ABSTRACT

The US Army Condition Based Maintenance program collects data from Health and Usage Monitoring Systems, Flight Data Recorders, Maintenance Records, and Reliability Databases. These data sources are not integrated, but decisions regarding the health of aircraft components are dependent upon the information stored within them. The Army has begun an effort to bring these data sources together using Machine Learning algorithms. Two prototypes will be built using decision-making machines: one for an engine output gearbox and another for a turbo-shaft engine. This paper will discuss the development of these prototypes and provide the path forward for implementation. The importance of determining applicable error penalty methods for machine learning algorithms for aerospace applications is explored. The foundations on which the applicable dataset is built are also explored, showing the importance of cleaning disparate datasets. The assumptions necessary to generate the dataset for learning are documented. The dataset is built and ready for unsupervised and supervised learning techniques to be applied.

NOTATION

\[ E_{in} \]: The in-sample error measured internally, and actively reduced during training.

\[ E_{out} \]: The out-of-sample error measured externally, and used for reporting based on test data never used during training.

\[ N \times p^* \]: The number of unique examples of component condition is \( N \). These examples are fully observed by a minimum number of data samples prior to teardown analysis or other equivalent ground truth grading procedure. The number of selected and extracted features is \( p \), and the maximum size of this vector is \( p^* \).

INTRODUCTION

The Army Aviation Condition Based Maintenance Plus (CBM+) Program has focused on monitoring the functional failure of dynamic components, recording engine parameters, and developing regime recognition. Digital Source Collectors installed on all manned rotorcraft record vibration and parametric data. Honeywell Aerospace and United Technologies Aerospace Systems manufacture the hardware, Modernized Signal Processing Unit (MSPU) and Integrated Vehicle Health Management System (IVHMS), respectively. These systems are collectively known as Health and Usage Monitoring Systems (HUMS). The goal of the CBM program is to reduce sustainment costs and enhance aircraft safety. The subject machine learning program is being completed by the same team that reviews the vibration data used to predict dynamic component failure internal to gearboxes and bearings. Reference 1 gives a summary of the background and motivation for the CBM+ efforts.
The purpose of this paper is to expand upon the machine learning and change detection program to improve the functionality of the HUMS. This paper will bring together the efforts to begin the unsupervised learning associated with two components: engine output transmissions (internal component health diagnostic) and turbo-shaft engines (logistics and maintenance diagnostic). Furthermore, the authors have explained the learning processes published by Wade (Ref. 1) to include valuable information from Brower and Antolick (Ref. 2 and 3), as well as to include a better understanding of error estimation techniques. In an effort to reduce repetition, the authors recommend that the audience read Reference 1 prior to continuing with this paper.

**Literature Review**

The Army machine learning process is presented and built from a *Concept of Operations perspective* by Wade (Ref. 1). The process is comprised of three steps: preparation, training, and finalization. The nomenclature of learning is also standardized for ease of communication across management and engineering teams. This process will be expanded and adjusted for the very specific cases of learning: a fault classification algorithm for a gearbox and a logistics maintenance aid for a turbo-shaft engine.

The effectiveness of an unsupervised learning methodology applied to asynchronous spectrums is investigated by Brower (Ref. 2). The purpose of the study was to identify possible methods of input parameter reduction prior to training a supervised learning model. This input reduction in turn lowers the Vapnik-Chervonenkis (VC) dimension of the model.

Principal component analysis (PCA) was selected as an unsupervised learning technique in an effort to identify fault characteristics without Subject Matter Expert (SME) involvement. This technique eliminates the problem of over-fitting, to which SME-extracted features are susceptible, e.g. condition indicators (CIs). This is particularly important because most or all of the current ground truth data cases were used for CI development. A secondary goal of this approach is the identification of possible characteristic patterns that human analysts are incapable of detecting due to the massive amount of data requiring inspection.

By normalizing the PCA input by the standard deviation, expected vibration components of healthy gearboxes were minimized while true anomalies were amplified. When applied to the test set of gearbox spectra, this methodology was shown to replicate the performance of the top three SME-developed-CIs. The resulting input set of the first 24 principal components resulted in a dataset reduction of 99.7% from the original size.

