AN IMPROVED NEURAL NETWORK-BASED DECODER SCHEME FOR SYSTEMATIC CONVOLUTIONAL CODE

A Thesis by

Andrew J. Zerngast

Bachelor of Science, Wichita State University, 2008

Submitted to the Department of Electrical Engineering and Computer Science
and the faculty of the Graduate School of
Wichita State University
in partial fulfillment of
the requirements for the degree of
Master of Science

December 2010

AN IMPROVED NEURAL NETWORK-BASED DECODER SCHEME FOR SYSTEMATIC CONVOLUTIONAL CODE

The following faculty members have examined the final copy of this thesis for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Master of Science with a major in Electrical Engineering.

_____
Hyuck Kwon, Committee Chair

_____
John Watkins, Committee Member

_____
James Steck, Committee Member

ACKNOWLEDGMENTS

I would like to thank my adviser, Hyuck Kwon, for his guidance and support. This work was partly sponsored by the Army Research Office under DEPSCoR ARO Grant W911NF-08-1-0256, and by NASA under EPSCoR CAN Grant NNX08AV84A. The views in this thesis are not the views of the Army Research Office or NASA.

ABSTRACT

This thesis explores the bit error rate (BER) characteristics of a convolutional decoder constructed of a backpropagation neural network (NN) using a newly proposed input shifting window. Due to the fact that each NN is independent and unique, multiple NNs are placed in parallel and utilize the majority rule to further improve BER performance. NNs are efficient in complex statistical systems because of their inherently fast parallel-processing speed and pattern-recognition abilities. It was found that with a code rate of ½ and a constraint length of K = 3, a NN with the optimal convolutional code generator polynomial, which is non-systematic, has poor performance and a NN with a systematic convolutional code generator polynomial shows comparable performance to a conventional hard-decision Viterbi decoder using the optimal convolutional code generator polynomial.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Neural networks (NNs) have attracted attention in communication systems due to their ability to model highly nonlinear systems, pattern recognition, and fast parallel processing speed. NNs are efficient for complex statistical systems like the communication system presented in this thesis. A neural network can be adapted to convolutional encoders with different constraint lengths without significantly increasing the complexity of the NN, while the traditional Viterbi decoder increases exponentially with an increase in constraint length.

1.1 Literature Survey

This thesis explored the use of a backpropagation NN as a convolutional decoder and introduced a new input shifting window for the NN. The input window was modified as compared to the input windows used in [1-6]. In reference [1], a backpropagation network with constraint length K=3 and code rate ½ using systematic code had better BER performance than a NN with non-systematic code. In reference [2], a backpropagation NN with a similar input shifting window as in reference [1] showed good results when decoding turbo code and approached the BER performance of a maximum a posterior (MAP) decoder. Application of the newly proposed input window to [2] with turbo code should be explored.

An iterative approach using a recurrent neural network (RNN) was explored in [3] with convolutional code and had improved BER performance over reference [2]. Reference [3] explored the use of a single RNN, as well as, a large number of RNN placed in parallel using the selection rule. In reference [3], the RNN BER performance is

comparable to the performance of a soft decision Viterbi decoder. In reference [4], an iterative approach using a RNN with two neurons was implemented and showed better BER performance than the multiple parallel RNNs used in [3]. The size of the input shifting window in [4] only used current and future received symbols and did not use past received symbols. Reference [5] used the same iterative RNN as in reference [4] and explored the effect of using different frames sizes on the RNN. The BER performance increased as frame size decreased. Placing similar iterative RNNs, as in [4,5], in parallel and using the selection rule further improved the BER performance. Applying previously decoded symbols to the input shifting window in references [4-6] will be explored in future work.

1.2 Application of NN to Communication Model

The goal of the NN in this thesis is to replace the conventional Viterbi decoder in the communication model. The NN is used in place of the Viterbi decoder and is trained to decode the code-bit signal generated with a convolutional encoder at the transmitter end. It will be shown that the NN does not have good performance with the non-systematic generator polynomial, since the NN is highly dependent on frame length due to burst errors. Extremely short frame lengths are not efficient or very useful in real-world applications. The NN is, however, not frame length-dependent when used with a systematic convolutional encoder.

1.3 Complexity Comparison

NNs are more complex at smaller constraint lengths when compared to a Viterbi decoder. Placing the NNs in parallel increases the complexity of the system. At K=3, the NN is approximately 25 times more complex than the Viterbi decoder. At K=3 for the NN

and K=7 for Viterbi decoder, the NN is approximately 1.5 more complex as shown in Table 1.1. By Moore's Law, performance of the decoder is more important than complexity concerns of the decoder. Even though the complexity of the NN is higher than Viterbi for the same constraint length, BER performance of the NN at low E_b/N_0 outperforms Viterbi.

TABLE 1.1

VITERBI VS. NEURAL NETWORK COMPLEXITY COMPARISON

Viterbi					NN		NNs in Parallel
		K=3	K=7			K=3, h=64	K=3, h=64
Multiplications	$8 * 2^{K-1}$ [8]	32	512		$2(2K+1) * h$	576	1728
Additions	$4 * 2^{K-1}$ [8]	16	256		$2(2K+1) * h$	576	1728

K: Constraint Length
h: Number of Hidden Neurons

1.4 Contribution of Thesis

This thesis contributes the following:

- Introduction of a different input shifting window into the decoder than the normal shifting window proposed in previous work [1-7]. The new shifting window is smaller, less complex, and more efficient.
- Placement of NNs in parallel using the majority rule, a design to improve bit error rate (BER) performance without regard to complexity (NNs were placed in parallel in [6] using the selection rule).

1.5 Expected Results

This thesis only attempts to improve the BER performance of a decoder with backpropagation networks [1] using a different input window into the NN and also placing NNs in parallel. By using previous estimated outputs of the network and feeding them back as inputs in the shifting window, improved BER performance with new proposed input shifting window when compared to normal input shifting window is achieved while reducing complexity. Since each NN is unique and independent, placing the NNs in parallel will further increase BER performance.

1.6 Organization

This thesis contains the following chapters. Chapter 2 provides an introduction to neural network theory and an example of training and testing a simple perceptron network. A theoretical model and explanation of a communications model is presented in Chapter 3. Chapter 4 shows how a NN can function as a convolutional decoder and how the NN is applied to the communications model. All simulation results and discussions are presented in Chapter 5. Lastly, conclusions are drawn and future work is presented in Chapter 6.

