Journal of Universal Computer Science, vol. 12, no. 10 (2006), 1439-1454
submitted: 19/6/06, accepted: 25/10/05, appeared: 28/10/06 © J.UCS

Fault Tolerant Neural Predictors for Compression of Sensor Telemetry Data

Rajasvaran Logeswaran
(Multimedia University, Malaysia
loges@mmu.edu.my)

Abstract: When dealing with remote systems, it is desirable that these systems are capable of operation within acceptable levels with minimal control and maintenance. In terms or transmission of telemetry information, a prediction-based compression scheme has been introduced. This paper studies the influence of some typical transmission and network errors on the encoded residue stream produced by a number of predictors used in the scheme, with the intention of identifying the more fault tolerant architecture that may be preferred as predictors. Classical linear predictors such as FIR and lattice filters, as well as a variety of feedforward and recurrent neural networks are studied. The residue streams produced by these predictors are subjected to two types of commonly occurring transmission noise, namely gaussian and burst. The noisy signal is decoded at the receiver and the magnitude of error, in terms or MSE and MAE are compared. Hardware failures in the input receptor and multiplier are also simulated and the performance of various predictors is compared. Overall, it is found that even small low-complexity neural networks are more resilient to faults due to the characteristics of their parallel architecture and distributed storage/processing characteristics.

Keywords: Data compression, error tolerance, neural networks, predictors
Categories: H.4.3, E.2, I.5.4

1 Introduction

In dealing with remote systems, sensor telemetry data is vital for monitoring the performance of the systems, ranging from the operations of remote installation, to micro-systems within. Data compression offers an effective way to reduce the size of the transmitted information, through reduction / removal of data redundancy, with the additional benefit of reduced storage space requirements. Predictive compression is popular when dealing with real-time data that is highly correlated in commonly known distribution patterns, as is the case with most telemetry data from remote sensors, due to their high compression performance [1]-[3]. The difference between most successive values in the telemetry data for a particular sensor is generally small, thus, emphasis is therefore placed on lossless (rather than lossy) compression to preserve accuracy.

In typical predictor-based lossless systems, it is the residues, i.e. difference between the original input and predicted values, which are transmitted to the receiver. Predictors decorrelate the input stream such that the distribution of the residues is almost white gaussian [4]. A good predictor produces residues that are of significantly lesser magnitude than the source samples, thus exacting feasible compression. At the receiver, the original values are restored by adding the received residues to values generated by an identical predictor at the receiver. To further improve the

compression performance, an additional lossless encoder may be used to further decorrelate the residue stream [3]-[6].

As residues are usually of small magnitude, they tend to be more susceptible to transmission and network errors due to their low signal to noise ratio (SNR). This paper is concerned with the choosing of error and fault tolerant predictors in order to maintain a high level of accuracy even when the compression system is subjected to common errors and disruptions, especially so when such problems are faced at remote sites where maintenance is both difficult (if not impossible) and costly (in terms of finance as well as in trusting the accuracy of the received values). The actual results in a practical system is also dependent on the error correction coding used to protect the transmission, but as there is a broad selection of them, such coding will not be included in the scope of this paper. The aim is to minimize the effect of error even before the application of such error correction coding.

A prediction-based scheme has been introduced for lossless compression of satellite launch vehicle (SLV) telemetry data, and has been shown to produce good compression performance in terms of speed and compression ability [1]. The scheme utilizes classical predictors such as the finite impulse response (FIR) and recursive lattice filters, which are popular choices for data compression [2], [5]. Neural networks are known for their error tolerance and graceful performance degradation properties in the presence of noise and network failures, and have also been successfully used in many pattern recognition schemes in the past [3]. Lately, neural networks have also been introduced as predictors for lossless compression in this scheme [1].

This paper examines the performance of a number of different classical and neural network architectures, with the purpose of comparing the error and fault tolerance capabilities of the predictors when subjected to some typical transmission conditions. A selected subset of tests are applied to the predictors, using a number of telemetry data sets or varying sizes, distribution and sourced from different remote sensors. The results and discussions in this paper is aimed in assisting in determining the choice of predictors for use in compression and transmission of telemetry data, as well as possibly in broader applications in other areas where low-cost low-complexity fault tolerant predictive devices are beneficial.

2 Predictors

Several architectures are chosen from the classical and neural network predictors for testing. In order to provide comparisons at the approximately similar level of complexity, and to minimize resource overheads and size of the implemented remote system, small topologies are chosen based on past experience and ongoing research [3], [6], [7]. The configuration and training information of the various predictors examined is discussed below and summarized in Table 2. The last column of the table provides the average estimated processing time for the predictors to compress the test data files, as a measure of the performance of the predictors in terms of speed. Such performance issues are discussed in the Conclusions section.

2.1 Classical Predictors

Two classical architectures in three different strategies are chosen from existing popular implementations [2] for testing.

2.1.1 Finite Impulse Response (FIR) Filter

The architecture of the FIR filter is such that it uses delays on its inputs to enable immediate past values to be used at each iteration. Taking advantage of this, it can be set up as a predictor, with the order of the predictor corresponding to the number of past values used to predict the present value.

