Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2009, Article ID 356120, 11 pages
doi:10.1155/2009/356120

*Research Article*

# A General Rate *K/N* Convolutional Decoder Based on Neural Networks with Stopping Criterion

**Johnny W. H. Kao,[1] Stevan M. Berber,[1] and Abbas Bigdeli[2]**

[1] *Department of Electrical and Computer Engineering, University of Auckland, Auckland 1142, New Zealand*
[2] *Queensland Research Laboratory, National ICT Australia, Brisbane QLD 400, Australia*

Correspondence should be addressed to Johnny W. H. Kao, j.kao@ece.auckland.ac.nz

Received 10 December 2008; Revised 9 April 2009; Accepted 30 April 2009

Recommended by Peter Tino

A novel algorithm for decoding a general rate *K/N* convolutional code based on recurrent neural network (RNN) is described and analysed. The algorithm is introduced by outlining the mathematical models of the encoder and decoder. A number of strategies for optimising the iterative decoding process are proposed, and a simulator was also designed in order to compare the Bit Error Rate (BER) performance of the RNN decoder with the conventional decoder that is based on Viterbi Algorithm (VA). The simulation results show that this novel algorithm can achieve the same bit error rate and has a lower decoding complexity. Most importantly this algorithm allows parallel signal processing, which increases the decoding speed and accommodates higher data rate transmission. These characteristics are inherited from a neural network structure of the decoder and the iterative nature of the algorithm, that outperform the conventional VA algorithm.



## 1. Introduction

With the rapid growth of digital communication systems in the recent years, lowering the power consumption and increasing the reliability of the system can both be achieved by using error-control coding (ECC). Convolutional code is a popular class of forward error-correcting codes, commonly used in space and satellite communication and digital video broadcasting [1–3]. The Viterbi Algorithm (VA) is well known to be the optimum method for decoding convolutional code based on *maximum likelihood* (ML) detection [4]. Furthermore, this algorithm is later extended into *Turbo* codes, which recently becomes very popular because of its closeness to the Shannon's limit [5–9]. A significant amount of research is devoted into optimising the turbo codes in order to achieve the minimum bit error rate (BER) [5, 6]. However, it is commonly known that the complexity of the Viterbi algorithm grows exponentially with the number of constraint length of the encoder [7, 8]. As a result, such algorithm becomes less favourable for modern communication systems where the constraint length of the encoder can easily reach up to seven or nine, making it almost impractical to implement [9].

Recently, researchers are finding different alternatives to VA, hoping to overcome this challenge. This has lead to many suboptimal decoding techniques. One of the techniques that have attracted considerable attention is one that is based on neural networks (NNs).

Neural networks (NNs) are characterised from the biological structure of the mammalian brain. There are several properties which make such structure suitable for digital communication applications. Some of these properties include highly parallelised structure, adaptive processing, self-organisation, and efficient hardware implementation [9, 10]. In particular, their capabilities to solve complex nonlinear problems make them well suited for decoding convolutional code, which itself is a nonlinear process.

Initially neural networks have only been applied for predicting errors in turbo decoders [11], or to optimise the transmission protocol [12]. However, during recent years, some work has been developed in which new decoding algorithms were proposed based purely on such structures.

In 1996, an early work of artificial neural net Viterbi decoder was investigated [13]. The Viterbi Algorithm was implemented using artificial analogue neurons. It was also noted that the proposed decoder fits very well for VLSI circuits. However, the neural network structure was still used for existing algorithms. In 2000, another convolutional decoder using recurrent neural network was developed [14]. It has shown again that the performance of neural network decoding approaches to that of VA, and it can be easily implemented in hardware.

In 2003, there was a report published that proposed a new decoding algorithm based on recurrent neural network (RNN) for a generalised rate $1/n$ convolutional encoder [15]. By deriving a general *noise energy function* with the *gradient descent algorithm* to minimise the function, the results again reinforce the promising result of RNN decoder compared with the traditional VA ones. Furthermore the computational complexity and speed of this system in comparison with the conventional VA has been investigated in that research. The author shows that the decoding complexity of a recurrent neural network decoder is only a polynomial function of the constraint length of the encoder, instead of an exponential function like in VA. Therefore the author suggests that this technique would be a good candidate to replace VA when a high constraint length encoder is required. However, the problem of developing an algorithm for a general rate $K/N$ convolutional code remains unsolved. Therefore this paper is not just an extension of the previous work, but its generalisation where a theoretical model of a $K/N$ encoder/decoder is developed. It was confirmed that the theory of $1/n$ encoder can be easily developed from this general theory for a $K/N$ encoder. Furthermore, due to this inherent nature of the iterative process, an efficient stopping criterion for optimising the trade-offs between performance gains and neural network decoding time is investigated as an extension to the previous work [15–17].

Unlike most existing convolutional decoders, which are based on trellis diagrams, the RNN decoder is a completely new approach to the decoding problem. Therefore it is important to note that the proposed method in this paper is not an implementation or optimisation of the existing trellis-based Viterbi or BCJR algorithms. Therefore all the conventional characteristics of the trellis decoding are not applicable in this paper. However, since the well-known Turbo decoder is also based on convolutional codes, the proposed RNN decoder can also be applied. The possibility of using the soft decision output of a $1/n$ neural network decoder to provide the soft output information which is required by the Turbo decoder was investigated in [16]. In addition, the stopping criterion used in this paper can also be applied for Turbo decoders or other methods of iterative decoding.

Another unique advantage of the RNN decoder is on its computational speed. Most of the modern high-speed convolutional decoders are trellis based, in which high decoding speed/throughput is achieved through introducing parallelism in the existing algorithm and careful hardware design during implementation. Two good representations of such high-speed decoders for convolutional codes can be found in [18, 19]. In those works, both implement the conventional trellis-based Viterbi Algorithm to their decoders, in which the overall speed can be increased by decoding on different segments of the received sequence in parallel. Nevertheless, as a consequence of using the trellis, the complexity would still increase exponentially if an encoder with a higher constraint length was used. In contrast, the proposed algorithm in this paper does not have such concern because it is not based on the trellis at all; hence the complexity only increases as a polynomial function (not exponential) of the constraint length of the encoder.