A clustering unsupervised machine learning technique was investigated by Antolick (Ref. 3) to determine if it was capable of reducing the dataset dimensionality. A clustering technique organizes data that has similar attributes into individual groups. For a HUMS application, the desired outcome of the simplest application of a clustering technique would be to group the data into either a healthy or faulted cluster. The desired outcome of a more advanced application of a clustering technique would be the organization of data into one healthy cluster and several faulted clusters, where each faulted cluster represents a particular fault mode.

The expectation maximization technique as part of a Gaussian mixture model was the initial clustering technique selected for demonstration due to its perceived suitability for a HUMS application. Expectation maximization requires the user to input the number of clusters into which it will organize the data into the specified number of unique Gaussian distributions using an iterative statistical probability likelihood process.

Expectation maximization was applied to a subset of an engine output gearbox ground truth dataset. The dataset used included a fleet-representative proportion of assumed healthy and faulted cases. The faulted cases included ball bearing, roller bearing, gear, and gearshaft faults. Nine SME-selected CIs were chosen as the input data for each ground-truth case. The algorithm was first executed specifying two distributions. The algorithm performed fairly well separating the data into appropriate categories. The technique was also successful in classifying components exhibiting incipient faults and functional failure. It then attempted to classify each fault mode into its unique cluster. This attempt was mostly unsuccessful. It was partially successful due to its ability to separate the gear from the bearing faults, but produced the undesired result of dividing the healthy data into separate clusters. It was determined that including additional input parameters (e.g. torque airspeed and usage time) into the model may provide the additional information needed to separate each fault mode into its own cluster.
Literature Summary

The results of the previous studies (Ref. 2 and 3), including the successes that the Army has achieved (Ref.1) indicate that there is tremendous value in continuing with the machine learning program.

The remainder of this paper will focus on the project that is currently underway within the US Army Aviation Engineering Directorate to improve the HUMS decision-making process using machine learning techniques to incorporate component history, diagnostics, and parametric data. The paper discusses the approaches that will be used to minimize learning error, the methods that will be used to deal with skewed populations, and the issues associated with creation of the N by p matrix for learning.

MEASURING ERROR

Drive train components are almost always healthy. In fact, at any given time, a rotorcraft can perform its mission with a very reasonable assumption that it will have no mechanical faults that would result in a precautionary landing or mission abort. From a diagnostics perspective, the fleet is significantly skewed towards health.

Supervised classification learning algorithms actively reduce misclassification error through an error function (sometimes called a penalty function), e.g. the distance between the hyperplane classification boundary and a point that is misclassified. The great majority of standard machine learning algorithms assume that the populations are of similar size, \( N_{\text{healthy}} \approx N_{\text{faulted}} \). Manned air vehicles are inherently skewed toward health, so much so, that to train a classification machine using the population distribution and the standard error function would likely result in a machine that always guesses that the aircraft is healthy. The authors have therefore focused on defining the best default methods for measuring misclassification error during training such that the machine does not need to just always guess healthy in order to minimize the error function. In this section, the authors will offer a simulation of what happens when the standard error function (which treats all misclassification error equally) is used on a skewed population and compare it to four methods for adjusting the learning process to achieve the required results. The relationships between the error methods described in the next sections are shown in Figure 1.

There is a limited amount of literature in machine learning regarding methods for training machines using skewed classes. The references are generally specific to the industries from which they were developed (Ref. 4).

![Figure 1. Summary of the four error methods recommended for use on skewed aircraft populations.](image-url)
Method 1: Normalized Error for Skewed Classes

The authors recommend that the classes be normalized by their size when the error function is computed so that the error measured on the dense class is given equal weight to that measured on the sparse class. This is a similar practice to over-sampling the smaller class. This takes the basic form shown in equation 1. The total error that the learning algorithm must reduce, \( Err(g) \), is the sum of the normalized error from the healthy data plus the normalized error from the faulted data.

\[
Err(g) = \frac{\sum_{i=1}^{N_{\text{healthy}}} Err(g(N_{\text{healthy}}))}{N_{\text{healthy}}} + \frac{\sum_{k=1}^{N_{\text{faulted}}} Err(g(N_{\text{faulted}}))}{N_{\text{faulted}}}
\]  

(1)

In Figure 1, box 1, a flow is shown for computing the Method 1 error. During training evolutions, the data are separated into healthy and faulted. Each data point is compared to the decision boundary learned by the machine to identify misclassifications. The total number of misclassifications and their distances from the boundary are recorded. The total number of misclassifications are summed and normalized by the appropriate population size (healthy population for false positives and faulted population for the false negatives). The information is sent back to the machine and used to create a new evolution of the boundary using the same hypothesis. This is iterated until the minimum number of misclassifications is found.

