Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 11 (1): 86-95, 2011
ISSN 1812-5654 / DOI: 10.3923/jas.2011.86.95

Offline Malay Handwritten Cheque Words Recognition using Artificial Neural Network

Omar Noori, Sharifah Mumtazah Syed Ahmad and Asma Shakil
Faculty of Information Technology, University Tenaga National,
43009, Km7 Jalan Kajang-Pochong, Malaysia

Abstract: The improvement of automated systems that are capable of recognizing human handwritings offers a new way of improving human-computer interface and of enabling computers to process handwritten documents more efficiently. In order to implement a high accuracy approach for handwriting recognition system, we need several pre-processing steps for the purpose of preparing the data for the feature extraction stage. There are many pre-processing steps used in this approach to enhance the perspective of our dataset, such as noise removal, image normalization and skeletonization. Two types of feature extraction techniques have been applied in this approach statistical and geometrical. Our experiments show that the statistical feature is reliable, accessible and offers more accurate results. An adaptive multi-layer feed-forward back-propagation neural network was used in our research. This study focuses only on Malay handwritten cheque words recognition using lexical matching on top of character recognition. The presented results show that our approach has successfully increased the accuracy of recognition from 98.15% by using pure character recognition to over 99% by using the newly proposed hybrid character and lexical cheque words recognition.

Key words: Offline handwriting recognition, back-propagation neural network, pattern recognition, character recognition, lexical verification, fed forward neural network, automated cheque processing

INTRODUCTION

Off-line Handwriting Recognition System remains an important and one of the challenging research areas to date. This is mainly due to the rising need for the automation of the off-line handwriting recognition, especially in applications that require huge amounts of handwritten paper documents to be processed and recognized daily. For example, batch processing tasks such as sorting out postal letters and documents, as well as clearing of bank cheques often necessitate that the handwritten postcodes and cheque details be recognized automatically and immediately. The challenges, meanwhile, lie mainly in recognizing human handwriting, which exhibits high level of handwriting variability that makes building an accurate and robust handwriting recognition system somewhat difficult. There are two main approaches in character recognition (Razak *et al*., 2009) and word recognition. In our system, we managed to make the benefit of character recognition system towards word recognition by building a Malay cheque words dictionary to test the lexical variances that could sometimes be found due to the error in the character recognition system or due to writer misprint some other times (Park and Govindaraju, 2000; Carbonnel and Anquetil, 2003) database will be depicted later in this study. This useful post-processing raises the whole recognition accuracy as illustrated in the flowchart in Fig. 1.

Park and Govindaraju (2000) had designed an offline handwritten word recognition, which is based on two ideas: an adaptive character recognition method and a recursive word recognition mechanism. That experiment shows that the selection of features improves recognition rate, using the same number of features as compared to a static model. This system is dependent on lexical similarity in character recognition, as well as word recognition decision. Meanwhile, in our system, we have employed the ANN to make character recognition before using lexical verification in word prediction and decision.

In the experiments, Gunter and Bunke (2005) had indicated that the ensemble methods considered in the paper were useful to improve classification accuracy in the application of handwritten word recognition. They summarized that by using both optimized base classifiers and enhanced ensemble methods, the recognition rate was far above 80% for large vocabularies used in the study.

Corresponding Author: Omar Noori, Faculty of Information Technology, University Tenaga National, 43009, Kajang, Malaysia
Tel: +060172046780

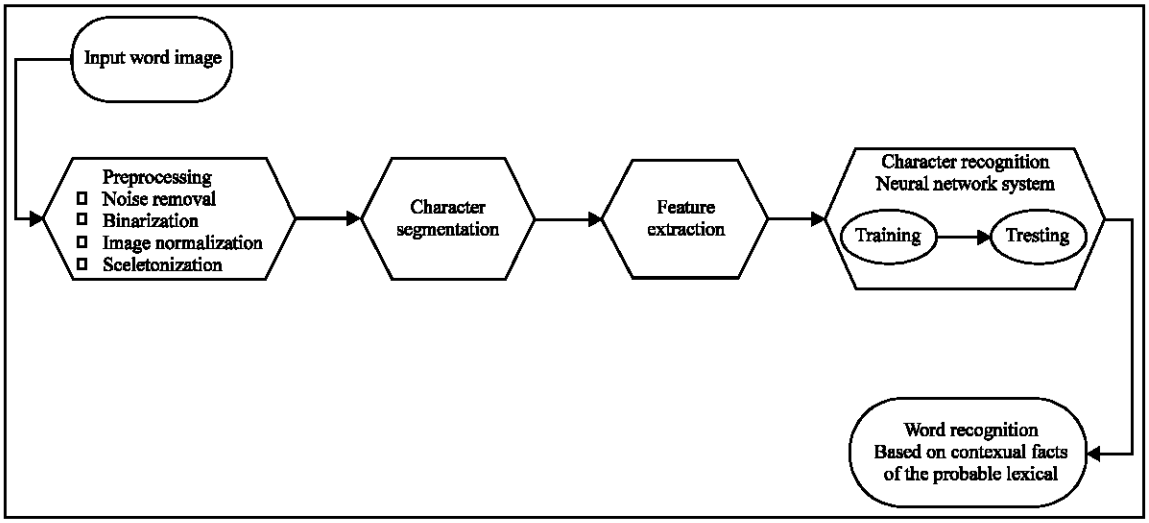


Fig. 1: System overview

Araokar (2005) had proposed an approach for the recognition of visual (optical) characters Optical Character Recognition (OCR) using artificial neural networks by applying a few preprocessing steps and feed the diminished image array to simple neural network system.