CHAPTER 2

NEURAL NETWORK INTRODUCTION

An *artificial neural network* is an information processing system that has certain performance characteristics in common with biological neural networks. Each NN consists of many nodes, or neurons, which process the information presented to it. Figure 2.1 shows a common model of a node.

The input vector, denoted by ***x***, is sent over connections to many nodes. The architecture of the connections determines the function of the NN. Each connection is associated with a weight and determines the weight vector denoted by ***w***; typically the weight vector is multiplied by the input vector.

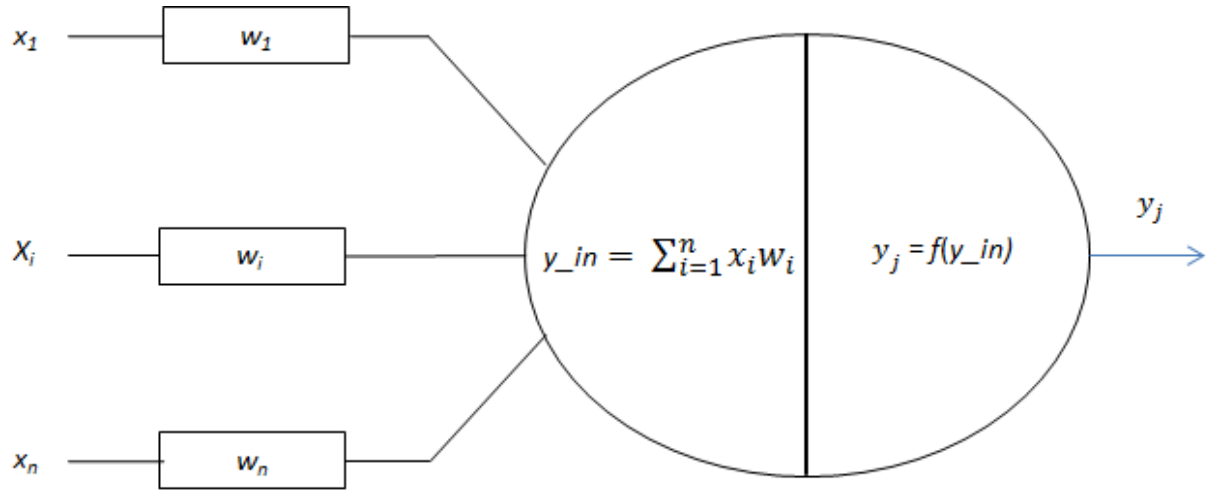


Figure 2.1 Mathematical model of node.

Each node sums the weighted input vector and applies the net input to an activation function to determine the output *y,* as shown in equations (2.1) and (2.2) [7]:

$$y\_in_j = \sum_{i=1}^{n} x_i w_i \tag{2.1}$$

$$y_j = f(y\_in_j) \tag{2.2}$$

where $f(y\_in)$ is the activation function of the node. The most commonly used activation functions are as follows:

- Binary step function (with threshold θ)

$$f(x) = \begin{cases} 1 & \text{if } x \geq \theta \\ 0 & \text{if } x < \theta \end{cases} \tag{2.3}$$

- Sigmoid function (with steepness factor σ)

$$f(x) = \frac{1}{1+e^{-\sigma x}} \tag{2.4}$$

- Hyperbolic tangent

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.5}$$

2.1 Perceptron Rule

The perceptron NN is one of the simplest and most basic examples of a neural network. It uses a feedforward iterative weight adjustment whereby the weights will converge if there are weights that exist to solve the system. For each training example in the training set, w_i is updated by

$$w_i(\text{new}) = w_i(\text{old}) + \mu \cdot (t - y) \tag{2.6}$$

where μ is the learning rate, and t is the target output of the network.

2.2 Neural Network Example

Generally small random initial weights are chosen for a neural network. Due to the fact that these weights are random, there exists the possibility of multiple networks as solutions to a given problem. Figure 2.2 is a very simple example of three networks that, after being trained, can act as an OR gate, a digital logic gate that implements logical disjunction.

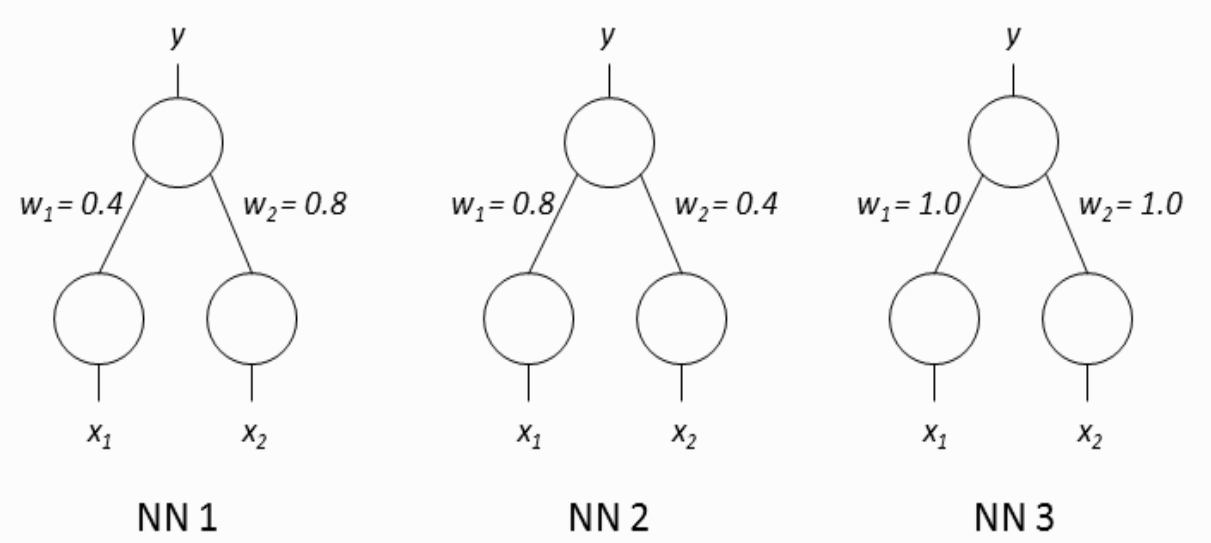


Figure 2.2 Three neural networks with different initial weights.

