Iterative Decoding of *K/N* Convolutional Codes based on Recurrent Neural Network with Stopping Criterion

Johnny W. H. Kao, Stevan M. Berber
Department of Electrical and Computer Engineering, University of Auckland
jkao004@ec.auckland.ac.nz, s.berber@auckland.ac.nz

Abstract

This paper outlines a novel iterative decoding technique for a rate K/N convolutional code based on recurrent neural network (RNN) with stopping criterion. The algorithm is introduced by describing the theoretical models of the encoder and decoder. In particular this paper focuses on the investigation of a stopping criterion on the iterating procedure in order to minimize the decoding time yet still obtain an optimal BER performance. The simulation results of a rate 1/2 and 2/3 encoders respectively in comparison with the conventional Viterbi decoder are also presented.

1. Introduction

The Viterbi algorithm (VA) was proposed in 1967 [1] and it is known as an optimum method for decoding convolutional code in sequential estimation based on maximum likelihood (ML) estimation [2, 3]. Later on, the algorithm is developed into *turbo codes*, which recently becomes very popular in digital communication. A large amount of research is devoted in designing and optimizing the turbo codes in order to achieve minimum bit error rate (BER) while maintaining adequate data rate [4-6]. However, the complexity of the Viterbi algorithm increases exponentially with the number of constraint length of the encoder [7]. As a result, this algorithm becomes more challenging and resource-exhaustive for modern communication applications where the constraint length in the convolutional code can easily reach up to seven or nine [8].

Recently, researchers are finding different alternatives to VA, hoping to combat exactly this issue. This has lead to the proposal of many suboptimal decoding techniques. One of the techniques that have caught more attention is one that is based on neural networks (NN).

Neural networks (NN) are based on the biological structure of the mammalian brain. There are several properties which makes this structure very attractive for digital communication applications. Some of these properties include: highly parallelized structure, adaptive processing, self-organization, and efficient hardware implementation [8, 9]. In particularly their capabilities to solve complex non-linear problems make this structure well-suited for decoding convolutional code, which itself is a non-linear process.

Initially neural networks have only been limited to predicting errors for turbo decoders [7, 10], or to optimize and enhance the transmission protocols [7].

However, during the recent years, there is a strong emphasis on designing new decoding algorithms purely based on the advantages of neural networks. In 1996, an artificial neural net Viterbi decoder was proposed in [11]. The Viterbi Algorithm was implemented using artificial analog neurons. It was also noted that the proposed decoder fit very well for VLSI circuits. In 2000, another novel convolutional decoder is proposed using recurrent neural network [12]. It has shown that its performance approaches very close to VA and it can be easily implemented in hardware.

In 2003, two further researches are attempted on recurrent neural network (RNN) decoders [13, 14]. The preliminary results all reinforce the promising BER performance of the RNN decoder. However, due to the inherent nature of the iterative process, it is difficult to reduce the decoding time if the number of iterations has to be arbitrarily chosen for each encoder. On the other hand, it is undesirable to sacrifice the decoder's performance by restricting on an under-estimated iteration number. This issue has never been properly addressed by any previous researches.

Therefore the motivation for this paper is to implement an efficient stopping criterion on the RNN decoder that can automatically identify the minimal iteration number, which would still give a near-optimum result in the least time. Furthermore, some other parameters such as the packet size are also investigated in order to optimize the result as much as possible.

2. Theoretical Background

2.1. Theoretical Model of the Encoder

Consider a rate K/N convolutional encoder that generates a set of coded N bits for a set of K message bits at input of the encoder at time instant t, as shown in figure 1. The encoder is composed of K sub-encoders defined by their own constraint length $L_1, L_2, \ldots, L_k, \ldots, L_K$, where generally each sub-encoder can have its unique constraint length.

The bits contained in the k^{th} sub-encoder cells are denoted by $b_k(T_0+t-i_k+1)$, and $T_0 = \max_k(L_k)$, where i_k=1, 2, …L_k. Each cell in the sub-encoder is connected to each output node of the encoder through the combination of the feedback logic depicted in figure 1.

