

Defect classification of highly noisy NDE data using classifier ensembles

Kai F. Goebel*, Weizhong Yan, Neil H. W. Eklund, Xiao Hu
GE Global Research, One Research Circle, Niskayuna, NY, USA 12309
Vishwanath Avasarala
Penn State University, College Station, PA, USA
Jose Celaya
Rensselaer Polytechnic Institute, Troy, NY, USA

ABSTRACT

In this paper, we present a feature selection and classification approach that was used to assess highly noisy sensor data from a NDE field study. Multiple, heterogeneous NDT sensors were employed to examine the solid structure. The goal was to differentiate between two types of phenomena occurring in a solid structure where one phenomenon was benign, the other was malignant. Manual distinction between these two types is almost impossible. To address these issues, we used sensor validation techniques to select the best available sensor that had the least noise effects and the best defect signature in the region of interest. Hundreds of features were formulated and extracted from data of the selected sensors. Next, we employed separability measures and correlation measures to select the most promising set of features. Because the NDE sensors poorly described the different defect types under consideration, the resulting features also exhibited poor separability. The focus of this paper is on how one can improve the classification under these constraints while minimizing the risk of overfitting (the number of field data was small). Results are shown from a number of different classifiers and classifier ensembles that were tuned to a set true positive rate using the Neyman-Pearson criterion.

Keywords: Classification, diagnostics, abnormal condition detection, information fusion, classifier fusion

1. INTRODUCTION

Condition monitoring in solid structures involves identification of defects that impact the normal equipment functioning. Advanced signal processing techniques are required to determine the relationship between system condition and sensor signals. Non-destructive evaluation (NDE) techniques typically rely on multi-sensor data fusion, since the information provided by a single sensor may not be accurate or adequate to estimate system condition. Typical systems for condition monitoring involve feature extraction from the sensor data and pattern recognition algorithms that operate on the extracted features to characterize the state of the system. To maximally exploit sensor information, sensor information is transformed into various domains, where different features are extracted with the potential of providing (or at least contributing) to the desired characteristics. Examples are frequency domain features like spectral analysis and time domain features like mean value, kurtosis etc. Performance of pattern recognition algorithms is contingent on quality of sensor data and the number of training samples available. In many practical applications, NDE sensor data are highly noisy and number of training samples available is limited.

While classification accuracy is the most widely used performance measure, there are cases where misclassifying one class may have much more severe cost consequences than others. There, classifier performance is typically specified by maintaining one type of error (the costly one) to a pre-determined level while using the other type of error as the remaining parameter that needs to be optimized. We call this type of performance requirements for classifier design the *constrained performance requirement*. For the NDE to be successful in such circumstances, the choice of pattern recognition algorithm and features used for classification becomes critical.

One particular set of algorithms is classifier ensembles. Classifier ensembles are one of the most active research areas in machine learning⁸. An ensemble of classifiers is a set of classifiers whose decisions or outputs are combined to arrive at

* goebelk@crd.ge.com; phone 1 518 387-4194;

a final classification decision. The individual classifiers in the ensemble can be any type of classifiers, preferably “unstable” classifiers like neural networks and decision trees. “Unstable” refers to the fact that a small change in the training samples will lead to a classifier with different classification performance ².

Classifier ensembles have been used for accuracy improvement ^{11,12} in various application domains ranging from optical character recognition (OCR) ¹⁴ to seismic signal classification ¹⁹. For ensembles to be more accurate than the constituent classifiers in the ensemble, individual classifiers need to be at least modestly accurate, but – more importantly – need to be diverse. That is, they need to make classification errors on different examples ¹¹. In addition, the type of aggregation methods also plays an important role in improving performance.

While current practice in ensemble design has been focusing to a large degree on improving classification accuracy ¹⁶, not much attention has been devoted to classification problems with constrained performance requirements ²³. Specifically, we want to know whether generating accurate and diverse classifiers for ensembles, which proves to work well for accuracy improvement, guarantees that an ensemble performs better with respect to the constrained performance requirement as well.

In the following, we attempt to demonstrate that classifier ensembles may be effective in improving classification performance for problems with low class separability and with constrained performance requirement. The rest of this paper is organized as follows. Section 2 provides an overview of feature selection and classifier design. Section 3 discusses classifier fusion. Section 4 addresses performance evaluation. Section 5 presents results that are discussed in Section 6 and summarized in Section 7.

2. CLASSIFICATION

2.1. Features

The principal issue faced in the problem at hand is a large feature space relative to the number of cases. In this situation, one can easily encounter the “curse of dimensionality”, i.e., performance of the classifier is based on small statistical abnormalities rather than reliable and generalizable features. Hence, a common strategy to avoid overfitting is to allow the classifier to only use a small number of features. The process of going from a large feature space to a smaller one is known as feature selection.

