Acad Year
(21/22)

Project No.
B062

Remaining Useful Life predictions for electrical machines by physics-based machine learning

Remaining Useful Life predictions for electrical machines
by physics-based machine learning

Nicholas Sung Wei Yong

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING
NANYANG TECHNOLOGICAL UNVERISITY, SINGAPORE

Year (2021/2022)

Remaining Useful Life predictions for electrical machines by physics-based machine learning

SUBMITTED

BY

NICHOLAS SUNG WEI YONG

U1822850C

Supervisor: Dr Yang Feng (IHPC)

Dr Chow Wai Tuck (NTU-MAE)

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

A final year project report presented to Nanyang Technological University in partial fulfilment of the requirements for the Degree of Bachelor of Engineering (Aerospace Engineering) Nanyang Technological University

Year 2021/2022

Abstract

Prognostic systems are paramount to the Industry 4.0 revolution. They play an increasingly active role in improving systems' safety, reliability, and productivity in multiple industries. Centred on predicting the Remaining Useful Life through Health Index formulation, this paper presents a Physics-Based Machine Learning prognostics framework that imposes knowledge of the system's Health Index as loss functions. A case implementation on experimental information from 8 induction engines was conducted. This paper carefully assesses the performance of the physics-based machine learning model by comparing the predictions with available experimental data. The investigations reveal that the physics-based machine learning model can accurately improve the Remaining Useful Life predictions by more than 30 percent as compared to direct methods. Owing to the additional knowledge, less labelled training data is required, and the predictions are scientifically consistent with the known knowledge. The foray of physics-based machine learning to prognostics evidences the great potential of its application for broader applications in prognostics.

Keywords: Physics-Based Machine Learning, Prognostics, Remaining Useful Life

Acknowledgement

The fulfilment of this final year project and the experiential learning from it would not have been possible without the guidance of two key personnel. They are Dr Yang Feng and Dr Chow Wai Tuck.

Firstly, I would like to thank Dr Yang Feng for accepting my request in doing the final year project under his supervision. Although I was initially not equipped with the skillsets of machine learning, he provided me with the opportunity to expose myself to this field and learn. The patience, guidance, and advice of Dr Yang helped me immensely in comprehending the principles of machine learning and the thought process of tackling such projects from the perspectives of a Machine Learning Scientist. Furthermore, he always made time when I had to clarify my doubts.

Secondly, I would like to thank Dr Chow for his constant support throughout this project. He was always approachable and generous in allocating time to clarify my doubts while providing me with invaluable insights.

Therefore, I express my utmost gratitude to them for their guidance, patience, and insights throughout this final year project. Overall, it was a great pleasure working on this final year project despite the many hurdles and challenges. Although the learning curve was steep, it provided me with an experiential learning through the difficulties faced and overcome.

Table of Contents

Abstract 3

Acknowledgement 4

Table of Figures 7

List of Table 8

1. Introduction 9

1.1. Background 9

1.2. Motivations 10

1.3. Objective 11

1.4. Scope 11

2. Literature Review 12

2.1. Different Approaches to Maintenance 12

2.1.1. Reactive Maintenance 12

2.1.2. Scheduled Maintenance 13

2.1.3. Predictive Maintenance and Condition-Based Maintenance 14

2.2. General Prognostics Pipeline 16

2.2.1 Data and Pre-processing 16

2.2.2 Feature Engineering 17

2.2.3 Feature to RUL prediction 19

2.2.4 Performance Evaluation 25

2.3. Induction Motor 27

2.3.1 Component Faults 27

2.3.2 Failure Mechanism 30

2.3.3 Past Work Done 30

2.4. Machine Learning 33

2.4.1. Neural Network 33

2.4.2 Physics Based Machine Learning 36

3. Methodology 39

3.1. Data and Pre-processing 39

3.2. Feature Engineering 40

3.2.1. Feature Extraction 41

3.2.2. Feature Selection 41

3.3. Feature to RUL Prediction 43

3.3.1. Physics-based machine learning (Kinematics Example) 43

3.3.2. Physics-based machine learning (Feature -> HI) 51

3.3.3. Ensemble Approach (HI -> RUL) 65

3.4. Performance Evaluation 69

4. Results 70

5. Conclusions 72

6. Future Work 73

References 74

Table of Figures

Figure 1. Bathtub Curve for Product Failure Behaviour 13

Figure 2: Comparison of different maintenance approaches 15

Figure 3: General Prognostics Pipeline 16

Figure 4: Components of Three-Phase Induction Motor 27

Figure 5: Problems in a Three-Phase Induction Motor 28

Figure 6: Arrangement of bearing and shaft 28

Figure 7: Stator and Rotor 29

Figure 8: Traditional Neural Networks structure 33

Figure 9: Illustration of weights, bias and activation function 34

Figure 10: Loss Function against weight 35

Figure 11: Physics-Informed Neural Network structure 37

Figure 12: Timeline for collecting steady state data 39

Figure 13: Trajectory of ball 43

Figure 14: Results of Kinematics Example 46

Figure 15: Comparison of results for extrapolation test data 47

Figure 16: Physics-Based Machine Learning Model (Rules) 51

Figure 17: ReLu function 52

Figure 18: Physics-Based Machine Learning Model (Boundary Conditions) 57

Figure 19: Overall Physics-Based Machine Learning Model 58

Figure 20: Health Index against Cycle for Motor 1 59

Figure 21: Interpolation of HI to obtain RUL 66

Figure 22: Extrapolation of HI to obtain RUL 67

Figure 23: Ensemble approach 68

List of Table

Table 1: Properties of signals.. 18
Table 2: Comparison of data-driven prognostics and model-based prognostics....... 23
Table 3: Different Types of Prognostics Performance Metrics 25
Table 4: Maintenance practice for induction motors 31
Table 5: Details of missing values in the motors’ data. 40
Table 6: Details of the Time Domain Features.. 41
Table 7: Feature Vector ranked according to Fisher's Ratio..................................... 42
Table 8: Training loss value of each motor 65
Table 9: Performance Evaluation of RUL predictions 70

1. Introduction

This chapter introduces prognostics and highlights the motivation, objectives and scope pertaining to the final year project.

1.1. Background

Prognostics is an engineering discipline that aims to predict the time a system will no longer operate normally. This predicted time is known as the remaining useful life (RUL), and the prediction is achieved via analysis of data through technical methods such as data-driven, model-based and hybrid techniques. Data collected are electrical signals, vibration signals, temperature, etc. These data are easily collected through sensors such as accelerometers, gyroscopes, and electrical sensors. Collected data and methods such as data processing, data mining and statistical analysis are used to evaluate the data [1].

These methods allow prognostics to predict the system's future performance by determining the deviation from its expected operating conditions. Efficient prognostics methods can lower the chances of severe breakdown, thus elevating the level of safety of these systems. Furthermore, they could significantly reduce the high expense of regular maintenance activities [2].

The capabilities of prognostics have made it an essential pillar in the evolution of Industry 4.0, which comprises big data, intelligent machines, and smart manufacturing. Industry 4.0 is essentially the integration of various technological developments pertaining Internet of Things (IoT), cloud computing, computational intelligence, and cyber-physical systems (CPS) [3]. The integration allows for continuous acquisition of data, development of a central database for local access and analysis while reducing latency. Furthermore, one of Industry 4.0 aims is to enhance factory processes by increasing efficiency, flexibility, and productivity to manufacture goods at higher quality and reduced costs. As such smart factories with smart sensors and self-sustainable intelligent machines via prognostics methods are in the early stages of development [4]. The development is also made possible with the utilization of industrial big data in prognostics. Structured data such as sensor signals and vibration signals can be acquired and analysed to study machine performance [5].

1.2. Motivations

Prognostics models are difficult to develop as it highly depends on the intrinsic complexities and uncertainties of the system. Existing prognostic models can be categorised as model-based approach, data-driven approach, and hybrid approach.

Model-based approaches leverage the current understanding of physical processes to predict RUL. With accurate numerical models of the processes and correct representation of the interactions between the components, model-based approaches can predict accurate RULs [6]. However, model-based approaches cannot be generalized to all systems because each system requires a particular model and developing such models can be very expensive and time-consuming. In addition, they cannot accommodate an overly complexed system and underlying models are generally simplified [7].

In comparison, data-driven approaches use either conventional numerical techniques or machine learning methods to detect changes in the state of the system to predict the RUL. Conventional numerical techniques that have been explored include linear regression and Kalman filters. Additionally, machine learning techniques such as neural networks, decision trees, and support vector machines have frequently been used too [8]. Data-driven approaches work well for complex systems and to analyse intermittent faults by detecting changes in the measured data. However, they may require a larger training data set and often result in broader confidence intervals.

Lastly, hybrid approaches leverage the benefits of both model-based and data-driven approaches to enhance prediction performances. This type of approach incorporates physical domain knowledge with data-driven models to predict degradation trend at various operating conditions and failure modes. Consequently, hybrid approaches can reduce the prediction errors induced by purely data-driven methods and ensure the predictions adheres to the system's physics. [9].

Although past studies have been conducted to develop an effective hybrid model for different prognostics problems, not much has been done to address the lack a generalized hybrid approach when there is little knowledge of the system. Details about the past studies with the respective hybrid models will be elaborated in the literature review section.

1.3. Objective

This paper investigates the feasibility of using a physics-based machine learning framework to predict the Remaining Useful Life for electrical machines.

1.4. Scope

A database of 8 induction motor provided by University of Tennessee will be pre-processed. The improved signals then undergo feature engineering to convert the time-series raw signals into an optimal number of discriminative features. In two-stage modelling, the relationship between feature to RUL is built. This is done through using a physics-based machine learning framework to predict Health Index (HI) from features. The physics-based machine learning framework will present three rules of the system's Health Index and methods to impose them in loss functions. After which, the predicted HI is mapped to RUL and the final RUL prediction is obtained through the multi-model ensemble. Finally, the performance of the RUL prediction is evaluated.

