A Neural-Net-Based Device for Monitoring Amtrak Railroad Track System

Ahmed Rubaai, *Member, IEEE*

***Abstract*—This paper demonstrates the feasibility of employing artificially intelligent automation for the task of monitoring the Amtrak railroad track system in a real-time transportation environment. The neural-net-based device (automation) processes several quantities that portray the localized existence of the Amtrak system. These quantities may be one or more of the following: location of the switch on the railroad track; time of observation; and the direction of travel. Given these quantities, it is desired that the state of the system, (which can only belong to one of several distinct categories) be predicted as outputs of the automation. Possible outputs are conditions classified as NORMAL, NOT NORMAL, REVERSE, and NOT REVERSE. Implicit in the choice of a configuration of inputs and outputs is the hypothesis of the existence of a multivariable mapping connecting these inputs and outputs—a mapping that hopefully coincides with the real-world dynamics of the railroad track. The neural-net-based device is tested on a specific, already-in-place transportation control system—the Centralized Electrification & Traffic Control (CETC) system operated by Amtrak on the northeast corridor. The CETC is chosen because of the clear value which such an operational safety and security monitor would bring to it. The test results obtained in this paper confirm that artificial neural networks can be effectively used to solve the pattern recognition problem posed by the Amtrak system. To the best of the author's knowledge, no similar work is outstanding, planned, or anticipated at this time.**

***Index Terms*—Amtrak system, detection, monitor, neural net, railroad track.**

I. Introduction

THE basic technologies to be exploited in this paper are those of neural networks and modern control concepts. Since neural nets have the ability to monitor highly complex patterns (many thousands of points of data) and "learn" them intimately without human programming, they can easily note (and alarm) when even subtle changes occur in these patterns. Theoretically, then, neural nets should be able to monitor all railroad operations over time and identify when anything abnormal (an exception) occurs—no matter what the cause, planned or unplanned, internally or externally caused. Even the sources of signals and commands must come from the "usual" places to avoid noting and exception. Even minor, unintentional operator errors will be noted which, if properly used, could also avoid almost any unpleasant operational situation, not just security threats.

With the neural net, an "overhead" or "manager" computer will receive all of the neural nets noted exceptions and compare them to the operating plan for that day. Exceptions that conform to the expected operations will just be logged, but no alarm sounded. However, the occurrence of any nonexpected exceptions will be cause for requesting human attention (alarm). Neural nets are such that they can "learn" extremely complex patterns and note minor variations automatically. With its rapid reconfigurability, total transportability, and "bulletproof" security, the neural net-based system will easily become the network security and control monitor [1]. It is now widely accepted that neural networks with three or more hidden layers are universal approximators [2]. They can be trained to learn and then emulate multidimensional mappings of very high complexity to a desired degree of accuracy. Our experience [3]–[5] has demonstrated that these networks suffice even for highly nonlinear mappings. However, to successfully design a neural network, one must have a good understanding of the nature of the function being approximated. Specifically, the following questions must be addressed.

1) What are the domain and the range space of the function?
2) Are there any constraints or intervals to which the input space of the function is restricted?
3) What are the variables that make up the domain space? Do they form a complete set? Are any of these redundant?
4) What is the degree of fidelity associated with training data (how representative is the data to a real-life situation)?

The answer to 1) dictates the number of neurons in the input and the output layers of the network. Addressing 2) and 4) simplifies the process of collecting training data, and the solution to 3) can only be obtained after a thorough understanding of the problem to be solved.

II. Problem Formulation

The challenge in this project is twofold: 1) to identify, characterize, and cost effectively acquire the massive amount of network command and indicator data in real-time from the system and transform it into state vector formats suitable for monitoring railroad track operation and 2) to develop the right combination of neural-net technologies to accommodate and handle the resulting data patterns. Since, in the the Centralized Electrification & Traffic Control (CETC) system operated by Amtrak there are more than 100 Remote Terminal Units (RTUs), each with up to several thousand signals involved, just handling the massive amount of data will itself be a challenge. In addition,

Paper MSDAD-A 02–45, presented at the 2002 Industry Applications Society Annual Meeting, Pittsburgh, PA, October 13–18, and approved for publication in the IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS by the Industrial Automation and Control Committee of the IEEE Industry Applications Society. Manuscript submitted for review October 15, 2002 and released for publication January 7, 2003. This work was supported by the U.S. Department of Transportation under Grant DOT97-1.

The author is with the Electrical and Computer Engineering Department, Howard University, Washington, DC 20059 USA (e-mail: rubaai@scs.howard.edu).

Digital Object Identifier 10.1109/TIA.2003.809443

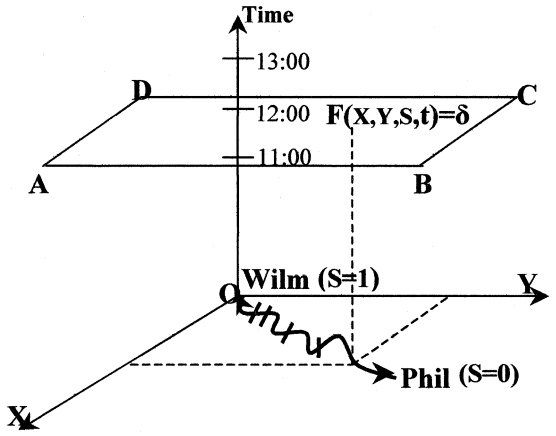

Fig. 1. Coordinate frame and pictorial representation of the frame.

information in these data ranges from almost never changing to relatively rapidly changing, which will require creative handling and transformation in order to present it in proper format to the monitor.

