UNIVERSITY OF CALIFORNIA
RIVERSIDE

Expectation and Knowledge Guided Neural Networks for Image Recognition

A Thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Electrical Engineering

by

Ahmed Mohamed El-Gabaly

December 2016

ABSTRACT OF THE THESIS

Expectation and Knowledge Guided Neural Networks for Image Recognition

by

Ahmed Mohamed El-Gabaly

Master of Science, Graduate Program in Electrical Engineering
University of California, Riverside, December 2016
Dr. Ping Liang, Chairperson

Psychological research **[21]** shows that in order to visually identify an object, our brain generates an initial hypothesis of what it is, and tests stored patterns from memory against this hypothesis to see if the expectations or predictions that are based on this hypothesis are true. If they are true, then the hypothesis is correct, if not, the hypothesis is not supported and the cycle repeats until it reaches a conclusion on what it sees. In other words, "we see what we expect to see". In this thesis, we show how a neural network based pattern recognition system built upon this psychological concept can help improve the success rate of visual pattern recognition using offline handwritten character recognition as an example. Where an initial neural network classification of an n letter word is separated using a threshold and refined over multiple iterations of a reasoning and recognition process using specifically chosen weighted letter mask templates applied on every new classification. Statistical knowledge guides what threshold to use and what weights to use for the masks while symbolic knowledge guides which letter mask templates to use.

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

Psychological research shows that the way we visually process the world around us is through an if/then/therefore pattern where a person looks at part of an unknown object and the brain immediately generates an idea or initial hypothesis of what it is, then the brain generates and tests stored patterns from memory against this hypothesis to see if the expectations or predictions that are based on this hypothesis are true or not. If they are true then the hypothesis is correct, if not, then the hypothesis is not supported and the cycle repeats until it reaches a conclusion on what it sees **[21].** It is known and experimentally proven that people often recognize words with misspellings and transposed letters without even noticing the errors **[24]** and can recognize words from handwritten scribbles, which contain ambiguous letters **[25].** Our daily experience version of this is "you see what you expect to see." This thesis is motivated by this feature of cognitive function where we designed a system that helps a feed forward neural network do more than just classifying a pattern based on supervised training using training samples; feed forward neural networks can learn the relationship between a pattern and its class through calculated weights using a training algorithm and when given a similar pattern it can classify it with an accuracy that is based on how well the training algorithm was tweaked. However, feed forward neural networks based pattern recognition systems do not go further than that. The aim of this work is to explore how the incorporation of contextual knowledge and visual pattern expectation based on a "you see what you expect to see" psychological cognitive model into these systems will improve their performance. Neural network based pattern recognition systems excel in classifying various signal types including visual, speech, sonar, radar, seismic, and other time varying signals **[22].** In this paper, we are mainly interested in visual signals, in particular, English hand written character recognition. Character recognition is the technology that enables the conversion of different types of documents such as scanned paper documents, PDF files or images captured by a digital camera into editable textual data. One type of character recognition is handwritten character recognition **[1]**, which is the ability of a computer to receive and interpret handwritten input from sources such as paper documents, photographs or touch screens. It has evolved through the years and is applied in many applications including converting handwritten documents to typed documents [**26**], Signature Verification where Image based handwritten signature verification is important in financial transactions when a hard copy of a signature is needed **[2]**, Postal-Address Interpretation where it facilitates postal service automation **[3]**, bank check processing where the recognition, verification and data entry of the information on checks can be done automatically **[4]** and Writer Recognition where it is used to determine the identity of a writer based on the analysis of their handwriting **[5]**. This thesis aims to describe an approach to improve the word recognition accuracy of a neural network for handwritten character recognition. The next section describes the background of handwritten character recognition systems reviewing popular methods that have been used to achieve handwritten character recognition. It is followed by our approach for creating a handwritten character recognition system. Then we describe the implementation of that system and the dataset that was used, and finally the analysis of the tests done on this system.

2. BACKGROUND

There are two types of handwritten character recognition, either offline handwritten character recognition or online handwritten character recognition. In the former, the completed handwritten character pattern is captured as an image and taken for testing purposes while in the latter, the time sequence of writing out a character pattern is captured as digitized points on a coordinate and each point of the character pattern is a function of pressure, time, slant, strokes and other physical parameters **[6]**. In this thesis, we are using offline handwritten character recognition. The general model for an offline handwritten recognition system is shown in Fig.1

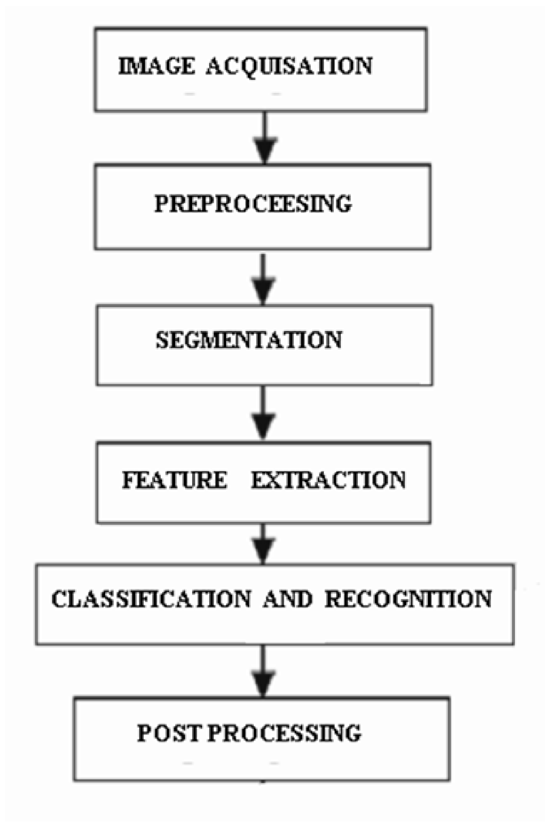


Fig 1, Offline handwritten recognition system general model

The steps of this general system are described in **[7]** as follows:

2.1. Image Acquisition

The recognition system acquires a digital image with a specific format such as JPEG, BMP, etc through a scanner, a digital camera or another digital input device like a pen. **[8]** Shows different ways in which images of English and Kaneda letters were acquired including photographing images containing those letters and handwriting the letters on a tablet PC using a pen.

2.2.Pre-Processing

This step is a series of operations on the scanned input image. It enhances the image rendering it suitable for segmentation. The role of pre-processing is to segment the interesting character pattern in the image from the background or the rest of the image. These operations include but are not limited to **[9]** Image Enhancement, Noise Removal, Skew Detection and Morphological Processing.

2.2.1. Image Enhancement

An example of Image Enhancement could be Histogram equalization. A histogram plots the frequency at which each grey-level value occurs in the image; histogram equalization stretches the image histogram across the entire spectrum of pixels (0-255). It increases the contrast of images and can be applied to normalize illumination variations in the image.

2.2.2. Noise Removal

Images acquired through the camera sensors can be contaminated by a variety of noise sources. These noise sources are stochastic variations rather than deterministic distortions, such as shading or lack of focus. One type of this noise is salt and pepper noise, which can be removed by a variety of filters including the median filters or Gaussian filters **[10]**

2.2.3. Skew Detection

It is the slight rotation of the characters in the acquired image when compared to the characters in the real document or scene. This can degrade the performance of the handwritten character recognition system as a whole. The popular methods used to detect and correct the skew in the acquired image are based on Projection profile, Fourier transform, cross-correlation, Nearest Neighbor connectivity, linear regression analysis and Hough transform. The Hough transform method is considered one of the best due to its accuracy and simplicity **[11]**

2.2.4. Morphological Processing

Preprocessing techniques might cause distortions in the image where some pixels might be removed producing holes to some parts of the image, for example, thresholding can create holes that can cause the characters to break into two or more objects. The way these problems can be solved is by morphological operations. These can include for example erosion/dilation or Opening/Closing. Erosion makes an object smaller by removing the pixels on its edges while dilation makes an object larger by adding pixels around the edges. They both do it using a thresholding technique, where the operation is done on a pixel based on its neighboring pixels. In the erosion method, if the number of zero pixels in the neighborhood of a pixel exceeds a certain threshold then this pixel is set to zero while in the dilation method if the number of pixels next to a zero pixel exceed a certain threshold then the zero pixel is set to the value of those pixels.

2.3.Character Segmentation and Image Size Normalization

This is the stage where the acquired image of the sequence of characters is decomposed into sub images of individual characters. According to **[12],** there are three popular approaches for character segmentation. First there is the dissection approach, in which segments are identified based on character like properties. Second, there is the recognition based approach in which the system searches the acquired image for components that match classes in its alphabet. Finally, there is the holistic approach in which the system seeks to recognize words as a whole, thus, avoiding the need to segment into characters. The result from the character segmentation stage provides isolated characters, which are ready to be passed into the feature extraction stage, therefore; the isolated characters are normalized into a specific size, decided empirically or experimentally depending on the application and the feature extraction or classification techniques used.

2.4.Feature Extraction

After Character segmentation and size normalization comes the Feature extraction stage which prepares the letters for the classification and Recognition stage. It is the process that captures the salient characteristics of the segmented characters; the goal of this step is to extract a set of features, which help maximize the recognition rate with the least amount of elements where each character is represented by a feature vector, which represents its identity. In **[14]** the different types of feature extraction methods are explained. They are divided into three types: first, there are statistical feature based methods, second, there are structural feature based methods and finally, global transformation techniques. Statistical methods use the information related to the statistical distribution of pixels in the image. Structural features are extracted from the structure and geometry of the character like number of horizontal and vertical lines, aspect ratio, number of cross points, number of loops, number of branch points, number of strokes or numbers of curves etc. on the other hand global transformation features are calculated by converting the character image to the frequency domain like Discrete Fourier Transform, Discrete Cosine Transform, Discrete Wavelet Transformation, Gabor filtering, etc.. In addition, the features extracted can be either high level or low level. Low level features include width, height, aspect ratio etc., while high level features include number and position of loops, straights lines, headlines, curves etc.. An example of feature extraction is shown in **[13]** where every character image is divided into equal zones, each of size 10x10 pixels. The features are extracted from each zone pixel by moving along the diagonals of its respective 10x10 pixels, which yields higher levels of recognition accuracy when compared to other methods that employ horizontal and vertical methods of feature extraction.

2.5.Classification and Recognition

This is the decision making part of the handwritten character recognition system. It uses the features extracted in the previous stage to classify a character pattern. The classification methods include statistical methods, kernel methods, artificial neural networks or a combination of different types. These methods fall under four general approaches of pattern recognition: Template Matching, Statistical Techniques, Structural Techniques and Neural Networks as suggested in **[15].**

These approaches can be described in **[16]** as follows,

2.5.1. Template Matching

This method recognizes the class of a handwritten character by comparing it to a template similar to it. Generally the way a template is matched to the character pattern in question is to find the degree of similarity between two vectors. These vectors are a different representation of the template being matched and the character pattern in question, they can be a group of pixels, shapes, curvature, etc.) in the feature space. The template matching techniques can be divided into direct matching, deformable templates and elastic matching.

2.5.1.1.Direct Matching

This is the method where a gray level or binary input character is compared to patterns stored in a database according to a similarity measure like (Jaccard or Euclidean, etc.). The direct matching technique can go from a simple one-to-one comparison based on pixels up to a complex decision tree. In addition, the technique can be a combination of multiple information sources as shown in **[17],** where a template for each character is generated using skeleton images created in the preprocessing stage. The ratio of a similarity function and a dissimilarity function is used to decide how close an input pattern is to the generated templates. The former defines the similarity between an input image with a given template image using a bit wise and operator over the entire image, while the latter defines the dissimilarity between an input image with a given template image using a bit wise exclusive or operator over the entire image.