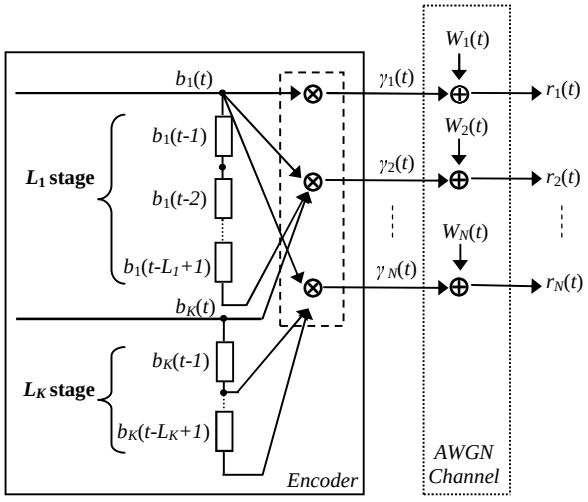

Figure 1. The structural diagram of the encoder, AWGN channel and the received data.

Hence the encoder can be represented by an impulse response matrix that contains all sub-matrices from each sub-encoder, i.e.,

$$\mathbf{g} = \left[\mathbf{g}_{L_1}^1 \quad \mathbf{g}_{L_2}^2 \quad \cdots \quad \mathbf{g}_{L_k}^k \quad \cdots \quad \mathbf{g}_{L_K}^K \right] \tag{1}$$

and the k^{th} sub-matrix, which represents the impulse response of the k^{th} sub-encoder, is expressed as

$$\mathbf{g}_{L_k}^k = \begin{bmatrix} g_{11}^k & g_{12}^k & \cdots & g_{1i_k}^k & \cdots & g_{1L_k}^k \\ g_{21}^k & g_{22}^k & \cdots & g_{2i_k}^k & \cdots & g_{2L_k}^k \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{n1}^k & g_{n2}^k & \cdots & g_{ni_k}^k & \cdots & g_{nL_k}^k \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{N1}^k & g_{N2}^k & \cdots & g_{Ni_k}^k & \cdots & g_{NL_k}^k \end{bmatrix} \tag{2}$$

To reduce the mathematical complexity, polar mapping of additive group {0, 1} is mapped into multiplicative group of {1, -1}, similar to [7] for block codes. Therefore the encoder becomes a process of mapping a N-dimensional coded vector, $\gamma_n(t) = [\gamma_0, \gamma_1, \ldots, \gamma_s, \ldots, \gamma_N]$, from a K-dimensional message vector, $b(t) = [b_1(t), b_2(t), \ldots, b_k(t), \ldots, b_K(t)]$, via

$$\gamma_n(t) = \prod_{k=1}^{K} \left[\prod_{i_k=1}^{L_k} b_k(T_0 + t + 1 - i_k)^{g_{n,i_k}^k} \right] \tag{3}$$

The received bits will also be N-bits which have been corrupted by additive white Gaussian noise.

2.2. Theoretical Model of the Decoder

The decoder's task is to find an estimate of a sequence of message bits that is the closest to the source message being sent. The problem of decoding then can be defined as a problem of finding the minimum difference between the message sequence sent and that one received, or as a minimization problem of the *noise energy function* defined as,

$$f(b) = \sum_{s=0}^{T} \sum_{n=1}^{N} \left\{ r_n(t+s) - \prod_{k=1}^{K} \left[\prod_{i_k=1}^{L_k} b_k(T_0 + t + s + 1 - i_k)^{g_{n,i_k}^k} \right] \right\}^2 \tag{4}$$

The gradient descent algorithm is employed to minimize the function by sequentially estimating a single bit from the previous estimate and the gradient of the function is used as an updating factor, i.e.,

$$b_k(t)_{new} = b_k(t) - \alpha \frac{\partial f(b)}{\partial b_k(t)} \tag{5}$$

where the last term is the gradient updating factor. The partial differential in respect to $b_k(t)$ can be derived as

$$\frac{\partial f(b)}{\partial b_{k'}(t+a)} = (-2) \sum_{s=1}^{T_0} \sum_{n=1}^{N} \left(g_{n,s_{k'}}^{k'} \right) \bullet \left[r_n(t+s+a-1) \prod_{k=1}^{K} \prod_{\substack{i_k=1 \\ i_{k'} \neq s}}^{L_k} b_k(t+s+a-i_k)^{g_{n,i_k}^k} - b_{k'}(t+a)^{g_{n,s_{k'}}^{k'}} \right] \tag{6}$$

where the variable a denotes the index of referred bit in the message sequence at decoding time t. This forms a basis of a neuron that is used for decoding through successive estimations.

3. Iterative Decoding Techniques

It is obvious that (5) implies that the decoding processing itself becomes an iterative procedure. Theoretically a larger number of iterations should always yield a more satisfactory result because the successive estimation can tend closer to the actual value. The cost however, is the long decoding time required. Therefore at this point it

becomes critical to have a suitable strategy that can minimize the decoding time without sacrificing the overall performance too much. This issue was never formally addressed before. In this section, three methods are proposed to deal with this difficulty.