2. Literature Review

This chapter first presents the most popular maintenance approaches (Reactive Maintenance, Scheduled Maintenance, Predictive Maintenance and Condition-Based Maintenance), underlining the advantages and disadvantages of each approach concerning the costs and disruption to the system's operation. Secondly, it illustrates the general prognostics pipeline (1. Data and Pre-processing 2. Feature Engineering 3. RUL Prediction 4. Performance Evaluation) and states the function of its main subprocesses in the context of predictive maintenance. Lastly, it introduces machine learning and specifically physics-based machine learning since its framework is presented in this paper.

2.1. Different Approaches to Maintenance

An effective maintenance approach is imperative for reducing expenses and the system's downtime. Adopting a particular maintenance approach depends mainly on the demands of the organization. Since each maintenance policy has its benefits and drawbacks, we will evaluate four different maintenance approaches: Reactive Maintenance, Scheduled Maintenance, Condition-Based Maintenance, Predictive Maintenance and Condition-Based Maintenance, according to its cost and disruption to the system's operation [10].

2.1.1. Reactive Maintenance

Reactive maintenance conducts repairs or replacement instantly after the system breaks down. The two main benefits of reactive maintenance are the low amount of supervision, resources, and expenses to keep the systems operating and the maximised operational time since the system will remain operational until the system breaks down [11]. However, this approach is risky from many outlooks. Firstly, it is potentially dangerous from a safety perspective as it allows the system to degrade to its worst health state which can result in irreparable failures (i.e. cracks and wear and tear) that are expensive and require a long time to repair. Secondly, from an operational standpoint, a breakdown will provoke unanticipated stops which may disrupt the output of the system. Lastly, from an energy conservation perspective, a fault may cause the system to consume more energy. For example, an induction motor with an existing fault will cause it to require more current to keep the torque constant. Therefore, reactive maintenance can lead to conspicuous costs from reparations of severe failures, relatively large unplanned system downtimes and higher energy consumption.

2.1.2. Scheduled Maintenance

Scheduled maintenance conducts maintenance operations according to a pre-defined maintenance schedule. The objective is to minimise failure and reduce the occurrence of unexpected downtimes by conducting maintenance activities under normal conditions. However, scheduled maintenance depends heavily on a customised schedule specific to the properties of the system. It requires experts with good knowledge of the system for thorough evaluation of the systems' failure behaviour and detailed analysis of the components to predict the next time of failure accurately. This evaluation of failure behaviour usually results in the bathtub curve [12] illustrated below.

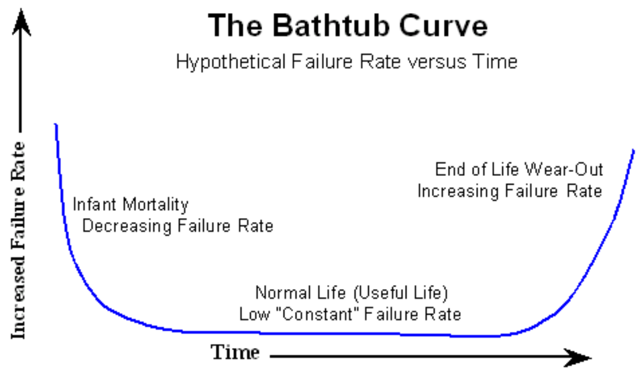


Figure 1. Bathtub Curve for Product Failure Behaviour

The bathtub curve in Figure 1 splits the lifecycle of any system into three parts. The first part is the "infant mortality" phase, and failures commonly occur immediately after installation due to mistakes made during installation or incompatibility between components. The second part is the "useful life" phase, and the system is assumed to function normally, and its failure probability is low and constant. The third part is the "end of life wear out" phase, where the rate of failure increases due to the system's natural degradation and wear-out.

The primary benefit of scheduled maintenance is the significant reduction of unplanned downtime. Additionally, the repair costs are significantly less than those in reactive maintenance since the systems will not operate until their breaking point. However, due to numerous factors such as the unpredictability of the complex systems, environmental conditions and internal malfunctions, there bears a high risk that the system's degradation behaviour deviates from its predicted trend. Hence, conducting maintenance based on a pre-defined schedule may incur more than reactive maintenance due to redundant repairs.

2.1.3. Predictive Maintenance and Condition-Based Maintenance

Predictive maintenance and condition-based maintenance techniques help determine the system's condition to estimate when maintenance should be conducted. This approach provides a good compromise between the regularity of maintenance and costs because maintenance is only performed when warranted [13].

The difference lies in their corrective measures when a faulty system state is detected. For condition-based maintenance, maintenance activities occur immediately after the fault is detected. The drawback of immediately replacing or repairing the component associated with the fault is in shortening the remaining lifespan of this component. A fault is a state of the component that deviates from the required specifications, but the component is often still usable and could continue operating for an extended duration without affecting other machine parts. Hence, it is often advised to predict when the fault will grow to failure and conduct maintenance then. Furthermore, from an operational standpoint, a breakdown will trigger unanticipated stops, which may disrupt the output of the system. In comparison, predictive maintenance aims to predict the RUL of the system to indicate when maintenance should be conducted. The benefit of predictive maintenance is the lower maintenance costs compared to condition-based maintenance since each component can be fully utilised without compromising safety and efficiency [14].

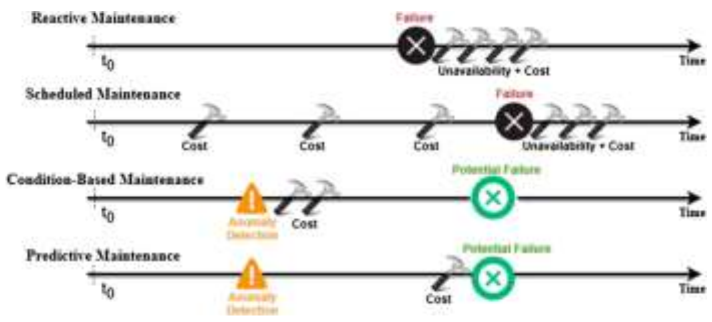


Figure 2: Comparison of different maintenance approaches

Figure 2 summarises the maintenance approaches presented above by demonstrating the cost and general approaches of each approach. Evidently, predictive maintenance is the most cost efficient.

2.2. General Prognostics Pipeline

Diagnostics and predictive maintenance are the foundations of effective predictive maintenance. Diagnostics primarily allows for fault identification and detection, while prognostics permit the prediction of RUL. Both diagnostics and prognostics are complementary since diagnostics detect the faults present and establish the current system's health which aids in prognostics. This section will focus on the main components constituting the general prognostics pipeline as illustrated in Figure 3, from data acquisition to performance evaluation of RUL.

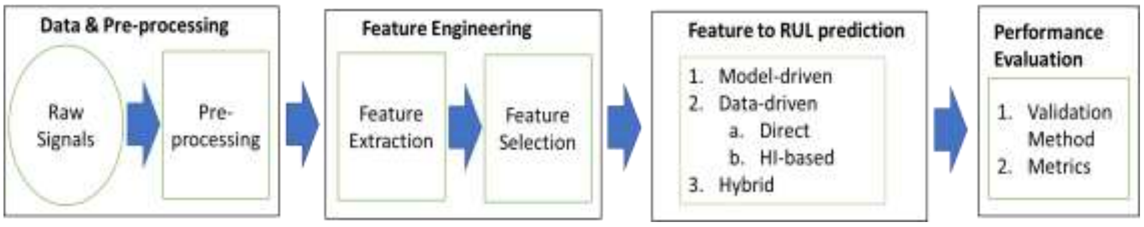


Figure 3: General Prognostics Pipeline

2.2.1 *Data and Pre-processing*

The first phase of the prognostics pipeline involves data collection and pre-processing. Data collection involves selecting appropriate types of sensors and devices, fixing them in suitable positions and determining the optimal sampling rate for data collection. Data collected can be presented in many forms such as electrical signals, vibration signals, temperature, etc.

Data can be collected when the sensors are positioned. These data cannot be processed by AI algorithms due to its complex data structure. Therefore, an additional data pre-processing step must be executed to transform raw data into an understandable format. Data pre-processing is mainly used to check the data quality which can be verified through its accuracy, completeness, consistency, timeliness, and interpretability [15]. The process usually cleans the data, mitigates the effects of noise, or reshape them so that data analysis techniques can interpret their new format.

The generated data are cleaner than the originals, but they may still contain substantial redundant information leading to misleading results. This motivates the next section covering feature engineering techniques which is used to reduce the dimensionality of the data and retain only the more significant information.

2.2.2 Feature Engineering

The raw measurements provided by sensors cannot be directly connected with the Health Index of the system or its RUL. This is because raw data usually has high data rate, information redundancy and contain a significant amount of noise due to environmental factors or sensor malfunctions. Furthermore, the form of data is often collected as complex time series, typically containing highly redundant information that hides the few but important features. As such, once raw signals are pre-processed, feature extraction is executed to obtain valuable features. Next, feature selection is performed where the most informative features are appropriately selected. After these steps have been achieved, the extracted features can be used as input into the prognostics model, which will predict the system's RUL. The following parts will cover some of the main techniques used for feature extraction and feature selection.

2.2.2.1 Feature Extraction

Feature extraction refers to the process of converting raw data into features that can be processed while keeping the information in the original data set [16]. These numerical features should be more informative and enhance the performances of the follow-up predictive models.