2.5.1.2.Deformable Templates and Elastic Matching

A different way of doing template matching is the use of deformable templates where an image deformation is used to match an unknown image against a database of known images. An example is shown in **[18]** where two characters are matched by deforming the shape of one to fit the edge power of the other where the degree of similarity is derived from the amount of bend needed, how well the edges fit together and the interior overlap between the distorted shapes.

2.5.2. Statistical Techniques

The core of these methods is the Bayesian decision rule. They are divided into non-parametric recognition and parametric recognition. In the former, prior information about the data is not needed, one of the best known non parametric methods used in character recognition is the Nearest Neighbor where an incoming pattern is classified using a cluster whose center is the minimum distance from the pattern over all the clusters. While in the latter, the priori information about the characters in the training data is used to generate a parametric model of each character which in turn is used to classify a new incoming letter using decision rules such as Bayesian or the maximum likelihood.

2.5.3. Neural Networks

They are the best classification methods for character recognition. The three main categories that define the structure of a neural network are the network topology which is the number of neurons used and how the neurons are interconnected, the node function which is the type of transfer function used by the neuron for calculating its output value and the training rules which specify the initialization of the neural network weights, and the training algorithm, i.e., how they are adjusted to improve the performance of the network. There are many other parameters that control the performance of the neural network including, the number of layers, and the number of neurons in each layer **[19]**.

The architecture chosen for a neural network depends on the complexity of the application it is used to solve. A multilayer neural network with the proper choice of parameters is capable of classifying any pattern.

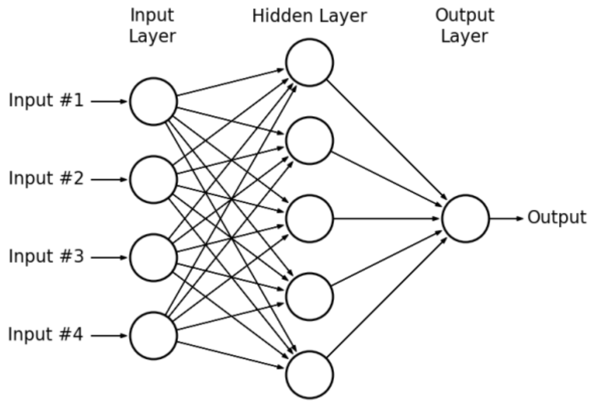


Fig 2, Architecture of multilayer neural network

The state of the art in solving the handwritten recognition problem is using deep learning. It allows for computational models composed of multiple processing layers to learn representations of data with multiple levels of abstraction. **[32] [33] [34]** show how deep learning neural networks have shown superior results when compared to traditional multilayer neural networks.

Figure 3 shows the main difference between traditional neural networks and deep learning neural networks.

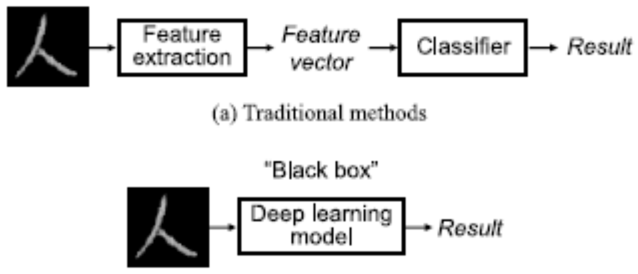


Fig 3, Traditional neural networks vs deep learning

In deep learning, both feature extraction and classification can be done automatically within the deep learning model, i.e. the input image is passed through the deep learning black box and an output classification is obtained. On the other hand, traditional multilayer neural networks require artificial feature design and manual tuning of the classifier, which greatly affects the performance of the neural network. In this thesis, the interest is in exploring the potential of adding knowledge and expectation to guide a regular multilayer neural network in improving its success rate in the application of handwritten character recognition.

Examples of different neural network topologies and training algorithms used to approach the offline handwritten character recognition problem are shown below:

In **[20]** a multilayer perceptron network was used to recognize Malayalam characters. The network consists of three layers, one input, one hidden and one output layer respectively, the input layer consists of 90 neurons, which receive binary pixels from a 15x12 binary image. The hidden layer contains a number of neurons decided based on optimal results on a trial and error basis, and the output layer contains a number of neurons equal to the number of Malayalam letters classes available. The training algorithm used is backpropagation with a network of 90 input layer neurons, 100 hidden layer neurons, 10 output layer neurons and initialized random weights.

The training steps are as follows for every character in the training set

1. Calculate the output classification of the neural network
2. Compare the obtained output with the label of the current character
3. Compute the error
4. Back propagate the error across each layer and adjust the weight values between the neurons
5. Check whether the error is minimum, if yes, then exit , Otherwise, continue the process

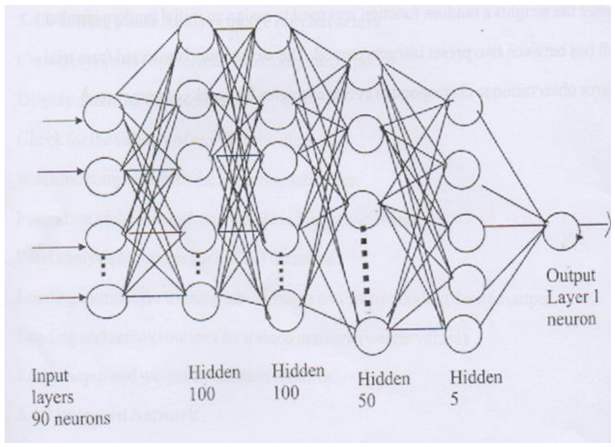


Fig 4, proposed Multilayer perceptron

In **[27]**, a feed forward back propagation neural network with an input layer of 35 neurons, 2 hidden layers with 100 neurons each and an output layer of 30 neurons was used to perform the classification of 26 English characters and 4 numeric characters. The training algorithm used is the gradient descent back propagation, which performs much faster than regular back propagation.

In **[28],** a feed forward neural network with 7000 neurons in the input layer, 70 neurons in the hidden layer and 52 neurons in the output layer, was used to classify uppercase and lowercase English letters. The output classification for the network is related to the output neuron with the highest activation. The dataset is divided into 3 sets, a training set, a validation set and a test set. The training algorithm used here is the scaled conjugate backpropagation, which tested on the validation set to make sure it does not over fit the results by minimizing the error on the validation set with a limit of 1000 epochs. The network was applied on the test set and its performance was measured.

In **[29]** a hybrid Hidden Markov model / Artificial neural network model is used. Each handwritten character image is preprocessed, the resulting feature vector, plus a left and right context, is processed by a multilayer perceptron. The outputs of the MLP, after dividing by the prior state probabilities are used as emission probabilities in the HMMs.

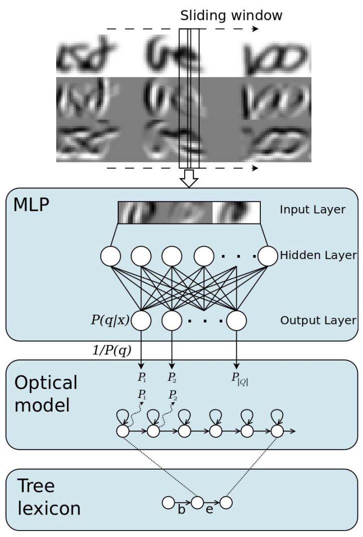


Fig 5, The proposed HMM/ANN hybrid model

The training of the system is done by an iterative Expectation-Maximization algorithm, where the training of the supervised ANN and Viterbi alignment of the training corpus are alternated, the training procedure is as follows:

1) Assign an initial labeling of the desired MLP outputs to every feature vector of the training and validation datasets. This labeling can be computed by dividing the image into equal parts or by using a previously trained hand recognition system in forced alignment mode
2) Assign an initial nonzero value to transition probabilities of the Markov chains
3) Train the supervised ANN with the training pairs, using the mean-squared error on the validation data set as the stopping criterion
4) Use the partially trained hybrid ANN/HMM models to perform a forced Viterbi alignment of the training data. This Viterbi procedure uses the class priors estimated from the relative frequencies of each class in the training data. This Viterbi alignment is used both for obtaining a new segmentation or labeling of the training and the validation sets and also for counting the number of times each HMM transition has been used. These counts are used to re-estimate the transition probabilities
5) Go to step 3 until convergence, this is, until the difference between two consecutive iterations is below a threshold

In **[30]**, a recurrent neural network is used to retrieve all instances of a given keyword in a handwritten document. The system consists of a CTC Token Passing algorithm in conjunction with a recurrent neural network. The architecture of this network is modified to overcome the vanishing gradient problem that describes the exponential increase or decay of information in the recurrent connections of the neural network, to achieve this, the nodes in the network are replaced by long short-term memory (LSTM) cells displayed in figure 6. The network is bidirectional, which means that the text lines in the document are processed from left-to-right and right-to-left because context from both sides of a character increases the chance of its recognition. The information from two separate input layers is collected in two separate LSTM layers, respectively and finally joined in the output layer, this is shown in figure 7. The output layer contains one neuron for every possible character as well as one additional neuron activated when no evidence about the presence of any character can be inferred and finally, the normalization of the output activations to sum up to one results in a vector that can be interpreted as a character probability vector, as shown in figure 8. On the other hand, the CTC Token Passing Algorithm for single words expects a sequence of letter probabilities of length t as input from the neural network, together with a word was a sequence of ASCII characters. The best path through the letter probability sequence is computed that corresponds with the letters from the input word w. The value of that path is then returned as a matching score, i.e. the probability that the input to the neural network was indeed the given word.

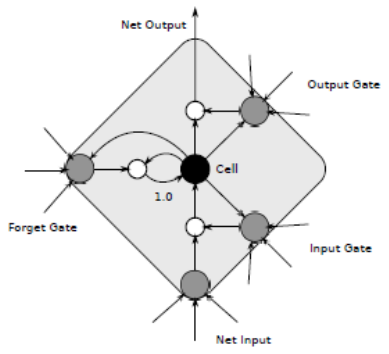


Fig 6, Gates control the information flow into and out of each LSTM node

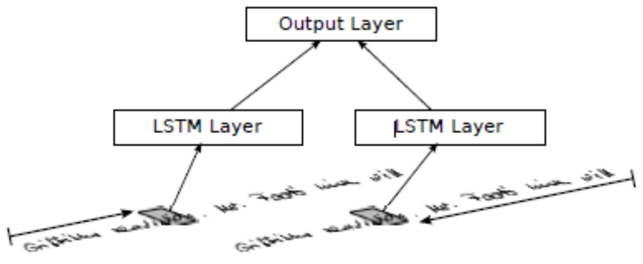


Fig 7, An illustration of the mode of operation of the BLSTM Neural Network. For each position, the output layer sums up the values of the two hidden LSTM layers

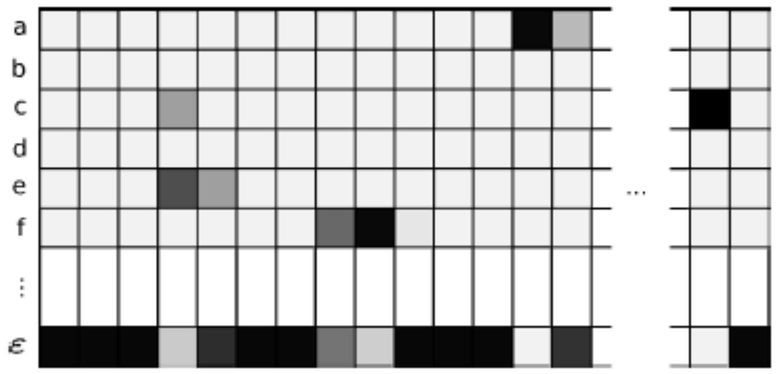


Fig 8, The activation level all nodes in the output layer

In **[31],** a method to train the members of a committee of one-hidden-layer neural nets to classify handwritten number characters is proposed. For a pattern recognition problem where pattern x is assigned to one of k possible classes. Using a softmax activation for the output layer of the neural nets and a 1 of k coding schemes for the target data, the outputs of the trained nets approximate the posterior class probabilities.

