**Using Neural Networks for  
Aerospace Health and Usage Monitoring Systems (HUMS) Applications**

Ryan Patton

SysEng5212 Spring 2020

May 8, 2020

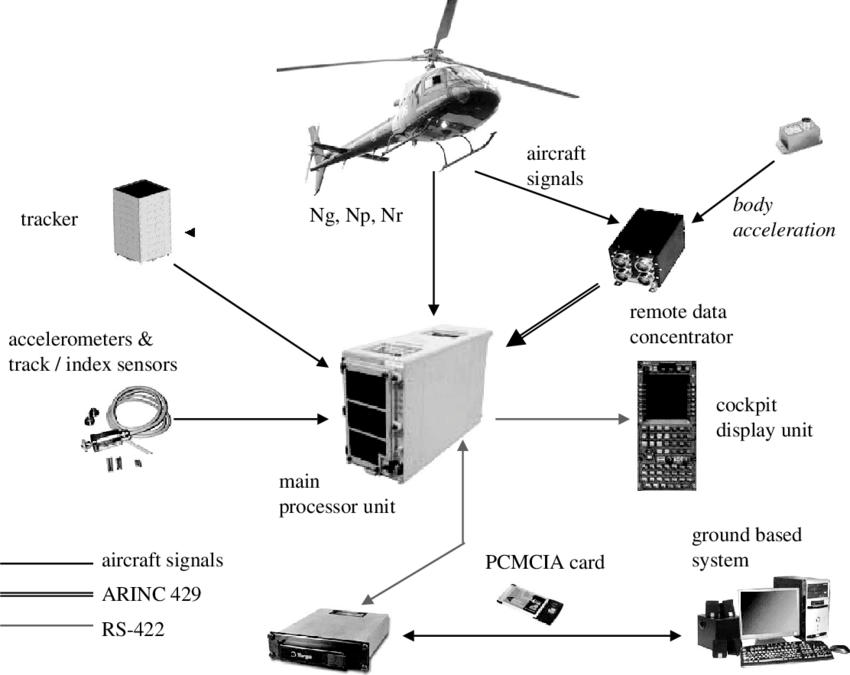


Table of Contents

[1. Introduction 3](#_Toc39823672)

[2. Approach 3](#_Toc39823673)

[2.1 Front End Pre-Processing 4](#_Toc39823674)

[2.2 Neural Networks in Trendability and Health Indictors 5](#_Toc39823675)

[3. Performance Results 6](#_Toc39823676)

[3.1 Performance Evaluation 6](#_Toc39823677)

[4. Conclusions 8](#_Toc39823678)

[5. References 10](#_Toc39823679)

List of Figures

Figure 1. Rotorcraft Sensor Locations (www.curtisswrightds.com) 3

[Figure 2. Ensemble Training 4](file:///C:\Users\Jim%20Smith\Documents\MST\SysE%20378\Project\Jim%20Smith%20SysE378%20Final%20Report.docx#_Toc229138439)

[Figure 3. Operational Settings’ Plot Clusters 4](#_Toc229138440)

[Figure 4. K-means Plot Centroidal Clusters 4](#_Toc229138441)

[Figure 5. Normalized Sensor Measurements 5](#_Toc229138442)

[Figure 6. Top Trending Sensor Selections 5](#_Toc229138443)

[Figure 7. Linearly Degrading Sensor Health Indicators 5](#_Toc229138444)

[Figure 8. Training Data Fused – Single Health Indicator 5](#_Toc229138445)

Figure 9. Validation Data Fused - Single Health Indicator…………………………………………...5

Figure 10. Neural Networks Architecture………………………………………………………………6

Figure 11. 50% Truncation………………………………………………………………………………6

Figure 12. 70% Truncation………………………………………………………………………………6

Figure 13. 90% Truncation………………………………………………………………………………6

Figure 14. RUL Estimation - 50%..................................................................................................7

Figure 15. RUL Estimation - 70%..................................................................................................7

Figure 16. RUL Estimation - 90%..................................................................................................7

Figure 17. RUL Prediction Error of Each Validation Ensemble Member…………………………...7

Figure 18. RUL Prediction Error – Box Plots at Truncation % Quantiles…………………………...7

