SANDIA REPORT

SAND91-8234 • UC-700
Unlimited Release
Printed August 1991

Application of Neural Networks to Flight Test Diagnostics

R. M. Wheeler, Jr. and D. A. Sheaffer

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94551
for the United States Department of Energy
under Contract DE-AC04-76DP00789

SAND91-8234
Unlimited Release
Printed August 1991

SAND--91-8234

DE92 003581

APPLICATION OF NEURAL NETWORKS TO FLIGHT TEST DIAGNOSTICS

Richard M. Wheeler, Jr.
Systems Research Division
Sandia National Laboratories
Livermore, CA 94551
and
Donald A. Sheaffer
Electronic Sensor Division
Sandia National Laboratories
Livermore, CA 94551

Abstract

A system has been designed which can provide summary information about specific noisy electrical pulses that are generated during flight testing. This is important from a telemetry viewpoint, since limited bandwidth often rules out transmitting all of the pulse data. The system is based on a neural network processing paradigm. The neural network serves as a mapping between pulse data inputs and pulse category outputs. Output categories correspond to presence or type of component failure. Extensive computer simulations have shown that the system can recognize qualitative pulse features which are useful for diagnostic purposes. A second version of the system, also using a neural network, was designed to perform data compression. In this case, an entire pulse is efficiently coded for transmission and the original signal is reconstructed upon receiving the coded transmission. Successful simulations for both systems have demonstrated feasibility and have led to a hardware development effort aimed at prototyping a fieldable system. Based on these results, it appears that the neural network approach may be applicable to other diagnostic and data analysis problems arising in component or system testing.

The authors would like to acknowledge the assistance of Dean Clark (8453) and John Williams (8453) who contributed much of the pulse data and clarified relevant telemetry issues and constraints. Insights into the pulse generation process were provided by Doyle Morgan (2561).

APPLICATION OF NEURAL NETWORKS TO FLIGHT TEST DIAGNOSTICS

I. Introduction

A variety of electrical signals are generated during system flight testing. Typically these signals appear as pulses corresponding to specific transient events which are short in duration. The signals, or aggregated measures of the signals, are transmitted to the ground by a telemetry system for subsequent analysis. In general, sending all of the measured data is prohibitive. This is due to the large number of functions and variables being monitored, the limited bandwidth of the telemetry system, and the short time available between occurrence of relevant events and the destruction of the test vehicle. In practice, the data actually transmitted for a particular variable is extremely limited--perhaps only a single number, such as the total energy in the signal. To better assess component performance, it is desirable to have more detailed information available.

There are several approaches available for providing more useful information. In spite of the constraints just mentioned, in some cases sufficient time is available to transmit the entire pulse of interest. This is preferred where possible since it avoids on-board processing and allows data analysis to be done on the ground where time is not critical. Alternatively, data compression techniques can be used to code the pulses of interest, transmit fewer bits than in the original signal, and later reconstruct a close approximation of the signal at a processing station. Human analysis can then be done using the reconstructed signal.

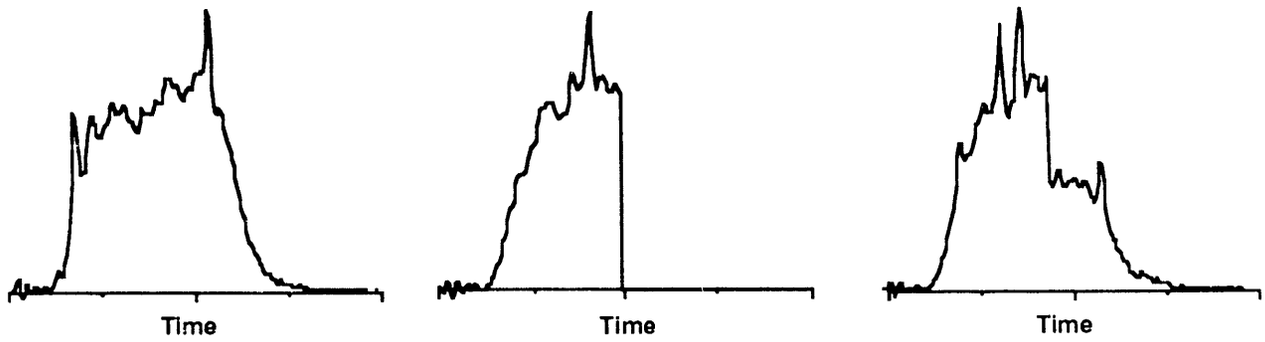
At the other extreme of processing is complete automated analysis leading to the transmission of high-level summary information; for example, that a fault occurred and if so what the failure mode was. Clearly, this increases requirements on the on-board processing system, while easing the burden on the transmission system.

The goal of this study was to design and test via simulation the feasibility of on-board processing systems that can provide more test information than currently available for pulse waveforms. Both high-level failure detection and data compression approaches have been investigated and are discussed in this report. In each case, a neural network is designed to carry out the required computations. A brief description of neural networks is given in section III, followed by the failure detection and data compression problem formulations in sections IV and V. Issues associated with hardware implementation are discussed in section VI. However, for context and background, we first describe some characteristics of the pulses and telemetry constraints assumed for the study.

II. Pulse Characteristics and Telemetry Constraints

1. Pulse characteristics

Typical transient pulses generated during flight testing are shown in Figure 1. Figure 1a represents a nominal pulse generated if the system is functioning normally. The pulse has a rapid initial rise, a sustained amplitude over most of the pulse width and a somewhat slower fall-off at the end. The pulse generally is quite noisy, but the detailed behavior that might be masked by the noise is not as important as the overall pulse shape. This means that the failure modes tend to show up as qualitative differences in the pulse shape and not in the fine structure of the pulse. The pulses in Figure 1b and 1c illustrate typical abnormal pulses, corresponding to two failure modes.



a) Normal b) Abnormal (severe failure) c) Abnormal (partial failure)

Figure 1. Typical Transient Pulses Generated During a Test.

The pulses used for this study were sampled to give 200 samples per pulse. With 8-bit quantization, this results in 1600 bits of data for the original signal.

2. Telemetry constraints

Telemetry constraints are largely dependent on the mode in which testing is done. For example, nondestructive testing allows more processing and longer transmissions due to longer operational times for the telemetry system. On the other hand, destructive testing can impose severe time constraints if events of interest occur close to the time at which the test apparatus is destroyed. We are primarily interested in the latter cases in which transmission of entire signals is not possible. In the finite time window between the start and end of the test, data must be measured, processed and transmitted. Clearly, a tradeoff exists as to the time spent on these functions. For a given transmission rate, the message length (amount of data) determines the time at which transmission must begin. But the message length can depend on the amount of processing done on board; presumably higher level on-board processing would result in less raw data to be sent.

Since only one of the the many signals generated during a flight test was chosen for the current feasibility study, it was assumed that other data transmission requirements would determine the processing and transmission constraints for the pulse of interest.

Transmission constraints:

Due to the other signals requiring transmission, the portion of the transmitted message allocated to the above 1600-bit pulse of interest is assumed to be on the order of 30 bits. The obvious goal is to find the most useful message possible of that length. For data compression techniques, this would require a challenging compression of about 50 to 1.

Processing constraints:

Regardless of the on-board processing executed, the final output must be contained in the above 30 bit message. An additional constraint, also imposed by the short operational time window, is that any processing must be complete by the time message transmission must begin. So not only must a meaningful message of 30 bits be created, it must be created fast. For the application studied, approximately 30 μsec were assumed available for processing. Space constraints are also critical on flight tests. The processing hardware necessary to implement the computations being suggested can be realized with special purpose electronics with modest space requirements. Details of the hardware implementation are provided in section VI.

III. Summary of the Neural Network Approach

Neural networks are highly interconnected networks of simple processing elements inspired by the structure and operation of a biological nervous system. An implicit hope in using nature as a guide is to be able to mimic in the artificial networks those powerful information processing capabilities (e.g. pattern recognition of visual or aural information) of biological organisms. Potential advantages of such an architecture are increased processing speed, fault tolerance and adaptability to new environments.

While many architectures and processing algorithms have been proposed, the most commonly used neural network is the multi-layered feedforward network shown in Figure 2. The circles represent the processing nodes (neurons) and the directed links depict the communication paths in the network. Each link has a weight w_{ij} associated with it indicating the strength of the connection between nodes i and j. In this model, the network is simply a static mapping between sensory signals and category classifications--a pattern recognition problem. The network receives sensory signals at the input nodes, propagates them in the feedforward direction through the hidden nodes, and produces a final output (vector) at the output nodes. Each processor in the hidden layer and output layer computes its own output according to the rule

$$x_j = f\left(\sum_{i=1}^{N} w_{ij}\, x_i\right) \qquad (1)$$

where x_i is the output of node i, w_{ij} the weight between nodes i and j, N is the number of nodes feeding into node j, and $f(\ )$ is a nonlinear operator with the properties: $f(\ )$ is smooth and monotonically increasing, $f(-\infty) = 0$ and $f(+\infty) = 1$. Although many nonlinear functions are possible, a common form of $f(\ )$, and the one used in the current study, is the S-shaped sigmoid function $f(x) = \frac{1}{1+e^{-x}}$.