Based on the data provided by Amtrak, the author makes the following statements about the function that the neural-net-based device is supposed to emulate.

- The output (range space) of the function is binary. Only the following four possible outcomes (1, 2, 3, 4) are feasible: 1 signifies a NOT NORMAL condition; 2 signifies a NOT REVERSE condition; 3 signifies a NORMAL condition; and 4 signifies a REVERSE condition.
- The domain space consists of the following fundamental quantities:
  1) position on the track;
  2) direction of travel;
  3) time of the day.

Fig. 1 presents a pictorial representation of the problem. It shows a coordinate system whose XY plane contains the railroad track to be monitored for abnormalities. The origin of the coordinate frame is chosen to be the starting point of the track (hypothetically chosen to be Wilmington (*Wilm*) in the figure). Assuming that the destination is Philadelphia (*Phil*), one can define an independent variable that quantifies the sense of direction of the journey. That is, $S = 0$ for a journey starting at *Wilm* and terminating at *Phil*, while a journey in the opposite direction beginning at *Phil* and ending at *Wilm* would have a value of $S = 1$. Points along the track at which training data for the network are available are shown in the figure as slashes intersecting the track. The geographic location of these slashes are specified by their x and y coordinate pairs with reference to the chosen coordinate frame (the choice of a Cartesian coordinate frame, here, is by no means absolute; it has been chosen for illustrative purposes only). The third axis denotes the time of day (for example, the plane ABC represents noon on a specific day). The point F ($X, Y, S = 0, t =$ noon) on the ABC plane is a potential input vector to be presented to the network if data is available at noon, at location (x, y) for a journey from *Wilm* to *Phil*. The first attempt at classification will focus on a subset of the training data. This subset is defined as the information extracted from the audit trail delog report (Amtrak real data) that corresponds to switch

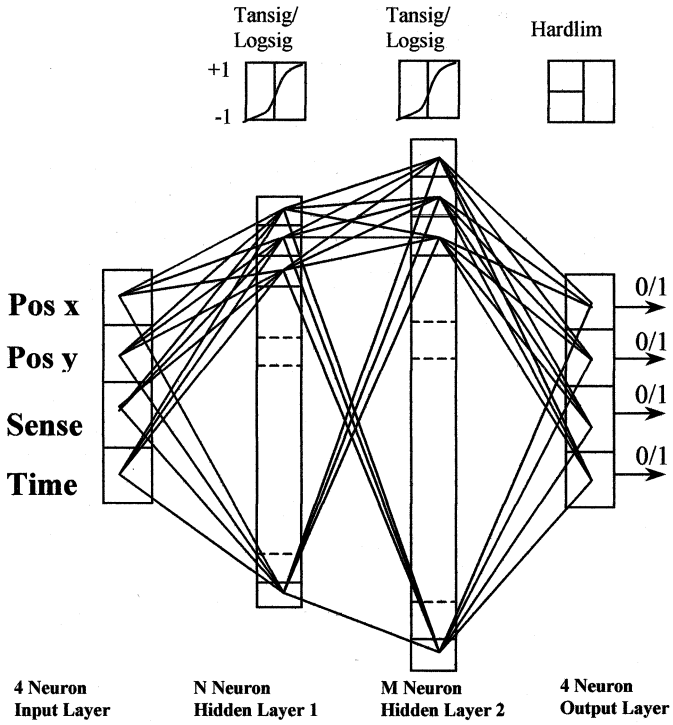

Fig. 2. Mapping the Amtrak problem to the multilayer network.

numbers 23 or 32, and for which the switch names are either locked switch (LW) or switch logic (WL). At this juncture, it is the understanding of the author that the elements of this subset are uniquely mapped to exactly one of four possible output categories (REVERSE, NORMAL, NOT NORMAL, and NOT REVERSE). The neural net will be trained on all elements of the subset over several epochs (each epoch corresponds to one pass of all elements in the subset to the neural-net-based classifier).

III. Intelligent Automaton for Monitoring the Amtrak Railroad Track Operations

Two distinct categories of neural-network-based devices have suggested themselves as possible candidates for implementing the proposed automation. However, no matter which category of neural network is used, the choice of the number and type of inputs and outputs is the single most important factor that determines the success or failure of the automation in its attempt at classification.