Domain knowledge is an important tool for feature selection. For example, there is motivation to discount features that are likely to just add noise on the grounds of the underlying physics. Another tool for feature selection is the Fisher discriminant ⁹, which measures univariate linear class separability. The operative equation for the Fisher discriminant

method is $D_{01} = \frac{|m_0 - m_1|^2}{\sigma_0^2 + \sigma_1^2}$ where m_i refers to the mean and σ to the standard deviation of the two classes.

Fundamentally, it rewards large distances between class means and penalizes large scatter within the classes. Generally, features with higher separability have the promise to contribute more to the overall classification, at least for linear classifiers. Figure 1 shows the separability expressed by the Fisher discriminant. However, additional measures have to be taken into consideration for feature selection. In particular, one has to assess the correlation of features. If the features with the highest separability are redundant, it is advisable to only use one of those redundant features, and other features with lower separability need to be added. Moreover, when the features are heavily biased towards a particular subset, it may be a prudent to restrict the number of features from these groups to enforce higher diversity of the resulting set. Furthermore, selection criteria such as robustness between different inspection runs (i.e., when the tool is exposed to slightly different environments) should be considered as well.

Fundamentally, to address the issue of overfitting, the feature selection should result in a small number of features. To that end, we constrained the training set to 10 features. Since each classifier used its own feature selection, the particular methods will be described in the classifier sections.

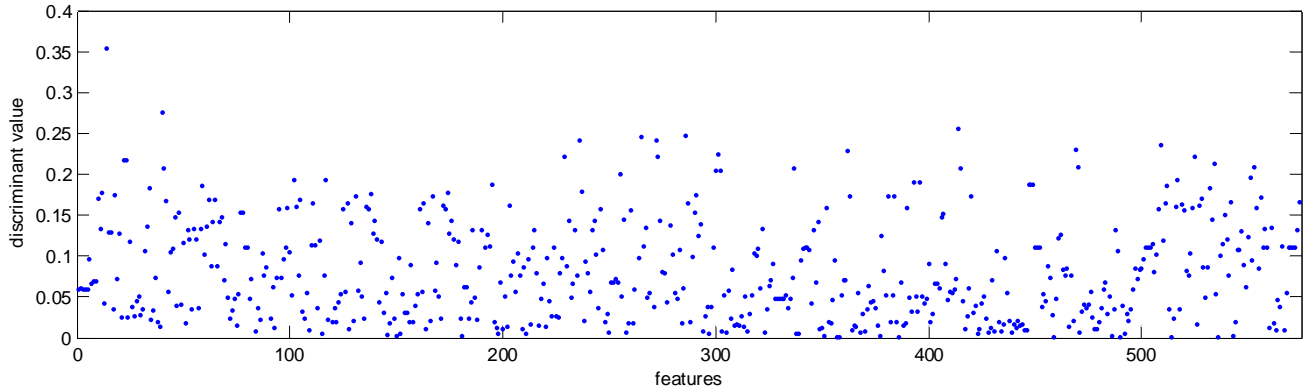


Figure 1: Separability for poor quality cases

2.2. Classifiers

The classifier will serve the purpose to find a separation between 2 different types of defects using a suitable subset of features. Fundamentally, a classifier tries find decision boundaries between selected features to separate the different classes.

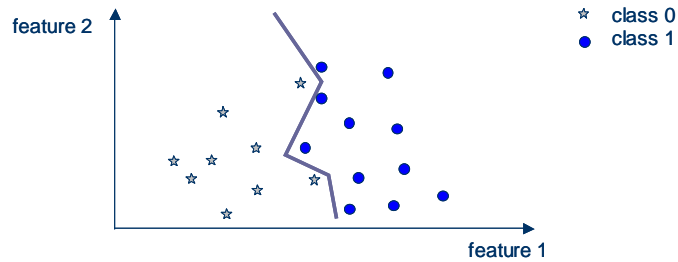


Figure 2: Decision boundary of classifier

A suite of diagnostic classifiers was built to integrate selected features to make a diagnostic decision. These models are linear regression, generalized regression neural network (GRNN), several random forests (RF), and support vector machine (SVM). Each model has its advantages and disadvantages. Since they are quite different in principle, they give different decisions about the potential defects thus meeting one requirement for a successful fusion. Specifically, by providing the diversity in the diagnostic models, it creates the opportunities of fusing the decisions from each model together to make a better assessment.