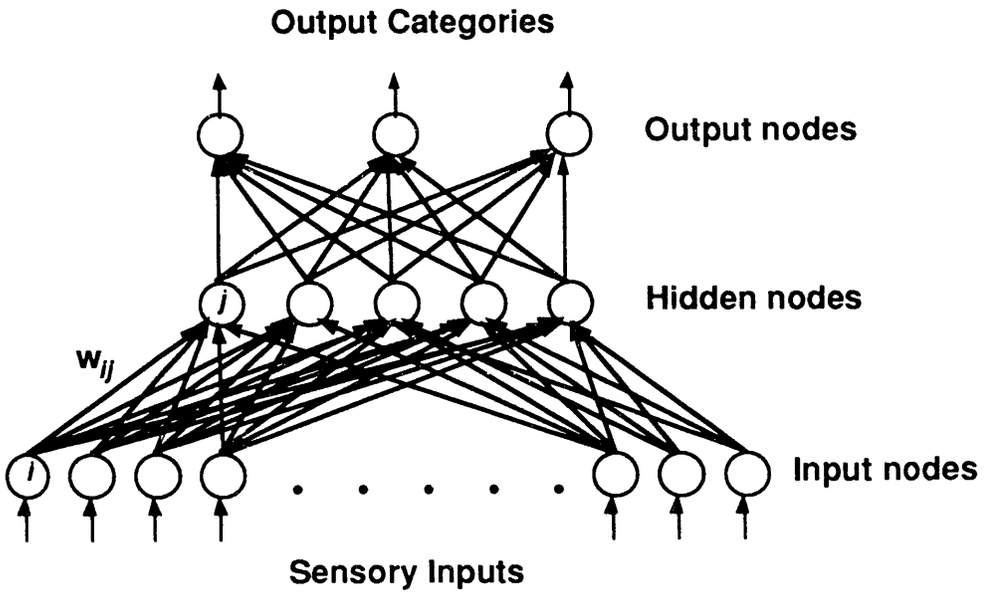


Figure 2. A 2-Layer Feedforward Neural Network.

The processing in this type of network is rather simple. However, the number of nodes will determine the number of computations required. The number of input and output nodes is set by the coding used for data representation. The number of hidden nodes is somewhat arbitrary; however, it should be large enough to permit the network to realize the desired input-output mapping, but small enough to avoid overspecification of the mapping (fitting the noise) as well as excessive computation. A key feature of this network is that the weights are adjustable and can be modified to reduce the error between the actual output produced and the desired output for a given input. A number of rules for modifying the weights (referred to as learning or training) are available. Since in the network of Figure 2 the output depends nonlinearly on the weights (see equation (1)), local optimization methods such as gradient descent are most often used. While network training via gradient descent can be very time consuming, it is done off-line. In implementation, the network nodes simply operate once each to produce the network output.

There is a rapidly growing literature in the neural network field, providing perspectives ranging from biology and psychology to engineering and computer science. Detailed discussions of the topic, including the type of network used here, are found in [1-3].

The challenges to the neural network approach in the current application are two-fold: 1) can a set of weights be found to transform the sampled pulse data into a useful 30-bit message?, and 2) can the computations required by the network (equation (1) for each node) be executed in the allotted 30 μsec?

IV. Failure Detection

Detecting abnormal component performance is one of the primary objectives of testing. This can often be done by visual inspection of the signal produced by component operation, provided an accurate signal is available. The principal product of analysis is the detection of abnormal behavior and, if found, the identification of the type of failure observed.

One can think of the system proposed here as an on-board "machine analyst" whose job is to quickly interpret a generated pulse and then transmit a short message about that pulse. This concept is shown in Figure 3. The analysis function is the task of a neural network, trained to discriminate abnormal pulse patterns from normal patterns.

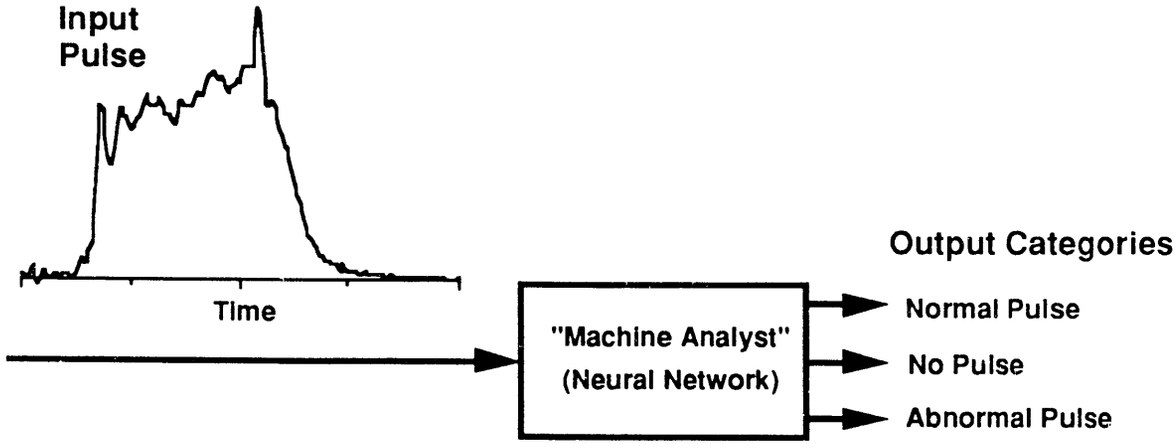


Figure 3. Failure Detection as a Classification Problem.

For the current evaluation, the goal of the machine analyst was to classify a given input pulse into one of three output categories--"normal pulse", "no pulse", or "abnormal pulse". The "no pulse" category was included to reflect a total failure in which case the pulse could be simply white noise. The abnormal category encompasses all types of abnormalities, regardless of specific failure mode. In this formulation then, the transmitted message reflects whether a pulse was generated and, if so, whether it was normal or abnormal. It does not distinguish between types of abnormalities.[1]

1. Neural Network Implementation

The neural network used for the failure detection problem is that shown in Figure 2, with 200 input nodes, 10 hidden nodes and 3 output nodes. The raw data consisting of 200 samples per pulse was used as the input. This input data was normalized by dividing all the data by the maximum value present in the data set. The 3 output nodes correspond to

[1]This problem was formulated to demonstrate proof of concept. In principle, more output categories can be used, corresponding to different classes of interest. In fact, the neural network model currently being implemented has several additional output nodes to account for different failure modes (see section VI).

the three categories shown in Figure 3 (output 1 = "normal", etc.) Output values range from 0 to 1. The coding used assigned a value of 1 to the output node corresponding to the correct category and a value of 0 to the other nodes. For example, if a normal pulse is presented at the input, the correct output vector should be [1,0,0]. After several experiments varying the number of hidden nodes, a value of 10 was used for all of the results presented here.

2. Experimental Data

The data set used for this study consisted of 120 pulses and included 60 "normal pulses", 20 "no pulses", and 40 "abnormal pulses". Examples of normal, no and abnormal pulses are shown in Figures 4, 5 and 6 respectively. The entire set of abnormal pulses is shown in Figure 6, since this is the most diverse and interesting category.

"Normal pulses" were generated from laboratory experiments and are representative of flight test pulses. "No pulses" were generated synthetically as zero-mean white noise random sequences. "Abnormal pulses" were created artificially by manipulating the normal pulses in various ways to reflect plausible failure modes. Hence, the pulses in Figure 6 are not real pulses, but do in some cases capture the features of pulses corresponding to known failure modes. In other cases, it was later learned that the abnormality created did not correspond to any known failure. This information was used to modify the data set for subsequent experiments with a VLSI neural network chip (see section VI). In summary, the abnormal data used in the simulations described below and in section V are more diverse than would reasonably be expected in actual tests. The fact that the neural network performed satisfactorily with these data gives confidence that a network trained with more realistic and less challenging data will perform even better.

3. Simulation Studies

The weight adjustment procedure, also called network training, uses the input-output data explicitly. The weight values are updated based on the error between the actual output of the network in response to a given input and the correct response (classification) for that input. Generally, the data set is divided into a training set and a test set. The training data is used to arrive at a set of weights that properly classify the training data. The test data are used to see how well the network performs on data it has not seen, using weights found during training. Clearly, it is necessary that the training set be representative of the entire data set so that generalization is possible.

A Note About Performance:

Since the network computes with real numbers, values of the output nodes will never be exactly 0 or 1; the output vector $[x_1,x_2,x_3]$ will consist of three real numbers. Intuitively, if the output is near the desired output vector, then the input is correctly classified. To quantify performance, two decision rules were used:

1) *Threshold rule:* This rule uses upper and lower thresholds. Proper classification requires that the output node corresponding to the correct category be above the upper threshold and all other output nodes be below the lower threshold. If another node exceeds the upper threshold, and the "correct" node and all other nodes are below the lower one, then a misclassification occurs. In all other cases, a "cannot classify" decision is made.