A. Intelligent-Automation-Based Multilayer Feedforward Network

One possible approach is to use a multilayered feedforward neural network that trained using the error backpropagation algorithm [6]. Fig. 2 shows the network architecture. The network chosen had four inputs (to match the number of components in the input vector), three hidden layers, and four neurons in the last hidden layer (to match the four output categories). There will be four neurons in the input layer (one input each for x, y, s, and t). A total of 20 and 30 neurons in the intermediate hidden layers were used. Either tansig (hyperbolic tangent sigmoid transfer function, with output restricted to the $(-1, 1)$ range) or logsig (logistic sigmoid transfer function, with output

restricted to the (0, 1) range) neurons will be used. Prior experience [2]–[4] has shown us that these suffice even for highly nonlinear mappings. There will be exactly four hardlim neurons (a neuron whose activation function is hard-limited to return 1 when presented with a positive argument and 0 otherwise) in the output layer. The nature of the hardlim activation function will guarantee that the output is bipolar (either 0 or 1). Note that the output vector [1 0 0 0] represents a NOT NORMAL state of the switch, while [0 1 0 0], [0 0 1 0], and [0 0 0 1] represent the NOT REVERSE, the NORMAL, and the REVERSE of the switch, respectively. According to information provided by Amtrak, a REVERSE condition results when the switch connects two distinct rail tracks where the angle of the connecting track is 15°, while a NOT REVERSE means the travel of the switch from REVERSE position to NORMAL position. This transition takes only 5 s for the relay along the track to drop. Furthermore, NOT NORMAL means the travel of the switch from NORMAL position to REVERSE position. This transition also takes only 5 s for the relay along the track to drop. The training data from Amtrak consist of a series of tabular entries as shown by

Inputs				**Output**
X Pos. (miles)	Y Pos. (miles)	Sense $(0/1)$	Time (of day)	$(0/1)$
0	0	0	0.00	1
5	0	0	1:00	1
5	5	0	1:15	1
.				.
.				
.				.
	.			.
	.		.	.
	.			
500	500	1	23:55	0.

Successful completion of the training will depend upon on how close the training data come to representing a real-life situation. The three-layer backpropagation network of Fig. 2 is superior to more conventional single-layer perceptron-type neuro-classifiers that have been traditionally used for similar applications. This is primarily because, unlike perceptrons, the network of Fig. 2 will not fail to classify input vectors that form a nonlinearly separable set. However, this network has one disadvantage caused by its binary output. A binary output necessarily implies that for each input the error is either 0 or 1. Consequently, there is more than a reasonable likelihood that, as training progresses, the proportional change in weights will result in them oscillating back and forth between two points, resulting in a neutrally stable trajectory in its parameter space.

The training data are a subset of data extracted randomly from the audit trail delog report that corresponds to switch numbers 23 or 32, and for which the switch names are either LW or WL. For this study, the training data extracted are restricted to the file named "temp01" provided by Amtrak. The four possible output categories in this file for the aforementioned constraints on switch numbers (23 or 32) and switch names (LW or WL) turns out to be REVERSE, NORMAL, NOT NORMAL, and NOT REVERSE. 257 such input training samples were obtained from temp01. The 257 input–output training vectors were presented to the network over several training epochs. After training, and the subsequent simulation, the results have to be unscaled so that the first component can be reinterpreted meaningfully (as seconds from the reference date). For example, suppose it can be concluded from heuristics that at switch location LW, and at time 09:30 hours, an output classification of NOT NORMAL is highly likely. Further suppose that owing to an unfortunate happenstance, the location of the switch was not chosen as a possible input type. When this happens, the network is subjected to training data that are incompatible with the mapping between the true inputs and outputs. This results in inadequate training performance and post-training generalization. The weights and biases of the network were initialized using the Nguyen–Widrow algorithm [7]. The weights and biases were then updated using the Lavenburg–Marquardt rule [8]. During training, this neural network exhibited very slow progress. This was the reason why this type of neural network was not considered further for this application.

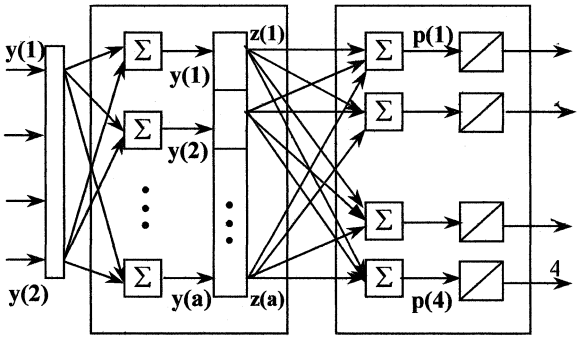

Fig. 3. Mapping the Amtrak problem to the LVQ network.