The same strategy of Neural Network for OCR was used with some changes in the system proposed by Gorgel and Oztas (2007) with the recognition rate of 83%. Most of the OCR techniques use very few characteristics and features from the original character image due to the shrinking process that they apply to it. Contrary to that, our system produces more feature extraction methods close to the next summarized system.

Tay (2002) had designed an offline handwritten word recognition system based on the hybrid of Artificial Neural Network (ANN) and Hidden Markov Model (HMM) and applied it on different data bases, giving it a 97.3% recognition rate. He also proposed many geometrical and statistical feature extracting techniques, which raised the overall recognition rate.

Huang and Srihari (2008) described an approach to separate a line of unconstrained handwriting. They proposed an average distance computed using three different methods. The system is evaluated using an unconstrained handwriting database, which contains 50 pages (1026 lines, 7562 words/images) of handwritten documents. The overall accuracy of this system is 90.82%, indicating a better performance compared to the previously used methods.

Ahmad *et al.* (2002) had used a neural network recognition system for numeral recognition in Malaysian cheques. His system is comprised of three main modules; detection module, extraction module and recognition module. Each module has several sub-modules that perform specific tasks to complete the objectives of that module, giving up to 98.6% rate of success for the handwritten numerals. Meanwhile, for our system, we had used the neural network for the sake of character recognition.

Table 1: Lexicon for Malay cheque words (Sharifah and Ahmad, 2007)

---------- Numbers ----------			Connectives
Satu	Lapan	Ratus	Dan
dua	Sembilan	Seribu	Sahaja
Tiga	Sepuluh	Ribu	Shj
Empat	Sebelas	sejuta	Ringgit
Lima	Belas	Juta	Sen
Enam	Puluh	Billion	Rgt
Tujuh	Seratus		

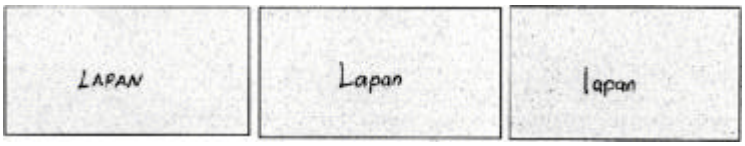
DATA BASE

The database used in this paper contains 26 different Malay Cheque Words (MCW) as shown in Table 1.

The database was compiled from words written by 310 different users, with each user providing a total of 26 words written in three different styles, i.e., words comprising of all small letters, all capital letters and capital letter for the first character followed by small letters as shown in Fig. 2. This combination offers a total of 23,400 words in the database, which was collected by our research team (Sharifah and Ahmad, 2007).

ARTIFICIAL NEURAL NETWORK (ANN)

Feed-forward Neural Network using a multi-layer back-propagation has been trained in this project. Three layers, starting with the input vector values, are derived



Word 8: LAPAN/Lapan/lapan Word 8: LAPAN/Lapan/lapan Word 8: LAPAN/Lapan/lapan

Fig. 2: Data base styles

from measurements of the selected features and bounded between 1 and 17. During learning, an input vector is propagated from the input layer to the output layer. In the learning phase, the learning rate α is preset and the weights start from zero.

The output of the hidden layer is as follows:

$$z_j = f(z\_in_j) \quad (1)$$

$$z\_in_j = \sum_{i=1}^{n} x_i v_{ij} \quad (2)$$

and the output layer will yielded:

$$y_k = f(y\_in_k) \quad (3)$$

$$y\_in_k = \sum_{j=1}^{p} z_j w_{jk} \quad (4)$$

where, normally f(x) is a sigmoid function as follows:

$$f(x) = \frac{1}{1+\exp(-x)} \quad (5)$$

The algorithm follows such that:

$$\Delta w_{jk} = \alpha \delta_k z_j \quad (6)$$

Where:

$$\delta_k = (t_k - y_k) f'(y\_in_k) \quad (7)$$

and

$$\Delta v_{ij} = \alpha \delta_j x_i \quad (8)$$

Where:

$$\delta_j = \sum_{k=1}^{m} \delta_k w_{jk} f'(z\_in_j) \quad (9)$$

where, w is the layer connections weights.

Generally, there are two sets of weights; input-hidden layer weights and hidden-output layer weights. These weights represent the memory of the neural network, where final training weights can be used when running the network.

Initial weights are generated randomly thereafter and weights are updated using the error (difference) between the actual output of the network and the desired (target) output. Weight updating occurs with each iteration and the network learns while iterating repeatedly until a net minimum error value is achieved.

