FAULT DETECTION FOR ENGINE BLADES USING HIERARCHICAL NEURAL NETWORKS

ARTHUR TERANISHI AND STEPHEN STUBBERUD
ORINCON Corporation
9363 Towne Centre Dr.
San Diego CA, 92121
United States of America

Abstract: - Ingestion of materials into an aircraft engine can cause damage to the engine blades. Detection of this foreign object debris can alert maintenance teams of potential problems before they can endanger the flight crew or the aircraft. Using eddy-current data for each of the blades, we monitored each blade position. A hierarchical neural network was used to track changes in the position of the blades. Changes in the blades's position would be tagged and recorded to indicate the ingestion of potentially damaging debris.

Key-Words: fault detection, neural networks, classification, anomaly detection, EBF, LVQ

1 Introduction

As seen in Figure 1, a jet engine that is used in Navy fixed-wing aircraft is a complex piece of machinery. While redundancy exists, faults and failures in components of the engine can lead to a catastrophic failure of the engine. To avoid such a failure periodic maintenance is performed on these engines. However, the maintenance is usually performed on a periodic time basis. Such maintenance schedules can be too early for some engines which can increase costs and too late in other cases which is even more costly.

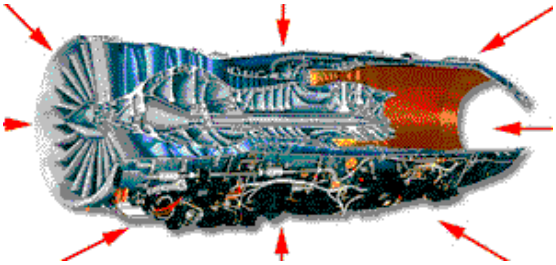

Fig. 1: An aircraft engine is complex piece of machinery comprised of blades, compressors, electronics, etc.

Parts of the engine that often encounters damage are the blades of the engine and the hub. The larger turbine blades at the front of the engine are often damaged when they ingest foreign material. The blades will deform which can cause them to crack which, in turn, can cause pieces to be ingested in the latter stages of the turbine. Jet engines on Navy aircraft are often damaged by ingesting debris that exists on the deck of an aircraft carrier. While this debris will damage blades in the engine, the detection of the damage is not necessarily apparent through visual inspection. In fact, significant damage and possible catastrophic failure of the engine may be several flight hours off.

In order to improve the maintenance of its aircraft, the Navy has started investigation of the use of imbedded sensors in the engine to detect faults before catastrophic failure. In this paper, we apply a neural-fuzzy classification technique to detect faults using embedded sensors data that measures the blade position. By using this classification technique, we are able to monitor blade position changes in the turbine which could indicate blade deformities.

In Section II of the paper we overview the hierarchical neural network (HNN) algorithm that we use to determine whether the engine is in a normal state or a fault state. In Section III, we describe the data set that was used to demonstrate our clustering algorithms capabilities, and, finally, in Section IV we present the results of our algorithm along with analysis.

2 The Hierarchical Neural Network

In this work, we looked at the fault detection problem as a simple pattern classification problem. Given a number of *objects* in the world, they are identified as belonging to a particular category (or to more than one or none of them). Our object is considered the angle of arrival of each blade. These measured angles or measurement are the input *features* from which we classify.

In general, we can have access to N features. We may think of these as forming an N-dimensional vector space:

$$F = \begin{bmatrix} feature_1 \\ feature_2 \\ \vdots \\ feature_N \end{bmatrix} \qquad (1)$$

The feature space spans the entire space that all measurements lie in. In other words, no measurement may lie outside the feature space but is considered a subspace of the feature space. In addition, there will probably be regions in the feature space which no real object (of the kind being considered in the particular problem domain) would possibly fall into. While the feature space can be linguistic in nature, a mapping from linguistics to a subset of real numbers is used.

With this framework in mind, pattern classification becomes a question of knowing what region in the feature space corresponds to a particular category or set of categories of the objects, i.e. what *class* (or classes) the object belongs to. Note that such a region may not necessarily consist of a single connected neighborhood of the space. Nor are the boundaries of a class necessarily "hard"-limited as seen classes T4 and T5 in Figure 2. If, for example, we classify various fruits with the features of color (wavelength), and diameter, the "grape" class will contain separate regions in the red, green, and violet parts of the spectrum, and the "blueberry" class will probably lie between the green and violet grapes in the feature space. The boundary between "Granny Smith" and "Golden Delicious" apple classes might well be "fuzzy", given this particular feature set, and there may in fact be overlap between the two regions, since you might imagine one of the greener Golden Deliciousness being the same size and color as a Granny Smith. There will certainly be overlap between classes and their super-classes, say, between the "Golden Delicious" and "apple" classes.

