VIETNAMESE CHARACTERS RECOGNITION BASED ON SUPERVISED AND UNSUPERVISED LEARNING

A MASTER'S PROJECT SUBMITTED TO
THE GRADUATE SCHOOL OF COMPUTER SCIENCE OF THE
UNIVERSITY OF COLORADO AT COLORADO SPRINGS
BY

HOANG HUY NGUYEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE

DECEMBER 2013

This master's project report for
Masters of Science in Computer Science

degree by

Hoang Huy Nguyen

has been approved for the
Department of Computer Science
By

_____
Dr. Jugal Kalita, Chair

_____
Dr. Edward Chow

_____
Dr. Xiaobo "Joe" Zhou

_____
Date

Table of Contents

Table of Figures

Chapter 1 - Introduction

An important capability that may allow users to interact seamlessly and naturally with the computer is to have intelligent handwriting recognition software. This advanced feature presents significant benefit when incorporated into tablets or mobile devices where a physical keyboard does not exist. While character recognition is a trivial task for the user, it can be quite challenging for a computer program due to the uniqueness of individual handwriting representation. An additional layer of complexity in classification is recognizing languages with diacritic marks.

In a recognition system, two distinct technologies for data input can be identified: online recognition or offline recognition. The difference between these two technologies is the underlying method of data collection. For an offline system, data input is the result of printed or handwritten text that has been scanned or photographed. It is a purely visual representation of text and does not refer to any dynamic information, such as how the character was written and in which order. On the other hand, online recognition is the process of capturing ink signals by using a digital pen or paper-based capture device [1]. This method of handwriting recognition requires intuitive text input from the user, and is the approach for this project.

Several promising handwriting recognition systems from proprietary and open sources have achieved confident recognition rates for Western characters, but none of the systems provides support for Vietnamese letters. This drawback creates the need for a reliable recognition tool that can recognize Vietnamese characters combined with one or more diacritic marks. The purpose of this project is to implement three neural network architectures and apply supervised and unsupervised learning techniques to recognize 32 uppercase letters in the Vietnamese handwriting system.

Vietnamese Writing

Vietnamese is a tonal language spoken by about 82 million people in Vietnam and about 3 million overseas Vietnamese [2]. It has different accents in various regions, but the writing system is standardized for official communication across Vietnam. The origin of Vietnamese writing script was based on Chinese characters; however, the introduction of Latin characters by Roman Catholic Missionaries in the 17th century has transformed the Vietnamese writing system. The contemporary system, referred to as the Vietnamese National Language or *QuốcNgữ*, is an adapted version of the Latin alphabet with additional diacritic marks. The overall Vietnamese writing system contains 29 letters in the alphabet { A, Ă, Â, B, C, D, Đ, E, Ê, G,H, I, K, L, M,N, O, Ô, Ơ, P, Q, R, S, T, U, Ư, V, X, Y}and 6 diacritic marks{-,/,\,^,~,.}. The diacritic marks can only apply to the vowel letters set {A, E, I, O, U, Y}, and some of the vowel letters can have multiple diacritic marks [3].

Chapter 2 - Background Research

Over the past several decades, one of the optimal solutions to tackle pattern classification problems involving character recognition has been through the use of artificial neural networks (ANNs). ANNs represent a mathematical model that simulates the functional aspects of biological neural networks [4]. It is a computation model that consists of artificial neurons and processes information using interconnected networks. While the concept of ANN may seem complex, its main function is to map an input space to an output space. This method has attracted researchers' interest mainly because of its resemblance to human learning and a comfortable level of complexity for practical application.

A few neural network architectures have been studied to solve the Vietnamese character recognition problem.

Hidden Markov Models

A possible framework for recognizing Vietnamese letters is to use a Hidden Markov Model (HMM) combined with a language model. The HMM is a probabilistic model that assigns probabilities to sequences of symbols. Transitions among the states are governed by a set of probabilities called transition probabilities [5]. HMM alone can be an effective method for handwriting recognition, but the performance could improve with the addition of a language model. This model consists of an array of trees built to represent all the Vietnamese letters. In each tree, the parent node is always the first character that makes up the alphabet letter. The advantage of this model is the integration of spell checking in the recognizing step [6]. While the combination of these two methodologies has resulted in considerable improvements in whole word recognition, it has not improved for isolated letters.

Optimized Cosine Descriptor

An alternative solution to recognizing Vietnamese characters is to store the sequence of each handwriting stroke. This approach uses the Modified Optimize Cosine Descriptor (MOCD) algorithm. MOCD operates by separating the letter into several strokes and ordering each stroke in a single sequence of points. It recognizes the letter by classifying the sequences of users' strokes to the desired template. The experiment results obtained from Nguyen and Bui from Vietnam National University proved the difficulty of recognizing letters with diacritic marks due to their smaller size [7]. While the experiment has obtained competitive results for non-diacritic letters, it would not be useful for this experiment because each individual's handwriting style is unique. By recording only the strokes for every letter, it would be difficult to achieve a high accuracy rate for Vietnamese letters with diacritic marks.

Fuzzy Neural Network

Fuzzy neural networks represent a concept derived from fuzzy logic that emulates the way the human mind thinks compared to traditional "crisp" logic theory. It is a machine-learning technique that finds parameters of a fuzzy system by exploiting approximation techniques from neural networks [8]. It operates by choosing as its output the most similar pattern to the input

pattern density. This architecture operates by "fuzzifying" the input pattern and then calculating the similarity density between the input pattern and all learned patterns. Next, the network reaches its conclusion by choosing the most similar pattern to the input pattern in density. Finally, the neural network performs a defuzzification process and sends the result to a non-fuzzy output [6]. An experiment led by University of Natural Science from Vietnam has shown that this method works well for printed Vietnamese characters [6] rather than handwritten letters.

Chapter 3 - System Designs

Feed-forward with Back-propagation

In practical neural network applications, the feed-forward neural architecture offers advantages because of its simplicity and the ability to achieve convergence due to its similarity to the biological neuron cell structure. It is a classification paradigm primarily focusing on mapping specific input space to the desired output space. Prior to performing classification, the network is presented with a set of inputs that is used to feed data into hidden layer. Every neuron in this layer directly connects to other neurons and data computation is performed to determine the weight or strength of each neuron. Once computation is completed within the hidden layer, it forwards the data to the next hidden layer or the activation layer.

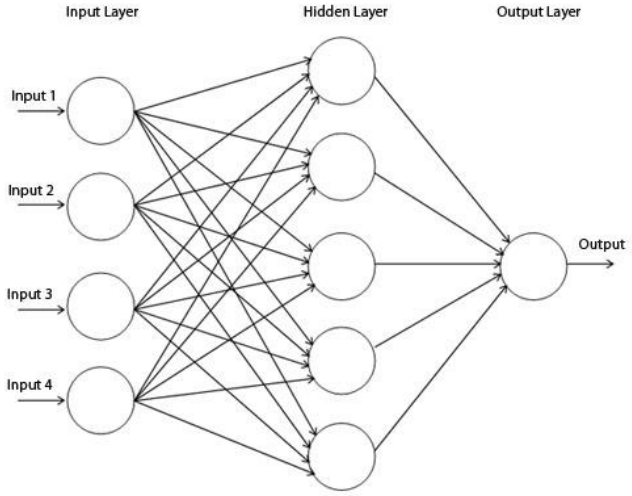


Figure 1 - Feed-forward Neural Network with Single Hidden Layer

Different sets of problem require different training algorithms to be effective, and for the feed-forward architecture, back-propagation is the most popular because of past success in solving the pattern recognition problems. Back-propagation learning is a supervised learning algorithm that consists of two phases: the forward phase and the backward phase. In the forward phase, it operates by propagating an input vector through the network by adding all the weighted inputs and producing the output. Then each output is compared to the target output, and the error between the actual output and the target output is calculated using the mean square error. During the second phase, the weights are adjusted using the calculated error values that are propagated from the output layer back to the hidden layer [9]. By repeating this process through a number of iterations or until a specific error threshold is met, the network is trained to predict an output.

An important factor that contributes to the success or failure of this architecture is determining the parameter that states the number of neurons within the hidden layer. Choosing this value is considered an art rather than science. If the number of neurons selected for this layer is too low, the network fails to find the optimal solution. On the other hand, too many neurons inside the hidden layer can lead to the state of over-fitting. This condition is presented when

there are rare patterns introduced to the network, forcing the network to adjust to specific features that have no causal relation to the desired output. It points the network down the path of not recognizing any new or unseen pattern. Another important parameter that requires adjusting during training phases is choosing the appropriate value for network learning rate. The rule of thumb is to start with a low value and gradually increase as appropriate. A high learning rate will force the network to learn faster, but can also lead to the state in which the network stops learning. A low learning rate can degrade the network performance because of additional time required for training [10].

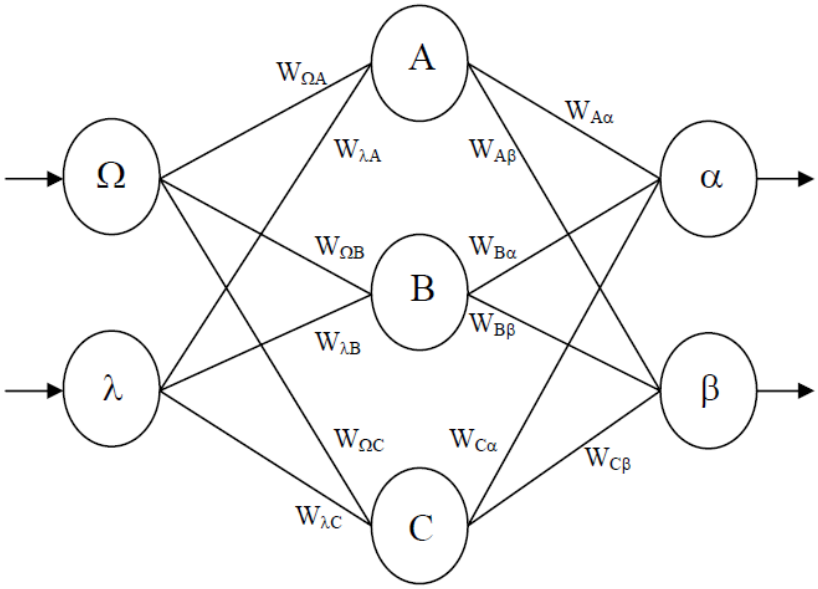


Figure 2 - Neural Network Computation for Hidden and Output Layer

Kohonen Neural Network

The Kohonen Neural Network is an unsupervised learning technique that uses self-organizing maps to classify training data without specific output targets. The network structure operates differently than the feed-forward network in that it contains only an input layer and output layer. The learning process obtains a set of input datasets, and the neurons compute their respective values to produce the discriminant value through Euclidian distance. The particular neuron that has the smallest discriminant value is declared as the "winning" neuron. This is defined as the concept of competitive learning in the Kohonen neural network. From there, the winning neuron determines the spatial location of a topological neighborhood for the surrounding neurons. Each neighbor surrounding the winning neuron receives minimal weight update compared to the actual winning neuron [20]. By iterating through the specified number of epochs, the Kohonen network begins to adapt and learn from the training dataset for classification.

In essence, the Kohonen Neural Network's primary function is to map relationships among the training patterns to reduce relationships among the output patterns, while preserving the general topological relationships [11]. When the network is presented with a validation dataset, it responds by selecting the "winning" neuron that has the most similarities to the

training data set. With this approach, a team of researchers at INRS Telecommunication has reached over 87% in Arabic handwriting recognition with diacritic marks [12].