Inputs arrive from the left and each incoming interconnection has an associated weight, wji. The perception processing unit performs a weighted sum at its input value:

$$\text{The sum takes the form net} = \sum_{i=1}^{n} w_i o_i$$

Weights associated with each inter connection are adjusted during learning. The weight to unit J from unit I is denoted as w_i after learning is completed the weights are fixed from 0 to 1.

MATERIALS AND METHODS

Pre-processing: The pre-processing step is a crucial element, particularly for offline handwriting recognition (Sharifah, 2008). This is primarily due to the noise that is introduced in the sample image during the scanning process of converting the physical image into its electronic equivalent format, which in turn necessitates that the noise be later removed so that only the relevant information is extracted as features. In addition to this, human handwriting that inherits a high level of variability in terms of style, size, position and orientation requires that a proper normalization technique be executed prior to the recognition process.

Noise removal is somewhat mandatory for off-line handwriting recognition due to the noises introduced during the scanning process. There are several factors that may produce noise, such as precision error of scanning process, existence of dust on the surface of the scanning device or on the surface of handwriting paper documents etc.

There are several techniques that can be applied for noise removal, many of which involve the use of filters that take into consideration picture information of the

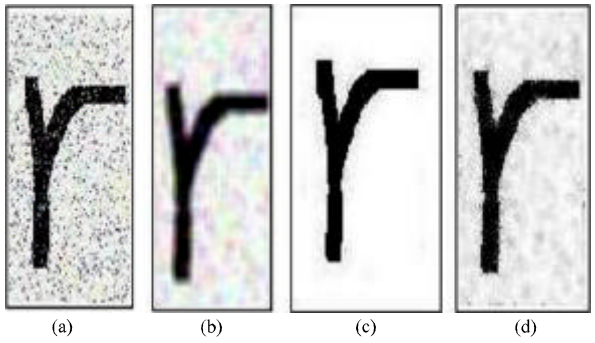


Fig. 3: Noise removal using different filters, (a) image with noise, (b) using average filter, (c) using median filter and (d) using wiener filter

neighboring pixels. These include the average filter, the wiener filter, the median filter and the two-dimensional order-statistic filter. Figure 3a-d show the noise removal technique using several of these filters.

The wiener de-convolution filter can be used effectively when the frequency characteristics of the image and additive noises are known to at least some degree. In the absence of noise, the Wiener filter reduces to the ideal inverse filter.

Median filtering is a nonlinear operation dependent on the procedure that each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image. This filter is normally used with popular noise removal types. On the other hand, the average filter is an image with n-by-n array filter containing equal weights. Plamondon and Srihari had performed selective and adaptive stroke filling with a neighborhood operator, which emphasized stroke connectivity, while at the same time, strict check aggressive over-filling (Plamondon and Srihari, 2000).

Other preprocessing steps such as binarization and sceletonization are optional, which are highly dependent on the chosen algorithm for feature extraction and recognition. Many studies have revealed the impact on the effectiveness of such processes on the system accuracy (Plamondon and Srihari, 2000; Broumandnia *et al*., 2008; Sharifah, 2008). Thus, it is necessary for engineers and developers to carefully study and design the two sub processes in any attempt to build such recognition systems. This, in turn, remains the prime objective of this section, which presents a state-of-the art review on the advancement of techniques implemented for preprocessing.

Segmentation: The segmentization review reported here concentrates at two main levels-word and character levels,

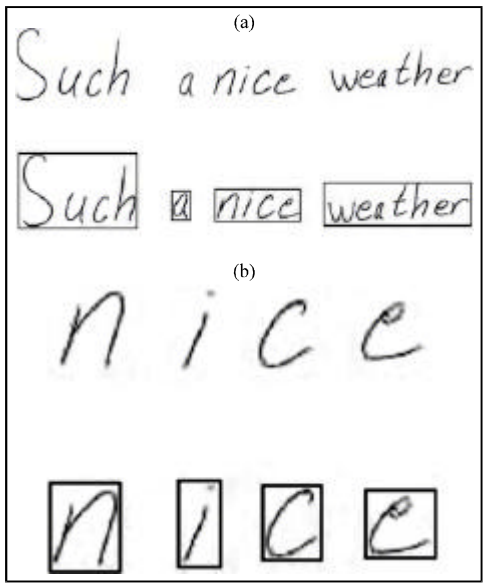


Fig. 4: (a) Word segmentation and (b) character segmentation

respectively. Usually, word segmentization is performed first, followed by character segmentization.

The basic approach is to identify physical gaps or empty spaces between the components which can either be words or the character itself (Mahadevan and Nagabushnam, 1995; Casey and Lecolinet, 1996; Sharifah, 2008).