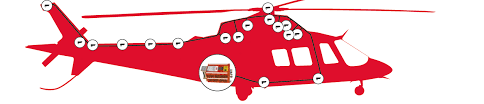
Figure 19. Mean Prediction Error with Standard Deviation……………………………………….....8

List of Tables

[Table 1. Listing of Project Files 1](#_Toc229138446)1

# Introduction

The emphasis placed on neural networks in recent years is justifiable, as their potential skyrockets with each innovation at the cross-section between algorithm development, big data management, and computing power. In the aerospace industry, neural network applications are advancing flight data management, composite structures, aircraft fleets in the systems-of-systems context, health and usage management systems (HUMS), and prognostics and health management (PHM) [6]. In terms of HUMS, neural networks reflect a coordinated effort to analyze massive amounts of data in real-time to examine certain component’s likelihood of failing during flight or requiring maintenance at the next grounding instance. The content of this paper examines the application of neural networks to predict malfunctioning or risk-posing parts’ likelihood of failing by evaluating turbofan engine degradation data with known instances of failing parts.

Multiple applications exist for predicting failing parts in the aerospace industry, however, most applications fixate on rotorcrafts as they show the highest likelihood for unpredictable failure [7]. Figure 1 shows commonly located sensors on a rotorcraft to identify failing parts. Modern HUMS applications are slowly transitioning from data analysis upon grounding to the analysis of sensor measurements in real-time for active flight communication [4]. Although rotorcrafts retain the highest likelihood of failing with perilous results, HUMS applications extend to fixed-wing aircraft, satellites, unmanned aerial vehicles, drones, rockets, and other various aerospace vehicles [6].

The project described in this paper attempts to implement a method for trending health indicators from sensor measurements. The data is taken from a turboprop engine with different operational settings and 20 sensor measurements dispersed throughout the engine [2]. After normalizing the data and discovering top sensors with the highest likelihood of failing, the data will be analyzed in MATLAB to come up with estimates for the remaining useful life (RUL) of each sensor. By producing RUL estimates, a rough estimate can be baselined for each parts chances of failing so preventive measures may be taken.

*Figure 1. Rotorcraft Sensor Locations (www.curtisswrightds.com)*

# Approach

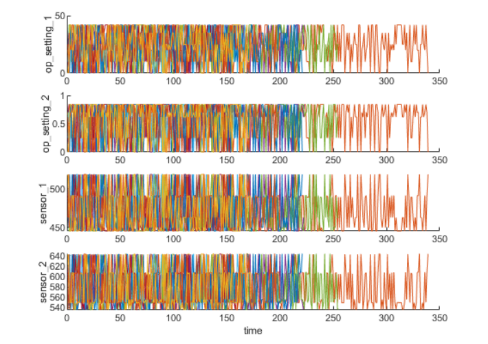
The approach utilized training and test datasets of turboprop engine degradation data, and then attempted to develop and train a MATLAB-based neural network in the deep learning toolbox. Obtaining a comprehensive dataset of a fixed-wing aircraft, or any aerospace vehicle, was prohibitive due to the proprietary and classified nature of the data [1]. Some data sources considered were lithium-ion battery degradation, gearbox fault detection, bearings fault detection, capacitor electrical stress levels, and small satellite power levels. Eventually, the turboprop engine degradation dataset was chosen because of its immediate application to a fixed-wing aircraft’s performance and because of the readily available community support for the dataset [10]. The dataset was pulled as the best option from <https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/>; National Aeronautics Space Agency (NASA) appears to be the only organization readily providing datasets for public use [2] [3]. On top of the turbofan engine degradation simulation dataset, NASA currently hosts a challenge for another similar dataset called Prognostics Health Management (PHM) 08 dataset where the true RUL values are not revealed, expecting the user to develop their own algorithms used in training and test sets provided in the package and to upload results for validation [3]. While the PHM08 dataset may be used for future study, the turbofan engine degradation simulation dataset provided a purposeful jumping off point for studying HUMS applications with verified results. The neural networks may assist with repeated, varying initial conditions, however, a helper function helps set those based on past results to avoid redundancy in the architecture.

The dataset contains 26 columns of data with 1000’s of rows of data per set. The first column represents unit number, the second column represents time, in cycles, the third through fifth column represent operational settings, and the sixth through twenty-sixth columns represent sensor measurements. Each row of data is taken from a single operational cycle. The data may be downloaded as text files with no identification (ID) for each column other than a separate text file giving context to the problem. Four total datasets are pre-divided into training and test sets so some may be utilized in the health indications and some in the RUL calculations. The dataset is a multiple multivariate time series where each time series is from a different engine where each engine used initializes with unknown wear.