Random forests

Random Forest⁴ is a classification method that applies bagging³ to a variation of classification trees⁵. A standard classification tree is constructed by splitting the data on the best of all possible features at each node. For RF, only a randomly selected subset (chosen always from the full set) of features is eligible to split each node. Moreover, each individual tree is constructed on a bootstrap sample of the data. Finally, in contrast to standard classification trees, the individual RF trees are not pruned; rather they are typically grown to 100% node purity. Although hundreds of trees may be developed, RF's are very quick to train (e.g., much faster than neural networks for a given data set and processor). Predictions are made by aggregating the predictions of the ensemble (majority vote for classification or averaging for regression). Random forest generally exhibits a substantial performance improvement over the single tree classifier such as CART and C4.5. It yields generalization error rate that compares favorably to Adaboost, yet is more robust to noise. The RF implementation used here was by Breiman et al.⁶.

One of the beneficial features of the RF algorithm is its use in variable selection. To estimate the importance of a particular variable, the values of that variable are permuted, and predictions of the out-of-bag cases for each tree are

made using the permuted data. For a particular case, the margin is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other classes. The RF importance measure for a particular variable is the average lowering of the margin across all cases when that variable is randomly permuted. This value can be calculated quite quickly for each variable in turn.

The data was preprocessed to address some obvious limitations. The initial data set was 141 cases by 575 features. Eight features were removed that were constant for all cases. 158 features were removed because they were highly correlated with other features in the data set ($r^2 > .97$). Finally, 47 variables were removed for having less than 15 cases not equal to the mode (this type of feature is unlikely to generalize well).

The RF variable importance score was used to down select from the remaining 362 features. Because the features are so poor, and many are highly correlated, the feature selection based on only one forest is not very sensitive. To overcome this, 8000 separate RFs were trained, each with 1000 trees and 20 variables examined at each split. Only the 10 features with median RF importance greater than zero over the 8000 runs were used for final classification. RF1 (Table 1) uses all 10 features; RF2-RF5 use 5 randomly selected (from the final set of 10) features. Each RF classifier was composed of 2000 trees examining 2 variables at each split.

Linear discriminant classifier

After ranking the features according to their individual discriminant value, the features are evaluated with respect to their cross-correlation. Where features are highly correlated, only one feature (the one with highest discriminant value) is retained. On top of that, the features underwent a forced group minimization process whereby features were selected to ensure maximum diversity by enforcing that features from sensors using different physical principles were selected (even when they had a somewhat lower discriminant score). Next, a linear discriminant classifier was employed. This classifier

attempts to find a projection (W) that maximizes class separability $J_{LDA}(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$, where $W = eig(S_W^{-1} S_B)$, the

within-class scatter matrix $S_W = \sum_{i=1}^c \sum_{x \in D_i} (x - m_i)(x - m_i)^t$, and the between-class scatter matrix $S_b = \sum_{i=1}^c n_i (m_i - m)(m_i - m)^t$.

The projection implies that the classifier will represent the original features with a smaller number of new features that are the linear combination of the original ones. Graphically, it projects the data from a high dimensional feature space to a low dimensional one.

Linear discriminant features may not necessarily be orthogonal. This is in contrast to the principal component analysis (PCA). PCA is an unsupervised transformation that is considered to be a very good tool for data representation. PCA features are orthogonal to each other and rotated to best describe data variation. LDA, on the other hand, is a supervised transformation that maximizes the class separability. It is therefore better suited for classification. Figure 3 illustrates the differences.

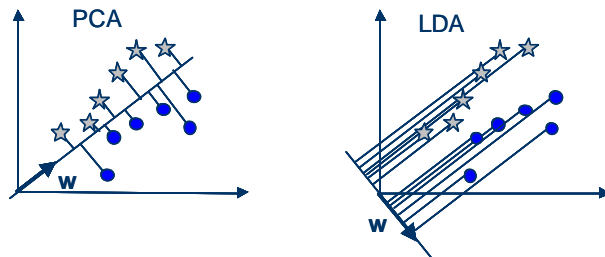


Figure 3: Differences of transformation of PCA and LDA

Note that the largest number of discriminant features is equal to $(C - 1)$. In our case, that means that there is only one discriminate feature. That in turn implies that only one decision threshold needs to be set. Limitations of the linear discriminant are that it does not deal well with nonlinearly separable data sets and classes with the same mean.

For the small sample size problem (which we encounter here), there are some additional considerations. Because matrix W is computed by inverting S_w , with small number of samples, S_w may become singular; Thus, inverting S_w can introduce a large error! Remedies are either regularization, i.e., $S_w \rightarrow S_w + \alpha I$ ($\alpha =$ small number), or through a subspace-based method where features are pre-transformed with a PCA and then subjected to the LDA.

Generalized regression Neural Network

Neural networks, usually composed of simple elements (called neurons) operating in parallel, are well known for their function approximation and pattern recognition performance¹. They have been successfully applied across a wide range of domains, such as finance, engineering, medicine, physics, etc.