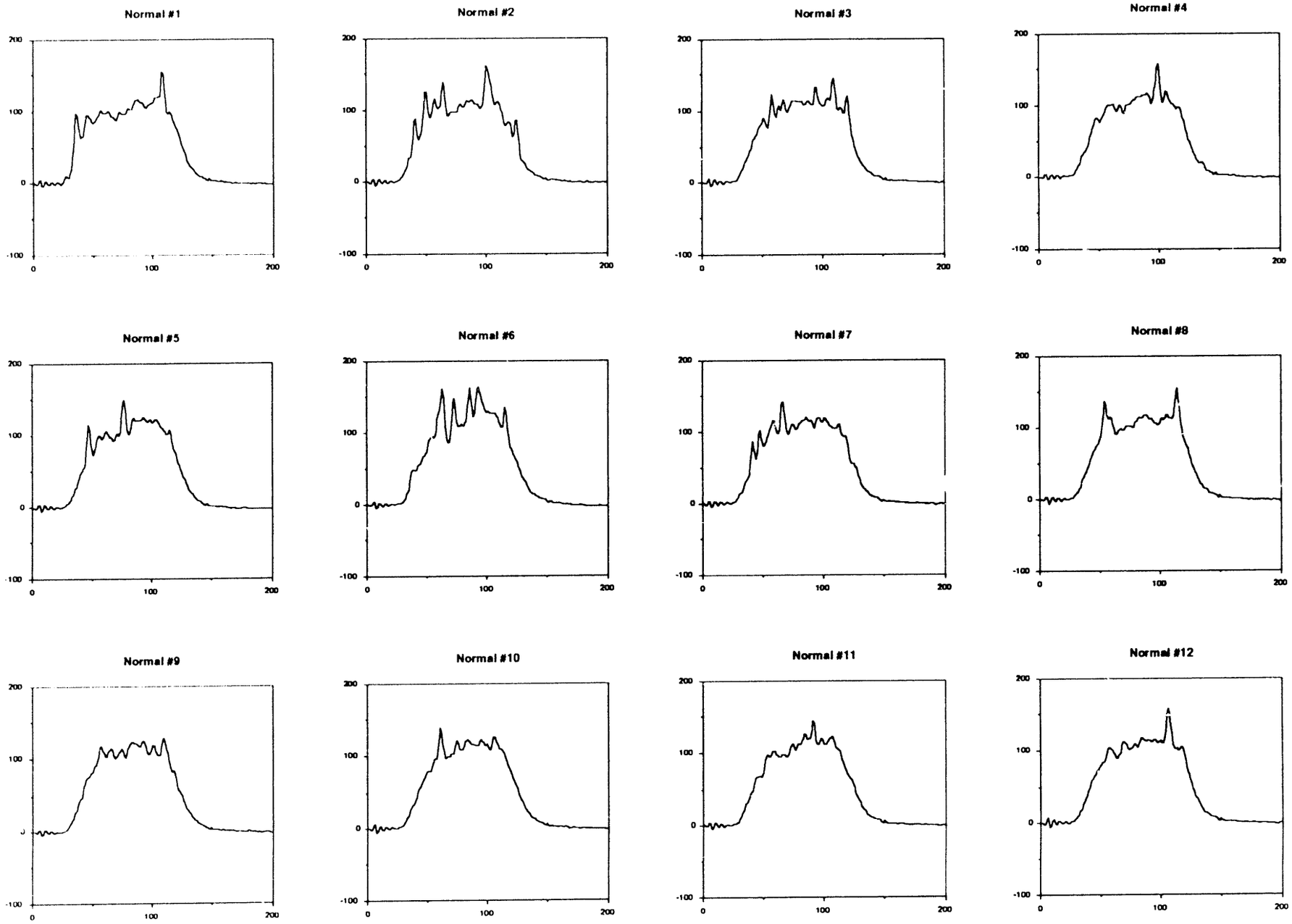


Figure 4. Typical Examples of Normal Pulses: Magnitude of 8-bit sampled pulse is plotted vs. sampling time.

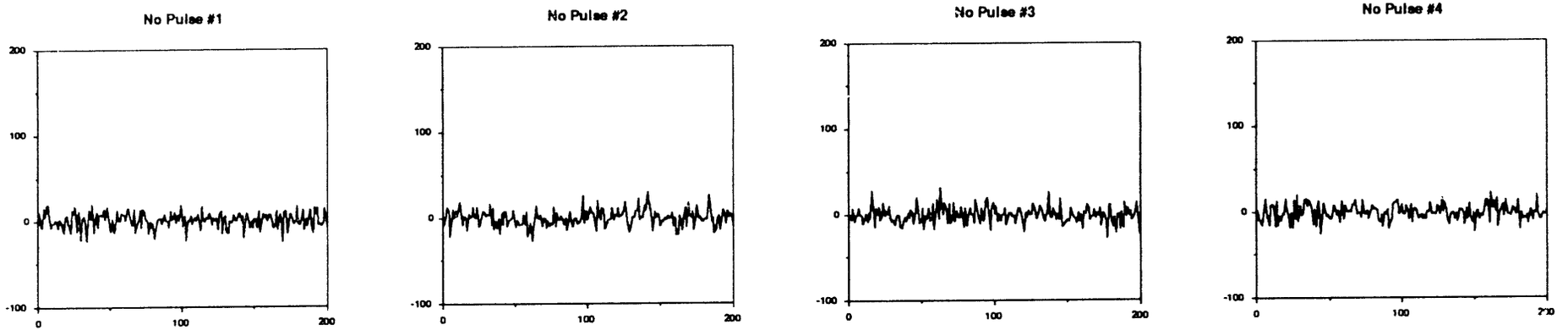


Figure 5. Typical Examples of "No" Pulses: Magnitude of 8-bit sampled pulse is plotted vs. sampling time.

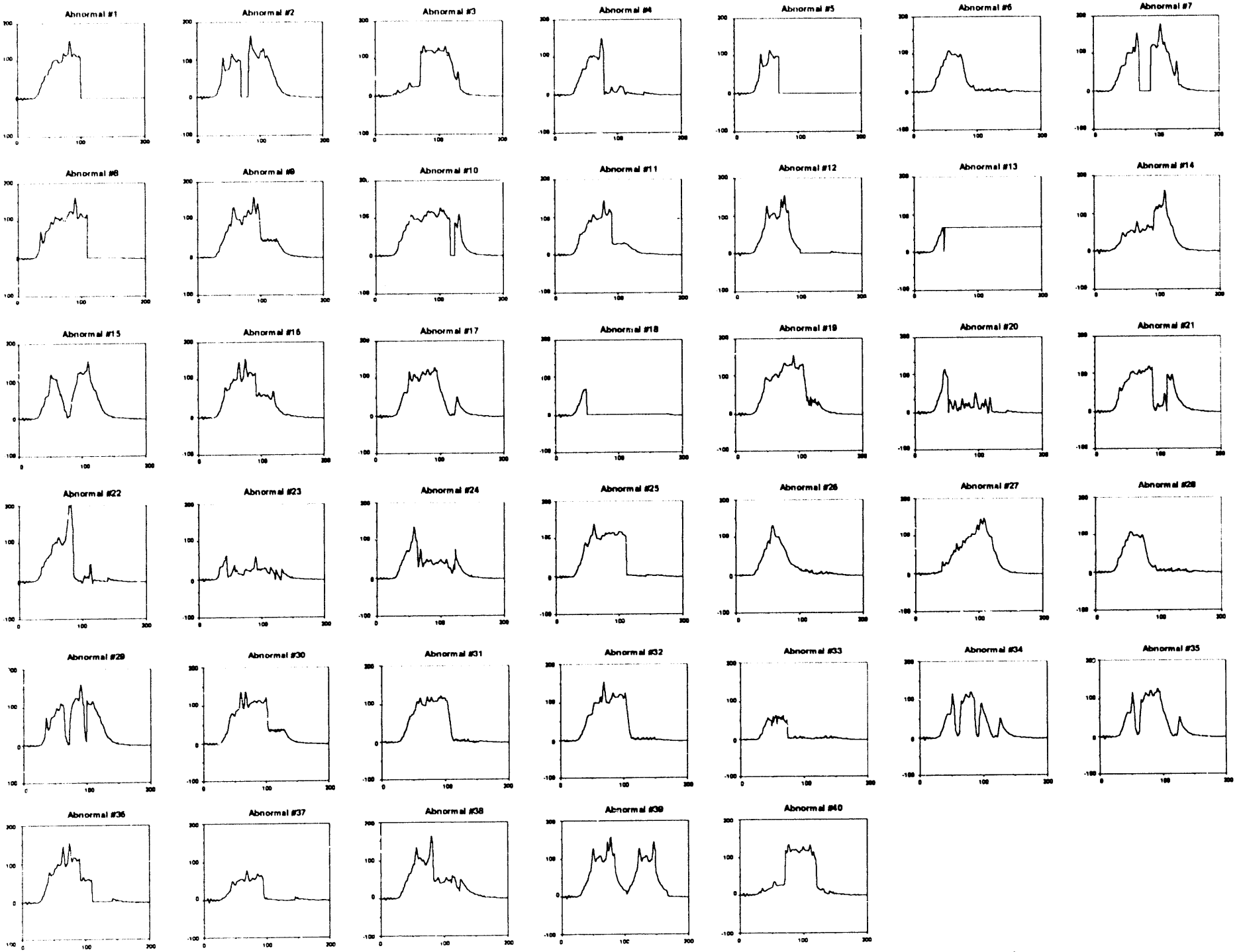


Figure 6. Typical Examples of Abnormal Pulses: Magnitude of 8-bit sampled pulse is plotted vs. sampling time.

2) *Winner-take-all rule:* In this case, the input is assigned to that class whose output node has the maximum value, regardless of the absolute magnitudes. If the maximum value belongs to the node corresponding to the true input category, then the classification is correct; otherwise it is incorrect.

For both decision rules, *error rate*, defined as the ratio of misclassified pulses to the number of pulses in the test set, was used as the measure of performance. While the results are similar for the two rules, only the winner-take-all rule results are reported below.

In the first experiment, the entire data set was used for training. This was done to determine the upper bound on performance. If satisfactory weights cannot be found in this case, then the approach may not be feasible. In fact, weights were found which were able to correctly classify every pulse in the data set. If every pulse expected to be seen in actual testing were close enough to a pulse in the data set used, then the network could be used confidently with the weights determined in this case. In general, it is desirable to test the ability of the network to generalize to patterns that it has not seen. This was done in the next set of experiments.

Use of Different Training Sets:

To test network generalization, different training sets were used in finding network weights. In each case, the remaining data were used for testing. Three different training set sizes were considered, corresponding to 30%, 50% and 75% of the total data. The specific members of the set were chosen randomly from each class. For example, in the 30% case a total of 36 pulses (.3(120)=36) were used for training, consisting of 18 normal, 6 no pulse and 12 abnormal. For each case, four different random selections were made, a new network was trained and tested, and the results were averaged to give an average error rate for each training set size. These experiments are summarized in Table 1. The number of errors and error rates shown in the table are computed using the winner-take-all decision rule.

Percent of Data Used for Training	Number of Training Patterns	Number of Test Patterns	Run Number	Training Set Errors	Test Set Errors	Average Error Rate (%)
30	36	84	1	0	17	9.5
			2	0	14	
			3	0	6	
			4	0	5	
50	60	60	1	0	4	5.0
			2	0	4	
			3	0	2	
			4	0	2	
75	90	30	1	0	1	1.3
			2	0	1	
			3	0	0	
			4	0	0	
100	120	0	1	0	0	0

Table 1. Training and Testing Performance of the Failure Detection Network.