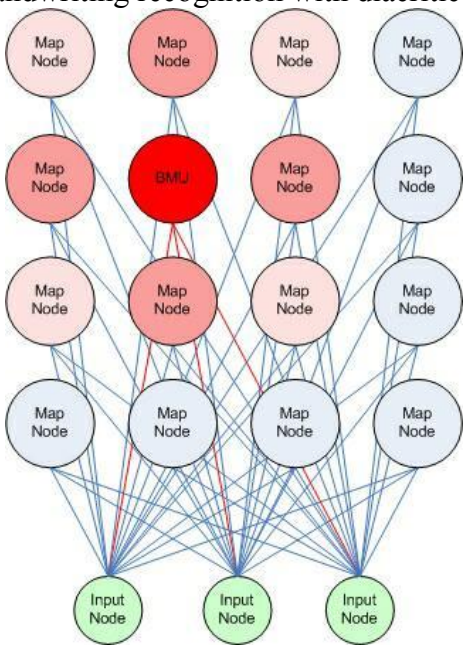


Figure 3 - Simple Kohonen Neural Network

Support Vector Machine with Radial Basis Function

A different approach for supervised learning that maximizes the predictive accuracy of a model without over-fitting the training data is the Support Vector Machine algorithm. The SVM learning concept is based on statistical learning theory that provides a framework for studying the problem of acquiring knowledge, making predictions, and making decisions from a set of data [17]. From a classification perspective, SVM treats all objects to classify as points in a dimensional space and finds the optimal hyper-plane with the maximum margin to separate them into two classes. Choosing which dataset belongs to which class is based on obtaining knowledge of the underlying probability distributions for a given set.

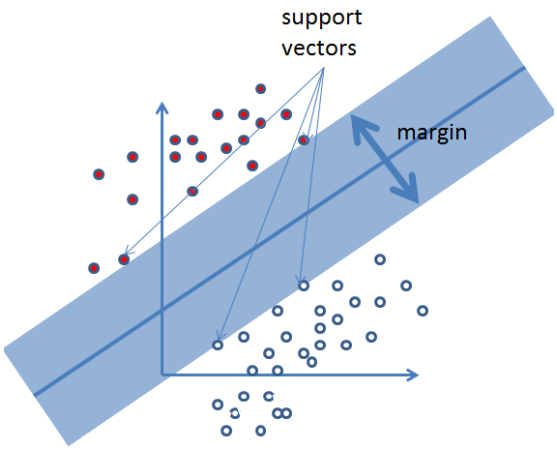


Figure 4 – Linear SVM Architecture

Complex problems involving a large number of features in the input space or having more than two classes cannot often separated using a single straight-line approach. For multiclass problems, a kernel function is employed to find hyper-planes by mapping input lower-dimensional data to higher dimensional data. In general, a kernel function is an approximate function that has the ability to identify similarities between data points. One of the most popular kernel functions for binary pattern classification is the Gaussian Radial Basis Function. This approximate function kernel is similar to the RBF network in that it performs classification by measuring an input's similarity to examples from the training set [19].

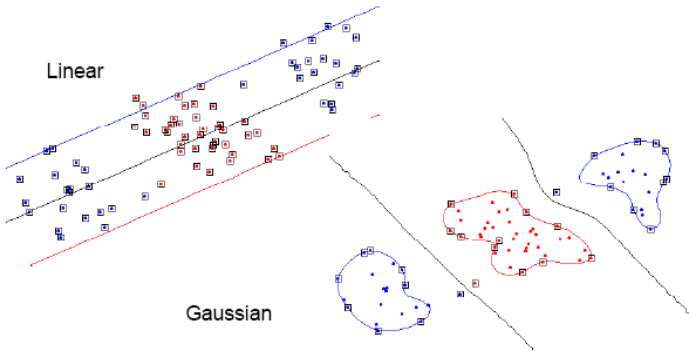


Figure 5 – Non-linear SVM Separates Data Points with Gaussian RBF Kernel

Chapter 4 - Implementation

The scope of this project is to implement three separate neural network architectures and perform a comparative analysis to determine the most successful learning algorithm for Vietnamese characters recognition. Several previous applications designed specifically for this system were analyzed but do not meet the objectives for this project due to low recognition rate and does not provide instant alphabet prediction. Therefore, a custom software application was developed. This new software tool has the capability to collect, train, and recognizes all 29 uppercase Vietnamese letters and three additional letters containing multiple diacritic marks.

A	**Ă**	**Â**	**B**	**C**	**D**	**Đ**	**E**	**Ê**
G	**H**	**I**	**K**	**L**	**M**	**N**	**O**	**Ô**
Ơ	**P**	**Q**	**R**	**S**	**T**	**U**	**Ư**	**V**
X	**Y**	**Ả**	**Ẽ**	**Õ**				

Figure 6 - Uppercase Letters for Classification

Prior to developing an application that has the ability to train and perform character recognition, a simpler XOR application was built to demonstrate the concept of a neural network. This was considered the first stepping-stone to solving complex problems involving feed-forward networks with back-propagation learning. The idea behind the XOR problem is to setup a network that accepts an input vector of two nodes and trains on the datasets to produce the expected output. The purpose of this implementation is not only to gain a deeper understanding of neural networks from a programming perspective, but also to validate that the neural network algorithm has the ability to learn and specify the correct outputs for the corresponding inputs. After successfully completing and testing the XOR problem, the handwriting recognition program is written as a Java GUI standalone desktop application with multithread support. The architecture layout for this application adapts a standard model for handwriting classification and is composed of several phases. It starts out with input data collection that traverses through enhancement and training, and ultimately ends up in the classification phase.

In order to achieve ease-of-use and maintainability for this application, Model-View-Controller and Observer patterns were followed. The application was developed in an Eclipse Integrated Development Environment along with Java Swing library to support graphical display. When the application is launched, three separate tabs are displayed to perform primary functions of data collection, training, and validation.

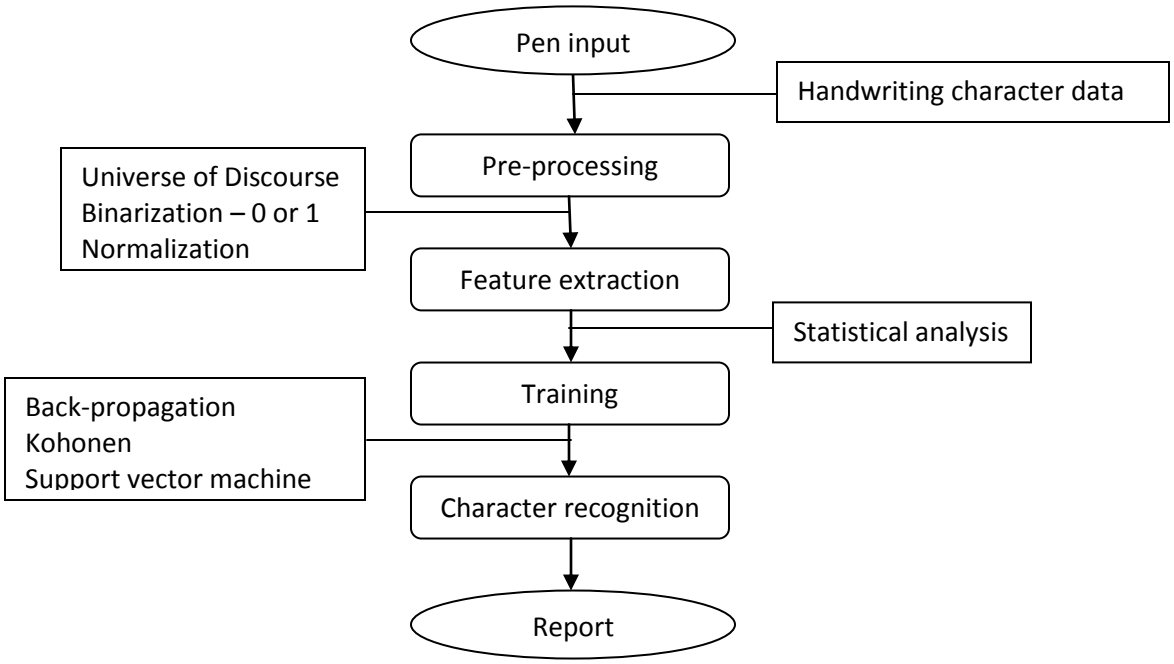


Figure 7 - Block Diagram Model for Vietnamese Character Recognition System

Data Collection

One of the crucial parts of building effective character recognition software is to have many training data instances available for that specific domain or character set. With more data, the neural network learning algorithm has broader exposure to many samples from different users. While training datasets for Western characters are widely available from various universities and research institutions, Vietnamese datasets are severely limited. In order to obtain training data for this project, a manual data collection sub-system was built within the application by utilizing a tablet with digital pen input. From the Data Collections tab, users can write an alphabetic character with the digital pen and save it to a file. While the digital pen was in motion, the application captured electronic ink signals from the x and y coordinates and stored these data into memory. When the user completed writing and pressed the Save button, the dynamic information was passed through the conversion method as an array for preprocessing and feature extraction.

Preprocessing

The main objective of the preprocessing phase is to discard irrelevant information in the input dataset and produce useable data for the training phase. These techniques were applied in the form of binarization, normalization, segmentation, or background noise reduction [15]. For online handwriting systems, binarization and normalization are sufficient to perform enhancement for the input dataset. Binarization is the process of transforming digital ink signals of x and y coordinates into a value of 0 or 1. When a character is being written, the image is converted to a grid of binary bitmaps in the context of 9x11grids. A value of 1 indicates that it was identified as a black pixel, whereas white is a value of 0. The decision to utilize a 9 x 11 data grid is to support letters with multiple diacritic marks. Vietnamese characters are part of a complex writing system that can have diacritic marks on top of, below, or on top of and below a character. With 99 inputs obtained from the data grid, a data normalization range of 0 to 1 is necessary to perform output computation for all learning methods [14].

One of the problems that began to surface during data collection was how to standardize every character written to have equal size. Individual user writing can generate smaller or bigger letter compared to one another, making it nearly impossible for training. The solution to this common problem is to implement the Universe of Discourse methodology. This approach finds the shortest matrix that fits the entire character skeleton so that every character output has equal size[13].

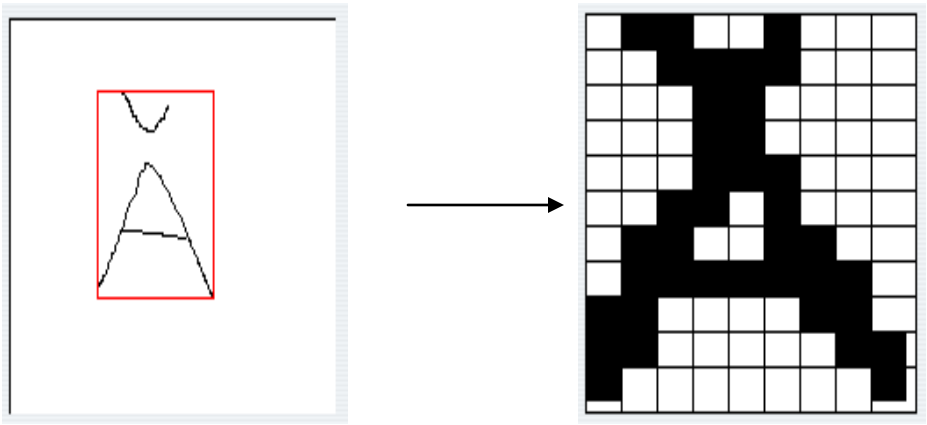
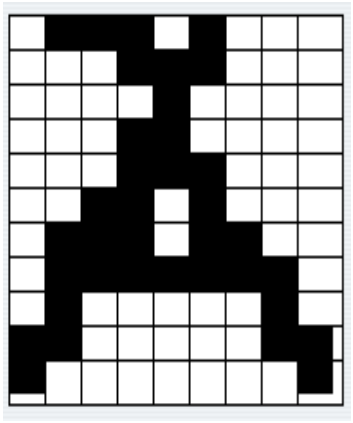


Figure 8 - Universe of Discourse Conversion for Handwriting Character



0	1	1	1	0	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	1	1	0	1	0	0	0
0	1	1	1	0	1	1	0	0
0	1	1	1	1	1	1	1	0
0	1	0	0	0	0	0	1	0
1	1	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1

Figure 9 -Converting Character to Binary Format Data Grid 9 x 11

Feature Extraction

The number of data elements obtained from the preprocessing phase can be substantial due to the high degree of variability and uniqueness in individual handwriting. The purpose of the feature extraction phase is to reduce this data to form a feature vector that maximizes the recognition rate with the least amount of elements. Several strategies are employed to extract characteristics for the dataset including statistical, structural, and global transformations. However, two of the most common methods are statistical and structural. In a statistical strategy, the feature vector is extracted by calculating a distribution of points that often have low tolerance to style variations. On the other hand, structural methods have high tolerance for distortions and style variations [15]. Statistical extraction was applied to the handwriting dataset by taking the bitmap binary data after preprocessing and dividing it into 11 non-overlap zones. From there,

the extraction class calculated object pixel densities by adding the number of object pixels within that zone and dividing it by the total number of pixels [16]. The results from these zone calculations were saved to a file and used as input neurons for neural network training.