The generalized regression neural network (GRNN) is a universal approximator and typically used for function approximation. It has been shown that, given a sufficient number of neurons, GRNN can approximate a continuous function to an arbitrary accuracy. GRNN has two layers of artificial neurons. While the first layer consists of radial basis neurons (whose transfer function is a Gaussian with a spreading factor), the second layer consists of neurons with a linear transfer function. In this paper, the inputs to the GRNN are the features and the single outputs are the target class labels. We set the spreading factor of Gaussian function as 1.5.

Linear Regression Neural Network

The basic concept of regression is to determine a functional relationship between two or more correlated variables that is often empirically derived from data and is used especially to predict values of one variable when given values of the others. A linear neural network, which essentially performs linear regression, can also be used as a decision fuser to integrate outputs from different classifiers. A linear network has one layer of S neurons connected to R inputs through a matrix of weight \mathbf{W} . The network output is $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$, where the input \mathbf{x} is R by 1 vector, weight \mathbf{W} is S by R matrix, bias \mathbf{b} is R by 1 vector and \mathbf{y} is S by 1 vector. Since the diagnostic model only needs to give a distinction between 2 defects as output, S equals to 1 here. The output of the linear neural network needs to be appropriately thresholded to yield the classification decision.

Support vector machine

We also consider here a Support Vector Machine (SVM) as the classification engine^{7,21}. A SVM is a maximum margin linear classifier. It is defined by a positive definite kernel function $k(x, x')$ specifying an inner product in the feature space $\Phi(x)$. $\Phi(x') = k(x, x')$ given a labeled data set as in Eq. (1). The Gaussian radial basis function

$$k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}} \quad (1)$$

can be used as a kernel function. Input patterns for feasible Lagrange multipliers of the optimization scheme are called support vectors¹⁷.

Two support vectors were generated. They largely differ in their preprocessing techniques. SVM₁ first ranks features by Fisher discriminant and t-stat score and further eliminates highly correlated features using cross-correlation. SVM₂ uses a similar process but adds a physics-based feature constraint where features from certain sensors were not allowed to occupy more than 50% of the final features.

3. CLASSIFIER FUSION

Motivated by a classifier performance ceiling where – despite extensive tuning attempts – performance lingers below desired limits, classifier fusion makes an effort to break through this ceiling. There are numerous approaches such as bagging and boosting¹⁰, Dempster-Shafer²⁰, model-based approaches¹⁵, fuzzy fusion¹³ or statistics-based approaches¹⁸ that attempt to address the core aggregation functions.

3.1. Averaging.

Breiman² and Krogh et al.¹² have shown that a simple average of the outputs of individual classifiers is an effective combining scheme. First, classifier output that is not scaled between [0, 1] was normalized as a preprocessing step using a non-linear regression of the form

$$x_n = \frac{1}{1 + e^{-\alpha x + bias}}. \quad (2)$$

The individual classifiers were then simply averaged.

3.2. Dempster-Shafer fusion.

Dempster-Shafer fusion²⁰ has the advantage to allow incorporation of classifier uncertainty estimates. In situations when there are multiple classifiers with different sensitivities and reliabilities it might be desirable to have mechanisms to cast doubt on the validity of the output. Discounted belief masses are combined using Dempster's rule of combination. The Dempster-Shafer theory starts by assuming a universe of discourse consisting of a finite set of mutually exclusive atomic hypothesis $\Omega = \{\theta_1, \dots, \theta_q\}$. Let 2^Ω denote the subset of Ω . The a function $m: 2^\Omega \rightarrow [0, 1]$ is called a basic probability assignment if it satisfies $m(\emptyset) = 0$ and $\sum_{A \subseteq \Omega} m(A) = 1$. Belief can be assignment to some set $A = \{a_1, \dots, a_n\}$. The belief function $bel: 2^\Omega \rightarrow [0, 1]$ is defined as $bel(A) = \sum_{\theta \subseteq A} m(\theta), \forall A \subseteq \Omega$. Most interesting to our problem is the ability to

combine two basic probability assignments via $m_{12} = \frac{\sum_{B \cap C = A} m_1(B)m_2(C)}{1 - K}$, where $K = \sum_{B \cap C = \emptyset} m_1(B)m_2(C)$. One critical

issue is how to extract the basic probability assignments from the classifier output. For the 2-classifier problem, we use the normalized classifier output y_i as assignment for class 1 and to use the values subtracted from 1 as the assignments for class 2. Critical for the fusion is also to model the degree of ignorance. We propose here to use an operation that assigns values close to the decision threshold (assumed to be 0.5) a relatively large degree of ignorance. This is

accomplished by $\theta_i = 1 - \frac{|y_i - 0.5|}{0.5}$.

Since Dempster-Shafer performs pair-wise fusion, it is somewhat dependent on the order of the fusion as well as which members get fused. If the number of classifiers is small, this can be done exhaustively.