The only difference between the two is that the gaps between words are usually much bigger compared to those between characters. However, in handwriting, there are exceptions because of additions in writing styles with leading and trailing cursive letters. This implies that the boundaries of the characters and sometimes words in a cursive handwriting are not always well defined. In addition, the segments of a character for instance, are not always well joint. Both these factors make segmentization base solely on identification of gaps and therefore is not always effective for robust handwriting recognition. If most segmentation can be obtained by finding white columns, the regular grid of entire character boundaries can be estimated (Fig. 4a, b). Segmentation points not lying near these boundaries can be rejected as probably due to broken characters; broken characters that may have been caused by the writer's error or in the process of scanning, more than one connected pieces form the character (some white column between the character pieces). In this case, segmentation points missed due to merged characters can be estimated as well and a local analysis can be conducted in order to decide where best to split the composite pattern.

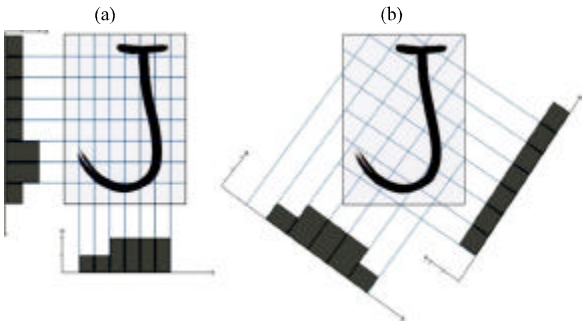


Fig. 5: Profiles for (a) left, right, top, down direction and (b) +45°, +135°, -45°, -135° diagonal directions

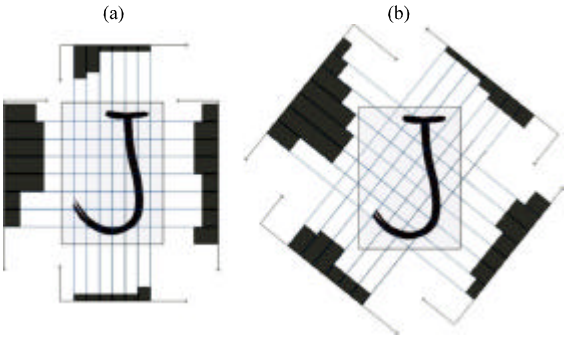


Fig. 6: Line crossing for (a) horizontal and vertical directions and (b) +45° diagonal and -45 ° diagonal directions

Feature extraction: For the feature extraction part, two sets of features had been used in this paper. Both were statistical sets (calculated based on the character image content), with the first set using the gravity center distance and pixel density (Gunter and Bunke, 2005). This set has round thirteen features, while the second set, which used ink crossings and profiles features, has around ninety six features (Tay, 2002).

Ink crossings: The features in this category compute the number of times that a background-foreground transition is encountered. For image size invariant purposes, the image is partitioned into eight stripes. Four directions, i.e., horizontal, vertical, +45° and -45° diagonal, of stripes are used Fig. 5a and b. The algorithm to compute the ink crossing for a line (in either of the 4 directions) is simply by running a pixel from the start of the line to the end and counting the number of transition that a pixel is in black (ink pixel) and the previous position is in white (background). Ink crossings for a stripe are the average of all lines for that stripe. In total, there are 32 features for this category.

Profiles: Profiles describe the shape information of a particular character. It is the relative distance of the first ink pixel position from the border. Eight directions of profiles, in which each direction has eight equally partitioned stripes, are used in this feature category. Figure 5a, b and 6a, b show the profiles in these eight directions.

Pixel density: In the pixel density group, a sliding window is used which gives segments of the image (Fig. 7).

The pixel density is calculated by dividing the number of black pixels in each window with the total number of pixels for the window (Ahmad *et al*., 2008), as depicted in Eq. 10:

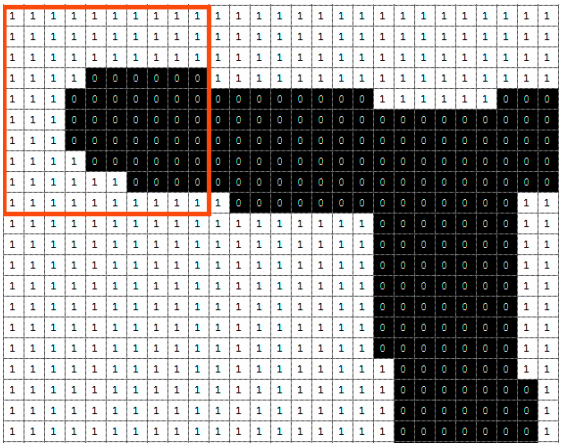


Fig. 7: Feature extracted based on sliding window

$$X_{PD} = \frac{\sum black\_pixels}{tot\_number\_of\_pixels} \quad (10)$$

Gravity center distance: To compute this feature we first use the same sliding window technique as used in pixel density, to calculate the center of gravity as depicted in the Eq. 11 and 12 to get $(X,Y)_{CG}$

$$X_{CG} = \frac{\sum_{i=1}^{n} X_i}{n} \quad (11)$$

$$Y_{CG} = \frac{\sum_{i=1}^{n} Y_i}{n} \quad (12)$$

where, n is the total number of black pixels.