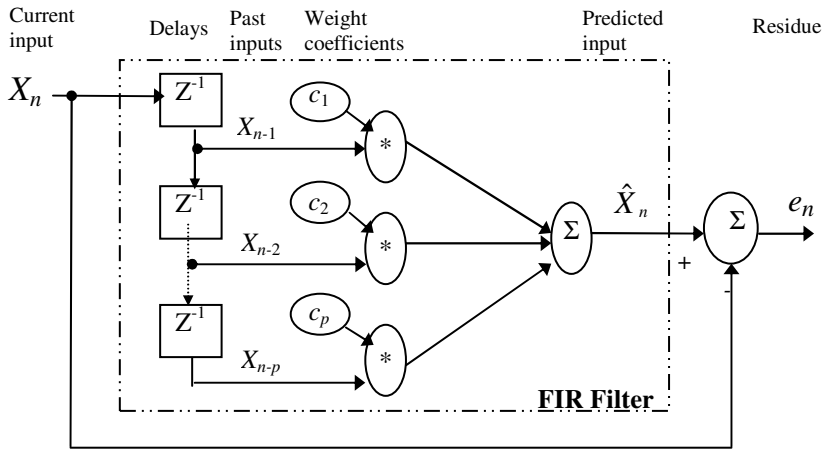

Figure 1: p^{th}-order Finite Impulse Response (FIR) predictor. The output is the residue (e).

An example of the structure of a p^{th}-order FIR predictor is given in Fig. 1 [2]. The predicted value ($\hat{X}_n$) of the n^{th} input (X_n), is the sum of the product of the past p input values and their corresponding coefficients (c_i), as given by (1). The residues (e_n) are calculated using (2) and the FIR predictor is implemented as a fixed model using fixed coefficients, given by (3), throughout the prediction process.

$$\hat{X}_n = \sum_{i=1}^{p} c_i \cdot X_{n-i} \tag{1}$$

$$e_n = X_n - \hat{X}_n \tag{2}$$

$$\hat{X}_n = 4X_{n-1} - 7X_{n-2} + 7X_{n-3} - 4X_{n-4} + X_{n-5} \tag{3}$$

The second strategy for testing is to use a fully adaptive FIR implemented with the *normalised least mean squares* (NLMS) algorithm [1], to adaptively adjust the coefficients to the telemetry data input patterns.

2.1.2 Recursive Lattice Filter

Recursive lattice filters have been shown to produce very good compression performance as predictors [3]-[6]. These filters have lattices (layers) that provide forward and backward prediction, as shown in Fig. 2. The adaptive *p*-lattice structure in Fig. 2 uses *the recursive least squares lattice* (RLSL) algorithm with *a priori* estimation errors and error feedback for prediction. The main feature of this high performance algorithm is that it handles adaptive forward prediction (η_p), adaptive backward prediction (β_p) as well as adaptive joint-process estimation (ξ_p). These processes are implemented through direct order updating of the forward reflection coefficients ($\kappa_{f,p}$), backward coefficients ($\kappa_{b,p}$) and joint-process regression coefficients (κ_p), to produced the predicted values. Detailed description of the algorithm and related algorithms, such as the Gradient Adaptive Lattice (GAL), can found in [2] and will not be discussed here.

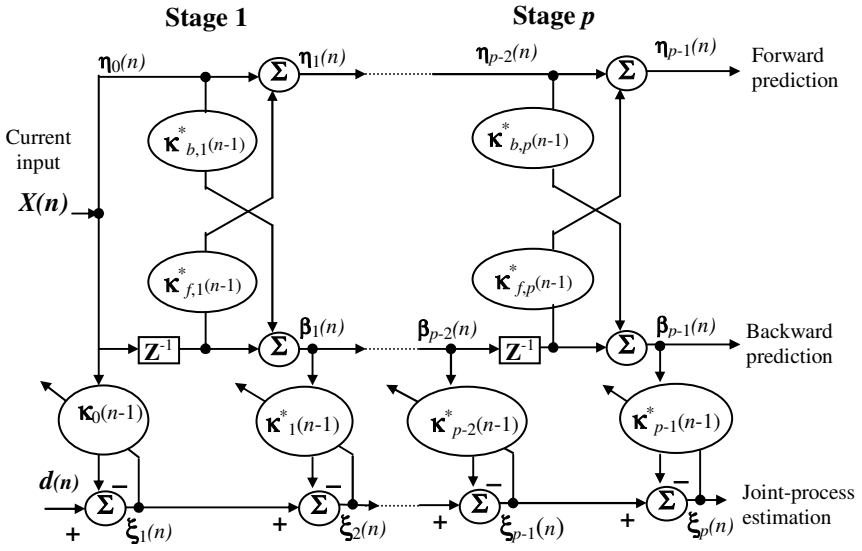

Figure 2: p[th]*-order Recursive Least Squares Lattice (RLSL) with a-priori feedback*

2.2 Neural Network Predictors

Certain properties of the artificial neural networks render them useful as predictors [2]. These properties include: the ability to self-learn, flexible internal organization, the ability to acquire knowledge even from noisy data through generalization, error tolerance to data inconsistencies, and parallel distributed processing and information storage that gives it speed as well as allows it to degrade gracefully when network failure occurs.