Manual or automated methods can accomplish feature extraction. Manual feature extraction involves identifying and describing the essential features for a specific problem and extracting those features. A generalised method for the feature extraction does not exist, and various context-dependent variables must be considered. In many cases, a sound knowledge of the problem can help make educated decisions about which features could be significant. Intensive research has developed feature extraction methods for different tasks based on the attributes of the data, its application, and the algorithmic and efficiency requirement [17]. For example, methods such as the SIFT [18] and SURF [19] algorithms are typically used for image recognition. In comparison, methods like mel-cepstral coefficients are generally used for speech recognition [20], [21]. In comparison, automated feature extraction utilizes technical algorithms to extract features automatically from signals without the need for manual supervision. This technique can be advantageous for quickly transferring raw data to developing machine learning algorithms. For example, wavelet scattering uses an Artificial Neural Network-based algorithm for unstable signals [22].

In the context of prognostics, relevant features are often selected according to the properties of the signals (Table 1).

Table 1: Properties of signals

Properties of signals	Example
Physical nature	Temperature, Pressure, Voltage, Acceleration
Dynamics	Cyclic, Periodic, Stationary, Stochastic
Sample Value Discretization	Continuous, Discrete
Sampling Frequency	

Most prognostics raw data is collected in the form of time series, and the extracted features can be categorized into the time domain, frequency domain, and time-frequency domain [23].

There are many time-domain feature extraction techniques that aim to reduce the number of features in the dataset by creating new features and removing the original features. The simplest method is though the fundamental statistical indicators such as mean and standard deviation. There are also other traditional methods such as auto and cross-correlation, convolution, fractal analysis [24] and correlation dimension [25]. Secondly, the frequency-domain extraction technique is often a spectral analysis of the signal. This is used to determine the frequency characteristics of a frequency domain signal in terms of quantities such as the energies and eigenvalues of the wave. Lastly, time-frequency domain feature extraction techniques often use signal representations as training data in machine learning and deep learning models. Traditional techniques to time-frequency domain feature extraction are short-time Fourier transform, Wavelet transform, and empirical mode decomposition. Other machine learning techniques such as Principle Components Analysis, Independent Component Analysis and Linear Discriminant Analysis have also been used analyse the way a signal's frequency components vary as functions of time.

2.2.2.2. *Feature Selection*

As explained above, the objective of feature extraction is to obtain features that are informative for the problem. This is achieved by creating new features from the existing dataset. This new set of features should be able to explain the data better. The next step is feature selection, which aims to reduce the number of the feature by selecting a subset of features that are most relevant for a specific objective. This is achieved through ranking the new features obtained from feature extraction based on significance and then keeping on the important features. Feature selection methods can be categorized into filters and wrappers methods [26].

The objective of filters is to find an optimal number of features based on the information content of the features. The selected features will be generally usable by most features to RUL prediction techniques. Feature selection techniques are based on the calculation of information-theoretic quantities, such as the Fisher's Ratio, Pearson coefficient or information gain. For instance, Minimum-Redundancy-Maximum-Relevance utilizes the highest statistic association between features to obtain the optimal subset of features.

Wrapper-based methods differ from filter methods in their benchmarks for assessing the distinctiveness of a subset of features. They use machine learning algorithms to get evaluate the distinctiveness of features where the performance is usually measured by accuracy or loss function. Wrappers are generally better since the machine learning algorithms are modified according to the task.

2.2.3 *Feature to RUL prediction*

After pre-processing the raw signals and conducting feature engineering to identify the distinctive features, these features are input into prognostic models to predict the RUL. This section will cover the three types of prognostic models which are data-driven approaches, model-based approaches, and hybrid approaches.

2.2.3.1 Data Driven prognostics

Data-driven approaches employ statistical models and machine learning techniques to detect system state changes and predict the RULs [27].

For statistical models, the goal is to find the probability density function of the RUL that is conditional on the history of operational profiles. Available past observed data pass through the model to estimate the RUL in a probabilistic way. These models heavily depend on their inherent mathematical properties and do not incorporate any physics or engineering discipline. Before the prevalence of machine learning, regression-based models were widely accepted to predict RUL for their simplicity [28] [29] [30]. In these models, the system's key features are mapped to the Health Index of the system. Next, the RUL can be estimated by observing, trending, and predicting the key features at a predefined threshold. Usynin et al. demonstrated this concept through a simple linear regression model [31]. Lu et al. presented a general nonlinear regression model to characterise the degradation path [32]. They characterised the degradation path with the general shape of the degradation and included other parameters to account for the randomness of the data. The RULs were predicted based on the failure threshold set. Whilst Lu et al. investigated regression models that calculated the RUL for a combination of components, Gebraeel et al. proposed a RUL prediction model intended for a single operating system using sensor-based monitoring signals [33]. This model incorporates previous and real-time data to enhance the dynamics of original regression methods, which only considered historical data. Apart from regression-based models, other statistical models such Wieners processes are used for degradation paths that vary bi-directionally over time [34]. Gamma processes are used for degradation paths that are monotonic and uni-directional [35]. This is often the case in wear and tear and fatigue crack propagation, where the degradation is gradual in small positive increments. The various statistical models analyse all heavily rely on the abundance and completeness of the data. This challenge in often faced by newly commissioned systems that do not possess any failure data. Additionally, techniques such regression-based models that use threshold for RUL predictions face a severe challenge of determining the failure threshold level. Lastly, statistical methods are not able to consider external environmental conditions and failure modes of the system [36].

For machine learning techniques for RUL predictions, they can be broadly categorised into Artificial Neural Networks (ANNs), Support Vector Machines (SVMs) and Decision Trees (DTs).

For ANNs, its first application was by Shao et al. Shao's team evaluated a three-layer neural network to predict the Health Index of a bearing [37]. Since then, much work has used and modified neural networks for different applications. Many have also strengthened the position that ANNs is superior to statistical methods. For example, Wang et al. proposed a Dynamic Wave Neural Network (DWNN) framework on cracked bearings [38]. Through five prognostics performance evaluations, they proved that DWNN method outperforms any statistical method (i.e. autoregression). The use of ANNs has proven to be effective for a large amount of noisy, numerical, and temporal data. It is also effective when the physical, statistical, or deterministic model is not known or impractical [39]. This scenario is often faced when the system is too complex or when it has multiple failure modes.

Secondly, for SVMs, they are considered supervised learning models capable of classification and regression. SVMs are especially effective because of an SVM training algorithm primacy in classifying unmarked data into pre-defined categories. Additionally, SVMs can accept small training data sets and multi-dimensional data. Hence, in prognostics, SVMs can be effectively used to process a high number of features and classify them as healthy against faulty. Huang et al. promoted a detailed evaluation of the past work done in SVM based RUL estimation [40]. In their work, they highlighted the ability of SVMs in mapping the Health Index and RUL of the system. SVMs have been established effective in different applications such as bearings [41], batteries [42], and aircraft engines [43].

Lastly, DTs are less commonly used for prognostics. Decision trees are widely accepted because they are simple to interpret and will be able to provide important insights for each outcome. In prognostics, decision trees have been validated to been effective for bearing [44], batteries [45] and turbofan engines [46]. However, the decision tree is ultimately limited by the availability of data since it requires a substantial training data set for it to act as a classifier between normal and faulty condition.

Based on the past work done, it is clear that data driven approaches are often used when there is insufficient understanding of the system or when the system is too complex. This is often the case for large systems where the interaction between the components of the whole system cannot be accounted for. Since no prior experiment is required to understand the system, the data-driven model can often be deployed quickly and cheaper compared to other approaches. This is often practical for small components such as bearings, gears and belts. However, they often have larger uncertainties and require a substantial amount of training data.

It is important to note whether the data driven approach uses Health Index formulation for the RUL prediction. For direct RUL predictions which does not use HI formulation, input data is directly used for the prediction of RUL. These methods are simple to implement but lack the understanding on the health state of the system. In comparison, HI formulation first maps the features to HI, then HI is used for RUL prediction. This two-stage prediction allows the incorporation of domain knowledge and has proven to be more effective than direct RUL predictions.

2.2.3.2 Model-based prognostics

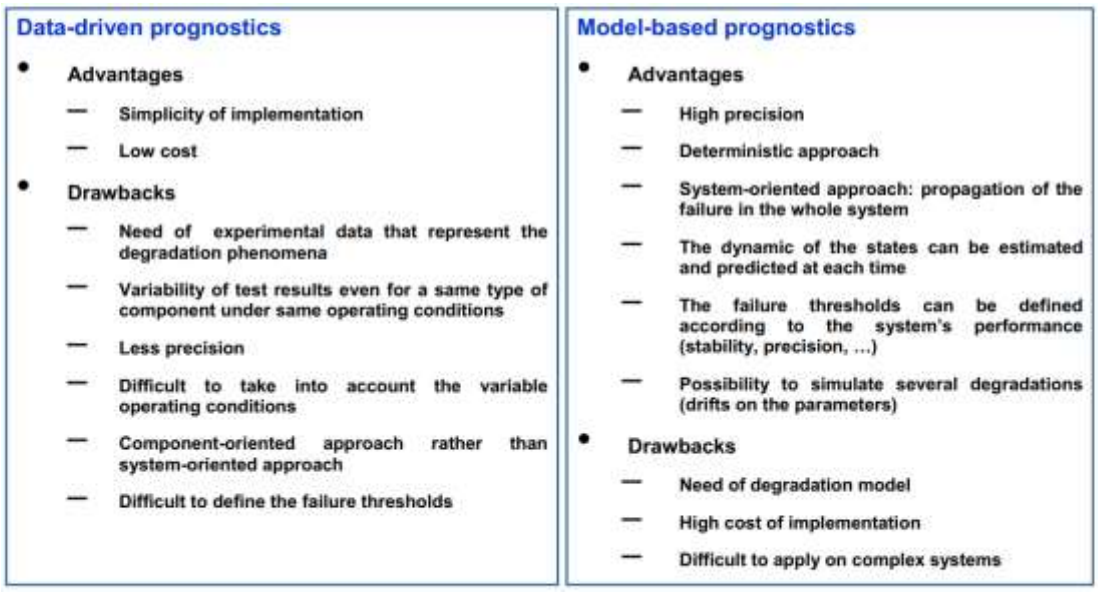
The model-based approach employs physical and mathematical models to predict the RUL of the system. These models incorporate domain knowledge of the system and utilise the concept of residual. The residual is the difference between the measured output of the system and the predicted output of the models. If the residual is large, it represents a fault in the system since the deviation between the actual measurements and the expected healthy output of the system is significant. On the contrary, if the residual is small, it represents that the system is healthy, and the slight deviation might be due to noise and modelling errors [47].