3.1. Fixed iterations

The easiest method would be simply to fix on a pre-determined number of iterations on the decoder. It totally becomes a designer's decision to force all the received information to pass through the required iterations before producing any output. The advantage of this method is that the designer can have greatest control on the iterative process which is directly linked to the overall performance of the decoder. However, without prior knowledge on the behavior of the encoder, it is very easy to set on an 'overly-estimated' large number. As a result, not only the final estimation can not give the performance gain as expected for the extra iteration cycles, it may even lead to a worse outcome, compared with a smaller number of iterations. This problem is particular evident for a single input encoder as demonstrated in the simulation results shown in the latter section.

3.2. Stopping Criterion

Another approach is to set a stopping criterion prior to the iterative procedure. Once this criterion is met, the iterative estimation will terminate immediately, regardless of the iteration number. This means that each code vector will not necessarily terminate on the same number of iteration. The designer can no longer decide on the iteration number because it has becomes an unpredictable factor.

One of the criterions is defined by the following condition: terminate the iteration if two successive iterations yield the same estimate of the source message. In another word, it basically asks decoder to stop estimating if it is situated into a local minimum of the noise function. Hence adapting this criterion implies that the decoding time can be kept to a minimal while still obtain a reasonable estimate. However because this simple criterion lacks the sophistication so that the minimum point it found is usually not a good strong local minima or even the global minima of the function, therefore performance loss is inevitable. This finding is later confirmed from the simulation results.

3.3. Extension of Stopping Criterion

The two approaches mentioned above all have its own advantages and disadvantages: the first method can always yield a good result but generally requires a long decoding time whereas the second can shorten the required processing time with a degraded performance. Therefore, a method is proposed that is a combination of both approaches, or can be regarded as an extension of the second one.

While still employing the same stopping criterion, this time the decoded estimate is forced to pass through a fixed minimum number of iterations before the criterion can be triggered. From simulation results, it shows that that this minimum threshold can be a value as small as five. Furthermore, it is observed that this modification can lead to a significant performance gain compared with the previous method for only an increase of 10% decoding time. Therefore, at the time of investigation, this becomes a favorable approach that generates the best result in the least time.

4. Simulation Examples

Two encoders with rate 1/2 and 2/3, which have been analyzed in [13] and [14] respectively are used to verified the theoretical findings. Only the final formulae are presented in this paper.

4.1. Example 1: 1/2 Encoder

Consider a rate 1/2 encoder with constraint length 3, which has the impulse generator matrix:

$$\mathbf{g} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

The estimation rule can be derived as,

$$b(t)_{new} = 1/3\left[r_1(t) + r_2(t)b(t-2) + r_2(t+2)b(t+2)\right] \quad (7)$$

This update rule allows the decoder to estimate a particular bit at time *t*, using the combination of the received signal and the previous estimate. This relationship can be represented using a neuron diagram, which has been illustrated clearly in [15].

One decoding cycle is completed when every message bit has been estimated from the received sequence. In the next cycle, the most recent message estimate is combined with the same received sequence to produce another new estimate. The iterative process continues until either a fixed number has been reached or the stopping criterion has been triggered.

Parallel processing is possible for such network by connecting multiple neurons together to form a complete neural network. Thus the overall processing speed can be further increased. This is one of the distinct advantages of this algorithm compared with other conventional methods, such as the Viterbi algorithm.

4.2. Example 2: 2/3 Encoder

Another example of an encoder with a higher rate of 2/3 has the generator matrix as:

$$\mathbf{g} = \left[\mathbf{g}_{L_1} \mid \mathbf{g}_{L_2} \right] = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right]$$

Using the general formulae of (5) and (6), a specific update rule for this encoder can be developed into:

$$b_1(t)_{new} = \frac{1}{3} \begin{bmatrix} r_1(t) b_2(t-1) b_2(t-2) b_2(t-3) \\ + r_2(t+1) b_2(t+1) b_2(t-1) \\ + r_3(t+1) b_2(t+1) b_2(t-2) \end{bmatrix} \tag{8}$$

$$\begin{aligned} b_2(t)_{new} = \frac{1}{7} \big[& r_2(t) b_1(t-1) b_2(t-2) + r_3(t) b_1(t-1) b_2(t-3) \\ & + r_1(t+1) b_1(t+1) b_2(t-1) b_2(t-2) \\ & + r_1(t+2) b_1(t+2) b_2(t+1) b_2(t-1) \\ & + r_2(t+2) b_1(t+1) b_2(t+2) \\ & + r_1(t+3) b_1(t+3) b_2(t+2) b_2(t+1) \\ & + r_3(t+3) b_1(t+2) b_2(t+3) \big] \end{aligned} \tag{9}$$

