TIME TO FAILURE PROGNOSIS OF A GAS TURBINE ENGINE USING PREDICTIVE ANALYTICS

A Thesis

by

LALITH MADHAV PEDDAREDDYGARI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee, Douglas L Allaire
Co-Chair of Committee, Richard Malak
Committee Member, Ulisses Braga-Neto
Head of Department, Andreas A. Polycarpou

August 2020

Major Subject: Mechanical Engineering

ABSTRACT

Maintenance costs and machine availability are the most important concerns of any company that owns large machinery, especially gas turbine engines. With the advent of the 4th wave of the industrial revolution also known as Industrial Internet of Things (IIoT), the focus has shifted onto optimal utilization of the equipment. Reduction in the installation costs of sensors helped companies to install them on their key equipment. The live data from the sensors can now be utilized to monitor the health of the machines. This thesis proposes a prognostic technique to predict time-to-failure of gas turbine engines using standard machine learning and deep learning techniques that puts the data from sensors to good use. Our proposed approach provides accurate feedback on the health of the machine to the concerned personnel. Our approach includes developing multiple Recurrent Neural Network (RNN) models to predict the sensor readings of the engine and then using a Support Vector Machine (SVM) to classify these readings as safe or failure. The objective of this thesis is to establish supporting evidence for the proof of concept of the created time-to-failure prognostic technique. We have demonstrated the performance of our approach on the data-sets made available by NASA with different failure modes and then compared the performance of our approach with the current industry standard for land-based gas turbine engines.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Douglas Allaire and my committee members, Richard Malak and Ulisses Braga-Neto for their continuous guidance and support throughout the course of this research project.

I would also like to express my gratitude to my colleagues, friends, department faculty and staff for making my time at Texas A&M University a memorable experience.

Finally, I would like to sincerely thank my parents and brothers for their continuous encouragement, love and support.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by the thesis committee consisting of Asst. Prof. Dr. Douglas Allaire (advisor) and Assoc. Prof. Richard Malak of the Department of Mechanical Engineering and Professor Ulisses Braga-Neto of the Department of Electrical and Computer Engineering.

All other work conducted for the thesis was completed independently.

Funding Sources

This work was supported by the Air Force Office of Scientific Research (AFOSR) under award numbers FA9550-15-1-0038 (program manager Fariba Fahroo), and FA9550-16-1-0108 (program manager Erik Blasch), and by the National Science Foundation under grant number CMMI-1663130. Opinions expressed in this paper are of the authors and do not necessarily reflect the views of the National Science Foundation.

NOMENCLATURE

C-MAPSS	Commercial Modular Aero Propulsion System Simulation
RUL	Remaining Useful Life
RNN	Recurrent Neural Network
SVM	Support Vector Machine
LSTM	Long Short Term Memory
MRO	Maintenance, Repair and Overhaul
IIoT	Industrial Internet of Things
GT	Gas Turbine
NASA	National Aeronautics and Space Administration
LPC	Low Pressure Compressor
HPC	High Pressure Compressor
LPT	Low Pressure Turbine
HPT	High Pressure Turbine
TRA	Throttle Resolver Angle
GRU	Gated Recurrent Unit
FMEA	Failure Modes and Effects Analysis
ML	Machine Learning

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS vi

LIST OF FIGURES viii

LIST OF TABLES x

1. INTRODUCTION 1

    1.1 Motivation 1
    1.2 Evolution of Industrial Maintenance 2
    1.3 Performance Degradation in a gas turbine 5
        1.3.1 Common physical faults 5
            1.3.1.1 Fouling 5
            1.3.1.2 Erosion 5
            1.3.1.3 Corrosion 6
    1.4 Failure trajectory dataset 7
        1.4.1 Sensors 8
        1.4.2 Damage propagation modelling 8
        1.4.3 Scenario 9
        1.4.4 Failure modes considered 10

2. BACKGROUND 11

    2.1 Introduction to Machine Learning 11
        2.1.1 Terminology 11
        2.1.2 Process 11
        2.1.3 Types 12
            2.1.3.1 Supervised learning 12
            2.1.3.2 Unsupervised learning 12
    2.2 Support Vector Machines 13
        2.2.1 Introduction 13
        2.2.2 History 13

2.2.3 Mathematical formulation 13
2.2.3.1 Statistical Learning theory 13
2.2.3.2 Formulation 14
2.2.4 Implementation 15
2.3 Recurrent Neural Networks 16
2.3.1 Introduction 16
2.3.2 History 17
2.3.3 Standard RNN cell 17
2.3.3.1 Notation 17
2.3.3.2 Formulation 17
2.3.3.3 Loss function 19
2.3.4 Different types of RNNs 20
2.3.5 Vanishing gradients problem 21
2.3.6 LSTM 22

3. METHODOLOGY 25

3.1 Current Industrial Approach 25
3.2 Our Approach 26
3.3 Model to predict engine failure using SVM and RNN 28
3.3.1 Data Preprocessing 32
3.3.2 Model Architecture 32

4. RESULTS 35

4.1 Engine failure prediction model with SVM and RNN 35
4.1.1 SVM model on FD001 35
4.1.1.1 Performance summary of SVM on FD001 35
4.1.2 SVM model on FD003 36
4.1.2.1 Performance summary of SVM on FD003 37
4.2 RNN Model predictions 37
4.3 Prediction of time-to-failure of an engine using our approach 38
4.4 Comparison between our proposed approach and the current industrial approach 40
4.4.1 On the dataset FD001 40
4.4.2 On the dataset FD003 42

5. SUMMARY AND CONCLUSIONS 46

5.1 Contribution 46
5.2 Conclusions and Future work 46

REFERENCES 48

LIST OF FIGURES

FIGURE Page

1.1 World MRO spend by segment in Aviation, 2018 1

1.2 Top avenues for saving in aviation industry 2

1.3 Evolution of Maintenance Strategies 3

1.4 Summary of the strategies 3

1.5 Classification of common faults in gas turbine engines 6

1.6 Maintenance process overview 7

1.7 Simplified diagram of engine simulated in C-MAPSS 8

1.8 Layout of various modules and their connections 10

2.1 Schematic of standard recurrent cell 18

2.2 Schematic of standard recurrent cell (unrolled) 19

2.3 Back propagation through time in RNN 20

2.4 Types of RNN architectures 21

2.5 LSTM with a forget gate 23

3.1 Flowchart for model to predict failure using bounds 26

3.2 Notional depiction of hypervolume bounds for multiple parameter information 27

3.3 Recurrent Neural Network for Time-to-Failure Estimation 28

3.4 Flowchart of Algorithm 29

3.5 Flowchart of the Algorithm (RNN model) 30

3.6 RNN Model used for Sensor prediction 33

3.7 RNN Model architecture used for Sensor prediction 33

3.8 RNN Model used for prediction of rest of the sensors 34

3.9 RNN Model architecture used for prediction of rest of the sensors 34

4.1 Confusion Matrix on the test set FD001 non-weighted classes 35

4.2 Confusion Matrix on the test set FD001 weighted classes 36

4.3 Results of SVM model on FD001 36

4.4 Confusion Matrix on the test set FD003 non-weighted classes 36

4.5 Confusion Matrix on the test set FD003 weighted classes 37

4.6 Results of SVM model on FD003 37

4.7 Predictions of Temperature, Pressure, Bypass ratio and Core speed using RNN models 38

4.8 Our approach on Engine 81 of FD001 39

4.9 Failure prediction of engine 81 using our proposed approach 39

4.10 Comparison between our proposed and the current industrial approach on engine 81 of FD001 40

4.11 Comparison between our proposed and the current industrial approach on engine 82 of FD001 41

4.12 Comparison between our proposed and the current industrial approach on FD001 ... 41

4.13 Comparison between our proposed and the current industrial approach on engine 81 of FD003 42

4.14 Comparison between our proposed and the current industrial approach on engine 82 of FD003 43

4.15 Comparison between our proposed and the current industrial approach on engine 83 of FD003 44

4.16 Comparison between our proposed and the current industrial approach on FD003 ... 44

5.1 Sensor failure detection using multiple parameter information exploitation 47

5.2 Sensor predictions to identify the modes of failure..... 47

LIST OF TABLES

TABLE Page

1.1 Different sensor variables in the Data set 9

1. INTRODUCTION

1.1 Motivation

In this competitive business world, every company expects the machinery to run without any unexpected breakdowns because equipment failure leads to significant wastage of resources and potentially affects the company economically. These failures may also lead to the injuries of the operators and in extreme cases, fatalities. This is the reason why the maintenance of machinery is of high importance in any company across any industry. Any machine that has moving parts is subject to wear and tear and hence requires appropriate maintenance.

Gas turbines are one of the most expensive pieces of equipment that are used both in industrial and aviation applications. Reliability and availability are the most sought out qualities in Gas turbines and a lot of money is invested globally for the operation and maintenance of these engines [1]. This trend is expected to continue given their importance in the respective applications. Globally, $ 69 billion was spent on Maintenance, Repair and Overhaul (MRO) in the year 2018 alone excluding overhead according to International Air Transport Association. Out of which 42% was spent on the engine maintenance alone. With 4.1 % increase per annum, it is estimated to reach $ 103 billion by 2028 [2]. Hence continuous monitoring and maintenance of these critical components is given high importance by companies.

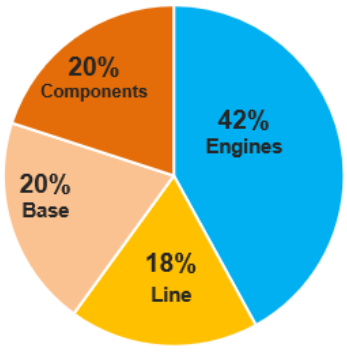

Figure 1.1: World MRO spend by segment in Aviation, 2018

Running these engines at the optimal working conditions is also very important because companies do not want their engines to consume more fuel as it reduces their profit margin and there is an increased probability of wear when running at suboptimal conditions. Therefore, operating these engines at their clean operating conditions does have a significant contribution in reducing the operating costs of these engines. This can be achieved by incorporating an improved maintenance policy and utilizing the latest advances in predictive analytics for the health condition monitoring of these engines.

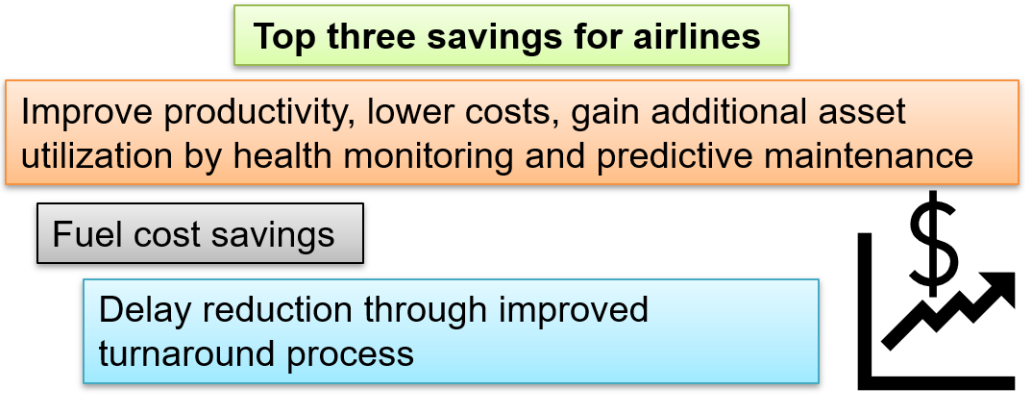

Figure 1.2: Top avenues for saving in aviation industry

In this work, we propose a model that utilizes the latest machine learning and deep learning techniques to predict the time to failure of gas turbine engines. We believe that the incorporation of this model in the existing maintenance policy would help the company in utilizing its resources to their complete potential which will, in turn, reduce the operational and maintenance costs increasing the profitability of the company.

1.2 Evolution of Industrial Maintenance

The First Industrial Revolution (18th Century) resulted in the shift to production using machines. Back then, the handling of the machinery was fairly simple i.e. use it till it fails and repair/replace it only when it no longer runs. This can be called reactive maintenance and it was

not planned as the downtime back then was not critical. With the introduction of electricity-driven machines in the wake of the Second Industrial Revolution, the need for proactive care of the machines was established. The second world war made the industry very competitive and downtime was no longer tolerable in the industry. This led to the birth of preventive maintenance where certain schedules were developed and regular maintenance was being performed and certain observations were also being noted to prevent the machines from unexpected failures. Though this reduced downtime this proved to be very expensive and inefficient. Later on, reliability centered maintenance and risk-based maintenance were developed which proved to be more efficient than the earlier practices. But still, these methods did not involve live tracking the health of the machines and hence there is scope for improvement.