3.3. Random forest meta-classifier

For the random forest (RF) fusion approach, the raw classifier output was treated as a new feature set for classification. The output from six selected classifiers (using the RF-based feature selection process outlined in section 2.2) were used to train RFs with 1000 trees selecting only one variable at random at each split. The classifiers selected were RF₂, RF₃, RF₄, RF₅, LDA, and SVM₁

4. PERFORMANCE EVALUATION

While overall classification accuracy (or alternatively overall classification error) is still the most popular measure of classifier performance, its underlying assumption is that the classification errors for all classes have equal cost consequences. Therefore, it tends to reward the best balance between different misclassifications. However, in real applications, different misclassification incur different costs. This cost criterion may drive the decision threshold of a classifier away from the point of overall best accuracy. In this paper, we consider the constrained performance measure. That is, a performance at a predefined setpoint. We will first define error types and accuracy and then discuss constrained performance.

4.1. Error types

To start, we define two types of classification errors (false positive and false negative) and ROC curves (short for Receiver Operating Characteristic or Relative Operating Characteristic) that show tradeoffs between the two types of errors for a classifier below. We restrict our discussion to two-class classification problems where the two classes are referred to as the positive and negative classes.

		Classes assigned by Classifier	
		Negative	Positive
True Classes	Negative	N^{00}	N^{01}
	Positive	N^{10}	N^{11}

Figure 4 - A typical confusion matrix for 2-class classification problems

Consider a classification problem specified by a test set, $\{\bar{x}_i, y_i\}$, $i = 1, 2, \dots, l$, where $\bar{x}_i \in R^D$ is the i^{th} input vector and $y_i \in \{0, 1\}$ (“0” stands for negative class and “1” for positive class) is the corresponding class label. Outputs of the classifier to the test data set are typically summarized in a *confusion matrix* as shown in Figure 4.

A classifier can make two types of errors, namely, false positive error and false negative error. In the field of statistical hypothesis testing, they are referred as Type I and Type II errors. $P(\text{false positive}) = P(\text{classified positive} | \text{negative})$, and $P(\text{false negative}) = P(\text{classified negative} | \text{positive})$.

The tradeoff between the two types of errors for a classifier can be better depicted by ROC curves. ROC curves represent the behavior of a classifier without regard to class distribution or error cost. Hence ROC curves are a true representative of classifier performance. In ROC space, the true positive rate, TPR (TPR=1-FNR), is plotted on the Y-axis and the false positive rate, FPR, is plotted on the X-axis, where the TPR is commonly referred to as “sensitivity” while (1-FPR) is called “specificity”. A point in ROC space corresponds to a (FPR, TPR) pair of a classifier. Classifiers with ROC curves located in the upper-left corner in ROC space are better because they represent classifiers that have lower false positive rate and higher true positive rate than the classifiers below them²². ROC curves are typically generated by varying either some parameters of the classifier (e.g. threshold in neural networks) or costs (e.g. in decision trees).

4.2. Overall accuracy

From the confusion matrix defined above, the accuracy can be calculated as follows.

$$acc = \frac{N^{00} + N^{11}}{N^{00} + N^{01} + N^{10} + N^{11}} \quad (3)$$

Another summary measure is the area under the curve (AUC) of the ROC.

4.3. Constrained performance

It is worthwhile to note that, for a given classifier, the accuracy above varies with decision actions. In fact, by varying the decision threshold, one can reduce one type of errors to any arbitrarily small number through sacrificing another type of errors. This is at the heart of constrained performance. Constrained performance requirements can be specified in different forms, depending on which rate (FPR or FNR or both) needs to be set to a predetermined level. For the rest of this paper, we only consider the constrained performance requirement as minimizing FPR while maintaining TPR (= 1-FNR) at a predetermined (large) number. Let the predetermined large number be α , and then the constrained performance can be expressed as

$$\min(FPR) \text{ subject to } TPR \geq \alpha \quad (4)$$

Mathematically, the constrained performance requirement is a constrained optimization problem that is somewhat difficult to solve. As a result, a practical approach to design a classifier to satisfy a constrained performance requirement is to generate a ROC curve first for the classifier by varying either the decision threshold, or some classifier structure parameters. Then find the corresponding FPR from the ROC curve based on the predetermined TPR number.

The constrained performance requirement (Eq. 4) can be graphically represented in ROC space as shown in Figure 5. With this requirement, the designer will attempt to obtain a better classifier by moving the interception point of the ROC curve and the line defined by $TPR = \alpha$ towards the Y-axis along line $TPR = \alpha$ (i.e., from point A to point B in Figure 5). This design strategy is different from that when accuracy maximization is the performance requirement. In the latter case, the designer is trying to obtain a better classifier by moving the ROC curve as close to point [0,1] as possible in the ROC space. As can be seen in Figure 5, if the predetermined TPR, α , is large (close to 1.0), the interception will occur in the “flat” portion of the ROC curve, which implies that a great amount of accuracy increase is needed to achieve even a small amount of reduction of FPR.