This encoder is important to study because it is a desirable characteristic for a coding scheme to cope with a multi-user environment. Traditional Viterbi decoders and turbo codes are only limited to a single input and hence are less suitable for modern large communication systems. Therefore the investigation of this encoder can verify the capability of this algorithm for processing multiple simultaneous data streams and the potential for the algorithm to be applied on a multi-user network.

5. Simulation Results and Discussion

A simulated digital communication system was implemented to validate the BER performance of such decoding scheme using the two encoders mentioned in the previous section. The rate 1/2 encoder and rate 2/3 encoder will be named as encoder A and B respectively from this point on for convenience.

All simulations were conducted by calculating the BER from transferring the encoded random binary message sequences in different packet sizes through an AWGN channel and decode with the recurrent neural network (RNN) decoder. In addition, uncoded BPSK and conventional soft-decision Viterbi decoders are also implemented in the simulation as a benchmark for comparison.

5.1. Effect of Packet Size

The main objective in this simulation is to investigate the effect of different packet sizes for the RNN decoder. The simulated E_b/N_o span is from 0 to 4 dB. A total of 50k test bits are sent across, in order to truly reflect the BER in the specified SNR range, according to [16]. The packets sizes simulated were: 8, 16, 32 and 64 bits per packet. Each packet is transferred independently, and the number of iterations is fixed at 50 cycles without the stopping criterion.

The results summarized in figure 2 and 3 show that the packet size does not impose too much effect for encoder A, whereas the effect is more significant for a multi-input encoder B. A possible reason for this is because of the header registers that are in front of the message bits act as a perfect estimate for the beginning few bits. As the length of the message increases, such effect is gradually diminished, therefore lead to the growing errors.

Nevertheless, the encoder A can have a BER that is comparable and somewhat close to the soft-decision Viterbi decoder. The performance margin between the RNN decoder and the VA decoder for encoder B is also very small too (even better in some cases), in small packet sizes.

Therefore it can be concluded that the transmitted packet size will have some impact on the decoder's BER performance depending on the choice of the encoder (in particular a single input or a multi-input one). As long as the size is kept reasonable small (around 8 bits per packet), then the RNN decoder is able provide an impressive performance.

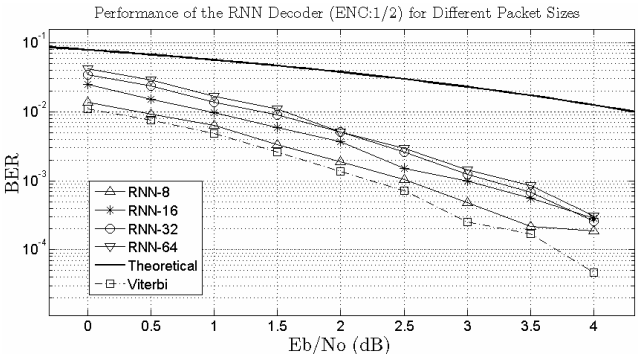

Figure 2. BER of the RNN decoder for a rate 1/2 encoder in different packet sizes.

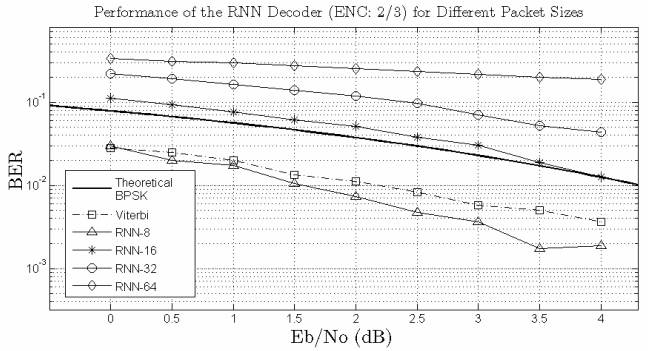

Figure 3. BER of the RNN decoder for a rate 2/3 encoder in different packet sizes.