For n-trained neural networks, the paper focuses on three different methods to build the corresponding committee:

1) Majority voting committee: choose the class with most votes from the n classifiers for a given input x, if two classes have the same number of votes, choose the first.
2) Average committee: average the class probabilities from the n classifiers and choose the class with highest average posterior class probability for a given input x
3) Median committee: take the median of the class probabilities from the n classifiers and choose the class with the highest median posterior class probability for a given input x.

The neural networks used in this paper were MLPs with one hidden layer 800 neurons; the transfer function used is a standard softmax output non-linearity with cross-entropy loss function and hyperbolic tangent hidden unit activation function. The inputs are normalized and the weights are initialized with a zero mean Gaussian. All the MLPs are trained for 500 epochs with a stochastic conjugate gradient algorithm.

2.6.Post Processing

This is usually the final stage of the character recognition system where the corresponding recognized characters are output in the structural text form by calculating equivalent ASCII values using the recognition index of the test samples.

The system proposed in this paper, is one that aims to mimic a "you see what you expect to see" cognitive model. Using a combination of a neural network, which generates the initial hypothesis, contextual knowledge about the words to be recognized, captured in a dictionary, knowledge about the handwritten characters captured in masks generated using the training patterns, knowledge about the proper weights to apply to the masks and expectation guided template matching using the masks.

3. APPROACH

The objective of this research is not to build the best neural network for handwritten character recognition but rather to show how contextual knowledge and expectation can be incorporated into neural network based pattern recognition systems in the application of handwritten character recognition, where a word consisting of isolated handwritten character patterns is applied to the system. An initial hypothesis with a certain confidence level is generated about such patterns and through the use of contextual knowledge and confidence levels about the word subsequent hypotheses are built upon the former one until a final conclusion is reached about the class of the input handwritten character patterns. On the macro level, the system consists of 2 main processes, the Init variables process and the Classify process. The former receives the dataset as input and outputs the trained neural network, test set, masks, masks weights and knowledge as input to the latter for classification.

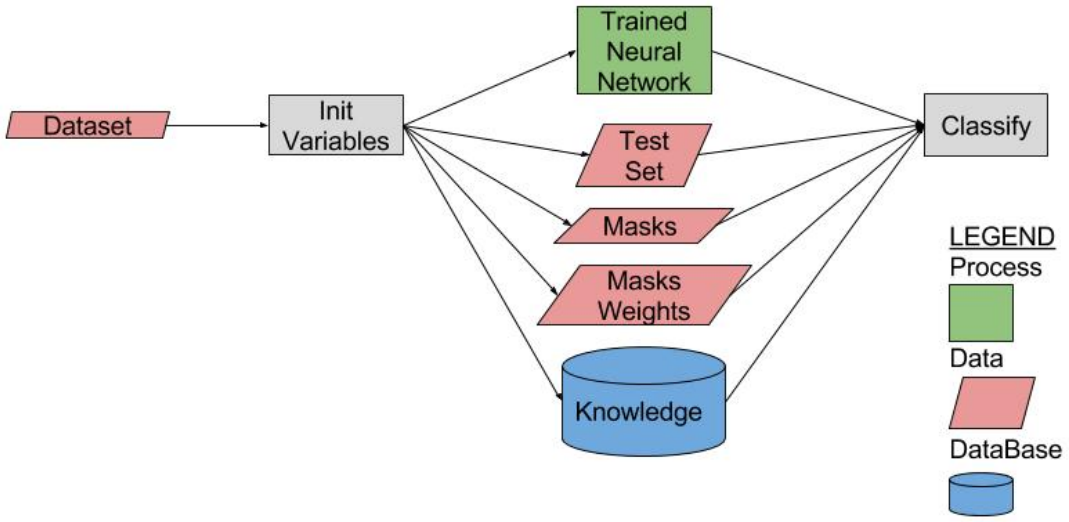


Fig 9, Macro Level

3.1.Init Variables

The Init variables process consists of four sub processes and the knowledge database. These processes are Preprocessing, Neural Network Training Algorithm, Create Masks and Weight Determination They can be described as follows:

3.1.1. PREPROCESSING

The aim of this process is to normalize the size of all the dataset letters and to convert them to gray scale. The steps of the process applied on every handwritten character image are as follows:

1) Compliment the image to make the letter white and the background black
2) Find the letter connected components
3) Get the coordinates of the bounding boxes around the labeled letter
4) Find the largest bounding box around the letter
5) Extract the bounded letter from the image
6) Resize extracted letter to 20x20
7) Convert the extracted letter image to grayscale
8) Normalize letter image pixel values between 0 and 1

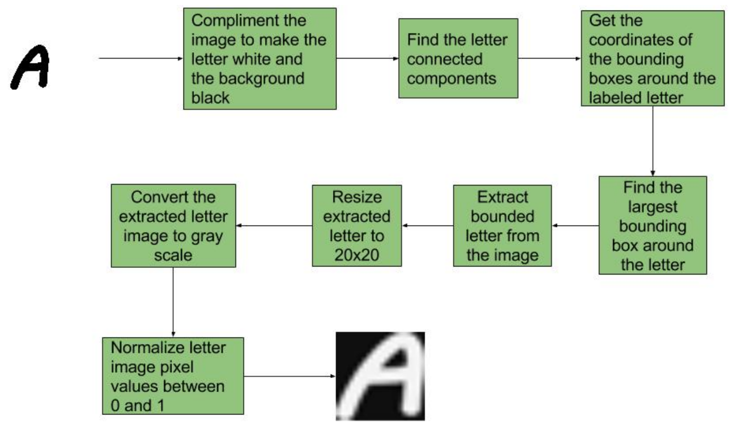

Compliment the image to make the letter white and the background black
Find the letter connected components
Get the coordinates of the bounding boxes around the labeled letter
Find the largest bounding box around the letter
Extract bounded letter from the image
Resize extracted letter to 20x20
Convert the extracted letter image to gray scale
Normalize letter image pixel values between 0 and 1

Fig 10, Preprocessing example

The 20x20 gray scale-handwritten character's images are each then reshaped into 400x1 vectors, then they are divided into a test set and a training set. The test set is used to test the performance of the system and the training set is used to train the neural network, creating the mask templates and determining the mask weights and the threshold that will separate the letter classifications into high confidence letters and low confidence letters

3.1.2. Neural Network Training Algorithm

The neural network used in this system is a combination of a feed forward neural network and a probabilistic neural network. In the Init variables process, they are both trained to produce the trained neural network which is passed to the Classify process.

3.1.2.1.The Feed Forward Neural Network training

The feed forward neural network is used to generate a 5x1 vector that represents the class of an input handwritten character 400x1 vector. Therefore, it is trained on 400 x n vectors which represent the training set characters and their corresponding binary 5 x n label vectors, the output is the trained feed forward neural network.

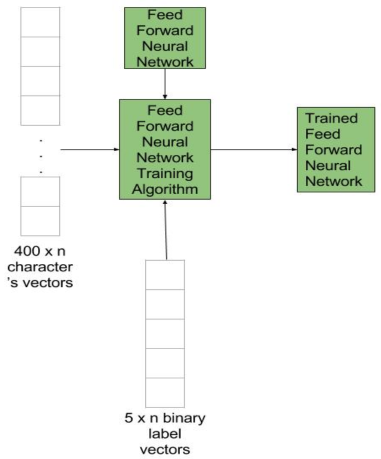

Feed Forward Neural Network
Feed Forward Neural Network Training Algorithm
Trained Feed Forward Neural Network
400 x n character 's vectors
5 x n binary label vectors

Fig 11, Feed forward neural network training algorithm

3.1.2.2.The Probabilistic neural network

The probabilistic neural network is used to map the 5x1 vector produced from the feed forward neural network to a 26x1 vector containing the confidence levels of the network towards all 26 letters classes where the index of the maximum confidence level in this vector represents the class of the input character. Therefore, it is created using 2 different representations of the 26 letters classes; a 26x5 matrix where every column is a 5 bit vector representing the index of a character and a 26x26 binary matrix where the diagonal elements are equal to one and each represents the class of each of the 26 characters.

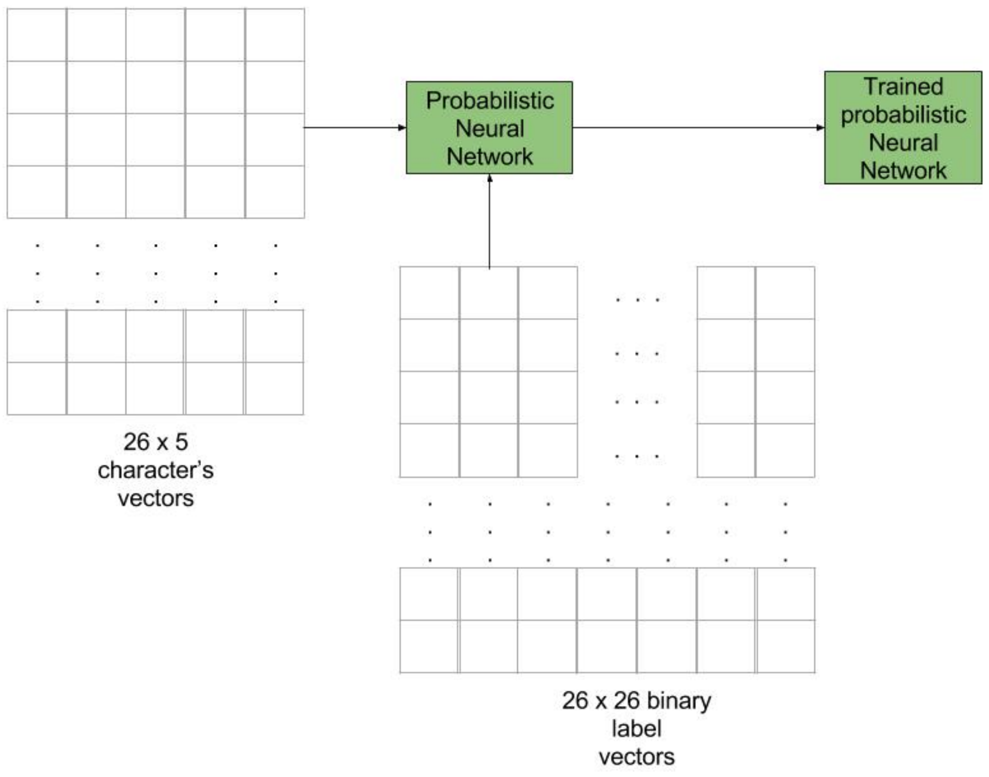

Probabilistic Neural Network
Trained probabilistic Neural Network
26 x 5 character's vectors
26 x 26 binary label vectors

Fig 12, Probabilistic Neural Network creation

These two networks are then connected to each other to produce the trained neural network, which would be used in the Classify process.

