Neural Nets for Radio Morse code Recognizing*

Hsin Chia Fu Y.Y. Lin Hsiao-Tien Pao**
Department of Computer Science and Information Engineering
**Department of Management Science
National Chiao Tung University
Hsinchu, Taiwan 300, ROC

Abstract

This paper proposes a neural network recognition system for hand keying Radio Morse codes. The system has been trained and tested on real world data recorded from amateur radio Morse codes. The overall recognizing process can be partitioned into 3 major parts, the preprocessing, the feature extracting, and the character decoding. The whole operation is able to be performed in real-time on a PC/486 system. Self-Organizing Maps are used intensively in the recognition system to adaptively learn the variation of the Morse code signal. The average performance of the recognition system has been achieved about 96.4% with a rejection rate of 6.5%. It is hoped that many of the techniques would be applicable to a wide range of DSP and recognition tasks.

1 INTRODUCTION

Having a reliable automatic means of detecting and recognizing radio Morse codes around the clock is essential for amateur radio communication and national security. We have developed an unsupervised neural network based hand-keying Morse code recognizer, which runs automatically with an accuracy better than 96% of correctness and does not get distracted or tired.

According to the amateur radio handbook [5], Morse codes are composed of long and short *beeps* (Marks: *di & dah*) separated by three silent *Spaces* (i.e. *element*, *character*, and *word Spaces*), and the duration of these signals should maintain a fixed ratio with each other. In real world, the radio Morse codes were generated by different people with widely levels of carefulness. Thus, hand-keying Morse codes were usually listened and recognized by well trained operators. Designing a machine to recognize Morse code seems a task where "neural net" techniques are expected to be relevant, since the recognition of codes requires closely mimicking human perception, to deal with low precision data, and to learn features from examples.

Raymond C. Petit [4] built a circuit to receive radio Morse code signals and output the recognized text messages on a teleprinter. His approach was based on five rules used to discriminate the element signals, i.e., di, dah, and 3 kinds of Spaces. A good performance was reported at 25 wpm (word per minute) at a code-practicing channel: W1AW.

Larry Ashworth [1] built a "detector" to recognize the radio Morse code signals. The recognizer was implemented by a recognition program on a PC. His circuit was more complex than Raymond's and needed a PC to execute recognition programs.

*This research is supported in part by the National Science Council under Grant NSC80-0408-1009-35.

0-8194-1201-5/93/$6.00

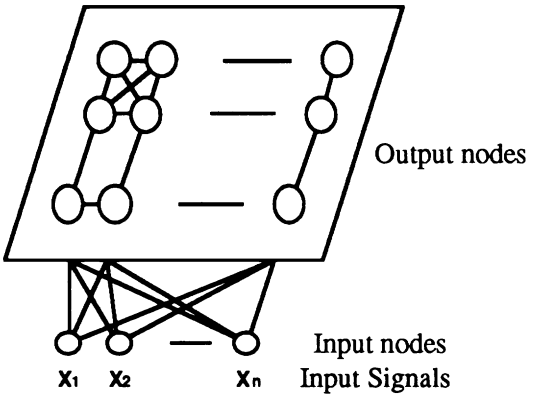

Figure 1: Self-Organizing Map model.

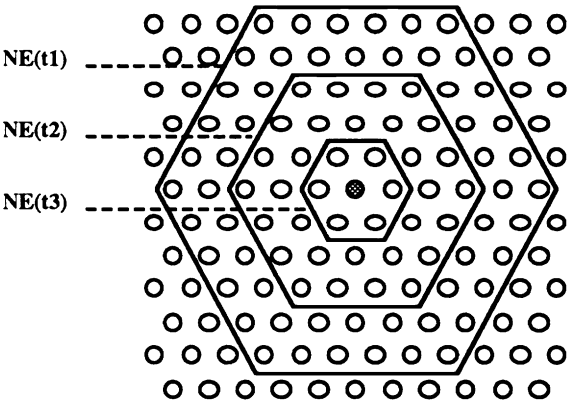

Figure 2: Neighborhood construction of an SOM.

The core concept of the proposed recognizer is based on the Kohonen's Self-Organization Feature Map to automatically generate thresholdings for Marks and Spaces form the first 40 received Morse codes. Also, since experienced operators are highly adaptive to the changing of duration length, we have adapted the LVQ techniques to update the thresholdings while recognizing each codes.

The overall recognizing process can be partitioned into 3 major parts, the preprocessing, the feature extracting, and the character decoding. In the preprocessing stage, the analog Morse codes signals are digitized by an A/D converter first, and then are sent to a PC/486 with DSP software to filter out environmental noise. After the preprocessing, the feature extraction is performed on the Self-Organization Feature Map networks (implemented by PC software) to identify each beeps to be either di, dah or Spaces. Finally, the stream of Marks and Spaces are decoded according to the Morse code book to their corresponding characters. Since the program code was optimized for execution speed, thus whole operation is able to perform in real time on a PC/486 system.

