Security testing attempts to verify the protection mechanisms built in to a system well, in fact, protect it from improper penetration. The system security must be tested for invulnerability from frontal attack must also be tested for invulnerability from rear attack. During security, the tester places the role of individual who desires to penetrate system.

5.8.2.4 Validation testing

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

a) The function or performance characteristics confirm to specifications and are accepted.
b) Validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Though there were deficiencies in the system they were not catastrophic [9].

5.8.2.5 User acceptance testing

User acceptance of the system is key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required. This is done in regarding to the following points.

a) Input screen design.
b) Output screen design.

6. CONCLUSION AND FUTURE ENHANCEMENT

The proposed technique is tested on a dataset of Tamil Characters. Five training samples for each character were used. The characters are resampled to 60 points and normalized to [0, 1]. A character feature matrices of size 60x15 was constructed and transform the features to an 8-dimensional subspace by performing ORC software. A nearest neighbor classifier is used to classify the test character in the subspace. If the estimated class label is one of the confusion pairs, we input the test character to an appropriate neural network based classifier at the next level. There is an increase in the classification accuracy of a few frequently confused characters after the neural network classifier. The improvement in performance is observed in both the validation/training and test sets.

In a ROC curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve is a measure of how well a parameter can distinguish between two diagnostic groups as shown in Fig. 8, 9, 10.

As a future enhancement to this project, in addition to giving visual feedback of the recognized character we can also provide audio feedback of the same. The same technique can be used to resolve confusions between triplets and quadruples of similar characters. The same recognition could be carried out not only for individual characters but Tamil words also.

