Implicit Learning in Autoencoder Novelty Assessment

Benjamin B Thompson[▲], Robert J Marks II[▲], Jai J Choi[✈▲], Mohamed A El-Sharkawi[▲], Ming-Yuh Huang[✈], Carl Bunje[✈]

[▲] Computational Intelligence Applications (CIA) Laboratory, Department of Electrical Engineering, University of Washington, Seattle, WA 98195

[✈] Boeing Phantom Works, The Boeing Company, Seattle, WA 98124

Abstract – When the situation arises that only "normal" behavior is known about a system, it is desirable to develop a system based solely on that behavior which enables the user to determine when that system behavior falls outside of that range of normality. A new method is proposed for detecting such novel behavior through the use of autoassociative neural network encoders, which can be shown to implicitly learn the nature of the underlying "normal" system behavior.

I. INTRODUCTION

The ability to detect an error within a system is critical in any real-world situation. Many methods are available for monitoring a system, detecting a fault, and ascribing a specific nature to the detected fault, thus enabling correction of the problem. However, a problem arises when no prior knowledge of what sort of faults may occur is given. In such a situation, the best one can hope to do is monitor the system for "normal" behavior, and when its performance falls outside of an acceptable range, set off some sort of flag indicating that the system is no longer normal, but *novel*. That is to say, all that is known about the abnormal system status is that it is *new* (from a prior-knowledge standpoint), not specifically "good" or "bad" in any definitive sense.

This paper proposes a new method of novelty detection through the application of a neural network autoencoder. In essence, an autoencoder is a neural network (in this case, with a standard, fully-connected feed-forward structure) in which the output has been trained to be identical to the input (more detail on the specific structure and nature below). The autoencoder has a number of uses, from principle component analysis and information compression, to the recovering of missing sensor data [1]-[4]. The most striking ability of autoencoders is their remarkable ability to *implicitly* learn certain underlying characteristics of the input data. That is, it can learn certain traits inherent to the input space without any prior knowledge or specific instruction to do so. For example, an autoencoder trained with colored Gaussian noise can implicitly learn the mean and autocovariance of the noise. Likewise, an autoencoder trained on parameterized chaos can learn the parameters of the chaos. This phenomenon is explored in greater detail below with specific attention on the novelty of the usage of a computer network hub.

II. THE AUTOENCODER

As shown in Figure 1, an autoassociative neural network encoder (or simply "autoencoder") has two primary features. The first, as previously mentioned, is the "autoassociative" nature of the network. That is, the input of the neural network is identical to the target output of the neural network during the training phase. The other key factor is the presence of a so-called "bottleneck" that occurs in at least one of the hidden layers of the neural network. The

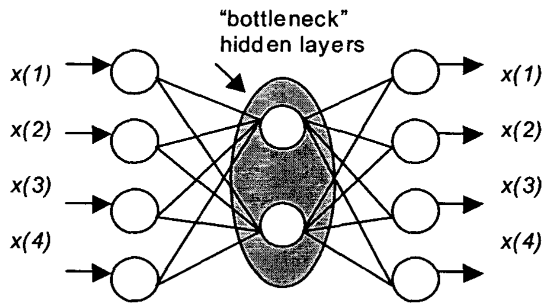

Figure 1: A basic representation of an autoencoder. Note that the inputs are identical to the outputs, and that there exists at least one hidden layer of lower dimenionality than the input/output layers.

size of the bottleneck, while very application-specific, must be of a dimension smaller than the input layer. This is the level at which the actual "encoding" takes place; specifically, the information contained in the entire input space is essentially projected onto a lower-dimensional space at the smaller layer. Note that, for this "encoding" to occur, the neural network must necessarily have a feed-forward type of structure (otherwise, the reduction in dimensionality is not guaranteed to occur). All neural networks discussed herein were trained using standard error-backpropagation methods, and all networks were fully connected [1].

As a final note on the behavior of an autoencoder, it is worthwhile to mention that an autoencoder essentially behaves as a projection of some input *x(t)* onto some set **C** in the input space. That is, if *x(t)* lies within the region for which the autoencoder is trained, it should remain relatively unchanged by the encoder (except for perhaps a reduction in detail due to a corresponding reduction in dimensionality). However, if *x(t)* is substantially different from anything previously seen by the encoder in its training phase, the output will resemble an approximation of $x(t)$ as some member of **C**.