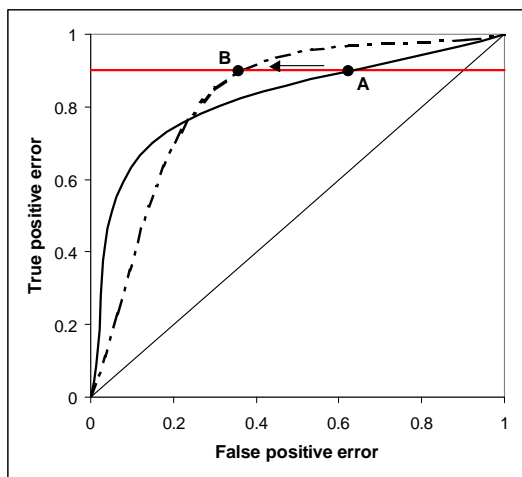


Figure 5: Graphic illustration of constrained performance in ROC space

5. RESULTS

The real-world application concerned in this paper is an automated data analysis system for non-destructive inspection. To perform classification, 575 features from different domains were extracted. For classifier design, 141 examples representing different conditions of the target object are used. Of the 141 examples, 51 are for the malignant defect condition while 90 are for the benign condition.

We show firsts the results from single classifiers followed by results from the fusion methods.

5.1. Single classifier results

A leave-one-out (LOO) approach was employed, where all except one data points were used for training. The one remaining data point was used for testing. The test point and the training sets were then rotated such that 141 different classifiers of each type were built. ROC curves were generated based on the output of these 141 points. This ensures best approximation of performance of the algorithms. Note that we used the leave one out procedure for both feature selection and classifier design. This will result in a more conservative performance assessment.

Table 1 – Performance of individual classification approaches

classifier	$FP_{TP=98\%}$	best accuracy
RF ₁	0.940	0.766
RF ₂	0.916	0.787
RF ₃	0.952	0.787
RF ₄	0.952	0.773
RF ₅	0.892	0.794
LDA	0.831	0.794
GRNN	0.892	0.780

LN	0.976	0.780
SVM ₁	0.928	0.787
SVM ₂	0.819	0.780

Table 1 compares the performance of the models in terms of their best overall accuracy as well as the minimum FP rate at the setpoint TP=98% on the 141 data points using the LOO procedure. The ROC curves of the individual classifier models are shown in Figure 6. SVM₂ give the best $FP_{TP=98\%}$ rate performance. Note that this classifier does poorly as measured by the overall best accuracy (which is not of interest here).

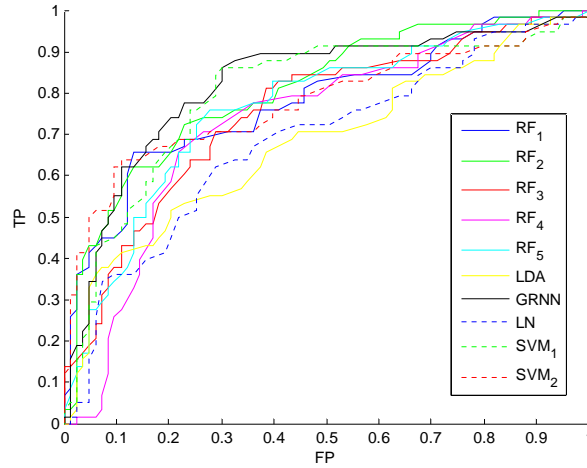


Figure 6 – ROC curves of the diagnostic models.

5.2. Fusion results

As a baseline fusion method, we employed the simple averaging scheme. While improving overall accuracy, averaging does not improve $FP_{TP=98\%}$. RF3 performs best with $FP_{TP=98\%} = 0.647$. The best Dempster-Shafer fuser is DS₂. It fuses the output of DS₁ (which is the Dempster-Shafer fusion of SVM₂ with LDA) with RF₅ and achieves $FP_{TP=98\%} = 0.747$. Adding further classifiers to the Dempster-Shafer fusion scheme does not improve the results as seen by poorer performance of DS₃ and DS₄. The fusion results are summarized in Figure 7 and Table 2.