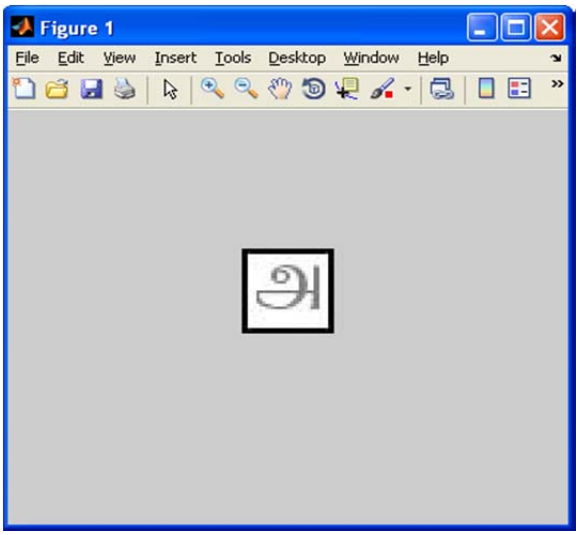

Fig 3: Grayscale image

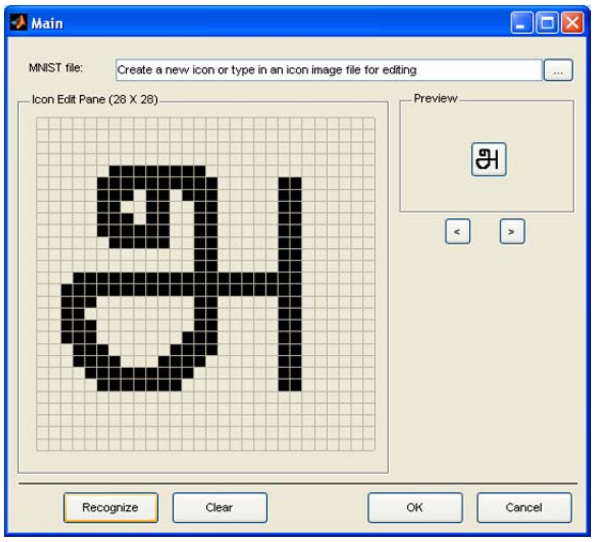

Fig 6: Character written on pad for recognition

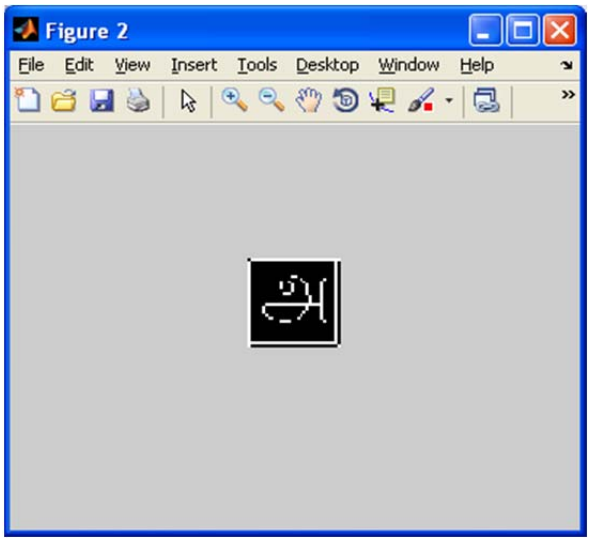

Fig 4: Converted binary image

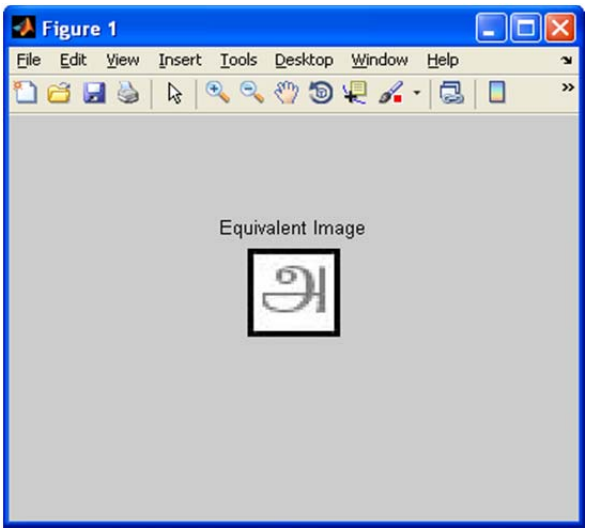

Fig 7: Displaying the recognized character

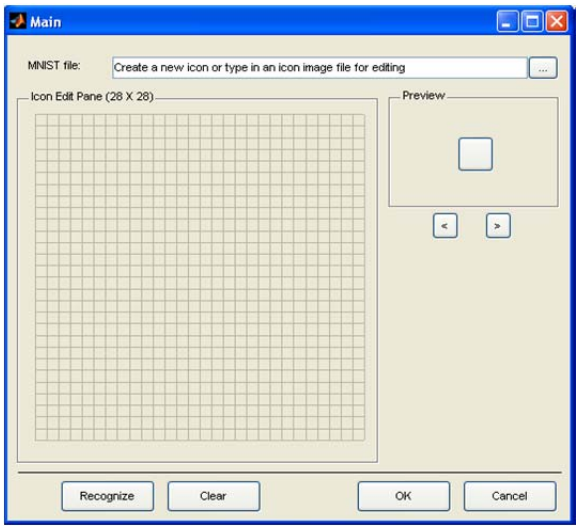

Fig 5: Pad for writing the characters

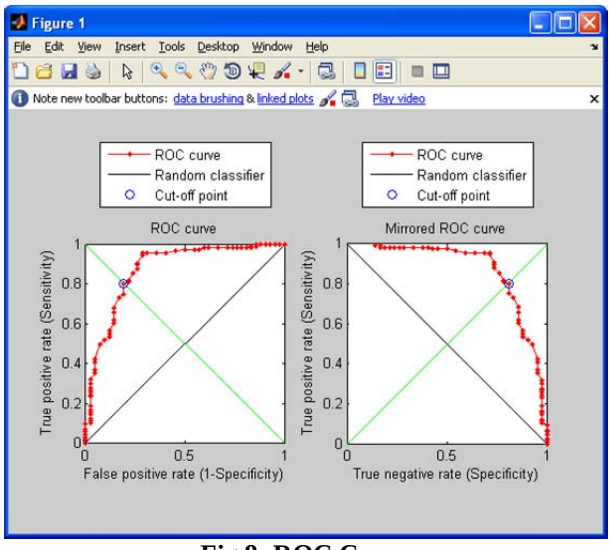

Fig 8: ROC Curve

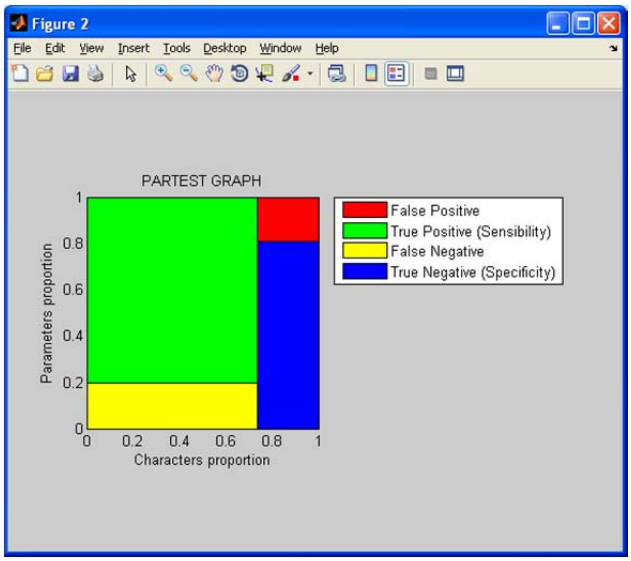

Fig 9: Sensitivity and specificity graph

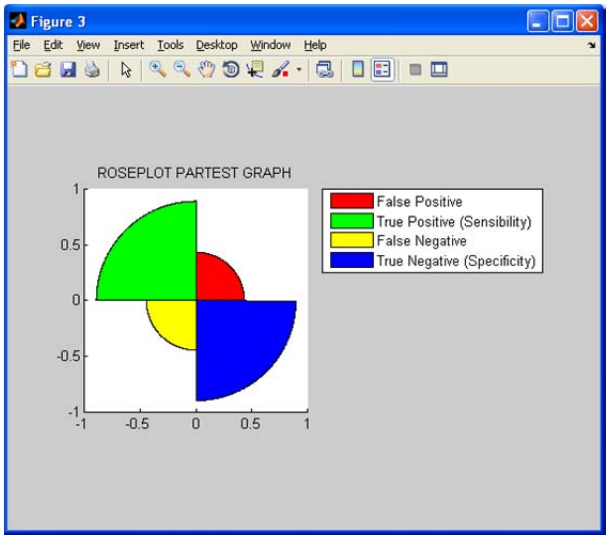

Fig 10: Roseplot partest graph

REFERENCES

[1] R. Jagadeesh Kannan, R. Prabhakar, "A Comparative Study of Optical Character Recognition for Tamil Script", European Journal of Scientific Research, Vol.35 No.4, 2009, pp: 570-582.

[2] R.Indra Gandhi, Dr.K.Iyakutti, "An Attempt to Recognize Handwritten Tamil Character Using Kohonen SOM" International Journal of Advance Networking and Applications, Volume 1, Issue 3, 2009, pp: 188-192.

[3] Dr.PSJ Kumar, "Neural network based block classification of computer screen image for desktop sharing" International Journal of Advanced Research in Computer Engineering and Software Engineering, Vol.4, Issue 8, Aug'2014, pp: 703-711.

[4] Dr. C P Sumathi, S Karpagavalli, "Techniques and methodologies for recognition of Tamil typewritten and handwritten characters: a survey" International Journal of Computer Science & Engineering Survey, Vol.3, No.6, December 2012, pp: 23-35.