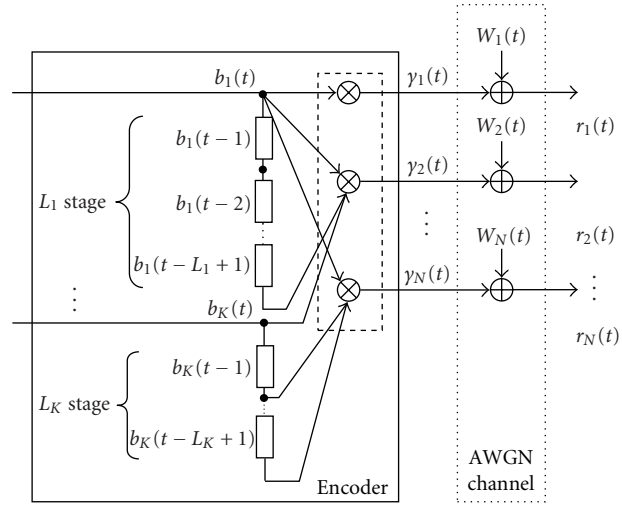


Figure 1: The structure diagram of the encoder, AWGN channel, and the received data.

This paper contains the following sections. Section 2 presents the theoretical model of a $K/N$ convolutional encoder and decoder. The iterative decoding techniques are addressed in Section 3. Section 4 describes the two examples used for simulation. All simulation result and discussions are presented in Section 5. Lastly, the conclusions are drawn in Section 6.

## 2. Theoretical Background

*2.1. Theoretical Model of the Encoder.* A general rate $K/N$ convolutional encoder, which generates a set of $N$-bit long code word for each set of $K$ message bits at the time instant $t$, is shown in Figure 1. Each input of the encoder has its own number of memory elements and hence carries different constraint lengths, defined by $L_1, L_2, \ldots, L_k, \ldots, L_K$. In another word, it can be visualised that the encoder is merely a combination of $K$ numbers of different subencoders, where each subencoder can have its own unique constraint lengths.

The bits contained in the $k$th subencoder cells are denoted by $b_k(T_0 + t - i_k + 1)$, and $T_0 = \max_k(L_k)$, where $i_k = 1, 2, \ldots, L_k$. Furthermore, each cell in the subencoder is connected to each output through the combination of a common feedforward logic depicted in Figure 1.

Therefore the encoder can be represented by a large matrix that contains all submatrices from each subencoder, shown as

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

where each submatrix corresponds to the impulse response of the $k$th subencoder and is expressed as

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

This expression is similar to one that is derived in [20].

In order to reduce the mathematical complexities, polar mapping of additive group $\{0, 1\}$ is mapped into multiplicative group of $\{1, -1\}$, similar to [11, 15].

Therefore the encoding becomes a process of mapping an $N$-dimensional coded vector, $\boldsymbol{\Gamma}(t) = [\gamma_1(t), \gamma_2(t), \ldots, \gamma_n(t), \ldots, \gamma_N(t)]$, from a $K$-dimensional message vector, $\mathbf{b}(t) = [b_1(t), b_2(t), \ldots, b_k(t), \ldots, b_K(t)]$, 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 additive white Gaussian noise (AWGN), represented by a set of noise samples, is added to the encoded set of bits generated at the time instant $t$. As the encoded set is transmitted through the channel in a serial sequence, the noise samples can be represented as independent and identically distributed random variables, $\mathbf{W}(t) = [W_1(t), W_2(t), \ldots, W_n(t), \ldots, W_N(t)]$. The received code word, which is corrupted by AWGN, is denoted by a noisy $N$-bit long code vector $\mathbf{r}(t) = [r_1(t), r_2(t), \ldots, r_n(t), \ldots, r_N(t)]$.

*2.2. Theoretical Model of the Decoder.* The main task for the decoder is to estimate a sequence of message bits that is the closest to the transmitted source message. Hence the decoding procedure can be redefined as a problem of finding the minimum difference between the message sequence to the one received. More strictly, it then becomes a problem of minimising the *noise energy* function defined as

$$\begin{aligned} f(b) &= \sum_{s=0}^{T} \|\mathbf{w}(t)\|^2 \\ &= \sum_{s=0}^{T}\sum_{n=1}^{N} \left(r_n(t+s) - \gamma_n(t+s)\right)^2 \\ &\Longrightarrow f(b) \\ &= \sum_{s=0}^{T}\sum_{n=1}^{N}\left\{ r_n(t+s) - \prod_{k=1}^{K}\left[\prod_{i_k=1}^{L_k} b_k(T_0+t+s+1-i_k)^{g_{n,i_k}^k}\right]\right\}^2. \end{aligned} \tag{4}$$

This noise energy function is a function of the coordinates of $\boldsymbol{\Gamma}$ and $\mathbf{r}$ vectors in $N$-dimensional Euclidean space. One way to minimise the function expressed above is to employ the *gradient decent algorithm* (GDA) [15]. The essence behind GDA is that by sequentially estimating each bit from the previous estimate, and the gradient of the function is used as an updating factor, that is,

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

The last term of the above expression is the partial derivative of the noise energy function in respect to $b_k(t)$, and $\alpha$ is the gradient update factor. Together they will adjust the estimate bit towards the desired value. The coefficient $\alpha$ for a $1/n$ code has been clearly explained in [17]. It is a constant that can be carefully chosen to eliminate any self-feedback, hence improving the processing speed and simplifying the overall expression.

Regarding the partial derivative, the previous investigation [15] only derives a general expression which is only suitable for a $1/n$ convolutional encoder (single input and multiple outputs). Now this expression is to be further generalised for a rate $K/N$ convolutional encoder (multiple inputs and multiple outputs), which then becomes

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

where the variable $a$ denotes the index of the referred bit in the message sequence at decoding time $t$.

Therefore, if a nonlinear activation function $f_a$ is applied into the GDA formula, then the whole equation used to estimate a single bit becomes

$$b_{k'}(t+a)_{\text{new}} = f_a\left(\frac{1}{\sum g^{k'}}\sum_{s=1}^{T_0}\sum_{n=1}^{N}\left(g^{k'}_{n,s_{k'}}\right)\times\left[r_n(t+s+a-1)\prod_{k=1}^{K}\prod_{\substack{i_k=1\\ i_{k'}\neq s}}^{L_k} b_k(t+s+a-i_k)^{g^k_{n,i_k}}\right]\right). \tag{7}$$

This is the critical formula used to estimate one message bit from a stream of received data. One cycle of iteration is completed when all parameters $b_{k'}(t+a)$ for $a = 0, 1, \ldots, T$ are updated, either in parallel or in series. Moreover, such a structure forms the basis of a neuron, which after combining together forms the basic prototype of a recurrent neural network. Parallel-processing is made possible because each neuron can function independent of each other. These concepts will be illustrated later through the simulation examples. Furthermore, the decoding complexity of this algorithm remains as a polynomial function of constraint length; therefore it can be practically implemented for a system that requires a higher order encoder for better error-control protection. The proof of complexity is detailed in Appendix A. Therefore the reduction in system complexity and parallel processing are the main advantages of this algorithm.