In the rest of this paper, we will brief the Kohonen's Self-organizing Map in Section 2. Then, the design of the proposed Morse code recognition system and details of generating thresholds and the feature extraction are presented in Section 3. In addition, the recognition of a character from the Morse code stream is presented in this section. Section 4 reports the experiment results, performance evaluation and comparisons with other methods. Finally, the summary and conclusions are presented in Section 5.

2 ALGORITHMIC ASPECTS OF SELF-ORGANIZING MAPS

2.1 Self-Organizing Maps

The basic structure of the SOM [2] consists of two (input and output) layers of neurons (see Figure 1). Input layer neurons are fully connected to the output layer neurons via weight-links which represent the "knowledge" that the SOM will generate through learning processes. Output neurons are related by the neighborhood-links which represent constraints during the SOM learning. As shown in Figure 2, the neighbors of the shaded neuron are the neurons enclosed in the largest hexagon at time t_1, and the neurons enclosed in the the smallest hexagon at time $t_3 > t_1$. The shape of the neighborhood assignments is not limited to hexagon, it can be either a circle or a square. The retrieving and learning scheme of the SOM are described in the followings.

Retrieving Phase Suppose that the connectivity pattern and weights of a Self-organizing Map is known, in response to inputs (patterns), the retrieving phase performs the iterative updating of activation values of each output neuron based on the following steps to produce the responding outputs:

- Step 1: Compute distance d_j between the input nodes and output node j using

$$d(y_j, \mathbf{x}) = \sum_i (w_{ji}(t) - x_i)^2. \tag{1}$$

  where $x_i(t)$ is the input to node i at time t, and $w_{ji}(t)$ is the weight from input node i to output node j at time t.

- Step 2: Select node j^* as the responding output node with the minimum distance d_j among all the output nodes (winner take all).

Learning Phase Based on the Kohonen's algorithm, the learning phase performs the iterative updating of the link weights for all the connections between input and output layers. The learning phase usually involves two steps: In the first step, the input training patterns are presented to the network as the retrieving phase, and then the responding output node j^* is selected. In the second step, the weights are updated for node j^* according to the following rules:

$$\mathbf{w_j(t+1)} = \mathbf{w_j(t)} + \eta(t)(\mathbf{x} - \mathbf{w_j(t)}), \quad \text{if } j \in NE_{j^*}(t), \tag{2}$$

$$\mathbf{w_j(t+1)} = \mathbf{w_j(t)}, \quad \text{otherwise.} \tag{3}$$

Where $\mathbf{w_j(t+1)}$ is the weights from $\mathbf{x}$ to the output node y_j, $0 < \eta(t) < 1$ is the learning rate that decreasing in time, and $NE_j(t)$ is the neighborhood assignment of neuron j, including j. The training ends when $\eta(t) = 0$.

Finally, the distribution of the output neurons will approximate the distribution of input data. If we want the distribution of the map neurons to be globally optimal, $NE_j(t)$ should be set large initially, and $\eta(t)$ reduces slowly to zero.

2.2 Fine Tuning of the Map by Learning Vector Quantization (LVQ) methods

Fine tuning of the SOM weights can be done by assigning the class that are associated with the codeword closest to the input vector. The classification accuracy can be improved if the $\mathbf{w_j(t)}$ are updated according to the Learning-VQ algorithm [3]. The idea is to adjust codebook vectors away from the decision surface to approximate the classification boundaries more accurately. Let $\mathbf{w_c(t)}$ be the codebook vector closest to $\mathbf{x}$. By applying a training vector $\mathbf{x}$ whose classification is known, we can update $\mathbf{w_c(t)}$ as follows:

$$\mathbf{w_c(t+1)} = \mathbf{w_c(t)} + \alpha(t)[\mathbf{x} - \mathbf{w_c(t)}], \quad \text{if } \mathbf{x} \text{ is correctly classified,} \tag{4}$$

$$\mathbf{w_c(t+1)} = \mathbf{w_c(t)} - \beta(t)[\mathbf{x} - \mathbf{w_c(t)}], \quad \text{if } \mathbf{x} \text{ is misclassified,} \tag{5}$$

$$\mathbf{w_i(t+1)} = \mathbf{w_i(t)}, \quad \text{for all } i \neq c. \tag{6}$$

Here, $0 < \alpha(t), \beta(t) < 1$ are learning rates decreasing in time. This algorithm tends to reduce the point density of the $\mathbf{w_i}$ around the Bayesian decision surface.