## Front End Pre-Processing

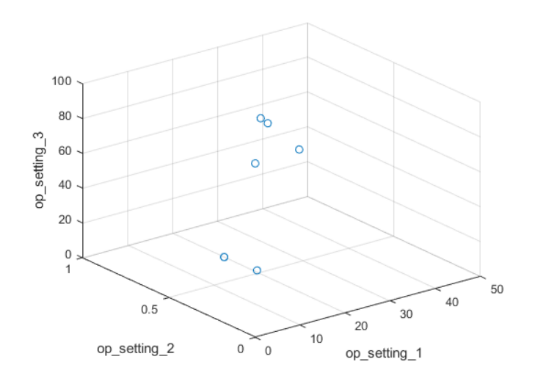
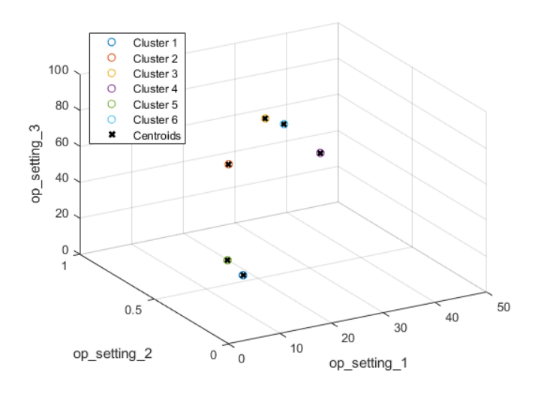
The pre-processing steps prepare the data to be trained and tested. Prior to implementing the neural network, the data is normalized relative to the operational settings by using K-means clustering, evaluating each sensor measurement along the ways to compare its centroidal fit in the clusters. The steps to reach trendability analysis, health indicator construction, validation, RUL estimation model break down, and prognostics performance are as follows:

1. Data Preparation
2. Working Regime Clustering
3. Working Regime Normalization

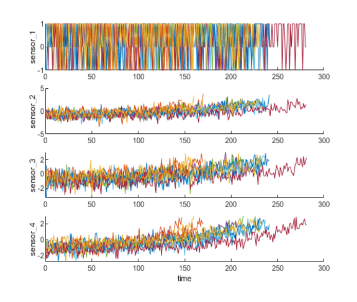
For data preparation, the data from the repository can be downloaded and stories directly to memory. The helper function assists the transition to convert to a cell array of timetables, measuring each of the 200+ simulated failure-to-runs as an ensemble. Each ensemble may be split into a training set and validation set for performance evaluation. For now, the ensemble data is simply visualized in a MATLAB graphics user interface (GUI) shown in Figure 2. From observing Figure 2, no trends yet exist for determining the sensors with the highest likelihood of failing.

For working regime clustering, the data analyzes the sensor measurements by how they tie into the three operational settings to see if it will lead to clearer degradation trends. To do so, a 3-dimensional (3D) scatter plot is appropriated to show 6 regimes clustering at points in the space in close proximity to one another. Figure 3 shows the clustering techniques used to locate the 6 clusters automatically. After identifying the clusters, the K-means algorithm is applied, varying initial conditions along the ways to optimize the cost. The K-means algorithm identifies the 6 working regimes by identifying each cluster’s respective centroid. Figure 4 presents this information.