Method 2: ROC Curve Penalty

Aeronautical Design Standard 79D (ADS-79D) (Ref. 5) defines two applications when computing error for diagnostics:

1. **Airworthiness Applications** require 6-nines of reliability. For this case, ADS-79D dictates that the True Positive Rate (TP) be greater than 99.9999% and the False Positive (FP) Rate be less than 5%.

2. **Advisory Applications** have only guidelines in ADS-79D. For this case it is recommended that the objective True Positive Rate be greater than 99%, with a threshold of 90%, and the objective FP Rate be less than 5%, with a threshold of 15%.

These two cases can be represented as Receiver Operating Characteristic (ROC) curves, and summarized by Tables 1 and 2. The authors have converted the performance criteria (percentages) into weight penalties caused by false diagnostic outputs.

These penalties can be used to multiply the error measured during training. As an example, the performance criteria of 99% TP rate is equivalent to 1% False Negative (FN) rate, which can be expressed as 1 out of 100. If the machine classifies a component as healthy, but it is actually faulted according to the ground truth data, the penalty for this misclassification would be 100.

As another example, the performance criteria of 5% FP rate can be expressed as 5 out of 100 then simplified to 1 out of 20. If the machine classifies a component as faulted, but it is actually healthy, the penalty for this misclassification would be 20. In this manner, the two types of error (FN and FP) can be expressed as a penalty (multiplier) based on the performance criteria; one just needs to convert the proportions to whole numbers that preserve their relationship according to the ROC curve.

**Table 1. Table of Penalties for Airworthiness Applications of Machine Learning Algorithms.**

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Healthy</th>
<th>Faulted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnostic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>TP: 0</td>
<td>FN: 1,000,000</td>
</tr>
</tbody>
</table>

**Table 2. Table of Penalties for Advisory Only Applications of Machine Learning Algorithms.**

<table>
<thead>
<tr>
<th>Ground Truth</th>
<th>Healthy</th>
<th>Faulted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnostic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Output</td>
<td>TP: 0</td>
<td>FN: 100</td>
</tr>
</tbody>
</table>

In Figure 1, box 2, a flow is shown for computing the Method 2 error. During training evolutions, the data are separated into healthy and faulted. Each data point is compared to the decision boundary learned by the machine to identify misclassifications. The total number of misclassifications and their distances from the boundary are recorded. The total number of healthy misclassifications are summed and then multiplied by the FN penalty. The total number of faulted misclassifications are summed and then multiplied by the FP penalty. The information is sent back to the machine and used to create a new evolution of the boundary using the same hypothesis. This is iterated until the minimum number of misclassifications is found.
**Method 3: Compound Error**

Methods 1 and 2 can be combined to create a compound method. First normalize the error, as in Method 1, and then penalize the error based on the Method 2 criteria. This is shown in equation 2 where $R_{FN}$ and $R_{FP}$ are the respective FN and FP multipliers from the appropriate application table above.

$$Err(g) = R_{FP} \cdot \frac{\sum_{i=1}^{N_h} Err(g(N_h,i))}{N_{healthy}} + R_{FN} \cdot \frac{\sum_{k=1}^{N_f} Err(g(N_f,k))}{N_{faulted}}$$

(2)

In Figure 1, box 3, a flow is shown for computing the Method 3 error. During training evolutions, the data are separated into healthy and faulted. Each data point is compared to the decision boundary learned by the machine to identify misclassifications. The total number of misclassifications and their distances from the boundary are recorded. The total number of misclassifications are summed and normalized by the appropriate population size (healthy population for false positives and faulted population for the false negatives). The information is sent back to the machine and a heuristic rule that balances the error is used to shift the boundary toward the desired TP and FP rates. The shifted model is reported. If the model is unable to achieve the desired rates, a new hypothesis must be generated.