3.1.3. Create Masks

The masks are created from the training set classes. For every training set class, its n number of different patterns are added to each other and divided by n. For example, if there are 28 different patterns for the 'A' character that the feed forward neural network was trained on. Then these 28 matrix patterns are added together element wise and the final matrix is divided by 28. The mask template equation is shown below where for n patterns of a character class C

$$\text{Mask}(C) = (\sum \text{pattern}(C))/n$$

3.1.4. Weight Determination

The masks created in the previous step are applied on the input letter patterns in the Classify process with a specific weight W2 that is a fraction of the weight W1 which is multiplied by the original letter matrix. To determine these weights, W1 is set to a constant value and W2 is set from 0.1*W1 to a maximum of 0.9*W1. To determine the weights for every mask, the algorithm is as follows:

```
For 1 < Mask < 26
 For 1 <= Training Set Letter <= n
  For W1 = w
   For 0.1*w <= W2 <= 0.9*w
    Training Set Letter After Mask application = (W1 * Training Set Letter) + (W2 * Mask)
     Training Set Letter After Mask application(Mask)(W2) Class =
      Trained Neural Network (Training Set Letter After Mask)
```

There for, each of the 26 masks are applied on the entire training set with mask strength W2, as the mask strength increases from 0.1 to 0.9, the training set letter pattern starts changing to the current mask letter. For example, if letter ‘B’ mask is applied on all training set ‘A’ letters, then by the time W2 equals 0.9, the trained neural network classification of the ‘A’ letters will be equal to the ‘B’. After applying all the masks on all the training set letters with W2 ranging from 0.1 to 0.9, the non mask hit rate is calculated as follows

Consider the following variables:

nonMaskHit = number of non mask training set letters equal to their labels after mask application

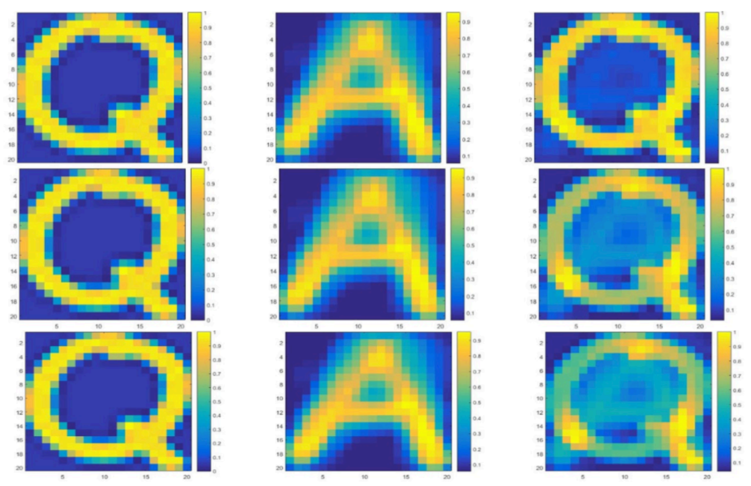
nonMaskMiss = number of non mask training set letters not equal to their labels after mask application

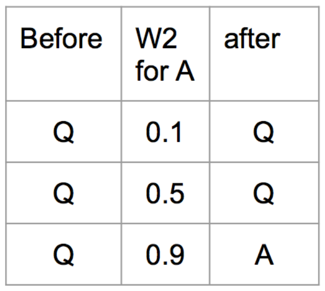
For 0.1 <= W2 <= 0.9

Non mask hit rate(W2) = nonMaskHit/(nonMaskHit + nonMaskMiss)

Using the non-mask hit rate, a metric is set for the choice of W2 specific to each mask, where for every mask, W2 is chosen as the smallest one where the non-mask hit rate >= 95% or non-mask hit rate >= 90% or non-mask hit rate >= 85% or non-mask hit rate >= 80% and the tests are done based on these choices.

Figure 13 shows the effect of the application of a mask on a letter, as the mask weight W2 increases the class of the letter on which the mask is applied moves closer to the mask letter class.





Before	W2 for A	after
Q	0.1	Q
Q	0.5	Q
Q	0.9	A

Fig 13, Mask Application example

3.1.5. Threshold determination

The threshold determination is guided by statistical knowledge from the training set letters. The neural network is applied on the entire training set. The confidence level of the classification of each character is recorded and the threshold is chosen as the average classification confidence level for the entire dataset.

3.1.6. Knowledge

The knowledge database is basically the contextual knowledge known about the problem at hand depending on the nature of the problem. In the case of the character recognition problem, the contextual knowledge is a dictionary of known words with different lengths which the classify process will search based on the threshold on the confidence level of the trained neural network on the input hand written character patterns, i.e. the contextual knowledge known about the handwritten character patterns presented to the system is the letters classified with a confidence level >= threshold in a word of length n.

3.2.Classify

This process receives the variables created in the Init Variables Process as initial inputs and uses the if/then/therefore approach on these variables to determine the class of an input test word. It uses three sub processes to reach a final conclusion on the classification of an input word. These processes are the apply mask pattern on character patterns, the neural network process from the Init Variables process and the SRU or symbolic reasoning unit to determine the classification of an input word made of different character patterns.

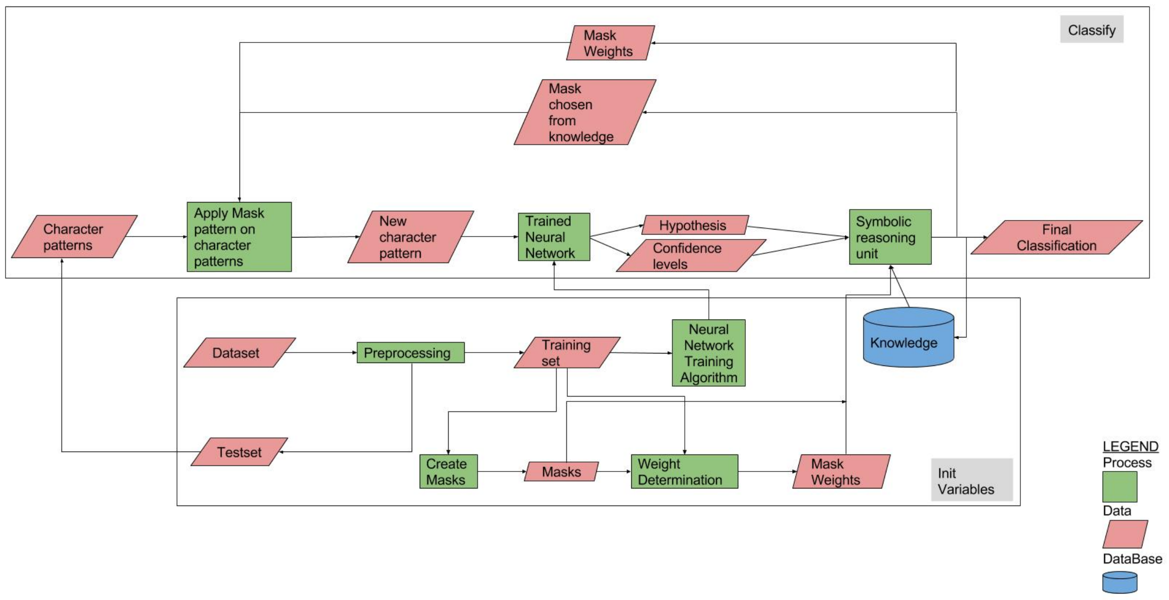


Fig 14 System Micro Level

The Classification process works as follows:

1) Consider an n letter word where each letter image pattern is randomly chosen from the test set, for example if the word in question is “OPEN” and there are 27 different handwritten character test image patterns for every character in the English alphabet, then the letter O image pattern is chosen randomly from the 27 ‘O’ image patterns available in the test set and so on for the rest of the characters
2) The Word patterns are passed through the Trained Neural network, an initial classification for each of the world letter image patterns is generated along with their confidence levels
3) The determined threshold chosen based on the training set statistics is used to separate the letter classifications into the high confidence letters group and the low confidence letter group where the former contains all the letters classifications equal to or higher than the threshold and the latter contains the letters classifications with a confidence level lower than the threshold
4) The Mask letters, the mask letter weights, the knowledge, the word classification, the high confidence letters, the low confidence letters, are all passed as input to the SRU which uses them all together to select a specific mask and its weight to apply it on the original world letter patterns to generate new patterns and reclassify them to generate a new list from the knowledge. The cycle repeats until a final conclusion is reached about the classification of the input word letters

3.2.1. SRU

The SRU or the symbolic reasoning unit is divided into two steps. The first one is based on the high confidence letters and the second is based on the low confidence letters. The high confidence letter step finds all the letter classifications in the list with a confidence level higher than the threshold and generates a new list of words. If the list size is one word then this is the final classification for the input word, otherwise the letters with a low confidence level and the new list are passed to the low confidence letters step which in turn uses the weighted masks to generate a new word classification and a new list. If the size of the list is one word then this is the final classification, if not then the strength of the mask is increased and the word classification and the new list is repassed to the low confidence letters step.

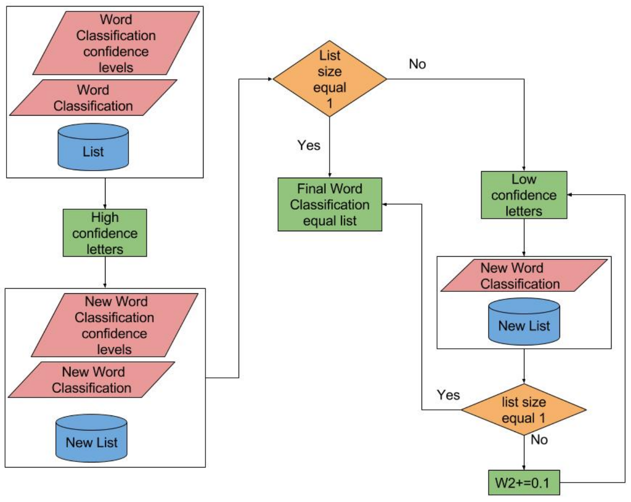
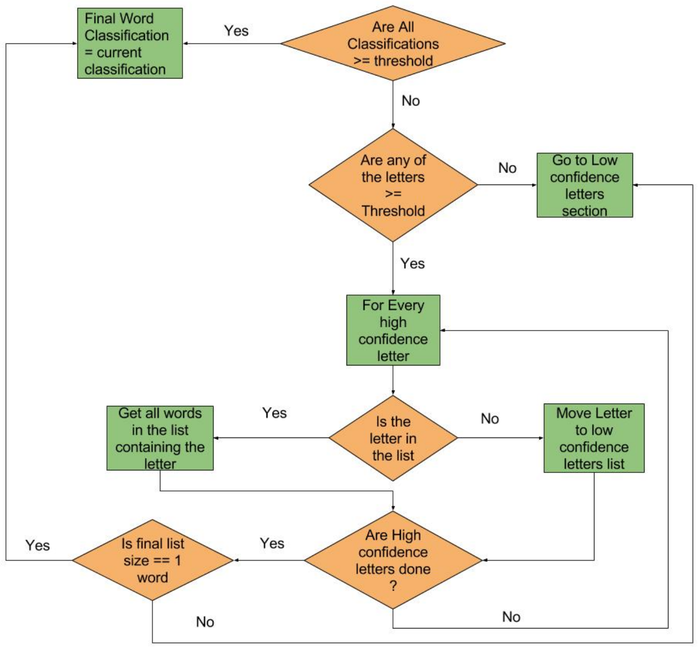


Fig 15, SRU Macro level

3.2.1.1.The high confidence letters

The high confidence letters step applied on every letter with a confidence level higher than the threshold



Final Word Classification = current classification
Yes
Are All Classifications >= threshold
No
Are any of the letters >= Threshold
No
Go to Low confidence letters section
Yes
For Every high confidence letter
Get all words in the list containing the letter
Yes
Is the letter in the list
No
Move Letter to low confidence letters list
Yes
Is final list size == 1 word
Yes
Are High confidence letters done ?
No
No

Fig 16, High confidence letters step

3.2.1.2.The low confidence letters

The Low confidence letters step applied on every letter with a confidence level lower than the threshold can be described as follows