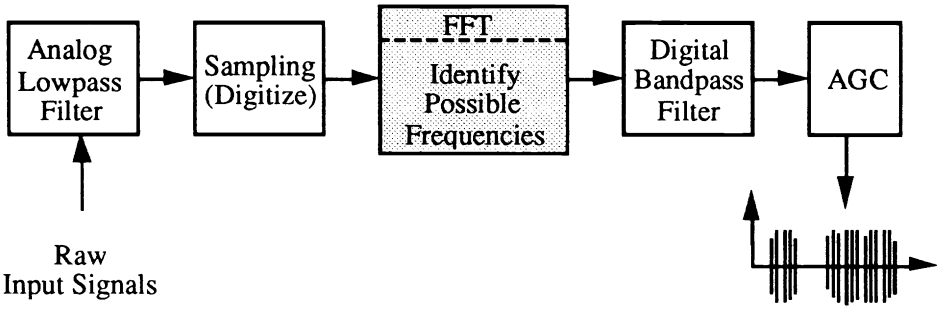

Figure 3: Signal Preprocessing.

3 THE DESIGN OF THE MORSE CODE RECOGNITION SYSTEM

The overall recognizing process can be partitioned into 3 major parts, the preprocessing, the feature extracting, and the character decoding.

3.1 Preprocessor

As shown in Figure 3, the first part of our system is the preprocessor which determines and extracts the central frequency of the input signals. First, the Fourier transform operation is performed on the radio signal, then a frequency component which has a "relative high" energy is selected as the central frequency f_i for tracing. Based on the the central frequency f_i, a bandpass filter (BPF) with a bandwidth BW is determined. The second step of the preprocessing is to extract the monitored signal using the BPF with the center frequency at f_i and bandwidth BW.

The third step of the preprocessing is to adjust the signal level, i.e. to eliminate the fading effect. Here, an Automatic Gain Controller (AGC) is used. The gain is computed by

$$G(t) = max(G_{max}, \frac{1}{\phi'_{env}(t)}). \tag{7}$$

Where G_{max} is the upper gain limit to prevent the signal from being over amplified, and $\phi'_{env}(t)$, which is computed by a rectifier envelope detector is the envelope of the bandpass filtered signal. The preprocessing steps are depicted in Figure 3.

3.2 Feature extractor

For the Morse code recognition, its features are the amplitude and the duration of the *Mark* and *Space* signals as shown in Figure 4. The amplitude is used to distinguish Marks from Spaces, and the duration is used to distinguish *di* from *dah*, and to distinguish *element*, *character*, and *word Spaces* from each other.

Similar to the concept of the Schmitt trigger [7], we use two thresholds to binarize the signal to increase its immunity to noise.

If the levels of the Mark and Space are fixed at two different values, using two fixed thresholds is sufficient to binarize the signals, as shown in Figure 5. Since the gain of the AGC is limited, when the signal level is too small to be amplified to an appropriate level, or the level difference between Marks and Spaces become too low, using two fixed thresholds would not be suitable, as depicted in Figure 6. In order to produce a proper result, two "adaptive" thresholds are used, i.e. $Th_{hi}(t)$ and

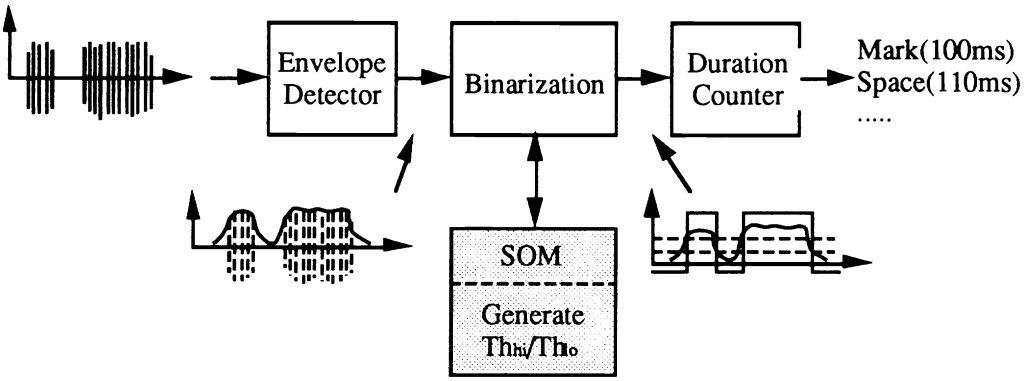

Figure 4: Feature extraction.