2.2.1 Training and Test Data

Block-adaptive training is implemented for the neural networks, as in [8]. The reason for this is to allow the network to adapt to input patterns, and at the same time minimize the likelihood of overfitting the predictor during training. This simple scheme curbs the potential of the problem of network rigidity, which reduces the network's generalization capabilities and adversely affecting prediction performance, without costly overheads involved in implementing more complex algorithms.

The block-adaptive method involves sequentially partitioning the input into blocks of a fixed number of samples. The first 20% of samples in a block are used to train the network, and the remainder would be predicted using the trained network. Different block sizes (S_B) are used in training to determine the most appropriate sizes for each architecture. Targeted performance goals are set to determine sufficient training (i.e. with accuracy of MSE=0.1 or until 10,000 epochs). As the networks are targeted for real-time performance, prior knowledge of the telemetry data distribution pattern is not used during training. The training set consists of only the first 20% of values of the current block. The training size (i.e. 20%) was chosen as a compromise to provide sufficient training to start the prediction by anticipating part of the input pattern of the block, but not too large that it significantly reduces the compression performance and increases processing time. Prediction is implemented by using the past p values to predict the current value, and then calculating and transmitting the residue.

Six test files are used in this paper, derived from the various sensors of a satellite launch vehicle (SLV). The telemetry data obtained from the various sensors provide varying types of telemetry readings, distributions, input patterns and file sizes, allowing the proposed system to be tested in a more robust manner to mimic the general capabilities expected by implementing the system for general telemetry compression. Some of the characteristics of the test data used are given in Table 1.

Test File	File Size (bytes)	Total no. of symbols	Sampling rate (symbols per sec.)	No. of distinct symbols	Max. freq. of a symbol	Max. value of a symbol	Source Entropy (bits per symbol)
td1	252305	28324	520	157	7131	66.135	3.128
td2	139571	11631	65	12	7484	1070.249	1.017
td3	55365	6778	65	43	1438	76.105	4.644
td4	131841	16052	130	191	3985	50.894	5.387
td5	184774	17232	65	240	349	4960.000	7.614
td6	74915	8662	65	6	2840	124.250	2.121

Table 1: Characteristics of the test data

Three categories of neural networks with five different architectures are briefly discussed below. All the networks possess only one output node as only one value is predicted at each iteration. The basic layout of the architecture used, training scheme and some transmission issues are detailed in [8]. The configurations specified in this paper were found, experimentally, to be optimum for the small architectural

requirements and test sets used, and may be used as a guide in determining the optimum settings for application on other data.

2.2.2 Standard Feedforward Networks

Three well known artificial neural network architectures are implemented in this category, namely the perceptron network (PN), the single-layer feedforward network (SLFN) and the multi-layer feedforward network (MLFN). Fig. 3 shows the architecture of the PN and the SLFN, both sharing a similar general architecture. However, the PN uses the hardlimiter or step function activation threshold (or transfer function, *f(.)*) in the output layer, whereas the SLFN was set up to use the linear function (as it performed better for the test sets than the conventional sigmoid function). Training (setting the weights and biases) of the PN is undertaken via the Rosenblatt / Perceptron rule [9]. Training of the 4th-order SLFN is by means of the *least mean squares (LMS)* algorithm [10]. Note that by convention, input nodes are not counted as a layer as they are not processing elements (PE) and have no activation functions. They do however determine the number of past values to be used during the prediction. As such, the 4th-order SLFN actually has 4 input nodes and 2 layers if the input layer is to be counted.

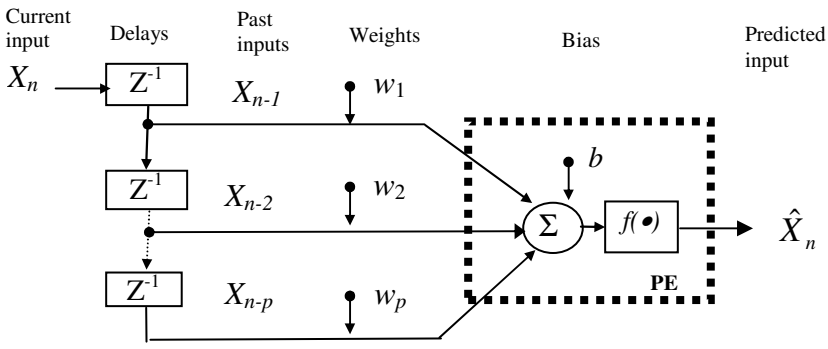

Figure 3: pth-order single-layer single-output Perceptron Network (PN) predictor. Similar architecture used by Single Layer Feedforward Network (SLFN) predictor, by replacing the step-function of f(.) with other activation functions

The architecture of the MLFN, also popularly known in some literature as the multi-layer perceptron (MLP), is given in Fig. 4. Trained using the backpropagation algorithm [10], the MLFN was set up to use the linear and sigmoid *f(.)* for the hidden and output layers, respectively.