Model-based prognostics can be broadly categorised into macro and micro levels. A macro-level is applicable to an entire system, and the model will define the relationship between input features. For example, the National Aeronautics and Space Administration uses a physics-based lumped parameter state-space model for their water recycling system prognostics [48]. In contrast, a micro-level model-based approach is often applied to the damage propagation of small systems where the operational conditions are mapped to the degradation of the system through dynamic equations. The applications of these are for bearings [49], batteries [50], cracks [51] and pumps [52].

Model-based approaches often face the challenge of over-simplifying the application through making multiple assumptions. If the assumptions made cannot accurately represent the model, it can severely affect the results. Additionally, since a model-based approach depends heavily on the physical understanding of the application, in-depth research of the problem can be a long, expensive, and manual process.

The advantages and drawbacks of data driven prognostics and model-based prognostics can be summarised in the table below.

Table 2: Comparison of data-driven prognostics and model-based prognostics



Data-driven prognostics

- Advantages
  - Simplicity of implementation
  - Low cost
- Drawbacks
  - Need of experimental data that represent the degradation phenomena
  - Variability of test results even for a same type of component under same operating conditions
  - Less precision
  - Difficult to take into account the variable operating conditions
  - Component-oriented approach rather than system-oriented approach
  - Difficult to define the failure thresholds

Model-based prognostics

- Advantages
  - High precision
  - Deterministic approach
  - System-oriented approach: propagation of the failure in the whole system
  - The dynamic of the states can be estimated and predicted at each time
  - The failure thresholds can be defined according to the system's performance (stability, precision, ...)
  - Possibility to simulate several degradations (drifts on the parameters)
- Drawbacks
  - Need of degradation model
  - High cost of implementation
  - Difficult to apply on complex systems

2.2.3.3 Hybrid prognostics

Hybrid approaches combine both data-driven approaches and model-based approach to leverage on the strengths of both to enhance the accuracy of RUL prediction. This type of approach has been extensively researched as their methods can be classified into the following classes [9]:

1. Use data-driven approach to map raw data to health state, and then use model-based approach to predict the remaining useful life.

As mentioned previously, the use of Health Index formulation allows for the understanding on the health state of the system. In this class of hybrid approaches, a data driven method is used to map the relationship between the system health state and the appropriate features. The health state can be gauged by comparing anomalous readings to the healthy baseline. Thereafter, a model-based approach is used to map the system health state to the remaining useful life. This model-based approach is selected based on the degradation path of the system (i.e. fault initiation or fault propagation model is used according to the deterioration status)

2. Use data-driven approach to predict the future measurements, and then use model-based approach to predict the remaining useful life.

This class of hybrid approach is often used when there is insufficient data, therefore, a data-driven approach is required to predict the future measurements. It has proven useful for long term predictions especially when the degradation does not follow a fault growth model. However, intuitively, if the prediction of the future measurement is poor, the remaining useful life prediction by the model-based approach will also be consequently inaccurate.

3. Use data-driven approach and model-based approach to predict the remaining useful life and combine their predictions

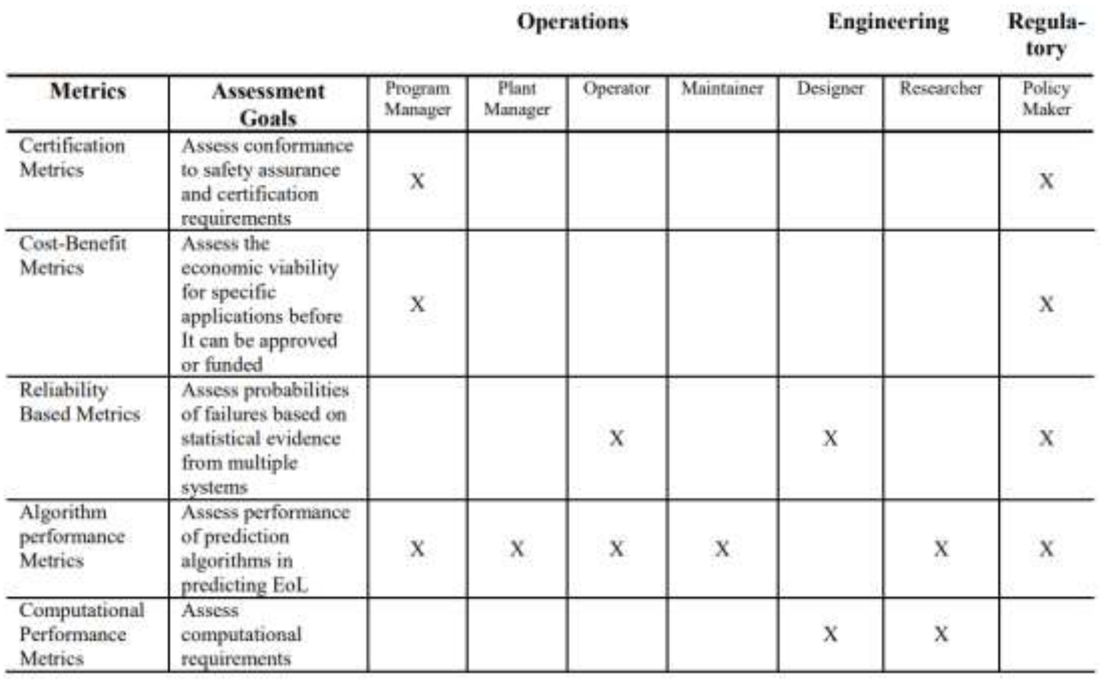
This class of hybrid approach simultaneously uses both the data-driven approach and model-based approach to predict the remaining useful life of the system. By combining their prediction, it can reduce the prediction uncertainty. However, similar to model-based approach, they require a detailed examination of the system due to the use of model-based approach.

Given the summary of the different classes of hybrid-based approach, the proposed physics-based machine learning model can also be classified as a hybrid based approach as it incorporates domain knowledge into data-driven approaches. However, this novel approach does not fall into any of the above classes. The physics-based machine learning will incorporate domain knowledge on the Health Index into the loss function of the neural network. Therefore, when the features are mapped to Health Index, the Health Index must adhere to the domain knowledge.

2.2.4 Performance Evaluation

Prognostic performance evaluation is imperative to determine if a prognostics system is accurate. There are many different types of prognostics performance metrics created for the various purposes of prognostics and the end-user. The diverse performance metrics can be summarised below (Table 3) [53].

Table 3: Different Types of Prognostics Performance Metrics



		Operations				**Engineering**		**Regula-tory**
Metrics	**Assessment Goals**	Program Manager	Plant Manager	Operator	Maintainer	Designer	Researcher	Policy Maker
Certification Metrics	Assess conformance to safety assurance and certification requirements	X						X
Cost-Benefit Metrics	Assess the economic viability for specific applications before It can be approved or funded	X						X
Reliability Based Metrics	Assess probabilities of failures based on statistical evidence from multiple systems			X		X		X
Algorithm performance Metrics	Assess performance of prediction algorithms in predicting EoL	X	X	X	X		X	X
Computational Performance Metrics	Assess computational requirements					X	X	

Each of these metrics depends on the final target audience. In terms of the research field, algorithm performance metrics are often used to assess the performance of prognostic techniques in predicting the RUL. These metrics can be classified into accuracy and precision. Accuracy is defined as the degree of closeness of the predictions to the actual failure time. Precision is defined as the spread of the projections performed simultaneously. This section will focus mainly on the accuracy of the RUL predictions therefore it will cover a list of error-based metrics.

Error-based metrics use the assessment of error (deviation between actual RUL and predicted RUL) to assess the performance. The prevailing error-based metrics are Root Mean Square Error and Mean Absolute Percentage Error. These metrics are simple and easy to understand. There are also other complex metrics such as:

1. Confidence intervals which measure an interval that contains a set percent of the predicted RUL at the same time index

2. Prognostics Horizon which measures the difference between the time index i when the predictions first meet the specified performance criteria and the time index fo the last cycle

3. α-λ Performance which measures whether the prediction falls within specified limits at particular cycles.

2.3. Induction Motor

The induction motor is the most common type of motor in residential, commercial, and industrial settings. It can be found in refrigerators, washing machines and furnaces, as well as conveyors, pumps, winders, wind tunnels and other industrial equipment. Its wide use in various applications can be attributed to its versatility and high reliability. However, in some industrial settings, induction motors are subjected to harsh environments such as extreme thermal and environmental stresses, mechanical damages, and adverse electrical conditions. Under these conditions, the potential cost incurred due to operational downtime could be significant. Hence, much research has been dedicated to the field of prognostics in induction motors to minimise the maintenance cost involved. This section will cover the components that usually fail, the main failure mechanisms of a three-phase induction motor and the work done in maintaining a three-phase induction motor.