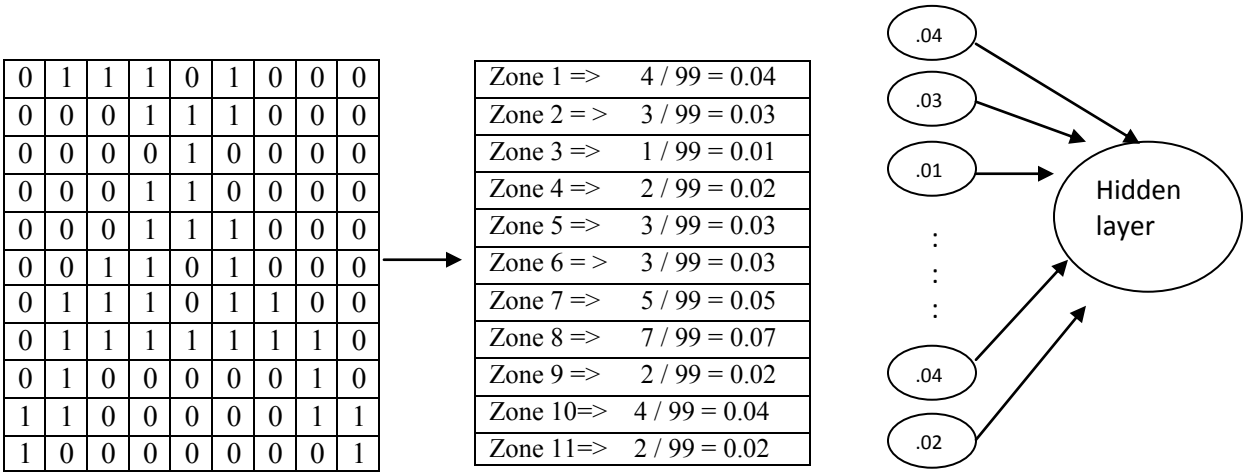


Figure 10 – Non-overlapping Zone with Density Calculation for Feature Extraction

Training

After all input datasets finish with preprocessing and feature extraction, machine training begins with selecting a training algorithm from the drop-down menu box. Three separate training algorithms were pre-defined, with optimal parameters in the XML configuration file obtained through many different experiments for this project. Each of these algorithms requires different type of data scaling behind the scenes to achieve maximum efficiency. When the program loads training data into memory, it first begins to perform scaling and applying k-fold cross-validation to the training set. K-fold cross-validation is the model selection approach that finds optimal parameters for training and measures performance for that model through data partition [18]. Given an input dataset, the k-fold cross validation process partitions the data into k separate sets, where k is the number of sets. Due to the limited amount of data for this implementation, 5-fold cross validation was used, where 4 sets were used for training and the last set was used for testing.

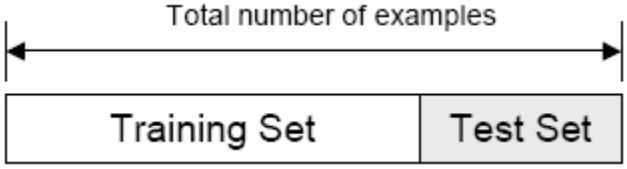


Figure 11 – Cross-validation Data Partition

Recognition

Depending upon the size of dataset and training algorithm selected, the training completion criteria is achieved either by meeting predefined error thresholds or a certain number of iterations. Once the training runs have been completed, a process will read in the test dataset file and perform automated character verification to determine the success rate for that specific training algorithm.

Chapter 5 - Experiments

The plan for this project was to implement an XOR problem to verify the neural network learning concept for a practical problem. The success criterion for the XOR problem is to obtain two input values and predict an output based on learning. Once the setup was coded and tested, the neural network was modified to accommodate a character recognition application with 99 input neurons and 29 neurons for the output layer. Even numerous attempts, the implementation failed due to some incorrect error calculation during the training phase.

Since manual implementation of the neural network is time-consuming and requires many verification steps, the decision was made to switch to a third-party library, the Encog Machine Learning Framework to provide neural network confidence and reduce complexity for this project. Encog is an open source framework developed by Heaton Research to provide advanced machine learning algorithms support [21]. Adding this framework, the user can invoke a library call through the API, or pass CSV file data to a specified training method. With the integration of Encog API, training data was read in as a text file and forwarded through the Encog engine with learning parameters configured in the resource XML file. The first few output results from back-propagation training obtained had a less than 70% success rate for a set of 29 characters. According to several research papers, several parameters required adjustment to improve the overall success rate.

The first parameter to modify was the number of neurons within the hidden layer. By adjusting this specific parameter, the program immediately achieved noticeable success and training time was tremendously decreased. The next parameter to be modified was the learning rate. Initially, learning rate was set around .7. This value forced the neural network to learn much faster at the cost of reducing recognition rate. After running approximately 135 different experiments with back-propagation, the perfect parameters for this network were a single hidden layer containing37 neurons and learning rate set at .03. With these settings, the neural network was able to achieve over 90% recognition rate.

Training the Kohonen neural network required a different approach compared to back-propagation and SVM. The only parameter that required adjustment was the learning rate. The architecture for this type of neural network contains only an input layer and an output layer. Instead of training the neural network for a specific target and perform network error reduction, the training performs neighborhood calculations through an iterative approach. The number that yields optimal character rates with respect to training time is approximately 5000 iterations.

As mentioned above, SVM is based on the statistical learning theory that performs classification based on finding the optimal hyper-plane that yields the largest margins between two classes. For multiclass problems, the kernel function is what gives SVM the capability for effective pattern classification. The kernel function used for this project was the default classification model of Gaussian Radial Basis Function [21]. It is the default library in Encog that has preset parameters required by SVM.

Problems Encountered

During different phases of the project, several challenges were encountered from implementing the application as well as integrating with the Encog framework. Some of the challenges are discussed below, along with the solutions for those problems:

Training Data

An important aspect of achieving a high recognition rate for any training algorithm was to train the network with good training data. When the project first implemented with 29 characters, training data was collected from four different participants who are most familiar with Vietnamese characters. One problem discovered during the training phase was that, or unknown reasons, the network yielded good results for the few first hundred samples but then started incorrectly recognizing letters, which were off by one index value. Additional research into training datasets indicated that, within the training set, one of the characters had an incorrect index value beginning at line 1243. Subsequently, the rest of the data had incorrect index values, resulting in incorrect recognition.

To prevent this error in the future, two verification techniques were added to the recognition application. The first function was added in the Data Collections tab to track every letter the user writes and guide them through the dataset. When the user writes a letter and saves it to a file, it immediately displays the next character that needs to be written. The second function was added in the Validation tab to allow the user to verify that all the letters have been written in the correct order.