Next to calculate the gravity center distance we need to find the distance to the left down corner from the coordinate of $(X,Y)_{CG}$ for which we applied the Eq. 13:

Table 2: Different features attributes

Features sets	No. of features	Recognition rate (%)
Ink krossings and profiles	96	98.5
Pixel density and gravity center distance	21	83.2

$$X_{DCG} = \frac{\sqrt{X_{CG} + Y_{CG}}}{D} \quad (13)$$

where, D is the Diagonal of the sliding window

The total features selected from these sets are thirteen features.

In our system, Feature selections are based on perception and are due to trial and error. We found that the first two groups of feature sets give more features, which lead to much better results as stated in Table 2.

These features are extracted from the refined images of a total number of 12,912 character images, taken to extract features from them to feed in as input for the training state, while 10815 are for the testing state.

CHARACTER RECOGNITION

In this part, the author will illustrate the general architecture of Artificial Neural Network (ANN) to overcome the challenges of off-line based Handwriting Recognition. The general formula of any machine learning method has two main functions; Training and Testing.

Based on the Eq. 1-9, the design will involve back propagation feed-forward artificial neural network.

The structure of our ANN system is comprised of three layers, including one input layer, one hidden layer and the final layer, representing the output layer as shown in Fig. 8.

Figure 8 ANN Architecture (three layers: input, hidden and output).

Network activating function: The first step in training a Feed Forward Network is to create the network object. The function newff creates a feed forward network. It requires three arguments and returns the network object. The first argument is a matrix of sample R-element input vectors, which is features in our system here. The second argument is a matrix of sample S-element target vectors, which is Prospective Characters Indexes here. The sample inputs and outputs are used to set up network input and output dimensions and parameters.

The third argument is an array containing the sizes of each hidden layer. The output layer size is determined from the targets. More optional arguments can be provided. For instance, the fourth argument is a cell array containing the names of the transfer functions to be used in each layer. The fifth argument contains the name of the

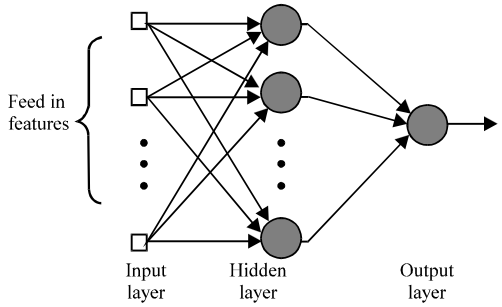


Fig. 8: ANN Architecture (three layers: input, hidden and output)

training function to be used. If only three arguments are supplied, the default transfer function for hidden layers is tansig and the default for the output layer is purelin. The default training function is trainlm.

Back propagation feedforward neural network is used here for its strong behavior in the literature and the well conducted results. Table 3 describes the flow and characteristics in this system.

As described earlier, the output layer for this network is one dimensional matrix for target vectors, which is prospective characters indexes. Each index (from 1 to 17) refers to a particular character need to adopt the network on them.

So, the group of used characters in our data base is like the following array:

l = ['a' 'b' 'd' 'e' 'g' 'h' 'i' 'j' 'l' 'm' 'n' 'o' 'p' 'r' 's' 't' 'u']

Then, the index for each character will be like the following:

a = 1; b = 2; d = 3; e = 4; g = 5; h = 6; i = 7; j = 8; l = 9; m = 10; n = 11; o = 12; p = 13; r = 14; s = 15; t = 16; u = 17

The training stage often takes too long because of the huge amount of calculations done on the data. The maximum training time is 12 h, while 4-6 h is the minimum based on the (trial and error) probabilistic.

Character state prediction (testing stage): After completing training the network, we save the workspace variables to be used for the final step in the ANN system named testing. A particular variable that resulted from the training stage is very useful here to predict the output array for the input vectors array in the testing stage, which is the network object net.

The testing stage receives an unseen group of characters, which should be predicted according to their

Table 3: List of parameters used for the three layers of artificial neural network

Name of parameters	Input layer	Hidden layer	Output layer
No. of nodes	55	40	1
Transfer function	Trainlm (levenberg-marquardt algorithm)		
No. of epochs	1200		
Performance function	Mse (mean square error)		
Transfer function	Logsig (Log-sigmoid)	Tansig (hyperbolic tangent sigmoid)	Purelin (pure linear)
Training goal	0.0005		

last trained characters. The performance of the testing stage is fully dependent on the training stage and its selections.

LEXICAL WORD VERIFICATION

Lexical post-processing corresponds to a clarification step, where contextual knowledge is introduced to the validation or the correction of the recognition results. Three different levels are usually observed in linguistic knowledge introduction: lexical, syntactic and semantic. Most researchers, who used this method such as (Srihari, 1994; Park and Govindaraju, 2000; Carbonnel and Anquetil, 2003; Joshi and Nagy, 2005) followed the procedure of word recognition by lexical similarity or prediction, meaning they used it inside the recognition system. In our research, we did character recognition using ANN and after that a lexical word prediction, which is a benefit for the sake of reducing possible errors, both semantically or syntactically (Bengio *et al*., 1995).