In no cases were any errors made in classifying the training set. When errors were made on the test set, they were only in misclassifying abnormal pulses, usually as normal pulses. This is not surprising since the abnormal pulses were created by distorting normal pulses. Depending on the amount of distortion, an abnormal pulse actually could be quite similar to a normal pulse. For example, in one 30% training set simulation, abnormal pulses #2, #10 and #29 (see Figure 6) were classified as normal and, apart from the narrow vertical notches present, they are very close to pulses that should be classified as normal. The only abnormal pulse classified as "no pulse" was #18, which is certainly a low energy pulse if not a "no pulse".

A second example shows the sensitivity of error rates to the specific composition of the test set when few errors are made. In two runs with the 75% training sets, one error was made in the first run and zero errors were made in the second. In the first run abnormal #2 was classified as normal, but in the second run it was properly classified even though the pulse was contained in the test set in both cases. Close inspection of Figure 6 shows that pulses #2 and #5 are identical over roughly half the pulse duration. In the first run pulse #5 was also in the test set and hence this similarity was never learned. However, in the second run pulse #5 was in the training set and the network generalized that pulse #2 was sufficiently similar to call it abnormal.

The above discussion of errors leads to the conclusion that errors made by the network are in some sense "reasonable" and not completely random. Abnormal pulses that are underrepresented in the training set or that are "close" to normal pulses are more prone to misclassification. This problem can be addressed by adding more data to the training set which are similar to the problematic pulses. In certain cases a larger number of network nodes may be required.

Some of the errors observed in these experiments involved pulses which subsequently were shown to be unrealistic. These were removed from the data set used to train the neural network chip described in section VI. For example, expert opinion determined that the "notched" pulses such as abnormal #2, #7, #10, #29, #34, and #35 were very unlikely to occur in an actual system. Therefore, from a practical standpoint, the problems observed in classifying these pulses are not of major concern.

A final observation is that error rate decreases monotonically as the training set size increases. This is shown in the last column of Table 1. Not surprisingly, performance improves as more training data are available. An indicator of network robustness is that using just half the data for training, an error rate of only 5% was obtained. Presumably, more data also implies more representative data, which is the key determinant of network performance. Assuming the entire data set used for this study is indeed representative of all pulses that could be observed, the performance of the neural network trained on the whole data set should be very good in actual testing.

V. Data Compression

Although the failure detection problem formulation can be viewed as a form of data compression, usually data compression implies reconstruction of the original data after transmission. Even in cases where original data cannot be obtained, an approximation to that data in the form of a reconstructed signal would assist component designers. A neural network can be used for this purpose and is described in this section.

1. Neural Network Implementation

The goal of any data compression scheme is to develop a code that eliminates redundant information. A neural network achieves this coding implicitly in the network weights. When properly trained, the weights contain all the information necessary to compress and reconstruct the data. A network to implement data compression is shown in Figure 7.

The general architecture is the same as in the previous case, but the number of output nodes is now equal to the number of input nodes. As before, the input nodes correspond to the pulse samples. However, the output nodes correspond not to categories, but to samples of a reconstructed pulse. If perfectly trained, the network should reproduce at the output the same signal that was presented at the input. Data compression is achieved by using a smaller number of hidden nodes than input and output nodes. Instead of transmitting the original samples, the values at the hidden nodes (computed from equation (1)) are sent as the compressed message. Assuming these values are quantized as 8-bit numbers, and there are M hidden nodes, the compressed message is 8M bits long. If there are N original samples, this results in a compression of N to M.

Notice that in this formulation only half of the network processing is done on-board (input-to-hidden layer compression). The hidden-to-output layer reconstruction is done off-line after the transmission has been received. Therefore only the first layer weights are needed on-board and only the second layer weights are needed for off-line processing.

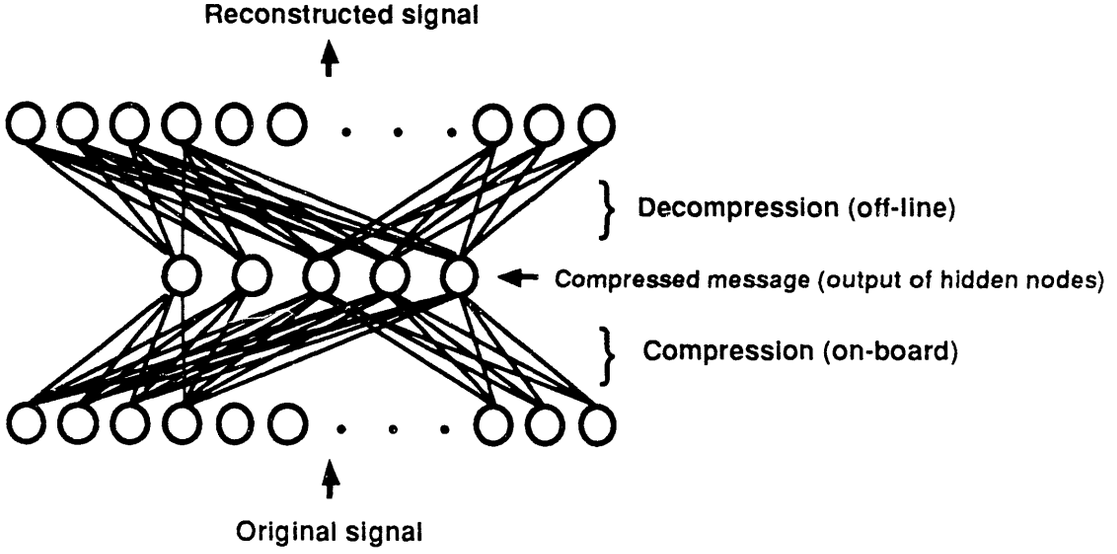


Figure 7. A Neural Network Architecture for Data Compression.

2. Simulation Studies

The same data set used in the failure detection network was used for training the data compression network. However, only 100 samples per pulse were used (every other sample of the original pulses) to reduce the number of weights to be trained. The differences between the 200-sample pulses and the 100-sample pulses are virtually undetectable by visual inspection. Therefore the network used in the simulations had 100 input and output nodes. The number of hidden nodes was varied to study the tradeoff between compressed message length (number of hidden nodes) and reconstruction accuracy.

Simulation results for four different hidden layer sizes (3, 4, 5 and 10 nodes) are illustrated in Figures 8-12. In each plot, the original pulse is the solid curve and the reconstructed pulse is the dotted curve. In Figure 8, four normal pulses and two no pulses are shown for the 3 hidden node case. The difference between the original and reconstructed pulses is seen to be mainly in the spike-like features. Since these features were not of primary interest, results for the normal and no pulse categories are only shown for the 3 node case (the approximation is even better with more hidden nodes). Even with only 3 nodes, these two pulse categories can be reconstructed rather well.

Figures 9-12 show the reconstruction results on eight abnormal pulses for 3, 4, 5 and 10 hidden nodes. In the 3 node case (Figure 9), the general trends are captured, but clearly some key features of the pulse are not reproduced, notably the vertical notches in pulses #2 and #7. These features do become evident in the reconstructions using 4 and 5 nodes (Figures 10 and 11). With 10 hidden nodes (Figure 12), the reconstruction of all pulses is nearly perfect. It appears that even in the 4 and 5 hidden node cases, the reconstruction is adequate for diagnosing qualitative aspects of performance currently of interest. As in the case of the failure detection network, the main problems in data compression are with sharp edges (e.g the "notch" pulses). As stated above, such phenomena are not thought to be relevant; however, even for these features, better accuracy performance can be obtained at the expense of decreased compression.

The qualitative results in Figures 8-12 can be quantified by using an appropriate error criterion. The reconstruction accuracy using the root-mean-square (RMS) error[2] between the original and reconstructed pulses is shown in Figure 13. Since for M hidden nodes the compressed message length is simply 8M, Figure 13 provides an explicit tradeoff between accuracy and message length. While reconstruction accuracy required by component designers has not been completely determined, the 4 node case (which is the "knee" in the curve of Figure 13) is a reasonable choice since a large marginal reduction in RMS error is achieved and the compressed message length is only 32 bits. This is consistent with the desired length of 30 bits and results in a compression of 25 to 1.

---

[2]Overall RMS error is computed by first finding the RMS error for each pair of pulses (original and reconstructed) and then averaging over all pulse pairs. For one pulse pair the error is:

$$\text{RMS error} = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_j - y_j)^2}$$

x_j and y_j are the jth samples of the original and reconstructed pulses, N is the number of samples per pulse.