These networks will be trained using the perceptron rule with the learning rate $\mu = 0.2$ and binary step function as the activation function with threshold $\theta = 0.5$, and without noise, as shown in Table 2.1. The network will then be tested with a different data set to show that each network is unique and, in this case, operates differently in the presence of noise, as shown in Table 2.2.

Notice that all three networks accurately model an OR gate with no noise. In the presence of a noisy signal, NN 1 produced one error denoted in grey, while NN 2 and NN 3 produced no errors. In this example, all initial values and inputs were intentionally chosen to show the effect of noise on this system. In a real-world application, the training set would contain noisy signals to create a robust network.

TABLE 2.1

PERCEPTRON TRAINING EXAMPLE WITHOUT NOISE

NN 1 Perceptron Training							
i = iteration	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	0.4	0.8
1	0	0	0	0	0	0.4	0.8
2	0	1	1	0.8	1	0.4	0.8
3	1	0	1	0.4	0	0.6	0.8
4	1	1	1	1.52	1	0.6	1
NN 2 Perceptron Training							
i = iteration	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	0.8	0.4
1	0	0	0	0	0	0.8	0.4
2	0	1	1	0.4	0	0.8	0.6
3	1	0	1	0.8	1	0.8	0.6
4	1	1	1	1.4	1	0.8	0.6
NN 3 Perceptron Training							
i = iteration	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	1	1
1	0	0	0	0	0	1	1
2	0	1	1	1	1	1	1
3	1	0	1	1	1	1	1
4	1	1	1	2	1	1	1

TABLE 2.2

PERCEPTRON TESTING EXAMPLE WITH NOISE

NN 1 Perceptron Testing							
time	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	0.6	1
1	0	0	0	0	0	0.6	1
2	0	1.2	1	1.2	1	0.6	1
3	0.8	0	1	0.48	0	0.6	1
4	0.8	1.2	1	1.68	1	0.6	1
NN 2 Perceptron Testing							
time	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	0.8	0.6
1	0	0	0	0	0	0.8	0.6
2	0	1.2	1	0.72	1	0.8	0.6
3	0.8	0	1	0.64	1	0.8	0.6
4	0.8	1.2	1	1.36	1	0.8	0.6
NN 3 Perceptron Testing							
time	x_1	x_2	*t*	*y_in*	*y*	w_1	w_2
0	-	-	-	-	-	1	1
1	0	0	0	0	0	1	1
2	0	1.2	1	1.2	1	1	1
3	0.8	0	1	0.8	1	1	1
4	0.8	1.2	1	2	1	1	1

CHAPTER 3

SYSTEM MODEL

In communication systems, a convolutional code is a type of error correction code that adds redundant information to improve BER performance. Figure 3.1 shows a typical communications system that utilizes a convolutional code.

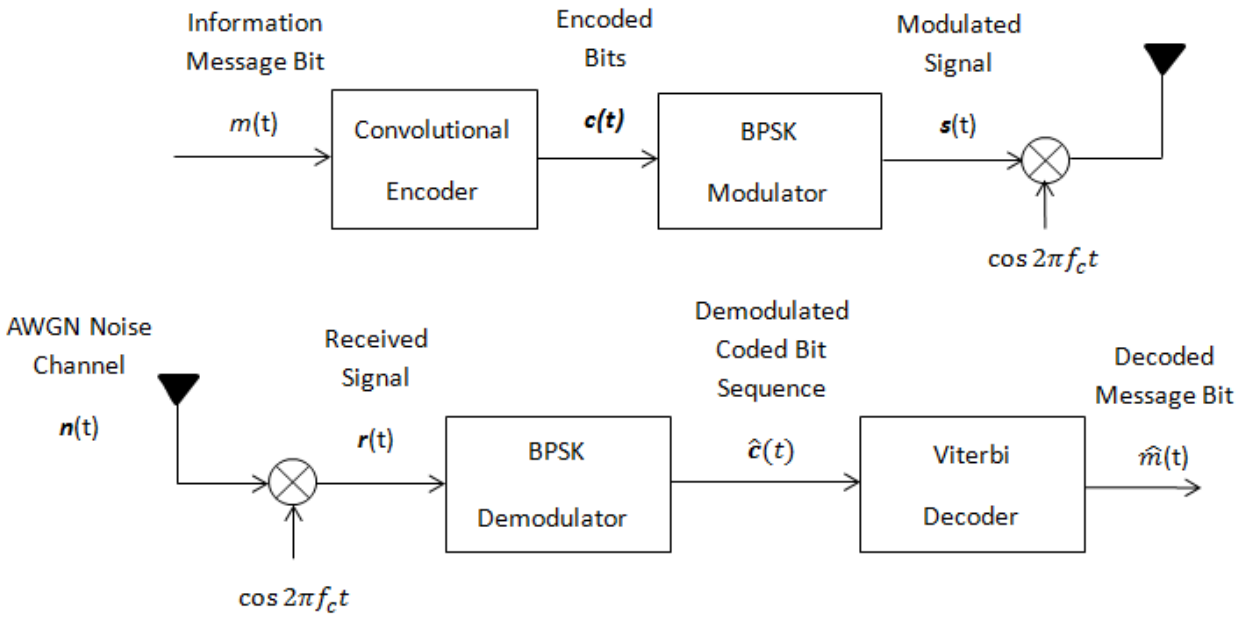


Figure 3.1 Block diagram of a communications system model.

The binary information message bits, $m(t)$, are encoded by a convolutional encoder. Coding of the information is determined by the constraint length and a generator matrix of the convolutional encoder. In this thesis, a ½ rate convolutional encoder with a constraint length of $K = 3$ was used, and two different generator matrixes, $\boldsymbol{G_1}$ and $\boldsymbol{G_2}$, were examined. The generator matrix determines the connections of the encoder, as shown in Figure 3.2.