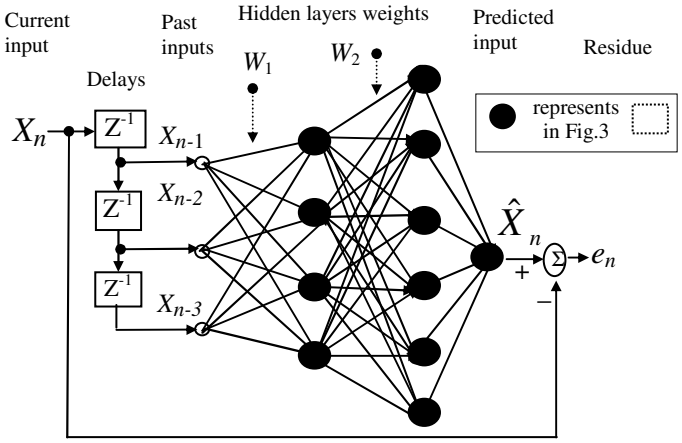

Figure 4: A 3rd-order 3-layer Multi Layer Feedforward Network (MLFN) predictor with two middle layers of 4 and 6 processing elements (PE), respectively. The output is the residue (e)

2.2.3 Radial Basis Network

The general architecture of a radial basis network is that of a feedforward network, but using radial basis functions (such as the gaussian distribution) instead of the standard activation functions (e.g. linear, sigmoid). To test this type of network, the popular two-layer generalized regression neural network (GRNN) [7], [11], commonly used as a function approximator, was set up with a gaussian and linear *f(.)* in the hidden and output layers, respectively. This neural network predicts by projecting the training window approximation function across S_B. This unconventional prediction method works well for data that is relatively stable or has repetitive properties.

2.2.4 Recurrent Network

A recurrent network is one which feeds some or all of one or more layers' output back into itself or into one (or more) of the preceding layers. This allows the output of each iteration to influence the outcome of consequent iterations. The two-layer Elman network (EN) [12] implemented has a recurrent linear hidden layer and a sigmoidal output layer, as shown in Fig. 5. This type of network is known to have the ability to recognize spatial and temporal structures in the input [9].

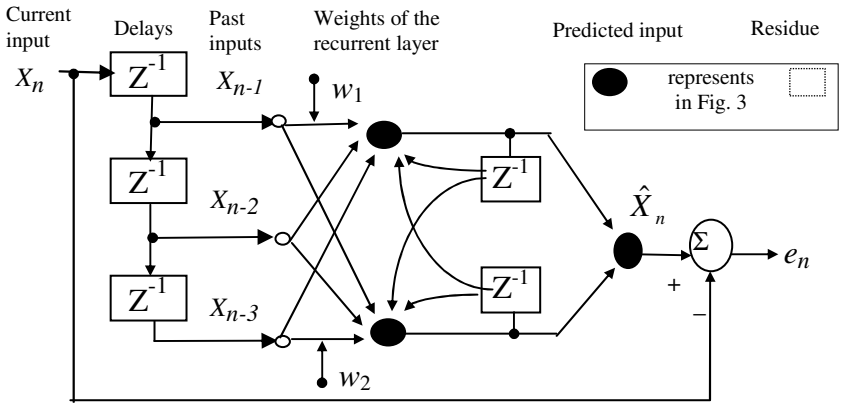

*Figure 5: A 3rd-order 2-layer Elman Network (EN) predictor with two feedback nodes in the recurrent hidden layer. Outputs residue (*e*).*

3 Influence Of Noise On Predictor Performance

External sources, as well as the medium used for transmission or storage, can cause the influence of noise on the transmmitted residue. The two common types of noise that affect transmission are random and burst noise. An example of random noise is the motion of electrons in the material, which is also referred to as thermal noise. On the other hand, high amplitude impulse interference caused by situations such as the occurrence of large electromagnetic waves, e.g. a flash of lightning, is known as burst noise.

3.1 Performance With Gaussian Noise

To test the predictors, are random white noise was generated to fit the gaussian distribution G(0,1) and added to the transmitted residue stream. The average noise applied was *targeted* at approximately 30% signal power, with signal-to-noise ratio (SNR) approximately 15 dB, as given in Fig. 6. A low SNR may signify a noisy transmission, and also small residues (i.e. good prediction).

The test is implemented as follows:

(a) the receiver is trained using clean samples,
(b) the trained receiver forced to predict using the noisy input,
(c) the error between the original source data and the restored data (i.e. predicted value + noisy residue) is measured,
(d) the mean squared error (MSE) is calculated.

The process is repeated for each test file, using the different predictors. The results obtained are given in Fig. 7. For relative comparison, the mean square value (MSV) of the source data is plotted on the graph. The mean absolute error (MAE) and its corresponding mean absolute value (MAV) of the source is plotted in Fig. 8 for further comparison of the performance of the architectures on the different test files.

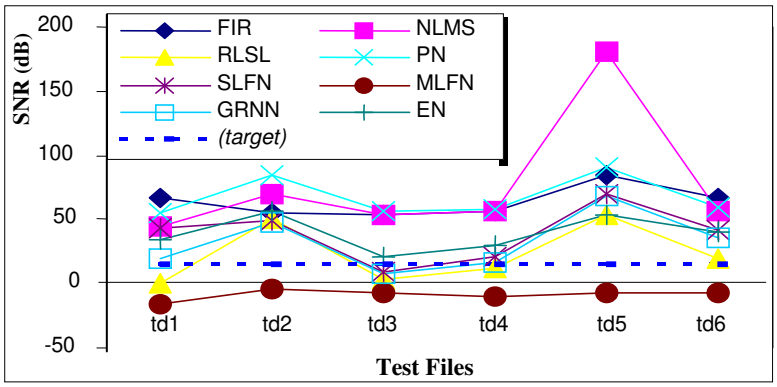

Figure 6: Signal-to-Noise-Ratio (SNR) achieved by residues with gaussian white noise, for each test file.