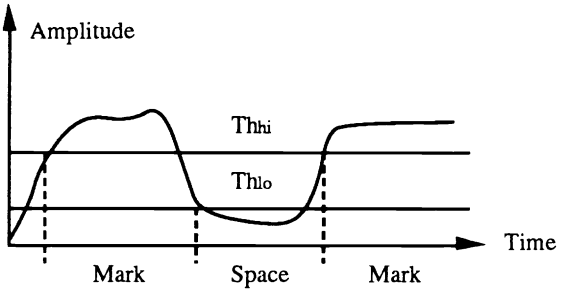

Figure 5: Fixed signal levels and fixed thresholds.

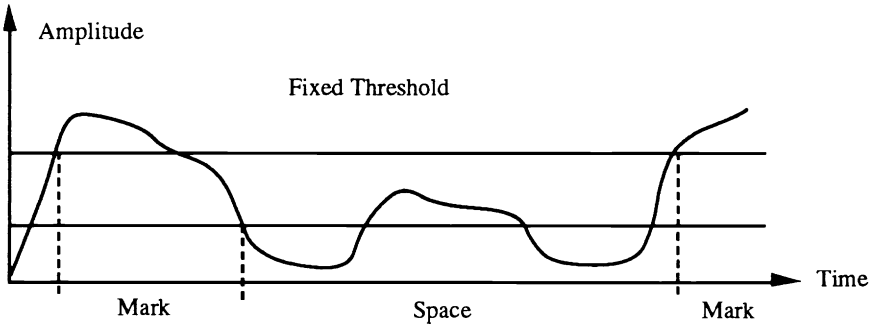

Figure 6: Varying signal levels and fixed thresholds.

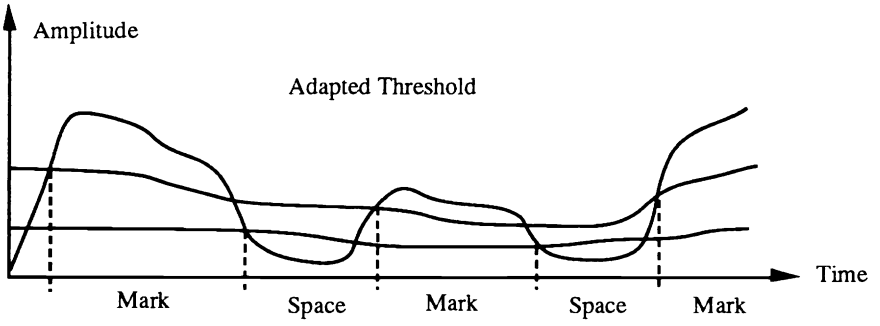

Figure 7: Varying signal levels and adaptive thresholds.

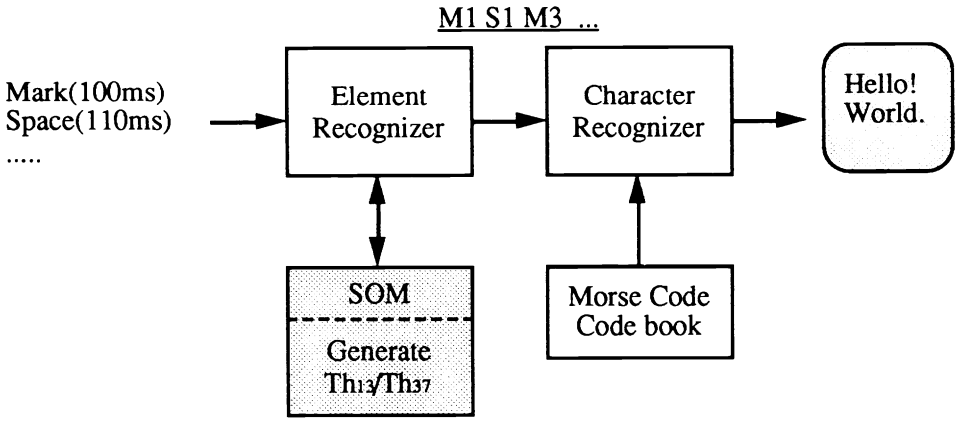

Figure 8: Element and character recognizers.

$Th_{lo}(t)$. These two thresholds will adapt themselves (using LVQ algorithm) according to the signal levels. Figure 7 depicts the extraction of a low level Mark.

Sometimes, due to a 'sudden' change in signal amplitude, the AGC in the preprocessor and the adaptive binarizer together still can not produce proper results. After two Marks were detected, the Space signal between them is checked to see if the Space signal is too short? If the Space is too short, two Marks will be bridged by reclassifying the space into a Mark, and a single longer Mark is formed. On the other hand, if a temporary noise induces a large amplitude within a Space, short Marks will be detected and they will be ironed flat to form a correct Space signal.