2.3.1 Component Faults

An induction motor comprises of many different components (Figure 4).

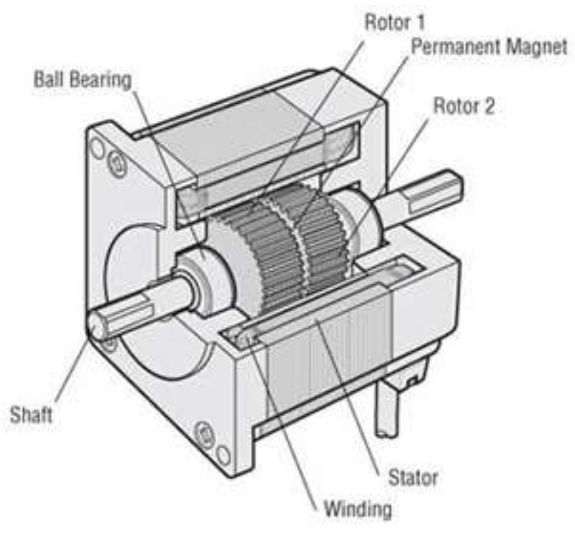


Figure 4: Components of Three-Phase Induction Motor

The usual faulty components for a three-phase induction motor are the bearings, stator and the rotor as seen in the pie chart below [54].

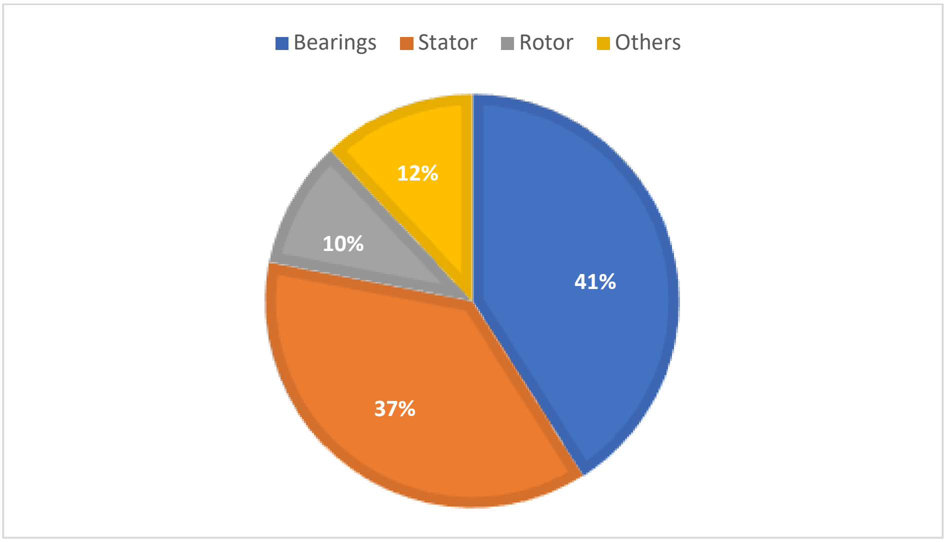


Figure 5: Problems in a Three-Phase Induction Motor

Based on Figure 5, it can be observed that the components that is most likely to fail is the bearings. Bearings provide the necessary support for the motor shaft and constraints the moving parts to the desired motion mode (Figure 6).

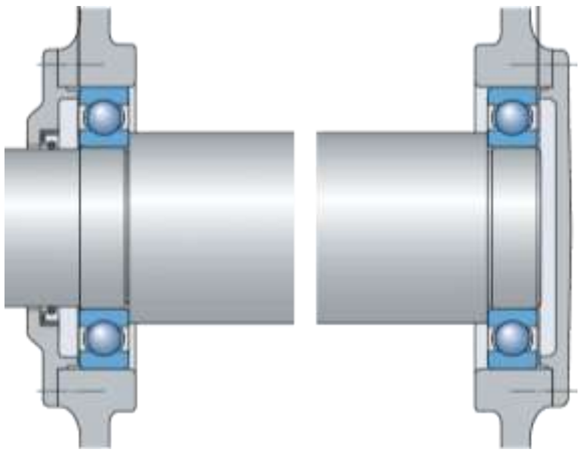


Figure 6: Arrangement of bearing and shaft

Bearings often fail due to the following reasons [55]:

- Lubrication Failure
- Mechanical Stresses
- Contamination & Corrosion
- Misalignment
- Distorted Components
- Poor Assembly
- Fatigue
- Inadequate Internal Clearance

Based on Figure 5, the other components that are most likely to fail is the stator and rotor. These two components work together to create torque in the induction motor. The stator is the stationary part of the induction motor whilst the rotor is the rotational part.

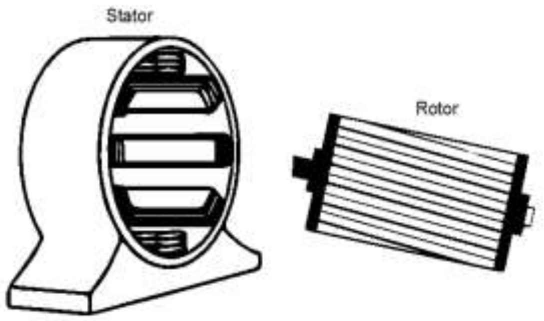


Figure 7: Stator and Rotor

When an alternating current is applied to the stator coil winding, it changes the polarity of magnetic field in the stator poles. The stators changing magnetic field cuts across the rotor's conductors and induces a current. This current creates a magnetic field around the rotor conductors. Hence, when the magnetic field from the stator interacts with the induced magnetic pole in the rotor, the rotor starts to spin. Since the rotor is attached to the motor shaft, the motor shaft starts to spin too, creating the torque for the induction motor.

The problems faced by stator windings are overheating, contamination, winding errors, and mechanical problems. The problems faced with rotor are broken broken bars or broke end rings, rotor misalignment and imbalance.

The multitude of possible problems that can induce a bearing, rotor and stator fault highlights the significance of proper maintenance to ensure that each component continues to operate under normal conditions. A fault can dramatically impact the entire system to fail as failure often results in damage to the surrounding equipment.

2.3.2 Failure Mechanism

The root cause of many failures is when the motor specification does not meet the demand of the operation. Apart from considering the usual parameters (voltage, frequency, speed, torque, power, duty cycle, etc.), other variables such as the mechanical, electrical and environmental conditions have to be assessed before choosing the suitable motor. For example, the bearing often faces some overlooked mechanical problems (successive overloads, pulsing load, repeated departures, vibration).

In terms of the electric conditions, the failures can result from the motor's electrical power system characteristics or the load feeder. These often occur when there are slow fluctuations of voltage which causes loss of stop power and when there is brusque fluctuations of voltage can result in failure in isolation.

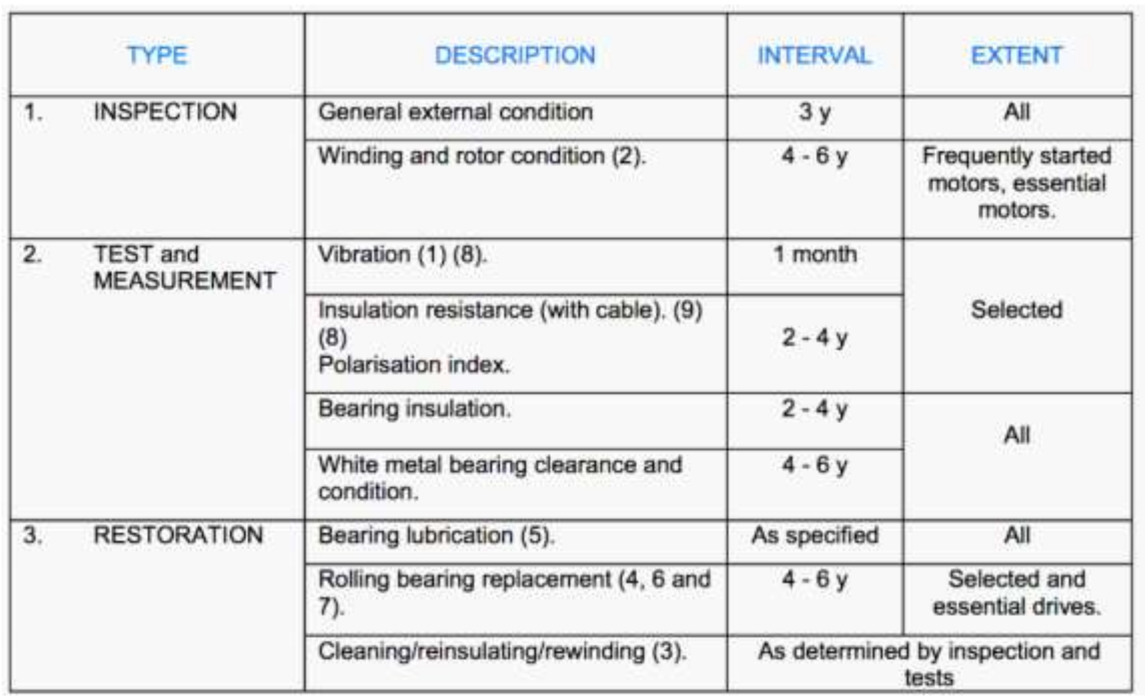
In terms of the environmental conditions, the failures can result from the characteristics of the machine's process. For example, high temperatures can damage components in isolation and humidity and pollution can cause imperfections and contamination of the isolation.

Evidently, the failures in electric machines depend on the specifications of the motor and the environment. In evaluating the different failure mechanisms, they serve as indicators for incipient faults for maintenance.