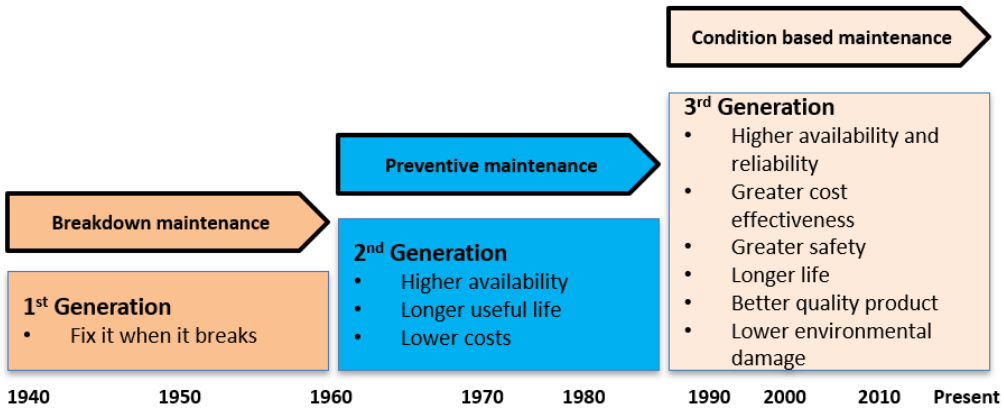

Figure 1.3: Evolution of Maintenance Strategies

Strategy	Run to Failure	Preventive	Predictive
Summary	Fix when it breaks	Maintenance based on a predetermined schedule	Condition based monitoring, uses analytics to predict machine failures

Figure 1.4: Summary of the strategies

With the advent of industry 4.0, the latest industrial revolution, Artificial Intelligence was incorporated into the manufacturing environment, forever changing the way the machines collect and interpret the data. Industrial Internet of things (IIoT) emphasizes the importance of maximizing equipment utilization. The substantial decrease in the installation costs of sensors further aided the installation of sensors in all machines both critical and perfunctory. These sensors can now seamlessly integrate with the company's IT infrastructure which gives a good basis for the data analysis models to monitor the health of the machines that are critical to the company.

Thus the availability of large sensor data, both historical and real-time has offered the scope for predictive analytics. This paved the path to predictive maintenance where we apply advanced analytics to predict machine failures before they occur. In [3], they proposed an RNN model based on LSTM to predict the RUL of Lithium-ion batteries. They used LSTM as it can model sequential data and capture long-term dependencies among the capacity degradation of lithium-ion batteries. In [4], they proposed an Embed-RUL approach for RUL estimation. Embed-RUL utilizes a sequence-to-sequence model based on Recurrent Neural Networks (RNNs) to generate embeddings for multivariate time series sequences. The embeddings for normal and degraded machines tend to be different, and are therefore found to be useful for RUL estimation. In [5], the author described the impact of CNN and LSTM on RUL estimation and also used data augmentation for improved accuracy. All these approaches are trying to predict the remaining useful life of the machine by training model on sensor readings from engines with their corresponding RUL. Our proposed approach differs from the above in the sense that we use the sensor readings to train our model to predict the sensor readings of the next n time steps using RNN with LSTM and classify them as failure or safe using an SVM. We believe this novel approach has the potential to predict if the engine fails in the next n cycles. In the background section we look at the literature of SVM, RNN implemented using LSTM and corresponding model architectures.

In this thesis, we developed various models to predict the remaining useful life left in a turbofan engine using RNN in conjunction with SVM to classify failures. This was achieved by training our models on the historical data of the gas turbine engines of the same type. The dataset is explained

in detail in section 1.4.

Before going into the description of the failure dataset, here is a brief description of failures in a gas turbine engine.

1.3 Performance Degradation in a gas turbine

Fouling and erosion are some of the common types of degradation which can be partially recovered through maintenance whereas airfoil distortions and platform distortions lead to permanent degradation which cannot be recovered. The deterioration can be identified as short term and long term based on the evolution time frame of the deterioration. Short term/rapid deterioration can happen any time during the engine's operation because this will most likely be due to a single event e.g., some foreign item entry into the compressor. Long term deterioration is the one that is more gradual and is the expected mode of degradation because the components are exposed to high temperatures constantly and contaminant accumulation in some parts of the engine [1]. We can generally develop prediction models for long term deterioration only.

1.3.1 Common physical faults

Some common physical faults that occur in a gas turbine have been discussed briefly.

1.3.1.1 Fouling

This is caused by the adherence of contaminants on the surface of the gas turbine components. This results in a change of airfoil shapes and an increase in surface roughness which leads to deterioration in the performance. Compressor failure leads to decreased flow capacity and lower efficiency. It has also been agreed in the literature that fouling influences the flow capacity more than the efficiency. Most of the total performance loss in a gas turbine is due to the compressor fouling. This deterioration can be recovered by performing appropriate maintenance [1].

1.3.1.2 Erosion

It is a gradual loss of the material from the surface of any component which is usually caused by contaminants such as sand, dust, water droplets, etc. Any of the gas-path components can be

subject to erosion but it usually has a higher influence in turbines than in compressors. Similar to fouling performance deterioration due to erosion can be represented using flow capacity and efficiency. It was also observed that industrial gas turbines are less prone to erosion than the aircraft engines [1].

1.3.1.3 Corrosion

Corrosion is an irreversible deterioration of the gas turbine components which occurs due to oxidation or chemical interaction with the air contaminants and combustion gases. This reduces the compressor flow capacity, efficiency of the compressor and turbine. This can be prevented by proper coating [1].

Several other modes cause performance degradation, they can be referred from the literature. Figure 1.5 below gives the classification of the gas turbine faults [6].

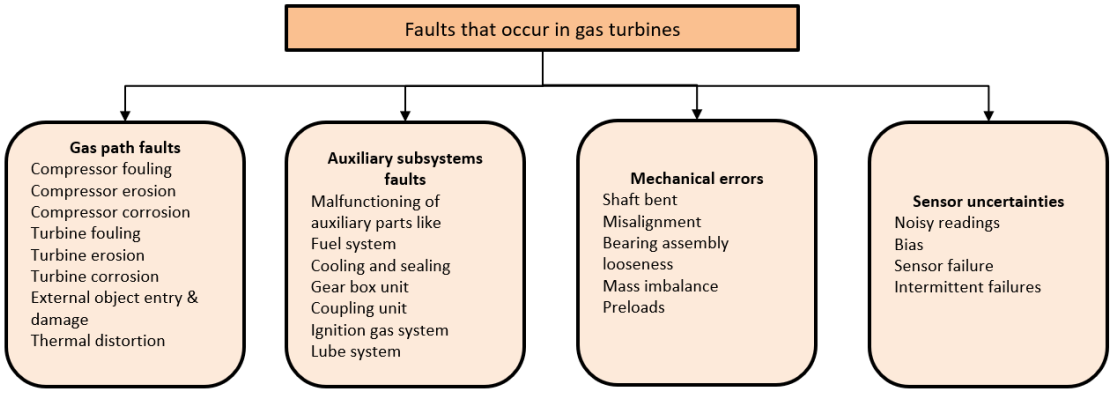

Figure 1.5: Classification of common faults in gas turbine engines

In figure 1.6, a basic overview of a maintenance process is explained. The physical faults change the engine performance parameters which can be observed by the deviations in the parameters that are being measured through sensors (temperature, pressure, shaft speed, fuel flow and power output). These are generally used as fault indicators in the engine health monitoring.

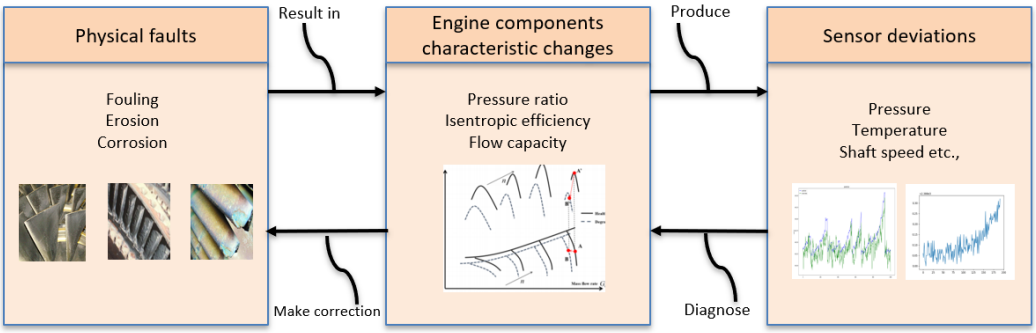

Figure 1.6: Maintenance process overview

Once we observe abnormal sensor readings we diagnose the reason behind the same and take corresponding corrective actions.

1.4 Failure trajectory dataset

The dataset used for testing the performance of our proposed prognostic technique was made available by NASA at their "Prognostics Center of Excellence Data Repository" [7]. This data was simulated using C-MAPSS, a tool for simulating a realistic large commercial turbofan engine. This software was coded using MATLAB® and SIMULINK® [8]. The tool emulates an engine model of 90,000 lb thrust class and includes atmospheric model capable of simulating operations at altitudes ranging from sea level to 40,000 ft, Mach numbers from 0 to 0.90 and sea-level temperatures from -60 to 103 °F.

Figure 1.7 describes the components of a turbofan engine used in C-MAPSS. A turbofan engine is a turbine engine where the first stage compressor rotor is larger in diameter than the rest of the engine. This stage is called the fan. The air passes through the fan near its inner diameter also passes through the remaining stages in the core of the engine and is compressed further. The air through the outer diameter does not pass through the core of the engine but it passes outside of the engine, thus this is called bypass air and the ratio of bypass air to core air is called the bypass ratio. This air accelerated by the fan contributes to the thrust as well at low forward speeds and

low altitudes. In this engine, thrust is developed by a fan rotor system [9].

Figure 1.8 shows how various modules are assembled in the simulation. C-MAPSS allows the user to simulate the effects of faults and deterioration in rotating components of the engine (Fan, LPC, HPC, HPT and LPT). This deterioration can be observed by the outputs which are being monitored using sensors. 21 variables have been chosen for this study.

1.4.1 Sensors

The sensors that are being taken into account can be seen in table 1.1. Altitude, Mach number and TRA are the operational settings and the rest are the sensors that are sensing the variables of interest.

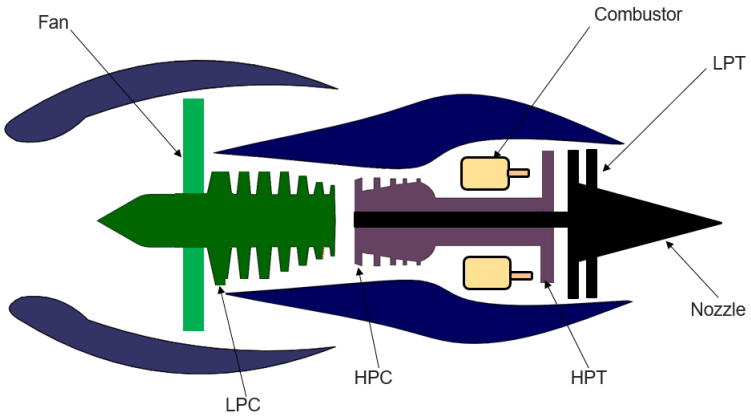

Figure 1.7: Simplified diagram of engine simulated in C-MAPSS

1.4.2 Damage propagation modelling

Generally, in degradation models, exponential behavior of fault evolution is common. This dataset was created with a generalized equation for wear, $w = Ae^{B(t)}$. Further details about this can be found at [8].

Symbol	description	Units
Unit No.	Unit number	–
Time	time, in cycles	–
Altitude	Operational setting 1	ft
Mach No.	Operational setting 2	–
TRA	Operational setting 3	–
T2	Total temperature at fan inlet	°R
T24	Total temperature at LPC outlet	°R
T30	Total temperature at HPC outler	°R
T50	Total temperature at LPT outlet	°R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio(P50/P2)	–
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to PS30	pps/ppi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	–
farB	Burner fuel-air ratio	–
htbleed	Bleed Enthalpy	–
Nf_dmd	Demanded fan speed	rpm
PCNfR_dmd	Demanded corrected fan speed	rpm
W31	HPT coolant bleed	lbm/s
W32	LPT coolant bleed	lbm/s

Table 1.1: Different sensor variables in the Data set

1.4.3 Scenario

Scenarios considered in this dataset were developed considering a number of engines throughout their usage history and that each engine might be used under different conditions. It was also assumed that we cannot solely quantify the damage accumulation after a flight based on the flight duration and conditions. This compels us to rely on the sensor data to get the required information on the health of the engine.