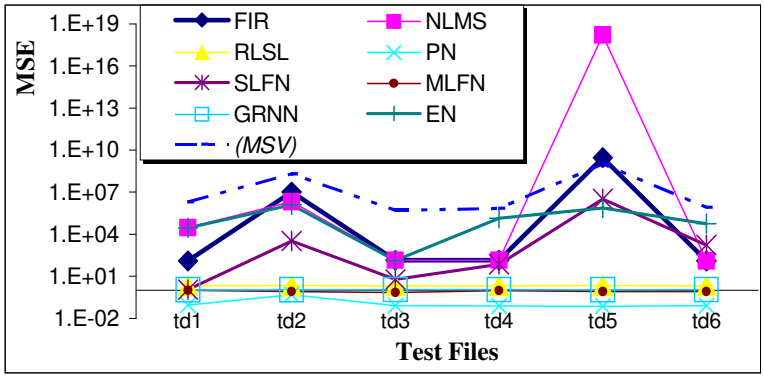

Figure 7: Mean Squared Error (MSE) achieved by residues for gaussian noise robustness test.

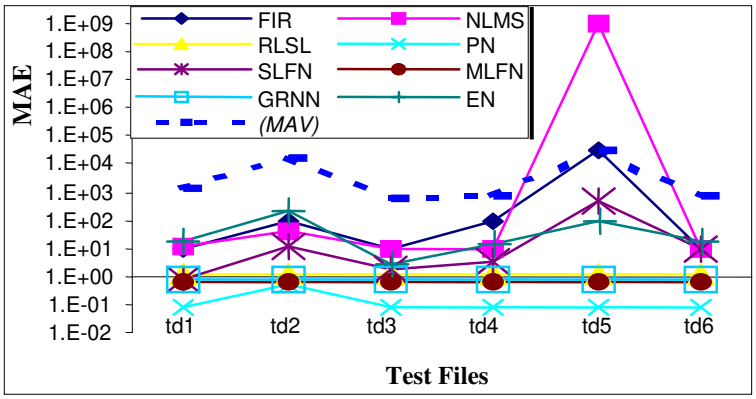

Figure 8: Mean Absolute Error (MAE) achieved by residues in gaussian noise robustness test, for each file.

Analysing the results from the three graphs, the following observations may be made. Even with large SNR, the traditional FIR and NLMS performed badly in the white noise test. The best MSE result was achieved by the PN, but its high SNR signifies poor compression ability and insensitivity to the input patterns. The GRNN and RLSL had similar performance, with the GRNN being slightly better in terms of its MAE values. Overall, among the tested predictors, the best white noise tolerance was achieved by the MLFN. The MLFN residues were of lower magnitude than the noise. Note that since negative SNR values could not be plotted on the logarithmic y-axes, these MLFN points are 'missing' from the figures. In general, the test concluded that the artificial neural networks were able to achieve better fault tolerance capabilities than the FIR and NLMS. From the MAE values in Fig. 8, there is evidence to suggest that the artificial neural networks are more adaptive and tolerant to transmission white noise.

3.2 Performance With Burst Noise

A similar process of testing was conducted to evaluate the performance of the predictors on residue stream influenced by burst noise. The gaussian distribution was replaced by short bursts (across five sample values) of signal with an amplitude approximately as large those of the residue stream. These burst were added to the residue stream at two instances, a quarter and three-quarters way through each file.

The results achieved by the predictors were plotted in Fig. 9 and 10. In this test, the NLMS performed significantly better than the EN and SLFN as the linear *f(.)* in the SLFN was unable to cope with the high amplitude errors, whereas the recurrent EN propagated past values from the bursts into succeeding iterations of the algorithm, adversely affecting its performance. The lowest MSE was achieved by the PN (the 'missing' points were when its MSE was 0).

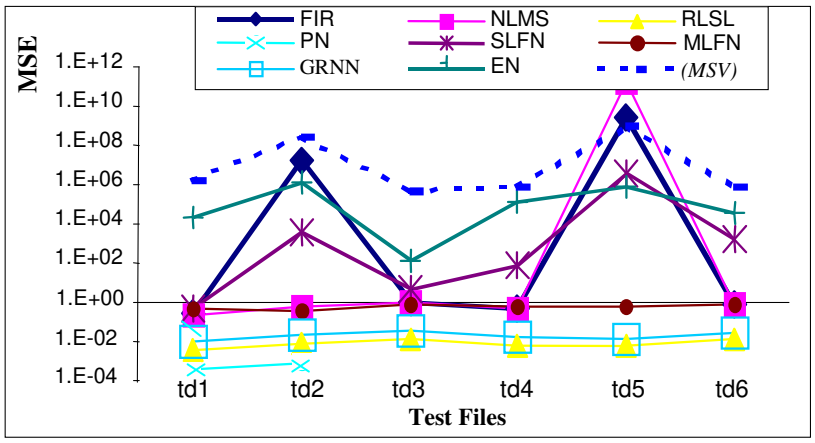

Figure 9: MSE achieved by residues in burst noise robustness test.