*Figure 2. Ensemble Training Data*



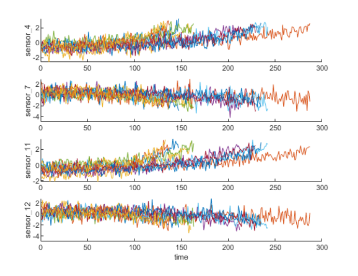
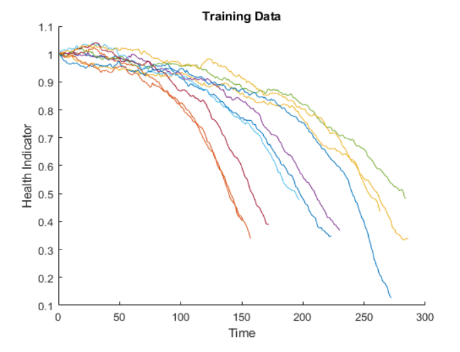
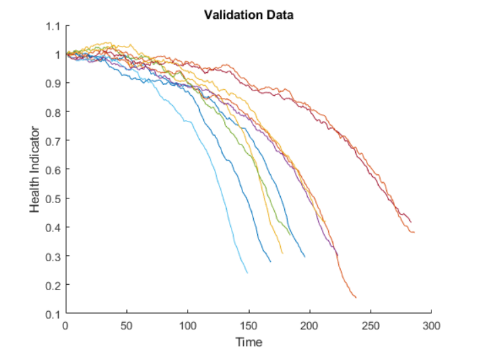
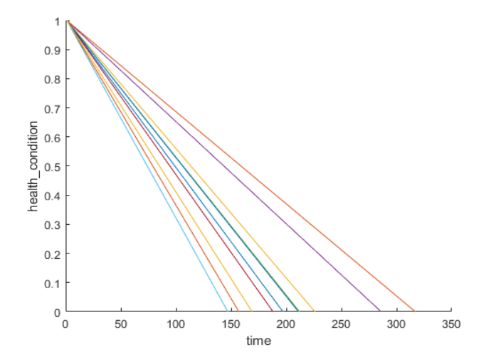
*Figure 3. Operational Settings’ Plot Clusters Figure 4. K-Means Plot Centroidal Clusters*

Now that the data may be grouped by working regimes, the regimes may be normalized. First, the mean and standard deviation for each sensor measurement need computing. Then, the statistics in each regime may be normalized by extracting the operating points of each row, computing its distance to each cluster centers, and finding the nearest cluster center. The mean subtracts from each sensor measurement and divides by the standard deviation for full normalization. When the standard deviation is close to 0, the normalized sensor measurement corrects to 0 because it provides no useful information for RUL estimation. Degradation trends may now be observed in select sensors shown in Figure 5.

## Neural Networks in Trendability and Health Indictors

With revealed trends, the training data proceeds to deduce a health indication for each sensor. The data passes through a linear transfer function (degradation model) and the resulting slopes may be ranked. The top sensors are selected by sorting through the data and visualized as shown in Figure 6. Note the trend improvements as compared to Figure 5 with both positive and negative trends.

*Figure 5. Normalized Sensor Measurements*

Combining all the sensor measurements into a single health indicator means the sensor measurements must fuse together. The linear transfer function is evident in Figure 7, as the health conditions degrade from 1 to 0 over time, with 1 representing complete health and 0 representing failure. The top trending sensor measurements are then used as regressors in a linear regression model. The bias sets to 0.5 and the regression coefficients are set from the sensor measurements. Figures 8 and 9 show the results of fusing the health indicator data after multiplying the sensor measurements by their associated weights, for the training and validation datasets respectively. Producing the fused single health indicator plot for validation data follows the same process as for training data.