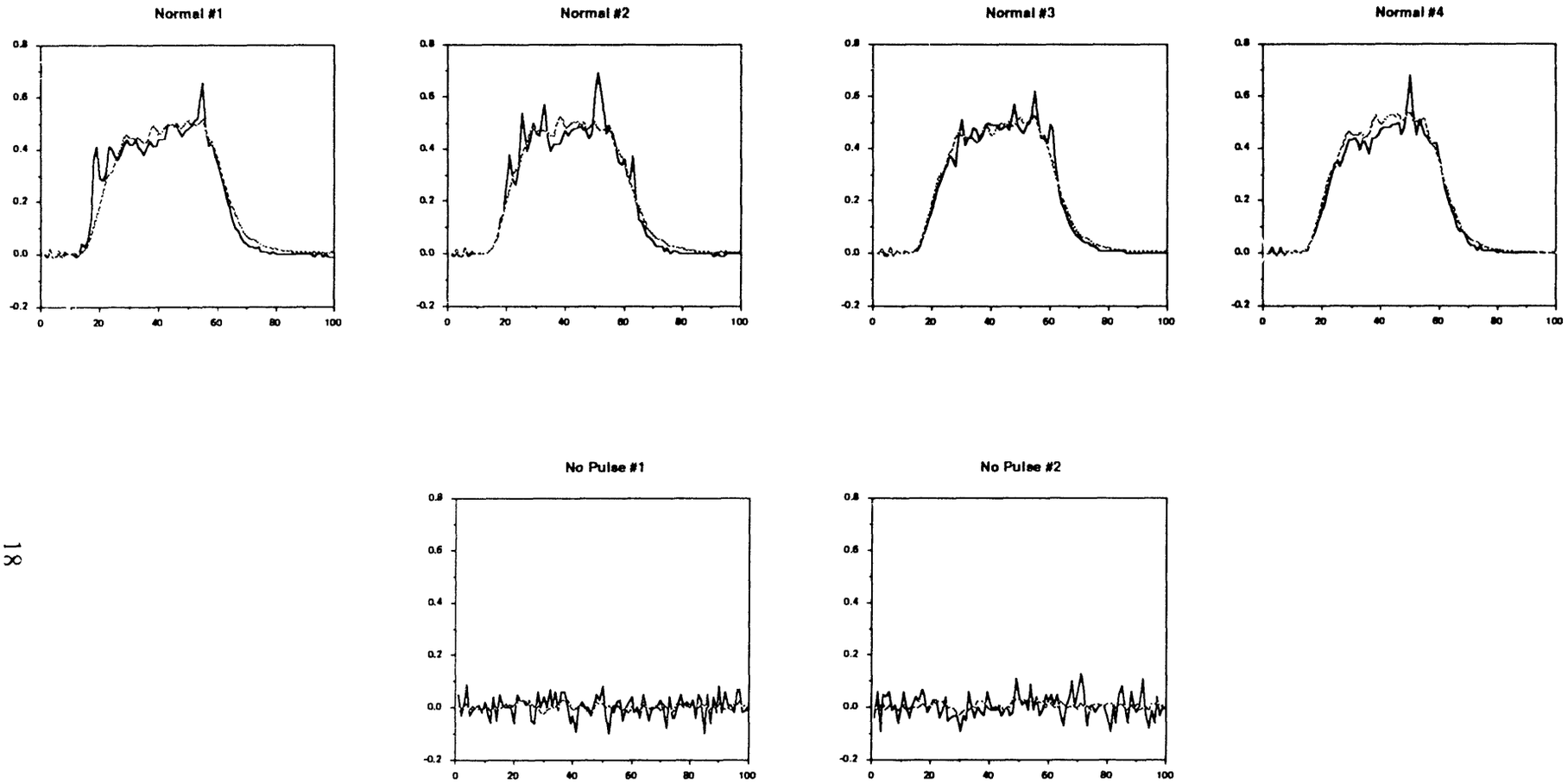


Figure 8. 3 Hidden Node Case: Data compression results for 4 normal and 2 no pulse samples. Original pulse is the solid curve. Reconstructed pulse is the dotted curve.

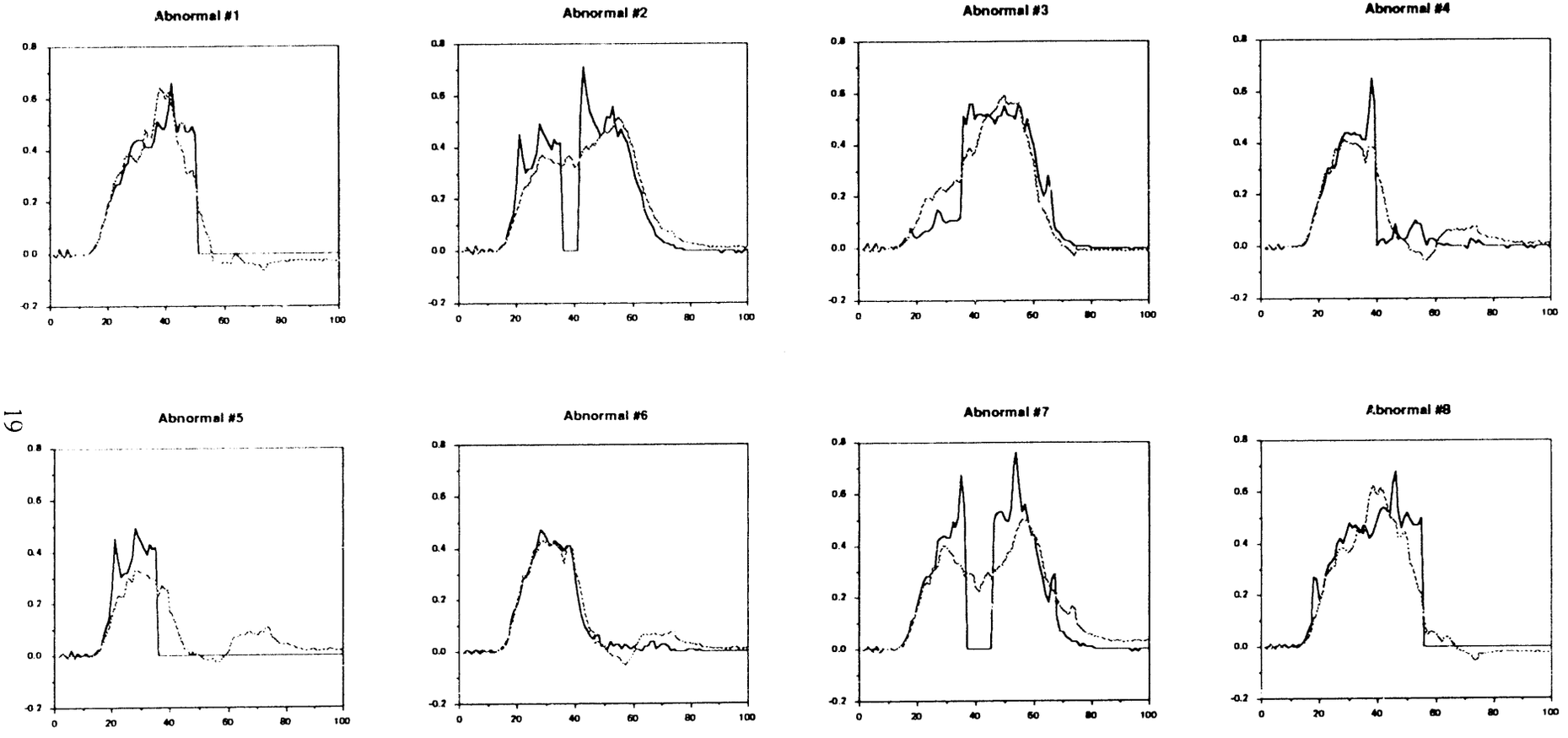


Figure 9. 3 Hidden Node Case: Data compression results for 8 abnormal pulse samples. Original pulse is the solid curve. Reconstructed pulse is the dotted curve.

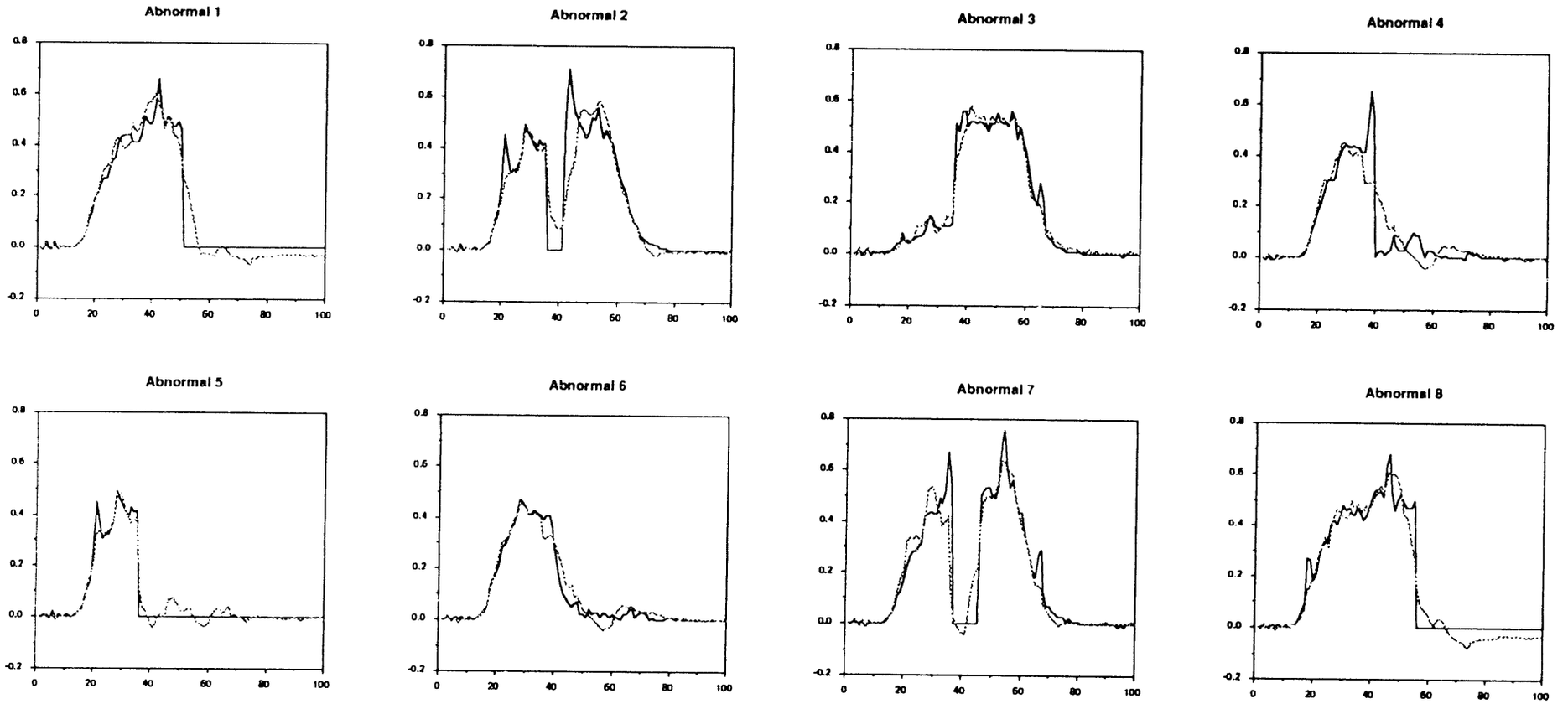


Figure 10. 4 Hidden Node Case: Data compression results for 8 abnormal pulse samples. Original pulse is the solid curve. Reconstructed pulse is the dotted curve.