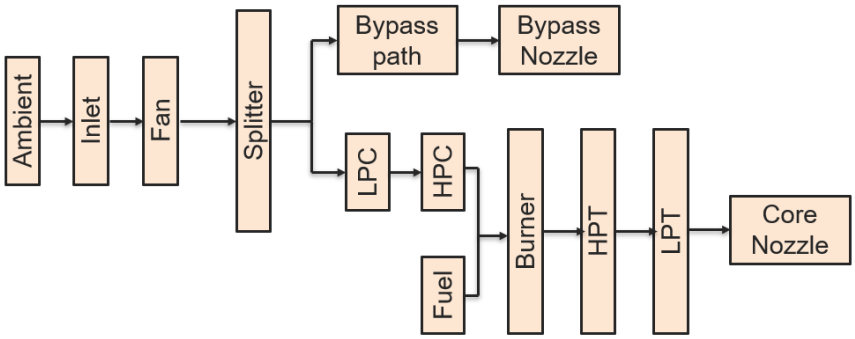

Figure 1.8: Layout of various modules and their connections

1.4.4 Failure modes considered

The failure modes considered in this dataset are the degradation in the HPC and Fan modules under six different combinations of Altitude, TRA and Mach number. For this study, the dataset with a single operating condition was considered and model to deal with multiple operating conditions will be done in the future. This dataset consists of multiple multivariate time series. Each series can be considered as the failure trajectory from a different engine i.e. a fleet of engines of the same type. Initial wear and manufacturing variation were also considered for the engines. The sensor noise was also taken into account. Each series starts with the engine functioning normally, then it pickups a fault because of which there will be a gradual degradation until the system fails. We tested our approach on two datasets, FD001 and FD003. In the FD001 dataset, there are 100 engines' failure trajectories with engines operating at sea level and the fault that occurs is High-Pressure Compressor (HPC) degradation. In the FD003 dataset, there are 100 engines' failure trajectories with the engines operating at sea level and the engine failures are due to HPC degradation and fan degradation.

2. BACKGROUND

2.1 Introduction to Machine Learning

Arthur Samuel coined the term Machine Learning (ML) in 1959 [10] and he stated that "it gives the computer the ability to learn without being explicitly programmed". It is the most influential and powerful technology in the present-day world. This is a tool that can convert information into knowledge. Over the past 50 years, there has been an exponential increase in data collection and this data will be useless unless we analyze it and identify the underlying patterns. Machine Learning techniques automatically find the hidden patterns in the complex data that would be difficult to identify otherwise. This knowledge could be used to make predictions and can also be used for complex decision making. Machine learning has already become a part of our life. Every time we use Google search, give voice commands to a Bluetooth speaker, use face unlock on our phone, we are using Machine Learning which is an integral part of the engine performing the given tasks. ML model which is a crucial component of that engine is constantly learning and improving its performance from every interaction.

2.1.1 Terminology

We call the set of data examples that has features that help in solving the problem as a dataset. These features are important pieces of data that help us in understanding the problem. We call the internal representation that the ML algorithm has learned after training on the data as the model. This is the output we get after the training of the algorithm.

2.1.2 Process

Generally, the process starts with the collection of data on which the algorithm is trained. Next comes data preparation where the data is processed and cleaned into an appropriate format. Extracting the best features and feature reduction also come under this. The next step is training where the ML algorithm is trained on the data. Next, the hyperparameters in the model are tuned using a cross-validation dataset. After this, the model performance can be evaluated on the test set.

2.1.3 Types

All the different approaches to perform Machine Learning can be broadly classified into two major categories, Supervised Learning and Unsupervised Learning.

2.1.3.1 Supervised learning

In supervised learning, the ML algorithm learns the mapping between the inputs and the outputs. Here the algorithm has access to features and corresponding labels. This is called supervised learning because it is similar to learning under the supervision of a teacher. Once the model is trained, it can predict the target responses when new data is given. The supervised learning problems can be divided into regression and classification problems. In a classification problem, the output would be a categorical variable whereas in a regression problem the output is a real continuous variable.

2.1.3.2 Unsupervised learning

Unsupervised learning is where the algorithm will only have the input data to train but no corresponding labels. Unsupervised learning is used to learn the underlying structure in data which can be used to understand hidden patterns within the data. These algorithms are called unsupervised learning because unlike supervised learning, the algorithm does not know the correct answers and there is no teacher to supervise. Here, these are made to discover interesting patterns in the data on their own. Unsupervised learning algorithms can be grouped into clustering and association problems. Clustering is where the algorithm discovers inherent groupings within the data whereas in association type learning, we want to discover the mappings that describe large portions of the data.

For developing our time-to-failure prognostic technique, we have used Support Vector Machines and Recurrent Neural Networks. Both of them were used as supervised learning problems in this work.

2.2 Support Vector Machines

2.2.1 Introduction

Support Vector Machine (SVM) is a well-known machine learning algorithm that has become quite popular over the years through its performance on classification problems. It can conceptually be described as an algorithm that non-linearly maps input vectors to a very high dimensional feature space where a decision boundary is constructed.

2.2.2 History

The SVM algorithm was put forward by Vladimir N. Vapnik and Alexey Ya.Chervonenkis through the development of statistical learning theory. But it can be said that the model close to the current form was formulated when a soft margin classifier was introduced by Cortes and Vapnik [11].

2.2.3 Mathematical formulation

2.2.3.1 Statistical Learning theory

The supervised learning problem can be specified as below in a statistical learning theory [12]. Consider a set j of training data $(x_1, y_1)..(x_j, y_j)$ in $R^n \times R$ sampled as per an unknown probability distribution function $P(x, y)$. $V(y, f(x))$ is the loss function that measures error when f(x) is predicted instead of the actual value y for a given x. The solution to the supervised learning problem would be finding the function(x) that minimizes the expectation of the error on the new data, this can be represented as below:

$$\int V(y, f(x))P(x, y)dxdy \tag{2.1}$$

Empirical Risk Minimization (ERM) principle is employed to infer the function over the hy-

pothesis space that reduces the expected error. Minimizing the empirical error can be written as:

$$\frac{1}{j}\sum_{i=1}^{j} V(y_i, f(x_i)) \tag{2.2}$$

2.2.3.2 Formulation

The traditional view of SVM is that they find an 'optimal' hyperplane as the solution to the learning problem. In the basic formulation of SVM (linear case), the hyperplane lies in the space of the input data x. Hypothesis space, in this case, is the subset of all hyperplanes of the form:

$$f(x) = w.x + b \tag{2.3}$$

In general formulation, SVM finds a hyperplane in a different space than the input space x. This space will be induced by a Kernel K, in this case, it is a dot product in that space. Kernel induces hypothesis space which is now a set of hyperplanes in the feature space. Set of functions in a Reproducing Kernel Hilbert Space (RKHS) defined by K can be referred from these papers [13] [14].

The hypothesis space of the SVM is a subset of a set of hyperplanes defined in an RKHS which can be described as

$$\{f : ||f||_k^2 < \infty\} \tag{2.4}$$

where k is the kernel that defines the RKHS and $\|f\|_k^2$ is the RKHS norm of the function [14]. In the linear case, Kernel $K(x_1, x_2) = x_1.x_2$ is the dot product and the functions considered are of the form described above in the hyperplane equation for the linear one and RKHS norm is simply $\|f\|_k^2 = \|w\|^2$, norm of w. The goal of the SVM is to find a solution that gives the optimal RKHS norm.

Another approach is using a loss function and we are going to consider this for the classification problems. The goal of classification is to minimize the misclassification error, so a loss function sign$(-yf(x))$ can be used for a binary classification problem and based on the sign of the function

f(x) classification will be done. Owing to computational reasons due to scaling, the actual loss function for SVM is $|1 - yf(x)|_+$ [13]. This is also called soft margin loss function because of the standard margin interpretation. The points for which the loss function is zero are the ones that have the margin of at least $\frac{1}{\|f\|_k^2}$.

$$\frac{yf(x)}{\|f\|_k^2} \tag{2.5}$$

For SVM Classification, this margin is a very important geometric quantity, kindly refer to the literature for further information on this concept.

To summarize, the SVM machine learning models work towards minimizing the empirical error while taking the complexity of the hypothesis space into consideration by minimizing the RKHS norm. In practice, SVMs give us the flexibility to perform a trade-off between empirical error and the complexity of the hypothesis space. The SVM classification can be represented by the below optimization problem :

$$\min_f \|f\|_k^2 + C \sum_{i=1}^{j} |1 - yf(x)| \tag{2.6}$$

Here, C is the regularization parameter that controls the tradeoff between the empirical error and the complexity of the hypothesis space to be used.

2.2.4 Implementation

SVM involves solving the optimization described above which can be converted to a Quadratic programming (QP) problem. The QP formulation for SVM classification is as follows:

$$\begin{aligned} \min_f \quad & \|f\|_k^2 + C \sum_{i=1}^{j} \xi_i \\ \text{s.t.} \quad & y_i f(x_i) \geq 1 - \xi_i \text{ for all i,} \\ & \xi \geq 0 \end{aligned} \tag{2.7}$$

Variables ξ_i are the slack variables which measure the error made at a point (x_i, y_i). In the above formulation, the number of constraints will be the same as the number of observations in the training data. This will become challenging when the observations are large. Several proposals have been made that speed up the SVM training. One such method proposed was the primal-dual optimization method which can be referred at [15]. SVM dual formulation :

$$\begin{aligned} \min_{\alpha_i} \quad & \sum_{i=1}^{j} \alpha_i - \frac{1}{2} \sum_{i=1}^{j} \sum_{l=1}^{j} \alpha_i \alpha_l y_i y_l K(x_i, x_l) \\ \text{s.t.} \quad & 0 \leq \alpha_i \leq C, \text{for all i}, \\ & \sum_{i=1}^{j} \alpha_i y_i = 0 \end{aligned} \tag{2.8}$$

Here, K represents the Kernel that was employed. This is usually referred to as the kernel trick, this is employed because in higher dimensional spaces, dot project may be intractable. So special kernel functions that operate on lower-dimensional vectors x_i and x_l are used and this produces a value equivalent to the dot product of higher dimensional vectors.

2.3 Recurrent Neural Networks

2.3.1 Introduction

Artificial neural network (ANN) is an interconnected group of artificial neuron layers that uses a mathematical or computational model for solving complex Artificial Intelligence (AI) problems. ANN can change its structure based on external or internal information that flows through the network. It is generally considered that given enough training data, neural networks can figure out the function that maps the inputs to the output. ANNs with recurrent connections are called RNNs. Recurrent Neural Network is a quite popular architecture among the classes of Artificial Neural Networks. RNNs which come under sequence models are widely used in the research where sequential data is involved. A standard network cannot work in the case where inputs and outputs are of different lengths in different examples and they do not share the features learned across different positions of the input. An RNN model is capable of handling these. It has an

advantage over other neural networks as it can process the input of varying length i.e the input need not be of fixed length. Feature sharing can reduce the number of parameters in our model and RNNs employ this technique. For large input, we can store the historical information that can be used later. [16]

2.3.2 History

As per the literature, John Hopfield is credited with the first formulation of Recurrent like neural network in 1982 [17]. This was built in the context of neuroscience. This was the first attempt at understanding the mechanism of content-addressable memory.

2.3.3 Standard RNN cell

In RNNs, the layers consist of neurons that are affected by both past and current inputs through feedback connections. Different RNN architectures can be made by organizing the recurrent layers. The inner connections and different types of cells give RNNs the flexibility to possess different capacities [18].

2.3.3.1 Notation

Consider X denotes the input training data and y is corresponding output. $X^{(i)<t>}$ denotes the t^{th} element in the sequence of the training example i and $T_x^{(i)}$ denotes the length of the input sequence. $y^{(i)<t>}$ denotes the t^{th} element of output sequence in the training example i and $T_y^{(i)}$ denotes the length of the output sequence [16].

2.3.3.2 Formulation

These are the most common cells found in the RNNs. They generally have sigmoid or tanh activation functions. Figures 2.1 and 2.2 illustrate the same RNN, figure 2.2 represents an unrolled version of the RNN in figure 2.1. In this network, $T_x = T_y$ and in the cases where they are not equal, the architecture will be different. Generally, we initiate random activation at time zero, usually, it will be a vector of zeros. In RNN, the parameters that are used for each time step are shared. The parameters governing the connection between $x^{<1>}$ to the hidden layer is the weight

matrix W_{ax} and it is the same set of parameters that will be used for every time step. Similarly, the activation of horizontal connections are governed by the same set of parameters W_{aa} and will be the same for every time step. Similarly, the parameters W_{ya} will also be shared across different time steps. It can also be observed that while predicting the output at each time step, information from not only the input at that particular time step but also from previous time steps is utilized (through the activation from the previous time step).