Table 2 - Performance of fusion approaches

fusion method	$FP_{TP=98\%}$	best accuracy
Averaging (all)	0.855	0.801
Averaging (RF ₂ , RF ₃ , RF ₄ , RF ₅ , LDA, SVM ₁)	0.831	0.794
DS ₁ : SVM ₂ w/LDA	0.831	0.766
DS ₂ : DS ₁ w/RF ₅	0.747	0.794
DS ₃ : DS ₂ w/GRNN	0.855	0.801
DS ₄ : DS ₃ w/RF ₂	1.000	0.794
RF fusion (RF ₂ , RF ₃ , RF ₄ , RF ₅ , LDA, SVM ₁)	0.647	0.770

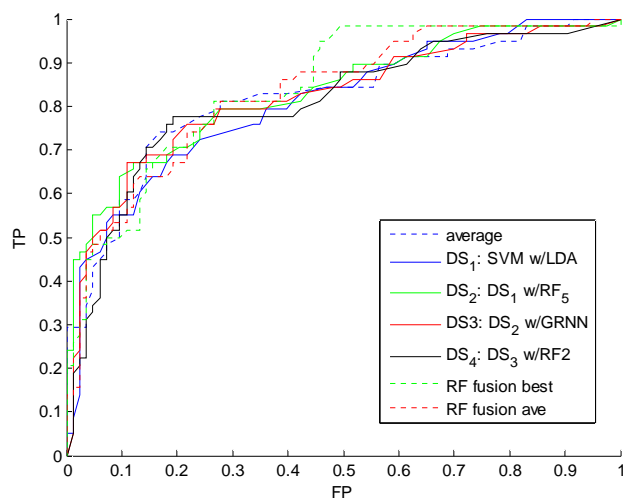


Figure 7 - ROC curves for different fusion methods

6. DISCUSSION

Fusing classifiers with low false positive rate seems to be a more successful strategy than fusing all classifiers indiscriminantly. For example, the averaging fusion, the Dempster-Shafer Fusion, and the RF-based fusion attain their best performance with low FP-classifiers. All of them use the LDA, RF₂, and RF₅. It is also interesting to note that the SVM₁, which is a lower overall performer, is part of a successful fusion strategy. This re-emphasizes the hypothesis that diversity in classifiers (namely, to err at different cases) is more important than overall best accuracy of the classifiers.

The success of the fusion may result in large part to the different feature selection criteria employed which ensured a higher diversity, a prerequisite for fusion to attain performance improvement. Certainly, the different classifiers contributed to the diversity as well. But because each classifier employed independent feature selection methods, it is reasonable to assume that they may have come up with different features (given that no particular feature appeared to stand out). Fusion (as used here) will therefore increase the number of parameters used in the overall classification, thus violating – at least in this special case of sparse data – the maximum parameter heuristic. While the leave-one-out method was also extended to the fusion design (a very conservative way to predict future performance), there is a residual chance that the performance improvement observed may not hold during validation.

One interesting feature of this problem is the dependence on the test statistic for fusion method selection. While the averaging approach is most accurate, the RF fusion has the lowest $FP_{TP=98\%}$. However, from a practical perspective, it is important to note that the variation in the $FP_{TP=98\%}$ statistic is much greater than the overall accuracy statistic. Figure 8 shows the histogram of $FP_{TP=98\%}$ (Fig 8a) and the overall accuracy (Fig 8b) for 1000 runs of the RF fusion. The standard deviation of the overall accuracy statistic, 0.0064, is an order of magnitude less than the variation in the $FP_{TP=98\%}$ statistic. This variation is likely a result of the instability of the length of the tail on the positive cases – slight variation in the location of the 98th percentile leads to huge variation in the number of false positives. Thus, while we can provide an estimate of the $FP_{TP=98\%}$, the confidence interval around this statistic is fairly large. This variation of FP statistics may have implications on ranking the fusion methods.

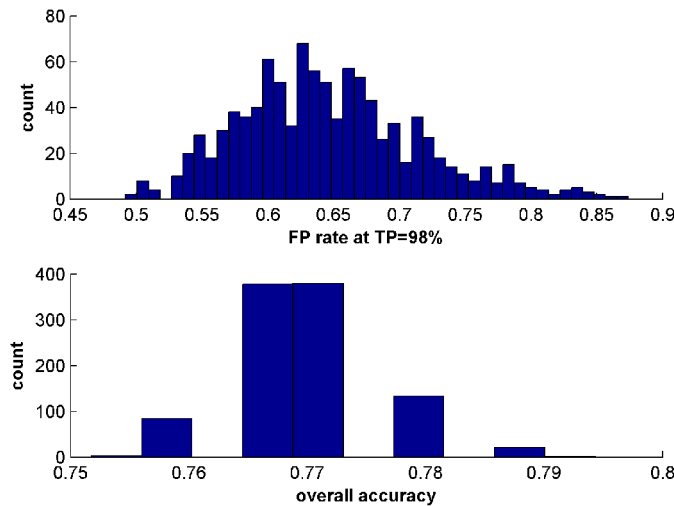


Figure 8. Histogram of $FP_{TP=98\%}$ statistic (top axes) and overall accuracy (lower accuracy) for 1000 runs of the RF fusion.