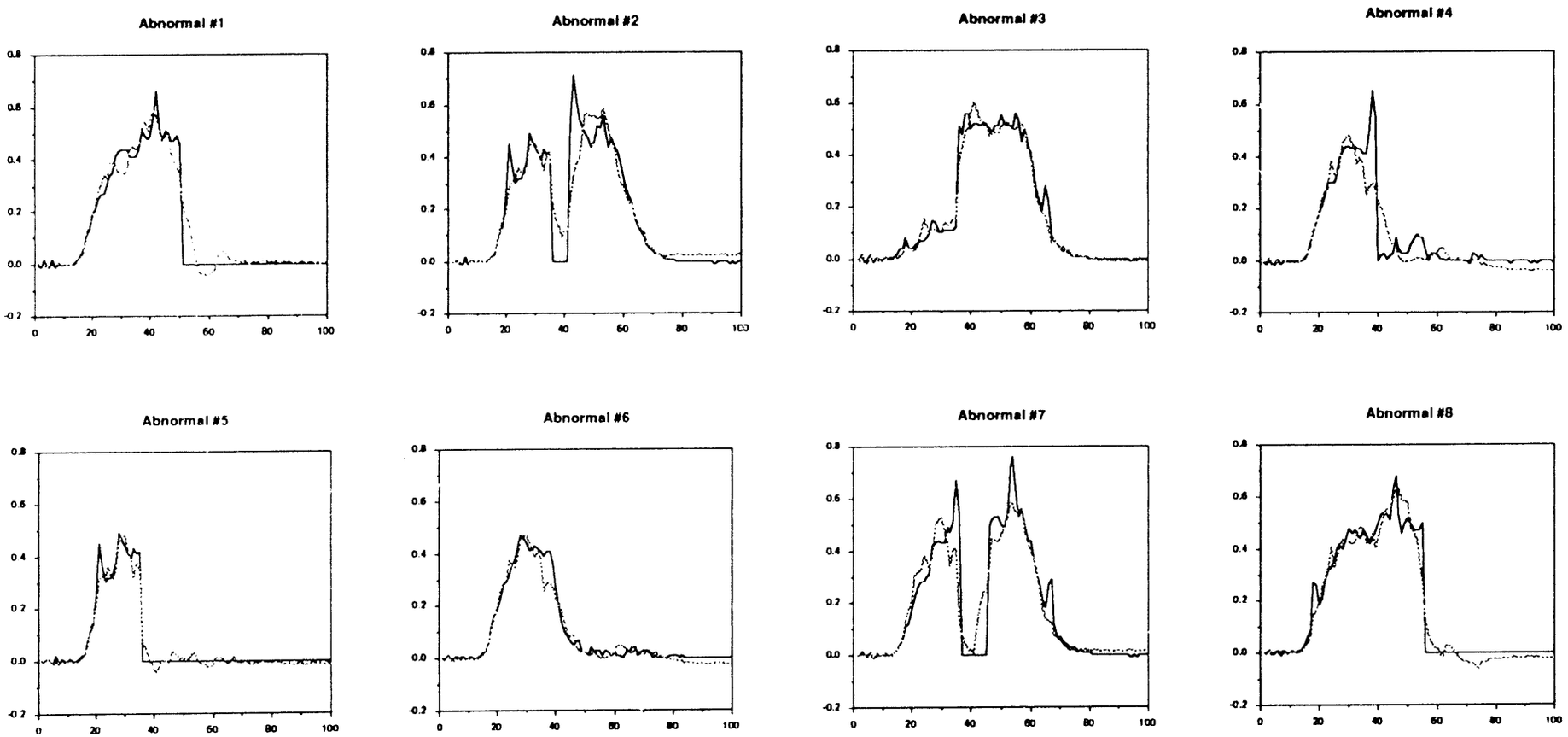


Figure 11. 5 Hidden Node Case: Data compression results for 8 abnormal pulse samples. Original pulse is the solid curve. Reconstructed pulse is the dotted curve.

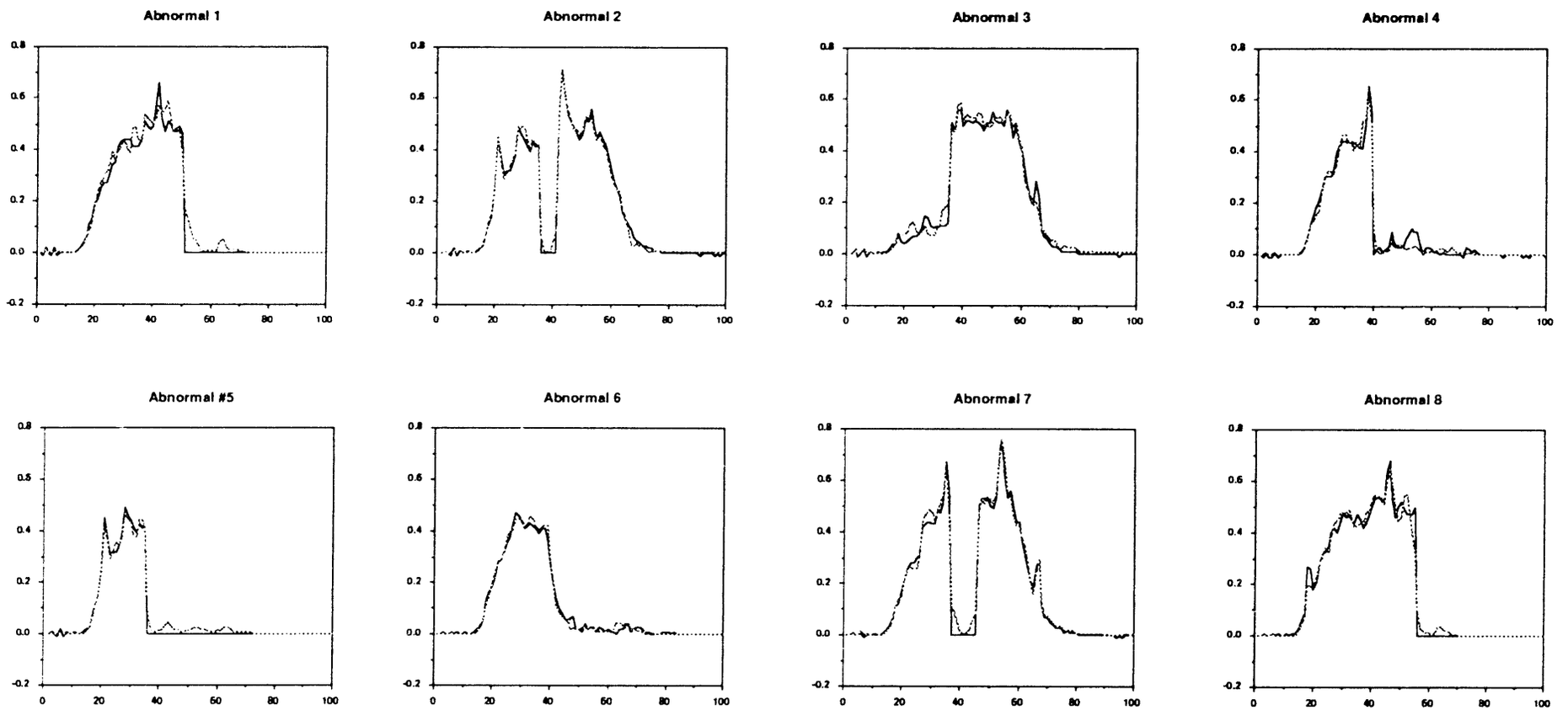


Figure 12 10 Hidden Node Case: Data compression results for 8 abnormal pulse samples. Original pulse is the solid curve. Reconstructed pulse is the dotted curve.

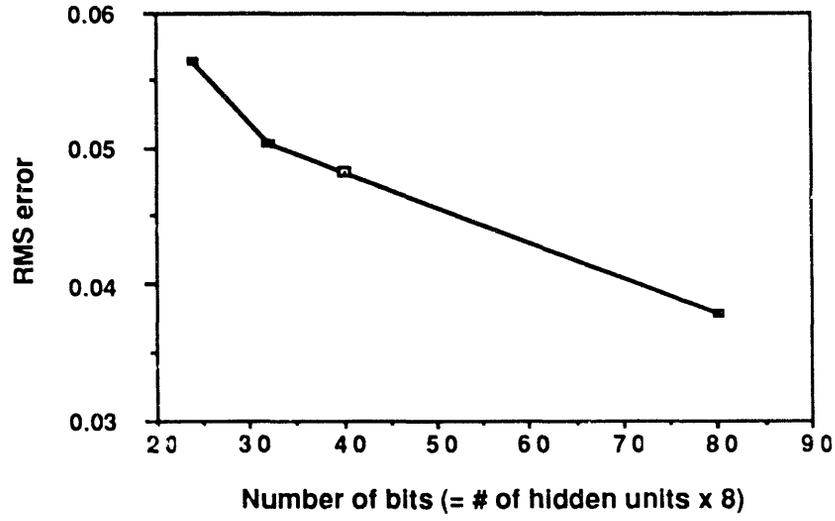


Figure 13. Tradeoff Between Reconstruction Accuracy and Message Length.