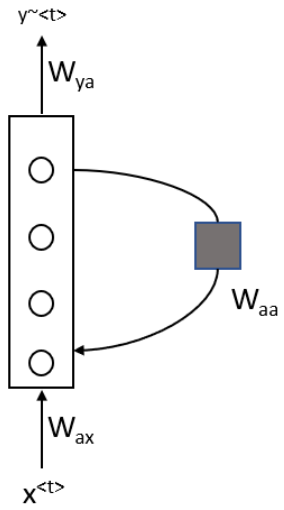

Figure 2.1: Schematic of standard recurrent cell

The mathematical formulation of this cell is as follows:

$$
\begin{aligned}
a^{<0>} &= \vec{0} \\
a^{<t>} &= g(W_a[a^{<t-1>}, x^{<t>}] + b_a) \\
W_a &= [W_{aa} W_{ax}] \\
[a^{<t-1>}, x^{<t>}] &= \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}
\end{aligned}
\tag{2.9}
$$

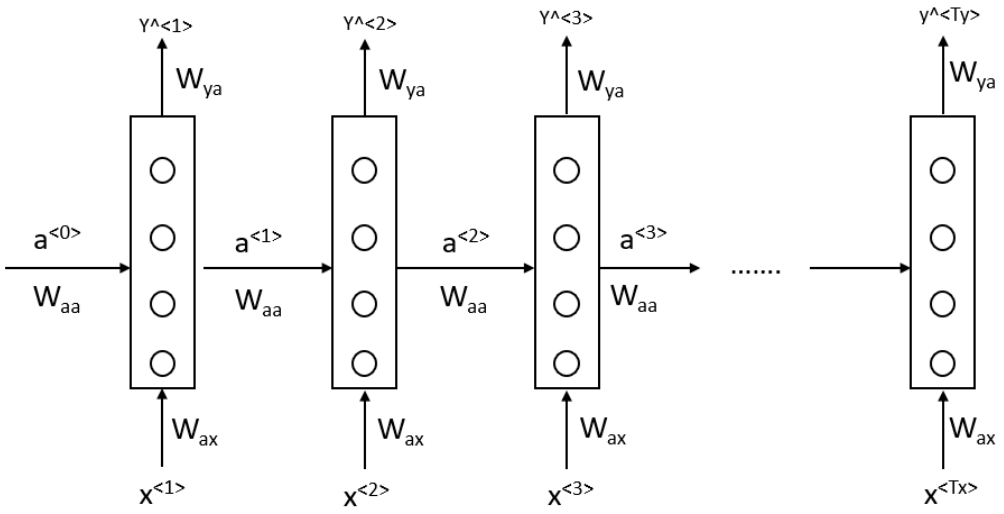

Figure 2.2: Schematic of standard recurrent cell (unrolled)

$$y^{<t>} = g(W_y a^{<t>} + b_y) \tag{2.10}$$

Here, W_a is W_{aa} and W_{ax} stacked horizontally, $[a^{<t-1>}, x^{<t>}]$ is $a^{<t-1>}$ and $x^{<t>}$ stacked vertically. $g()$ represents the corresponding activation function and b represents bias [16].

2.3.3.3 Loss function

We can use the appropriate loss functions depending on our problem. If the problem is a classification problem, we can use a cross-entropy loss function and if it is a regression problem, then we can use the mean squared error as the loss function.

$$L^{<t>}(y\hat{}^{<t>}, y^{<t>}) = -y^{<t>} log(y\hat{}^{<t>}) - (1 - y^{<t>}) log(1 - y\hat{}^{<t>}) \tag{2.11}$$

This is the loss for a single example and the loss for the entire sequence is given by the summation over all the calculated single example losses of the sequence.

$$L^{<t>}(y\hat{}, y) = \Sigma_{t=1}^{T_y} L^{<t>}(y\hat{}^{<t>}, y^{<t>}) \tag{2.12}$$

However, one weakness in this basic recurrent neural network cell is that the information from much earlier in the sequence cannot influence the output of a later sequence. In other words, these standard recurrent cells are not capable of handling long-term dependencies. As the gap between the related input grows it is difficult to connect the information. This was attributed to the vanishing or exploding gradients which will be discussed later in section 2.3.5 [18][16].

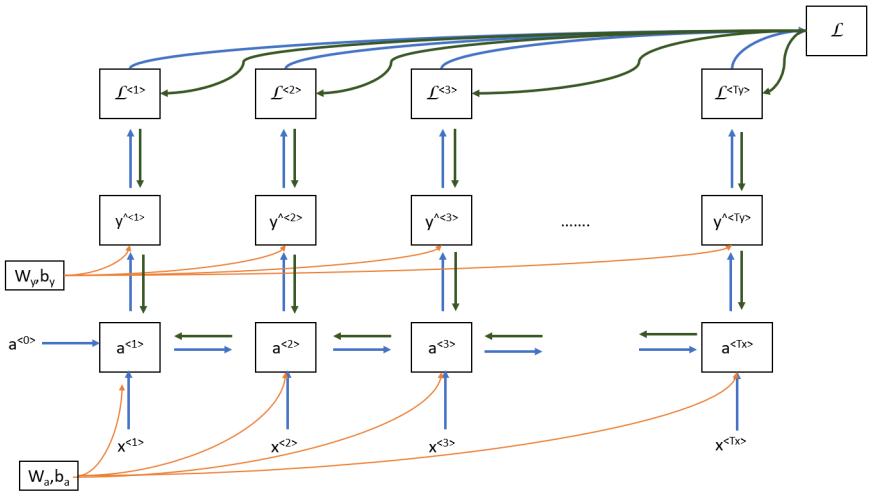

Figure 2.3: Back propagation through time in RNN

Figure 2.3 presents a detailed view of forward and backpropagation in RNN. Forward propagation is denoted by a blue line while the red line indicates backpropagation. Backpropagation here is called backpropagation through time because we pass the activation from one element in the sequence to another backward in terms of time. Backpropagation facilitates the computation of all the appropriate quantities that are required to take the derivatives of parameters which are then utilized to update the parameters using gradient descent algorithm [16].

2.3.4 Different types of RNNs

The RNN described above is a type of RNN architecture in which $T_x = T_y$, but in general, this is not the case. In cases where they are not equal, we should use a different RNN architecture. The architecture we used above is called many to many architecture. Figure 2.4 shows the different

types of RNN architectures [19]. We have employed many to one architecture in our problem where we will be giving a sequence as an input and the model will predict the output of the next time step [16].

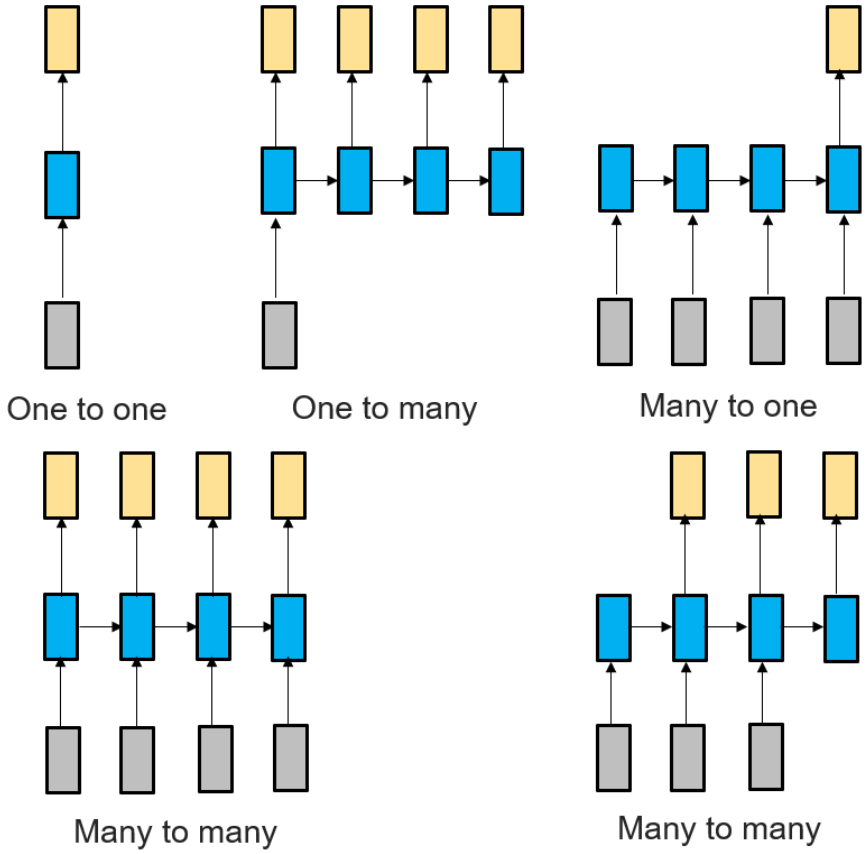

Figure 2.4: Types of RNN architectures

2.3.5 Vanishing gradients problem

In some cases, the data may have very long term dependencies where the data from much earlier in the sequence can affect what comes much later. Basic RNN that was described earlier is not very good at capturing such very long term dependencies. This is because of the vanishing gradients problem just like in very deep neural networks. It will be difficult for errors in the outputs associated with earlier time steps to affect the computations that are made later. In practice, what it

means is that it might be difficult for the RNN to realize that it needs to memorize some information that will impact future predictions. The basic RNN mentioned above only has local influences, for example, output $y\hat{}^{<t>}$ will be closely influenced by values close to it but not by the values much earlier [16].

It will be very difficult for the error to backpropagate to the beginning of the sequence and hence it will be difficult for it to modify how the neural network is doing computations earlier in the sequence. This is the weakness of the basic RNN and they tend to be poor at capturing long term dependencies. For the case of exploding gradients, applying gradient clipping would suffice, but to deal with the vanishing gradients the architecture of the RNN has to be modified slightly [16].

2.3.6 LSTM

Long short-term memory is a type of RNN architecture that enables us to capture the long term dependencies in the data. It is more powerful and complex compared to the Gated Recurrent Unit (GRU) which is another type of RNN that can help in solving the vanishing gradient problem. Unlike feedforward networks, LSTMs have feedback connections. LSTM was proposed in 1997 by Sepp Hocheriter and Jurgen Schmidhuber.

The first attempt to deal with long-term dependencies were made by Hochreiter and Schmidhuber in 1997 [20], they proposed the LSTM cell. They increased the remembering capacity of the standard recurrent cell by introducing the concept called 'gate' into the cell. There are multiple variations in this, but we have used the LSTM cell with a forget gate. In 2000 Gers, Schmidhuber and Cummins modified the original LSTM by introducing a forget gate.

Mathematically this can be represented as below :

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c) \tag{2.13}$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{2.14}$$

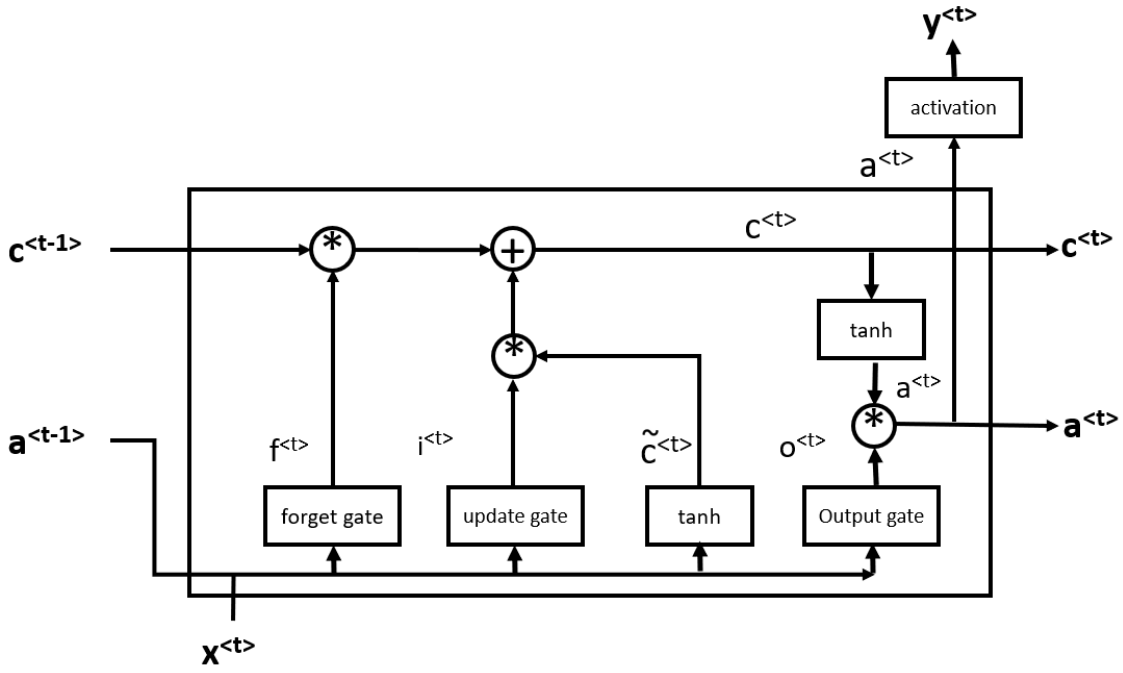

Figure 2.5: LSTM with a forget gate

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{2.15}$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o) \tag{2.16}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \tag{2.17}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>} \tag{2.18}$$

where $c^{<t>}$ denotes the cell state,Γ_u represents update gate, Γ_f represents forget gate, Γ_o represents output gate, W_c, W_u, W_f, W_o are the weights, $\tilde{c}^{<t>}$ denotes a candidate for the cell state, operator '*' represents point-wise multiplication of the two vectors. When updating the cell state the input (update) gate can decide which information to be stored, forget gate can decide what information to throw away, when forget gate's value Γ_f is 1 it keeps the information if its 0 it gets rid of all the information and the output gate takes the decision on what information to output based on the state of the cell. In GRU we have an update gate, a relevance gate and a candidate cell variable while in LSTM we have an update gate, a forget gate, an output gate and a candidate cell variable. There

is no clear evidence in the literature that LSTM is better than GRU or that the GRU is better than LSTM. But LSTM is often the first choice as it is more powerful and general.