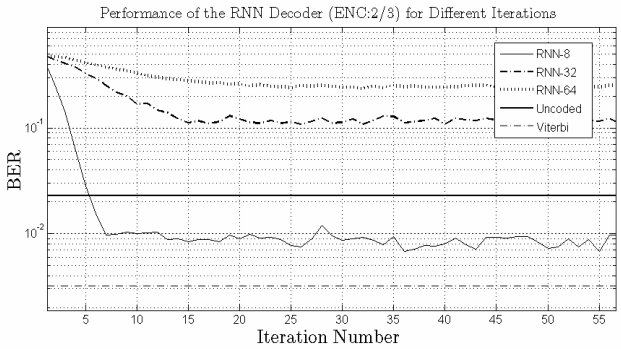

Figure 5. BER of the RNN decoder for a rate 2/3 encoder in different iterations at a SNR of 2 dB.

5.2. Effect of Different Iterations

It is critical to study the effect of different number of iteration for the RNN decoder in order to find the optimum number, which is especially useful in implementing the first method of the decoding techniques discussed earlier.

The simulation result from figure 4 indicates that for encoder A the BER drops sharply after a few initial cycles of iterations, then errors begin to rise again as iteration number continue to increase. This implies that a designer must be very cautious not to allow the iteration procedure to carry on further than the necessary value. Otherwise not only extra decoding time is wasted, more undesirable errors are bound to occur, which ends up as a 'lose-lose' situation.

This problem is not so evident for encoder B, as shown from figure 5. However encoder B would take a few more iterations than encoder A to reach a more stable BER. This is especially more obvious for larger packet sizes. After the errors have settled, then it remains reasonably constant for the rest of the iterations. Therefore this simulation vividly illustrates that it is unnecessary to locate the global minima of noise energy function, defined in (4), through iteratively estimate the message. This is because the local minima would not have too much difference with the global minima since the errors always converge to a constant after some small value of iterations.

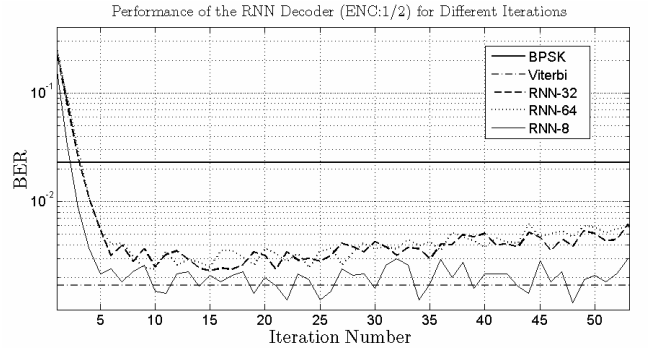

Figure 4. BER of the RNN decoder for a rate 1/2 encoder in different iterations at a SNR of 2 dB.

5.3. Effect of Stopping Criterion

This simulation is comparing the performance of employing the simple stopping criterion and the extension of this criterion. Other than the final BER, the other key aspect that is noticed from this simulation is the average iterations required to reach the stopping criterion, ending the decoding procedure. This parameter is important because it directly relates to the required processing time of the decoder.

For both encoders, the maximum number of iteration allowed is set to 30, in case that the stopping criterion (S.C.) is never reached. The extension S.C. simply adds a minimum threshold of 5 iterations before the criterion can be triggered. The packet size is fixed to 8 bits. This result can be summarized in the following table,

Table 1. Average iterations required to reach the stopping criterion for both encoders.

	Encoder A (rate 1/2)		**Encoder B** (rate 2/3)	
	Stopping Criterion	*Extension of S. C.*	*Stopping Criterion*	*Extension of S. C.*
Average Iterations	3.04	6.02	4.6	6.3

Both encoders coherently require around 10% more decoding time after the new extension of the simple stopping criterion is applied. Nevertheless, referring to figure 6 and 7, the BER has decreased significantly after this adjustment. This improvement is especially evident on a high SNR. Furthermore, for both encoders, the BER curves provided by the RNN decoder after the modification are almost identical or even better than the conventional Viterbi decoder. Therefore this small loss in the decoding time due to the threshold requirement can be easily compensated by its outstanding decoding performance.