III. NOVELTY DETECTION

Novelty detection refers not to any sort of error correction, nor specifically to error detection. Rather, it is a method by which one can establish some level of "normal" system behavior, and from that, assess a level of novelty to some sort of aberrant output of a system. Figure 2 illustrates this.

A number of methods have been proposed for novelty detection, including matched filtering, fitting an ellipse to a set of data [5]-[7], and projecting onto a linear manifold of a linear fit of the training data [3]. This paper presents a method employing the autoencoder [1],[9]. The shell **G** in Figure 2 indicates the signal space in which resides all "good" system behavior. In real-world situations, this space can be determined by a number of methods, from *a priori* knowledge of the system and (hopefully) accurate system models, to simply collecting a sufficient amount of certifiably good data and forming a hull around that data. One relatively simple method is to fill out the *convex hull* of the known data, and to assume that anything falling within the hull can safely be considered normal, and anything outside of the hull can be treated as novel. The region bordered by **C** in Figure 2 demonstrates the creation of a convex hull from data that is not necessarily convex. Obviously, the data must be sufficiently close to convex for the convexity approach to be valid; otherwise, distinctly novel data lying within the concave regions outside the actual "good" signal space could be labeled as normal.

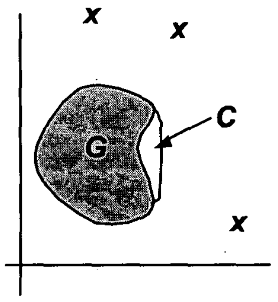

Figure 2: **G** represents the entire signal space of "normal" system behavior. **C** is the convex hull of space **G**. The x's represent novel behavior of the system.

Once a region of "normalcy" versus a region of "novelty" has been established, a method for determining whether or not a given point lies within either space must be developed. This is the thrust of this work. Figure 3 demonstrates the concepts underlying the method for detecting novelty proposed by this paper. Some input signal $x(t)$, taken from the output of the system we wish to monitor, is pre-processed before being used in novelty detection in order to put it into some form useable by an autoencoder. This process is very application-specific, as will be seen below in Section IV. An autoencoder is then trained on normal system behavior to the extent that the autoencoder implicitly learns the underlying behavior of the system. When normal data is presented to the autoencoder, it should pass through the encoder with minimal distortion; thus, the output of the summing junction $z(t)$ should be very small. Given the projection nature of the autoencoder as described above, any large departure from normal behavior in $y(t)$ should also be very distinct in $z(t)$, since $y_\eta(t)$ represents the "normal behavior" portion of the input $y(t)$. Thus, in a raw form, we have that $z(t)$ represents some measure of novelty of $x(t)$, the output from some system. As described in the figure, that raw measure of novelty can then be post-processed via a variety of methods to determine $\eta(t)$, a definitive level of the novelty of the given input to the novelty detector.

IV. EXPERIMENTAL RESULTS

In order to determine the effectiveness of an autoencoder novelty detection system, a number of different scenarios were investigated in a variety of ways. First, a simple heuristic model was developed. The "normal system behavior" was modeled as simple colored noise, and "novelties" took the form of pulses or similarly deviant artificial signals additively introduced into the noise. Following extremely promising results from that situation, some real-world data was obtained and a novelty detector was formed around it, with equally promising results.

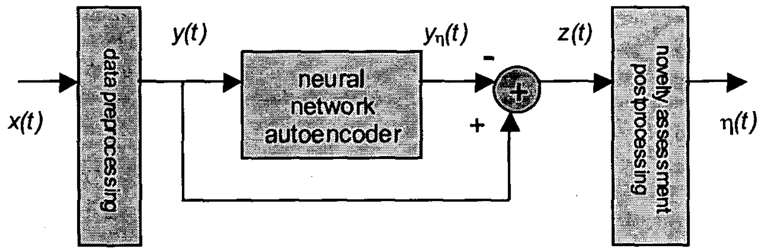

Figure 3: A model of the proposed novelty detector. $x(t)$ represents the raw input signal from a given system. $y(t)$ is that input processed in some manner ot make it useable by the encoder. $y_\eta(t)$ is the output of the autoencoder. $z(t)$ is the difference between the input and the output of the autoencoder. $\eta(t)$ is some measure of novelty determined by the post-processing methods used in the final block