[5] Dr.Amitabh Wahi, Mr.Sundaramurthy.S, Poovizhi.P, "Recognition of Handwritten Tamil Characters using Wavelet International Journal of Computer Science & Engineering Technology", Vol. 5 No. 4, Apr 2014, pp: 335-340.

[6] M.Sivasankari, Dr.S.P.Victor, "Multilingual Handwritten Text Verification", International Journal of Computer Science and Information Technologies, Vol. 5, 2014, pp: 3605-3607.

[7] Harshal Bobade, Amit Sahu, "Character Recognition Technique using Neural Network", International Journal of Engineering Research and Applications, Vol. 3, Issue 2, March - April 2013, pp:1778-1783.

[8] Prashant M. Kakde, Vivek R. Raut, "Performance analysis of handwritten Devnagri characters recognition through Machine intelligence", International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 7, July 2012, pp: 238-247.

[9] Dhiraj K. Das, "Comparative Analysis of PCA and 2DPCA in Face Recognition", International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 1, January 2012, pp: 330-336.

BIOGRAPHY

Corresponding Author

Dr. P. S. Jagadesh Kumar, Professor in the Department of Computer Science and Engineering, Don Bosco Institute of Technology, Bengaluru has 16 years of teaching experience, including 6 year of research experience in the field of image compression. He received his B.E. degree from University of Madras in Electrical and Electronics Engineering discipline in the year 1999. He obtained his M.E degree in 2004 with specialization in Computer Science and Engineering from Annamalai University, Chidambaram and his Ph.D. from Anna University, Chennai.