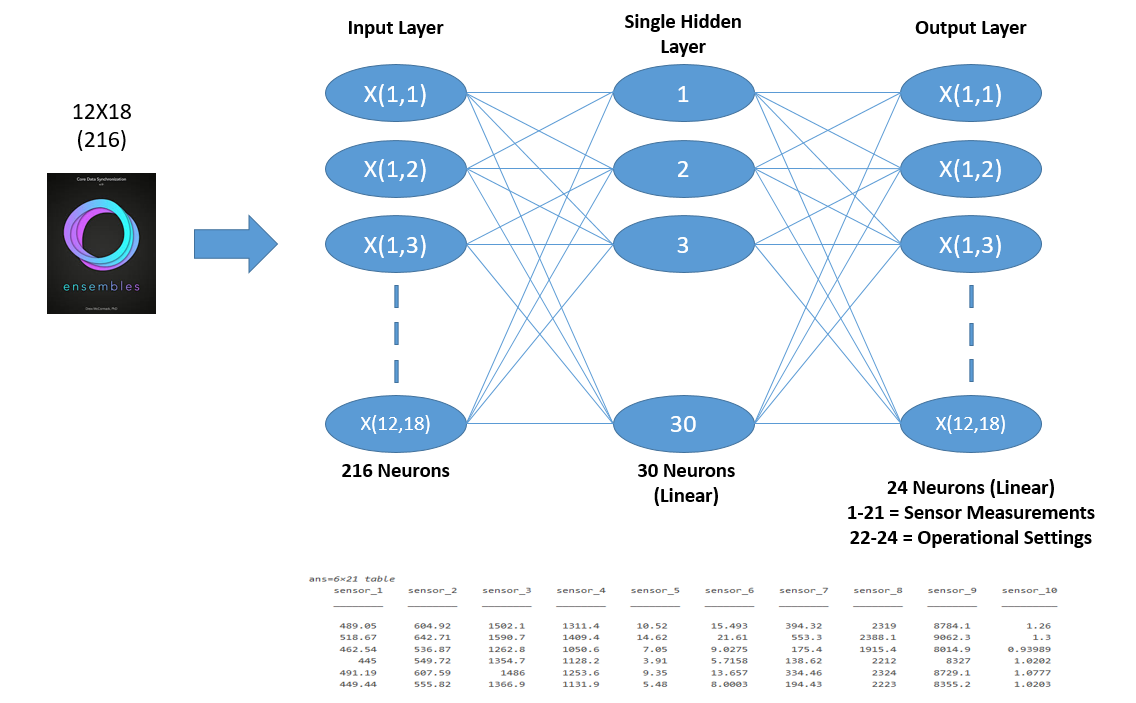
*Figure 6. Top Trending Sensor Selections*

*Figure 9. Validation Data Fused - Single Health Indicator*

*Figure 8. Training Data Fused - Single Health Indicator*

*Figure 7. Linearly Degrading Sensor Health Indicators*

A typical neural network architecture may look like the following in Figure 10 based on 216 run-to-failure ensembles. The single hidden layer helps reduce the complexity of the output layer. Varying neural network inputs may be seen in FinalProjectCode.m in lines 68-102. The architecture compliments a learning rate of 1e-5, a weight initialization based off the sensor regressors, a bias of 0.5, and epochs up to 10,000. The goal is to run the code 5-10 times with varying initial conditions based off of the resultant observations to determine the best cost solution in terms of the learning rate, weight initializations, batch size, number of epochs, etc. For example, epochs would max out around 10,000 instead of 100,000 (arbitrarily chosen value) because the computing cost does not yield significant improvements as the results converge fairly quickly.



*Figure 10. Neural Networks Architecture*

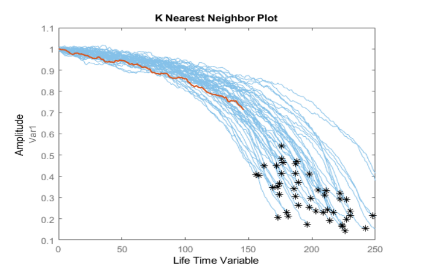
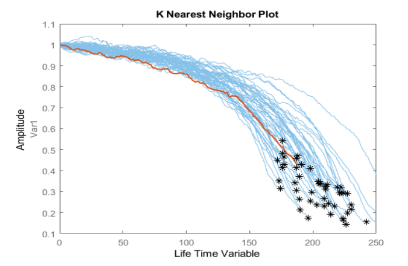
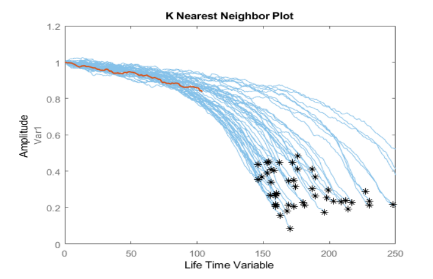
The neural network architecture presented only needs one hidden layer and uses a linear transfer function because the data trends nicely, with expected results.

# Performance Results

Comparing Figure 8 and 9 between the training and validation data, simple observations show similar trends between the pair. After network training, the code may be iterated, switching up the initial conditions in the helper function which will result in different regression values. Various conditions reveal the degradation trends quickly match between the training and validation sets, even at lower numbers of epochs. Due to the datasets containing known failure points, a strong emphasis was not placed on testing performance as much as performance evaluation, especially since general trends matched between training and validation.