3.3 Recognizer

The recognizer is divided into two parts, namely the *element recognizer* and the *character recognizer* as shown in Figure 8. The element recognizer classifies the binarized signals into Marks or Spaces, and then the character recognizer matches a stream of Marks or Spaces from the element recognizer according to Morse code book to decide which character that the code stream represents.

3.3.1 Element recognizer

Due to the different operators' keying-style, keying-speed, and skill, the human-keyed duration ratio does not follow the radio Morse code rules strictly. Therefore, we need a learning scheme to adapt the actual Marks and Spaces ratio.

We use SOMs to construct the element recognizer. The weights of these SOMs are initiated according to the data received at the beginning of messages. A simple clustering algorithm is use to initialize the weight values of the SOMs, in order to speed up initializing processes and preventing misclassifying of early input data.

In the first trying of designing SOMs, two SOM nets are used to recognize Marks and Spaces respectively. One SOM generates the threshold Th_{m13} for recognizing of M1 and M3, another SOM generates the thresholds Th_{s13} for S1 and S3, and Th_{s37} for S3 and S7. Since it can not distinguish S3 from S7 good enough, we made slight changes on the SOM for recognizing Spaces. A SOM is design to generate only one threshold Th_{s13} to separate S1 from others, i.e., both S3 and S7. Then, another SOM is designed to distinguish S3 and S7 with a threshold Th_{s37} of the average length of S3 times 1.4. With this modification, the element recognizer improves greatly.

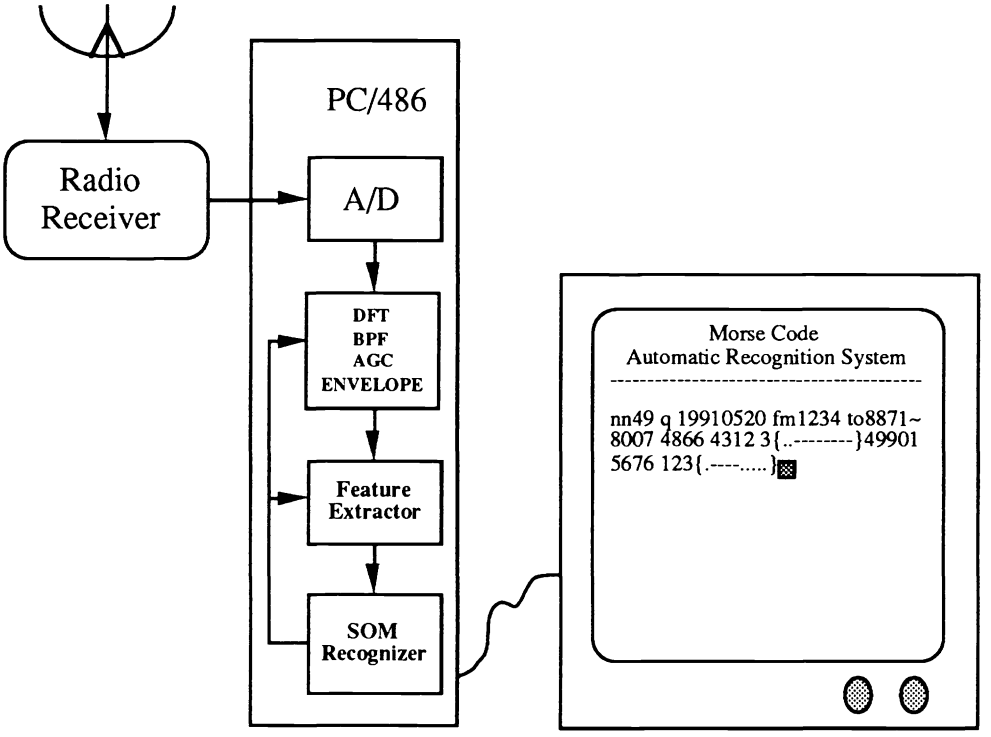

Figure 9: System diagram.

3.3.2 Character recognizer

When the element signals are identified, the recognized Marks and Spaces are collected in a buffer. Then, If an S3 (character Space) or an S7 (word space) is received, they are matched with the codes in the Morse code book.

If the character recognizer can not match the code stream with a character in the code book, we could adapt heuristic rules to "correct" some possible misclassifying Marks and Spaces in the code stream by the element recognizer, then try to match them with the code book again. This process could improve final recognition rate and also it may induce serious mistakes in the final result, since any misclassification will cause the message to be interpreted to the wrong meaning. Therefore, it is better to reject the code stream as an error, and leave it to the operator to listen and to recognize the radio Morse code signals.

The objective is not simply to maximize the number of classified codes, nor to minimize the number of errors. The objective is to minimize the error of the whole operation, which is a tradeoff between the rejection rate and the error rate.