B. Intelligent Automation-Based Learning Vector Quantization (LVQ) Network

The author proposes an LVQ neural network for the task of monitoring intermodal network systems in a real-time transportation environment. Fig. 3 shows the architecture of an LVQ network. It consists of two layers. The first layer from the left is called the competitive layer, while the second layer is called the linear layer. The input $\mathbf{x} \in \Re^4$ is a typical input training vector, chosen randomly from the aforementioned subset. An LVQ network is usually presented with a set of training vectors and, for each input vector, a scalar specifying to which of several possible output categories the input vector belongs. The categories that the competitive layer of the LVQ network classifies are dependent on the distance between the input vectors. If two input vectors are very similar, the competitive layer will probably put them into the same class. A word of explanation is needed here. An independent software program has been written that scanned several files and extracted data from the audit trail delog report that match the criteria for membership in the subset. The smallest time corresponding to all data in the subset was chosen as the reference epoch for time. The times corresponding to all other input vectors are positive offsets from this reference epoch. This positive offset is the first component of all input vectors. Assuming that there are "a" neurons in the

competitive layer, $\mathbf{W1} \in \Re^{a\times 4}$ is the weight matrix for the synapses connecting the input layer and the competitive layer. The variables $\mathbf{y}$ and $\mathbf{z} \in \Re^a$ are, respectively, the input to, and the output of the competitive layer. The competitive layer learns to classify input vectors into groups, each of which is associated with one or more competitive layer neurons. The (second) linear layer then transforms each competitive layer group into a distinct user-specified target category. As soon as the input vector $\mathbf{x}$ is presented to the network, a competition takes place. This competition is among all of the "a" row vectors ($\in\Re^{1\times 4}$) of $W1$, and it is won by the one row vector that is "closest" to the input vector $\mathbf{x}$ in a Euclidean sense. If the ith row of $W1$ happens to be closest to $\mathbf{x}$, then the ith neuron of the competitive layer is said to have won the competition.

The nature of the competitive layer transfer function, and the rule according to which its input is calculated, is carefully chosen in such a way that the output vector $\mathbf{z}$ consists of all 0's except in the ith element, which has a 1. To see how this is done, the following notation is introduced. Let $\mathbf{W1}_i$ denote the ith row vector of $\mathbf{W1}$ and let $||\mathbf{x}||$ denote the Euclidean norm of $\mathbf{x}$ (i.e., $||\mathbf{x}|| = (\mathbf{x}(1)^2 + \mathbf{x}(2)^2 + \cdots + \mathbf{x}(a)^2)^{0.5}$). Furthermore, the largest element of $\mathbf{x}$ located at index i is denoted by $[\mathrm{Max}(\mathbf{x})]_i$, then,

$$\mathbf{y}_i = ||\mathbf{W1}_i - \mathbf{x}|| \tag{1}$$

$$\mathbf{z}_i = [\mathrm{Max}(\mathbf{y})]_i\, \delta_{ij} \tag{2}$$

where δ_{ij} denotes the familiar Kronecker tensor product.

As (1) and (2) show, the output of the competitive layer $\mathbf{z}$ is a 0–1 binary output vector. Its lone nonzero (and unity) component signifies the row number of $W1$ that is closest to input $\mathbf{x}$. Prior to training, the LVQ network designer carefully (and often heuristically) chooses the number of competitive layer neurons "a." At the same time he/she must also partition the competitive layer into segments—each segment associated with a specific output category. In the present problem, four distinct output categories exist. The "a" competitive layer neurons may then be nonuniquely partitioned into four segments (each of which may contain a varying number of neurons). This partitioning is reflected in the choice of the weight matrix $W2$. The elements of $W2$ are all either 0's or 1's, moreover, each column vector of $W2$ has a 1 in its jth component implying that the ith competitive layer neuron has been assigned to output category j. For example, when $a = 8$, a possible choice of $W2$ is

$$W2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Here, the first three competitive layer neurons are assigned to the first category (REVERSE), only the fourth competitive layer neuron is assigned to the second category (NORMAL), the fifth and sixth neurons are assigned to the third category (NOT NORMAL), while the seventh and eighth neurons map to the last category (NOT REVERSE). Thus, the linear layer weights in $W2$ are fixed at design and never vary during training. Although the choice of $W2$ is nonunique, its initialization is influenced by a desire to preserve the statistical distribution of how the combined set of input training vectors map to each classification category. For example, if 17% of the input training vectors belong to category 2 (NORMAL), $W2$ is initialized in such a way that approximately 17% of the neurons in the competitive layer are assigned to finding vectors belonging to this category. In addition, it is assumed that there exists a function that maps these fundamental quantities of either: 1 (NOT NORMAL), 2 (NOT REVERSE), 3 (NORMAL) or 4 (REVERSE). According to information provided by Amtrak, a reverse condition results when the switch connects two distinct rail tracks where the angle of the connecting track is 15°. It is the job of the neural-net-based system to learn this (rather complicated) mapping.

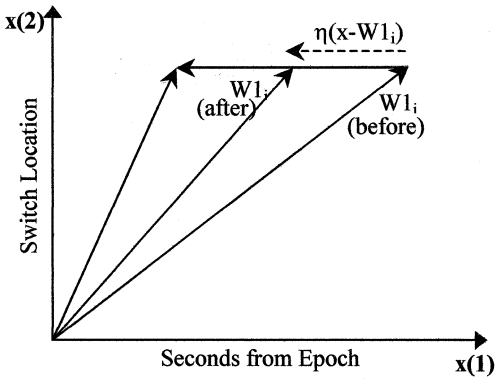

Fig. 4. Weights updates for winning neuron i.