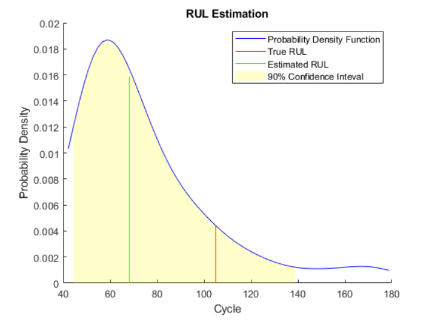
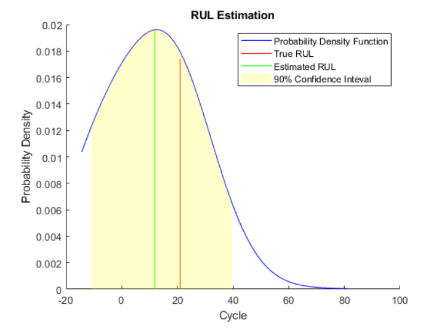
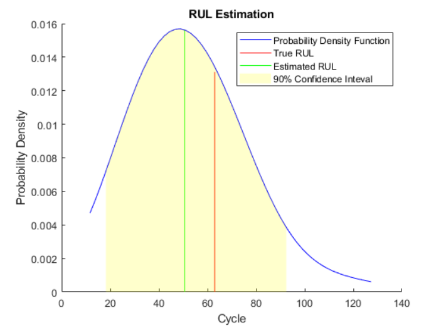
## Performance Evaluation

Fifty ensemble members are sporadically chosen for each one ensemble member in the validation dataset to fit a probability distribution and residual-based similarity RUL model. Each fused dataset fits with a 2nd order polynomial so the median of the distribution may produce an estimated RUL [10]. For performance evaluation, validation data samples of 50%, 70%, and 90% are evaluated to predict its RUL. The validated truncated data at 50% and its nearest neighbors, 70% and its nearest neighbors, and 90% and its nearest neighbors may be seen in Figures 11-13.



*Figure 11: 50% Truncation Figure 12: 70% Truncation Figure 13: 90% Truncation*

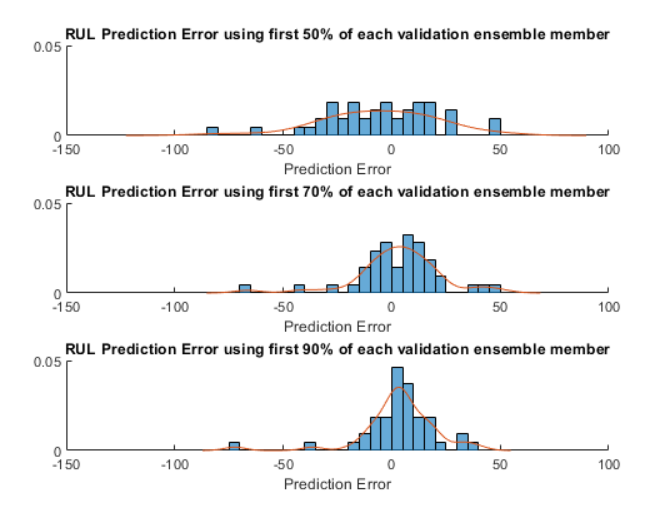
A comparison between the estimated RUL and true RUL may be seen in Figures 14-16, with a probability distribution of the estimated RUL at 50%, 70%, and 90.



*Figure 14: RUL Estimation – 50% Figure 15: RUL Estimation – 70% Figure 16: RUL Estimation – 90%*

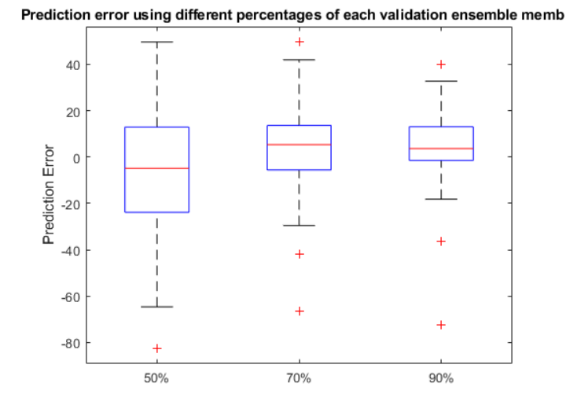
Between the probability distributions seen in Figures 14-16, the gap closes between estimated RUL and true RUL. At 50%, a relatively large error exists between the estimated RUL and true RUL when operating in an intermediate health stage. As failure approaches, the lines split into approximately two modes in the RUL distribution. At 70%, the RUL estimation improves, showing more balanced results. The validation data at 90% of the lifetime yields the best results. When near-failure is reached, the RUL estimation nearly matches the true RUL.