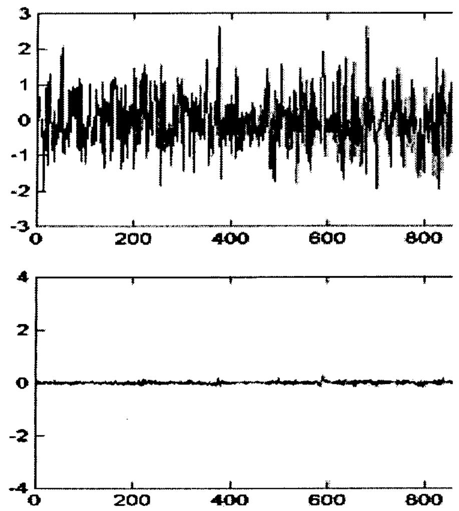
Figure 4: Example of the "normal" system behavior described in section IV.A. The top plot is an example of the colored noise; the bottom plot is that noise run through the novelty detector.

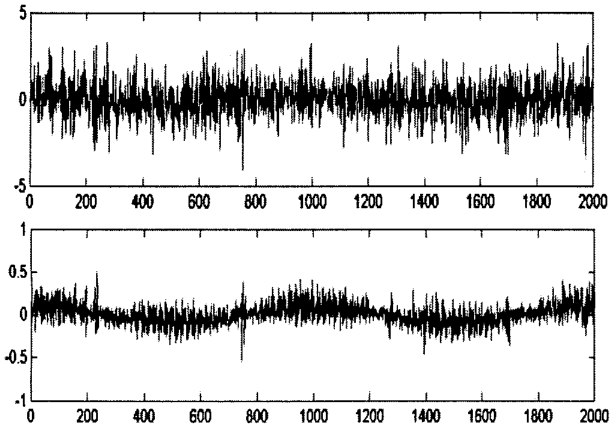
Figure 6: performance of novelty detector with additive small sinusoid. The sinusoid added had an amplitude of 0.5, well within the [-2,2] dynamic range of the "normal" system behavior. Note how prevalent the sinusoidal behavior is after the novelty detection is applied.

A. Novelties in the Presence of Artificially Generated Colored Noise

1) Generation of data: A system's "normal" mode of behavior is colored noise. The noise was obtained by generating discrete Gaussian white noise (with adjustable mean and standard deviation), and then using a standard Butterworth filter to "color" the noise in a band-pass sense. Thus, the system was governed essentially by three parameters: μ, σ, and β (mean, standard deviation, and bandwidth, respectively).

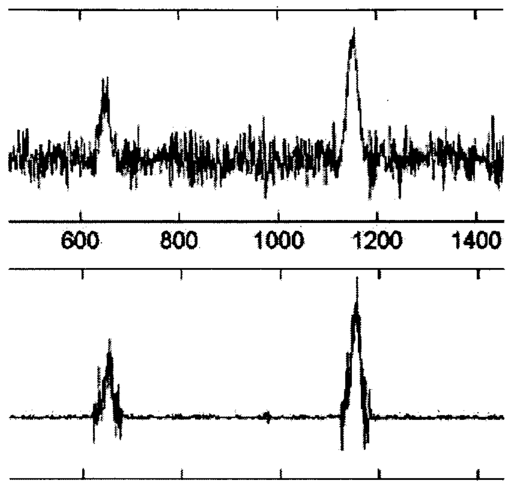
Figure 5: performance of novelty detector on normal system behavior with additive pulse novelty.

The method by which the network was trained is as follows: a neural network of 20-18-20 structure (20 inputs/outputs, and a single hidden layer) was created and trained as an autoencoder. The training data was generated as described above, and a "sliding window" of 20 data points at a time was taken from that data set to be used as a training pattern. That is to say, given a data set $\{x_i\}$ consisting of the colored noise, the i^{th} training pattern would be the sequence $\{x_{20(i-1)}, x_{20(i-1)+1}, \ldots x_{20(i)}\}$. Roughly 20,000 training patterns were generated in such a manner, and the network was trained until a sufficient level of convergence was reached.