**Simulation**

Use of machine learning on aircraft safety data is a relatively nascent concept and as such, the authors needed to generate an appropriate method for handling error that is aerospace-specific. The previous sections describe the methods that the authors intend to use moving forward with this program. As part of this process, it is a good exercise to understand how these techniques work from both a theoretical and practical perspective. In order to prevent biasing of the gearbox and engine dataset described later in the paper, the authors have put together a simulation of a skewed, linearly inseparable classification problem to demonstrate how the methods described in the previous sections would function.

The dataset used to demonstrate these methods is a mixture of two Gaussian distributions, with slightly different means and variances. For the purposes of this simulation, the data for the orange class was sampled at 2% the size of the blue class, as illustrated by Figure 2. A linear Support Vector Machine (SVM) generated the four boundaries shown in Figure 2. The solid line represents a boundary generated using the default error method. In this case, the SVM learns a decision function that assigns all data to the blue class. The dashed line represents a learned boundary using Method 1, class normalization error. In this case, the boundary has a 2% FN rate and a 9.8% FP rate. Both the dotted line and variegated dotted line represent learned boundaries using Method 3 (compound error). The difference between these two lines is the performance criteria, the dotted line is the result of using the Advisory Criteria and the variegated line is the result of using the Airworthiness Criteria. A ROC curve table is presented in Table 3 that summarizes the results.
Figure 2. Comparison of decision boundaries created by changing the error method on a skewed population. The blue group is similar to the healthy population in an aerospace application and the orange group is similar to the faulted population.

Table 3. ROC Curve table for four decision boundaries learned by an SVM using different error methods.

<table>
<thead>
<tr>
<th>Weight Scheme</th>
<th>% False Negatives</th>
<th>% False Positives</th>
<th>% Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>100.0</td>
<td>0.0</td>
<td>98.0</td>
</tr>
<tr>
<td>Class Norm</td>
<td>2.0</td>
<td>9.8</td>
<td>90.4</td>
</tr>
<tr>
<td>Class Norm with Advisory Penalty</td>
<td>0.0</td>
<td>14.3</td>
<td>85.9</td>
</tr>
<tr>
<td>Class Norm with Safety Penalty</td>
<td>0.0</td>
<td>16.2</td>
<td>84.1</td>
</tr>
</tbody>
</table>

A simulation of Method 4, class normalization with performance heuristics, is shown in Figure 3. The normalization is used to create a boundary, as was done in the dashed line of Figure 2. A gray scale is used to demonstrate the heuristic boundary moving across the dataset so that desired performance can be achieved. Shades of gray represent shifting boundaries of the same SVM model. The ROC curve representing the best performing boundary is shown in Figure 4. Notice that the ROC curve approaches the desired performance goals of 99% TP rate and 10% FP rate for the Advisory case for the training data. A test dataset was held back for this simulation. The test set follows the performance of the training set very well.

Figure 3. Example of Error Method 4, where the linear SVM is trained using class normalization, then a boundary based on the learned function is assigned based on the required performance.

Figure 4. ROC Curve for the heuristic boundary shift example.

The error performance can be significantly improved when a non-linear decision boundary is created. A final simulation using a Neural Net on the same data set using Method 4 is shown in Figure 5. In this case gray scale decision boundaries are shown again, except that they are non-linear contours that are not equidistant. The heuristic method applied to this non-linear model provides excellent results, as shown in the ROC curve in Figure 6.
The outputs of the training step shown in Figure 7 give engineers the first opportunity to estimate $E_{out}$ using the error predicted by the VC Dimension in equation 3, where $N$ is the number of data points, $d_{vc}$ is the VC dimension, and $\delta$ is 1 minus the confidence bound. Use of this expression and the error methods shown above will result in a decision point to complete the Training Step and move on to the Finalization Step shown in Figure 7 which starts with a milestone event, opening the Pre-Test Data Vault.

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{\delta}{N} \ln \left( \frac{4(2N)^{d_{vc}+1}}{\delta} \right)}$$

Here, $E_{out}$ is the unknown, out-of-sample error, which is bounded by $E_{in}$, the in-sample error. The in-sample error is the error on the training set. Using the pretest methodology, the error at the milestone event is also bounded by the Hoeffding Inequality (equation 4 below), where $E_{in}$ is the error on the pretest set.