3.4 The integrated Morse code recognition system

Figure 9 illustrates the overall system configuration. The received radio Morse codes signals are digitized at 4KHz sampling rate by an A/D converter. The digitized Morse code signals then are sent to a PC/486 for feature extraction and character recognition processings. Since the program code was optimized for execution speed, thus the whole operation was able to perform in real time.

Table 1: Five-rule method recognition rates.

Data#	Char#	Bit#	Mark-e%	Space-e%		Reject%	Char-e%	Postedit-e%	Comment
				S1-> S37	S37 ->S1				
1	108	427	0.0	0.7	2.3	15.7	1.9	2.8	
2	168	688	0.4	0.0	8.0	4.2	9.5	10.1	Difficult to p.e.
3	168	665	8.0	8.9	0.2	1.8	42.9	44.0	Unrecognizable
4	152	554	0.9	9.6	0.0	0.0	42.8	44.1	Unrecognizable
5	156	660	0.3	8.5	4.6	23.1	26.3	31.4	Unrecognizable
6	124	529	1.5	2.1	0.6	4.8	3.2	1.6	
7	152	611	0.2	0.2	3.9	19.1	3.9	3.3	Most text connected
8	176	681	5.1	6.8	0.1	2.8	35.8	35.8	Unrecognizable
9	192	726	2.5	4.4	0.1	1.6	24.0	24.5	Unrecognizable
10	136	564	0.0	1.2	1.6	14.7	3.7	2.9	
Average	153	610	1.9	4.2	2.1	8.8	19.4	20.1	Correct rate: 79.9%

4 EXPERIMENTAL RESULTS AND COMPARISONS

In this section, we present only the experiments of the element recognizer, since it is the core of the recognition system. Three different approaches include the five-rule method by Raymond [4], multi-layer perceptron (MLP) neural network, and the proposed Self-Organizing Map (SOM) neural network were simulated and their performances were compared.

4.1 Testing results of three methods

4.1.1 Five-rule element-recognizer

The five heuristic rules used to recognize the class of the elements are:
Let $m(n)$, $s(n)$, and m_{dah} denote the lengths of the n^{th} received $Mark$, the n^{th} received $Space$, and the last dah, respectively.

1. If $m(n) \geq 2 * m(n-1)$, then $m(n)$ is classified into a dah.
2. If $m(n) \leq \frac{1}{2} * m(n-1)$, then $m(n)$ is classified into a dot.
3. If $2 * m(n-1) > m(n) > \frac{1}{2} * m(n-1)$, then $m(n)$ is classified into the same class as $m(n-1)$.
4. If $s(n) \geq \frac{3}{4} * m_{dah}$, $s(n)$ is classified into a character Space.
5. If $s(n) \geq 2 * m_{dah}$, $s(n)$ is classified into a word Space.

Table 1 lists the testing results of the five-rule recognition method. In the table, 'Bit#' is the number of the Mark/space pairs. 'Mark-e%' is the Mark error rate. In Space error rate 'Space-e%', left column is the rate that an S1 is misclassified into an S3 or S7, and the right column, the rate that an S3 or S7 is misclassified into an S1. 'Reject%' is the reject rate, 'Char-e%' is the character error rate, and 'Postedit-e%' is the character error rate after manual postprocessing (rejected data are counted as errors in this column). Note that the first two error rates (Mark-e% and Space-e%)

Table 2: MLP recognition rates.

Data#	Char#	Bit#	Mark-e%	Space-e%		Reject%	Char-e%	Postedit-e%	Comment
				S1-> S37	S37 ->S1				
1	108	427	0.9	3.7	4.4	17.6	13.0	18.5	
2	168	688	1.2	0.9	1.5	7.1	4.2	8.3	
3	168	665	2.7	0.3	1.7	3.0	14.9	18.5	
4	152	554	1.1	2.7	0.5	0.0	11.2	7.2	All connected
5	156	660	0.8	7.6	2.7	7.1	25.0	23.7	All connected
6	124	529	0.4	0.4	3.6	9.7	0.8	4.8	
7	152	611	0.5	0.8	3.1	11.8	1.3	7.9	
8	176	681	1.6	0.9	3.1	7.4	10.8	17.6	
9	192	726	1.2	0.4	1.8	3.1	10.9	11.5	Average correct rate
10	136	564	0.9	4.3	1.6	9.6	13.2	11.0	
Average	153	610	1.1	2.2	2.4	7.6	10.5	12.9	87.1%

do not count the errors introduced by the preprocessor and the feature extractor, but the latter three column(Reject%, Char-e%, and Postedit-e%) have.