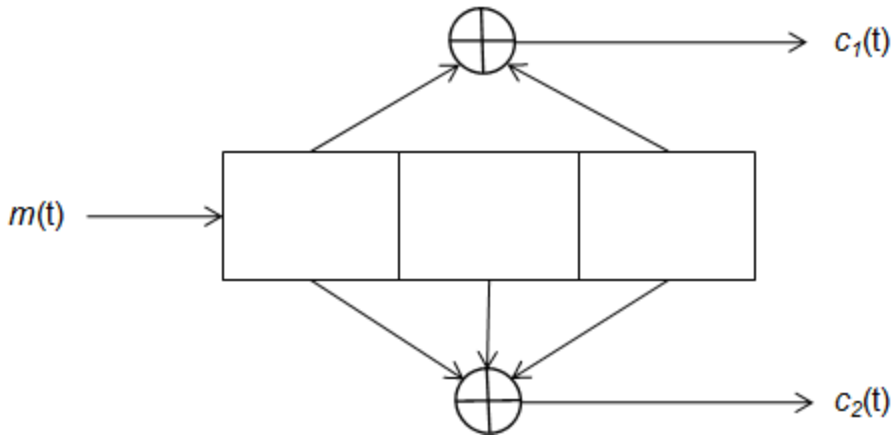


Figure 3.2 Convolutional encoder with generator matrix $\boldsymbol{G_1}$.

The coded bits, $\boldsymbol{c}$(t), are denoted by

$$\boldsymbol{c}(\mathrm{t}) = (c_1, c_2). \tag{3.1}$$

$\boldsymbol{G_1}$ is the optimal generator matrix for a ½ rate convolutional encoder when K = 3. $\boldsymbol{G_2}$ is systematic generator matrix and is examined due to the NN's poor performance with $\boldsymbol{G_1}$.

$$\boldsymbol{G_1} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{G_2} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The coded bits, $\boldsymbol{c}$(t), are sent to the binary phase-shift keying (BPSK) modulator and mapped to {-1, 1}, as shown in equation (3.3).

$$s_i(\mathrm{t}) = \begin{cases} -1 & \text{if } c(\mathrm{t}) = 0 \\ 1 & \text{if } c(\mathrm{t}) = 1 \end{cases} \tag{3.3}$$

The modulated bits $\boldsymbol{s}$(t) are up converted to the carrier frequency by multiplying $\boldsymbol{s}$(t) by $\cos(2\pi f_c t)$ and then transmitted over an additive white Gaussian noise (AWGN) channel, $\boldsymbol{n}$(t). At the receiver, the received signal $\boldsymbol{r}(\mathrm{t})$ is denoted by

$$\boldsymbol{r}(\mathrm{t}) = \boldsymbol{s}(\mathrm{t}) + \boldsymbol{n}(\mathrm{t}) \tag{3.4}$$

is down converted back to the base band by multiplying $\boldsymbol{r}$(t) by $\cos(2\pi f_c t)$ and then demodulated by mapping $r_i(t)$ to {0, 1}, as shown in equation (3.5).

$$\hat{c}_i(\mathrm{t}) = \begin{cases} 0 & \text{if } r_i(\mathrm{t}) < 0 \\ 1 & \text{if } r_i(\mathrm{t}) \geq 1 \end{cases} \tag{3.5}$$

The Viterbi decoder is used to decode $\hat{\boldsymbol{c}}$ and recover the original signal $\hat{m}$. The Viterbi decoder is a maximum likelihood decoder that has the best BER performance. The drawback of the Viterbi decoder is that complexity increases exponentially with an increase in constraint length, which makes the decoder very resource-consuming.

CHAPTER 4

APPLICATION OF NEURAL NETWORK TO COMMUNICATION SYSTEM

4.1 Neural Network as Decoder

This thesis involved exploring the use of a NN to replace the Viterbi decoder. In the communications model, the BPSK demodulator and Viterbi decoder were replaced with a codeword window block and the NN, as shown in Figure 4.1, which is a simplified representation of the model in Figure 3.1 that is used for simulation.

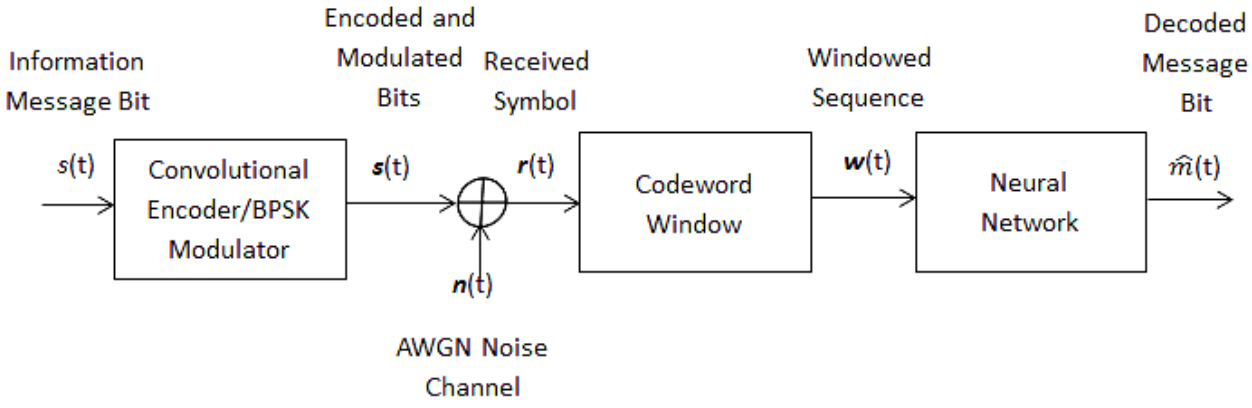


Figure 4.1 Simplified NN communications model.

Due to the pattern recognition ability of the neural networks, the NN does not use the maximum likelihood rule, but instead is taught the codeword mapping by a windowed sequence, ***w***(t). Then the problem becomes an issue of pattern recognition. Previous work [1-7] used a similar shifting codeword window where the received symbol to be decoded is centered on past and previous received symbols, as shown in Figure 4.2.