2) Testing of the novelty detector: The initial test of the neural network was as follows: a new sequence of noise $\{g_i\}$ was created in the same manner as before (i.e., with the same three parameters). This sequence was run through the novelty detector using the same method used in training. If correctly working, the novelty detector should reduce those differenced sequences to a very low level. Figure 4 shows the results of such a test, and very clearly the novelty detector is working as expected. The original input sequence is shown for comparison. Following that, Gaussian-shaped pulses of varying sizes were additively added into another noise sequence, and the entire resulting data set was run through the detector. The results from this can be seen in Figure 5. Again, the novelty detector works remarkably well. The background noise has been dramatically reduced, while the novelty of the Gaussian pulses is brought out very prominently.

Fearing that the detector was working simply as a thresholding mechanism (that is, merely passing anything relatively large, while eliminating all small variations), a smaller signal was added to the "normal" noise sequence. Specifically, a sinusoid of amplitude well within the dynamic range of the "normal" system behavior was introduced into the "system." Very clearly, while the resultant has no

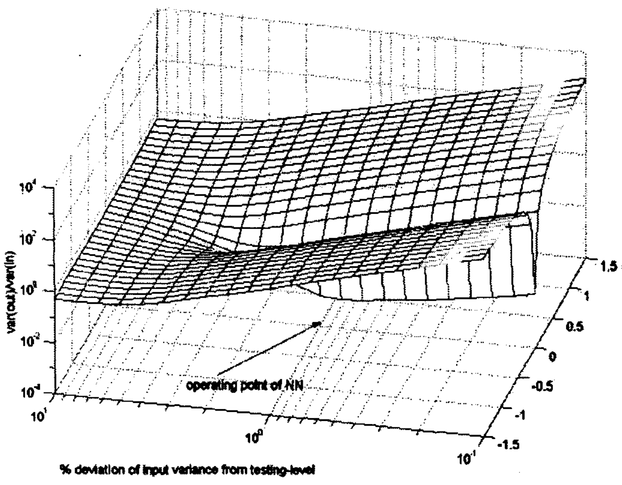

Figure 7: Sensitivity of novelty detector to changes in both mean and variance. Note minimum at operating points of both parameters.

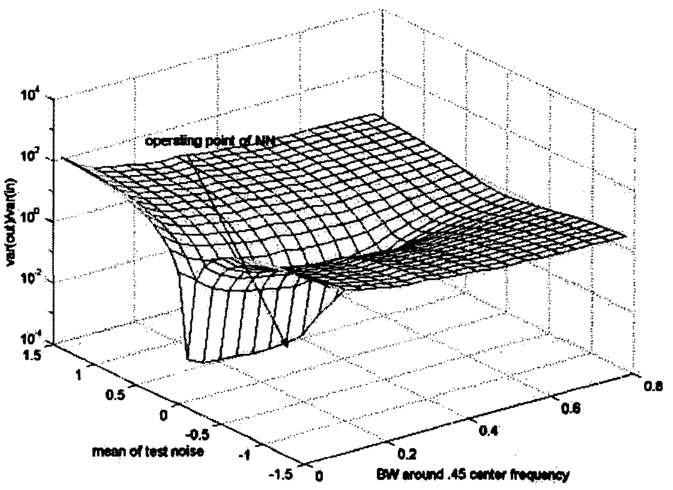

Figure 8: Sensitivity to changes in mean versus changes in bandwidth. Minimum as expected.

novelties visible to the unaided eye, the novelty detector does an excellent job of reducing the background noise and enhancing the presence of the weak sinusoid. This effect can be seen in Figure 6.

3) Sensitivity and implicit learning: At this point, an investigation into the underlying behavior of the novelty detector became warranted. Specifically, it would be useful to determine what sorts of modifications are needed on the input to contribute to novelty. Given that we have three distinct control parameters of the noise as mentioned above, it is instructive to see how the novelty detector reacts to systematic changes in those parameters.

To accomplish this, new noise sequences were generated in the same manner, with two distinct parameters adjusted simultaneously. The mean was adjusted simply by adding or subtracting a constant from the entire sequence (the training-level of that parameter was at $\mu = 0$). The variance was adjusted by scaling the sequence appropriately, while conserving the mean (initially, $\sigma = 1$). The bandwidth parameter was a bit more complicated to adjust. Initially, given a normalized bandwidth of [0,1], a center-frequency of 0.45 was set, and the band [0.3,0.6] was set as the pass-band of the colored noise. To adjust this, the center frequency was maintained at 0.45 while the width of the passband was adjusted symmetrically around that central value.