## 3. Iterative Decoding Techniques

The GDA formula implicitly states that the decoding process has become an iterative procedure, similar to Turbo codes. Theoretically a larger number of iterations should always yield a more satisfactory result, because the estimation should tend closer to the actual value after successive iterations. The cost on the other hand is the excess decoding time required. Therefore at this point, some strategies are investigated in order to find the optimum trade-off between the performance and time. In this section three methods are presented to deal with this problem.

*3.1. Fixed Iteration.* The most trivial method would be simply to fix on a predetermined number of iterations on the decoder, regardless of the result of estimation. All the received information is forced to pass through the required number of iterations before producing any result. The distinct advantages of this method are that it is very simple to implement, and also the overall decoding performance is totally under the designer's control. This method is most suitable for situations where the optimum decoding result is required, regardless of the processing time.

However, without sufficient trials and the prior knowledge on the behaviour of the encoder/decoder, setting an "overly-estimated" large number would not yield the extra performance gain as expected, compared to a smaller iteration, and it also wastes much unnecessary decoding time too.

*3.2. Stopping Criterion.* The second approach is to set a stopping criterion on the decoder prior to decoding. The decoding process would only terminate once this criterion has been met, regardless of the number of iterations it has gone through. This is the initial attempt to remove the constraint on the iteration number, and the decision to terminate is based on the result obtained from each iteration cycle. The main difference from the first method is that each code vector will not necessarily finish decoding at the same time; hence the time and the performance is no longer within the designer's control. The aim of using this method is to reduce the processing time required by minimising the number of iteration.

One of the criterions is defined by the following condition: terminate the decoding if two successive iterations yield the same estimate of the source message. The underlying assumption is that since two successive iterations give the same estimate, then the estimate must be very close to the actual value; hence there is no point to continue searching further. In another word, this criterion is restricting the decoder to stop searching once it found the first minimum of the noise energy function. This minimum may be a local or global minimum point. However this criterion lacks the sophistication to continue searching for the global optimum solution. Therefore some performance loss is inevitable for its small decoding time. This finding is later reinforced from the simulation results. Therefore this solution is more suitable in places where the decoding time is at a higher priority compared to its performance.

*3.3. Extension of Stopping Criterion.* From the first two approaches we realise that there is an inevitable trade-off between the desired decoding accuracy and the process time. The first method would always yield a much better result than the second but requires a much longer decoding time, especially if it is restricted to a high number of iterations, whereas the second approach can reduce the required processing time with a degraded decoding performance. In practice however, it is desirable for the decoder to hold both advantages. In this paper, a method is proposed that is not a separate approach from the previous two, but merely a combination of both, or can be regarded as an extension of the second one.

While still employing the same simple criterion on the decoder, now the decoded estimate is forced to pass through a minimum number of fixed iterations before the criterion can be triggered. The idea of this approach is raised from observing the simulation results, noticing that the errors reduce drastically after only initial few cycles of iterations. Hence this minimum threshold value can be a very small integer; thus not much time will be wasted. Furthermore, it is observed from the simulations that this slight modification can significantly increase the performance of the simple

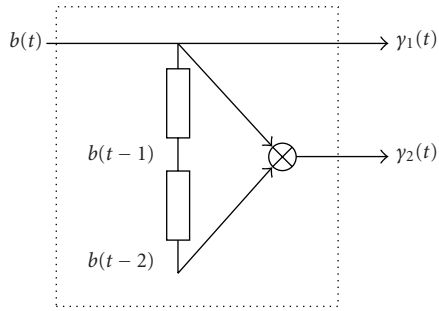


Figure 2: The structural diagram of the rate 1/2 convolutional encoder.

stopping criterion, as described previously. Therefore, at the time of investigation, this becomes a favourable approach that yields the best trade-off in terms of the decoding performance and the processing time.

## 4. Simulation Examples

In order to verify the theoretical decoding capability of the RNN decoder, two encoders with rate 1/2 and 2/3 are investigated.

*4.1. Example 1: 1/2 Encoder.* Consider a simple rate 1/2 convolutional encoder with a constraint length of 3, which has the impulse generator matrix

$$\mathbf{g} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}. \tag{8}$$

The structure of this encoder is illustrated as in Figure 2.

Using the general GDA formula with the specified value of $K = 1$, $N = 2$, and $T_0 = 3$, the update rule for this particular encoder can be expressed as

$$b(t+a)_{\text{new}} = f_a\Big(\frac{1}{3}r_1(t+a) + \frac{1}{3}r_2(t+a)b(t+a-2) + \frac{1}{3}r_2(t+a+2)b(t+a+2)\Big). \tag{9}$$

Using the combination of the received soft values as inputs and the previous estimate, the update rule allows the decoder to estimate a particular bit at time $t$. One iteration will be completed when all the information bits ($a = 0, 1, \ldots, T$) have been processed sequentially throughout the entire sequence. The relationship between the received code word and the estimated message word can be depicted using a neuron diagram shown in Figure 3.

The place where multiplication occurs is marked with a circle in the neuron diagram. Each estimated bit needs to pass through a sigmoid activation function before feeding back into the network for the next iteration. This process carries on before it is terminated either by reaching to a fixed number of iterations or by triggering a stopping criterion as mentioned above.

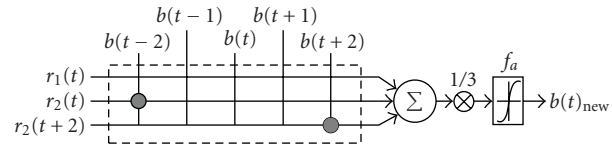


Figure 3: The neuron diagram of the RNN decoder in Example 1.

In addition, the processing speed can be further increased if multiple neurons are used to estimate the entire message sequence concurrently. This parallel signal processing ability can thus accommodate a higher data transmission rate.