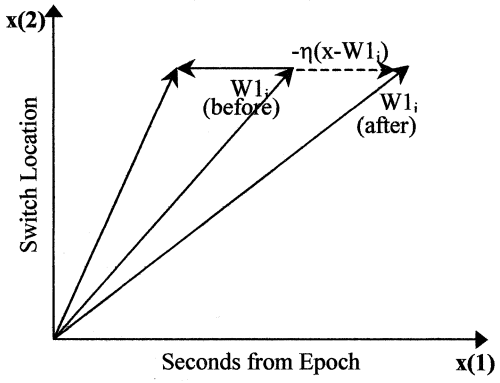

Fig. 5. Weights updates for losing neuron i.

The underlying philosophy behind the training of an LVQ network is to "reward" those competitive layer neurons that win the competition and to punish those that do not. If a neuron wins the competition, so do all other neurons that belong to its partitioned segment. From (1), it is clear that the synaptic weights that connect the input layer to the winning neuron are closest to the input vector. This neuron is, however, rewarded by moving these weights closer to the input vector. For example, suppose the ith neuron is the winner, then, the change to the weights in the ith row of $W1$ is calculated as

$$\Delta\mathbf{W1}(i, j) = \eta\mathbf{z}(i)(\mathbf{x}(j) - \mathbf{W1}(i, j)). \tag{3}$$

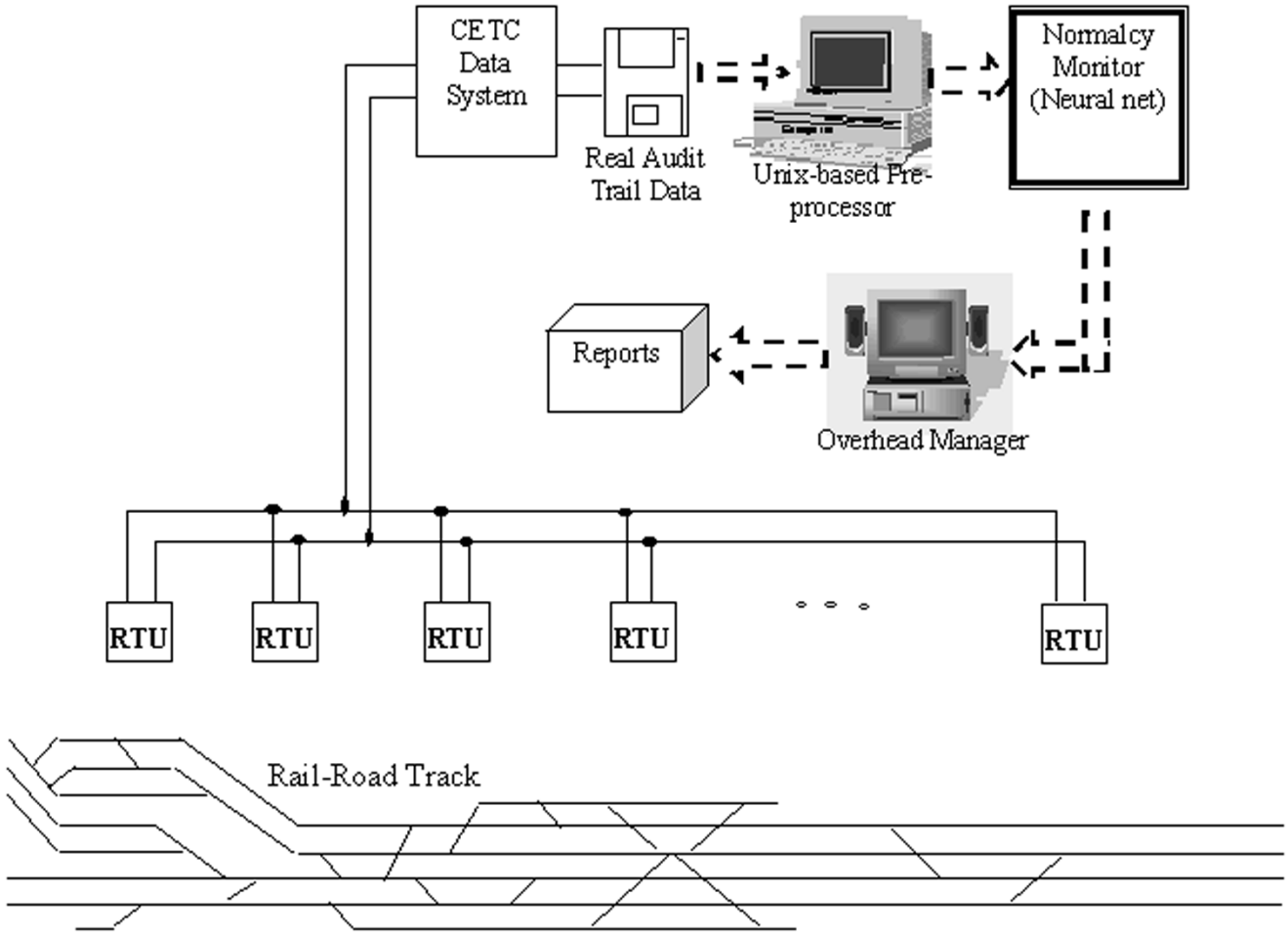

Fig. 6. Functional block diagram of the test setup.