As shown in Table 1, some input data works well, but most of the input data are not recognizable. This phenomena could come from this method's high sensitivity to outlier's Mark and Space signals. For example, if a rather short dot is received at some time, then the following dots with normal duration will be recognized into dahs according to rule 1. Space signals have these problems, also.

4.1.2 Multilayer perceptron element recognizer

We also applied the Multilayer Perceptron (MLP) [6]to recognize the Mark and Space elements. The MLP has two layers of neurons in addition to the input layer. The input layer contains 17 nodes, the data come from the feature extractor are sequentially piped into the input nodes. The hidden layer contains 10 nodes and the output layer contains 2 (for M1, M3) or 3 (for S1, S3, and S7) nodes.

Table 2 lists the testing results on the ten data sets. The character error rate is about 10.5%, which is rather high. After post-processing, the error rate is still at 12.9%. Like the five-rule element recognizer, MLP element recognizer also makes a lot of errors which are unable to be recovered. We also tried other MLP structures with a larger scaling and recognition windows and different hidden units. However, their results are still not encouraging.

We would like to give some explanations to this result. First, the MLP may be over-trained, or there are not enough training data. Second, the structure of the MLP we used may be not suitable for this application. In summary, the behaviour of MLP neural networks can not be easily controlled and there is no systematic procedures for improving its performance currently.

4.1.3 SOM element recognizer

The recognition rate of the SOM element recognizer are listed in Table 3.

Compared to the previous two approaches, the SOM error rates are the lowest. It rejects 7.4% of the data, and makes less than 2.5% of errors. In addition, lots of the rejected data can be recovered and the post-edited error rate is about 5.4%.

Table 3: SOM recognition rates.

Data#	Char#	Bit#	Mark-e%	Space-e%		Reject%	Char-e%	Postedit-e%	Comment
				S1->S37	S37->S1				
1	108	427	0.0	0.0	2.3	18.2	1.9	7.4	
2	168	688	0.1	0.0	0.1	1.2	3.6	5.4	
3	168	665	0.2	0.0	0.0	0.0	6.5	7.7	
4	152	554	0.0	0.2	0.0	0.0	2.0	3.3	
5	156	660	0.2	1.1	2.4	20.1	1.3	5.1	Most texts connected
6	124	529	0.0	0.2	0.6	4.8	0.8	0.8	
7	152	611	0.0	0.0	1.6	10.5	0.0	5.3	
8	176	681	0.0	0.3	0.3	4.5	2.8	10.2	
9	192	726	0.0	0.1	0.0	0.0	5.7	8.3	Average correct rate
10	136	564	0.0	0.2	1.8	14.0	0.0	0.0	
Average	153	610	0.05	0.2	0.9	7.4	2.5	5.4	94.6%

4.2 Comparisons and evaluations

For element signals (Marks and Spaces), SOM has the minimum error rates. This indicates that SOMs are able to capture the signal features. For character recognition (with or without postprocessed), SOM also has the minimum error rates. For reject rates, three approaches all are about the same (five-rule method is the highest). One possible reason to this result may be the inherent ambiguity of the element signals.

For precisely keyed signals, the five-rule method is the fastest method. However it is too sensitive to outlier, which happens frequently in real word. Therefore, it is restricted for practical applications. As shown in Table 2, the results of MLP is not good. The SOM, compared to other two approaches, is the best approach to Morse code recognition, either in recognition rate or computation load. And it also has the learning ability to catch the variation of the input signals. The comparisons are summarized in Table 4.

4.3 Large data set tests

We have also tried another 26 data sets of total 6940 characters using SOM element recognizer to further benchmark its performance. The system run in real time to decode these radio signals. In average, the reject rate was about 6.5%, the character error rate was 2%. After postprocessing, the error rate was 3.6%. These values were better than previous results.

4.4 Discussions

Setting proper parameter value is very important for training and retrieving of a neural network. In SOM, the learning rates α and β are even more critical to the final recognition results. What are the best values for this purpose? Is there a range of values for α and β that will provide the network for adequate performance? We have used the same ten experiment data sets as Section 4.1 to compute the character error rate with respect to α and β that varies from 0 through 1.

As shown in Figure 10, we see that Mark signal has a rather flat distribution of error rate along the α axis. If α is set too large, the network will be very sensitive to outlier as five-rule recognizer. Therefore, a smaller value is preferred. For β parameter, when β increases, the error rare increases.

Table 4: Experiment results comparison.