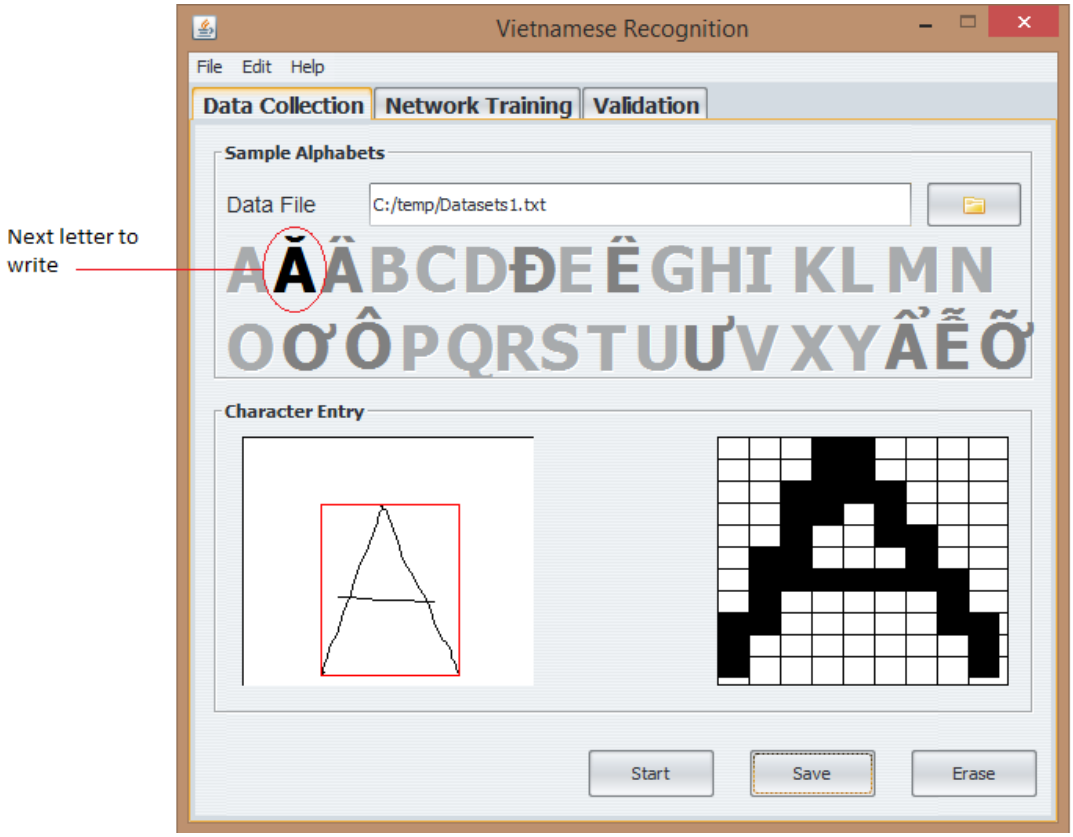


Figure 12 - Data Collection Character Tracker

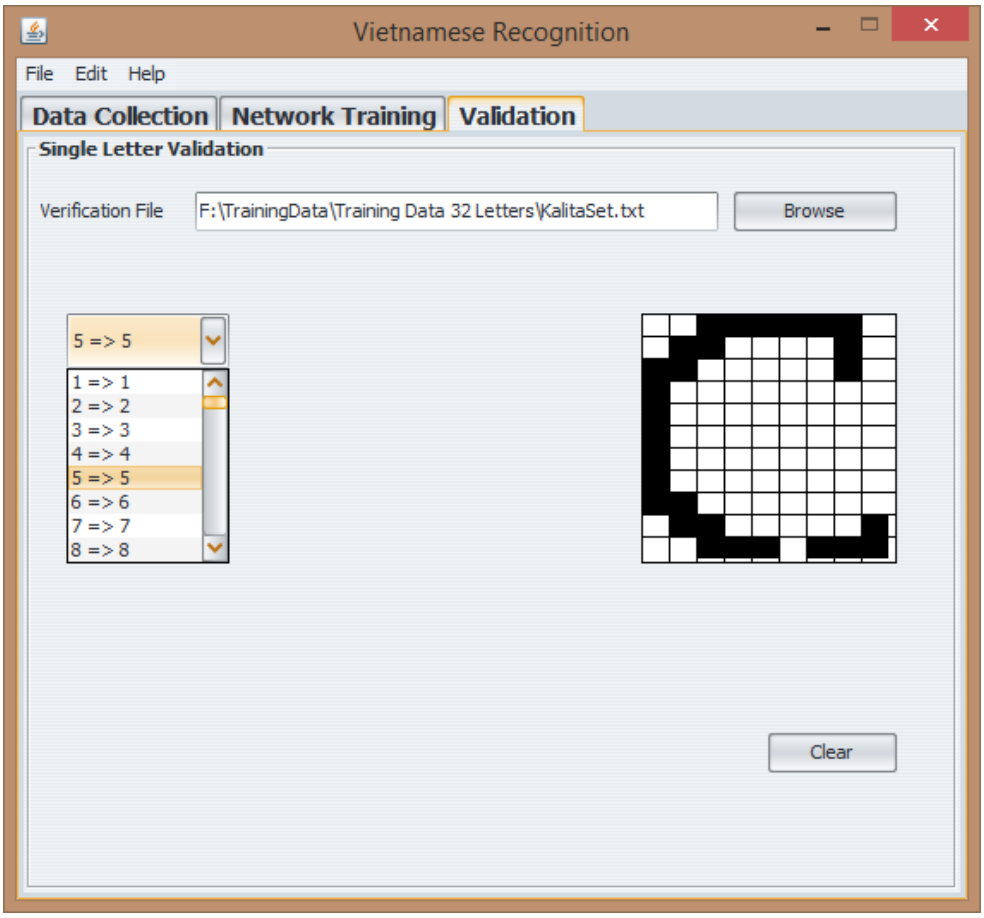


Figure 13 - Training Dataset Manual Verification

Bipolar Sigmoid Activation vs. Binary Sigmoid Activation

Scaling character data into a useful format for training is accomplished in the data preprocessing step. The most recommended scaling pattern classification is to convert character data to the form of binary activation. Although this may be true for other classification problems, our experiments indicated that bipolar activation returns better results. A bipolar pattern is better than binary because using the number 0 leads to learning nothing and 0 is often considered background noise during the training phase [22].

Feature Extraction Methods

The feature extraction solution used in this project was to equally partition pixel data into 11 separate zones. By adding all the number of black pixels for each zone and dividing by the sum of all pixels, actual input datasets were reduced to only 11 neurons. As a result, the recognition rate was substantially lower than expected. Thus, this approach was abandoned and all 99 inputs were used for training.

Chapter 6 - Results

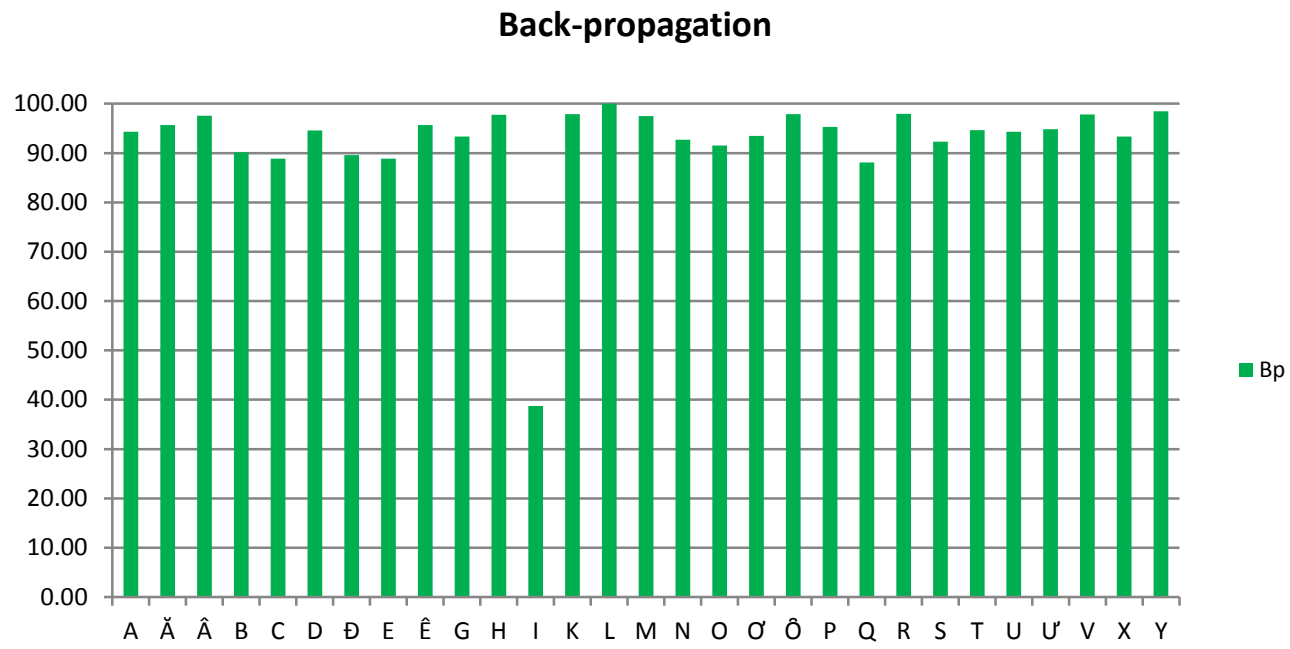
For the first experiment, the scope was to perform character recognition for 29 uppercase Vietnamese characters using a training set of 1537 handwriting characters. The dataset consisted of written samples from 4 different participants that are familiar with Vietnamese characters. The output results below are experiments conducted with an average of5 experiments for a given dataset and training algorithm. The training process began by loading a text file with 1537 characters and performing k-fold cross validation by randomly partitioning data into 5 sets. All characters received a random value of 1-4 for training and 5 for testing. This approach was applied to all subsequent training algorithms for both experiments.

Initial Experiment - 29 Characters

Back-propagation

The following results were obtained with the Encog neural network engine constructed with 99 input neurons, 37 neurons for the hidden layer, and 29 neurons for the output neuron to represent each character. The learning rate for all 5 experiments was set at .07.

- Training: 1246.2 samples
- Test: 290.8 samples
- Time: 12.68 seconds
- Success rate: 93.2%

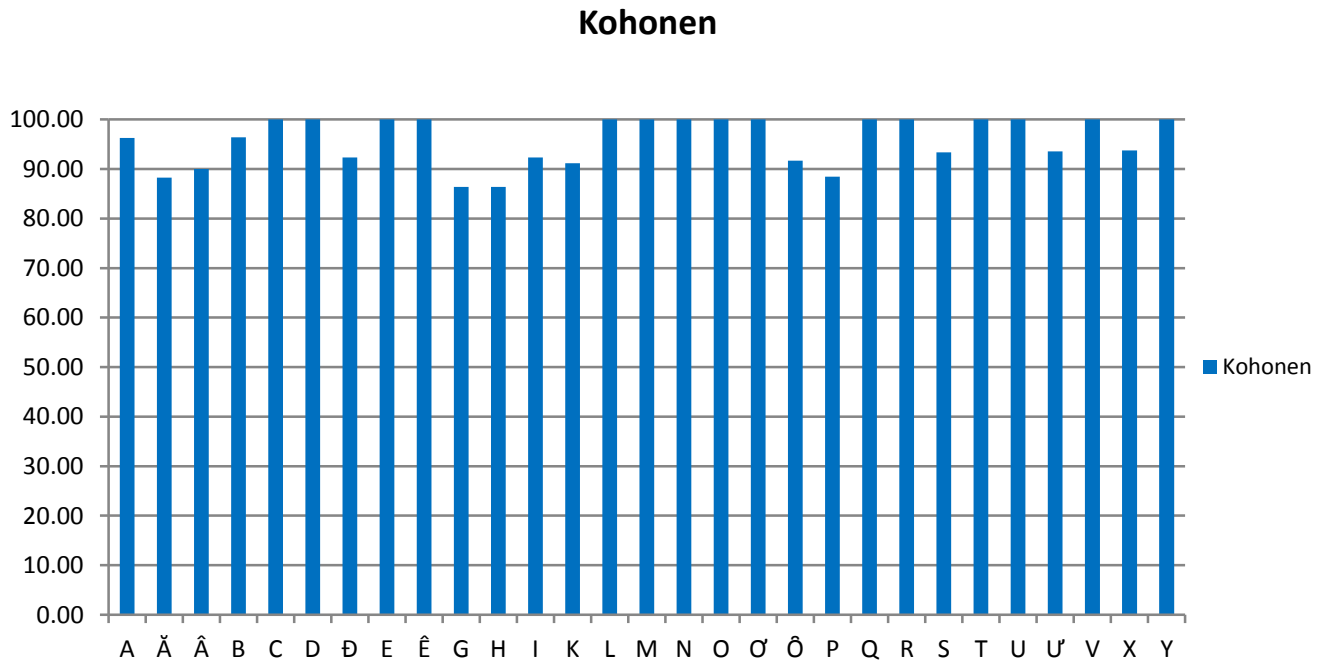


The graph above indicates the success rate for each character from the Vietnamese alphabet. With back-propagation learning, all characters performed well above 80 percent success rate, with the exception of the letter "I". The letter "I" achieved low score due to bad training data obtained from the users' writing..

Kohonen

Kohonen training with the Encog library required training for individual sets. Unlike back-propagation, training data feeds through the network in sequence order starting with the first character and iterating to the last character. Datasets are manually divided into approximately 80% for training and 20% for validation. The Kohonen neural network training engine was implemented with 99 input nodes and 29 output nodes with zero hidden layer.

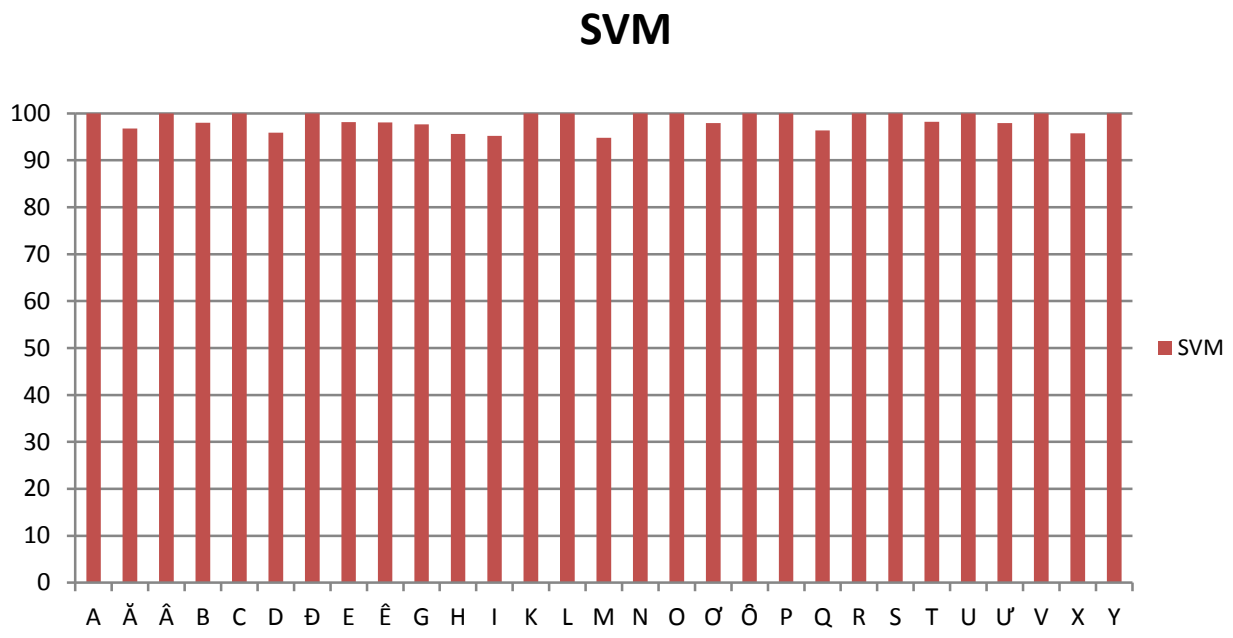
- Training: 1184.5 samples
- Test: 352.5 samples
- Time: 11 seconds
- Success rate: 95.5%



SVM

Character recognition with SVM requires the use of kernel function for multiclass problems. Each individual class represents a character. The radial basis function kernel was implemented with default values of epsilon=.001, gamma=1, and coefficient=0. The difference between this method and back-propagation is that there is only one output as opposed to 29 outputs. When the network performs recognition, it produces an index value that maps to a specific character in the alphabet. A value of 1.0 would indicate that the network identified the input dataset as letter "A".