*4.2. Example 2: 2/3 Encoder.* The second example is of an encoder with a higher rate of 2/3, which has the impulse generator matrix as

$$\mathbf{g} = [\mathbf{g}_{L_1} \mid \mathbf{g}_{L_2}] = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right]. \tag{10}$$

The structural diagram of this encoder can be referred in Appendix A. Again, a specific update rule for this encoder can be derived as

$$\begin{aligned} b_1(t)_{\text{new}} = f_a\Big(\frac{1}{3}[&r_1(t)b_2(t-1)b_2(t-2)b_2(t-3) \\ &+ r_2(t+1)b_2(t+1)b_2(t-1) \\ &+ r_3(t+1)b_2(t+1)b_2(t-2)]\Big), \end{aligned}$$

$$\begin{aligned} b_2(t)_{\text{new}} = f_a\Big(\frac{1}{7}[&r_2(t)b_1(t-1)b_2(t-2) \\ &+ r_3(t)b_1(t-1)b_2(t-3) \\ &+ r_1(t+1)b_1(t+1)b_2(t-1)b_2(t-2) \\ &+ r_1(t+2)b_1(t+2)b_2(t+1)b_2(t-1) \\ &+ r_2(t+2)b_1(t+1)b_2(t+2) \\ &+ r_1(t+3)b_1(t+3)b_2(t+2)b_2(t+1) \\ &+ r_3(t+3)b_1(t+2)b_2(t+3)]\Big). \end{aligned} \tag{11}$$

As the above expression shows, for a higher rate encoder, there are more terms involved in the update formula, because the estimation of one source message also relies on the estimation of the other message. This dependent characteristic is predictable because both sources share a common encoder; hence the output of the encoder would therefore contain the mixture of information from both inputs. As expected the update rule can be portrayed as a neural model because, in essence, this decoding algorithm shares the same underlying principle as a typical error-back-propagation neural network model.

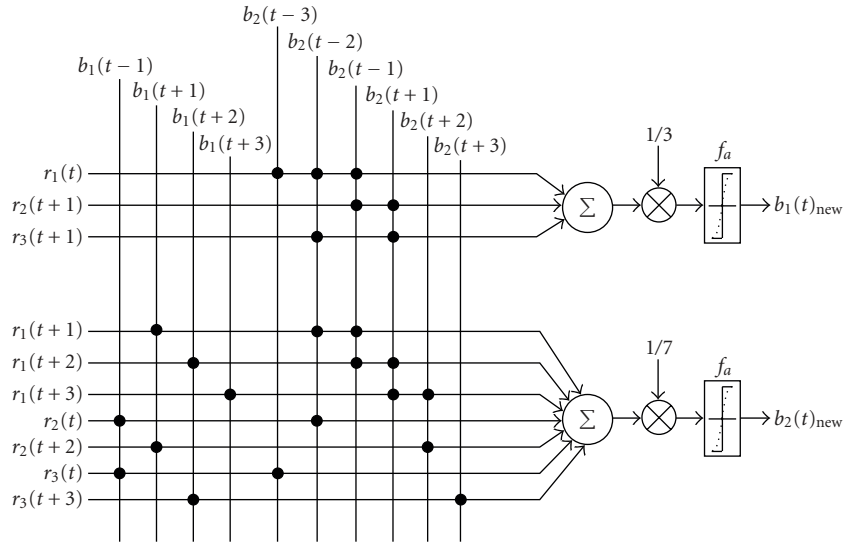


Figure 4: The neuron diagram of the RNN decoder in example 2.

Figure 4 illustrates the structure of one neuron. For a message sequence that has $T$ bits, then $T$ neurons can be constructed using the same structure except that each neuron would then possess an extra delay in every term if compared to a previous neuron. The whole neural network decoder is completed when all the neurons are connected together shown in Figure 5.

The main stages in decoding are outlined as follows. At first the receiver observes the incoming code words in parallel. In this case, it would be denoted as $r_1(t)$, $r_2(t)$, and $r_3(t)$, respectively, and each code word is $T$ bits long. Initially all the estimated bits are uniformly set to one. Then the three code words enter the neural network that also contains $T$ number of neurons. The received code words would interact with estimated message sequence, denoted by $b_1(t+a)$ and $b_2(t+a)$, as depicted in Figure 5, producing another set of newly estimated sequences. The newly estimated messages will replace the previous one, completing one cycle of decoding. The updated estimated will reinteract with the same code word until another set of updated messages replaces the current one. This iteration process continues repeatedly until the stopping criterion has been triggered by the decoder. Then the most current estimate will pass through a hard-limiting function in order to transform the soft values back into the binary symbols, before releasing as the final decoded output.

In addition, a higher code rate can increase the speed of signal transmission, as $K$ bits in parallel are fed to the encoder, which produces $N$ bits in parallel. Therefore the investigation of this encoder is of importance because it can verify the capability of this algorithm for processing multiple streams of data simultaneously. For this particular encoder, each input can represent the message source from one user, and both users share a common encoder. It would also imply that the hardware cost can be reduced when applied in practice because less number of encoders is required.

## 5. Simulation Results and Discussion

A simulated digital communication system using the two encoders described in the previous section was designed in order to verify the bit error rate (BER) performance of the RNN decoding scheme. For the sake of convenience, the rate 1/2 encoder and the rate 2/3 encoder shall be referred as encoders A and B, respectively, from this point on.

All the simulations were conducted by calculating the BER of decoding the encoded messages that are randomly generated and passed through an AWGN channel. The message sequences are transmitted in packets of different sizes. Each packet is encoded, transmitted, and decoded separately, and therefore the result from each packet can be regarded as independent. Numerous numbers of packets (in order of thousands) are required in order to truly reflect the BER in the specified signal to noise (SNR) range [21]. 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 Pack Size.* The main objective in this simulation is to investigate the effect of different packets sizes for the RNN decoder. The simulated $E_b/N_o$ span is from 0 to 6 dB. A total of more than 80000 test bits are sent across to measure the BER accurately. The packets sizes simulated were 8, 16, 32, and 64 bits per packet. Each packet is decoded for 20 cycles of iterations before producing the final result.