Measurement of $E_{\text{vect}}$, Figure 7, is governed by the Hoeffding Inequality in equation 4, where $\epsilon$ is the acceptable error bound and $N$ is the number of samples. Simply stated, the inequality gives a probability that the measured test performance and the expected general performance of the learned function are within a given error bound (Ref. 6). From these measurements of error, additional learning is possible, specifically the ranking of the models and re-learning on the best model. The final decision utilizes the same error procedure, except that the Hoeffding probability and the error bounds are reported to the customer and the prototype algorithm is complete.

$$\Pr[|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2N}$$

It is important to note that the Hoeffding Inequality is a much tighter bound on the error than the VC-dimension inequality. This is the reason for using a test set, as it allows models of sufficient complexity to learn the unknown classifier function, while still enabling a reasonably tight bound on the reported expectation of out-of-sample error.
BUILDING A FOUNDATION

The engineering team for the subject program has been working for nearly six months to create the foundations for accomplishing two prototype algorithms that can be fielded to make decisions regarding the health status of engine output gearboxes and turbo shaft engines. The Preparation phase of generating the N by p matrix is completed. The next phase, which has not been completed, is to assign appropriate rows to the Training Data, Pre-Test Data Vault, and Test Data Vault. This section of the paper will be focusing on the process used to generate the N by p matrix. It is important to note that there are a large number of considerations when building this matrix and thus the team has gone to great efforts to document the assumptions and simplifications used in this process.

Generating the Engine Output Gearbox Dataset

The goal of the gearbox machine learning program is to take the available data concerning the gearbox, and determine three pieces of information: Is the gearbox healthy or faulted? If it is faulted, then (a) how severe is it, and (b) how much longer can it be used? The initial data set needs to focus on the first of these three questions, is the gearbox healthy or faulted? The dataset must meet the following two important criteria:

1. Representative distribution. The dataset must reflect the probability distribution being modeled, which must be selected with care to represent the true problem being modeled.
2. Consistent formulation. The dataset must have a set of variables that are meaningful for most or all of the examples. (A bad example would be the measurement of CI X within one flight hour of removal, which due to various issues is not possible to define for more than a small fraction of the fleet; a good example would be the last known magnitude spectrum before the removal.)

The next section outlines the issues with the available data.

Available Gearbox Data Issues

Defining, acquiring, building, cleaning, and organizing this data set is complicated by several problems. The underlying data comes from several different sources: the HUMS records vibration data;
the flight computer records the bus data; the reliability database contains teardown analyses; and the maintenance database contains the component removal/replacement records. These data sets have different standards of reporting, different transmission requirements, and different timing. This leads to two primary problems: missing data, and uncoordinated data.

Further, the dataset is sparse. Mechanical failures are rare in aviation systems, which is good for users but bad for statisticians. There are few examples of faults, and even fewer examples of ground-truth healthy (parts which were removed, torn down, analyzed, and shown to be airworthy, after having flown and data being acquired).

The problems with missing data, uncoordinated data and sparse data will complicate nearly every stage of the dataset creation.

Representative Distribution

In this case, the distribution being modeled is the probability that a gearbox is faulted, given a recent HUMS vibration measurement of that gearbox. For the nose gearbox, this distribution was interpreted as being approximately represented by the fraction of faulted gearboxes at any given time in the fleet, about 3%. Therefore, the dataset must have 97 healthy examples for every 3 faulted examples.

Both the sparse data problem and the missing data problem were encountered when building the gearbox dataset. Of the approximately 40 ground-truth (teardown) examples, only a few (<5) are known to be healthy, and several (>8) have no available HUMS vibration data. Without discarding nearly all the faulted data, another source of healthy data was required to complete the distribution.

Since there was an insufficient number of ground truth health examples, additional data was found that could be presumed healthy. By selecting data from a component during a period 50 hours after installation but 200 hours prior to removal, as was done by Brower (Ref. 2), there is high confidence that the data represents a healthy component. This expands the available ground truth examples out to the required number, approximately 1,000.