- Training: 1229.6 samples
- Test: 307.4 samples
- Time: 90.6 seconds
- Success rate: 98.5%



Incorrect Letter Identification

The following table displays the most common mistakes in letter recognition for each algorithm. With an average of 5 experiments, a letter with diacritic marks was often incorrectly identified as another letter with different diacritic marks. Results with a "-" indicate that the letter achieved a 100% success rate and with no incorrect identification.

Back-propagation

A	Ă	Â	B	C	D	Đ	E	Ê	G	H	I	K	L	M	N	O	Ơ	Ô	P	Q	R	S	T	U	Ư	V	X	Y
Ă	A	Ă	G	O	Đ	D	B	P	C	N	Ư	X	-	H	U	C	Ô	Ư	S	S	E	C	I	Ư	U	Y	V	X

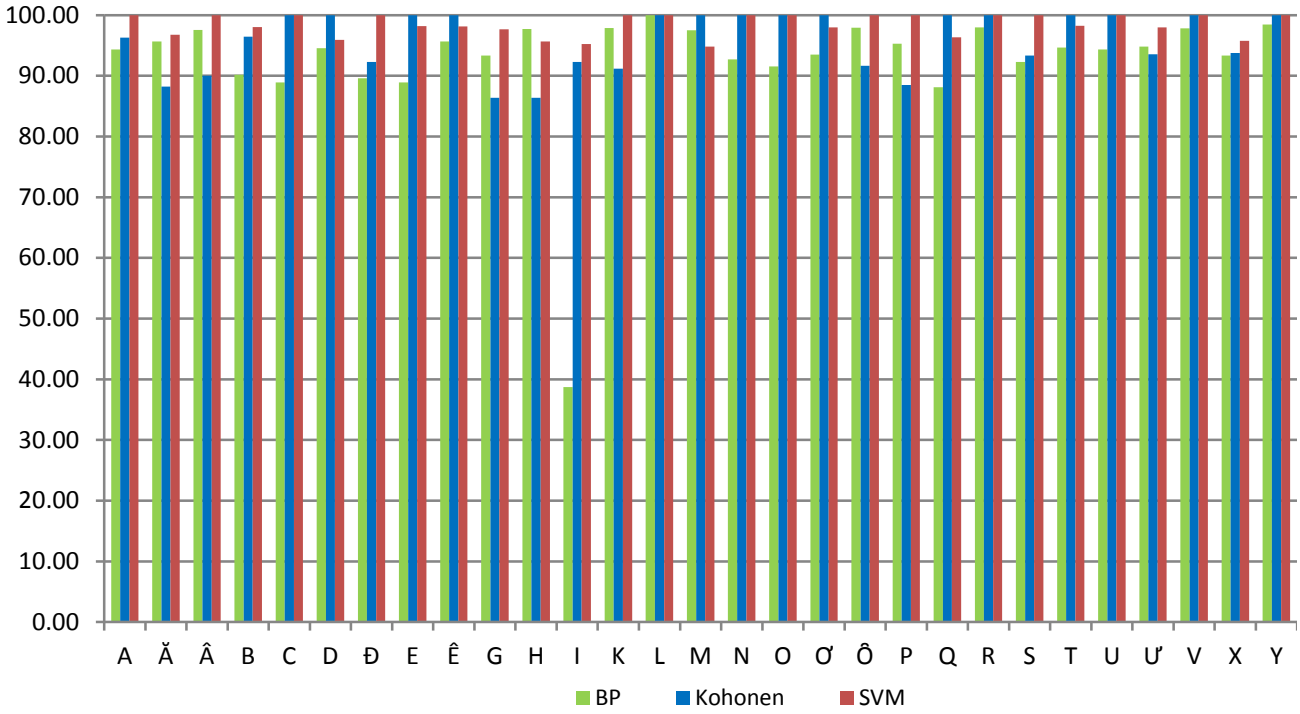
Kohonen

A	Ă	Â	B	C	D	Đ	E	Ê	G	H	I	K	L	M	N	O	Ơ	Ô	P	Q	R	S	T	U	Ư	V	X	Y
Ă	X	Â	E	-	-	D	-	-	Q	P	T	R	-	-	-	-	-	O	S	-	-	K	-	-	U	-	V	-

Support Vector Machine

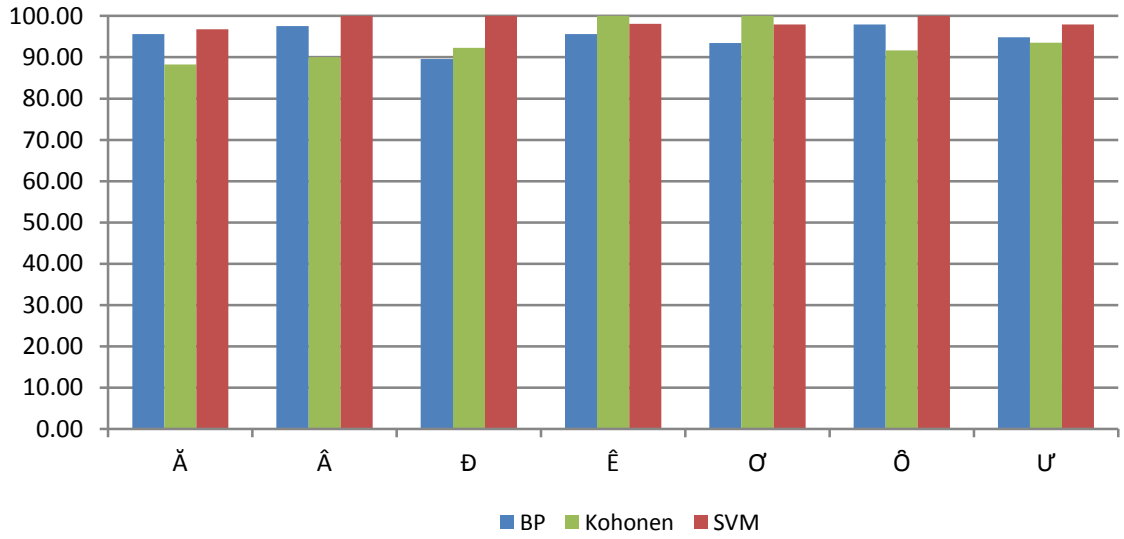
A	Ă	Â	B	C	D	Đ	E	Ê	G	H	I	K	L	M	N	O	Ơ	Ô	P	Q	R	S	T	U	Ư	V	X	Y
-	A	-	I	-	Đ	-	B	B	C	L	T	-	-	H	-	-	-	-	-	O	-	-	I	-	U	-	I	-

All Training Algorithms Comparison

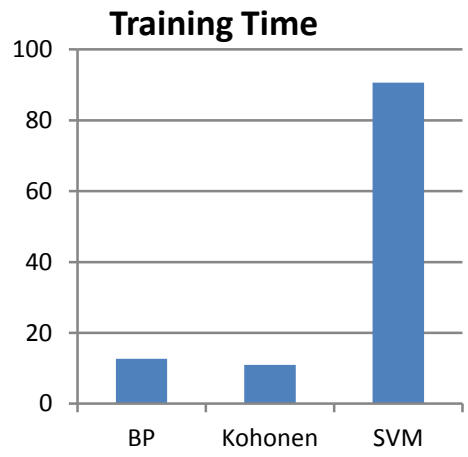


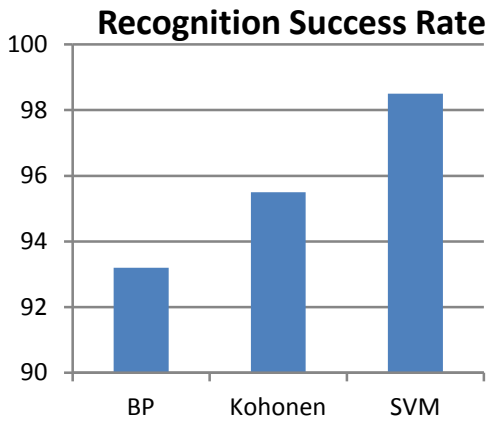
Diacritic Marks Comparison

Of the29 Vietnamese characters, 7 contain diacritic marks. SVM achieved higher success rates over Kohonen and Back-propagation for 5 out of 7 {Ă Â Đ Ô Ư}, whereas Back-propagation obtained higher success over Kohonen for 4 out of 7 {Ă Â Ô Ư} characters.



Network Metric Performance





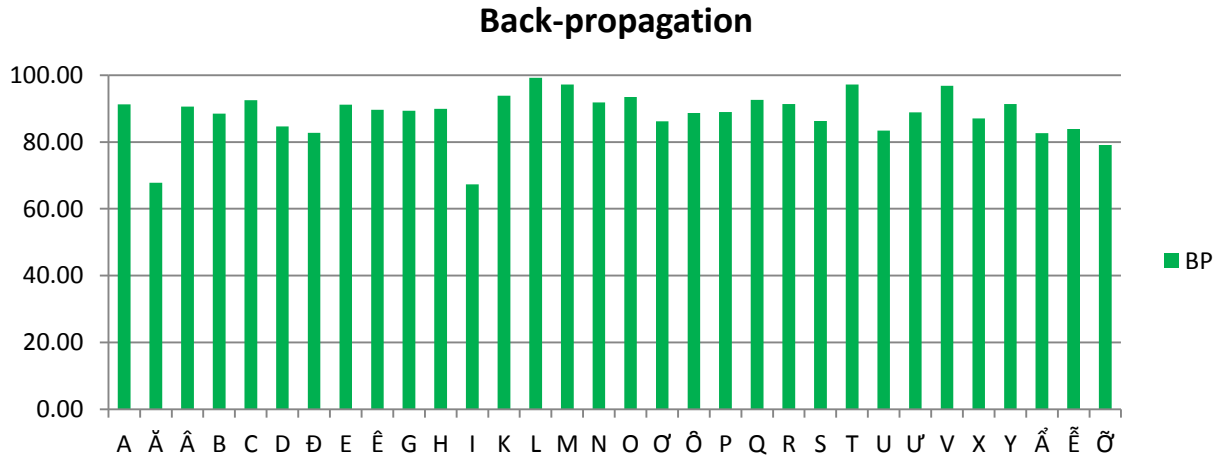
Second Experiment - 32 Characters

Using the same training techniques and algorithms as the initial experiment, the scope was expanded to 32 characters, where the last 3 contain characters with multiple diacritic marks. The datasets are also expanded to 14 different writers, where 6 of the writers are not familiar with Vietnamese characters. The numbers of datasets collected were4448 available for training and testing.