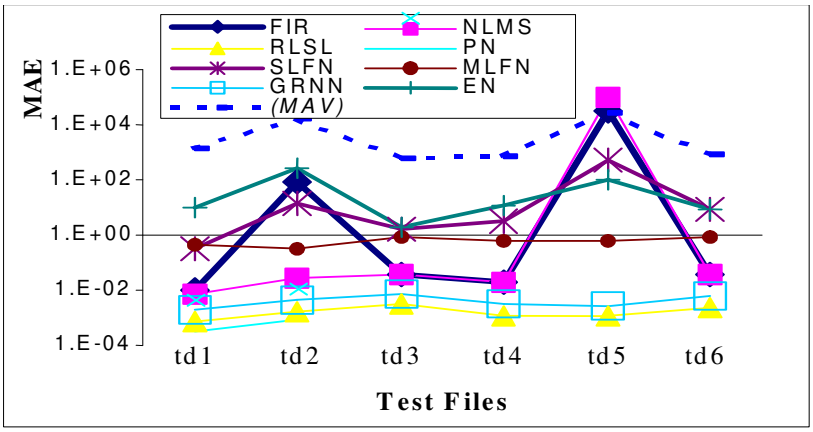

Figure 10: MAE achieved by residues in burst noise robustness test.

From both the test cases for noise, it can be concluded that the performance achieved by the predictors were comparable. In cases where the results obtained is above MSV (e.g. FIR and NLMS for file td5), the inaccuracy of the received transmission would be unacceptable.

4 Hardware Failure

When evaluating systems, especially those afixed at remote sites, fault tolerance to hardware errors is vital. The predictors were tested for two types of hardware failures, as described below.

4.1 Input Receptor Failure

A test was simulated to mimic the case where an input receptor is spoilt and none of the values sent to the node contribute to the prediction process. In this test, the first input node was forced to fail by setting all values propagated from the first node to the next stage (affects all PE in the next layer the of the neural network predictors) to zero.

From the results in Fig. 11, the plots for RLSL and SLFN overlap the MSV. This means that the error was similar to the difference between the original residue values, thus the accuracy of the restored stream is doubtful. The best performance was by the MLFN, followed by the NLMS. The FIR performed badly, while the PN and GRNN achieved MSE and MAE values of zero.

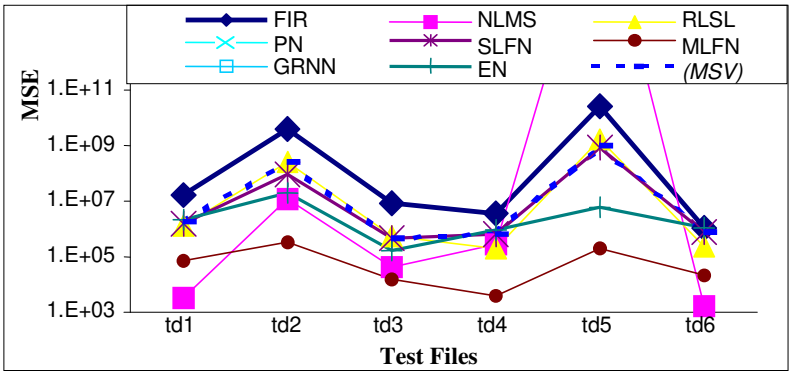

Figure 11: MSE achieved by residues in hardware input failure robustness test.

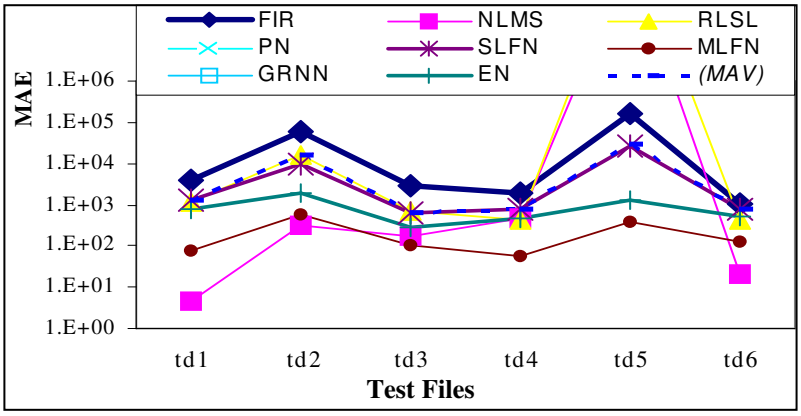

Figure 12: MAE achieved by the hardware input failure robustness test.

4.2 Multiplier Failure

As the artificial neural network architecture relies heavily on employing multipliers to integrate the weight coeffieints of the connections between the layers in its decision making process, this test was devised to simulate a multiplier error. This was undertaken by setting a weight coefficient from the first input node to the first node in the next layer is set to zero. Unlike the test for the input node failure, this test only affects one of the weighted inputs to the first PE of the layer immediately following the input layer, i.e. the first hidden layer for multi-layer architectures, or the output layer for single-layer architectures. The results achieved are given in Fig. 13 and 14.