This procedure is comprised of two major steps:

- Building the used Malay Dictionary from the database
- Finding higher lexical similarities from the dictionary using the recognized words

The proposed technique for preprocessing depends on building the word dictionary to make an array for the expected letters combined in each word. As mentioned earlier, the system uses 17 different letters from the alphabets starting from 1-18. The ANN system puts zero after each recognized series of letters that represents a word. This particular step is very significant to give an indication for the next step in checking the lexical similarity and preparing for the next word. The similarity checking process is dependent on the percentage of comparison between the resulted words from the ANN system and all the words in the dictionary.

To do so; simply we need to compute the percentage Variance Accounted For (VAF) between two signals.

The VAF is calculated as:

$$V = (1 - \frac{\text{variance}(y - y\_est)}{\text{variance}(y)}) \times 100\% \quad (14)$$

The VAF of two signals that are the same is 100%. If they differ, the VAF will be lower.
And:

v = vaf(y,y_estimate)

where:
y : Signal 1, word from ANN
y_est : Signal 2, word from the dictionary

The word with highest percentage of similarity will be selected as the output word.

TRAINING, TESTING AND DISCUSSION

The ANN module is trained only for character recognition; where else the system testing has been carried out for both character and lexical recognition. The number of samples used for system training and testing is as follows:

- 16091 sample for training
- 12912 sample for testing

Training: Time consuming: time consuming in the training state is not significant always that is because training state will be done in hidden one time.

Even though, In the first experiment of three layers, if we compare the time of the training phase using the same dataset, same environment and the same number of iterations, the result was about two hours for three layers vs. three hours for four layers system Fig. 9a and b.

Cost and memory requirement: Our experiments show, that three layers are suitable and can be run efficiently, while the same system worked slowly and occasionally gets hanged when we train four layers ANN. That is due to the redundancy in computations in the additional hidden layer. That is in additional to the heavy behavior to the MATLAB in general.

As mentioned earlier, we made our final decision to use the ANN system structure as a result of trial and error, based on several separate training experiments done on each feature extraction techniques, after which the best results were taken, as mentioned in Table 3.

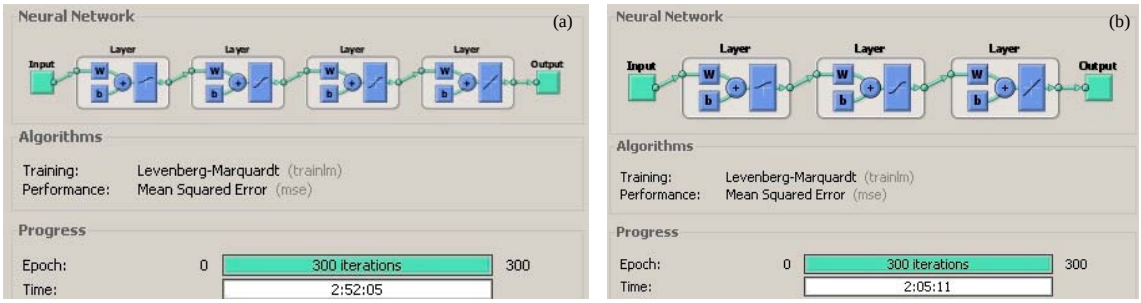


Fig. 9: (a) Four layers ANN training time and (b) three layers ANN training time

Table 4: List of all the training experiments used in the ANN system

RUN	Feature name	No. of epochs	Performance	Regression	Gradient	No. of layers	No. of nodes	Transfer function
1	Profile	10000	0.0095	0.990	0.0011	2	[40,1]	Trainrp
2	Profile	10000	0.0022	0.997	0.001	3	[40,20,1]	Trainrp
3	Profile	1200	0.0005	0.9999	0.00025	3	[55,45,1]	Trainlm
4	Profile	250000	0.133	0.997	0.457	3	[60,30,1]	Trainrp
5	Profile	1000	0.00038	0.999	0.010	3	[40,20,1]	Trainlm
6	Pixel density	6306	0.038	0.96	0.00083	3	[40,20,1]	Trainlm
7	Pixel density	650000	4.52	0.9266	0.199	3	[40,20,1]	Learngdm
8	Pixel density	100000	0.068	0.92	0.0086	3	[40,20,1]	Trainlm
9	Center of gravity	6914	0.061	0.936	0.0015	3	[40,20,1]	Trainlm
10	Center of gravity	3000	4.05	0.934	0.285	3	[40,20,1]	Trainlm
11	PS and CG	600	1.53	0.975	0.103	3	[55,45,1]	Trainlm
12	PS and CG	6147	0.007	0.992	0.00061	3	[40,20,1]	Trainlm
13	PS and CG	17622	0.033	0.966	2.4×10^{-5}	2	[40,1]	Trainlm
14	Ink crossings	1700	0.0123	0.987	8×10^{-5}	2	[20,1]	Trainlm
15	Ink crossings	873	0.0002	0.999	0.0007	3	[40,20,1]	Trainlm
16	Ink crossings and profile	1200	4.95×10^{-4}	0.99999	5×10^{-4}	3	[55,40,1]	Trainlm
17	Ink crossings	250000	0.441	0.993	0.035	3	[15,7,1]	Trainlm
18	Ink crossings	300	0.128	0.998	0.039	4	[35,20,15,1]	Trainlm
19	Ink crossings	300	0.058	0.999	0.0033	3	[35,20,1]	Trainlm

While Table 4 lists all the trial and error experiments and the equivalent system design for each.