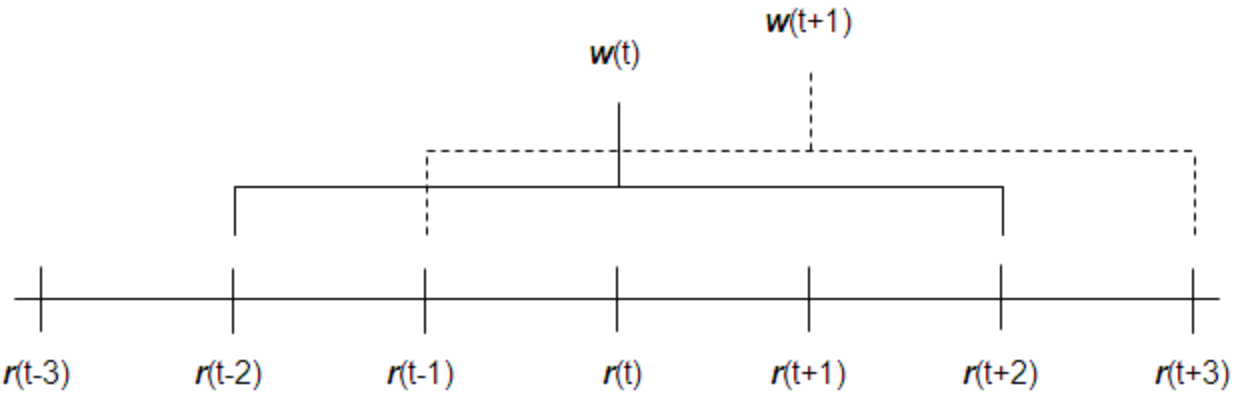


Figure 4.2 Typical input window into NN.

This thesis used a slightly different shifting window. Previously decoded symbols were used in place of past-received symbols, as shown in Figure 4.3.

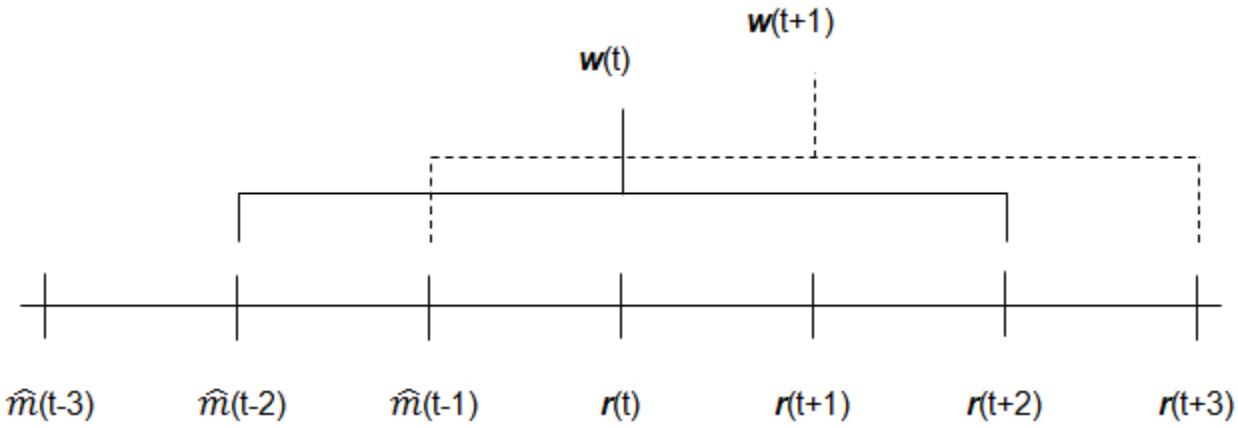


Figure 4.3 Proposed method input window into NN.

The symbol to be decoded was centered on a window size of 2K–1. This window size was used because the symbol being decoded was only affected by K–1 symbols before and after the symbol passed through the convolutional encoder.

Training data was generated randomly using MATLAB for different values of bit-energy-to-noise ratio (E_b/N_0), and target values of $\hat{m}$ in the shifting window were assumed to be m as shown in Table 4.1. The NN is an externally recurrent

backpropagation network with one hidden layer consisting of 64 neurons and utilized the tansig activation function. The network was trained using the Levenberg-Marquardt algorithm, which has the fastest convergence rate of training algorithms and is also the most computationally intensive. The mean squared error (MSE) of the training data is 10^{-12} and the MSE of the testing data is 10^{-11}.

TABLE 4.1

EXAMPLE TRAINING/TESTING DATA

Symbol Time	$s_1(t+2)$	$s_2(t+2)$	$s_1(t+1)$	$s_2(t+1)$	$s_1(t)$	$s_2(t)$	$\hat{m}(t-2)$	$\hat{m}(t-1)$	$\hat{m}(t)$
1	-1.1503	-1.4427	-1.5076	-0.7396	1.0771	1.1625	0	0	1
2	-1.3209	-0.9406	0.0548	0.3510	1.2895	-1.4813	0	1	1
3	0.7168	1.1469	1.1134	-0.7072	-1.1203	0.7795	1	1	1
4	1.0325	-1.3860	-1.1931	0.5215	-0.7862	0.6172	1	1	1
5	-1.0490	1.1818	-0.8872	0.8234	0.9876	-1.1233	1	1	0

Different network and window sizes were tested. A NN with 50 hidden neurons is achievable but very difficult to train several networks to have the same probability of error. 64 hidden neurons were chosen due to the low testing MSE and repeatability of creating networks with approximately equal probability of error. Higher hidden neuron counts were also tested, 100 and 120. The higher hidden neuron counts had the same performance with MSE and repeatability as the NN with 64 hidden neurons.

Various window sizes were tested with the NNs. The window size of 2K-1 received symbols before the symbol to be decoded and 2K-1 previously decoded symbols after the symbol to be decoded had the best BER performance. Window sizes of 2K and 2K+1 centered around the symbol to be decoded were also tested but had worse BER performance when compared to the 2K-1 window size. Also a window with 2K-1 previously received symbols, 2K-1 past received symbols and 2K-1 past decoded

symbols were tested and the BER performance was not as good as the new 2K-1 window size. The results are shown in Figure 4.4