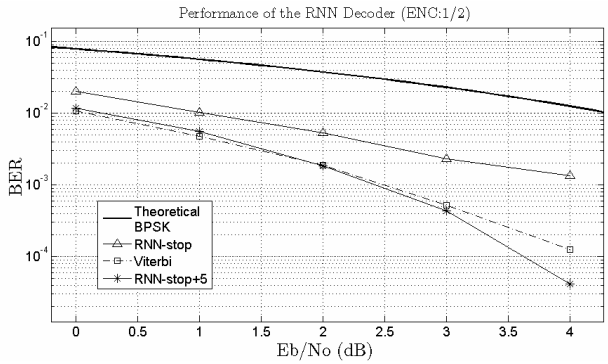

Figure 6. BER of the RNN decoder for a rate 1/2 encoder under stopping criterion and stopping criterion with a minimum of 5 iterations.

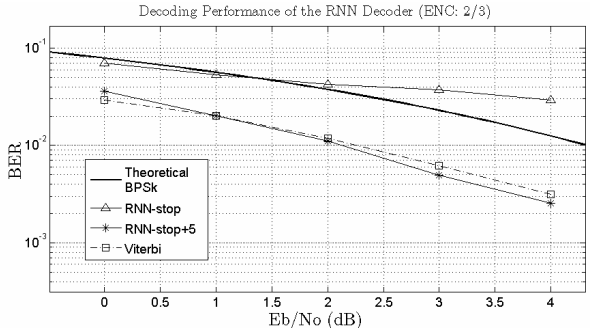

Figure 7. BER of the RNN decoder for a rate 2/3 encoder under stopping criterion and stopping criterion with a minimum of 5 iterations.

6. Conclusions

In conclusion, a novel iterative decoding technique for a general rate *K/N* convolutional code based on neural network is presented in this paper. From the results of simulation, it indicates that as long as appropriate parameters are carefully chosen: specifically a small packet size accompanied by a suitable stopping criterion, the RNN decoder has the potential to outperform the traditional Viterbi Algorithm. Moreover, this algorithm can be easily adapted for a multi-user environment as its support for multiple-input encoders. Therefore this decoding scheme may possess great values for modern and future communication systems.

7. References

[1] A. J. Viterbi, "Error bounds for convolutional codes and an aymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on*, vol. 13, pp. 260-269, 1967.

[2] D. G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R. Wall, *Coding Theory: The Essentials*. New York: Marcel Dekker Inc., 1991.

[3] S. B. Wicker and S. Kim, *Fundamentals of Codes, Graphs, and Iterative Decoding*. U.S.: Kluwer Academic Publishers, 2003.

[4] M. Jezequel and R. Pyndiah, *Turbo Codes: Error-correcting Codes of Widening Applications*. London: Hermes Penton Science, 2002.

[5] B. Bougard, A. Giulietti, and L. Van der Perre, *Turbo Codes-Desirable and Designable*. New York: Kluwer Academic Publisher, 2004.

[6] V. Branka and J. Yuan, *Turbo Codes- Principles and Applications*, 3rd ed: Kluwer Academic, 2002.

[7] M. E. Buckley and S. B. Wicker, "The design and performance of a neural network for predicting turbo decoding error with application to hybrid ARQ protocols," *Communications, IEEE Transactions on*, vol. 48, pp. 566-576, 2000.

[8] M. Ibnkahla, "Applications of neural networks to digital communications - a survey," *Signal Processing*, vol. 80, pp. 1185-1215, 2000.

[9] A. Rantala, S. Vatunen, T. Harinen, and M. Aberg, "A Silicon Efficient High Speed L = 3 rate 1/2 Convolutional Decoder Using Recurrent Neural Networks," presented at European Solid-State Device Research Conference, 2001.

[10] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary n-cube," *Information Theory, IEEE Transactions on*, vol. 35, pp. 976-987, 1989.

[11] X.-A. Wang and S. B. Wicker, "An artificial neural net Viterbi decoder," *Communications, IEEE Transactions on*, vol. 44, pp. 165-171, 1996.

[12] A. Hamalainen and J. Henriksson, "Novel use of channel information in a neural convolutional decoder," presented at Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, 2000.

[13] P. J. Secker, S. M. Berber, and Z. A. Salcic, "A generalised framework for convolutional decoding using a recurrent neural network," presented at Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on, 2003.

[14] S. M. Berber and Y.-C. Liu, "Theoretical interpretation and investigation of a 2/n rate convolutional decoder based on recurrent neural networks," presented at Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on, 2003.

[15] S. M. Berber, "Bit Error Characteristics of a soft decision output convolutional decoder that is based on the applications of neural networks," *Information Theory, IEEE Transactions on (in submission)*, 2005.

[16] S. M. Berber, "An automated method for BER characteristics measurement," *Instrumentation and Measurement, IEEE Transactions on*, vol. 53, pp. 575-580, 2004.