We have used Keras API to implement RNN with LSTM cell and the information about the architecture has been explained in the next section.

3. METHODOLOGY

We have established the need to predict failure before its occurrence in the gas turbine engines in the previous sections. In this section, we introduce the current industry standard for the land-based gas turbine engines and this was considered as the benchmark to compare our proposed technique's performance. We then, describe our approach to predict the time-to-failure of a gas turbine engine. We then, present our prognostic technique which uses standard machine learning algorithms (SVM and RNN) to estimate time to failure of a gas turbine engine using the dataset described earlier in section 1.

3.1 Current Industrial Approach

This is the approach currently being employed in the industry for the detection of failure of land-based gas turbines. This approach uses the historical sensor data to get the engineering estimates of the maximum and minimum values of parameters corresponding to that component. The bounds are narrower than the bounds recommended by Original Equipment Manufacturers because this gives some buffer time to react to failure. If the observed sensor readings of the parameters are within these bounds the engine is considered to be operating normally. Even if a single parameter goes out of the bounds the engine is supposed to have reached failure and the corresponding personnel is alerted about the imminent failure of the engine.

Algorithm 1 :Model to predict failure using bounds

1: Get historical sensor data of different engines of the same model monitored till their failure
2: Preprocess the data and figure out the sensors that have the most information regarding the failure prediction
3: Decide upon the bounds for each sensor based on the historical data from the sensors
4: Classify the live data for each sensor as safe or unsafe
5: **if** Any of the sensor is classified as unsafe **then**
6: Alert the corresponding personnel about the critical condition of the engine
7: **else**
8: Continue with the next live reading from the sensors
9: **end if**

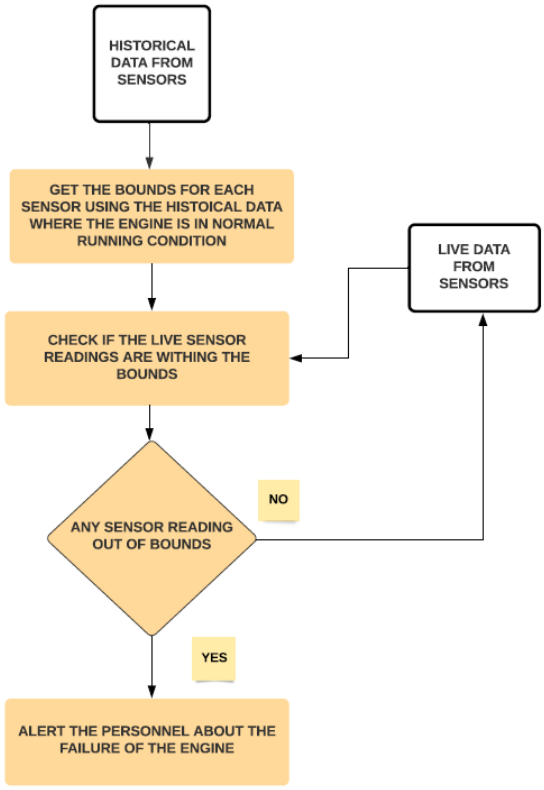

Figure 3.1: Flowchart for model to predict failure using bounds

The use of this approach was observed to raise false alerts, much before the engine reaches its critical limit. Thus, this approach leads to a conservative use of the equipment leading to a significant loss of capability.

3.2 Our Approach

To improve upon the current industrial approach, we propose the creation of bounding hypervolumes in high dimensional space within which the unit is historically known to be operating normally. Figure 3.2 depicts this concept pictorially. Here, a current operating point of a unit in three-dimensional parameter space is shown. Five hypervolumes are also shown in green. These volumes (which can be arbitrarily shaped, e.g., not necessarily boxes) represent historical values of the multiple parameters being monitored under different operating states of the unit. When the operating point in parameter space is within the appropriate volume for the given unit state, the

unit is considered to be operating normally with a high probability. Among the other benefits, this approach will eliminate any start-up/shutdown false alerts. We used Support Vector Machines (SVMs) for the creation of bounding hypervolumes. Other classification algorithms should also be explored and this will be explored in future work.

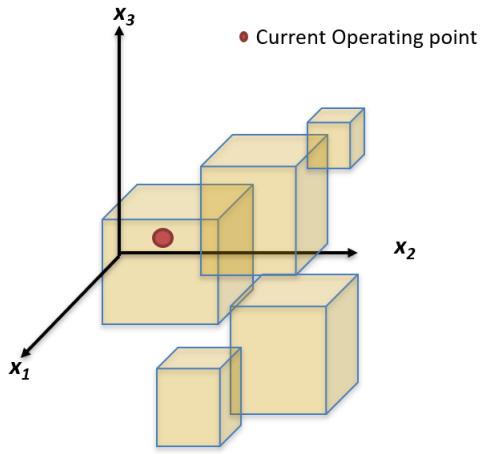

Figure 3.2: Notional depiction of hypervolume bounds for multiple parameter information

The current industrial approach does not have the capability to estimate time-to-failure as the alert is triggered when an abnormal reading is observed. Time-to-failure estimation is required for better decision making and automation of the decision-making process. We used Recurrent Neural Networks (RNNs) for forecasting the sensor readings. This technique has seen some success in recent past [21]. The concept of using a recurrent neural network (RNN) [22] for time-to-failure estimation is shown in Figure 3.3. Here, an RNN has been trained to forecast time series data associated with sensed parameters for a given component or unit. These forecasts can then be fed back to the multiple parameter correlation model described previously to determine when failure is expected to occur. Since the multiple parameter approach can provide probabilistic information regarding when bounds will be exceeded, this information can also be used here to probabilistically predict when failure will occur. This can then be used within a prescriptive analytics approach to decision-making based on the results of an FMEA. The main challenge for the training of this

approach is the availability of relevant data. A careful review of all available data for components and units would be required to determine what architectures are possible. This information should eventually be supplemented by synthetic data generated by digital twins. This data should be validated (e.g., against both historical data that is available and via "Turing test").

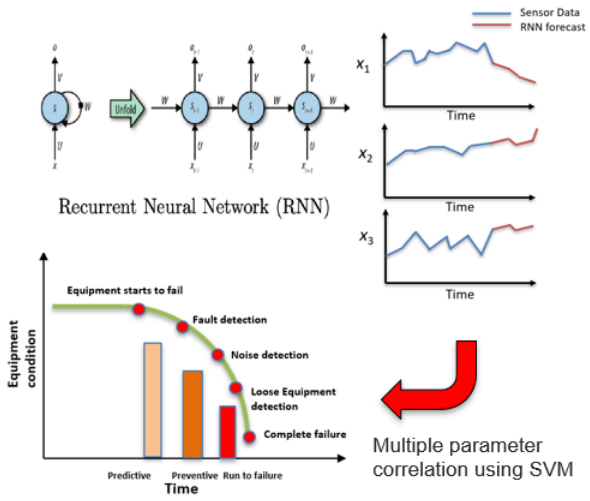

Figure 3.3: Recurrent Neural Network for Time-to-Failure Estimation

3.3 Model to predict engine failure using SVM and RNN

We have introduced our approach for estimating the time-to-failure of a gas turbine engine earlier. To accomplish the prediction of time-to-failure, we developed a model that would tell us if the engine will fail in the next n cycles (cycles here refer to a unit of time for e.g., the time interval between two successive sensor readings recorded). This model can tell us if the engine fails in the next n cycles if we have enough training data. This way, we can fix the n as per our requirement and thus be able to predict if our engine is close to failure or not. Currently, we are predicting 10 cycles into the future and plan to predict more cycles in the future. To create bounding hypervolumes that denote normal operating conditions, we trained an SVM model on the training data. We have the choice to choose the parameters that can give high sensitivity or in other words high true positive rate. We then trained a model for each variable that can forecast the next n time steps' readings

of each sensor. Once we predicted all the sensor readings, what we now have are likely the future sensor readings of the engine. We can classify them using the SVM we trained earlier. If we get any time step classified as a failure, then we alert the concerned personnel that the corresponding engine has reached its critical safety limit and needs to be looked at.

We developed RNN models using Keras (2.3.0), a high-level neural networks API written in python, built on Tensorflow. Keras is more user friendly and we can build simple or complex neural networks with minimal lines of code using Model and Sequence APIs. For advanced operations in the future, Tensorflow is recommended. We used Scikit learn (0.21.2) package for implementing the SVM.