For the "losing" neurons, the synaptic weights are moved away (weakened) from the input vector by applying a similar weight change rule (but with a sign change) as follows:

$$\Delta \mathbf{W1}(i, j) = -\eta \mathbf{z}(i)(\mathbf{x}(j) - \mathbf{W1}(i, j)). \tag{4}$$

Equations (3) and (4) are valid for all $1 \leq j \leq 2$. The symbol η here denotes the learning rate, and it determines the magnitude of the push given to the weight vectors (toward or away form the input vector). Figs. 4 and 5 illustrate the weight update for winning and losing neuron i, respectively. Equations (3) and (4) are collectively referred to as the Kohonen [9] weight change rule for training LVQ networks. After several epochs of training, the end result is that the competitive layer neurons move toward input vectors, which belong to their category, and away from input vectors that belong to other categories. Training is concluded once the network correctly classifies all input vectors from the subset.

IV. Test Setup

Amtrak did provide the author with data corresponding to several routes. One particular route (Philadelphia–Wilmington) was chosen among several available because it contained the most anomalies in terms of switch statuses. Choosing one particular route also introduces a measure of simplicity to the problem without resulting in too much loss of generality in the method of approach taken to solve this problem. Because this attempt is more of a "proof-of-concept" endeavor, the decision to restrict the data set to a single route was taken. Therefore, to assure that the date will not be skewed, real CETC data recorded as part of what Amtrak calls the "Audit Trail" are employed. The audit trail delog is created by the Data Tracking System based on information fed to it by a set of RTUs along the rail road track. A functional block diagram of the test setup is shown in Fig. 6. The data provided by Amtrak are time tagged and stored on electronic media just as it is received in the CETC. These data are then input into the test system as shown in Fig. 6. In the test setup, Audit Trail is loaded into the Data Formatting computer's memory. When the test is running, these data will be reformatted, just as it would be in the final system, for continuous presentation to the Normalcy Monitor (a neural-net-based system). After several complete presentations, the Normalcy Monitor will "learn" the pattern and establish an internal representation of it. It is from this representation that any new variations will be perceived. With each subsequent presentation of the data, the normal representation is compared to the mathematical representation of the data being received at that time. If and when variations from the original (normal) representations are discovered an alarm will be output reporting the variation and identifying the data that caused it.

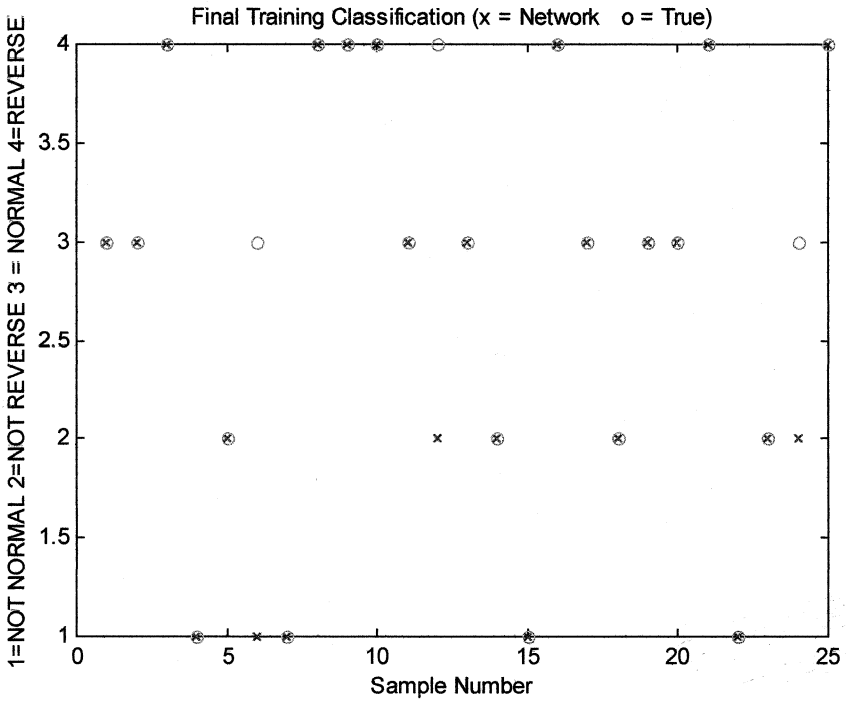

Fig. 7. Network classification at the conclusion of training.

It is planned that the Overhead Manager computer will be forewarned about some of the variations by a high-level instruction representing a change in the railroad's daily operating plan from which a human could easily predict some of the variations to be received. Therefore, it is the task of the Overhead Manager computer to sort out which variations were expected and which are to be reported to humans for further investigation. This aspect has not been dealt with in this paper and will form the basis of future research.