Consistent Formulation of Data

To generate a function which receives realistic input and generates an answer of healthy or faulted, the available data must be gathered and formatted into a set of inputs, one for each example, along with known or presumed truth state (each example is labeled healthy or faulted). With N examples and p inputs, the total dataset is represented by a matrix of N rows and p columns. The input columns, the variables with a measurement from each example, must each be something that can be defined for every example.

Data from each of the data streams (teardown analyses, HUMS records, and bus data) has its own time stamping protocol, making it difficult and sometimes impossible to construct meaningful data groups for all the gearboxes. Further, different models of aircraft have different levels of data reporting making it difficult to relate the histories of each component. These are the steps performed to fuse the data into the most complete, consistent and coordinated data matrix possible. For each ground truth example gearbox:

1. Select the valid history start/stop times.
   a. For each teardown analysis, choose the time from installation until removal.
   b. For presumed green examples, choose the time from 50 hours after installation until 200 hours before removal.

2. Collect all HUMS (vibration/CI) data for the valid history, including the HUMS timestamp. For each CI, compute the statistical change detection (SCD) flag (trend/no-trend).

3. For each HUMS record, collect the closest (in time) flight computer record, if any, and store airspeed, torque and temperature data with the HUMS record. If there is no such record, then indicate as such in the data by using Institute of Electrical and Electronics Engineers Not-a-Number-Format, or IEEE NaN.

4. Group all variables by common timestamp, i.e. the spectrum, its CI values and trend flags, and the bus data nearest it.

5. Group all variables by regime and timestamp. For example, the flat pitch ground regime captures one spectrum and two different synchronous time averages (STAs) with slightly different timestamps; find ordered sets of spectrum, STA1, and STA2 that are close in time and call them a group. If some data cannot be assigned to complete regime/timestamp group, then assign it to an incomplete group and put placeholder values (IEEE NaNs) in the other variable slots.

6. For each variable-length regime/timestamp-group history, summarize the variable-length history
using a set of functions, each of which returns a fixed-length vector.

a. Most recent complete measurement (e.g., a complete set of spectrum, STA1, STA2, and all computed CIs). This results in approximately 10,000 columns: 8,192 for a spectrum, 1,500 for various STAs, and the remainder from CIs, CI trends, and bus/aircraft-meta data.

b. Five most recent (possibly incomplete) measurements. This results in approximately 50,000 columns.

c. Statistical summary values of each scalar CI history, including mean, median, max, min, and IQR. This results in fewer than 1,000 additional columns.

To provide additional important information, this N by p' matrix is accompanied by a table of information describing each row (tail number, faulted status, severity color, etc.) and a table of information about each column (CI name, whether the column is CI/trend/metdata/raw, if raw what bin, if history what point of history, what function was used to summarize the data history into the column, etc.)

The resulting N by p' data set was approximately 1000 rows by 62,000 columns. These dimensions may seem excessive, but by including the raw signals (and the histories of the raw signals), individual engineers or learning machines can search for new, better CIs. The data is stored in the HDF5 file format.

**Engine Data Distribution Calculation**

To calculate the engine healthy-to-faulted distribution to be used for machine learning, the approach taken was to calculate the probability of an engine being removed for a low power/low torque (LPLQ) condition given that it was just operated. Figure 8 shows the process used to calculate this metric.

The reader should note that this approach is different from the one used for the engine output gearbox since on-wing condition was not considered prior to removal. Only probability of legitimate removal for an LPLQ condition was considered.

In Figure 8, the maintenance database, the HUMS database, and the reliability database are represented. The maintenance database pulls together the Army’s rotorcraft flight and maintenance records. It is used to obtain flight hours and removal records for the turbo-shaft engines to be used in the machine learning process. The HUMS database provides the turbo-shaft engine parametric data. The reliability database houses Army rotorcraft drivetrain reliability information based on drivetrain component teardowns. In the reliability database, turbo-shaft engines experiencing legitimate LPLQ removals were queryable.

Notes and assumptions from the LPLQ probability calculation process are discussed below:

1. **Fleet Size:**
   a. **Assumption:** The 2,000 aircraft fleet estimate was adequate given the turbo-shaft engine model analyzed with available HUMS recorded parametric data. This number is loosely based upon metrics documented over the past decade. Fleet size is known to fluctuate due to new aircraft, and aircraft retirements, etc. Exact fleet size for the historical period of confirmed LPLQ removals would be difficult to obtain.