The simulations results are summarised in Figures 6 and 7. It shows that the packet size does not impose too much effect for encoder A (i.e., single input), whereas the effect is much more significant for encoder B (i.e., multiple inputs). At this point of research, the authors speculate that this phenomenon might be related to the initial header registers that are in front of every packet. These registers, because their states are already known prior to decoding, therefore

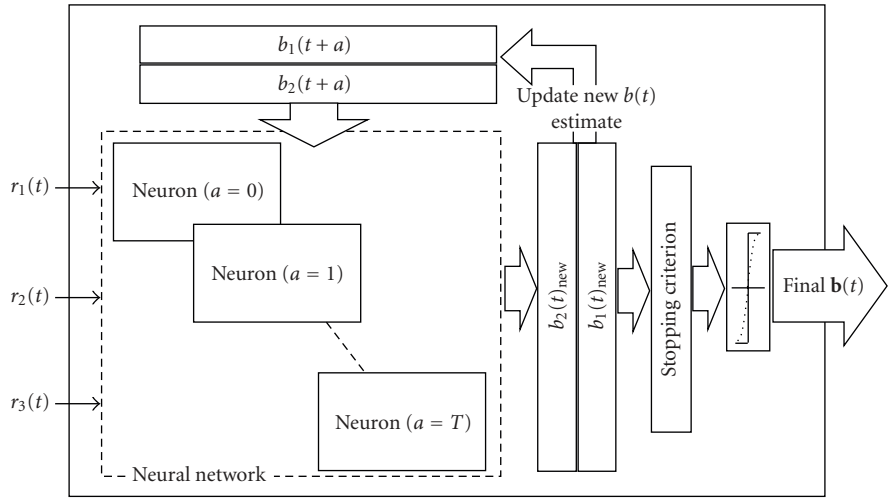


Figure 5: A conceptual diagram of the complete neural network decoder.

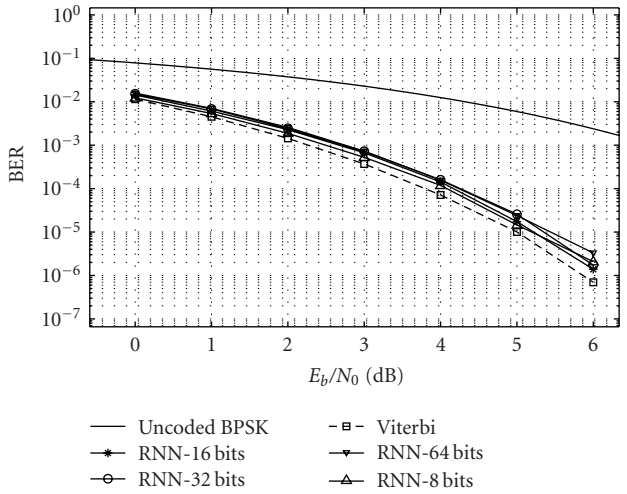


Figure 6: BER of the RNN decoder for a rate 1/2 encoder in different packet sizes.

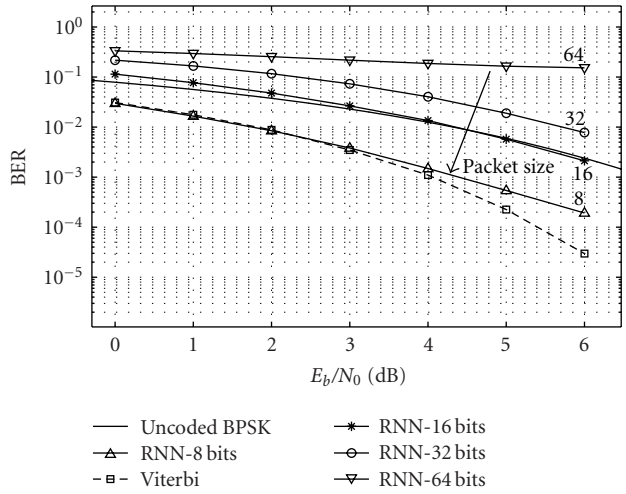


Figure 7: BER of the RNN decoder for a rate 2/3 encoder in different packet sizes.

act as a perfect guide for the initial few bits of the estimated message. As the length of the message increases however, such effect is gradually diminished, leading to more errors propagate along the later parts of the message. This problem is more noticeable for encoder B because the estimation of one message also depends on the estimation of the other; hence errors are more likely to propagate and grow much more rapidly than for encoder A.

Nevertheless, the BER for encoder A is indeed comparable to the conventional Viterbi decoder. Similarly, this simulation also shows that the performance margin between encoder B and VA is close too, when the packet sizes are relatively small.

Therefore it can be concluded that when a multiple input encoder is going to be employed into a system with neural network decoders, then the size of the transmitted packet will have an impact on the decoding accuracy. As long as the packet size is kept as small as possible, the RNN decoder can provide an impressive performance.

*5.2. Effect of Different Number of Iterations.* A simulation was carried out to observe any changes in the BER for the RNN decoder in different numbers of iterations. This information is especially useful in implementing the first method of the stopping criterion as discussed earlier. In this investigation the SNR has to be fixed at 2 dB, and then the BER were recorded for the RNN decoder undergoing different numbers of iterations.

Figure 8 shows that for encoder A the BER drops sharply after only a few initial cycles of iterations. After the drop, the errors rises again continuously on a constant slope as the number of iterations increases, which is different to what one would normally expect. This is more evident for transmission of large packets (i.e., 32 or 64 bits). This implies

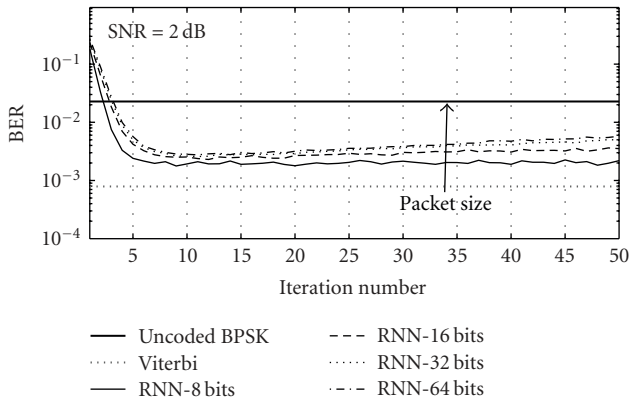


Figure 8: BER of the RNN decoder for a rate 1/2 encoder in different iterations at an SNR of 2 dB.