Since, for this particular novelty detector, the level of novelty is determined by the deviation about zero of the output signal, the variance of each length-20 section is a good measure of that novelty. Wanting some consistent measure of the ability of the novelty detector to remove the "normal" portion of the behavior, a good sensitivity measure is $var(\eta(t))/var(x(t))$. Thus, we are able to investigate changes on the output with respect to changes in the input. If the input to the system is very similar to the normal operation of that system, then the output of the novelty detector will be very close to zero. However, as the input deviates from normal operation, clearly that value will rise. Therefore, theory would indicate that the sensitivity should hit a local minimum around some operating point. The average result of such a calculation over a number of realizations for the same set of parameters was obtained to increase accuracy.

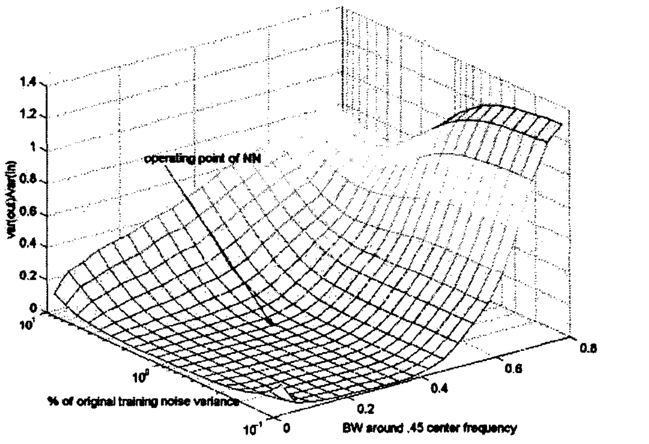

Figure 9: Sensitivity to changes in variance versus changes in bandwidth.

Using the methods described above, an investigation into the variance of two simultaneous parameter adjustments was performed. Given the expectations outlined above, we anticipate some sort of bowl-like form, with the shallowest point occurring around the operating point. Figures 7, 8, and 9 show these plots for changes in μ and σ, β and μ, and β and σ, respectively. Here, we see what we have predicted; the

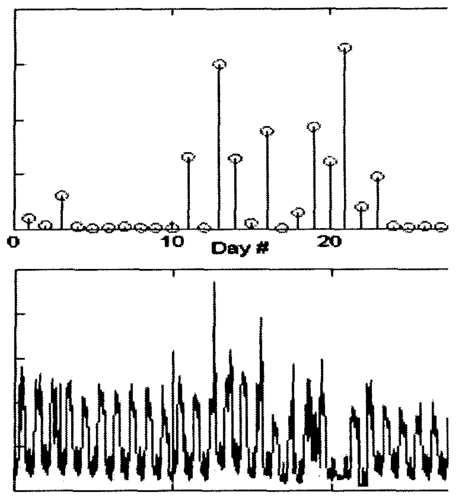

Figure 10: the novelty measure described in (1) applied to the network server utility data. The top plot represents the novelty for the corresponding "hump" (a single day) on the bottom plot.

minima of the plots all occur around the operating points of their respective parameters.

The most remarkable aspect of these sensitivity plots is that the novelty detector has, in effect, *implicitly* learned the salient features of the underlying system behavior. At no time in the training were the specific values of β, μ, or σ revealed explicitly to the process; in fact, the detector was, in a sense, completely oblivious to their existence. Rather, the autoencoder learned the parameters independently of any outside manipulation or influence.

B. Network Hub CPU Usage Data

1) Description of data: In an effort to apply the concepts of novelty detection described herein to a real-world situation and in order to better determine the robustness of these methods, some raw data was obtained. On a particular network server, a measure of the average CPU utility over a period of 15 minutes is taken every quarter hour, for a total of 96 data points per day, from Monday through Friday. This data was collected over a period of roughly 10 weeks. It is desired to determine when, over a given period of time, the performance of the machine falls out of some "normal" mode of operation. To expand the base data set, the entire data set is divided into four separate sets, such that the first set represents the first data point of each hour, the second set the second data point of each hour, etc. Then, each 24-hour period (measured between successive 12:00 am data points) is treated as a single 24-dimensional data point. Thus, we are limited in this line of investigation to determining the novelty over an entire day rather than at any given instant.