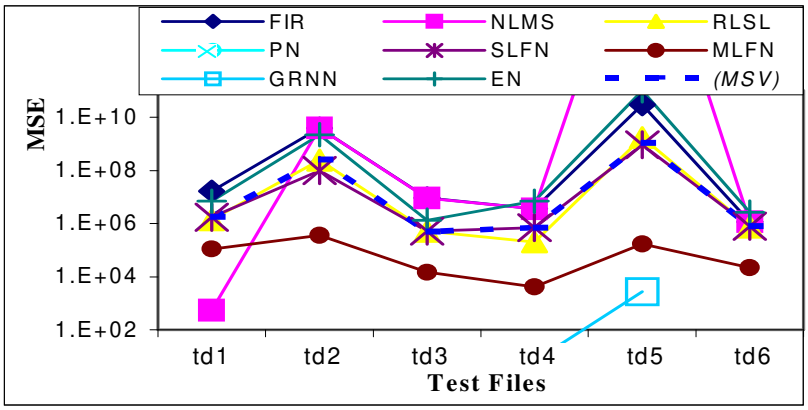

Figure 13: MSE achieved by residues for hardware multiplier failure robustness test

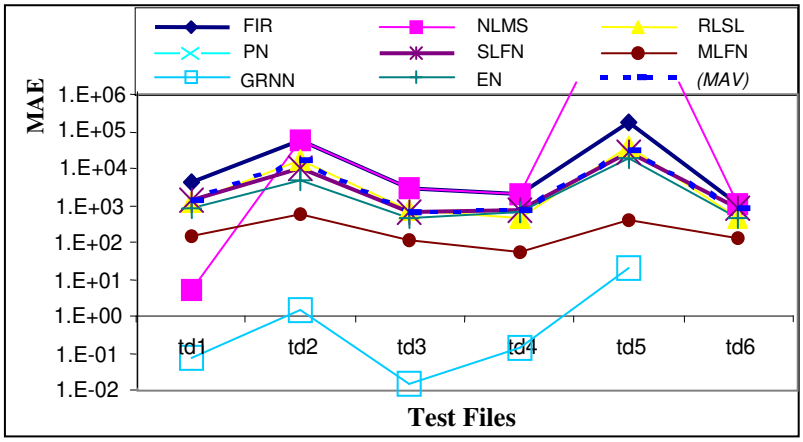

Figure 14: MAE achieved by residues for hardware multiplier failure robustness test

From the plots, the MLFN produced acceptable results, the GRNN was better with an average MSE of below 1.E+02, and the PN achieved MSE=0. The accuracy of the RLSL and SLFN under this test is unacceptable as their plots are very close to (or overlapping) the source MSV, and the remaining predictors performed even more poorly with higher MSE values.

From both the hardware tests, it was found that the errors recorded by the linear predictors were close to or higher than the MSV and MAV, making them very susceptible to hardware failures. On the other hand, good tolerance was exhibited by three of the neural network architectures (i.e. PN, GRNN and MLFN), while the SLFN and EN performed poorly.

5 Conclusion

This paper has evaluated the fault tolerance of several neural networks and linear architectures that were implemented as predictors in a lossless compression system for telemetry data from small remote sensors. Tests were undertaken two types of transmission noise and two types of hardware failures. Results of the comparisons, in terms of MSE and MAE, show that in most cases, the neural networks, such as the PN, GRNN and MLFN, were more fault tolerant than the linear predictors, with exception to the RLSL that performed well.

In the interest of robust testing, telemetry test files of different sizes and distribution, were used to evaluate the predictors. The compression ability of the predictors have to be taken into account in advocating a suitable predictor for the lossless compression system. The performance of the predictors, in terms of the compression ratio (CR) achieved for each of the test files is given in Fig. 15.

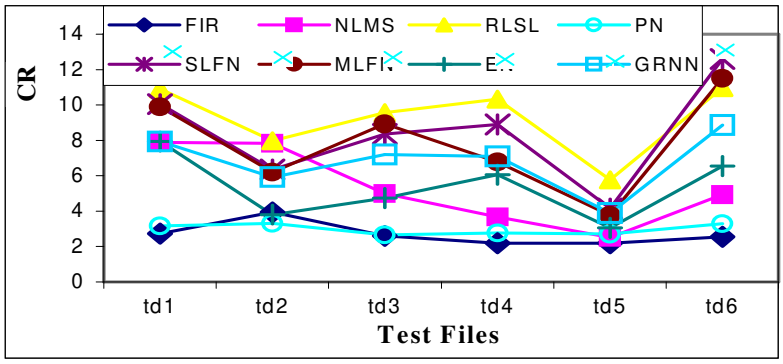

Figure 15: Compression ratio (CR) performance achieved by predictors for each test file, using the different predictors.