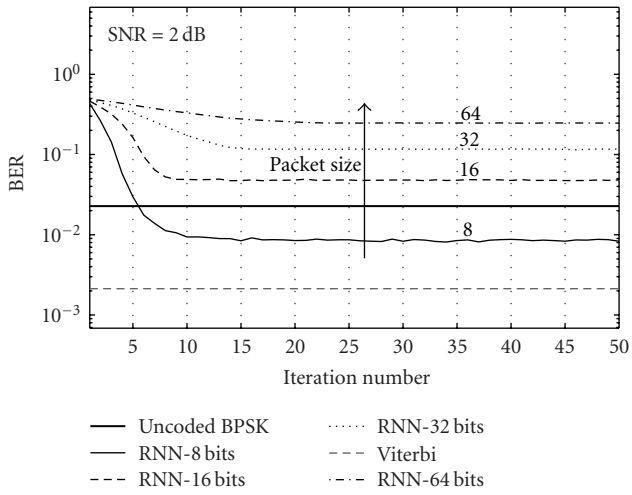


Figure 9: BER of the RNN decoder for a rate 2/3 encoder in different iterations at an SNR of 2 dB.

that designer must consider cautiously the iteration number when applying encoder A into the system. If the iteration number is set too high, then not only is extra decoding time wasted but also the decoder would produce more errors than expected, which ends up as a "double-lose" situation.

Conversely, such problem is not so obvious for encoder B, as shown in Figure 9. However the decoder needs to have a few more iterations before the BER starts converge to an approximately constant value. The result of this simulation further reinforces that the coding gain is not always proportional to the number of iterations in the decoding process. This becomes another advantage of adapting RNN decoders, because the decoding time can be further conserved by keeping just a small iteration number. This finding coincides with the results obtained from Turbo codes [22, 23], where the iteration number for the Turbo decoder is usually kept around five, since there is only small performance improvement after that.

*5.3. Effect of Stopping Criterion.* The objective of this simulation is to compare the decoding performance of employing

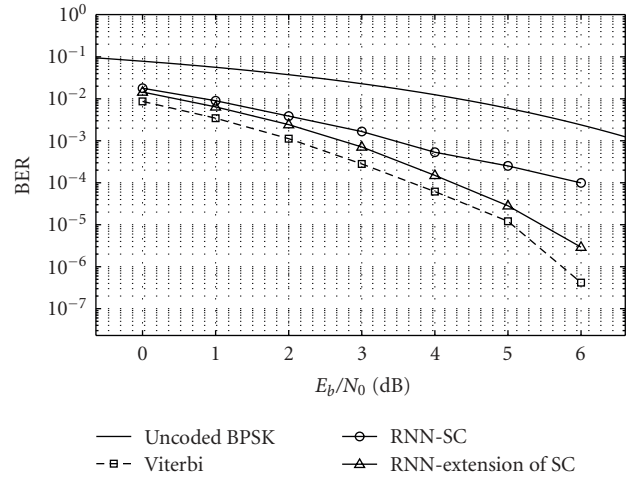


Figure 10: BER of the RNN decoder for a rate 1/2 encoder under two different stopping criterions.

the simple criterion and the extension of this criterion described in Sections 3.2 and 3.3, respectively. Other than the final BER, the average number of iterations taken before triggering the criterion and terminating the decoding is also recorded down. This parameter is important because this number is directly linked with the required processing time of decoder. Since the main goal of using the stopping criterion is to reduce the decoding time, hence ideally this parameter should be as small as possible.

For both encoders, the maximum number of iterations is restricted to 50, in case that the stopping criterion (SC) is never fulfilled, and there is no obvious performance gain beyond that point (shown from Figures 8 and 9). The extention of SC simply adds another minimum threshold of 5 iterations before SC can be triggered. This threshold value is derived from observing the simulation results, where the BER drops sharply after the first five cycles of iterations. The packet size is fixed at 8 bits. The following table summarises the results on the iteration number when adapting different stopping criterions.

When the extension of SC is applied, both decoders would need an extra few cycles before the decoding process terminates. This is reflected on about 10% increase in the decoding time. However, for the slight increase in time, the BER has decreased significantly after this modification. Figures 10 and 11 show that this improvement is more obvious at a higher SNR. Both encoders demonstrate this outstanding performance improvement achieved in the extension of SC; as a result they are all very comparable with the conventional Viterbi decoders.

The simple criterion however, due to its simplicity in the design, would terminate decoding when the estimate is located on first local minimum of the noisy energy function. Although this method uses the least time in simulation, the results are not as satisfactory as expected. The difference between SC and the extension of SC is more pronounced for encoder B (multiple inputs) than for encoder A (single input). In short, the extension of the simple stopping

Table 1: Average number of iterations required to reach the stopping criterion.

| | Encoder A (rate 1/2) | | Encoder B (rate 2/3) | |
|---|---|---|---|---|
| | Stopping criterion | Extension of SC | Stopping criterion | Extension of SC |
| Average iterations | 3.04 | 6.02 | 4.6 | 6.3 |

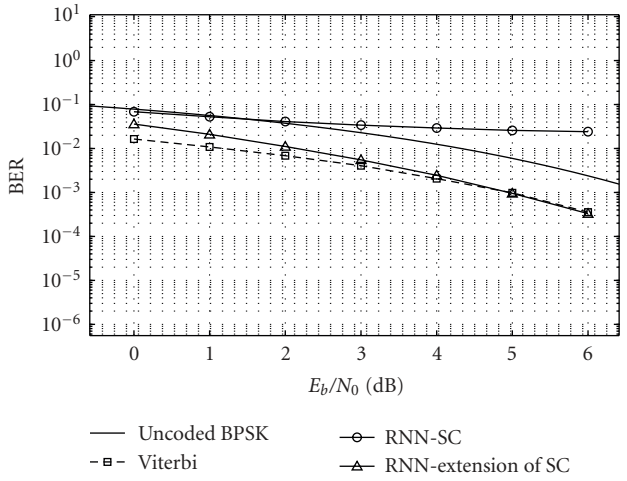


Figure 11: BER of the RNN decoder for a rate 2/3 encoder under two different stopping criterions.

criterion is an efficient way of yielding the optimal decoding accuracy in a minimum amount of time.

## 6. Conclusions

In conclusion, a mathematical model of a general rate $K/N$ convolutional encoder and recurrent neural network decoder is developed and analysed in this paper. A theoretical model of a rate $K/N$ convolutional encoder, based on noise energy function, is derived. Then the task of decoding the encoded message that is corrupted in noise reduces to a problem of minimisation of this multivariable noise energy function. The problem of decoding is solved using the gradient descent algorithm. Such algorithm can be modelled as a recurrent neural network. The complexity of such approach remains as a polynomial function of the order of the encoder, which is lower than the complexity of conventional techniques, that is, an exponential function. The independency of each neuron enables possible parallel-processing to further increase the decoding efficiency.