2) Convex combinations of insufficient training data: It became evident, especially in the early stages of this line of research, that the data set provided was insufficient to fully characterize the signal space [8]. Thus, a method for generating more data points was needed. One option is simply employing jitter – that is, adding random noise to existing data points [9]. This method, however, proved to be insufficient in satisfactorily filling out the space. However, under the assumption that the space is convex, we can, as described above, generate many *convex combinations* of the given data set to completely fill out the convex hull with (hopefully) valid training data.

3) Results: The bottom of Figure 10 shows that there are clearly some days (each "hump" is essentially an entire day from midnight-to-midnight) that fall into a noticeable pattern, while some distinctly fall out of that range, either above or below it. The data was then segregated into "good" and "novel" data sets simply through heuristic eyeballing of each 24-hour period. The majority of the "good" data points are used for training data, while the remainder is used for testing. A large number of convex combinations (with additive-noise smoothing) are generated to fill the convex hull of the data set. This expanded data set is then used to train an autoencoder, which is in turn used to construct the novelty detector. The autoencoder had of course 24 inputs and outputs, with 3 hidden layers in a 20-16-20 structure.

After creating the novelty detector, some measure of novelty is required. Since we have 4 data points per hour, we can easily split each day up into 4 vectors, each of which can be run through the detector. These sequences can then be recombined to form a 96-point-long sequence. Using this recombination, the following metric is used:

$$\eta(k) = \sum_{i=1}^{24} \sum_{j=1}^{4} x_{i,j,k}^2 , \qquad (1)$$

where k is the day over which the novelty is measured, i is the hour, and j is the quarter of the hour, resulting in η, the daily measure of novelty. One can further set some arbitrary threshold value, such that any novelty measured above it is flagged "novel," and anything below, "normal".

Using the metric for novelty described in (1), the entire set of network data was run through the detector. Figure 10 shows partial results, displaying the original data along with the measure of novelty for each day. Note that the novelty detector picks out days that fall both above and below the "nominal" operating levels. Compare, for example, the results for days 13 and 21. Both had an equivalently high level of novelty; but looking at the corresponding actual data for those days, 13 seems to be well above the nominal level of behavior, and 21 is clearly well below that level. This is not to say that the detector acts simply as a thresholding mechanism; observe that days 1 and 3 appear to be within the dynamic range of "normal" system behavior; however, they both register some level of novelty.

At this point, some further investigation into the sensitivity of the detector to various parameters is needed. The most obvious (and easily adjustable) parameters for this configuration are relative amplitude and convexity. For the amplitude case, random "good" vectors are chosen, and tested using the detector, after being multiplied by a range of scalar multiples. This is repeated over a large number of random

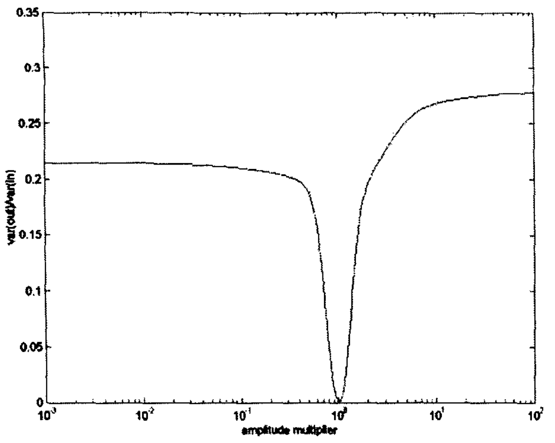

Figure 11: Sensitivity of server-load novelty detector to changes in amplitude. x-axis is the scalar multiplier of the input vector. Note the minimum at 1.