V. Test Results

Out of the 257 potential training vectors, ten were randomly removed from the training sample. These ten were designated to test the post-training generalization capability of the network. Out of the 247 remaining input vectors, a fixed number "N" were randomly selected with which to train the LVQ network. The number of the competitive layer neurons was selected based on the value of a user-specified parameter called "Neuron-To-Sample-Ratio." Typically, a value of 2 was used for this parameter. This means that if "N" input vectors are used, "$2N$" competitive layer neurons would be employed in an attempt to perform the classification. A better understanding of the training process will be possible if the software package is run and the graphical training progress report is monitored. Utilizing a "Neuron-To-Sample" ratio of 2, Figs. 7–10 display the classification of trained inputs for $N = 25$. Fig. 7 exhibits the performance after prolonged training. It displays both input vectors "+" and the competitive neurons (weight vectors) "○" at the conclusion of training. It is clear from this figure that the competitive layer neurons (shown as "○" symbols) tend to disintegrate into four distinct groups. The distribution of the training samples is shown as "+" symbols. Each group of neurons clusters around training data vectors that belong to its target class. Each classification category is color coded. The competitive layer neurons assigned to a given category are shown in the same color as the category. As training progresses, it is observed that neurons assigned to a given category are attracted by (and tend to surround) input training vectors that belong to the same category. Training was terminated after 100 000 epochs of training. Each epoch corresponds to one pass of the 257 training samples through the network. Of 25 samples, 22 were classified correctly and three were classified incorrectly. 88% were classified correctly as displayed in the pie chart of Fig. 8. In the pie chart, the bigger slice that comes out (exploded) corresponds to the correct classification

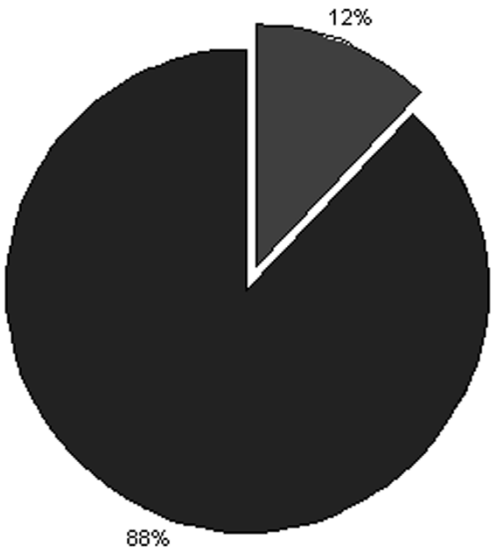

Fig. 8. Percentage classification of trained inputs at the conclusion of training.

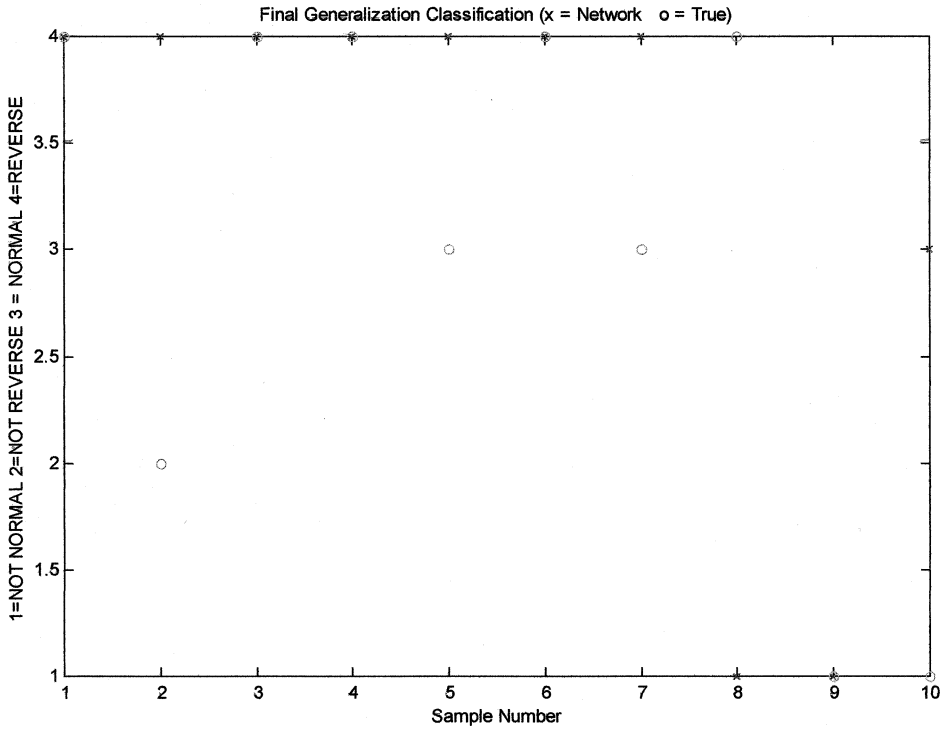

Fig. 9. Network classification at generalization.