Back-propagation

Similar to back-propagation training in the first experiment, the dataset was partitioned into training set and test set. The exception was that, instead of 29 output neurons, the second experiment contained32 output neurons for each letter.

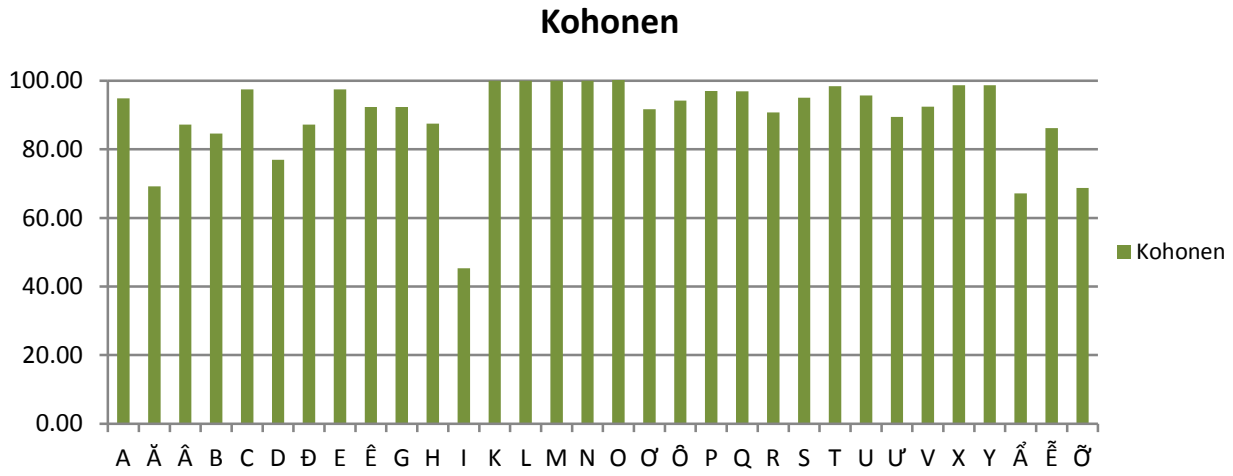
- Training: 3570.8 samples
- Test: 878.4 samples
- Time: 23.6 seconds
- Success rate: 87.5%



Additional training data and additional handwritten characters decreased the recognition rate to an average of 5% for back-propagation. For characters with multiple diacritic marks, the success rate was approximately around 80% average.

Kohonen

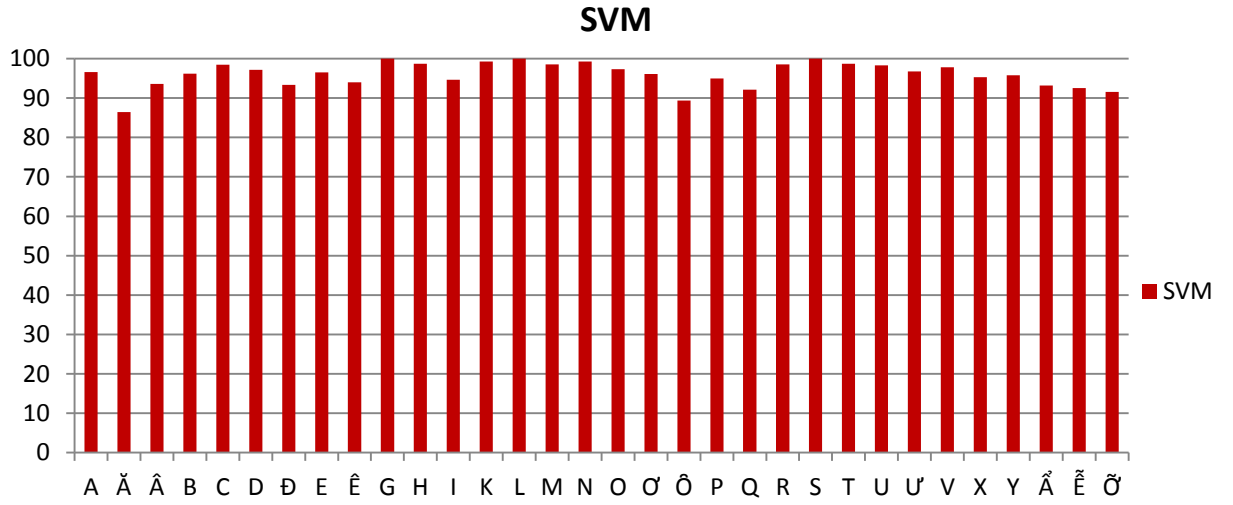
- Training: 3412 samples
- Test: 878 samples
- Time: 17.2 seconds
- Success rate: 90.3%



Variations in characters from different participants resulted in lower recognition rates for most of the characters across the board for the Kohonen neural network. As for multiple diacritic marks, the recognition rate is in the high 60% for 2 out of the 3 characters.

SVM

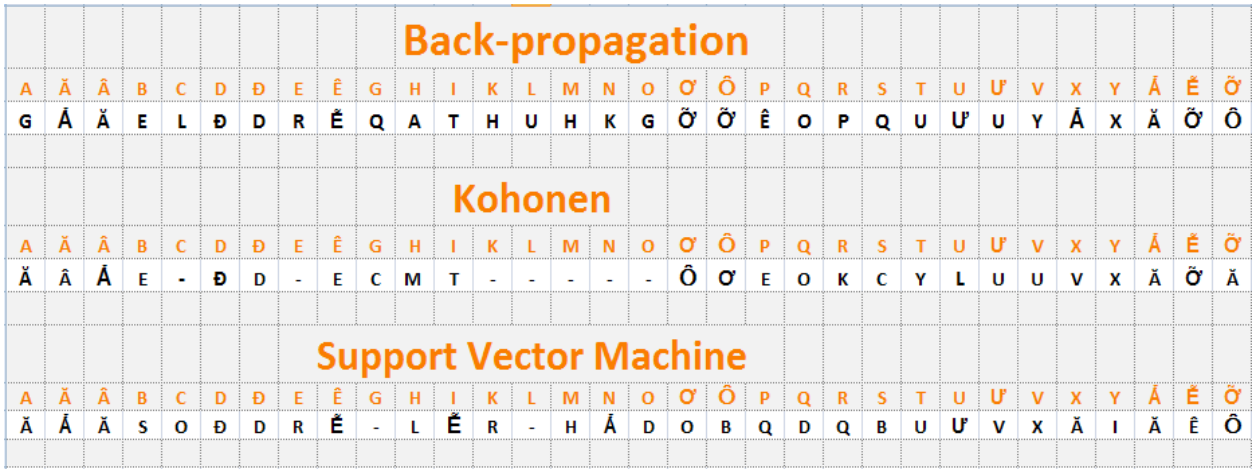
- Training: 3550 samples
- Test: 897 samples
- Time: 536.2 seconds
- Success rate: 95.9%



Out of all three training algorithms, SVM consistently achieved the highest recognition rate for all characters. Using the same configuration as the first experiment, the network success rates for multiple diacritic marks are over 90%.

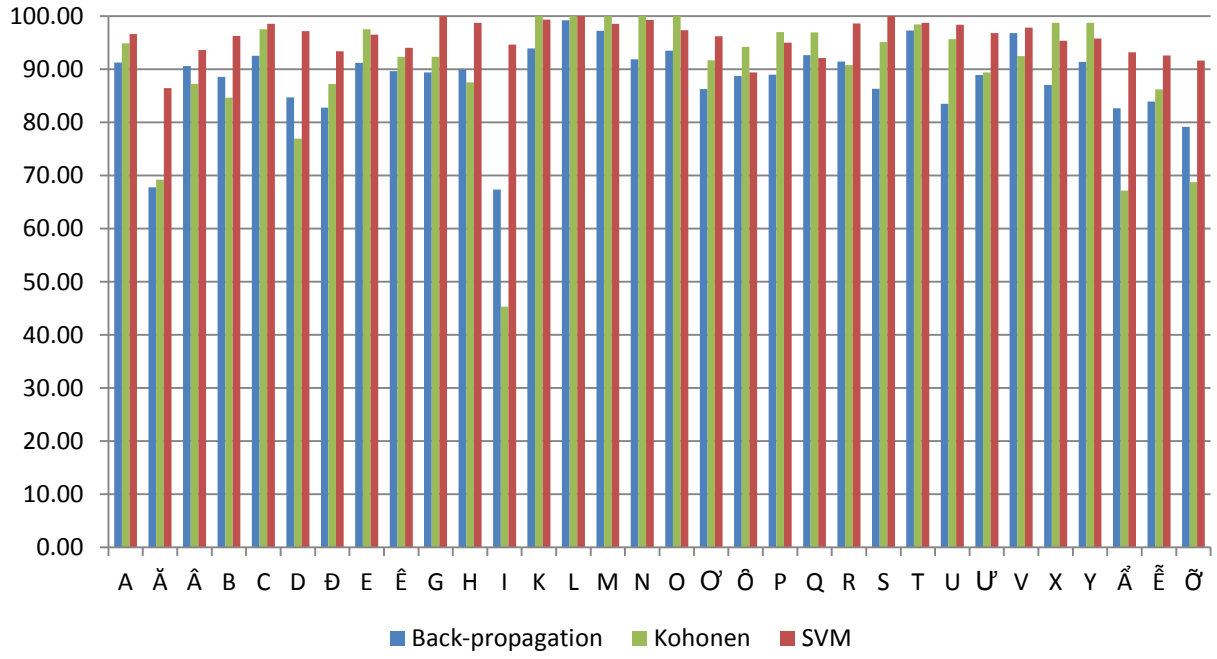
Incorrect Letter Identification

Increased training data and test data has resulted similar patterns when it comes to incorrect letter identification. For letters {D, Đ, Ễ} all three algorithms output the same wrong letter when it performs identification.



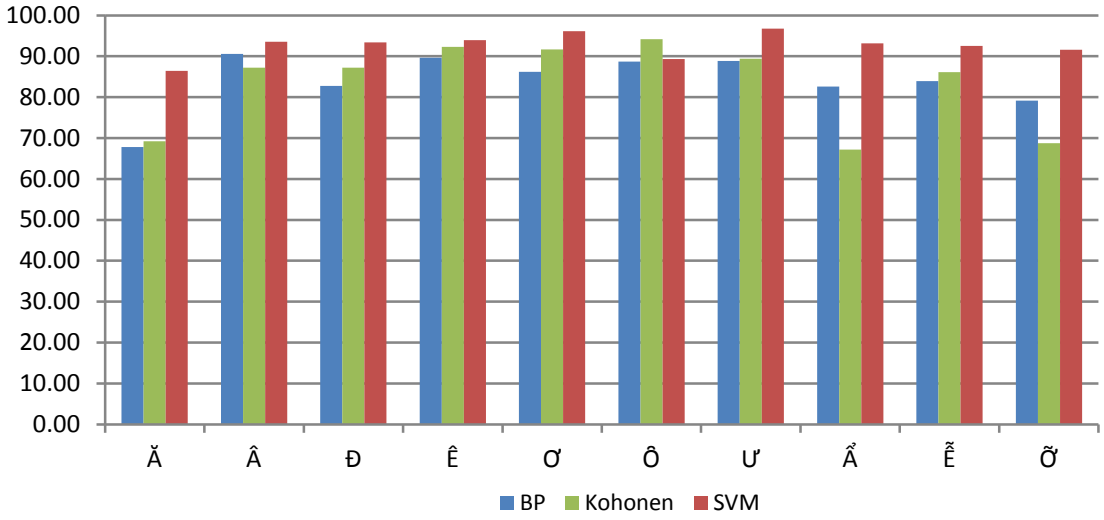
All Training Algorithms Comparison

Similar to the initial experiment results, SVM recognition success rate has surpassed the other algorithms even for characters with diacritic marks.