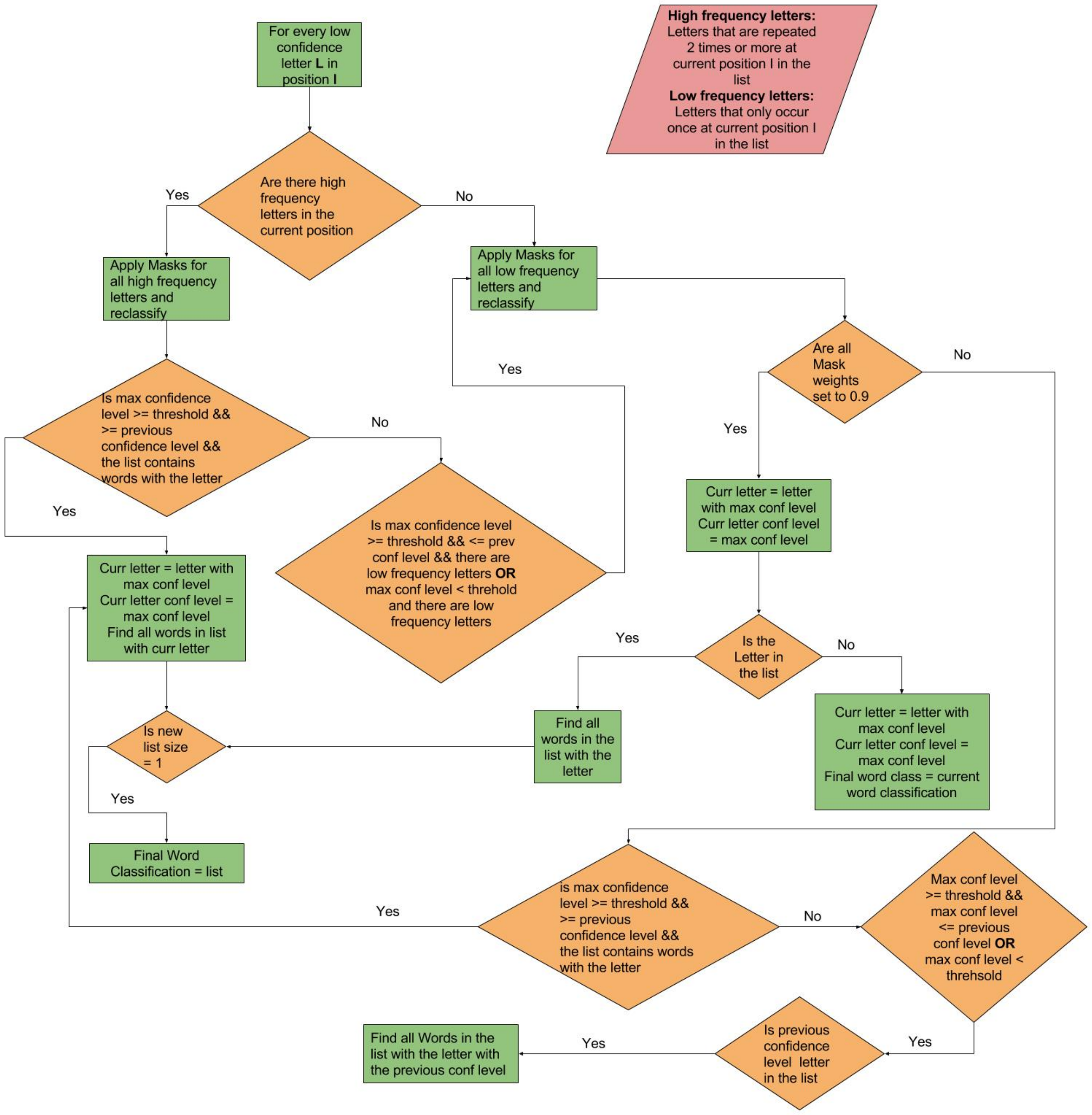

For every low confidence letter **L** in position **I**
High frequency letters: Letters that are repeated 2 times or more at current position I in the list
Low frequency letters: Letters that only occur once at current position I in the list
Are there high frequency letters in the current position
Yes
No
Apply Masks for all high frequency letters and reclassify
Apply Masks for all low frequency letters and reclassify
Yes
Are all Mask weights set to 0.9
No
Yes
Is max confidence level >= threshold && >= previous confidence level && the list contains words with the letter
No
Yes
Curr letter = letter with max conf level Curr letter conf level = max conf level
Is max confidence level >= threshold && <= prev conf level && there are low frequency letters **OR** max conf level < threhold and there are low frequency letters
Curr letter = letter with max conf level Curr letter conf level = max conf level Find all words in list with curr letter
Yes
Is the Letter in the list
No
Find all words in the list with the letter
Curr letter = letter with max conf level Curr letter conf level = max conf level Final word class = current word classification
Is new list size = 1
Yes
Final Word Classification = list
Yes
Is max confidence level >= threshold && >= previous confidence level && the list contains words with the letter
No
Max conf level >= threshold && max conf level <= previous conf level **OR** max conf level < threhsold
Find all Words in the list with the letter with the previous conf level
Yes
Is previous confidence level letter in the list
Yes

Fig 17, Low confidence letters

4. IMPLEMENTATION

The system was implemented on MATLAB using the Neural Network toolbox to create, train and use the neural networks.

4.1.The Feed Forward Neural network

It consists of 3 layers: the first layer has a connection from the network input, the hidden layer has a connection to the previous layer and the final layer produces the networks output. The following steps are taken to create and train the feed forward neural network:

1) Load the dataset
2) Initialize the training set
3) Initialize the test set
4) Initialize the label matrix
5) Initialize the feed forward neural network using the feedforwardnet function while specifying the number of neurons in the hidden layer and the training algorithm
6) set the number of epochs
7) set the regularization parameter to prevent over fitting
8) call the train function with the feed forward neural network, the training set and the label matrix as input

The parameters chosen by which the neural network had the best performance were as follows:

400 neurons in the hidden layer, 1000 epochs, regularization = 0.95 and the Trainscg function or the scaled conjugate gradient back propagation, which performs much faster then the regular back propagation method to train the feed forward neural network **[23].**

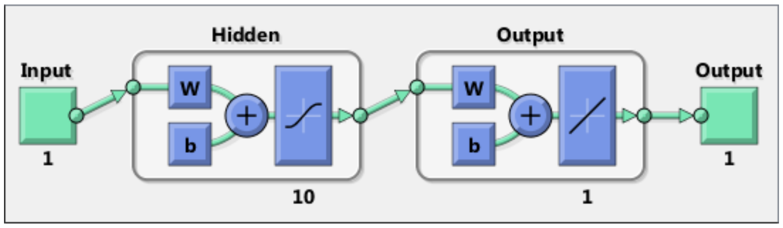
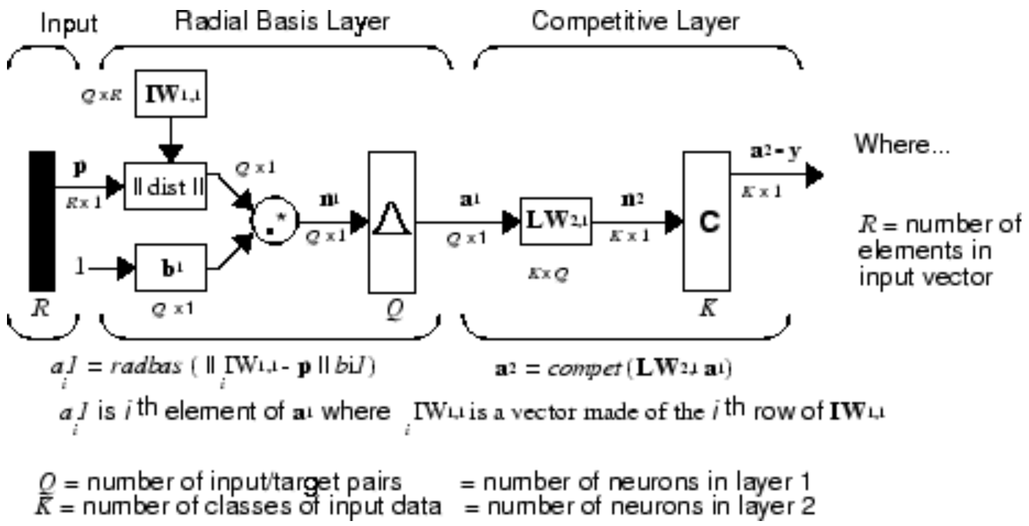


Fig 18, Example Feed forward neural network diagram

4.2.The probabilistic neural network

This neural network consists of 2 layers. When an input is presented, the first layer computes the distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input vector is to the training input. The second layer sums these contributions for each class of inputs to produce as its net output a vector of probabilities. Finally, a complete transfer function on the output of the second layer picks up the maximum of these probabilities, and produces a 1 for that class and 0 for the other classes. The neural network architecture is shown below:



Input
Radial Basis Layer
Competitive Layer
Q x R
IW1,1
p
R x 1
|| dist ||
Q x 1
1
b1
Q x 1
n1
Q x 1
R
Q
a1
Q x 1
LW2,1
K x Q
n2
K x 1
C
K
a2 = y
K x 1
Where...
R = number of elements in input vector
a1i = radbas (|| iIW1,1 - p || b1i)
a2 = compet (LW2,1 a1)
a1i is i th element of a1 where iIW1,1 is a vector made of the i th row of IW1,1
Q = number of input/target pairs = number of neurons in layer 1
K = number of classes of input data = number of neurons in layer 2

Fig 19, Probabilistic network architecture

It is assumed that there are *Q* input vector/target vector pairs. Each target vector has K elements. One of these elements is 1 and the rest are 0. Thus, each input vector is associated with one of K classes. The first-layer input weights, $IW_{1,1}$ are set to the transpose of the matrix formed from the *Q* training pairs, P. When an input is presented, the || dist || box produces a vector whose elements indicate how close the input is to the vectors of the training set. These elements are multiplied, element by element, by the bias and sent to the radbas transfer function. An input vector close to a training vector is represented by a number close to 1 in the output vector a1. If an input is close to several training vectors of a single class, it is represented by several elements of a1 that are close to 1. The second-layer weights, $LW_{1,2}$, are set to the matrix T of target vectors. Each vector has a 1 only in the row associated with that particular class of input, and 0s elsewhere. The multiplication Ta1 sums the elements of a1 due to each of the K input classes. Finally, the second-layer transfer function, compet, produces a 1 corresponding to the largest element of n2, and 0s elsewhere. Thus, the network classifies the input vector into a specific K class because that class has the maximum probability of being correct.

The parameters chosen to create the probabilistic neural network are as follows:

1) The $IW_{1,1}$ matrix is a 26x5 matrix where each column vector represents a letter in the alphabet
2) The $LW_{2,1}$ matrix 26x26 target matrix where the ith column vector has the ith element set to 1 and the rest of the elements to 0
3) The transfer function is changed from compete to linear to give as output the confidence levels which are to be used by the SRU to decide the class of the incoming input letter.

5. DATASET

The dataset used to test the proposed system is the Chars74K dataset [8]. It consists of 64 classes of English characters, i.e. numbers from 0 to 9, uppercase letters from A-Z and lower case letters from a-z. The dataset contains 7705 characters obtained from natural images, 3410 hand drawn characters using a tablet PC and 62992 synthesized characters from computer fonts which gives a total of 74K images. For the sake of this paper, the main interest is on the 3410 characters drawn on a tablet PC. Those 3410 characters are divided into 55 sample PNG binary images for every character class. The preprocessed dataset, post processed dataset for the letter 'A' for example are shown below in figure 20 and 21, the post processed dataset of 55 letters is divided at a ratio of 50:50 where 27 letters are used for the training set and for creating the mask and 28 letters are used for the test set. The masks generated for the upper case letters and the lower case letters are shown in figures 22 and 23 and An example of a word made from the preprocessed letters is shown in figure 24.