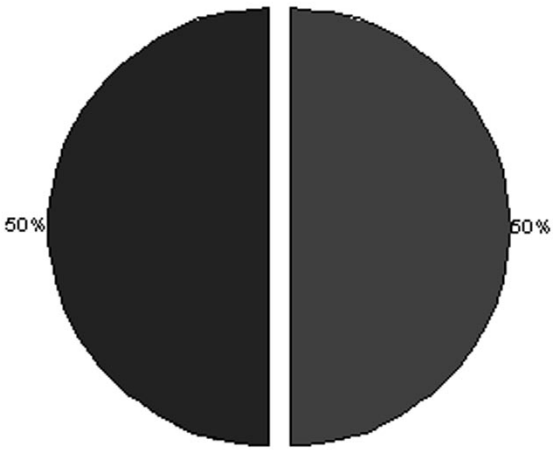

Fig. 10. Percentage classification at generalization.

percentage. The color of the slice is chosen randomly and has nothing to do with the classification. In all cases, after training, the success rate in classification was in the 85%–100% range.

To test the ability of the network-based intelligent system at generalization (correct classification of samples with which the network was *not* trained), the ten samples previously removed from the training set were now presented to the network. Figs. 9 and 10 illustrate network classification at generalization. Of the ten untrained inputs, five were classified correctly and five were classified incorrectly. 50% were classified correctly. This resulted in promising performance with a success rate of 50%–70%. Furthermore, it was observed that the success rate of both types of networks dropped from 90%–100% to 50%–70% when used with a greater number of training samples ($N \approx 100$). However, the results obtained so far represent an important first step in this work. They confirm our belief that neural networks (more specifically LVQ-type neural networks) are best suited to solving the pattern recognition problem posed by the dynamics of the Amtrak railroad track.

VI. Conclusions

This paper has proposed a design for a neural-network-based device that can be used to detect and classify generic railroad operating conditions as normal and abnormal. The intended neural net is trained online to capture the nonlinear mapping that transforms a specific location, time of the day, and direction of travel into a quantitative statement of whether or not an abnormal operating condition is possible at these inputs. The proposed LVQ neural network is a special case of self-organized competitive networks that trained in a supervised manner. It is versatile and very robust. Test results have shown that it has no trouble classifying inputs even when they is not linearly separable. It generates piecewise-linear decision surfaces that are close to the Bayesian optimum.

As part of future research of this work, the author will look at means and ways of improving the algorithms suggested in previous sections. One approach under consideration is to modify the learning rate adaptively during training so that the learning processing is accelerated.

Acknowledgment

The author would like to express his gratitude to the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] M. G. Simoes, C. M. Furukawa, A. T. Mafra, and J. C. Adamowski, "A novel competitive learning neural network based acoustic transmission system for oil-well monitoring," *IEEE Trans. Ind. Applicat.*, vol. 36, pp. 484–491, Mar./Apr. 2000.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multi layer neural networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[3] A. Rubaai, R. Kotaru, and D. M. Kankam, "Online training of parallel neural network estimators for control of induction motors," *IEEE Trans. Ind. Applicat.*, vol. 37, pp. 1512–1521, Sept./Oct. 2001.

[4] A. Rubaai and R. Kotaru, "Adaptation learning control scheme for a high performance permanent magnet stepper motor using online random training of neural networks," *IEEE Trans. Ind. Applicat.*, vol. 37, pp. 495–502, Mar./Apr. 2001.

[5] ——, "Neural net based robust controller design for brushless DC motor drives," *IEEE Trans. Syst., Man, Cybern.*, vol. 29, pp. 460–473, Aug. 1999.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Data Processing*, D. Rumelhart and J. McClelland, Eds. Cambridge, MA: MIT Press, 1986, vol. 1, ch. 8, pp. 318–362.

[7] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Proc. Int. Joint Conf. Neural Networks*, vol. 3, July 1990, pp. 21–26.

[8] M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm for the efficient training of artificial neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 989–993, Nov. 1994.

[9] T. Kohonen, *Self-Organization and Associative Memory*. Berlin, Germany: Springer-Verlag, 1984.

Ahmed Rubaai (S'88–M'88) received the M.S.E.E. degree from Case Western Reserve University, Cleveland, OH, and the Dr.Eng. degree from Cleveland State University, Cleveland, OH, in 1983 and 1988, respectively.

In 1988, he joined Howard University, Washington, DC, as a Faculty Member. He is currently a Professor of Electrical Engineering. His research interests include high-performance motor drives, research and development of intelligent applications for manufacturing systems, and industrial applications. He is the Lead Developer of the Motion Control and Drives Laboratory at Howard University and is actively involved in many projects with industry, while engaged in teaching, research and consulting in the area of artificial intelligence. In addition to his academic career, he has initiated several intelligent-based network system industrial applications, including applications in Amtrak railroad operating conditions, and electric drive system applications.

Dr. Rubaai is a Member of the IEEE Industry Applications, IEEE Control Systems, IEEE Power Engineering, IEEE Aerospace and Electronic Systems, and IEEE Industrial Electronics Societies. He is also a Member of the American Society for Engineering Education. He is a recipient of the 1997 and 1998 Howard University Faculty Teaching Excellence Award. He was also the recipient of the ASEE Middle Atlantic Section Distinguished Educator Award in April 2001. In addition, he served as Organizer, General Chair, and Technical Committee Program Chair for the 1998 ASEE Fall Regional Conference of the Middle Atlantic Section.