2.3.3 Past Work Done

The current maintenance procedures on induction motors can be divided as either scheduled maintenance or preventive maintenance. Induction motors are often subjected to harsh environmental conditions, onerous starting duty, and high structural vibration; hence many companies conduct scheduled maintenance for the motor and their components to preserve the motor’s lifespan and its efficiency. The scheduled maintenance practice for induction motors can be summarized in the table below.

Table 4: Maintenance practice for induction motors



TYPE	DESCRIPTION	INTERVAL	EXTENT
1. INSPECTION	General external condition	3 y	All
	Winding and rotor condition (2).	4 - 6 y	Frequently started motors, essential motors.
2. TEST and MEASUREMENT	Vibration (1) (8).	1 month	Selected
	Insulation resistance (with cable). (9) (8) Polarisation index.	2 - 4 y	
	Bearing insulation.	2 - 4 y	All
	White metal bearing clearance and condition.	4 - 6 y	
3. RESTORATION	Bearing lubrication (5).	As specified	All
	Rolling bearing replacement (4, 6 and 7).	4 - 6 y	Selected and essential drives.
	Cleaning/reinsulating/rewinding (3).	As determined by inspection and tests	

Evidently, these scheduled maintenances specifically target the three main components (bearing, rotor and stator) that often fail. The first type of scheduled maintenance is a simple inspection, motors are generally checked for its cleanliness where they should be free of dust, debris and oil. Also, the winding of the rotor and stator are checked for any fraying, loose connections, or rusting. The second type of scheduled maintenance is testing and measurement. This is a more detailed form of maintenance as vibration and electrical testing is conducted. Vibration testing monitors the levels and patterns of vibration signals to detect any abnormal vibration events and evaluate the overall condition of the motor. In electrical testing, voltage and current imbalance is the key area of concern as it leads to the decrease in efficiency and shortens the lifespan. Voltage imbalance occurs when the voltage of the three phases differs from one another. Another main concern is undervoltage, where at lower voltages, motors run at a reduced full-load efficiency, hence it starts to heat run, causes a larger slip and reduced torque. These factors lead to a shorter lifespan. Lastly, for restoration, loose, cracked, or overheated components are replaced, bearings are also lubricated and monitored according to their known service life.

Although scheduled maintenance can ensure that the motor preserves its lifespan, conducting maintenance based on a pre-defined schedule may incur more due to the machine downtime and redundant maintenance conducted on a healthy motor.

In comparison to scheduled maintenance, predictive maintenance does not involve regular maintenance procedures and only conducts maintenance right before the predicted time of failure. These predictive maintenance techniques analyse the past and current performance of motors to estimate the remaining useful life of the motor.

Different industries often have different predictive maintenance procedures. Some trend only resistance, vibration, and thermography, while others incorporate techniques such as Motor Current Signature Analysis to enhance the results. Hence, a standardised technique does not exist, and it is often unclear on the required procedures to evaluate a motor’s health [56]. Nonetheless, such programs have proven to be effective in cutting maintenance cost by providing non-intrusive monitoring, removing the need for job permissions, and maintaining a safe workplace for the workers. Through real-time tracking of the various components, the fault can be isolated and repaired quickly. Therefore, this paper intends to establish a practical and functional procedure for predictive maintenance of induction motors.

2.4. Machine Learning

Machine learning is a data analytics technique that uses models to learn from data and identify patterns without explicitly programming. Making up the backbone of machine learning is the neural network. This chapter will introduce a neural network and its shortcomings. It will proceed to introduce physics-based machine learning and elaborate on how it can overcome the limitations faced by a neural network.

2.4.1. Neural Network

Neural network aims to make predictions of different data sets with the same pattern through a set of algorithms. At a basic level, the neural network comprises of three node layers. They are the input layer, the hidden layer, and the output layer. An input (denoted as x) is entered into a neural network to predict an output (denoted as u), as depicted in the figure below.

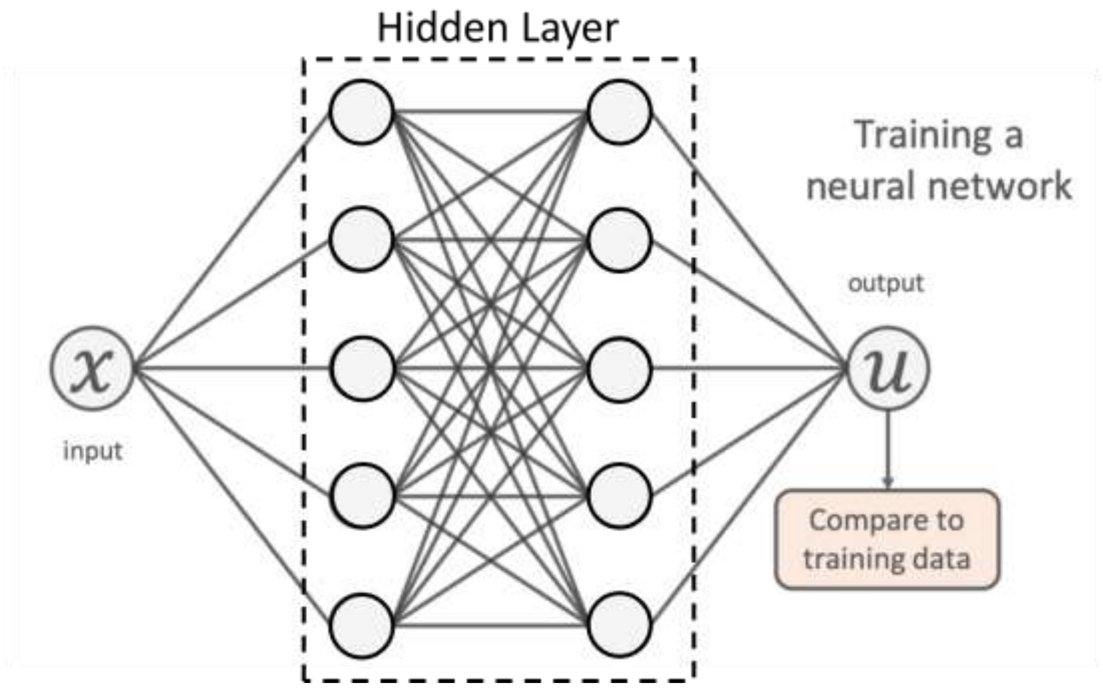


Figure 8: Traditional Neural Networks structure

The neurons are organized in a hierarchy of layers made up of nodes. Each node is connected to the previous layer through a linear regression model. This means that when each of the nodes in the previous layer are multiplied by their respective weights and then summed with the bias, it will form the value of the weighted sum.

The formula for this weighted sum is as shown in the equation below.

$$\text{Weighted Sum} = \sum(\text{weights} * \text{input}) + \text{bias} \qquad (1)$$

Evidently, these weights determine the importance of any given variable. The more important inputs will have larger weights hence contributing more to the output than the other weights. The addition of bias reduces the variance and hence introduces flexibility and better generalisation to the neural network.

Afterwards, the weighted sum passes through an activation function which will transform this weighted sum to produce an output. This is illustrated in the figure below.

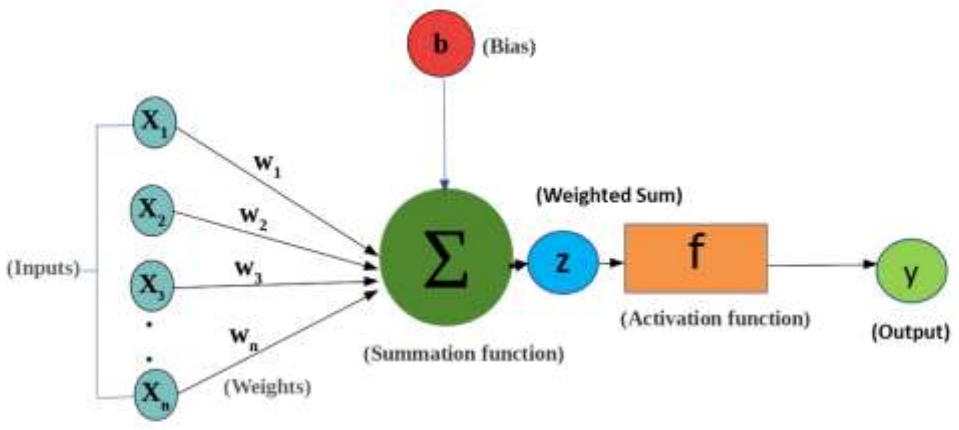


Figure 9: Illustration of weights, bias and activation function

From a macro perspective, when the output of one node serves as the input of the next node, the data is passed from one layer to the next. The above process constitutes as forward propagation where the training data runs through the network to obtain the output data.

The above illustrates how forward propagation numerically uses linear regression for each node to obtain the final output prediction in the last layer. However, when the prediction of the first few iterations is inaccurate, back propagation is used to determine how a single training sample would efficiently adjust the weights and bias to reduce the inaccuracy. Using back propagation for numerous training examples and averaging the desired changes for every sample will enable us to obtain gradient descent of the loss function. The loss function is the average of the mean squared error between the expected output and the calculated output.

$$\text{Loss} = \frac{1}{N}\sum_{i}^{N}(u_{\mathrm{NN}}(x_i;\theta) - u_{\mathrm{true}}(x_i))^2 \qquad (2)$$

With each iteration of forward propagation and back propagation for each training sample, the neural network model adjusts its weights and biases through gradient descent so that the loss function is minimised. When the loss function finally reaches a point of convergence, or the global minimum, the weights and biases of the model will be finalised, and the model is trained.