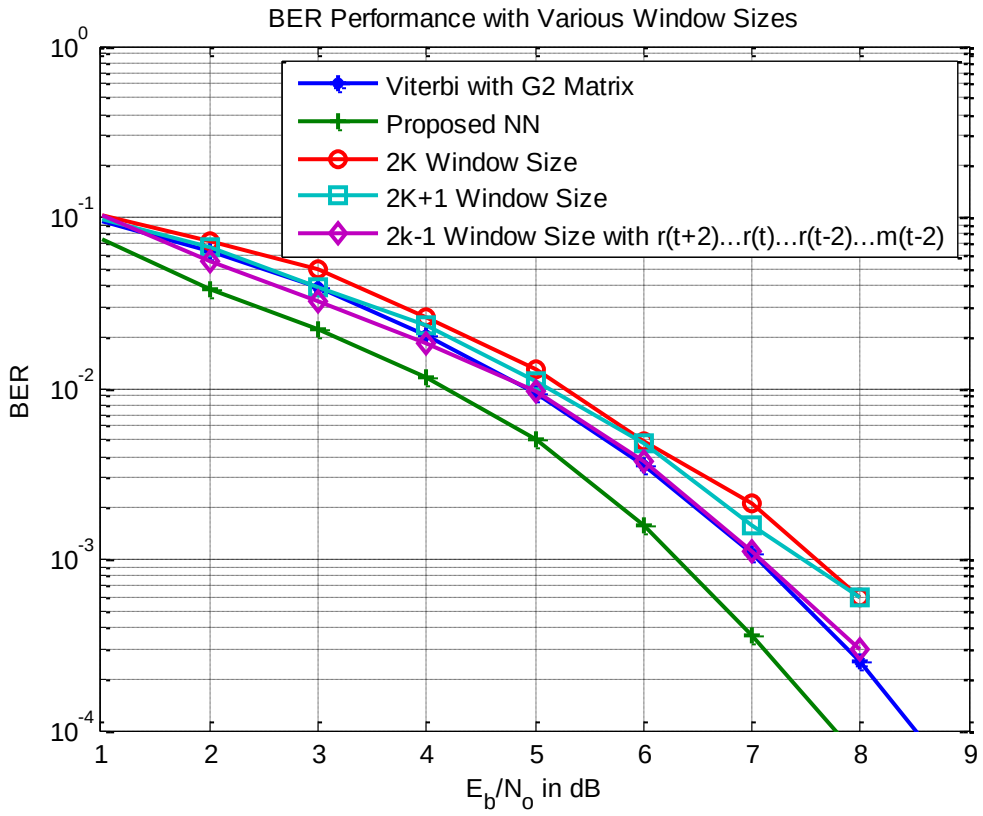


Figure 4.4 BER performance of NN with various window sizes.

A NN was also tested with non-systematic generator matrix $\boldsymbol{G_1}$ and systematic generator matrix $\boldsymbol{G_2}$. When non-systematic code was used, errors propagated and caused large burst errors. To minimize the burst error length, different symbol frame lengths were tested. As the frame length decreased the BER performance improved but is still not comparable to hard decision Viterbi with generator matrix $\boldsymbol{G_2}$. When using the systematic code, no burst errors occurred and BER performance was independent of

frame length and beat hard decision Viterbi with generator matrix $\boldsymbol{G_2}$. The final NN was trained using the systematic code and the results are shown in Figure 4.5.

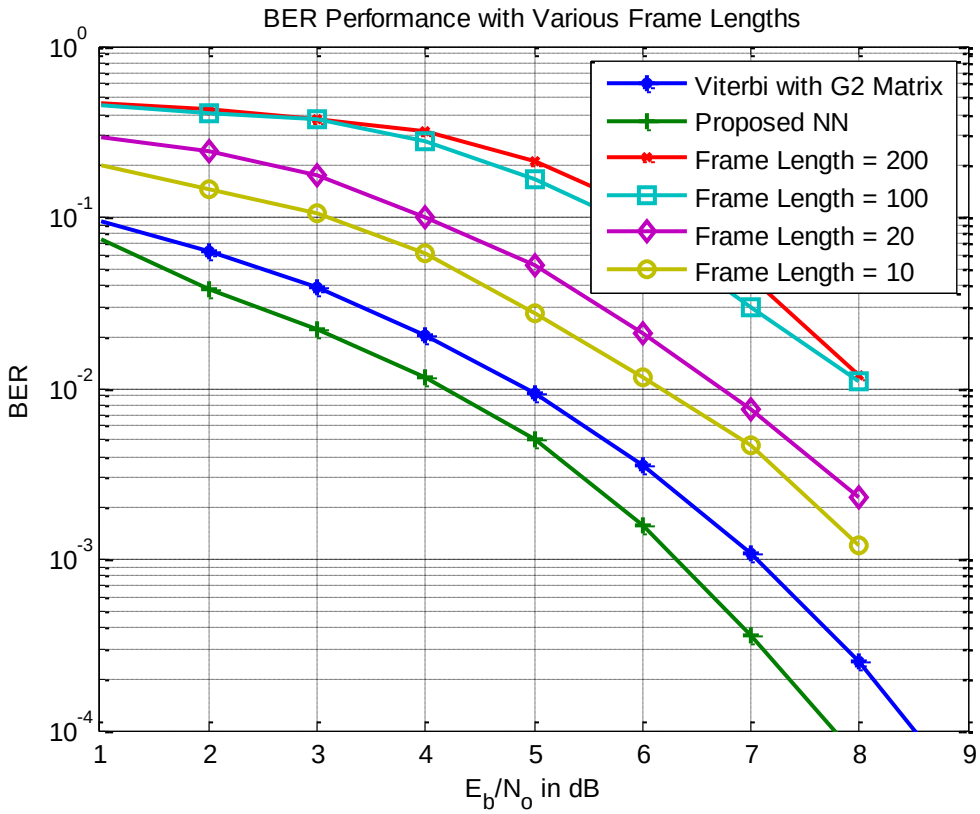


Figure 4.5 BER performance of NN with various frame lengths.

4.2 Placing Neural Networks in Parallel

The NNs were placed in parallel because each network has similar BER performance, and each network is unique and independent. The decoded symbol was chosen by majority rule, as shown in Figure 4.6. The majority rule was chosen because it performs well when each branch input is equally probable. Each NN has an approximately equal probability of error.