The results shown in Figures 8-13 were derived using the entire data set for training. These results, though preliminary, are encouraging. Testing the data compression network on new pulses, as in the failure detection network, would be a useful evaluation of network generalization. This will be pursued in follow-on studies if the data compression concept is found to be valuable by design engineers.

VI. Hardware Implementation and Real-Time Performance

We have seen that a properly trained neural network is capable of extracting useful summary information from low level data. In the telemetry application, the message to be sent using either the failure detection network or the data compression network satisfies the specified bit length constraint of 30 bits. The question remains as to whether either network can be realized in hardware to do the required processing in the short time available in many actual flight tests. Initial investigative work leading to a system level design has been completed. The critical features of the design are described in this section.

1. Neural Network Hardware

Until recently, the hardware for this system did not exist. A new exploratory device from Intel Corporation has the capability to meet the extreme requirements of the telemetry application. This integrated circuit (IC), the ETANN component 80170NX, implements in analog circuitry a neural network capable of performing 2 billion connections per second. One connection is equivalent to one multiply/accumulate in traditional von Neumann digital computing. Two main advantages of this chip over traditional hardware are the high computation speed of the device and the ability to tolerate localized failures in executing the neural network computations.

The ETANN component has 64 inputs and outputs and is capable of computing a two-layer feedforward neural network with as many as 64 hidden nodes in 6 μsec. The chip contains 10,240 programmable weights which are stored in the form of analog charges using an

Intel patented technology called the CHMOS III EEPROM process. A development system is used to program the IC weights. All inputs and outputs to the IC are analog voltages, while the control lines to the IC are digital (5 volt logic compatibility).

Due to the low precision of the ETANN IC, equivalent to 7-bit precision or 128 quantization levels in digital computing, the neural network algorithm performance of the real chip is not as precise as computer simulation (the dynamic range is limited). For this reason, we have chosen to implement the failure detection network rather than the data compression network which requires more precision. Consequently, the hardware design presented below refers to the classification network as described in section IV.

2. System Design

A system block diagram for the classification network is shown in Figure 14. There are three major blocks to the system. The input analog bucket quantizer is used to convert the time duration pulse to quantized parallel data for input to the neural network at one time.

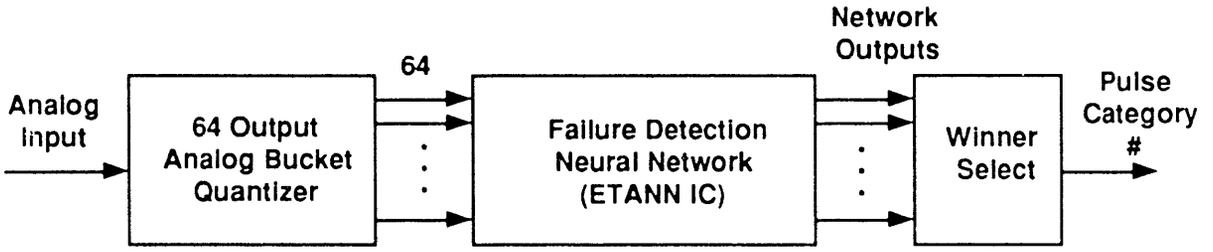


Figure 14. System Block Diagram of the Transient Pulse Classifier.

The ETANN IC computes the neural network processing as discussed in section III. The output comparator network is used to select the pulse category using the winner-take-all rule described earlier. Since the bucket quantizer is nearly instantaneous, once a pulse is generated the total system time for classification is the 6 μsec for feedforward computing plus the propagation delay of the comparator network (150 nsec). This time is well within the 30 μsec assumed to be available on a flight test. The three blocks in Figure 14 are described below.

Analog Bucket Quantizer

The analog bucket quantizer plays two roles as the input block to the ETANN IC. First, it must quantize each time division bucket as the analog pulse presents itself. Second, in each bucket, it must prescale the analog signal to maximize the dynamic range of the neural network at the input. One bucket is illustrated in Figure 15. The circuit consists of a fast amplifier (slew rate = 300V/sec) at the input, a resistor divider scaling network, a transistor switch, and a capacitor storage. At the proper time the switch is turned on and the capacitor is charged. The capacitor must hold the charge until all the buckets are full (one for each pulse sample), at which time the voltages are simultaneously presented to the neural network input. The resistor divider network values can be determined from prior knowledge of the known transient pulses.

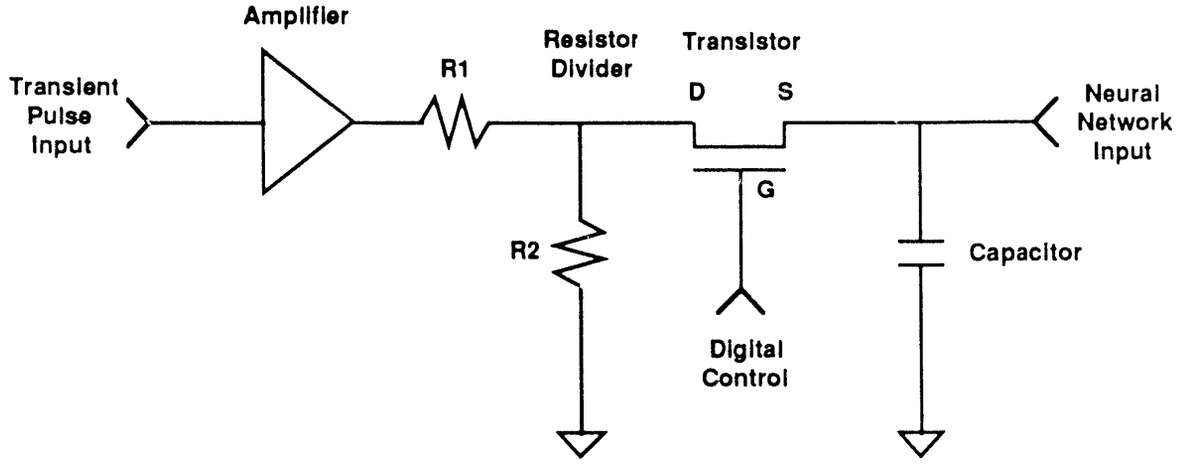


Figure 15. One Analog Bucket Quantizer Circuit.

Due to the chip architecture, there must be 64 input buckets to the ETANN neural network chip. Each bucket corresponds to one analog time slice of the transient pulse. However, 64 individual bucket quantizer circuits are not necessary. This is because prescaling can be performed in groups of adjacent time slices. Initially, six resistor divider networks will be used to cover the dynamic range of the signal. Six amplifiers will be required, one for each resistor divider network. To generate the 64 outputs of the quantizer block, 64 capacitors and transistors (p-channel field effect type) will be used.

Timing for the analog bucket quantizer will be achieved by a 64 bit shift register clocked at the desired time slice sample rate. Shifting an activating level through the register will provide time slice analog bucket storage timing. The main requirement is that a "start to measure" signal must be provided. The accuracy of the system will depend heavily on this "start to measure" signal. A 20 percent time skew will be acceptable for this measurement. Skew in excess of this will result in classification errors due to the different divider networks used in the bucket quantizer to maximize dynamic range.

The operation of the analog bucket quantizer network is a proven technology. The design is very similar to an analog bucket brigade network. Such networks have been used in the audio and low frequency video world as delay units for many years. The main difference is that the bucket brigade is serial in and serial out and our network is serial in and parallel out. Also, for our application the bucket quantizer must operate at a slightly higher frequency than is normal for traditional bucket brigade networks. At this frequency signal degradation may become a problem. Allowable levels of signal degradation will be determined experimentally with a prototype printed circuit board currently under construction. The board will be used to analyze this problem in greater depth.

Physically, the bucket quantizer block will be fabricated as a hybrid in the final system. Preliminary estimates from our hybrid lab indicate that a 1 by 3 inch rectangular shaped hybrid can be built. No major difficulties are apparent at this time.

ETANN Neural Network IC

The second block to the system is the ETANN IC. This component requires a minimal amount of peripheral control circuitry. Excluding the 64 analog inputs, enabling and

clocking are the main requirements. A clock pulse is needed to latch the parallel analog inputs simultaneously and begin the first layer of processing. A second clock pulse is required to start the second layer processing 3 μsec after the first layer is started. The result can be observed at the output 3 μsec after the second layer computation is started.

A feasibility demonstration using the ETANN component was performed at Intel. A two-layer neural network classifier, as shown in Figure 2, was implemented. The network used 64 inputs, 64 hidden nodes and 8 output nodes, one for each of 8 output categories. The ETANN chip was trained using the Intel development system for a specified set of data.

A data set of 410 pulses was created to investigate certain attributes that might be present in a real system. Among these attributes were the measurement of time skew, tolerance to amplitude variation, tolerance to spike noise, and classification of different classes and subclasses. Pulses reflecting these variations were included in the set along with many of the pulses shown in Figures 4-6. Expert opinion on likely failure modes led to the exclusion of some of the abnormal pulses of Figure 6, which were considered unreasonable. Therefore, the data set used in training the ETANN chip was more representative of expected pulses and not as variable as the set used in the simulations of sections IV and V. All of the data pulses were used for training the ETANN component. Time limitations did not allow for training and testing on different data sets. After training, the network classified correctly the entire training set. As indicated earlier, although training can never be exhaustive, the training set actually used must be as representative as possible of the pulses expected to be generated in actual tests. Since perfect training performance does not guarantee perfect test performance (e.g. see section IV results), more extensive training and testing will be done prior to setting the weights in the prototype system.

As previously stated, the abnormal data used in sections IV and V were useful in demonstrating basic capabilities of the neural network, but were not truly representative of actual failure modes. To better capture pulse features of interest, eight classes of patterns were used for the ETANN chip experiments and are illustrated in Figure 16. It is felt that these classes correspond to pulses more likely to occur given realistic failure modes. There were three basic classes of pulses used for training, termed "normal", "drop-out", and "partial drop-out". Within two of the classes, normal and drop-out, there were subclasses. The three subclasses for normal pulses dealt with time skew (start time misalignment), and are referenced as early, mid and late. Drop-out was broken into four classes -- early, early-mid, mid and late. Time skew was used for drop-out pulses but ignored in classification. Within subclasses as much deviation as possible was given to the pulses without moving into adjacent classes. For each pulse, amplitude variation of 50 to 150 percent was used and random noise was added. An exact count of each class of data pulses is given in Table 2. The ability of the ETANN component to classify these pulses correctly shows promise for use in an actual system.