In order to verify the theoretical findings and performance of the RNN decoder from various aspects, simulations were carried out by using two encoders with rates 1/2 and 2/3. All results show that the packet size and the number of iterations have certain influence on the BER performance of the decoder. However, as long as the packet size is kept relatively small and employs a suitable stopping criterion, then the RNN decoder achieves a similar decoding accuracy with the traditional Viterbi decoders, with a reduction in complexity. Therefore this decoding algorithm, which has the same decoding performance and has more desirable

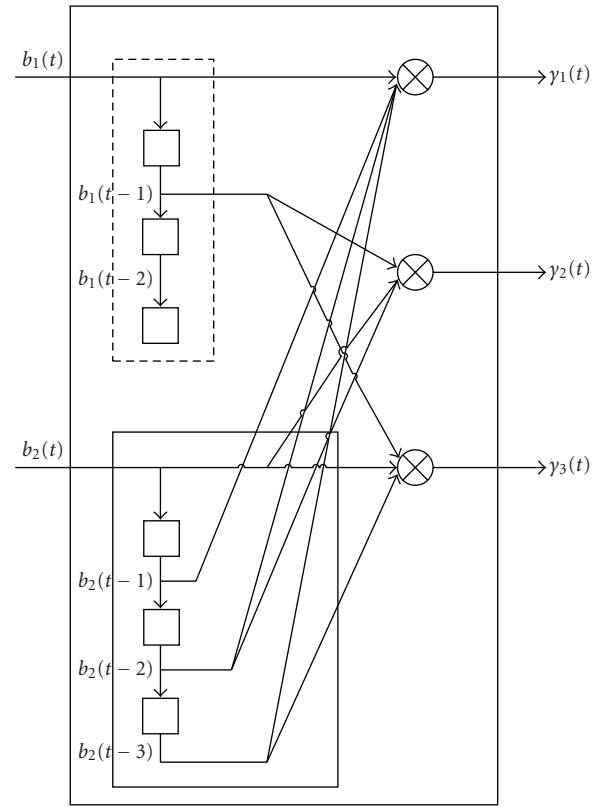


Figure 12: The structural diagram of the rate 2/3 convolutional encoder.

characteristics that are superior to the conventional decoder, will possess great potentials for future communication systems.

## Appendix

## A. The Computational Complexity of the RNN Decoder

In an earlier paper, Secker has analysed the computational complexity and speed of the RNN decoder [15]. However, that study was limited for a rate $1/n$ convolutional encoder (i.e., single-input encoders). An extension of that study has been undertaken using a similar approach, to derive the RNN complexity for a general rate $K/N$ encoder having any arbitrary number of inputs and outputs.

The first assumption is that the RNN decoder has $T + 1$ number of neurons, where $T$ is typically chosen to be about five times the constraint length of the convolutional code. For a general case, we can choose $T$ as $T_o$, which is the maximum

number of constraint length out of all subencoders. Thus the number of neurons is

$$\text{Num}_{\text{neurons}} = T + 1 = 5T_o + 1. \tag{A.1}$$

The complexity of each neuron can be examined via analysis of the update equation for each neuron as given in

$$b_{k'}(t+a)_{\text{new}} = f_a\left(\frac{1}{\sum g^{k'}}\sum_{s=1}^{T_0}\sum_{n=1}^{N}\left(g_{n,s_{k'}}^{k'}\right) \times\left[r_n(t+s+a-1)\prod_{k=1}^{K}\prod_{\substack{i_k=1\\ i_{k'}\neq s}}^{L_k} b_k(t+s+a-i_k)^{g_{n,i_k}^{k}}\right]\right), \tag{A.2}$$

where $K, N$, and $L$ denote the number of input, the number of output, and the constraint length of a subencoder.

*A.1. Number of Additions.* Equation (A.2) only shows the update equation for one input. For a $K$-input encoder, we can see there are $K(T_o \times N)$ in the summation and terms for which $g_{n,s_{k'}}^{k'} \neq 0$.

Thus, the number of addition operations required is

$$\text{Num}_{\text{add/neuron}} = \sum_{k'=1}^{K}\left[\left(\sum_{s=1}^{T_0}\sum_{n=1}^{N}\left(g_{n,s_{k'}}^{k'}\right)\right) - 1\right]. \tag{A.3}$$

In the worst case, wehre all $g_{n,s_{k'}}^{k'}$ terms are one (i.e., the encoder generator matrix, **g**, has all ones and no zeros), the number of additions per neuron is

$$\text{Num}_{\text{add/neuron}} = K(T_oN - 1) = KT_oN - K. \tag{A.4}$$

Moreover, we will assume that all subencoders' constraint lengths are equal; thus we will denote $T_o$ as $L$ from this point on. As there are $5L + 1$ neurons to be updated per iteration, the total number of additions per iteration of the network in the worst case is

$$\begin{aligned}\text{Num}_{\text{add/iter}} &= (5L+1)\times(KLN - K)\\ &= 5KL^2N - 5LK + KLN - K\\ &= L^2(5KN) + L(KN - 5K) - K.\end{aligned} \tag{A.5}$$

For a rate $1/n$ encoder (i.e., $K = 1$), the expression would fall equal to the equation obtained in [15], which is a special case of this general expression.

Thus, given a fixed number of encoder input $K$ and output $N$, the number of additions per iteration can at worst increase polynomially with the encoder constraint length, $L$. With fixed $L$, the number of additions per iteration can at worst increase linearly with $N$ and $K$. The order of addition complexity can be expressed as

$$\text{Num}_{\text{add/iter}} = O(KNL^2). \tag{A.6}$$

*A.2. Number of Multiplications.* From inspecting equation (A.2), we can see that the bulk of multiplication operations occurs from the term inside the square brackets as shown in

$$r_n(t+s+a-1)\prod_{k=1}^{K}\prod_{\substack{i_k=1\\ i_{k'}\neq s}}^{L_k} b_k(t+s+a-i_k)^{g_{n,i_k}^{k}}. \tag{A.7}$$

The number of multiplication operations required to evaluate this expression depends on the elements of the generator matrix of that subencoder, $g_{n,i_k}^{k}$, that can be calculated as

$$\sum_{k=1}^{K}\sum_{\substack{i_k=1\\ i_k\neq s}}^{L_k} g_{n,i_k}^{k}. \tag{A.8}$$