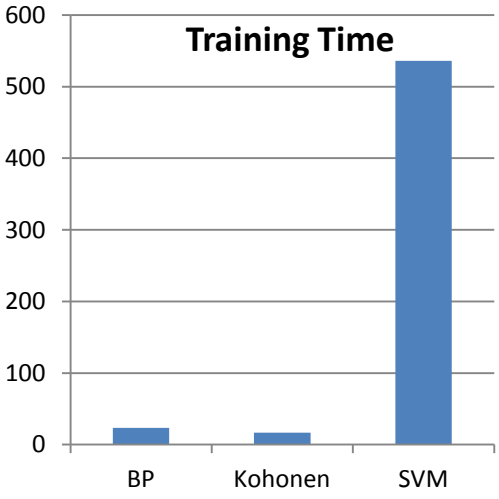


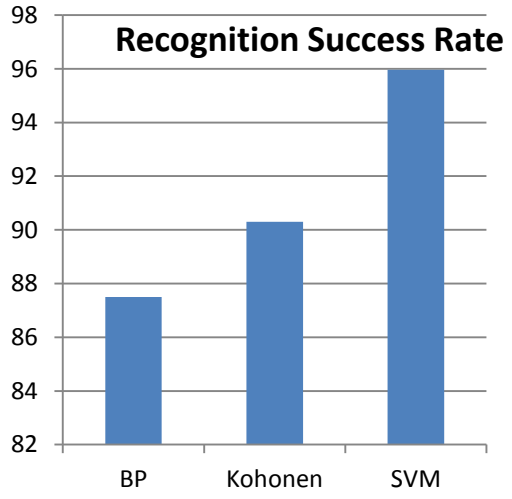
Diacritic Marks Comparison

For 32 Vietnamese alphabets, it contains 7 single diacritic marks and 3 multiple diacritic marks. SVM achieved better rates for 9 out of 10 {Ă Â Đ Ê Ơ Ư Ẩ Ễ Ỡ} over Kohonen and Back-propagation. With more training data, Kohonen network correctly identified 7 out of 10 diacritic marks over Back-propagation {Ă Đ Ê Ơ Ô Ư Ễ}.



Network Metric Performance





Chapter 7 – Conclusion, Future Research

Recognizing online Vietnamese characters involves many different challenges, ranging from uniqueness of individual handwriting to recognizing multiple diacritic marks. In this project, three training algorithms were implemented to perform comparative analysis for Vietnamese character recognition. Through two different experiments with over 4400 collected training datasets from 14 different writers, SVM obtained a recognition rate over 95%for all characters, as well as characters with diacritic marks. Furthermore, the project completion status yielded a practical Vietnamese character application that gives users the capability to collect, train, and performs recognition with artificial neural network technology.

Even though SVM has exceeded 95% success rates, it only focuses on a subset of Vietnamese characters. Additional improvements can be made to expand this application to recognize all Vietnamese characters. Several techniques for preprocessing should be explored further, including segmentation, feature extractions, and stroke analysis. Additional training data from many different users and other neural network technologies are worth considering for expansion this project to recognize all Vietnamese characters.

References

[1] Vision Objects. What is Handwriting Recognition. 2009.
<http://www.visionobjects.com/handwritingrecognition>

[2] Simon Ager. Writing Systems & Languages of the World. 1998-2010.
<http://www.omniglot.com/writing /vietnamese.htm>

[3] Wikipedia. Vietnamese Language.
<http://en.wikipedia.org/wiki/Vietnamese_language>

[4] David E. Rumelhart, Bernard Widrow, Michael A. Lehr. The Basic Ideas in Neural Networks. Communications of the ACM (1994) 87-91.

[5] Vui Hai Quang, Hoang Diem, Pham Nam Trung, Lam Tri Tin, Nguyen Duc Hoang Ha, An Nguyen. (1998). A System for Recognizing Vietnamese Document Images Based on HMM and Linguistics. University of Natural Sciences, Ho Chi Minh, Vietnam.

[6] Bac Hoai Le, Thai Hoang Le, Kiem Hoang. A Fuzzy Neural Network for Vietnamese Character Recognition. University of Natural Sciences, Ho Chi Minh, Vietnam.

[7] DuyKhuong Nguyen, TheDuy Bui (2008). Recognizing Vietnamese Online Handwritten Separated Characters.Vietnam National University, Hanoi, Vietnam.

[8] Scholarpedia. Fuzzy Neural
Network.<http://www.scholarpedia.org/article/Fuzzy_neural_network>

[9] Paul E. Keller, and Kevin L. Priddy. Artificial Neural Network. Washington: SPIE - The International Society for Optical Engineering, 2005.

[10] Anil K. Jain, Jianchang Mao, K. M. Mohiuddin (1996). Artificial Neural Networks: A Tutorial. IBM Almaden Research Center.

[11] Heaton Research. Introducing the Kohonen Neural Network . December 2007.
<http://www.heatonresearch.com/articles/6/page2.html>

[12] Neila Mezghani, and Amar Mitiche. On-line Recognition of Handwritten Arabic Characters Using a Kohonen Neural Network. 2002. <http://portal.acm.org/citation.cfm?id=856908>

[13] Dileep Dinesh. A Feature Extraction Technique based on Character Geometry for Character Recognition. Amrita School of Engineering, Kollam, Kerala, India.

[14] Heaton, Jeff T. Introduction to Neural Networks with Java. Missouri: Chestfield. November 2005.

[15] Brijesh Verma, Jenny Lu, Moutmita Ghosh, Ranadir Ghosh. A Feature Extraction Technique for Online Handwriting Recognition. University of Ballarat, Ballarat, Australia. 2004

[16] Pritpal Singh, SumitBudhiraja. Feature Extraction and Classification Techniques in O.C.R. Systems for Handwritten Gurmukhi Script – A Survey. 2011. Department of Electronics and Communication Engineering, UIET, Panjab University, Chandigarh, India.

[17] VikramadityaJakkula. Tutorial on Support Vector Machine. School of EECS, Washington State University.

[18] Ricardo Gutierrez-Osuna. Validation. March 2001.
<http://research.cs.tamu.edu/prism/lectures/iss/iss_I13.pdf>

[19] William S Noble. What is a Support Vector Machine? Nature Biotechnology 24 (2006): 1565-1566.

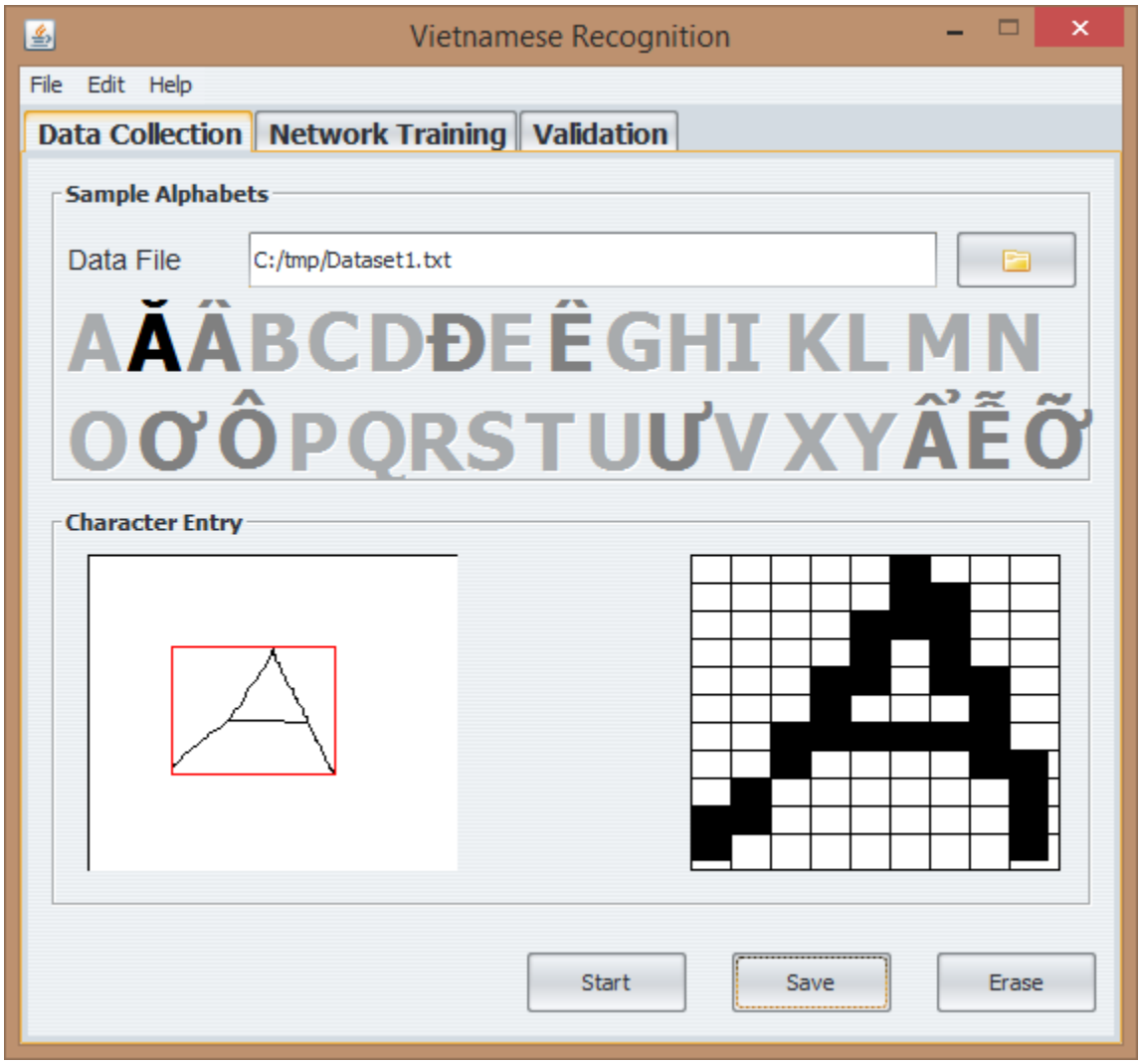
[20] Shyam M. Guthikonda. Kohonen Self-Organizing Maps. Wittenberg University. December 2005.