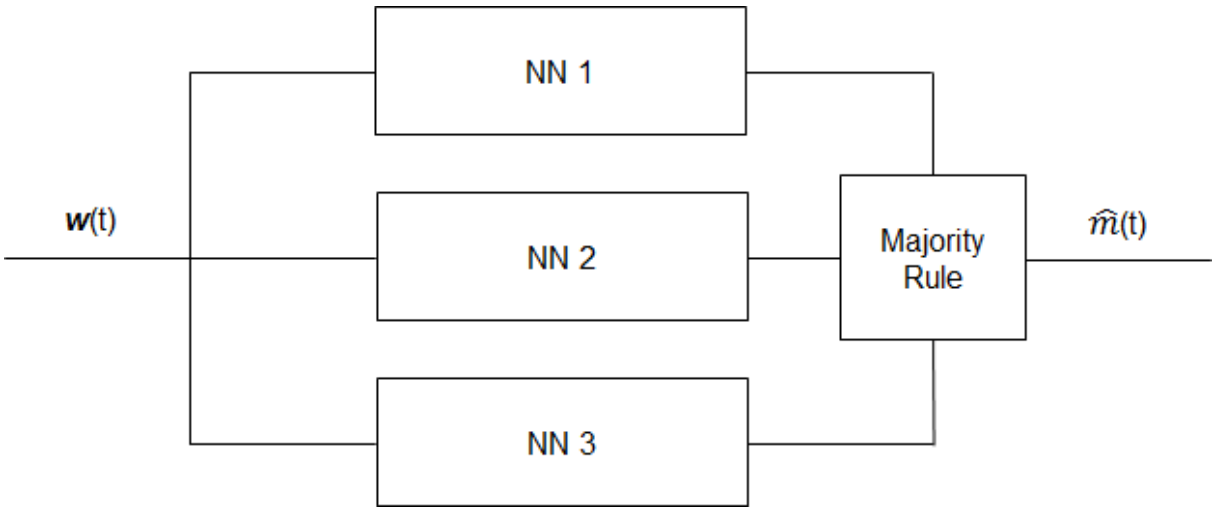


Figure 4.6 Multiple NNs placed in parallel using majority rule.

Assuming that the NNs are independent, the expected error probability of the parallel networks using the majority rule is as follows:

$$P_{b,maj}(E) = \Pr(\text{two or more bit errors simultaiously occuring}) \quad (4.1)$$

$$P_{b,maj}(E) = 1 - \Pr(\text{one branch bit or zero bit errors occur}) \quad (4.2)$$

where

$$\Pr(\text{one branch bit or zero bit errors occur}) = \left[\binom{0}{0}\binom{0}{0}\binom{0}{0}\right] \quad \binom{1}{0}\binom{0}{1}\binom{0}{0} \quad \binom{0}{0}\binom{1}{0}\binom{0}{0} \quad \binom{0}{0}\binom{0}{0}\binom{1}{0}\Big]. \quad (4.3)$$

The first column in equation (4.3) represents no errors in the three branches, and the columns that follow represent a single error in its respective branch with a probability P_b. Equation (4.3) then becomes

$$\Pr = (1-P_{b1})(1-P_{b2})(1-P_{b3}) + P_{b1}(1-P_{b2})(1-P_{b3}) + \cdots + P_{b3}(1-P_{b1})(1-P_{b2}) \quad (4.4)$$

If $P_{b1} = P_{b2} = P_{b3}$, then

$$Pr = (1-P_b)^3 + \binom{3}{1}P_b(1-P_b)^2 \quad (4.5)$$

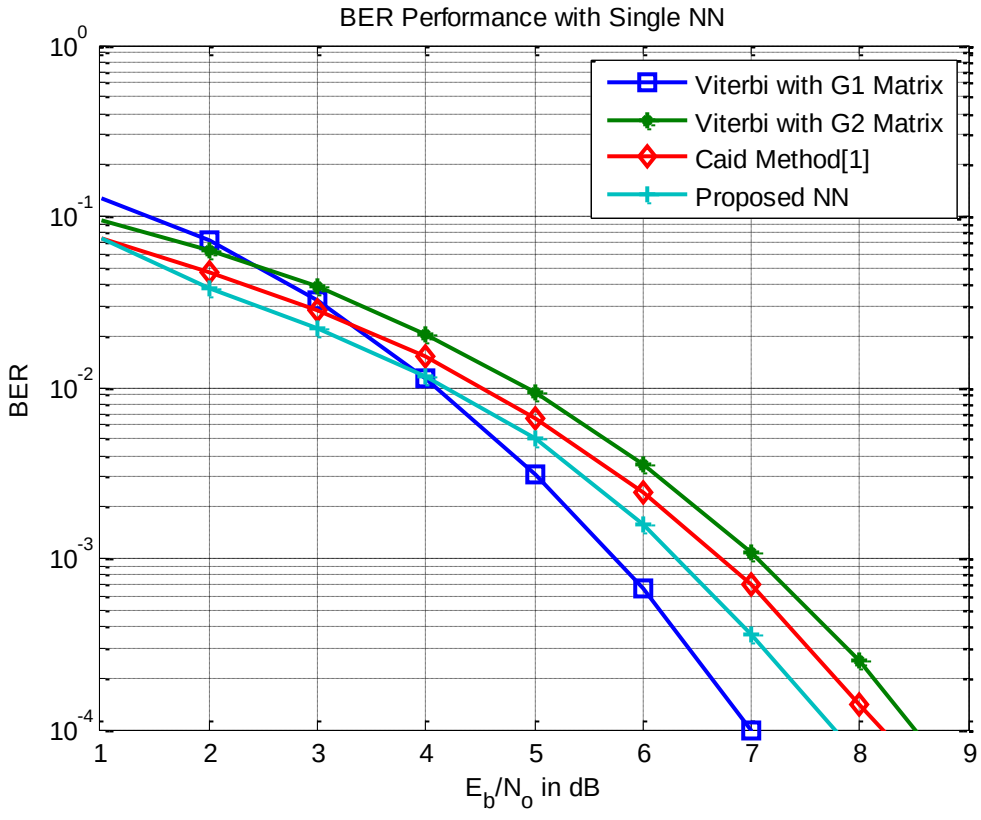
and equation (4.2) becomes

$$P_{b,maj}(E) = 1 - [(1-P_b)^3 + \binom{3}{1}P_b(1-P_b)^2]. \quad (4.6)$$

CHAPTER 5

DISCUSSION/SIMULATION RESULTS

A neural network with the proposed input shifting window was simulated using MATLAB, and the BER vs. E_b/N_0 performance was found to be comparable to the Viterbi decoder with the non-systematic generator matrix $\boldsymbol{G_1}$ and outperformed a Viterbi decoder with generator matrix $\boldsymbol{G_2}$ by 1 dB. The NN with the proposed sliding window outperformed the NN with the conventional sliding window [1] by ½ dB. At E_b/N_0 less than or equal to 4, the NN decoder outperformed the Viterbi decoder with $\boldsymbol{G_1}$ by 1 dB, and at higher E_b/N_0, the Viterbi decoder with $\boldsymbol{G_1}$ outperformed the NN decoder by almost 1 dB, as shown in Figure 5.1.



g

Figure 5.1 BER performance of single NN decoder.