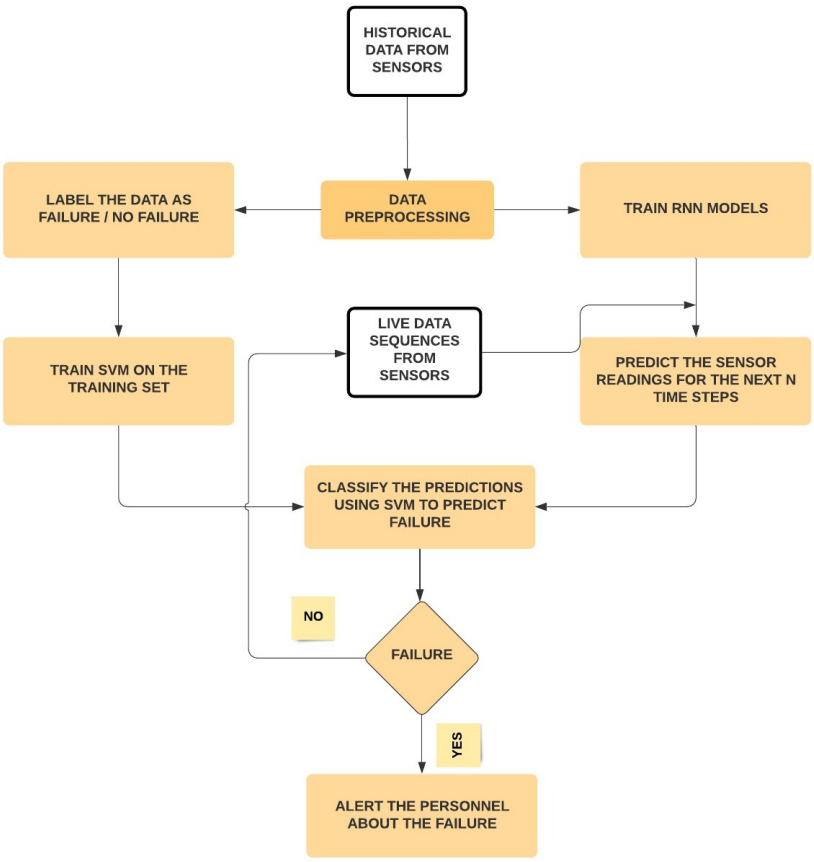

Figure 3.4: Flowchart of Algorithm

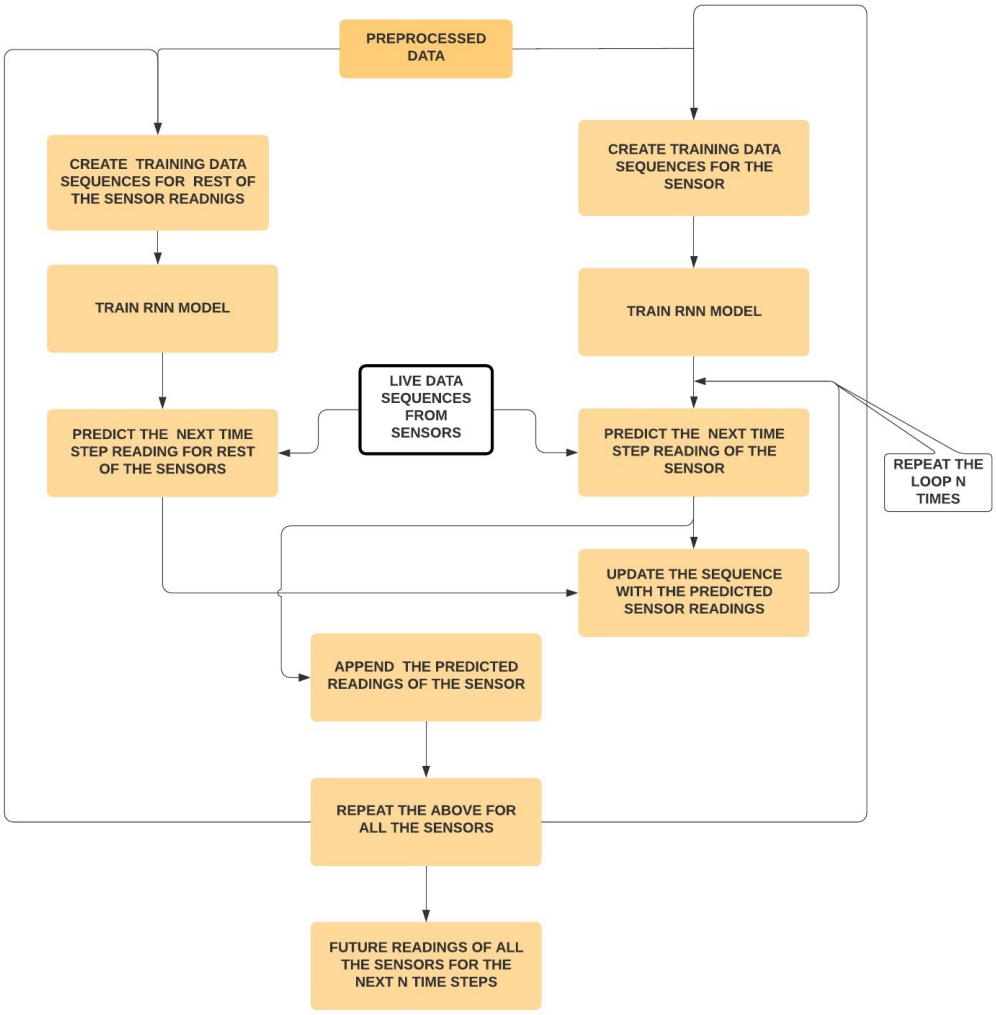

Figure 3.5: Flowchart of the Algorithm (RNN model)

Algorithm 2 Pseudo-code for the prediction of time-to-failure of the gas turbine engine

1: Get historical sensor data of different engines of the same model monitored till their failure
2: Preprocess the data
3: Label the data as failure and safe, last 15 cycles of each engine's failure trajectory was considered as failure and rest were considered to be safe.
4: Fit a Support Vector Machine model to this data
5: Tune the parameters to ensure high sensitivity
6: **for** i from 1 to n engines **do**
7: Create time sequences of the sensor data which will be used to train the RNN
8: Repeat the above for loop on the test data we want to predict RUL on.
9: **end for**
10: **for** i from 1 to n engines **do**
11: Create the corresponding label sequences to train the RNN
12: **end for**
13: Repeat the above for loops on test data to test the performance of the model
14: **for** j from 1 to m variables **do**
15: Create a RNN model corresponding to each variable with LSTM and dropout layers, that can predict the next reading of the variable by looking at a fixed sequence of past readings
16: **end for**
17: **for** i from 1 to 10 **do**
18: Predict the next reading of each variable using the trained RNN model for the corresponding variable and concatenate all these sensors values, this represents the predicted sensor readings for the next time step.
19: Replace the first reading from the sequence and append the predicted time step to the sequence. We now have the sequence of the same length but with updated sensor readings.
20: Use this updated time sequence to predict the sensor readings for the next time step.
21: **end for**
22: We obtain the predicted sensor readings for the next 10 time steps into the future.
23: Use the trained SVM model to classify for each of the time steps
24: **if** SVM predicts any one of the time steps as failure **then**
25: Alert the concerned authority that the corresponding engine has reached its critical limit
26: **else**
27: Discard the first time step sensor reading of the time sequence, append the actual sensors readings observed at the next time step and repeat from step 14
28: **end if**

3.3.1 Data Preprocessing

In this dataset, for each engine, we have a time series data starting at some point in its life cycle and the readings are recorded till the failure occurs. We also keep track of the time (in cycles) from the time we started recording the readings until failure.

Using MinMaxScaler function in the Scikit learn package, we performed normalization of the data. The advantage of using normalization is that all the variable values will now be on a common scale and this will help the gradients to converge faster.

RNN is capable of taking time sequence data as input to predict the output once it is trained on similar data. To train the RNN model, we generated sequences of sensor data for each engine and we created the corresponding labels for each of these sequences. We used these sequences of sensor data and the labels to train the RNN models. The complete algorithm and the architecture of the RNN model are described in algorithm 2 and figures 3.4 to 3.9.

In this model, we labeled each time step to be a failure or no failure. We considered the last 15 readings in the failure trajectory of each engine to be a failure for this problem. But for the industrial data on which this model will be applied this limit can be set accordingly. This labeled data was used to train the SVM to classify the failure.

3.3.2 Model Architecture

For simplicity, we created RNN models with the same architecture to predict each of the sensor readings. Each model has a stacked LSTM with a dropout layer in between followed by another dropout layer, then we have a dense layer followed by a linear activation layer which can be seen in figures 3.6 and 3.7. The number of parameters associated with each layer can also be observed in figure 3.6.

We have another RNN model to predict all the remaining sensor readings of the next time step (n+1) to facilitate the prediction of the sensor of interest for (n+2)nd step. Thus, we remove the first time step of the last sequence and append the predicted sensor readings to predict the sensor readings for the future time steps. This second RNN model must also be developed for each of the

Layer	Type	Output	Parameters
lstm_1	LSTM	(None,15,100)	50000
dropout_1	Dropout	(None,15,100)	0
lstm_2	LSTM	(None,50)	30200
dropout_2	Dropout	(None,50)	0
dense_1	Dense	(None,1)	51
activation_1	Activation	(None,1)	0
Total params: 80,251			
Trainable params: 80,251			
Non-trainable params: 0			

Figure 3.6: RNN Model used for Sensor prediction

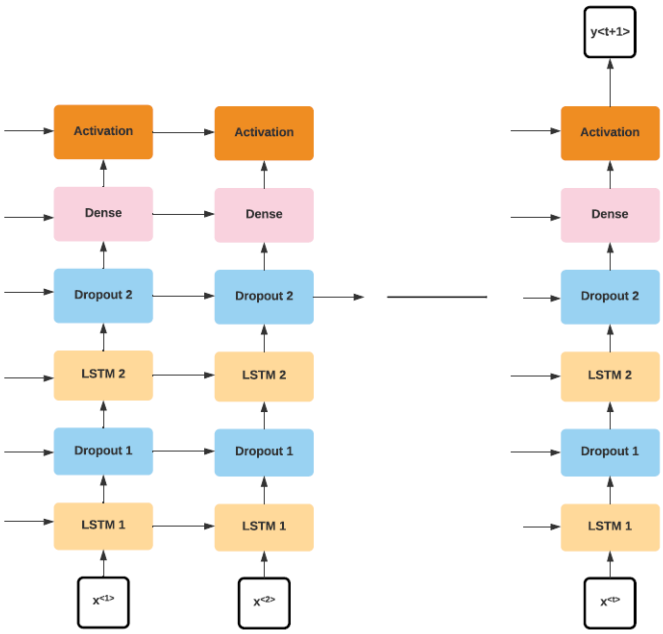

Figure 3.7: RNN Model architecture used for Sensor prediction

sensors to facilitate multi-step prediction. This model's architecture can be seen in figures 3.8 and 3.9.

Once, all these models have been trained, we tested the performance on a test set. For a particular sequence, we were able to predict the next 10 time steps and then used the trained SVM model to classify these time steps.

We have developed a model that will tell us if the engine will fail in the next 10 cycles (cycles

Layer	Type	Output	Parameters
lstm_1	LSTM	(None,15,100)	50000
dropout_1	Dropout	(None,15,100)	0
lstm_2	LSTM	(None,50)	30200
dropout_2	Dropout	(None,50)	0
dense_1	Dense	(None,24)	1224
Total params: 81,424			
Trainable params: 81,424			
Non-trainable params: 0			

Figure 3.8: RNN Model used for prediction of rest of the sensors

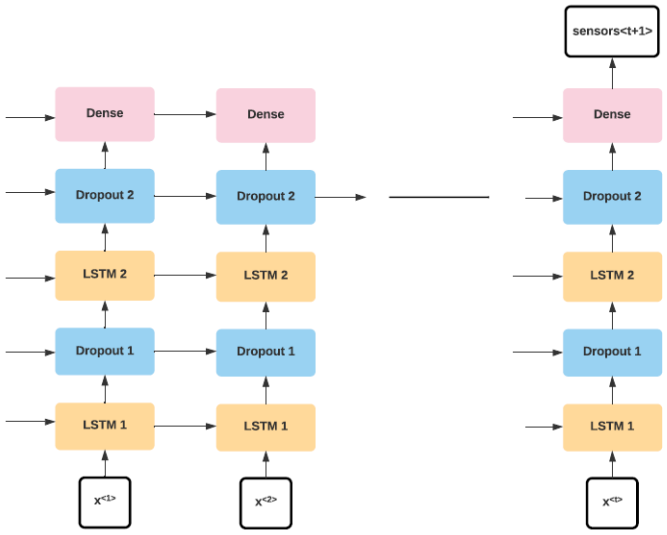

Figure 3.9: RNN Model architecture used for prediction of rest of the sensors

here refer to a unit of time for e.g., the time interval between two successive sensor readings recorded can be considered an hour for better understanding). Based on the data at hand, our model was accurate for 10 cycles. But in the industry, where we can have a large dataset due to continuous monitoring, we are sure that this model can be used to predict more than 10 cycles into the future. Different RNN architectures for different sensors will be explored in the future.

4. RESULTS

4.1 Engine failure prediction model with SVM and RNN

Our prognostic approach was established in the earlier section. Our prognostic technique uses an SVM model to create bounding hypervolumes using sensor data when the engine is operating normally and RNN models to forecast the sensor readings. In this section, we will first describe the SVM model and RNN models used and then present the results of our approach on the datasets, FD001 and FD003. We have also compared the performance of our proposed approach with the current industry standard for land-based gas turbine engines.

4.1.1 SVM model on FD001

SVM model was trained on 80 out of 100 engines' failure trajectories. As explained before the last 15 cycles of each trajectory were considered failures. We employed a radial basis function as the kernel and the hyperparameters after tuning using grid search were gamma = 0.01 and C=1000. We have created another SVM model using weighted classes and the weights used were $\{0-1, 1-10\}$. These are the results from both the non-weighted and weighted SVM models:

		PREDICTED	
		0	**1**
TRUE	**0**	**4162**	31
	1	55	**245**

Figure 4.1: Confusion Matrix on the test set FD001 non-weighted classes

4.1.1.1 Performance summary of SVM on FD001

The confusion matrix of both the non-weighted and the weighted SVMs was used to measure the performance. The results are summarized in figure 4.3. The accuracy of the non-weighted

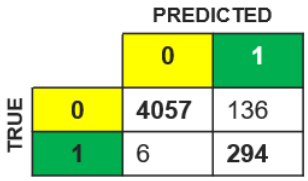

Figure 4.2: Confusion Matrix on the test set FD001 weighted classes

	Accuracy(%)	Sensitivity(%)	Specificity(%)	False Positive rate(%)
SVM (No weights)	98.1	81.7	99.3	0.7
SVM (Weighted)	96.9	98	96.75	3.25

Figure 4.3: Results of SVM model on FD001

SVM model was 98% with a sensitivity of 82%. For the weighted SVM, the accuracy reduced to 97% but the sensitivity increased to 98% along with an increase in false-positive rate to 3.3%. We can choose either of these models depending on the criticality of the equipment being monitored.