After estimating the validation data at 50%, 70%, and 90%, a repeat evaluation procedure using 100% of validation data may compute the error between estimated RUL and true RUL for each breakpoint. Figure 17 reflects this process.



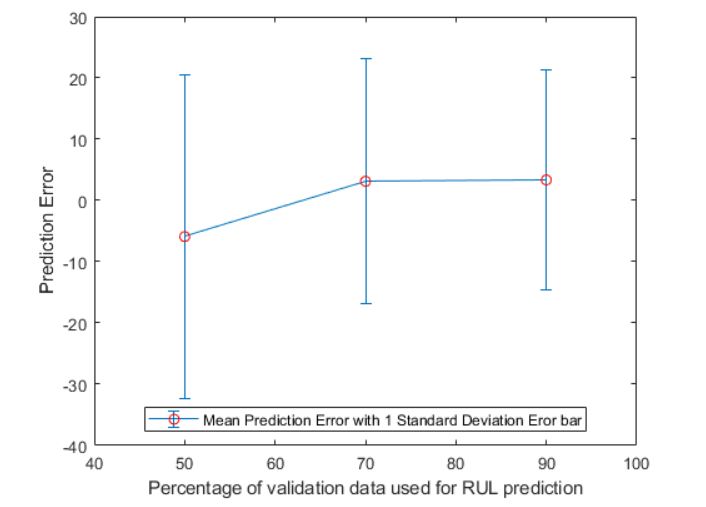
*Figure 17. RUL Prediction Error of Each Validation Ensemble Member*

An alternate plot shown in Figure 18 shows the prediction error as box plots with the media, 25-75 quantile and outliers.



*Figure 18. RUL Prediction Error – Box Plots at Truncation % Quantiles*

The final plot, shown in Figure 19, visualizes the concentration of the error around 0 as more data is added.



*Figure 19. Mean Prediction Error with Standard Deviation*

Since the data does congregate around 0 with more data, improved results can be expected as more data is added. This fact points towards the success of the regression modeling and RUL estimation.

# Conclusions

This rudimentary experiment provides a solid foundation to build upon for future HUMS testing. The comprehensive linear degradation health indicator model feeds successfully into a RUL prediction tool using regression modeling. Neural networks enhance the likelihood of identifying failing parts on an aircraft’s turbofan engine and the relative rate of degradation to other parts on the engine. The sensitivity of the network can be counteracted by multiple runs, feeding in different initial conditions until the best cost solution is reached to move forward into predictions, confident that the end results reflect an imperfectly optimal solution. The fundamental algorithms used, however, show room for improvement. Potential algorithms being explored for their potential in aerospace applications follow [5] [9]:

1. Atypical Flight Segment Identification with Morning Report
2. Inductive Health Monitoring with Clustering and Statistical Methods
3. Anomaly Detection with Density-Based Clustering
4. Anomaly Detection in Nominal Flight Data Sequences
5. Anomaly Detection with Multiple Kernel Function
6. Anomaly Detection with Nearest Neighbor and Symbolic Dynamic Filtering
7. HUMS with Load Cycle Analysis, Component Failure via Sensor network with Engine, Transmission, Vibration Sensors, and Angular Shaft Speed Indicators
8. Flight Operational Quality Assurance Guidelines
9. Frequent Patten Growth Algorithm

At the component level, the above equations reveal better results than the network presented by improving blade health monitoring (in rotorcraft), propeller balancing (in rotorcraft), performance trending, prognostics, gas path debris monitoring, mechanical diagnostics, usage/life tracking, and diagnostics/built-in-tests [8]. At the aircraft level, the above equations reveal better results than the network presented by improving corrosion monitoring, exceedance monitoring, flight regime recognition, individual aircraft tracking, loads monitoring, operational usage monitoring, mission capability assessment, and maintenance management [8]. Finally at the fleet-level, the equations improve fleet trending, force life management, performance metrics, knowledge discovery, and data gathering [8].