So how do EBF networks categorize? The architecture for the standard elliptical basis function (EBF) neural net is shown in Figure 3. There are two steps involved with training the EBF neural network. The first step is the clustering of all of the training set input feature vectors to form the hidden-layer elliptical basis units (EBUs) which take the form

$$EBU(i) = e^{-(u-m_i)^T C_i^{-1}(u-m_i)} \qquad (2)$$

where u is the measurement vector, m represents the mean of the i^{th} cluster and C is the associated covariance.

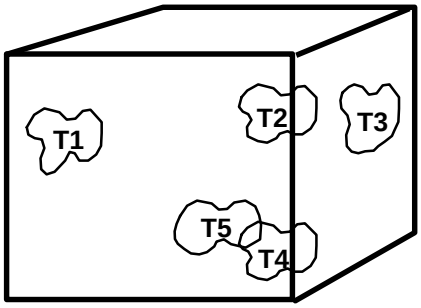

Fig. 2: A three dimensional feature space can have disjoint and overlapping classes along many regions that contain no known patterns.

A learning vector quantization (LVQ) algorithm (an example of an unsupervised learning algorithm) is used to find *prototype* vectors for some B number of N-dimensional clusters. The set of training input vectors is then divided up into B subsets where each subset, *i*, contains the input vectors which are closest according to some metric (most often, Euclidean distance) to the i^{th} prototype vector. Then the means and variances of each dimension of each subset are estimated; the means become the centroids of the EBUs and the variances set the width of the ellipsoid dimensions. Effectively, the clusters are being modeled as multi-dimensional Gaussian distributions. Finally, a "fuzzy factor" is applied which shrinks or expands uniformly each dimension of the EBUs. When an input vector is applied to the trained EBUs, they each produce an activation which is high when the input vector lies inside the particular ellipsoid and low when it lies outside.

In the second training step, the weights between the EBUs and the output units are trained using a least mean-squares (LMS) algorithm. Each training set input vector is applied to the first layer of the network, such that a set of EBU responses is formed for each one. The EBU responses are then tagged with the truth class outputs and these are used as exemplars to train the weights. In the hierarchical neural network (HNN) code, there exists the option of either using an iterative perceptron-style

algorithm or a fast pseudo-inverse technique for the LMS function approximation. After the weights are trained, there may or may not be a training of thresholds of the output units on a separate training data set; there will be if “hard” (i.e., output units “on” or “off”) classification is desired. Note that it is possible to train the system to give multiple classifications for particular inputs; this might be appropriate if a class is, for example, a subcategory of another.

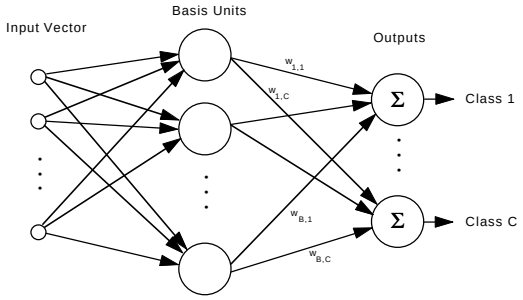

Fig. 3: Standard EBF Neural Network Architecture

The EBF network is similar to other network architectures which use basis function units, such as the fuzzy min-max neural network [2].

An important variation of the standard EBF method is the class-dependent elliptical basis function (CD-EBF) neural network, shown in Figure 3. Here, each of the EBUs now only models a single class. This partitioning of the EBUs is achieved during training of the CD-EBF neural net. The first training step is similar to that of a generic EBF net. The main difference is that instead of performing one LVQ to create B EBUs, the LVQ step is repeated for each of the classes that are considered as follows:

For each class c = 1, ..., C

- Pick the number of basis units desired to model class c. (In Figure 2, each of the classes is implied to have B EBUs associated with it.)
- Perform an LVQ to determine the prototype vectors of the EBUs for class c.
- Estimate the means and variances for each EBU associated with class c as above, and adjust the dimensions with the fuzzy factor.