Normal	Drop-out	Partial Drop-out
41 early	48 early	60 partial
39 mid	112 early-mid	
20 late	60 mid	
	30 late	

Table 2. Composition of 8 Categories Used for Training the Intel ETANN Chip.

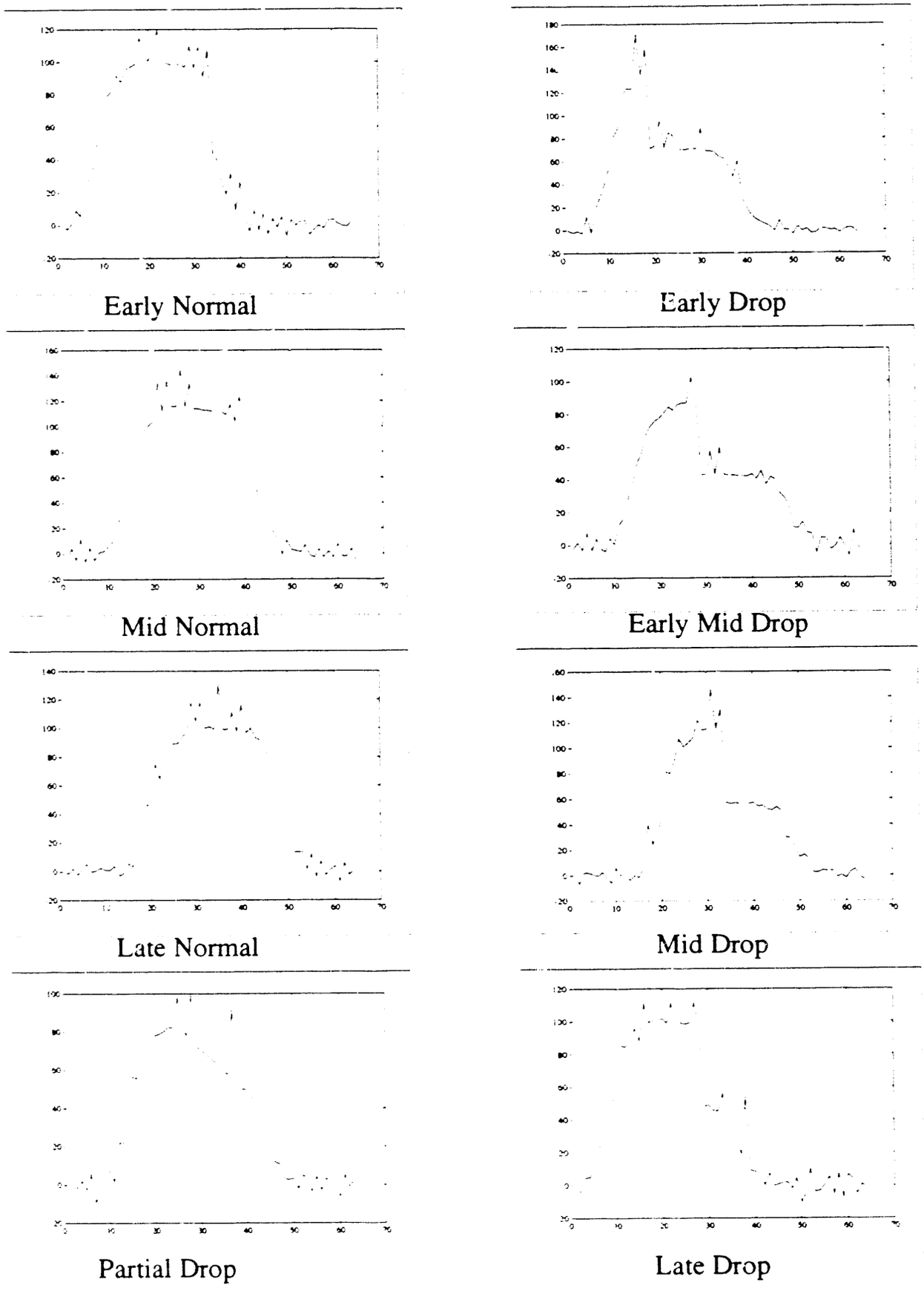


Figure 16. Eight Pulse Categories Used for Real Chip Experiment at Intel.

The physical size of the ETANN chip may present problems in an actual telemetry system. The IC is a 208-pin grid array, 2.25 inches square. The main problem is that in actual flight hardware solder joints are generally visually inspected. The ETANN device has internal pins which will be hidden from view, so a compromise will be required. One advantage is that not all 208 pins are required to compute the feedforward neural network algorithm. For example, 64 pins on the device are reserved for training and weight programming and not used once the chip has been trained.

Another concern with the ETANN chip is the retention time of the stored analog weights. Intel specifies a 10-year life before the analog weights drift significantly. This does not appear to present a problem in telemetry since boards are used soon after construction.

Winner Select

The final block of the transient pulse measurement system is used to select a winning classification. This block requires an analog comparator for each neural network output. The logic will select as the classification that output category with maximum value. Finally, the comparator outputs will be encoded into a short binary number (well under the 30-bit constraint) for transmission. Each of these numbers will correspond to a different pulse category.

VII. Conclusion

A neural network based system has been proposed to perform on-board pattern recognition to assess transient pulse shape characteristics. The system has been designed to provide more information than is currently available about transient pulses generated during flight testing. Two network formulations have been investigated. One performs an on-board analysis function to classify the raw pulse data into a number of output categories, indicating component performance. The second performs a data compression to efficiently reduce the bandwidth required for transmission.

Simulation studies have demonstrated the feasibility of both formulations. In the failure detection case, perfect classification of the training data was achieved. When tested with new data, performance was dependent on the size of the training set used. Errors that were made appeared to be reasonable, given the data used for training. For the data compression case, only a few hidden nodes were necessary in order to achieve high accuracy in approximating the original data. This means that an acceptable compromise is feasible between transmitted message length and reconstruction accuracy.

A more thorough understanding of the range of pulse patterns that can be expected in actual tests is required to fine tune the network weights. However, the initial analysis clearly demonstrates that the architectures proposed exhibit impressive performance on input-output mapping problems using reasonably realistic data. The versatility of the neural network approach does not depend critically on the specific type of data available. This suggests that the approach can be applied, with marginal development effort, to other data analysis problems arising in flight testing.

A hardware implementation effort has proposed a three-stage transient pulse classifier system. The system is now in the development phase. Some design problems have been identified and are being investigated. At present, there appear to be no major "show stoppers". In the near future, a prototype printed wiring board will be completed which

will be capable of making a transient pulse classification in real time for demonstration purposes.

References

[1] Rumelhart, D.E. and J.L. McClelland (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Vols. 1 and 2*, Cambridge: Bradford Books/MIT Press.

[2] Lippmann, R.P. (1987), An introduction to computing with neural nets, *IEEE ASSP Magazine*, v.4, pp.4-22.

[3] Simpson, P.K. (1990), *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*, Pergamon Press.

UNLIMITED RELEASE

INITIAL DISTRIBUTION

1000 V. Narayanamurti
1400 E. H. Barsis
1410 P. J. Eicker
2334 S. M. Kohler
Attn: 2334 S. L. Stanton
2533 W. C. Riggan
2560 J. T. Cutchen
2561 D. K. Morgan
5000 R. L. Hagengruber
Attn: 5100 J. L. Wirth
5300 E. E. Ives
Attn: 5360 D. J. Bohrer
5310 M. E. John
5311 R. J. Gallagher
5312 L. D. Brandt
5312 R. M. Wheeler (20)
5313 J. E. Marion
5320 R. L. Rinne
5350 J. B. Wright
5355 R. G. Miller
5356 C. A. Pura
5370 J. W. Hickman
Attn: 5376 P. G. Heppner
5376 R. L. Bierbaum
8000 J. C. Crawford
8200 R. J. Detry
Attn: 8210 W. D. Wilson
8240 C. W. Robinson
8245 R. J. Kee
8300 P. L. Mattern
Attn: 8300A T. F. Bramlette
8300A J. Vitko
8300B J. S. Binkley
8316 R. E. Stoltz
8360 W. J. McLean
Attn: 8361 D. R. Hardesty
8362 R. W. Carling
8362 D. L. Siebers
8400 R. C. Wayne
Attn: 8440 H. Hanser
8480 L. A. West
8430 L. A. Hiles
8431 W. G. Wilson, Actg
Attn: 8431 F. R. Hansen
8432 W. G. Wilson
Attn: 8432 L. M. Napolitano
8432 G. F. Schils
8433 M. H. Rogers
8450 J. F. Barham
8451 T. R. Harrison

8451-1 V. Barr
8453 C. F. Acken
8453 D. F. Clark
8453 P. A. Larson
8453 D. A. Sheaffer (2)
8453 J. R. Williams
8454 A. L. Hull
8455 B. Stiefeld
8484 L. N. Tallerico
9100 R. G. Clem
9130 A. C. Watts, Actg
9132 A. C. Watts
Attn: 9132 T. M. Criel
9132 R. S. Edmunds
9132 J. C. Gilkey
9133 L. D. Hostetler
Attn: 9133 M. M. Moya
9133 R. J. Fogler
9210 H. M. Dumas
9220 G. H. Mauth

8535 Publications for OSTI (10)
8535 Publications/Technical Library Processes, 3141
3141 Technical Library Processes Division (3)
8524-2 Central Technical Files (3)

END

DATE FILMED

01/09/91