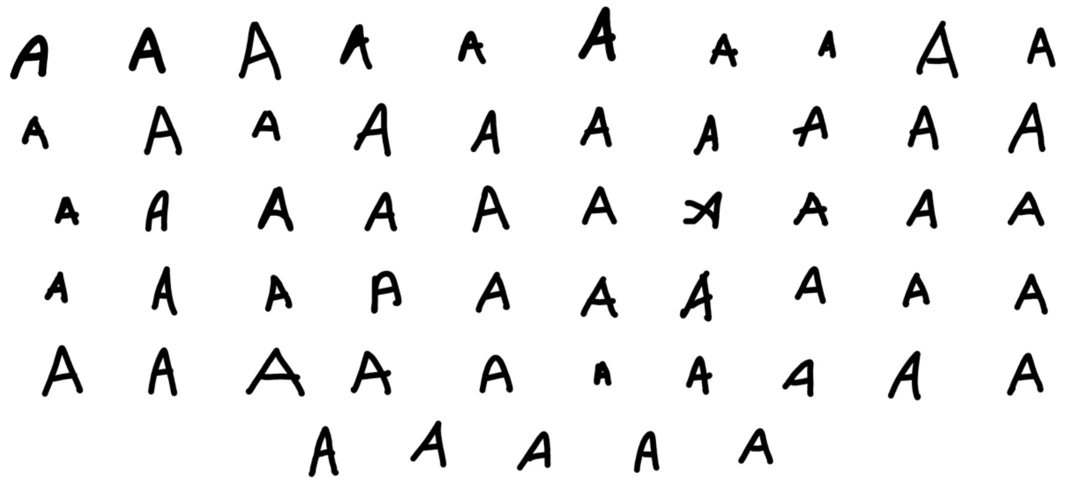


Fig 20, Preprocessed Dataset for the letter 'A'

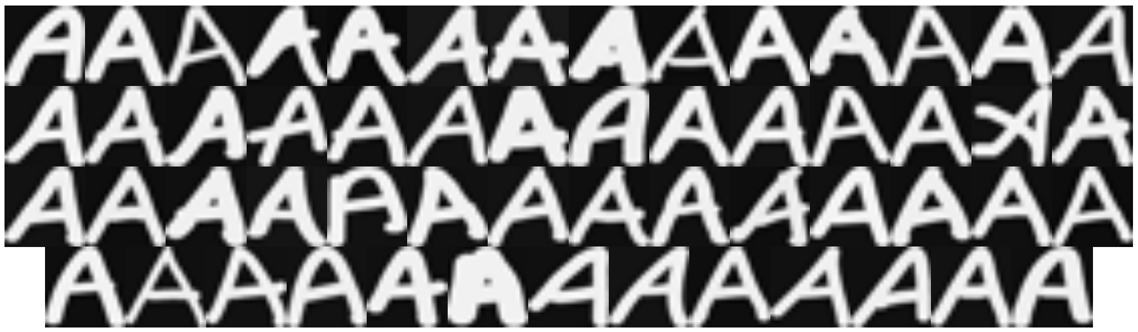


Fig 21, Post Processed Dataset for Letter 'A'

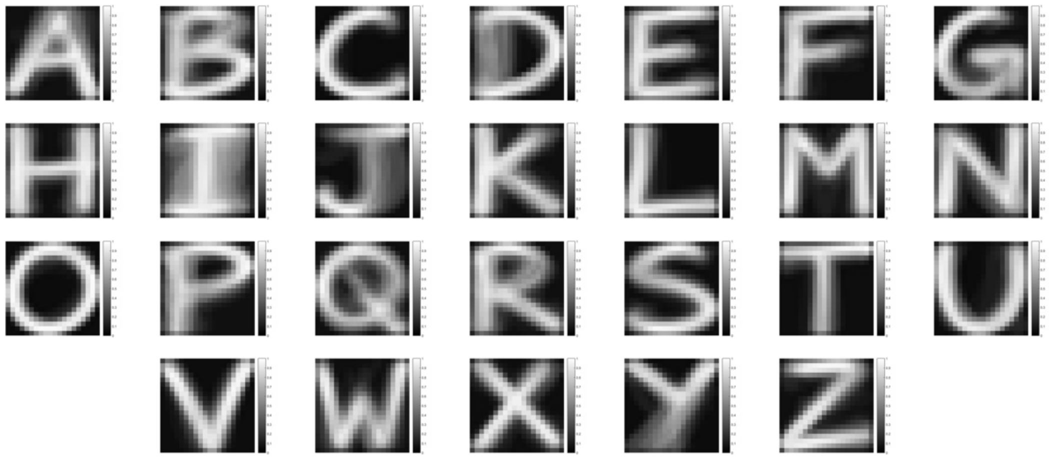


Fig 22, Uppercase Letters Masks

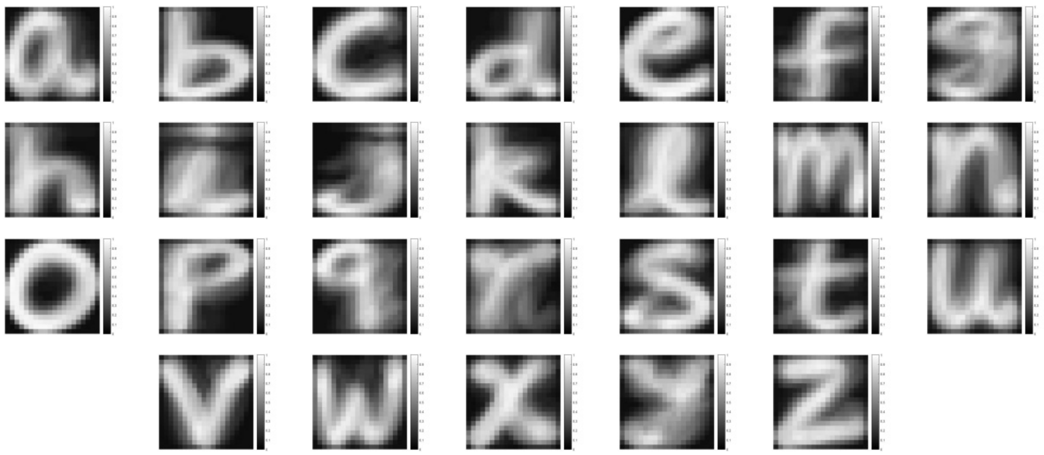


Fig 23, Lowercase Letters Masks

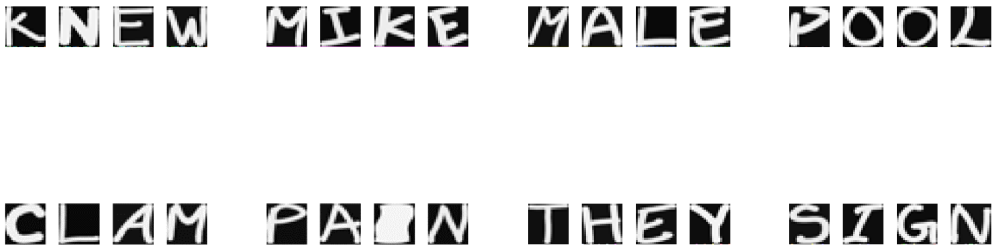


Fig 24, sample words used in tests

6. Tests and Results

Testing the proposed system requires applying it on upper case letters and lower case letters and applying three different tests on randomly chosen words with randomly chosen handwritten patterns for every letter in these words: they are the neural network test, the neural network dictionary test and the neural network symbolic reasoning unit test. These tests can be described as follows:

6.1.The neural network test

The purpose of this test is to assess the performance of the neural network on a set of words, the steps of this test are as follows:

1) Generate an n random words list
2) Pass all the words to the neural network
3) Compare the words classifications to the original list
4) Success rate = (number of words classified correctly / length of the list) x 100

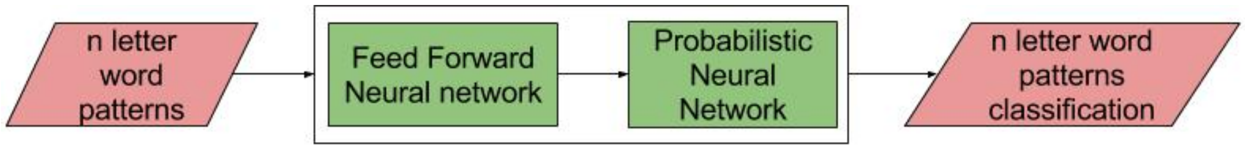

n letter word patterns
Feed Forward Neural network
Probabilistic Neural Network
n letter word patterns classification

Fig 25, The Neural Network Test

6.2.The neural network dictionary test

This test uses the dictionary to determine the class of the words after they are classified by the neural network based on the confidence level generated by the neural network on these words and thus helps improve the classifications in the neural network test. The steps of this test are as follows:

1) Generate an n random words list
2) Pass all the words to the neural network to get the words classifications and confidence levels
3) Identify the high confidence letters classifications which are the word letter classifications with a confidence level equal or greater than a threshold
4) If the word letters classification is equal to the original word, then the success rate for the current word equals 1
5) If there are no high confidence letters classifications for the word, then the success rate for this word equals 0
6) Other wise find all words with high confidence letters in the dictionary
7) If n words are found, then the success rate for the current word equals 1 / n
8) Total success rate = ((Σword success rate) / n) * 100

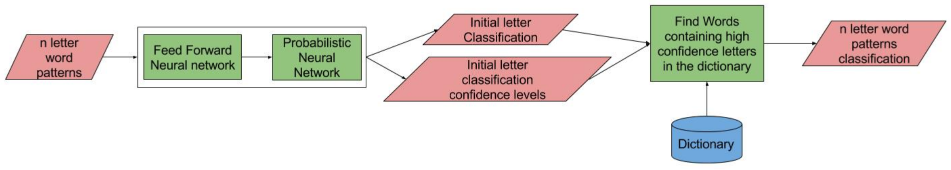

n letter word patterns
Feed Forward Neural network
Probabilistic Neural Network
Initial letter Classification
Initial letter classification confidence levels
Find Words containing high confidence letters in the dictionary
Dictionary
n letter word patterns classification

Fig 26, The Neural Network dictionary test

6.3.The neural network Symbolic reasoning unit test

The last test as shown previously in figure 13, uses the neural network, the dictionary and the SRU to come up with a final classification for the incoming word based on the confidence level of the initial classification done by the neural network and thus using contextual knowledge and expectation to determine the final classification for the input and thus improve the results of the above 2 tests.