At this point, there are two options. The first is to perform the LMS step as with the generic EBF neural net. The second is do no second-layer (LMS) training, and during classification, simply select the class of the EBU that has the highest activation, in a "winner-take-all" fashion. This is possible since all the EBUs now are associated with only a single class. Under this "LVQ-training-only" option, the CD-EBF neural net becomes a nearest-neighbor classifier with the class-dependent EBUs defining prototype “models” for each of the classes considered.

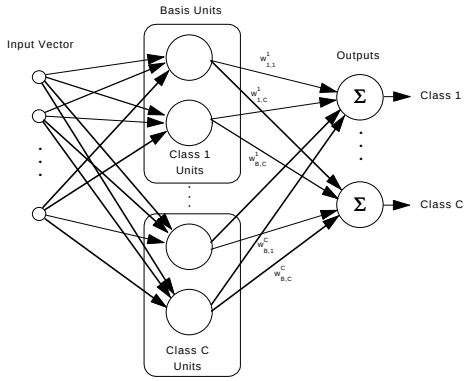

Fig. 4: Class-Dependent EBF Neural Network Architecture

For our engine fault monitoring, the CD_EBF was employed.

3 Outline of the Experimental Data

The data that was employed for our testing came from an actual F-119 jet engine test run. The engine was set to operation and various foreign object debris (FOD) was introduced into the intake of the engine. The FOD ranged from paintless paint to rubber ball to material used in an aircraft carriers flight deck. Tests were run prior to the introduction to the FOD as well.

The actual data was sample raw eddy-current data as seen in Figure 5. As described in [3], the eddy-currents vary as a blade passes through the sensor. In the test system four sensors were used. The first three measured the individual blades at three different points in the rotation. The fourth sensor measured a full rotation of the turbine shaft. The rotation sensor provided us with the rpm’s of the turbine and a reference point from which we could measure the times of arrival from each blade. The data sets were broken down into each revolution of the data. The data was processed to provide a detection of a blade and a time of arrival relative to the reference time. Then each blade’s times of

arrival over the various rotations were normalized to an angle of arrival relative to their estimated rpm.

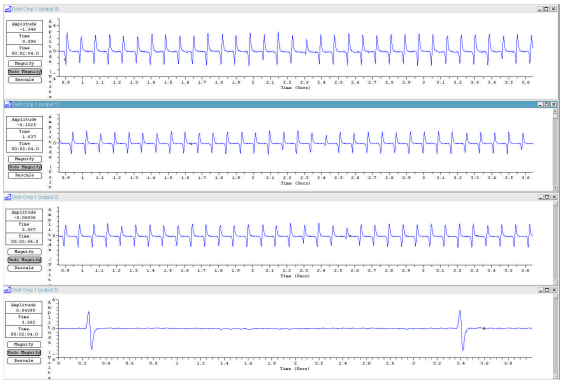

Fig.5: Three sets of eddy-current data report on individual blades while the fourth set reports total shaft rotation.

These angle of arrivals of each blade over a run provided a noisy interpretation of the data as seen in Figure 6. To filter the noise a windowed mean of the data was used. The window cover one hundred samples. An example of the result is shown in Figure 7.

Selected portions of the angle of arrival data were transformed into training and testing sets. Instead of developing a HNN to detect FOD events, the HNN was trained to recognize normal/non-FOD impacted angle of arrivals. Deviations from "normal" angle of arrivals will appear to the ellipsoidal-basis function (EBF) neural networks (NN) as an anomaly. Seven EBF neural networks were trained to compose the HNN. The training set was composed of 121,600 exemplars. Each blades data was processed separately through the HNN classifier/event detector.

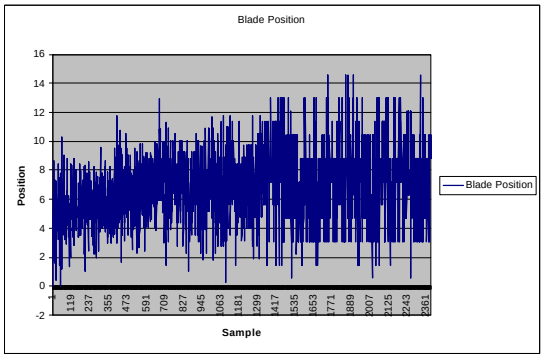

Fig. 6: The blade position is measured relative to the rpm of the engine .