A more complete model presented in this paper would require additional pre-processing methods (use different transfer functions other than linear) and would provide more testing feedback rather than simply graphically determining the results are satisfactory for moving to prediction performance evaluation. Also, the data came from operational cycles with unknown engine wear. If the engine wear were to be quantified before predictions, the results may return as real operational values instead of going through normalization to compare their relative failure. The PHM08 dataset shows promise for investigation of more advanced HUMS algorithms and data pre-processing methods for more satisfactory results. More data outputs in the code would lead to higher statistical traceability to troubleshoot exactly where the data does not match as correctly as expected. Further, the results may be limited by the simplifications provided in MATLABs canned deep learning toolbox. Improvements may be obtained by implementation in Python or another language.

HUMS will continue to hold a steadier place in the future of the aerospace industry as defense and commercial companies discover minor HUMS improvements may save them millions of dollars and more importantly, their reputations as worldwide leaders in quality and safety.

# References

1. A. Gavrilovski, H. Jimenez, D. Mavris, A. Rao, S.-H Shin, I. Hwang, and K. Marais, “Challenges and Opportunities in Flight Data Mining: A …,” ResearchGate, AIAA SciTech, 23-Sep-2016, San Diego, CA. [Online]. Available: <https://www.researchgate.net/profile/Arjun_Rao7/publications/290194327_Challenges_and_Opportunities_in_Flight_Data_Mining_A_Review_of_the_State_of_the_Art/links/56955f5aeab58a9a4e4f4/Challenges-and-Opportunities-in-Flight-Data-Mining-A-Review-of-the-state-of-the-Art.pdf?origin=publication_detail>. [Accessed: 15-March-2020].
2. A. Saxena and K. Goebel (2008). “Turbofan Engine Degradation Simulation Data Set,” NASA Ames Prognostics Data Repository (<http://ti.arc.nasa.gov/project/prognostic-data-repository>), NASA Ames Research Center, Moffett Field, CA.
3. A. Saxeena and K. Goebel (2008). “PHM08 Challenge Data Set,” NASA Ames Prognostics Data Repository (<http://ti.arc.nasa.gov/project/prognostic-data-repository>), NASA Ames Research Center, Moffett Field, CA.
4. D. He and E. Bechhoefer, “Development and Validation of Bearing Diagnostic and Prognostic Tools using HUMS Condition Indicators,” *2008 IEEE Aerospace Conference*, Big Sky, MT, 2008, pp. 1-8.
5. D. Wade, R. Lugos, and M. Szelistowski, “(PDF) Using Machine Learning Algorithms to Improve HUMS Performance”, [Online]. Available: <https://www.academia.edu/2695660/Using_Machine_Learning_Algorithms_to_Improve_HUMS_Performance>. [Accessed: 20-Apr-2020].
6. “Health and Usage Monitoring Systems,” Wikipedia, 07-Apr-2020. [Online]. Available: <https://en.wikipedia.org/wiki/Health_and_usage_monitoring_systems>. [Accessed: 25-Apr-2020].
7. H. Yan, X. Liqing, L. Xingwang, W. Ying, and F. Haiyan, “The Investigation and Demand Analysis of HUMS on the Civil Helicopter,” 2017 Prognostics and Health management Conference (PHM-Harbin), 2017.
8. J. Kappas and P. Frith, “From HUMS to PHM: Are We There Yet?,” Semantic Scholar, 2017. [Online]. Available: <https://www.humsconference.com.au/Papers2017/Non_on_USB_Proceedings/191_HUMS2017_Kappas.pdf>. [Accessed: 25-Apr-2020].
9. J.M. Pena, F. Famili, and S. Letourneau, “Data Mining to Detect Abnormal Behavior in Aerospace Data,” *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining – KDD ’00*, 2000.
10. “residualSimilaritymodel,” *MATLAB & Simulink*. [Online]. Available: <https://www.mathworks.com/help/predmaint/ug/similarity-based-remaining-useful-life-estimation.html>. [Accessed: 18-Apr-2020].

**Appendix A: Explanation of Files Used in this Project**

Table 1. Listing of Project Files

|  |  |
| --- | --- |
| **Matlab File** | **Description** |
| FinalProjectCode.m | Main project script; performs training and testing |
| HelperFunction.m | This function assists with regime normalization and fusing the data; need visualization of the clusters and varying initial conditions to the best cost values can be inputted into the function |