6.4.Test sets

The test sets used to test the performance of the system are the 100 words test set and the dictionary test set. In the former, 100 random words are chosen from the dictionary and applied on the above tests. While in the latter, the dictionary itself is used as the test set. The results of the two tests are as follows:

6.4.1. The 100 words test

100 words are randomly chosen from the dictionary, the letters of each word are randomly chosen from the test set and the non-mask hit rate is set at 100%, 95%, 90%, 85% and 80%. The three above tests are applied on uppercase 4 letter words, uppercase 6 letter words, lowercase 4 letter words and lowercase 6 letter words, the results are as follows:

	UpperCase_4	UpperCase_6
NN	41%	35%
NN + Dict	56%	60%
SRU_0.6-0.9	70%	78%
SRU_100%	66%	72%
SRU_95%	67%	79%
SRU_90%	68%	79%
SRU_85%	68%	79%
SRU_80%	69%	78%
	LowerCase_4	**LowerCase_6**
NN	23%	20%
NN + Dict	29%	32%
SRU_0.6-0.9	43%	52%
SRU_100%	41%	47%
SRU_95%	42%	51%
SRU_90%	43%	51%
SRU_85%	44%	53%
SRU_80%	44%	51%

Table 1, 100 words test

6.4.2. The dictionary test

The dictionary size available for 4 letter words is 5526 words and the dictionary size available for the 6 letter words is 15788 words. The three tests are applied on uppercase 4 letter word dictionary 1st half, uppercase 4 letter word dictionary 2nd half, uppercase 4 letter word full dictionary, with 20% of the words being not available in the dictionary, uppercase 6 letter word dictionary 1st half, uppercase 6 letter word dictionary 2nd half, uppercase 6 letter word full dictionary with 20% of the words being not available in the dictionary and the non-mask hit rate is chosen at 85%. Where the letters of each word in those datasets are randomly chosen from the test set, the results are as follows:

	UpperCase_4_Dict_1st half	**UpperCase_4_Dict_2nd half**	**UpperCase_4_all Dict_20% unknown**
NN	47%	47%	46%
NN + Dict	58%	57%	56%
SRU_85%	71%	73%	62%
	UpperCase_6_Dict_1st half	**UpperCase_6_Dict_2nd half**	**UpperCase_6_all Dict_20 unknown**
NN	29%	30%	31%
NN + Dict	57%	57%	54%
SRU_85%	78%	79%	65%
	LowerCase_4_Dict_1st half	**LowerCase_4_Dict_2nd half**	**LowerCase_4_all Dict_20 unknown**
NN	29%	25%	27%
NN + Dict	35%	31%	33%
SRU_85%	49%	47%	41%
	LowerCase_6_Dict_1st half	**LowerCase_6_Dict_2nd half**	**LowerCase_6_all Dict_20 unknown**
NN	14%	12%	14%
NN + Dict	29%	24.60%	27%
SRU_85%	52%	49%	42%

Table 2, Dictionary test

7. ANALYSIS

After examining the above results, it appears that the recognition of the words is divided into 4 categories including, words classified correctly by the neural network, words incorrectly classified by the neural network but corrected with the dictionary, words incorrectly classified by the neural network and the dictionary but corrected by the SRU and words not included in the dictionary, the analysis of these categories is described below

7.1.Correct classification with the neural network

This is the case where the neural network classifies the words letters correctly without the use of the dictionary and without taking into consideration the confidence level of the letters classifications. The idea here is that the words letter patterns in question are close enough to the training set letter patterns to give a correct classification. Examples of correctly classified words by the neural network are shown below:

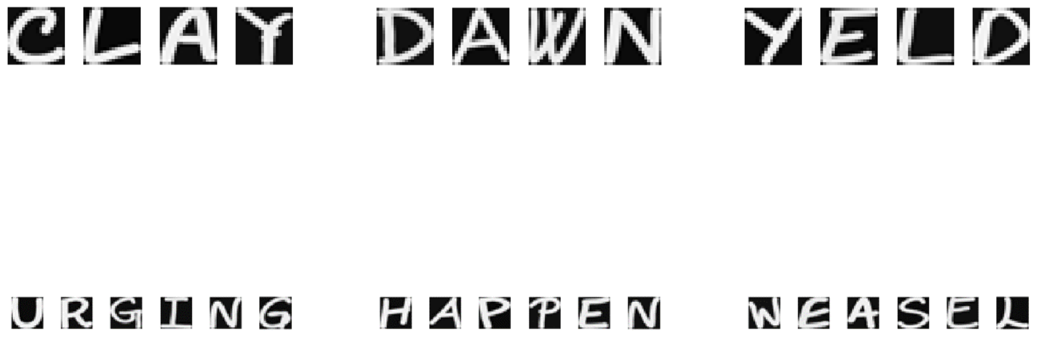
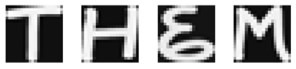


Fig 27, neural network test correct classifications

7.2.Incorrect classification with neural network and corrected with dictionary

In this case, the system aims to correctly classify the missed words from the neural network by searching for the high confidence letters with confidence levels equal to or greater than 95% in the dictionary and therefore giving the most probable classification in the dictionary for the current word. Examples of this procedure are shown below:



```
classification result = T H C M
confidence levels = 0.9964 0.9959 0.8966 0.9892
high_conf_letters = T H M
low_conf_letters = C
353 Words found for Letter T in position 1
22 Words found for Letter H in position 2
1 Words found for Letter M in position 4 = Them
Word_SuccessRate = 1
```

```
classification result = R P Y G
confidence levels = 0.8117 0.9952 0.9080 0.9839
high_conf_letters = P G
low_conf_letters = Y R
61 Words found for Letter P in position 2
2 Words found for Letter G in position 4
Word_SuccessRate = 0.5
```

Fig 28, Neural Network Dictionary test correct classifications

7.3. Incorrect classification with neural network and dictionary and corrected with the SRU

In this case, the proposed SRU system helps correct the misclassification done by the neural network and the dictionary, where if the classification of the word in question is incorrect by the neural network system and the neural network dictionary system, it is corrected by the neural network SRU system. Examples of these corrections are shown below:

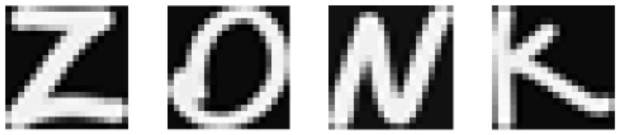


Fig 29, Neural network SRU Example 1

classification result = **Z, N, N, K**
confidence levels = **0.9918, 0.8666, 0.9845, 0.9057**
High confidence letters = **Z, N**
Low confidence letters = **K, N**
56 Words found for Letter **Z** in position **1**
8 Words found for Letter **N** in position **3** = **ZANY, ZINC, ZINE, ZING, ZINS, ZONA, ZONE, ZONK**
for Position **4**, Letters With **Highest Frequency** In Pos = **E**
Letters_With_**Lowest_Frequency**_In_Pos = **A, C, G, K, S, Y**

Apply mask for High frequency letters

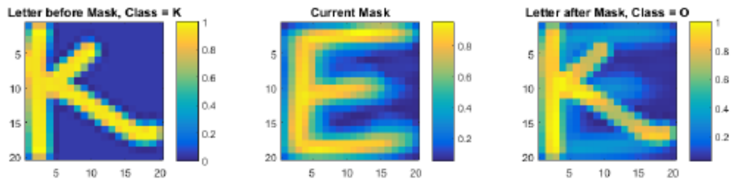


Fig 30, Apply mask for high frequency letters Example 1

High frequency letter confidence level after mask application = **0.8368**

Apply Mask for low frequency letters

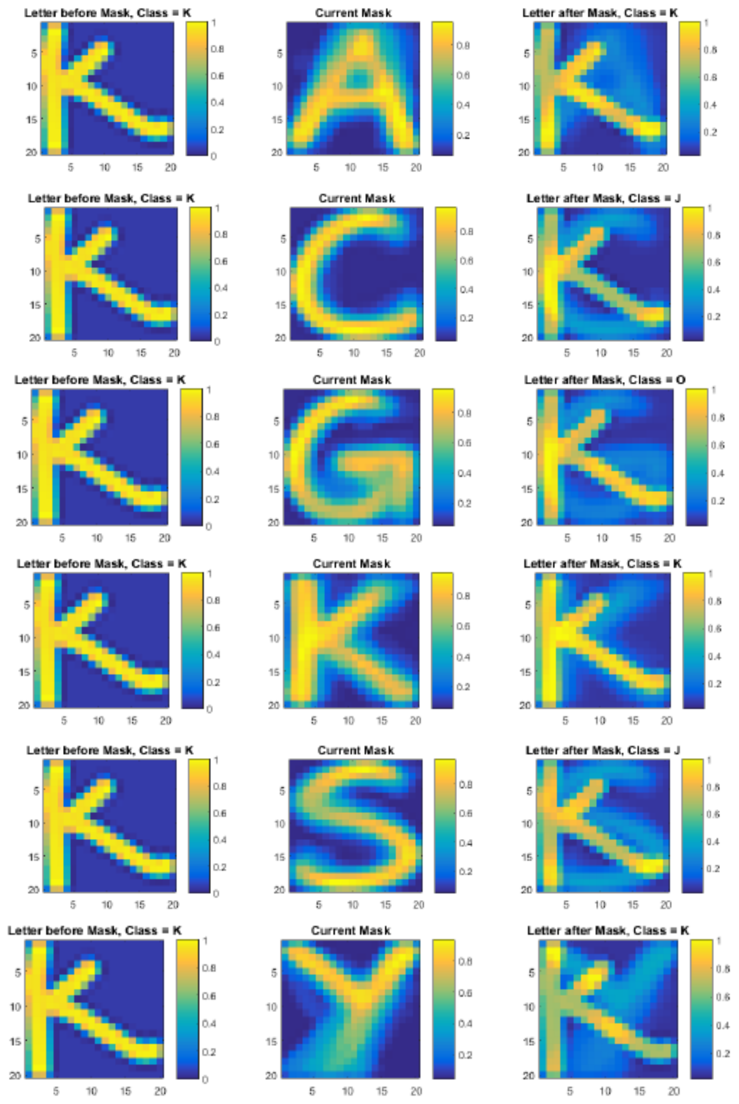


Fig 31, Apply masks for low frequency letters Example 1

Low frequency classes after mask application = **K, J, O, K, J, K**
Low frequency letters confidence levels after mask application = **0.9340, 0.8140, 0.8317, 0.9738, 0.8286, 0.8781**
1 word found for letter K in position 4 = **ZONK**
Final classification from SRU = **ZONK**

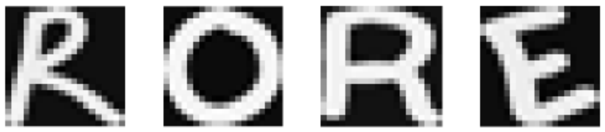


Fig 32, Neural network SRU Example 2

classification result = **R, O, R, Q**
confidence levels = **0.9503, 0.9991, 0.9674, 0.8531**
High confidence letters = **O, R, R**
Low confidence letters = **Q**
1014 words found for letter **O** in position **2**
98 words found for letter **R** in position **3**
3 words found for letter **R** in positon **1 = RORE, RORT, RORY**
for Position **4**, Letters With **Highest Frequency** In Pos = **[ ]**
Letters_With_**lowest_frequency**_In_Pos = **E, T, Y**

Apply masks for low frequency letters

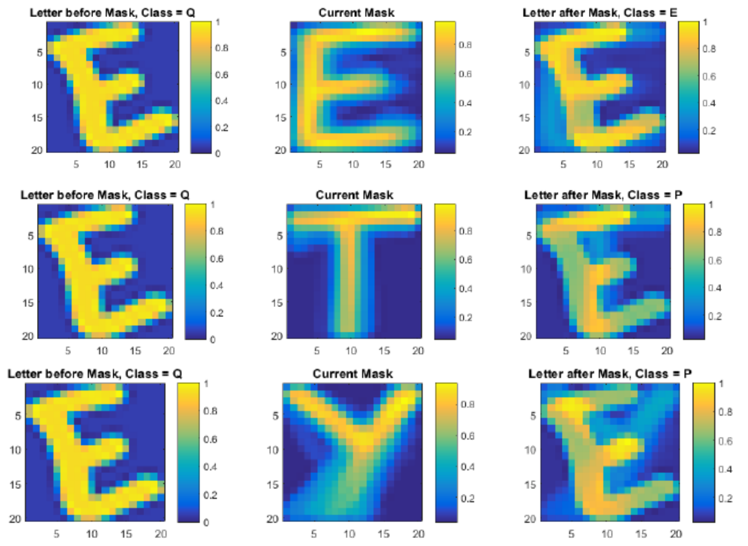


Fig 33, Apply masks for low frequency letters Example 2

Low frequency letters classes after mask application **= E, P, P**
Low frequency letters confidence levels after mask application = **0.8687, 0.8499, 0.8483**

Increase W2 by 0.1 until the classification confidence levels are higher than the threshold and the previous confidence level, if W2 is 0.9 and none of the confidence levels is higher than the threshold and the previous confidence level then choose the classification with the maximum confidence level

Apply masks for low frequency letters at w2 = 0.9

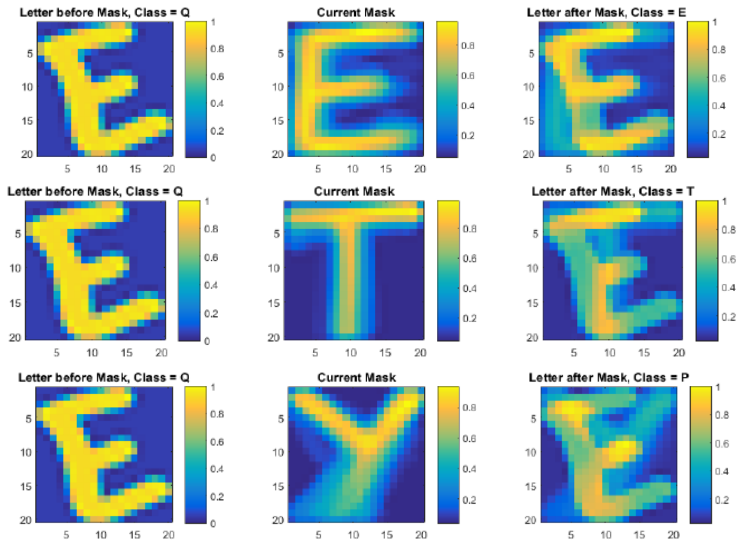


Fig 34, Apply masks for low frequency letters

Low frequency letters classes after mask application **= E, T, P**
Low frequency letters confidence levels after mask application = **0.9291, 0.8827, 0.8436**
1 word found for letter E in position 4 = **RORE**
Final Classification from the SRU **= RORE**

7.4. Words not included in the dictionary

This case shows what happens when the words in question are not in the dictionary.