2. **Maintenance Database:**
   a. Monthly flight hours were averaged for a sample of aircraft over the period for which confirmed LPLQ removals were available.
   b. The Monthly average was first calculated for the aircraft, then the fleet.
      i. Months with no flight hours (i.e., aircraft in PMI, RESET, etc.) were included in aircraft averages.
   c. **Assumption:** The sample is representative of the fleet.

3. **HUMS Database:**
   a. The HUMS database contained data on 1,581 aircraft obtained over last 8 months. HUMS data is not available on all fleet aircraft due to several potential issues: the HUMS may be inoperable or not be installed on the aircraft, data may not be transmitted to the necessary servers, or an aircraft may be inoperable during the time period of interest.
   b. Per aircraft operation, the mean time delta between startup and shutdown of both engines was used for average run length calculation.
The probability of an engine being red (i.e., removed for a valid LPLQ condition) proved to be extremely low (1.8E-4) when calculated by the process shown above. To obtain a ratio of red-to-green engine operations equal to 1.8E-4, an unobtainable number of known green operations would be necessary to offset the verified red cases identified through the Reliability Database. Since the Army is not able to verify green engine data for that number of operations, as many verifiable green operations as possible were collected and assumed sufficient for the Engine dataset. Parametric data for the green and red operations were stored in the HDF5 files format.

**HDF5 Format construction**

Parameters recorded from bus data for aircraft operations were put in the HDF5 files for use in the Engine learning and testing processes. These parameters included the following for each given engine:

1. Outside Air Temperature (OAT)
2. Turbine Gas Temperature (TGT) of the respective engine
3. Torque of the respective engine
4. Compressor Speed (NG) for the respective engine
5. Power Turbine Speed (NP) for the respective engine
6. Anti-Ice for the respective engine
7. Indicated Airspeed (IAS)
8. Barometric altitude
**HDF5 Validation**

Three green HDF5 files and a single red file were constructed to contain the Engine dataset in the format shown in Figure 9.

After the HDF5 files were generated, the following checks were performed to ensure that the data stored in the HDF5 files did in fact match the data that it was intended to replicate:

1. Ensure that regimes documented are accurate and aligned for given operations
2. Ensure that invalid points are eliminated from parameter vectors in the HDF5 file
3. Ensure that HDF5 parameter down sampling is performed correctly
4. Ensure that the parameters from the correct side are stored in the HDF5 file
5. Ensure that each of the HDF5 files links correct parameters to the correct operation

Figure 10 demonstrates how checks 2-5 were evaluated against a gas generator parametric data stream sampled during the beginning of an operation. The blue data were exported from the HUMS database for the operation without filtering out invalid data points. Red data points were exported with invalid data.
point filtering on. The gray points are the equivalent down sampled data stream stored in the HDF5 file.

Figure 10. Gas generator parametric data from the HDF5 file plotted against equivalent data directly from the HUMS database.

CONCLUSIONS

This paper has completed the foundational work that is necessary to start a machine learning process for two aircraft components: engine output gearbox, and turbo-shaft engine. The paper has outlined the error methods that will be used for training and the data cleansing and vaulting process. The data is now in a state that can be used for in-sample training.

FUTURE WORK

The foundations for fusing the Army datasets associated with component health and history (HUMS, Bus Data, Maintenance Data, and Reliability Data) have been created so that machine learning techniques can be implemented. These foundations of process control, error estimation, performance criteria, and N by p’ setup are critical to the success of the program. Generating this documentation prior to proceeding with unsupervised and supervised machine learning will result in a reduction of model bias and variance.

This foundational work will now be used to generate specific machines for determining the health status of engine output gearboxes for advisory generation. The goal of the health status of the gearboxes is to provide the project office with an immediately usable prototype that works within the confines of the Army maintenance structure. This work will also be used to generate a logistics algorithm that predicts the possibility that a LPLQ engine will need to be replaced or repaired. The goal of the engine LPLQ project is to provide the project office with an algorithm that functions within their own infrastructure and can be used to provide spare parts when needed.

REFERENCES