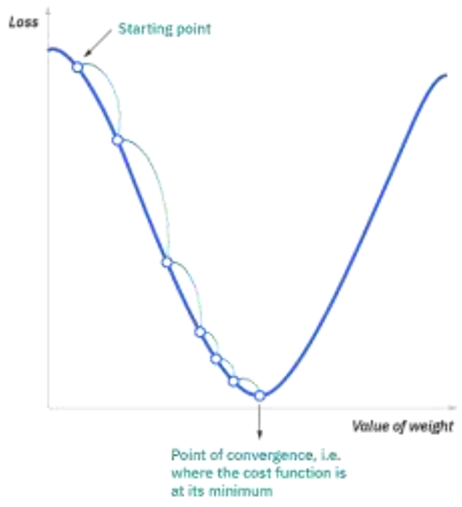

Loss
Starting point
Value of weight
Point of convergence, i.e. where the cost function is at its minimum

Figure 10: Loss Function against weight

For simple problems, a shallow neural network of one or two hidden layer is used due to the inherent problems with multi-layered neural network with gradient-based propagation methods. These problems include vanishing gradient and exploding gradient which makes too small or too large updates on the weights, as a result, backpropagation stops minimizing the errors effectively. To solve these problems, many studies have been done on optimal weight initializations and better learning algorithms for improved accuracy.

Despite the proven efficacy of neural network, it is only a purely data-driven approach which requires a large training data set. Moreover, it is unable to consider the physical characteristics underlying the problem. Hence, PINNs are modelled to embed the knowledge of any physical laws despite low data availability.

2.4.2 Physics Based Machine Learning

Physics-based machine learning combines physical models and machine learning models. The use of physics-based machine learning is often applied in complex physical, biological, environmental, and engineering systems. This is because data acquisition in these fields can be expensive, and data-driven methods cannot make accurate predictions with a small training data set. Nonetheless, there is already extensive physical domain knowledge of these systems in the form of physical laws, equations and rules. Hence, the fundamental idea of physics-based machine learning is to incorporate physical domain knowledge into machine learning techniques by adding this knowledge as a regularisation agent. This regularisation agent in the loss function will limit the number of possible solutions by ensuring that they adhere to the physical laws imposed. As such, the physics-based machine learning model will arrive at accurate solutions despite a smaller training data set. Raissi et al. can be considered the pioneers of this field when they presented its effectiveness in the Schrodinger equation, Allen-Cahn equation, Burger’s equation and Navier–Stokes equations [57]. Whilst they presented the use of physics-based machine learning for both data-driven solutions (predicting state or temporal evolution) and data-discovery (obtaining a parametrization for a physical term from the observations) of partial differential equations, this literature review will present the work done in the former.

Similar to traditional neural networks, given an input (denoted as x and t), a neural network can be used to output a prediction of its value (denoted as u), as depicted in the figure below.

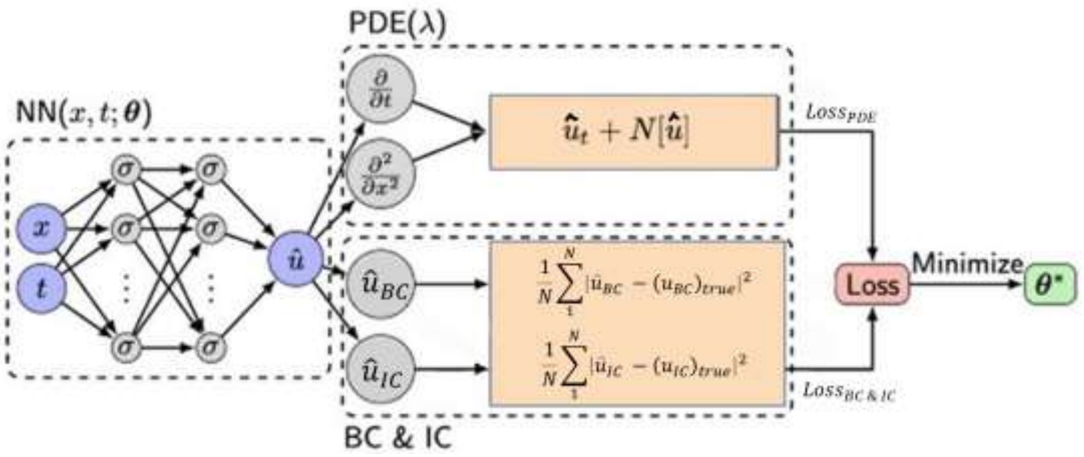


Figure 11: Physics-Informed Neural Network structure

However, unlike the traditional neural network, PINNs leverage governing physical equations in neural network training. This is achieved through partial differential equations (PDE) which is encapsulates the physical laws of many systems. Given an unknown quantity denotes as *u(x,t),* a generalised nonlinear partial differential equation can be defined as:

$$u_t + N[u] = 0 \qquad (3)$$

In this equation, u_t denotes the rate of change of the quantity with respect to time and $N[u]$ denotes a nonlinear operator that can contain any PDE. This can come in the form of high order spatial derivatives.

In data-driven solutions, the unknown quantity *u(x,t)* can be approximated using the neural network. If the prediction is accurate, the solution should fit the PDE and the residual *f(x,t)* (as defined below) should be zero.

$$f := u_t + N[u] \qquad (4)$$

This residual will therefore be used a regularisation agent to encode the physical domain knowledge in the loss function. However, most PDEs do not have unique solutions unless the initial and boundary conditions are specified. Hence, the PDE functions to remove any solutions that does not adhere to the physical law and the known boundary conditions serves to constrain the solution to specific points.

The loss function can therefore be defined as:

$$Loss_{total} = \alpha * Loss_{BC\,\&\,IC} + \beta * Loss_{PDE} \qquad (5)$$

where

(1) $Loss_{BC\,\&\,IC} = \frac{1}{N}\sum_{1}^{N}|\hat{u}_{BC} - (u_{BC})_{true}|^2 + \frac{1}{N}\sum_{1}^{N}|\hat{u}_{IC} - (u_{IC})_{true}|^2$ represents the sum of mean square error for both the boundary conditions (BC) and initial conditions (IC)

(2) $Loss_{PDE} = u_t + N[u]$ represent the residual term of the PDE

(3) α and β represent the hyperparameters that scale the contribution of $Loss_{BC\,\&\,IC}$ and $Loss_{PDE}$ respectively.

Given a sufficiently expressed neural network architecture and sufficient number of known boundary conditions and initial conditions, it is proven that an accurate prediction can be made when the above loss function is minimised.

However, listed below are some of the challenges that are still faced in the minimisation of the loss function.

(1) No guarantee that the loss function is at its global minimum
(2) Hyperparameters varies based on application

3. Methodology

This chapter presents the methodology used to predict the RUL of 8 induction motors based on the methods covered in the general prognostics process covered in the literature review. The data pre-processing and feature extraction stages were generated by Yang and his team [58].

3.1. Data and Pre-processing

The raw sensor data were collected by Professor J. Wesley Hines and his team from the University of Tennessee [59]. Their study involved the cyclical thermal ageing of 3-phase, 3600RPM industrial induction motors to induce accelerated insulation breakdown and corrosion within the motors. At the start of the experiments, each motor was new, and they were cyclically thermal aged until they could not start up. The following describes the thermal ageing process.

1. Heat motor in laboratory-grade oven at 140° C (or 160° C) for 72 hours

2. Remove and allow to air cool for 6 hours. 91

3. Quench in enclosed shallow water pool for 15 minutes (this acts as the suggested replacement for the recommended humidity chamber)

4. Immediately place back in the oven and heat again for 72 hours.

5. Air cool for 18 hours before performing "Startup Testing"

After each cycle of thermal ageing, the motor is started up where it will undergo a transient stage followed by a steady-state stage. The data collection will occur during the steady-state stage, as illustrated below.

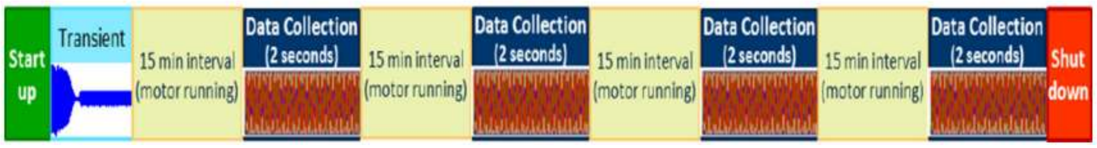
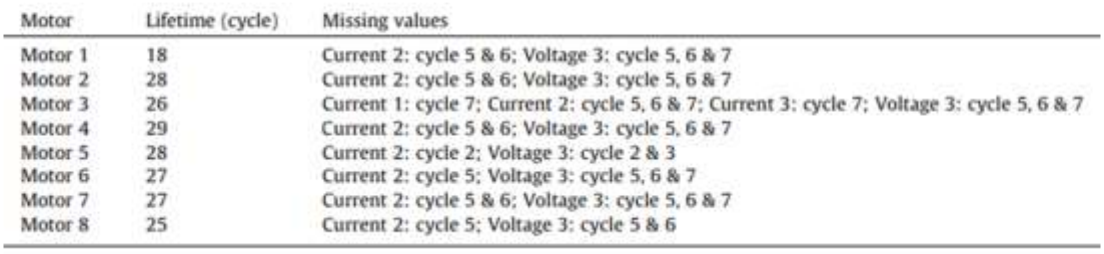

Start up
Transient
15 min interval (motor running)
Data Collection (2 seconds)
15 min interval (motor running)
Data Collection (2 seconds)
15 min interval (motor running)
Data Collection (2 seconds)
15 min interval (motor running)
Data Collection (2 seconds)
Shut down

Figure 12: Timeline for collecting steady state data

During each cycle, 8 seconds of steady state data signals was collected from 13 data channels of raw signals. They came in the form of 3 phases of current (Current 1, 2, 3), 3 phases of voltage (Voltage 1, 2, 3), 2 accelerometer signal (Vibration 1, 2), 2 dynamometer load signal (output current and voltage), tachometer signal (motor speed), thermocouple signal (temperature) and acoustic sensor signal (sound). 8 motors experienced similar types of failure modes and their data was used in this paper. Among the 13 channels of raw signals, 2 dynamometer load signals were excluded since they were measured by connecting the motor to a specific load equipment which will be impractical in real-life applications. Hence, the remaining 11 channels of raw signals will be used for the analysis in this paper. However, the data set collected has missing information as reflected below.