Results of the simulations show a couple of unusual observations relative to the communication theory:

- Coding gain shifts the curve to the left. In this simulation, the curve is shifted downward, which is generally caused by diversity gain. Diversity gain is achieved by using multiple antennas.
- The network used processed information (output of the NN) as input, and BER performance was expected to increase because processing data removes information and makes the performance worse.

Placing the proposed NNs in parallel further improved the BER performance. This was expected since NNs are individually not optimal solutions. Each NN was expected to produce errors, but the errors were not expected to always occur on the same symbols. Using the majority rule took advantage of this fact and improved the BER performance. When P_b = 10^{-3} in equation (4.6), then $P_{b,maj}$ = $3*10^{-6}$. By using multiple NNs, the observed BER in Figure 5.2 was not as significant as in equation (4.6) and was only ½ dB better than the single NN at higher E_b/N_0. The BER performance was not a significant improvement to the proposed single NN and was still only comparable to the Viterbi decoder, as shown in Figure 5.2.

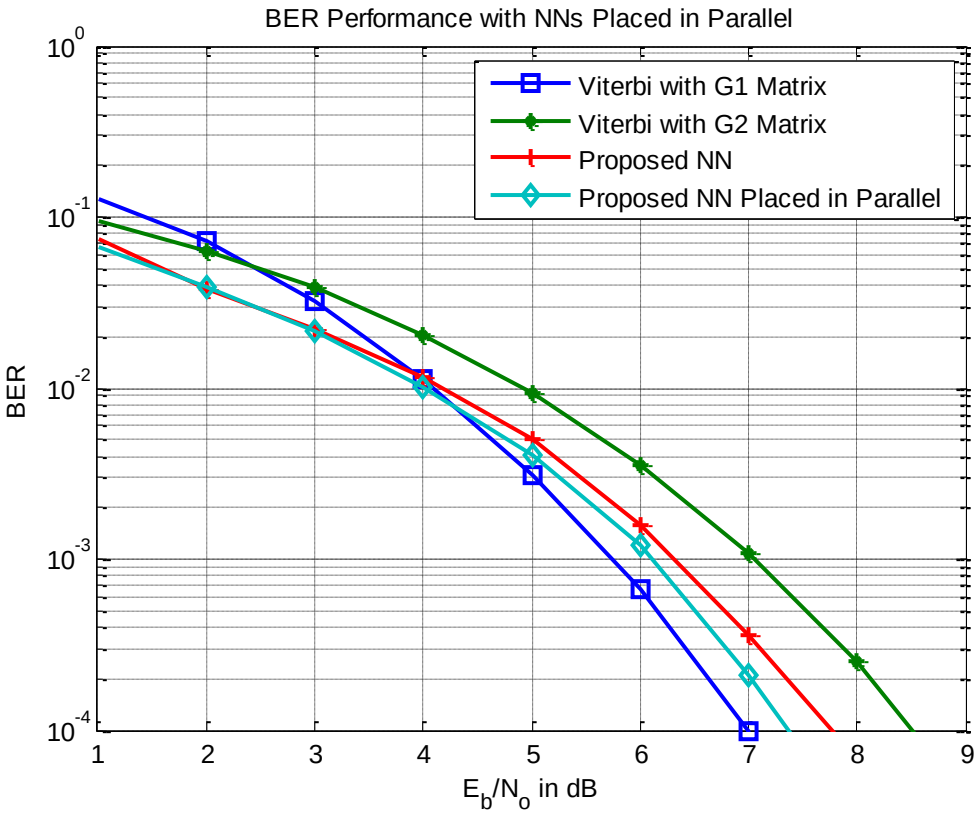


Figure 5.2 BER performance of parallel NN decoder.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

The proposed shifting window improved the BER performance of the decoder. Using just a backpropagation NN was not sufficient to completely outperform the hard decision Viterbi decoder. A recurrent NN and a NN that takes an iterative approach had better BER performance [3-6]. Applying previously decoded symbols to the input shifting window to any of these networks may improve the BER performance of these networks.

6.2 Future Work

Further study using the proposed shifting window with more complex NNs, especially wavelet and convolutional networks, could be explored. Different combining and selection rules could be explored by placing the NNs in parallel.

REFERENCES

REFERENCES

[1] Caid, W. R., and Means, R. W., “Neural Network Correcting Decoders for Convolutional Codes,” IEEE, *Proceedings of the Global Telecommunications Conference*, Vol. 2, 1990, pp. 1028 - 1031.

[2] Annauth, R., and Rughooputh, H.C.S., “Neural Network Decoding of Turbo Codes,” IEEE, *Proceedings of the Conference on Neural Networks*, Vol. 5, 1999, pp. 3336–3341.

[3] Hamalainen, A., and Henriksson, J., “A Recurrent Neural Decoder for Convolutional Codes,” 1999 IEEE, *Proceedings of the International Conference on Communications*, Vol. 2, 1999, pp. 1305–1309.

[4] Berber, S.M., “A Soft Decision Output Convolutional Decoder Based on the Application of Neural Networks,” IEEE, *Proceedings of the Military Communications Conference*, Vol. 3, 2005, pp. 1495–1500.

[5] Johnny W. H. Kao, Stevan M. Berber, and Abbas Bigdeli, “A General Rate K/N Convolutional Decoder Based on Neural Networks with Stopping Criterion,” *Advances in Artificial Intelligence*, Vol. 2009, Article ID 356120, 11 pages, 2009. doi:10.1155/2009/356120

[6] Hueske, K., Gotze, J., and Coersmeier, E., “Improving the Performance of a Recurrent Neural Network Convolutional Decoder,” IEEE, *Proceedings of the International Symposium on Signal Processing and Information Technology*, 2007, pp. 889–893.

[7] Fausett, L., *Fundamentals of Neural Networks Architectures, Algorithms, and Applications*, Chapters 1 and 2, Prentice Hall, Upper Saddle River, NJ, 1988.

[8] Cui, H. and Rapajic, P, “Complexity Comparison of Iterative Channel Estimation,” http://www.elec.uow.edu.au/staff/wysocki/dspcs/papers/011.pdf, December, 2010.