	Five Rules	MLP	SOM
Sensitivity to outlier	**High**	**Depends**	**Low**
Data preparation	**No**	**Yes**	**No**
Ease of training	**No training**	**Difficult**	**Easy**
Ease of control	**Easy**	**Difficult**	**Medium**
Computation load	**Very low**	**High**	**Medium**
Test results	**Poor**	**Poor**	**Good**

Therefore, the region of the minimum error rate for β is about in $[0.05, 0.1]$, which was chosen to our previous experiment value ($\alpha = 0.1$, $\beta = 0.03$).
For Space signals, the behaviour of α is about the same as mark's. However, as we see a sharp rising of error rate when a small α value is set. This means the space signals are highly sensitive to the parameter setting. The best value is about in $[0.1, 0.3]$. Again, our previous experiment value ($\alpha = 0.1$) was chosen in this interval. For β, the Space recognition error rate increases when β increases. A good value for β can be in the range of $[0.025, 0.1]$ (experiment value was 0.02).

In these figures, there is no place where zero error rate is. This means that Mark and Space signals were ambiguous of themselves. Hence, using the SOMs alone to perform the whole recognition task is not enough. It seems that certain high level recognition techniques, such as language models and AI methodologies are needed.

5 CONCLUSIONS

An automatic Morse code recognition system was built and it could run in real time to decode the radio Morse code signals. The system could automatically trace the incoming Morse code signals with different central frequency, and reject Morse code like noisy signals. And, SOM neural networks were used intensively in generating various classification thresholds. We have obtained very good results on this difficult task. Our methods include low-precision and analog processing, massively parallel computation, extraction of biologically-motivated features, and learning from examples. We emphasize that classical engineering, signal processing, and the latest learning-from-examples methods were all absolutely necessary. Without the careful engineering, a direct adaptive network attack would not succeed, but by the same token, without learning from a very large database, it would have been excruciating to engineering a sufficiently accurate representation of the probability space.

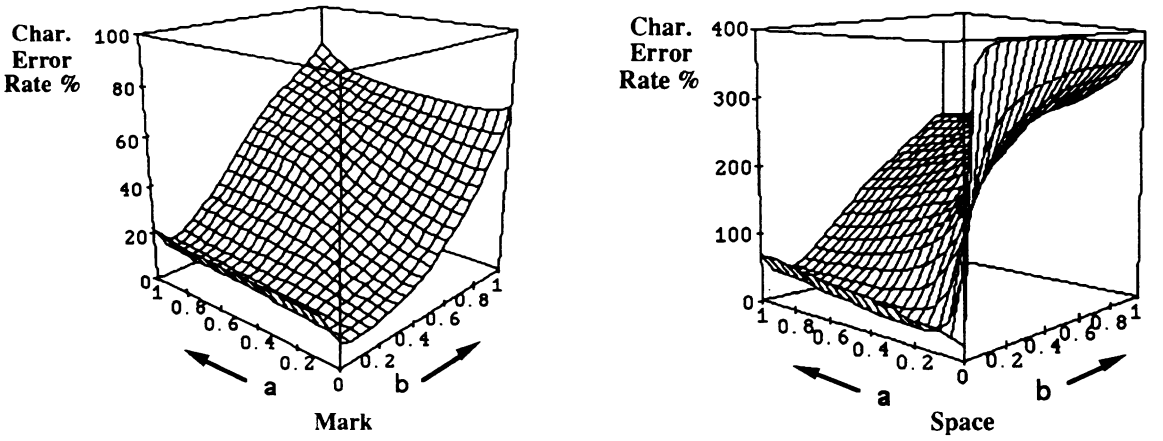

Figure 10: Character error rate vs. different SOM parameter values.

References

[1] Larry Ashworth, "Morse/RTTY Detector," Radio Electronics, Vol. 61, No. 4,5, 1990.

[2] Teuvo Kohonen, "The Self-Organizing Map," Proceedings of the IEEE, vol. 78, No. 9, pp. 1464-1480, 1990.

[3] Teuvo Kohonen, "Learning vector quantization," Neural Networks, vol. 1, suppl. 1, p. 303, 1988.

[4] Raymond C. Petit, "The Morse-A-Verter – Copying Morse Code by Machine," QST, Vol. 55, No. 1, 1971.

[5] The Radio Amateur's Handbook, 53rd ED., American Radio Relay League, Newington, April 1976.

[6] D. E. Rumelhart and G. E. Hinton and R. J. Williams, "Parallel distributed Processing (PDP): Exploration in the Microstructure of Cognition (Vol. 1)," MIT Press, Cambridge, Massachusetts, Editor: D. E. Rumelhart and J. L. McClelland and the PDP Research Group, Chapter 8, pp. 318-362, "Learning Internal Representations by Error Propagation", 1986.

[7] John F. Wakerly, "digital design princiles and practices," Englewood Cliffs, N.J.: Prentice-Hall Inc. 1990.