Bearing in mind that the above expression only accounts for one subencoder, therefore the total number of multiplication per neuron for $K$ number of subencoders is

$$\text{Num}_{\text{mult/neuron}} = \sum_{k'=1}^{K}\left\{1 + \sum_{s=1}^{T_o}\sum_{n=1}^{N}\left[\left(g_{n,s_{k'}}^{k'}\right)\sum_{k=1}^{K}\sum_{\substack{i_k=1\\ i_k\neq s}}^{L_k} g_{n,i_k}^{k}\right]\right\}, \tag{A.9}$$

where the +1 term is included for the multiplication by $1/\sum g^{k'}$ as in (A.2). In the worst case scenario, where all the elements of the encoder generator for all subencoders are 1, the number of multiplications per neuron is (assuming $L = T_o = L_k$, for $k = 1, \ldots, K$)

$$\begin{aligned}\text{Num}_{\text{mult/neuron}} &= K(1 + NLK(L-1))\\ &= K(1 + NKL^2 - NKL)\\ &= K + NK^2L^2 - NK^2L.\end{aligned} \tag{A.10}$$

As there are $5L + 1$ neurons to be updated per iteration, the total number of multiplications per iteration of the network is (in the worst case)

$$\begin{aligned}\text{Num}_{\text{mult/iter}} &= (5L+1)(K + NK^2L^2 - NK^2L)\\ &= 5LK + 5NK^2L^3 - 5NK^2L^2\\ &\quad + K + NK^2L^2 - NK^2L\\ &= 5NK^2L^3 - 4NK^2L^2 + KL(5 - NK) + K.\end{aligned} \tag{A.11}$$

Once again, this expression would fall equal to the equation obtained in [15], which is a special case of a $1/n$ encoder.

Thus, given a fixed number of encoder input $K$ and output $N$, the number of multiplications per iteration can at worst increase polynomially with the encoder constraint length, $L$. With fixed $L$, the number of multiplications per iteration can at worst increase linearly with $N$ and polynomially with $K$. The order of multiplication complexity can be expressed as

$$\text{Num}_{\text{mult/iter}} = O(NK^2L^3). \tag{A.12}$$

It is interesting to note from the above expression that the number of inputs, $K$, has a stronger impact on the number of multiplications required than on the number of outputs, $N$. This was not discovered in the previous work before because only a single input encoder was assumed.

## References


[1] S. Haykin, *Communication Systems*, John Wiley & Sons, New York, NY, USA, 4th edition, 2000.

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

[3] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.

[4] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[5] B. Vucetic and J. Yuan, *Turbo Codes: Principles and Applications*, Kluwer Academic Publishers, Norwell, Mass, USA, 3rd edition, 2002.

[6] M. Jézéquel and R. Pyndiah, Eds., *Turbo Codes: Error-Correcting Codes of Widening Application*, Hermes Penton Science, London, UK, 2002.

[7] A. D. Houghton, *The Engineer's Error Coding Handbook*, Chapman & Hall, London, UK, 1st edition, 1997.

[8] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*, Cambridge University Press, Cambridge, UK, 2003.

[9] M. Ibnkahla, "Applications of neural networks to digital communications: a survey," *Signal Processing*, vol. 80, no. 7, pp. 1185–1215, 2000.

[10] 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," in *Proceedings of the 27th European Solid-State Circuits Conference (ESSCIRC '01)*, pp. 441–444, Villach, Austria, September 2001.

[11] M. E. Buckley and S. B. Wicker, "The design and performance of a neural network for predicting turbodecoding error with application to hybrid ARQ protocols," *IEEE Transactions on Communications*, vol. 48, no. 4, pp. 566–576, 2000.

[12] J. Bruck and M. Blaum, "Neural networks, error-correcting codes, and polynomials over the binary $n$-cube," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 976–987, 1989.

[13] X.-A. Wang and S. B. Wicker, "An artificial neural net Viterbi decoder," *IEEE Transactions on Communications*, vol. 44, no. 2, pp. 165–171, 1996.

[14] A. Hamalainen and J. Henriksson, "Novel use of channel information in a neural convolutional decoder," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '00)*, vol. 5, pp. 337–342, Como, Italy, July 2000.

[15] S. M. Berber, P. J. Secker, and Z. A. Salcic, "Theory and application of neural networks for $1/n$ rate convolutional decoders," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 8, pp. 931–949, 2005.

[16] S. M. Berber, "Soft output decision convolutional (SONNA) decoders based on the application of neural networks," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 1, pp. 1–13, 2008.

[17] P. J. Secker, S. M. Berber, and Z. A. Salcic, "A generalised framework for convolutional decoding using a recurrent neural network," in *Proceedings of the Joint Conference of the 4th International Conference on Information, Communications & Signal Processing, and the 4th Pacific Rim Conference on Multimedia (ICICS-PCM '03)*, vol. 3, pp. 1502–1506, Singapore, December 2003.

[18] P. J. Black and T. H.-Y. Meng, "A 1-Gb/s, four-state, sliding block Viterbi decoder," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 797–805, 1997.

[19] G. Fettweis and H. Meyr, "High-speed parallel Viterbi decoding: algorithm and VLSI-architecture," *IEEE Communications Magazine*, vol. 29, no. 5, pp. 46–55, 1991.

[20] S. M. Berber and Y.-C. Liu, "Theoretical interpretation and investigation of a $2/n$ rate convolutional decoder based on recurrent neural networks," in *Proceedings of the Joint Conference of the 4th International Conference on Information, Communications & Signal Processing, and the 4th Pacific Rim Conference on Multimedia (ICICS-PCM '03)*, vol. 2, pp. 1201–1205, Singapore, December 2003.

[21] S. M. Berber, "An automated method for BER characteristics measurement," *IEEE Transactions on Instrumentation and Measurement*, vol. 53, no. 2, pp. 575–580, 2004.

[22] L. Trifina, H. G. Balta, and A. Rusinaru, "Decreasing of the turbo MAP decoding time using an iterations stopping criterion," in *Proceedings of the 7th International Symposium on Signals, Circuits and Systems (ISSCS '05)*, vol. 1, pp. 371–374, Lasi, Romania, July 2005.

[23] B.-S. Shim, D.-H. Jeong, S.-J. Lim, and H.-Y. Kim, "A new stopping criterion for turbo codes," in *Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT '06)*, vol. 2, pp. 1107–1111, Phoenix Park, Korea, February 2006.