Scanning this table, we discovered the following:

- Increasing the Number of Iterations is not always crucial. This means that sometimes we may increase the number of iterations but we fail to notice a distinct decrease in the performance (error)
- Pixel density features and gravity center distance are weak if they work separately. Their behavior could be improved if we combine them together (PS and CG). Yet the profile features and the ink crossings are better in their results
- We can observe the excellence of the three layer system on the equivalent two layer system for the same parameters
- At the same time, the three layer system is better than the four layer system for the same criteria
- The transfer function trainlm is slower than trainrp, yet it gives the best results compared to the others with the same parameter selection

Table 5: Comparisons between the three layers and four layers network

Comparison	3 layers network	4 layers network
Yielded recognition result	99.0006%	98.9511%
First improvement on the result	99.0602%	98.9611%
Second Improvement on the Result	99.0841%	98.9699%
The No. of correct recognized letters	12783	12779
No. of wrong recognized letters	124	133
Final results after word dictionary comparison	99.1197%	99.0015%

Testing: In the testing, the three layers and four layer systems show rather different results, where the three layer system achieved better results as depicted in Table 5.

Final design test: training set: The final design which was adopted for our approach was FF-Back-Propagation Neural Network referred to in Table 3. This architecture comes as result of trial and error, as shown in Fig. 10a and b. A sample of the similarities between the training data and the target, where blue is the target and red is the signal of training data output, which shows the alternation between 0.96 up to 1.06. Figure 10b presents the target of the training for the whole dataset.

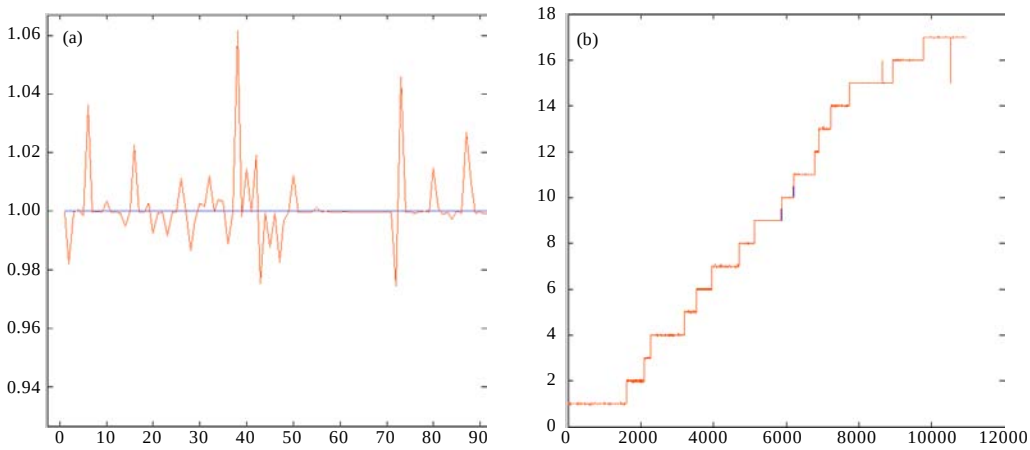


Fig. 10: (a) Sample of the similarity between the training data and the target, where straight line is the target and oscillated line is the signal of training data output and (b) the target of the training data

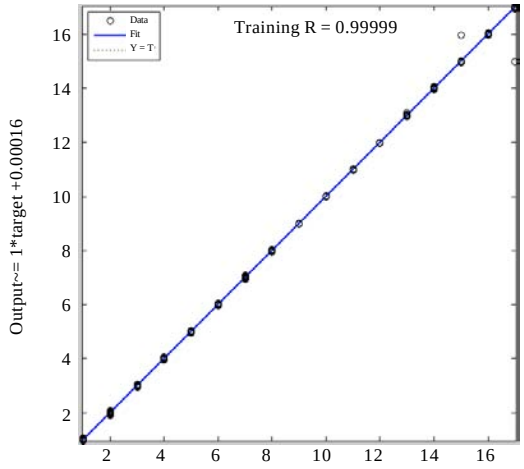


Fig. 11: Target data regression

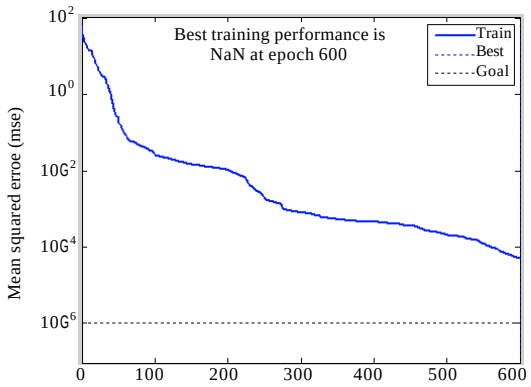


Fig. 12: Training error rates

Afterwards Fig. 11 shows the identification of the aligned trained data on the fit target, where the imaginary diagonal represents the target ideal data regression. Our trained data regression is 0.99999 when the ideal is 1.