From the results, it is observed that the RLSL provides the best compression, followed by SLFN, MLFN and GRNN. The PN, which displayed the best fault tolerance results in the earlier experiments, achieved very low CR, as with the FIR. The hardlimiter activation in the PN is capable of producing only binary output, thus

it was insensitivity to small variations in the data stream, making it both fault tolerant but also a poor predictor. Another factor vital to the lossless system under study would be for real-time application. The average processing time taken by each predictor, as well as a summary of the predictor configurations, is given in Table 2 below. From the table, it is found that the GRNN has the best processing speed, albeit having to be trained more often as its optimum S_B is small (but its training time for each block is a fraction of the other neural networks).

Predictor	**Network size & (activation function)**			**Training parameters**		**Avg. Time**
	input	*hidden*	*output*	S_B	*rule*	**(s)**
FIR	5	-	1	-	Fixed coefficients	0.24
NLMS	5	-	1	1	adaptive NLMS	0.33
RLSL	2	2 (lattice)	1	1	adaptive RLSL	0.36
PN	2	-	1 (step)	50	Perceptron / Rosenblatt Rule	0.22
SLFN	4		1 (linear)	1500	LMS	0.22
MLFN	2	1 (linear)	1 (sigmoid)	1000	backpropagation	0.24
GRNN	5	5	1	50	function approximation	0.21
EN	5	2 (sigmoid)	1 (linear)	50	backpropagation	0.43

Table 2: Configuration, Training and Processing Time achieved by the chosen predictors

Consolidating the results of the fault tolerance tests for noise and hardware failures, the compression performance achieved, average processing time and network configuration, it would appear that among the test architectures, the best choice of a predictor for lossless data compression would be the GRNN. Other good performers include the neural MLFN and classical adaptive RLSL, although the RLSL has a relatively lower processing speed and may not be as suitable for real-time applications. It is also noteworthy that it has been shown that in certain cases, multi-level / multi-stage compression techniques can, not only dramatically increase compression performance, but may also significantly increase processing speed [13], [14].

When implementing neural predictors, it is worthy to note that neural networks are generally capable of better pattern recognition and data compression than linear networks. To realize the extent of this potential, however, a sufficient number of PEs would be required. The neural architectures chosen in this paper are very small, in order to minimize complexity and overheads, in order to simulate very small circuitry for remote sensors. Processing time of a larger network on parallel neural hardware would not be significantly greater, but training and simulation time would be affected. This option is not explored here, as the aim of this paper was to provide relative comparisons with the linear predictors using networks of approximately similar low complexity, with the benefit of minimal cost and size of architecture for use in small

supplementary systems, such as an add-on module to the transmitter. In the interest of extending this work, experiments with neural and non-neural hybrid networks, optimized so as to produce efficient processing and lower overheads, may be attempted to realize further compression of telemetry data.

Acknowledgements

The author would like to thank M. U. Siddiqi and C. Eswaran for their valuable contributions towards this research work.

References

[1] Logeswaran, R. and Eswaran, C. (1999) Neural network based lossless coding schemes for telemetry data. *Proc. IEEE Intl. Geosci. and Remote Sens. Symp.*, 4, 2057-2059.

[2] Haykin, S. (1996). *Adaptive Filter Theory*, 3rd. edition, Prentice Hall, New Jersey.

[3] Dony, R.D. (1995) Neural network approaches to image compression. *Proc. of the IEEE*, 83(2), 288-303.

[4] Stearns, S.D. (1995) Arithmetic coding in lossless waveform compression. *IEEE Trans. on Sig. Process.*, 43(8), 1874-1879.

[5] McCoy, J.W., Magotra, N. and Stearns, S. (1994) Lossless predictive coding.. *IEEE Midwest Symp. on Circuits and Sys.*, 927-930.

[6] Logeswaran, R. and Eswaran, C. (1999) Effect of encoders on the performance of lossless two-stage data compression schemes. *IEE Electron. Lett.*, 35(18), 1515-1516.

[7] Logeswaran, R. (2000) Application of Generalised Regression Neural Networks in lossless data compression. *Systems and Control : Theory and Applications*. WSES Press, Greece, 269-274.

[8] Logeswaran, R. (2001) Transmission issues of artificial neural networks in a prediction-based lossless data compression scheme. *IEEE Intl. Conf. on Telecomm.*, 1, 578-583.

[9] Demuth, H. and Beale, M. (1996). *Neural Network Toolbox (for use with MATLAB)*. MathWorks, Massachusetts.

[10] Widrow, B. and Lehr, M.A. (1990) 30 Years of adaptive neural networks : Perceptron, Madaline and Backpropagation. *Proc. of IEEE*, 78(9), 1415-1442.

[11] Wasserman, P. D. (1993). *Advanced Methods in Neural Computing*. Van Nostrand Reinhold, New York , 155-161.

[12] Elman, J. L. (1990) Finding structure in time. *Cognitive Sc.*, 14, 179-211.

[13] Logeswaran, R. (2002) Enhancement of Lempel-Ziv coding using a predictive pre-processor scheme for data compression. *Advances in Info. Sci. and Soft Computing*. WSEAS Press, Greece, 238-243.

[14] Logeswaran, R. and Eswaran, C. (2001) Performance survey of several lossless compression algorithms for telemetry application. *Intl. J. of Comp. and Appl.*, 22(1), 1-9.