[21] Heaton Research. Encog Machine Learning Framework. 2013.
<http://www.heatonresearch.com/encog>

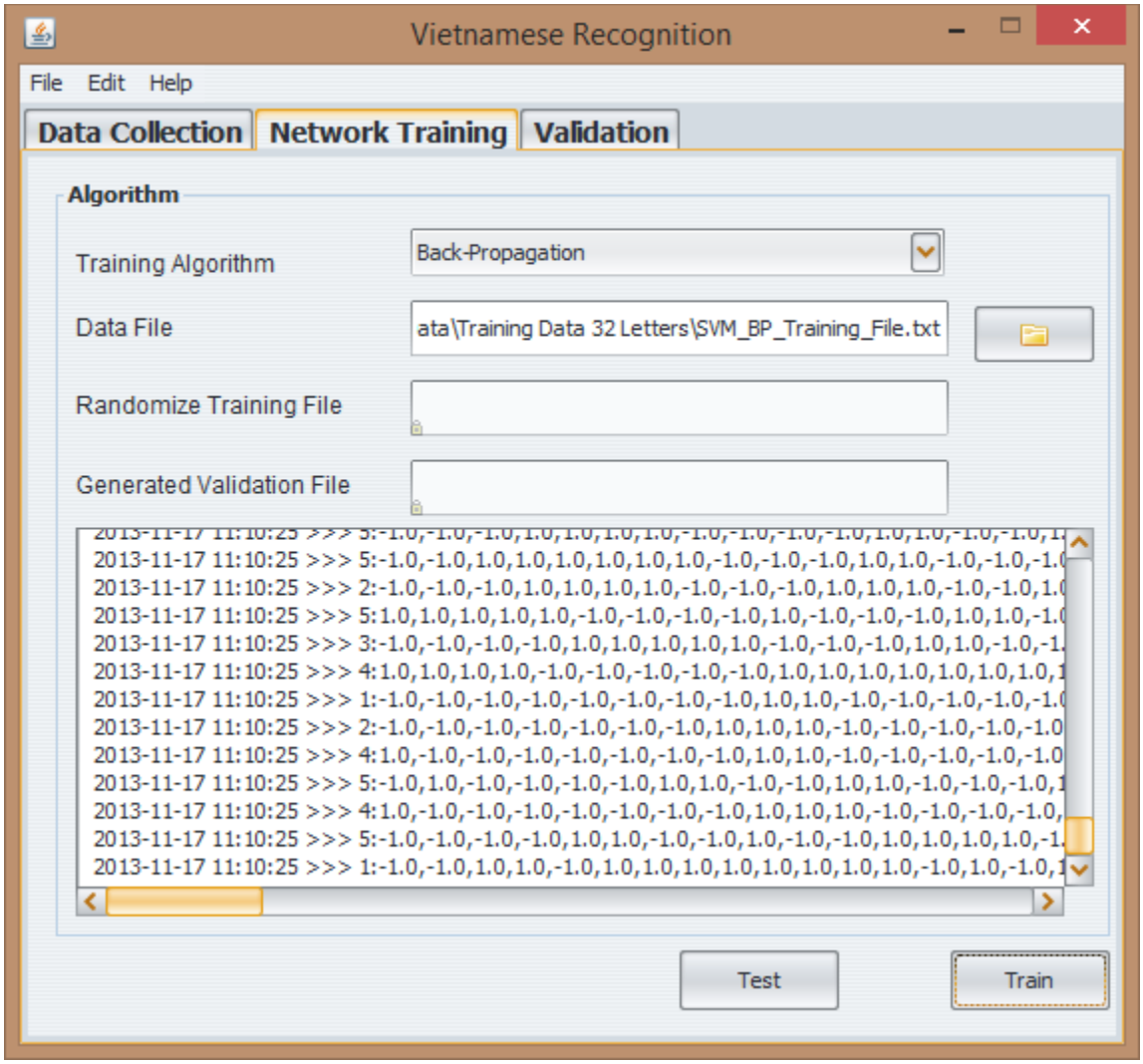
[22] John Paxton. Introduction to Neural Networks. 2003.
<http://www.cs.montana.edu/paxton/elsalvador/nn/chapter6.ppt.>

Appendix A - Recognition Application

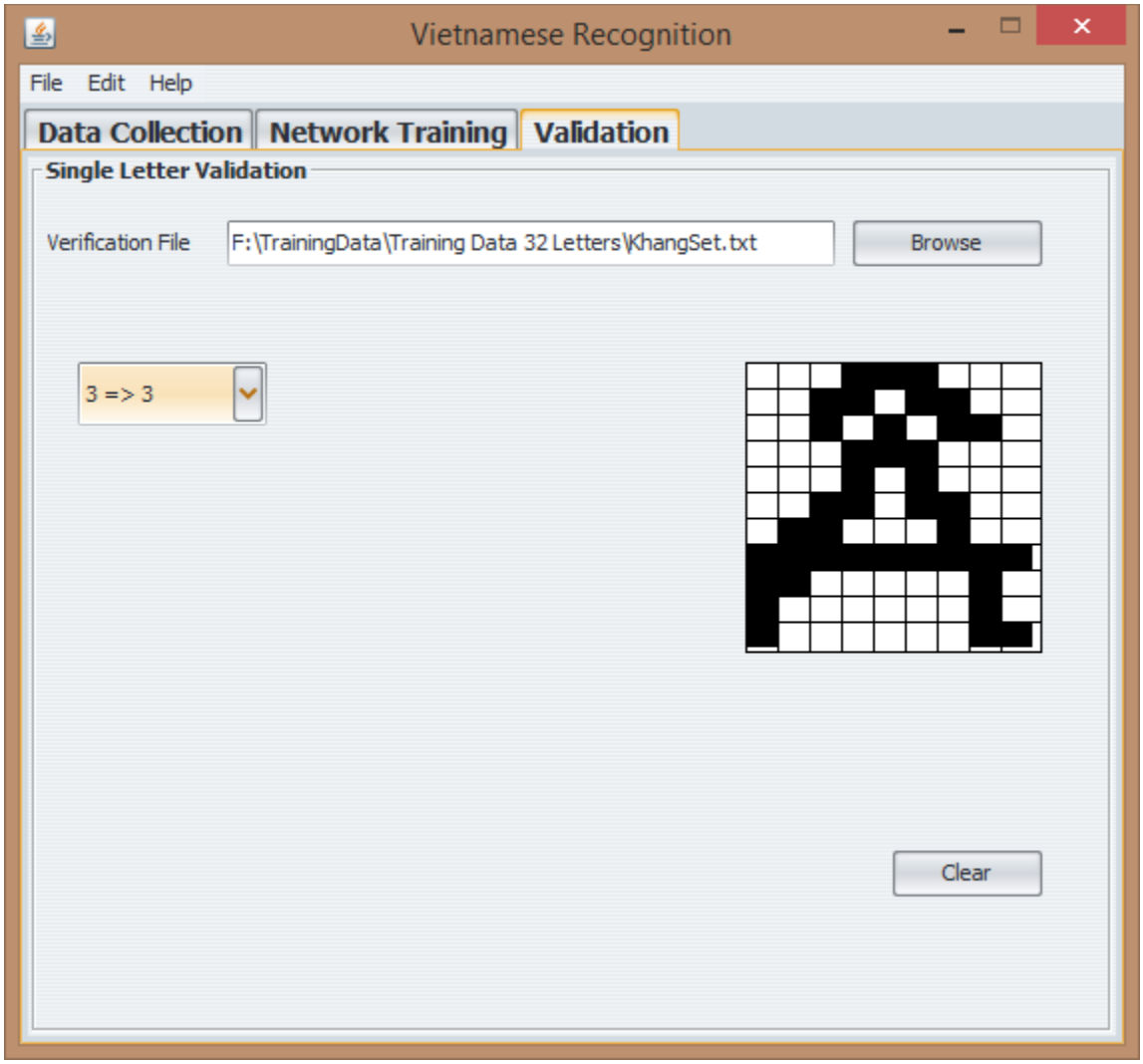
Data Collection



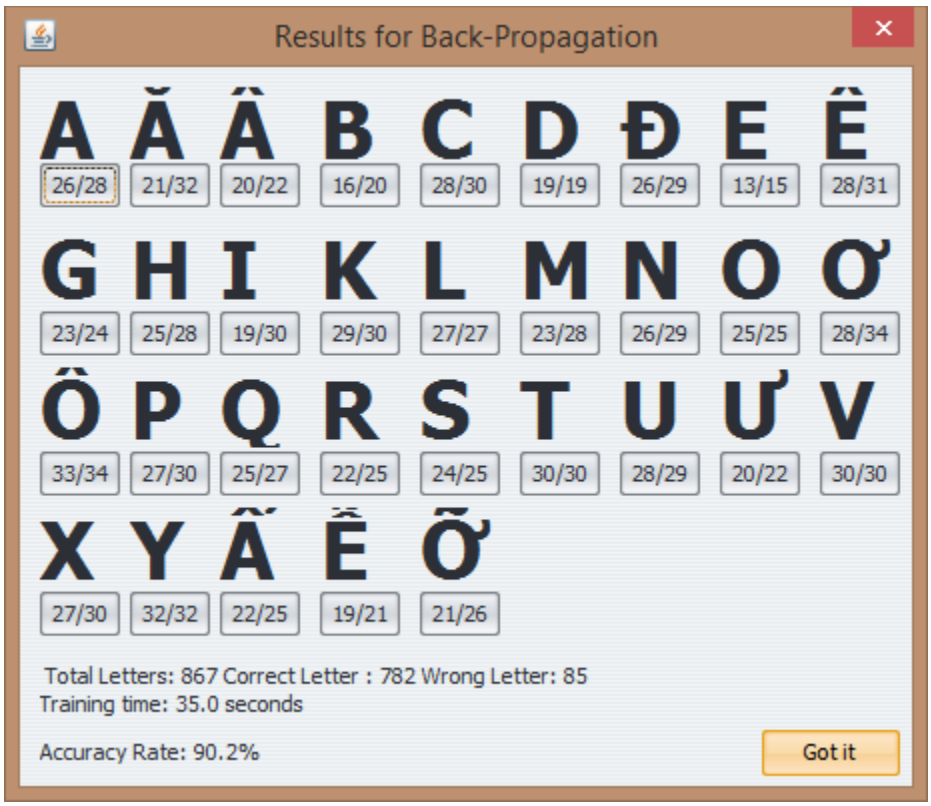
Training

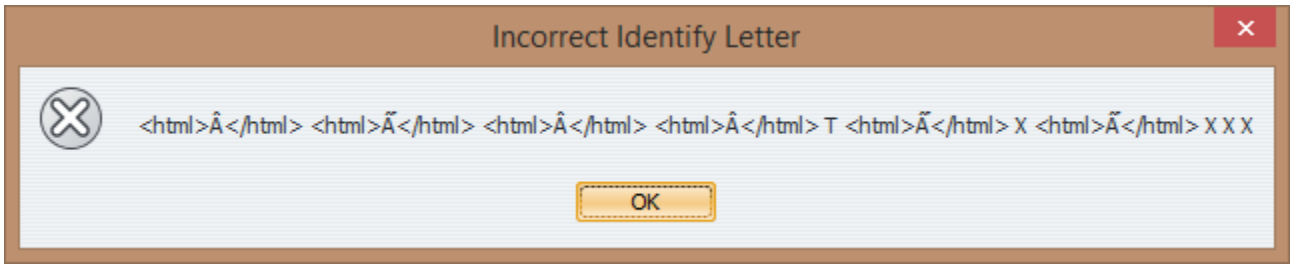

Vietnamese Recognition
File Edit Help
Data Collection
Network Training
Validation
Algorithm
Training Algorithm
Back-Propagation
Data File
ata\Training Data 32 Letters\SVM_BP_Training_File.txt
Randomize Training File
Generated Validation File
2013-11-17 11:10:25 >>> 5:-1.0,-1.0,1.0,1.0,1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.0,-1.0
2013-11-17 11:10:25 >>> 2:-1.0,-1.0,-1.0,1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,-1.0,-1.0,1.0
2013-11-17 11:10:25 >>> 5:1.0,1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0
2013-11-17 11:10:25 >>> 3:-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.
2013-11-17 11:10:25 >>> 4:1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1
2013-11-17 11:10:25 >>> 1:-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,-1.0
2013-11-17 11:10:25 >>> 2:-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,-1.0
2013-11-17 11:10:25 >>> 4:1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,-1.0
2013-11-17 11:10:25 >>> 5:-1.0,1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.0,-1.0,1
2013-11-17 11:10:25 >>> 4:1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,-1.0,1.0,1.0,1.0,-1.0,-1.0,-1.0,-1.0,
2013-11-17 11:10:25 >>> 5:-1.0,-1.0,-1.0,-1.0,1.0,1.0,-1.0,-1.0,1.0,-1.0,-1.0,1.0,1.0,1.0,1.0,-1.
2013-11-17 11:10:25 >>> 1:-1.0,-1.0,1.0,1.0,-1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,-1.0,1.0,-1.0,1
Test
Train

User Character Verification



Recognition Output





Appendix B - Sample Character Data