In the Fig. 12, the error rate using MSE function has been calculated and been denoted on less than 5×10^{-4} as stated in Table 4 for 1200 iterations.

CONCLUSION

Handwriting recognition is the ability of a computer to receive and translate intelligible handwritten input from sources such as digital documents, photographs, touch-screens and other devices. Handwriting has mainly been classified into offline and online and in this research, we will focus on the offline handwriting recognition. An Adaptive Neural Network approach has been designed and implemented for Malay cheque words recognition. Objectively, this new approach has passed 98.15% for character recognition, while word bank comparisons in the dictionary have increased the ratio of recognition to more than 99%.

ACKNOWLEDGMENTS

This project has been funded partly by University Tenaga National, under the project number J5100A4103. The author would like to take this opportunity to thank his supervisor Dr. Sharifah Mumtazah Syed Ahmad, for her unconditional support and guidance in the preparation of this research work.

REFERENCES

Ahmad, A.R., A.R. Wahap, M. Khalid and R. Yusof, 2002. A neural network based bank cheque recognition system for Malaysian cheques. Proceedings of the International Conference on ai in Engineering and Technology, June 17-18, Sabah, Malaysia, pp: 1-7.

Ahmad, S.M.S., A. Shakil and M.A.M. Balbed, 2008. Offline signature verification system using hidden markov model in matlab environment. Proceedings of the 7th WSEAS International Conference on Applied Computer and Applied Computational Science, April 6-8, Hangzhou, pp: 536-540.

Araokar, S., 2005. Visual character recognition using artificial neural networks. Arxiv Preprint cs/0505016. http://arxiv.org/abs/cs/0505016.

Bengio, Y., Y. LeCun, C. Nohl, C.A.T. Burges and T. Bell, 1995. Lerec: A NN/HMM hybrid for on-line handwriting recognition. Neural Computation, 7: 1289-1303.

Broumandnia, A., J. Shanbehzadeh and M.R. Varnoosfaderani, 2008. Persian/arabic handwritten word recognition using M-band packet wavelet transform. Image Vision Comp., 26: 829-842.

Carbonnel, S. and E. Anquetil, 2003. Lexical post-processing optimization for handwritten word recognition. Proc. 7th Int. Conf. Document Analysis Recognition, 1: 477-481.

Casey, R.G. and E. Lecolinent, 1996. A survey of methods and strategies in character segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 18: 690-706.

Gorgel, P. and O. Oztas, 2007. Handwritten character recognition system using artificial neural networks. J. Elect. Electronics Eng., 7: 309-313.

Gunter, S. and H. Bunke, 2005. Off-line cursive handwriting recognition using multiple classifier systems-on the influence of vocabulary, ensemble and training set size. Optics Lasers Eng., 43: 437-454.

Huang, C. and S.N. Srihari, 2008. Word segmentation of off-line handwritten documents citeseer. Proc. SPIE, 6815: 68150E-68150E.

Joshi, A. and G. Nagy, 2005. Online handwriting recognition using time-order of lexical and signal co-occurrences. Proceedings of the 12th Conference International Graphonomics Society June 2005, Italy, pp: 201-205.

Mahadevan, U. and R.C. Nagabushnam, 1995. Gap metrics for word separation in handwritten lines. Proc. 3rd Int. Conf. Document Anal. Recognit., 1: 124-124.

Park, J. and V. Govindaraju, 2000. Using lexical similarity in handwritten word recognition. IEEE Comput. Soc., 2: 2290-2290.

Plamondon, R. and S.N. Srihari, 2000. On-line and off-line handwriting recognition: A comprehensive survey. 1EEE Trans. Pattern Anal. Mach., 22: 63-84.

Razak, Z., N.A. Ghani, E.M. Tamil, M.Y.I. Idris and N.M. Noor *et al*., 2009. Off-line jawi handwriting recognition using hamming classification. Inform. Technol. J., 8: 971-981.

Srihari, R.K., 1994. Use of lexical and syntactic techniques in recognizing handwritten text. Proceedings of the workshop on Human Language Technology, March 08-11, Association for Computational Linguistics, Morristown, USA., pp: 427-431.

Sharifah, M.A. and A.R. Ahmad, 2007. Hybrid online and offline Malaysian signature and malay handwriting data collection. http://metalab.uniten.edu.my/~abdrahim/p13.pdf.

Sharifah, M.A., 2008. Reviews on Pre-Processing Techniques for Off-line Handwriting Recognition System. MSTC., Kuala Lumpur.

Tay, Y.H., 2002. Offline handwriting recognition using artificial neural network and hidden markov model. Ph.D. Thesis, Universiti Teknologi Malaysia.