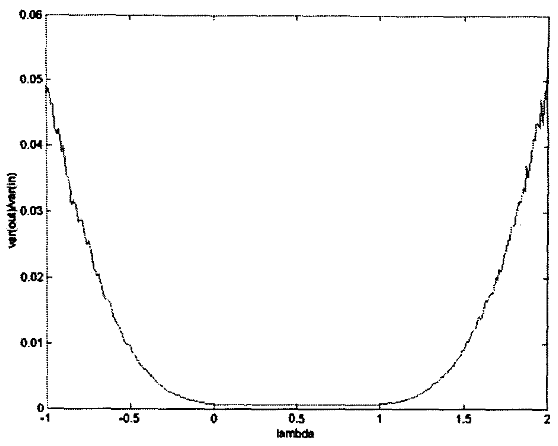

Figure 12: Sensitivity to convexity of input vector. x-axis is the lambda-value used in creating a convex combination. This result is the average over many realizations.

choices of input, and averaged. Figure 11 shows the resultant sensitivity curve. Once again we see the minimum directly at the operating point of the network (a multiplier of unity). The "sensitivity to convexity" follows in such a manner: since the network was trained on a large number of convex combinations of the input vectors, any convex combination of two input vectors should lie within the operating region of the novelty detector. Thus, any combination of two vectors $\lambda x_1+(1-\lambda)x_2$, with $\lambda \in [0,1]$, should result in a minimum on the graph. When λ deviates outside of the [0,1] range, we should see an increase in the level of novelty. Figure 12 clearly demonstrates this exact behavior. Note that this curve is generated by taking a very large number of pairs (around 500), performing the sensitivity analysis on each pair, and averaging the results together.

Again, we see, to a stunning degree, the level of implicit learning being performed by the novelty detector. At no time was a specific level of amplitude, or even the concept of convexity, introduced as an explicit parameter during the training phase of the novelty detector. And yet, again, we see that the autoencoder clearly "knows" about these parameters, in the sense that it rejects things falling outside of an acceptable range. This is a remarkable property of this method of novelty detection.

V. CONCLUSION

Clearly, a neural networks approach to novelty assessment has a great deal of potential. Using an autoencoder to learn the underlying nominal behavior of a system is an effective method by which one may construct a novelty detector, and such learning has more substance than a simple memorization or table look-up process. This is evident in the implicit manner in which the detector can learn underlying system behavior, demonstrated most clearly in the sensitivity analysis described herein.

VI. ACKNOWLEDGEMENTS

This work was supported in part by The Boeing Phantom Works at The Boeing Company.

VII. REFERENCES

[1] R.D. Reed and R.J. Marks II, *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks.* Cambridge, MA: MIT Press, 1999.

[2] S. Narayanan, R.J. Marks II, J. Vian, J. Choi, M.A. El-Sharkawi, and B. Thompson, "Set Constraint Discovery: Missing Sensor Data Restoration Using Auto-Associative Regression Machines," Proceedings of the International Joint Conference on Neural Networks May 2002, forthcoming.

[3] T. Kohonen, *Associative Memory: A System Theoretic Approach,* Springer-Verlag, Berlin, 1977

[4] H. Ko and M. Arazulla, "Background Noise Suppression for Signal Enhancement by Novelty Filtering," IEEE Transactions on Aerospace and Electronic Systems, vol. 36 no. 1, pp 102-113, January 2000.

[5] S.E. Guttormsson, "Novelty Detection of Shorted Turns in Turbo-Generated Rotors," MSEE Thesis, University of Washington, 1997

[6] S. Guttormsson, R.J. Marks II, M.A. El-Sharkawi and I. Kerszenbaum, "Elliptical novelty grouping for on-line short-turn detection of excited running rotors", IEEE Transactions on Power Engineering, Energy Conversion, IEEE Transactions on Volume: 14 1 , March 1999 , Page(s): 16 -22

[7] R.J. Streifel, R.J. Marks II, M.A. El-Sharkawi and I. Kerszenbaum, "Detection of Shorted-Turns in the Field Winding of Turbine-Generator Rotors Using Novelty Detectors: Development and Field Test", IEEE Transactions on Energy Conversion, vol.11, no.2, June 1996, pp.312-317.

[8] A. Khotanzad, R. Afkhami-Rohani, T.L. Lu, A. Abaye, M. Davis, and D. Maratukulam "ANNSTLF – A Neural-Network-Based Electric Load Forecasting System," IEEE Transactions on Neural Networks, vol. 8, no. 4, pp 835-846, Jully 1997

[9] R. Reed, R.J. Marks II and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing and training with jitter", IEEE Transactions on Neural Networks, vol. 6, no.3, May 1995, pp. 529-538.