While a true test for future performance will likely be only achieved in validation runs from additional field tests, assessment of performance confidence under the circumstances encountered here will be the subject of future work. Generally, classifiers and classifier fusion methods that can integrate the $FP_{TP=98\%}$ minimization into the design by making them part of the fitness function (such as the RF approach) have an inherent advantage over “blind” approaches that attempt to find the best balance between the error types through maximization of overall accuracy.

7. SUMMARY

In this paper, we present a method to increase classification accuracy for highly noisy sensor data and features with poor separability. The goal was to differentiate between two types of defects where one defect needed to be identified with high sensitivity. Features were selected and multiple classifiers were designed using an overall leave-one-out loop. The classifiers were then fused using different fusion strategies. Fused results show improved false positive rate at the required performance setting compared to any single classifier alone.

REFERENCES

1. Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford University, Press.
2. Breiman, L. (1996a), “Stacked regression”, *Machine Learning*, Vol. 24, No. 1, pp 49-64.
3. Breiman, L. (1996b), “Bagging predictors”, *Machine Learning*, Vol.24, No.2, pp 123-40.
4. Breiman, L. (2001), “Random forest”, *Machine Learning*, 45(1), pp. 5–32.
5. Breiman, L.; Friedman, J.; Olshen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth, Belmont, CA.
6. Breiman, L. and Cutler, A. (2004), *Random Forest Toolbox Version 5.1*, http://www.stat.berkeley.edu/users/breiman/RandomForestes/cc_software.htm, University of California at Berkeley.
7. Cawley, G.C. (2000), *Support Vector Machine Toolbox v0.50 beta*, <http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox>, University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K. NR4 7TJ.
8. Dietterich, T.G. (1999), “Machine learning research: four current directions”, *AI Magazine*, Vol. 18, No. 4, pp97-136.
9. Fisher, R.A., (1936). The use of multiple measurements in taxonomic problems. *Ann. Eugenics* 7, 178-188.
10. Freund, Y. and Schapire, R. (1999), “A Short Introduction to Boosting”, *J. Japanese Society Artificial Intelligence*, Vol. 14, No.5, pp. 771-780.
11. Hansen, L. and Salamon, P. (1990), “Neural network ensembles”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 10, pp 993-1001.

12. Krogh, A. and Vedelsby, J. (1995), "Neural network ensembles, cross validation, and active learning", In Tesauro, G., Touretzky, D. & Leen, T. (Eds), *Advances in Neural Information Processing Systems*, Vol. 7, pp231-238, Cambridge, MA, MIT Press.
13. Loskiewicz-Buczak, A. and Uhrig, R. (1994), "Decision Fusion by Fuzzy Set Operations", *Proc. third IEEE Conf. Fuzzy Systems*, Vol. 2, pp.1412-1417.
14. Mao, J. (1998), "A case study on bagging, boosting, and basic ensembles of neural networks for OCR", *Proceedings of IEEE World Congress on Computational Intelligence*, Vol. 3, pp1828-33
15. Nelson, M. and Mason, K. (1999), "A Model-Based Approach to Information Fusion". *Proc. Information, Decision, and Control*, pp. 395-400.
16. Opitz, D.W. (1999), "Feature selection for ensembles", *Proceedings of 16th International Conference on Artificial Intelligence*, pp379-384.
17. Platt, J.C. (1999), "Fast training of support vector machines using sequential minimal optimization", *Advances in Kernel Methods - Support Vector Learning*, (Eds) B. Scholkopf, C. Burges, and A. J. Smola, MIT Press, Cambridge, Massachusetts, chapter 12, pp. 185-208.
18. Rao, N.S.V. (2000), "Finite Sample Performance Guarantees of Fusers for Function Estimators, *Information Fusion*, Vol. 1, no. 1, pp. 35-44.
19. Shimshoni, Y. and Intrator, N. (1998), "Classification of seismic signals by integrating ensembles of neural networks", *IEEE Transactions of Signal Processing*, Vol. 46, No. 5, pp1194-1201.
20. P. Smets (1994), "What is Dempster-Shafer's model?" *Advances in the Dempster-Shafer Theory of Evidence*, Yager, R., Fedrizzi, M., and Kacprzyk, J., (Eds.), John Wiley & Sons, New York, pp. 5-34.
21. Vapnik, V.N. (1995), *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.
22. Yan, W., Goebel, K., and Li, J. (2002), "Classifier Performance Measures in Multi-Fault Diagnosis for Aircraft Engines", *Proceedings of SPIE, Component and Systems Diagnostics, Prognostics, and Health Management II*, vol. 4733, pp. 88-97.
23. Yan, W., and Goebel, K. (2004), "Designing Classifier Ensembles with Constrained Performance Requirements", *Proceedings of SPIE; Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications 2004*, pp. 59-68.