4.1.2 SVM model on FD003

The training was done similar to that on FD001. We employed a radial basis function as the kernel and the hyperparameters after tuning using grid search were gamma = 1 and C=10. We have created another SVM model using weighted classes and the weights used were $\{0-1, 1-10\}$. These are the results from both the non-weighted and weighted SVM models:

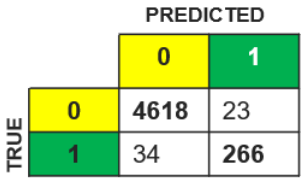

Figure 4.4: Confusion Matrix on the test set FD003 non-weighted classes

		PREDICTED	
		0	1
TRUE	0	4551	90
	1	6	294

Figure 4.5: Confusion Matrix on the test set FD003 weighted classes

	Accuracy(%)	Sensitivity(%)	Specificity(%)	False Positive rate(%)
SVM (No weights)	98.8	88.7	99.5	0.5
SVM (Weighted)	98.1	98	98.1	1.9

Figure 4.6: Results of SVM model on FD003

4.1.2.1 Performance summary of SVM on FD003

The confusion matrix of both the non-weighted and the weighted SVMs was used to measure the performance. The results are summarized in figure 4.6. The accuracy of the non-weighted SVM model was 99% with a sensitivity of 89%. For the weighted SVM, the accuracy reduced to 98% but the sensitivity increased to 98% along with an increase in false-positive rate to 2%. We can choose either of these models depending on the criticality of the equipment being monitored.

4.2 RNN Model predictions

For the training of RNN models with the architecture described in the earlier section, we used 80 engines' failure trajectories in sequences of 15 time steps. Mean Squared Error (MSE) was used as the loss function, rmsprop was the optimizer with a batch size of 200 and epochs of 100. As explained earlier, we used two different RNN models to forecast 10 time steps of a particular sensor.

Figure 4.7 illustrates the predictions made by the RNN models for the sensors T24, NRc, P30 and BPR. This is the prediction of these sensors for 5 different engines over the course of their lifetime, in the figure, each peak or trough represents the failure of an engine. The model is

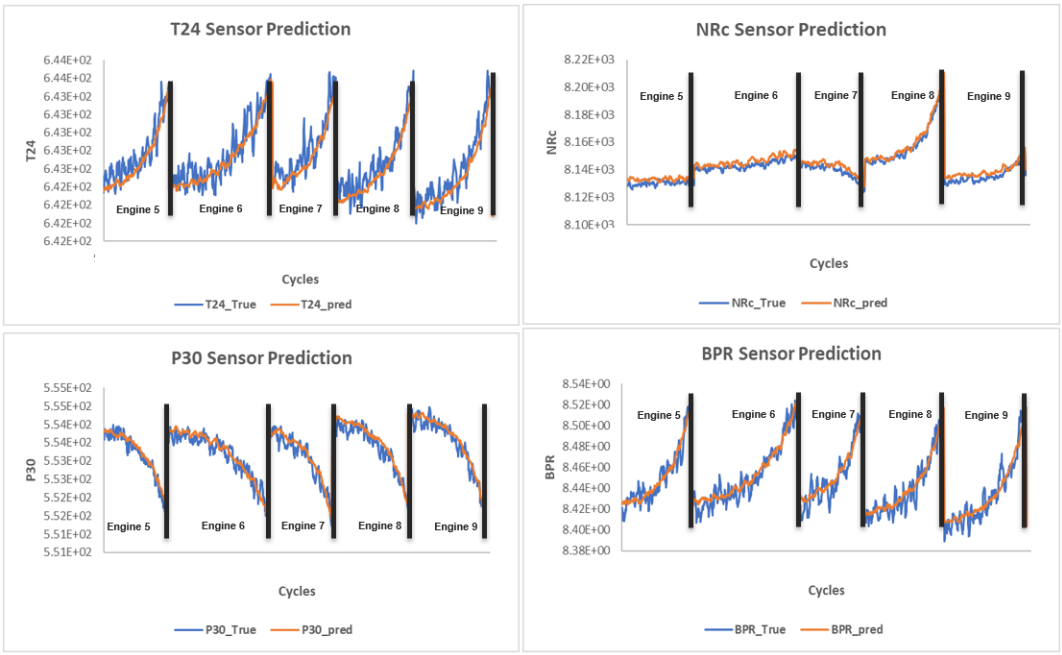

Figure 4.7: Predictions of Temperature, Pressure, Bypass ratio and Core speed using RNN models

capable of understanding the patterns of the variables as the predictions made by our RNN models are pretty close to the actual sensor readings. Since our objective was to provide evidence for the proof of concept, we did not explore more complex RNN architectures, they will be explored in future work.

After training all the models for different sensors, we used the trained SVM to classify the predicted sensor readings of the future time steps. We predicted 10 cycles into the future and classified them as a failure or no failure. It was observed that as the engine reaches the failure limit the classifier was able to predict the failure.

4.3 Prediction of time-to-failure of an engine using our approach

We applied our proposed time-to-failure prognostic technique on engine 81 of the FD001 dataset. The output from our proposed technique is shown in the following figure 4.8. This engine has a lifetime of 240 cycles. Actual failure starts at the 225^{th} cycle. To test our approach, we applied it at three different stages in the life cycle of this engine. Case 1 is where the engine is far away from failure and our approach correctly predicts no failure in the next 10 cycles. We

then considered the stage where the engine was close to failure and our approach predicted at 215th cycle that the engine will be failing at 224th cycle which is close to the actual failure. We also considered the case where the failure has already started, as expected the approach indicated that the engine has entered failure. Figures 4.8 & 4.9 give a pictorial description of the same.

Case 1:
Sequence input : **100:115** cycles
Predicted : **116:125** cycles
Classified none to be failure

Cycle	Failure Prediction
116	0
117	0
118	0
119	0
120	0
121	0
122	0
123	0
124	0
125	0

Case 2:
Sequence input : **200:215** cycles
Predicted : **216:225** cycles
Classified **224th & 225th** as failure

Cycle	Failure Prediction
216	0
217	0
218	0
219	0
220	0
221	0
222	0
223	0
224	1
225	1

Case 3:
Sequence input : **215:230** cycles
Predicted : **231:240** cycles
Classified all to be failure

Cycle	Failure Prediction
231	1
232	1
233	1
234	1
235	1
236	1
237	1
238	1
239	1
240	1

Figure 4.8: Our approach on Engine 81 of FD001

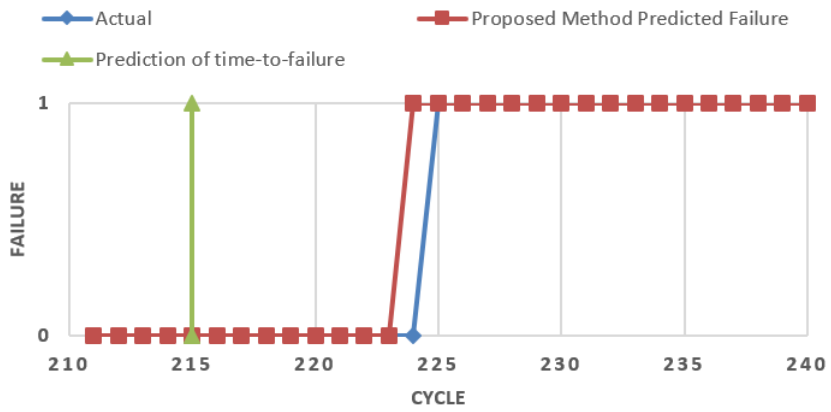

Figure 4.9: Failure prediction of engine 81 using our proposed approach

4.4 Comparison between our proposed approach and the current industrial approach

We compared the performance of our proposed approach with the current industrial approach on the datasets, FD001 and FD003.

4.4.1 On the dataset FD001

We compared the two approaches on the engine 81 of FD001 dataset (figure 4.10). The current approach predicted the failure at 211th cycle whereas the actual failure starts at 225th cycle leading to sub-optimal utilization of the unit. On the other hand, our approach predicts failure from 224th cycle at 215th cycle and hence, offers an opportunity for the better use of the unit over the current approach.

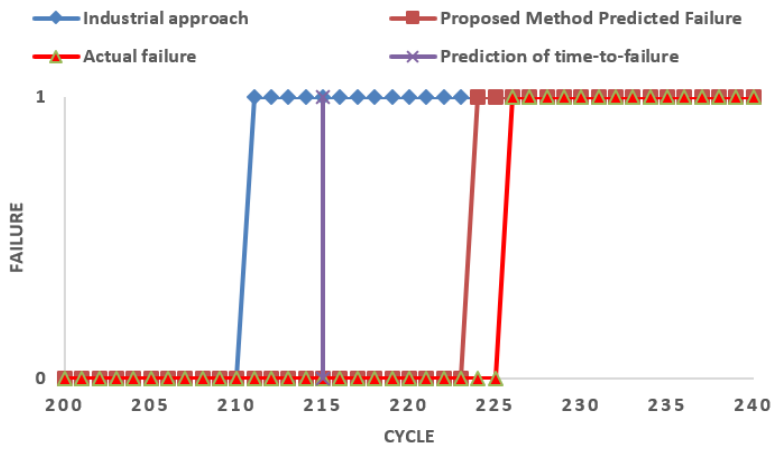

Figure 4.10: Comparison between our proposed and the current industrial approach on engine 81 of FD001

Similarly, we compared both the approaches on the engine 82 of FD001 (figure 4.11). The actual failure of this engine starts from 191st cycle but the current industrial approach predicts failure at 10th cycle, which can be considered a false alert leading to a significant wastage of capability of the engine. Our proposed method predicted at 194th cycle that the engine is going to fail at 204th cycle. Though there is a delay in failure prediction by our method, this was actually

predicted at 194th cycle which is five cycles before the actual failure giving us enough time to prepare for it.

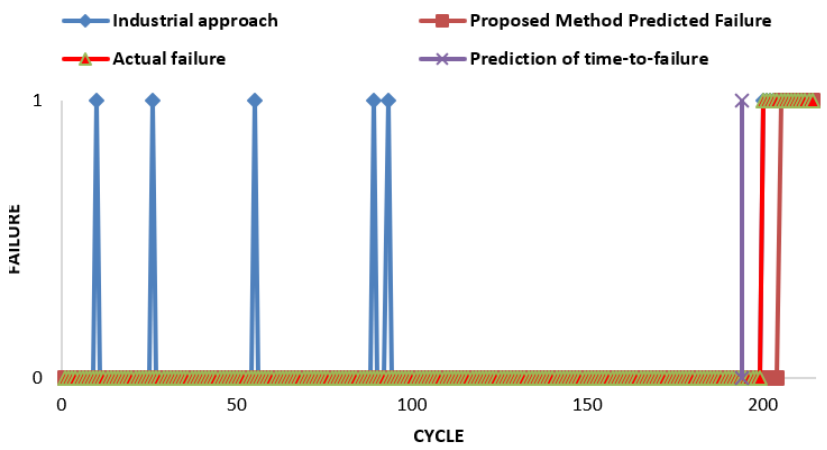

Figure 4.11: Comparison between our proposed and the current industrial approach on engine 82 of FD001

Engine No	Single parameter method	Proposed method		Actual start of failure	Comments
		Predicted at time step	Predicted failure step		
81	211	**215**	224	**225**	
82	10	**194**	204	**199**	False alert
83	5	**278**	282	**278**	False alert
84	231	**242**	251	**252**	
85	169	**174**	177	**173**	
86	223	**254**	262	**263**	
87	158	**159**	168	**163**	
88	190	**191**	201	**198**	
89	199	**202**	204	**202**	
90	142	**147**	152	**139**	
91	109	**120**	124	**120**	
92	45	**318**	319	**326**	
93	6	**146**	151	**140**	False alert
94	237	**232**	238	**243**	
95	4	**267**	270	**268**	False alert
96	299	**314**	318	**321**	
97	189	**186**	190	**187**	
98	29	**147**	152	**141**	False alert
99	173	**174**	177	**170**	
100	6	**182**	189	**185**	False alert

Figure 4.12: Comparison between our proposed and the current industrial approach on FD001

We have compared the two approaches on the complete test data of 20 engine failure trajectories. The results are illustrated in figure 4.12. The current industrial approach as stated earlier was giving many false alarms leading to conservative use of the units as we are losing significant capability by servicing them before they have reached their critical limits. Our approach generally outperformed the current industrial approach and gave a better opportunity to utilize the units to their complete potential.

4.4.2 On the dataset FD003

We compared the two approaches on engine 81 of the FD003 dataset (figure 4.13). The current approach predicted the failure at 20th cycle which is a false alert as the actual failure starts at 333rd cycle which leads to sub-optimal utilization of the unit. On the other hand, our approach predicts failure from 332nd cycle at 328th cycle and hence, offers an opportunity for the better use of the unit over the current approach.