Examples of the outcomes of this case are shown below:

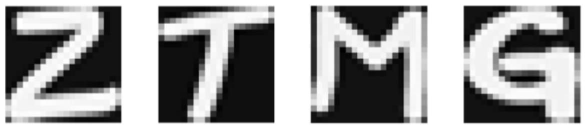


Fig 35, Words not included in the dictionary Example 1

Final classification from the neural network = ZTMG

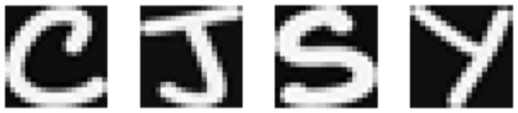


Fig 36, Words not included in the dictionary Example 2

classification result = **C, S, S, Y**
confidence levels = **0.9847, 0.7941, 0.9253, 0.9868**
High confidence letters = **Y, C**
Low confidence letters = **S, S**
267 Words found for Letter **Y** in position **4**
18 Words found for Letter **C** in position **1** = **CAGY, CAKY, CANY, CAVY, CHAY, CITY, CLAY, CLOY, COKY, COLY, CONY, COPY, CORY, COSY, COWY, COXY, COZY, CRAY**
for Position **3**, Letters With **Highest Frequency** In Pos = **A**
Letters_With_**Lowest_Frequency**_In_Pos = **G, K, L, N, O, P, R, S, T, V, W, X, Z**

Apply mask for high frequency letters

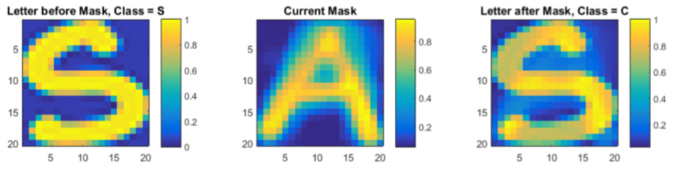


Fig 37, Apply masks for high frequency letters

High frequency letters after mask classes = **C**
High frequency letters after mask confidence levels = **0.9310**
Apply masks for low frequency letters
Low frequency letters classes after mask application = **C, C, A, C, C, C, C, S, S, S, C, S, S**
Low frequency letters confidence levels after mask application = **0.9230, 0.8988, 0.8863, 0.9023, 0.8973, 0.8765, 0.8941, 0.9694, 0.8779, 0.8849, 0.9292, 0.8994, 0.9217**
1 word found for letter **S** in position **3** = **COSY**
Final classification from SRU = **COSY**

The 100-word test and the dictionary test suggest the following about the proposed system:

1) The neural network SRU system almost doubles the success rate of the neural network system on an n number of randomly chosen words with randomly chosen handwritten character patterns for every character in those words.
2) The initial success rate of the neural networks is highly affected by the performance of the neural network on the character dataset which consequently affects the word success rate based on the length of the words; the success rate of the neural network on the handwritten characters test set was 82% for the uppercase letters dataset and 72% for the lowercase letters dataset.
3) The success rate of the neural network system and the neural network and dictionary system is inversely proportional to the length of word in question, however the SRU significantly increases the chances of getting the correct classification.
4) The SRU highly depends on the knowledge in the dictionary, i.e., the more words available in the dictionary, the higher are the chances of achieving the correct classification of the word in question which is evident by the fact that a word not in the dictionary has a high probability of not being classified correctly unless the distance between the word letters patterns and the training set is short enough to be classified correctly by the neural network.

8. CONCLUSION

In this thesis, it is proven that the application of contextual knowledge and expectation in neural network based pattern recognition systems in the application of offline hand written character recognition based on the concept of ‘you see what you expect to see’ can improve the success rate of neural networks. However, additional improvements can be done to increase the overall performance of the system. Training the neural network on a much bigger dataset will improve its classification confidence levels and thus increase the probability of getting a correct classification by the symbolic reasoning unit. In addition, dynamically saving unknown words in the dictionary will increase the probability of correctly classifying those words the next time they are applied to the system. In addition, the proposed system can be expanded to more pattern recognition applications. For example, in face recognition where the neural network can be trained on different facial patterns and then by the use of contextual knowledge about the patterns, the system can improve the neural network’s classification on new patterns.

9. REFERENCES

1) Impedovo, Sebastiano, ed. *Fundamentals in handwriting recognition*. Vol. 124. Springer Science & Business Media, 2012.

2) Ooi, Shih Yin, et al. "Image-based handwritten signature verification using hybrid methods of discrete Radon transform, principal component analysis and probabilistic neural network." *Applied Soft Computing* 40 (2016): 274-282.

3) Wanchoo, Ankita S., Preeti Yadav, and Alwin Anuse. "A Survey on Devanagari Character Recognition for Indian Postal System Automation."*International Journal of Applied Engineering Research* 11.6 (2016): 4529-4536.

4) Jayadevan, R., et al. "Automatic processing of handwritten bank cheque images: a survey." *International Journal on Document Analysis and Recognition (IJDAR)* 15.4 (2012): 267-296.

5) Halder, Chayan, Sk Md Obaidullah, and Kaushik Roy. "Offline Writer Identification and Verification—A State-of-the-Art." *Information Systems Design and Intelligent Applications*. Springer India, 2016. 153-163.

6) Dash, Tirtharaj, and Tanistha Nayak. "English Character Recognition using Artificial Neural Network." *arXiv preprint arXiv:1306.4621* (2013).

7) Prasad, Kauleshwar, et al. "Character Recognition Using Matlab" s Neural Network Toolbox." *International Journal of u-and e-Service, Science and Technology* 6.1 (2013): 13-20.

8) de Campos, Teófilo Emídio, Bodla Rakesh Babu, and Manik Varma. "Character Recognition in Natural Images." *VISAPP (2)*. 2009.

9) Alginahi, Yasser. *Preprocessing techniques in character recognition*. INTECH Open Access Publisher, 2010.

10) Chan, Raymond H., Chung-Wa Ho, and Mila Nikolova. "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization." *Image Processing, IEEE Transactions on* 14.10 (2005): 1479-1485.

11) Kumar, Deepak, and Dalwinder Singh. "Modified approach of hough transform for skew detection and correction in documented images."*International Journal of Research in Computer Science* 2.3 (2012): 37.

12) Casey, Richard G., and Eric Lecolinet. "A survey of methods and strategies in character segmentation." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.7 (1996): 690-706.

13) Pradeep, J., E. Srinivasan, and S. Himavathi. "Diagonal based feature extraction for handwritten alphabets recognition system using neural network." *arXiv preprint arXiv:1103.0365* (2011).

14) Rajbala Tokas,Aruna Bhadu, "A comparative analysis of feature extraction techniques for handwritten character recognition", International Journal of Advanced Technology & Engineering Research, Volume 2, Issue 4, pp. 215-219, July 2012

15) Cheriet, Mohamed, et al. *Character recognition systems: a guide for students and practitioners*. John Wiley & Sons, 2007.

16) Sahu, Vijay Laxmi, and Babita Kubde. "Offline Handwritten Character Recognition Techniques using Neural Network: A Review." *International journal of science and Research (IJSR)* 2.1 (2013): 87-94.

17) Pandey, Abhinav, Shashank Sawant, and Eric M. Schwartz. "Handwritten Character Recognition using Template Matching." APA

18) del Bimbo, Alberto, Stefania Santini, and Jose Sanz. "OCR from poor quality images by deformation of elastic templates." *Pattern Recognition, 1994. Vol. 2-Conference B: Computer Vision &amp; Image Processing., Proceedings of the 12th IAPR International. Conference on*. Vol. 2. IEEE, 1994.

19) Shah, Mansi, and Gordhan B. Jethava. "A literature review on hand written character recognition." *Indian streams research journal* 3.2 (2013): 1-19.

20) Kumar, Manoj, and Sandeep Chandran. "Handwritten Malayalam Word Recognition System using Neural Networks." International Journal of Engineering Research and Technology. Vol. 4. No. 04 (April-2015). ESRSA Publications, 2015.

21) Lawson, Anton E. "The Neurological Basis of Self-Regulation." *The Neurological Basis of Learning, Development and Discovery: Implications for Science and Mathematics Instruction* (2003): 27-55.

22) Maren, Alianna J., Craig T. Harston, and Robert M. Pap. *Handbook of neural computing applications*. Academic Press, 2014.

23) Møller, Martin Fodslette. "A scaled conjugate gradient algorithm for fast supervised learning." *Neural networks* 6.4 (1993): 525-533.

24) Johnson, Rebecca L., and Morgan E. Eisler. "The importance of the first and last letter in words during sentence reading." Acta psychologica 141.3 (2012): 336-351.

25) Perea, Manuel, and Stephen J. Lupker. "Can CANISO activate CASINO? Transposed-letter similarity effects with nonadjacent letter positions." Journal of memory and language 51.2 (2004): 231-246.

26) Fischer, Andreas, et al. "A combined system for text line extraction and handwriting recognition in historical documents." Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on. IEEE, 2014.

27) Hirwani, Amrita, and Sandeep Gonnade. "Handwritten Character Recognition System Using Neural Network." International Journal 2.2 (2014).

28) Hentschel, Jeff, and Justin Li. "Using Neural Nets to Recognize Handwriting."

29) Espana-Boquera, Salvador, et al. "Improving offline handwritten text recognition with hybrid HMM/ANN models." IEEE transactions on pattern analysis and machine intelligence 33.4 (2011): 767-779.

30) Frinken, Volkmar, et al. "A novel word spotting method based on recurrent neural networks." IEEE transactions on pattern analysis and machine intelligence 34.2 (2012): 211-224.

31) Meier, Ueli, et al. "Better digit recognition with a committee of simple neural nets." 2011 International conference on document analysis and recognition. IEEE, 2011.

32) Chen, Li, et al. "Beyond human recognition: A CNN-based framework for handwritten character recognition." 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR). IEEE, 2015.

33) Acharya, Shailesh, Ashok Kumar Pant, and Prashnna Kumar Gyawali. "Deep learning based large scale handwritten Devanagari character recognition." 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA). IEEE, 2015.

34) Maitra, Durjoy Sen, Ujjwal Bhattacharya, and Swapan K. Parui. "CNN based common approach to handwritten character recognition of multiple scripts." Document Analysis and Recognition (ICDAR), 2015 13th International Conference on. IEEE, 2015.