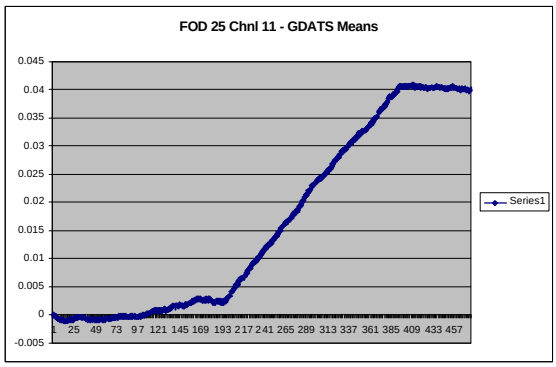

Fig. 7: Mean filtering provides a manageable representation of the data

4 ANALYSIS

We processed the data on 38 blades. Figure 8 displays the time history of 38 blades in the upper chart for the case where a rubber ball is dropped into the engine. The white indicates normal activity while the discoloration indicates a variation from the initial mean angle of arrival as detected by the HNN. The lower chart presents the final time slice of the data as processed by HNN system. According to the HNN classifier/event detector, during the last revolution of data processed, the FOD event impacted blades 9, 10, 12, 13, 14, 15, 16, 18, 22, 24, and 37.

Figure 9 demonstrates that blades experiencing deviation in the positive direction can fit nicely into a possible damage assessment scale.

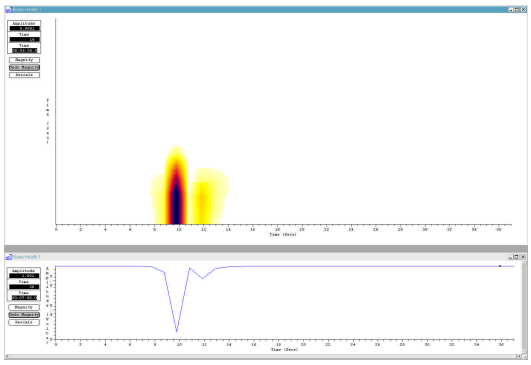

Fig. 8: A time history of relative blade angles shows that the angles of the blades change when FOD is introduced.

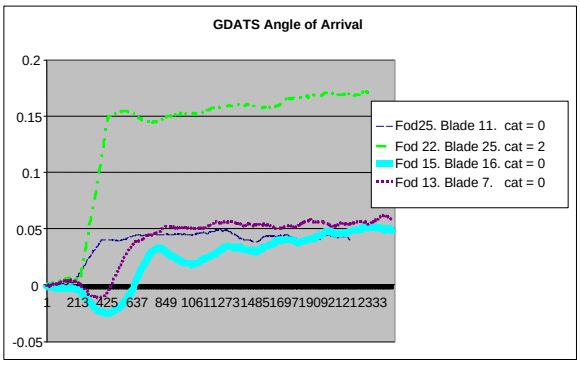

Fig. 9: The HNN detects the change in position of the blades that indicate which blades have been impacted by FOD and may be damaged.

5 Conclusions

In this effort, we developed a prototype FOD event detection system that will be integrated with other sensors to detect potential damage to a turbine engine. By smoothing the data sets we yielded data that clearly show FOD events with our HNN.. The eddy-current sensors also provide information on the length of the blade, the pitch of the blade and other features that were not exploited in this study. Future studies are encouraged to explore the benefits of using these other features that eddy-current sensors provide.

The development of the FOD event detection system provided an excellent opportunity to investigate the imple-mentation of a multi-sensor data fusion system from various sensors. However, to properly train and test the neural networks, synchronized multi-sensor data sets are required. The employment of a multi-channel and multi-rate data acquisition is necessary. The fusion of the HNN results into another HNN would be simple process.

The use of HNN as a tool for engine anomaly detection has been shown. It is a useful tool with great promise.

References:

[1] Brotherton, T., T. Johnson, and G. Chadderdon, "Classification and Novelty Detection Using Linear Models and a Class Dependent Neural Network," *Proceedings of 1998 World Congress for Computational Intelligence (IJCNN)*, Anchorage, Alaska, pp. 876-879, May, 1998.

[2] Patrick K. Simpson, "Fuzzy Min-Max Neural Networks—Part 1: Classification," IEEE Transactions on Neural Networks, vol. 3, pp. 776-786, 1992.

[3] Terpay, G.W. and G.G. Zipfel, Jr. "Measuring Blade Condition in a Gas Turbine Engine Using Eddy-Currents," preprint, 2000.