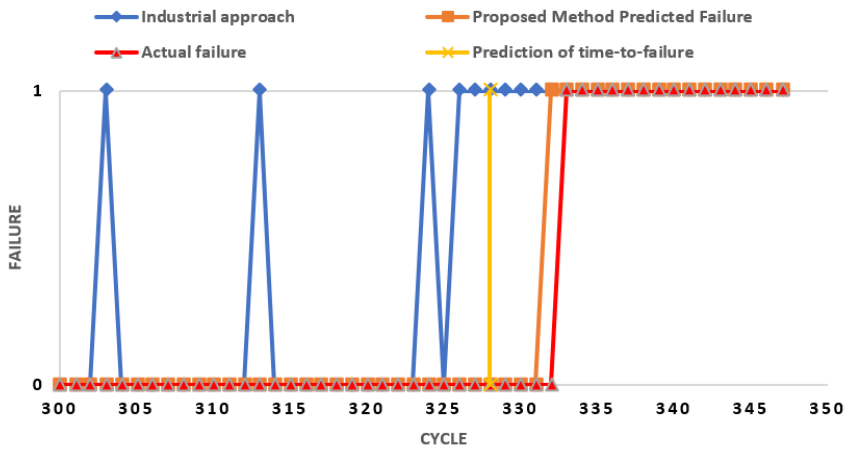

Figure 4.13: Comparison between our proposed and the current industrial approach on engine 81 of FD003

Similarly, we have compared the two approaches on engine 82 of the FD003 dataset (figure 4.14). The current approach predicted failure at 245th cycle when the actual failure starts at 271st cycle which leads to conservative use of the unit as it is being serviced before reaching its critical

limit. This indicates that we are not extracting the maximum potential from the unit. On the other hand, our approach predicts failure from 274th cycle at 268th cycle and hence offers an opportunity for the better use of the unit over the current approach. Though there is a delay in failure prediction by our method, this was predicted at 268th cycle which is a few cycles before the actual failure giving us enough time to prepare for it.

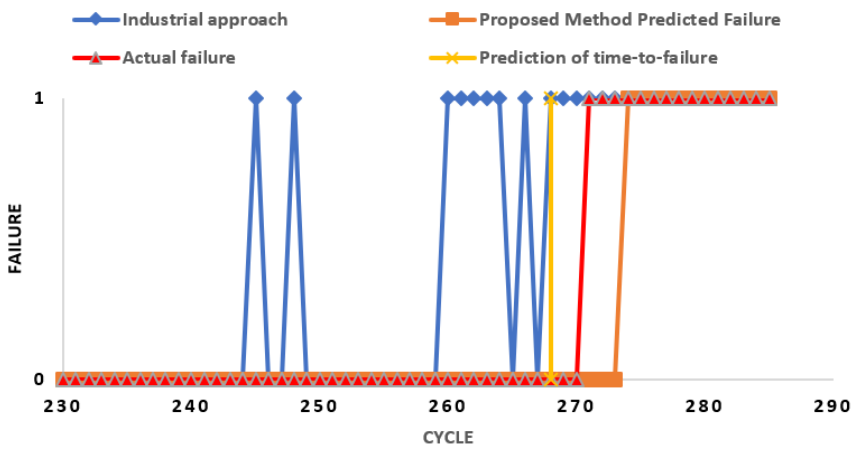

Figure 4.14: Comparison between our proposed and the current industrial approach on engine 82 of FD003

Similarly, we have compared the two approaches on engine 83 of the FD003 dataset (figure 4.15). The current approach predicted failure at 140th cycle when the actual failure starts at 167th cycle which leads to underutilization of the unit. On the other hand, our approach predicts failure from 161st cycle at 155th cycle and hence offers an opportunity for the better use of the unit over the current approach. Here, our proposed approach predicted failure much closer to the actual failure than the current industrial approach.

We have compared the two approaches on the complete test data of 20 engine failure trajectories. The results are illustrated in figure 4.16. The current industrial approach as stated earlier was giving many false alarms leading to conservative use of the units as we are losing significant capability by servicing them before they have reached their critical limits. Our approach generally

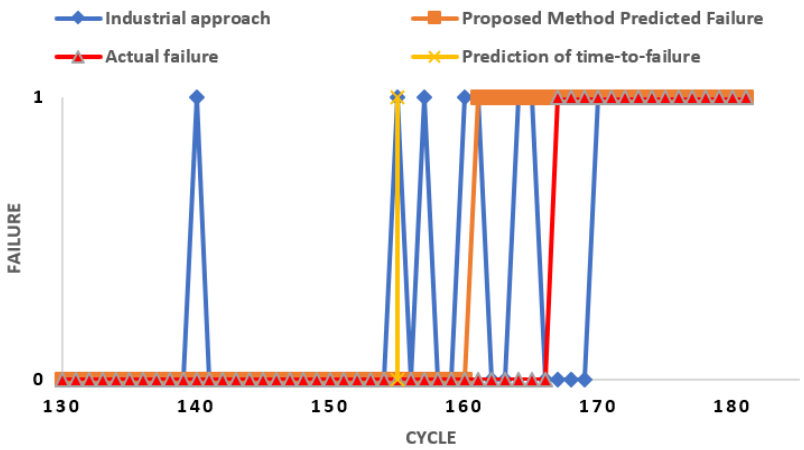

Figure 4.15: Comparison between our proposed and the current industrial approach on engine 83 of FD003

outperformed the current industrial approach and gave a better opportunity to utilize the units to their complete potential.

Engine No	Single parameter method	Proposed method		Actual start of failure	Comments
		Predicted at time step	Predicted failure step		
81	20	**328**	332	**333**	False alarm
82	**245**	**268**	274	**271**	
83	140	**155**	161	**167**	
84	109	**212**	217	**212**	False alarm
85	232	**249**	255	**252**	
86	306	**305**	314	**327**	
87	156	**154**	160	**158**	
88	67	**310**	313	**308**	False alarm
89	1	**198**	201	**193**	False alarm
90	155	**163**	168	**167**	
91	138	**139**	143	**142**	
92	**47**	**126**	136	**144**	False alarm
93	151	**151**	156	**157**	
94	**332**	**365**	372	**378**	
95	134	**142**	148	**152**	
96	**447**	**465**	473	**477**	
97	234	**260**	265	**261**	
98	271	**289**	295	**293**	
99	84	**114**	121	**131**	False alarm
100	124	**126**	136	**137**	

Figure 4.16: Comparison between our proposed and the current industrial approach on FD003

Thus, we demonstrated that our model in general performed better than the current industrial approach in predicting the time-to-failure of a gas turbine engine. We also plan to develop better

metrics of performance and quality for this comparison which will be done in collaboration with the industrial partners.

5. SUMMARY AND CONCLUSIONS

The developed prognostic technique showed great promise. Although RNN models developed were with basic architectures, the results observed were very promising. This prognostic technique is very relevant to the industries that use gas turbine engines in particular and any equipment that requires special care in general.

5.1 Contribution

We created a time-to-failure prognostic model using two standard machine learning techniques that showed potential for use on industrial equipment. We established supporting evidence for the proof of concept of the time-to-failure prognosis model. An approach that can be further improved with the availability of more sensor data to train on has been developed. We demonstrated the approach on the available datasets and the proposed technique generally outperformed the current industrial approach. We also plan to develop better metrics of performance and quality for this comparison which will be done in collaboration with the industrial partner.

5.2 Conclusions and Future work

Our prognostic technique showed a lot of promise in predicting the time-to-failure of a gas turbine engine and hence can be extended further with more complex neural network architecture. We look to improve upon these models and try other model architectures to improve the performance. We also plan to incorporate more features into the model e.g., Detection of sensor failures [23]. We plan to accomplish this via dimension reduction of the hypervolumes that bound a parameter set under the current operating condition. This concept is shown notionally for a three-dimensional parameter vector in figure 5.1. For example, if an operating point falls outside of the bounds of the current operating condition hypervolume (top plot of figure 5.1), it is necessary to rapidly test whether this was due to a sensor error. This can be accomplished via a leave-one-out approach, by constructing many lower-dimensional hypervolumes (lower plots) that indicate whether or not the abnormal behavior is linked to a sensor error or it is indeed an issue with the unit itself. If the issue

is with the sensor, relevant lower dimensional hypervolumes, with associated probabilistic edges can be used. The basis of an FMEA can be used to determine how critical the replacement of the erroneous sensor is. This should include an evaluation of the accuracy and confidence provided by lower-dimensional bounding hypervolumes with probabilistic support vector machines for accurate sensors [23]. We also plan to develop models that not only predict the failure but can also offer better insight to the technical personnel about the mode of failure by leveraging sensor predictions like in figure 5.2. Thus as mentioned before there is a lot of scope for this project which can be explored in future study.

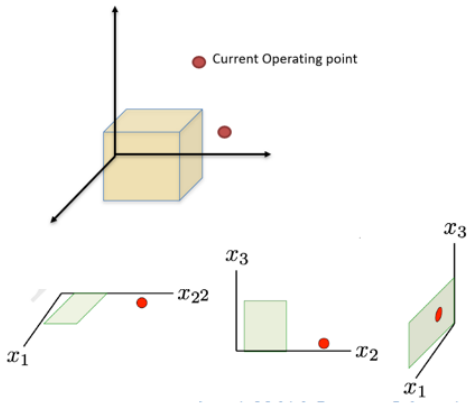

Figure 5.1: Sensor failure detection using multiple parameter information exploitation

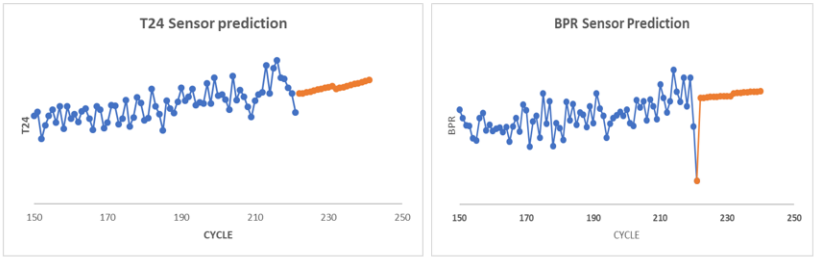

Figure 5.2: Sensor predictions to identify the modes of failure

REFERENCES

[1] A. D. Fentaye, A. T. Baheta, S. I. Gilani, and K. G. Kyprianidis, "A review on gas turbine gas-path diagnostics: State-of-the-art methods, challenges and opportunities," *Aerospace*, vol. 6, no. 7, p. 83, 2019.

[2] IATA's-MCTG, "Airline maintenance cost executive commentary," *Alton Aviation Consultancy MRO Forecast*, p. 14, 2019.

[3] Y. Lui, G. Zhao, X. Peng, and C. Hu, "Lithium-ion battery remaining useful life prediction with long short-term memory recurrent neural network," in *Annual conference of the prognostics and Health Management Society*, 2017.

[4] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, "Predicting remaining useful life using time series embeddings based on recurrent neural networks," *arXiv preprint arXiv:1709.01073*, 2017.

[5] L. Jayasinghe, T. Samarasinghe, C. Yuen, and S. S. Ge, "Temporal convolutional memory networks for remaining useful life estimation of industrial machinery," *arXiv preprint arXiv:1810.05644*, 2018.

[6] M. Tahan, E. Tsoutsanis, M. Muhammad, and Z. A. Karim, "Performance-based health monitoring, diagnostics and prognostics for condition-based maintenance of gas turbines: A review," *Applied energy*, vol. 198, pp. 122–144, 2017.

[7] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set," *NASA Ames Prognostics Data Repository*, 2008.

[8] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *2008 international conference on prognostics and health management*, pp. 1–9, IEEE, 2008.

[9] C. A. S. Team, "Airplane turbofan engine operation and malfunctions basic familiarization for flight crews," *CAST, https://www.cast-safety.org/pdf/engine_operation_text.pdf*, 2011.

[10] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[12] L. Wang, *Support vector machines: theory and applications*, vol. 177. Springer Science & Business Media, 2005.

[13] V. Vapnik and V. Vapnik, "Statistical learning theory wiley," *New York*, vol. 1, 1998.

[14] G. Wahba, *Spline models for observational data*, vol. 59. Siam, 1990.

[15] T. Trafalis, "Primal-dual optimization methods in neural networks and support vector machines training," *ACAI99*, vol. 5, no. 1, pp. 459–478, 1999.

[16] Y. B. Andrew Ng, Kian Katanforoosh, "Sequence models- deep learning specialization," *https://www.coursera.org/learn/nlp-sequence-models*, 2019.

[17] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[18] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[19] A. Karpathy, "The unreasonable effectiveness of rnns," *in http://karpathy.github.io/2015/05/21/rnn-effectiveness*, 2015.

[20] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, pp. 473–479, 1997.

[21] R. G. Nascimento and F. A. Viana, "Fleet prognosis with physics-informed recurrent neural networks," *arXiv preprint arXiv:1901.05512*, 2019.

[22] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[23] R. Swischuk and D. Allaire, "A machine learning approach to aircraft sensor error detection and correction," *Journal of Computing and Information Science in Engineering*, vol. 19, no. 4, 2019.