Table 5: Details of missing values in the motors' data.



Motor	Lifetime (cycle)	Missing values
Motor 1	18	Current 2: cycle 5 & 6; Voltage 3: cycle 5, 6 & 7
Motor 2	28	Current 2: cycle 5 & 6; Voltage 3: cycle 5, 6 & 7
Motor 3	26	Current 1: cycle 7; Current 2: cycle 5, 6 & 7; Current 3: cycle 7; Voltage 3: cycle 5, 6 & 7
Motor 4	29	Current 2: cycle 5 & 6; Voltage 3: cycle 5, 6 & 7
Motor 5	28	Current 2: cycle 2; Voltage 3: cycle 2 & 3
Motor 6	27	Current 2: cycle 5; Voltage 3: cycle 5, 6 & 7
Motor 7	27	Current 2: cycle 5 & 6; Voltage 3: cycle 5, 6 & 7
Motor 8	25	Current 2: cycle 5; Voltage 3: cycle 5 & 6

The missing values are substituted with values from the previous cycle. For example, the missing Voltage 3 values at cycle 5 in Motor 1 was replaced with signals from cycle 4. This replacement method was adopted for its simplicity and because the missing values were little.

Through the data & pre-processing stage, the raw time-series signals collected through sensors are pre-processed to remove possible issues such as noise and/or missing data. The pre-processed signals are then undergone feature engineering for further processing.

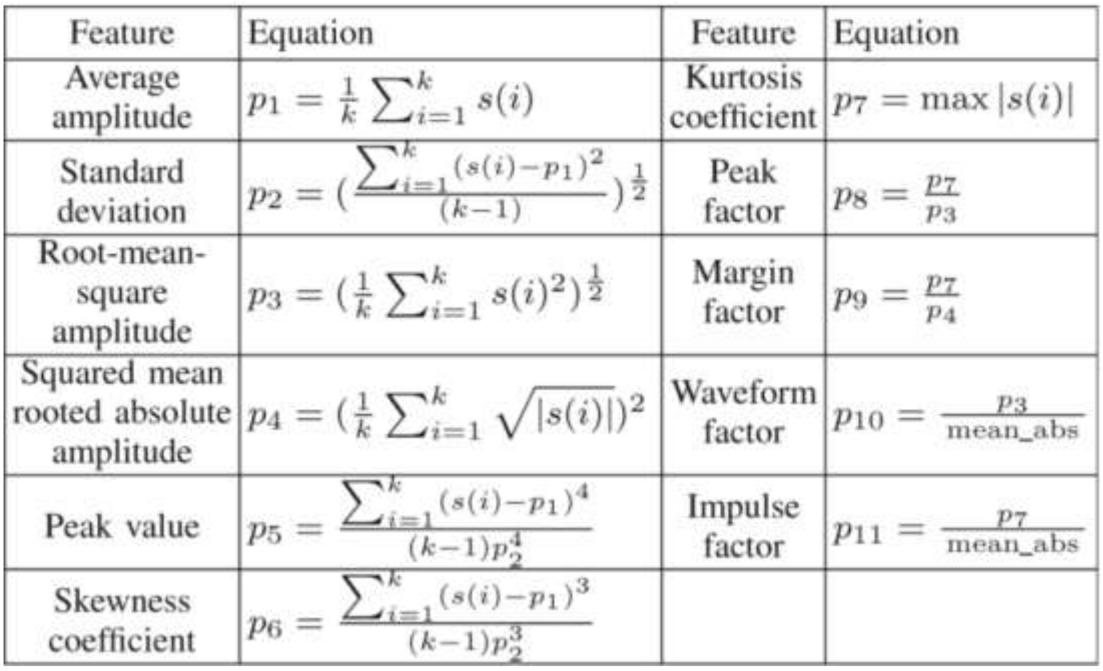
3.2. Feature Engineering

In the feature engineering, the raw signals are processed by segment, where new features are generated from each segment (by feature extraction), and useful features are selected (by feature selection). This section will discuss the methods used to convert time-series raw signals into a manageable data structure whilst preserving the characteristics of the raw data.

3.2.1. Feature Extraction

The raw signals were processed by segments of 0.5 s in the experiments. Using segmentation, feature extraction was conducted on every 0.5 s of signals in producing a set of new features. In this study, the main feature extraction technique used is statistical indicators (Table 6) [60].

Table 6: Details of the Time Domain Features



Feature	Equation	Feature	Equation
Average amplitude	$p_1 = \frac{1}{k}\sum_{i=1}^{k} s(i)$	Kurtosis coefficient	$p_7 = \max\lvert s(i)\rvert$
Standard deviation	$p_2 = \left(\frac{\sum_{i=1}^{k}(s(i)-p_1)^2}{(k-1)}\right)^{\frac{1}{2}}$	Peak factor	$p_8 = \frac{p_7}{p_3}$
Root-mean-square amplitude	$p_3 = \left(\frac{1}{k}\sum_{i=1}^{k} s(i)^2\right)^{\frac{1}{2}}$	Margin factor	$p_9 = \frac{p_7}{p_4}$
Squared mean rooted absolute amplitude	$p_4 = \left(\frac{1}{k}\sum_{i=1}^{k}\sqrt{\lvert s(i)\rvert}\right)^2$	Waveform factor	$p_{10} = \frac{p_3}{\text{mean\_abs}}$
Peak value	$p_5 = \frac{\sum_{i=1}^{k}(s(i)-p_1)^4}{(k-1)p_2^4}$	Impulse factor	$p_{11} = \frac{p_7}{\text{mean\_abs}}$
Skewness coefficient	$p_6 = \frac{\sum_{i=1}^{k}(s(i)-p_1)^3}{(k-1)p_2^3}$		

The 11 statistical indicators are applied to each of the 11 channels of raw data to obtain 121 features.

3.2.2. Feature Selection

The next step is feature selection, which aims to reduce the 121 features by selecting a subset of features that are most relevant for the modelling process. This is achieved through ranking the new features obtained from feature extraction based on significance and then keeping on the important features. In this experiment, Fisher's ratio was utilised to select the most important features. Fisher's ratio measures the discriminating power for each feature and its formula is reflected below.

$$FR(X_j) = \frac{(m_{j(1)} - m_{j(2)})^2}{\sigma^2_{j(1)} + \sigma^2_{j(2)}} \quad (6)$$

where $m_{j(cl)}$ and $\sigma^2_{m(cl)}$ are the mean and variance of feature X_j containing two classes cl = 1 (healthy) and cl = 2 (unhealthy). The larger the fisher's ratio FR(X_j), the more discrimination the feature X_j is. The two classes of healthy and unhealthy data are obtained by assuming that the motor is healthy for the first four cycles and unhealthy for the last four cycles.

The 20 most distinctive feature vectors were obtained, and their corresponding Fisher's Ratio value is tabulated in Table 7. Each feature vector can be interpreted as "name of raw data_ type of statistical indicator". For example, for the most discriminative feature vector, labelled "Voltage2_p8", it represents the raw signal data from Voltage 2 undergoing statistical indicator, p8 (Peak Factor).

Table 7: Feature Vector ranked according to Fisher's Ratio

Ranking	Feature Vectors	Value
1	Voltage2_p8	2.4736
2	Voltage2_p10	2.2286
3	Voltage1_p8	1.8676
4	Voltage1_p10	1.4923
5	Voltage2_p11	1.4898
6	Temperature_p4	1.2602
7	Temperature_p1	1.2602
8	Temperature_p3	1.2602
9	Temperature_p5	1.2594
10	Voltage1_p11	1.2379
11	Tachometer_p4	0.9196
12	Voltage2_p9	0.6058
13	Tachometer_p8	0.5098
14	Voltage1_p9	0.4765
15	Tachometer_p10	0.4331
16	Tachometer_p3	0.4042
17	Tachometer_p2	0.4041
18	Tachometer_p11	0.3721
19	Tachometer_p9	0.3497
20	Tachometer_p7	0.3215

In this feature selection stage using Fisher's Ratio, the 121 feature vectors have been reduced down to 20 feature vectors. These 20 feature vectors then be used an input into our physics-based model.

The codes accompanying explanations are available on the following Google Collaboratory link: https://colab.research.google.com/drive/1Hhk-NIvYLOwtGOo74Wetx9BvisBfYQUy?usp=sharing

3.3. Feature to RUL Prediction

The prediction of the RUL will be achieved through a two-stage modelling process. In the first stage, feature vector will be mapped to the Health Index (HI) of the system through a physics-based machine learning framework. In the second stage, HI values are then mapped to the corresponding RUL values, from which the final RUL prediction is produced by ensembling them (averaging was used in this paper). This chapter will first present the simple application of the physics-based machine learning model to exemplify its principles. Next, it will present the proposed physics-based machine learning framework. Lastly, it will explain the ensemble approach used to map HI prediction to RUL.

3.3.1. Physics-based machine learning (Kinematics Example)

This section will cover a simple kinematics case study to demonstrate the principles and effectiveness of physics-based machine learning. Imagine the scenario where a rugby player kicks the ball upwards, and the ball is not subjected to air resistance. The expected projectile of the ball will be a parabolic motion as illustrated below.

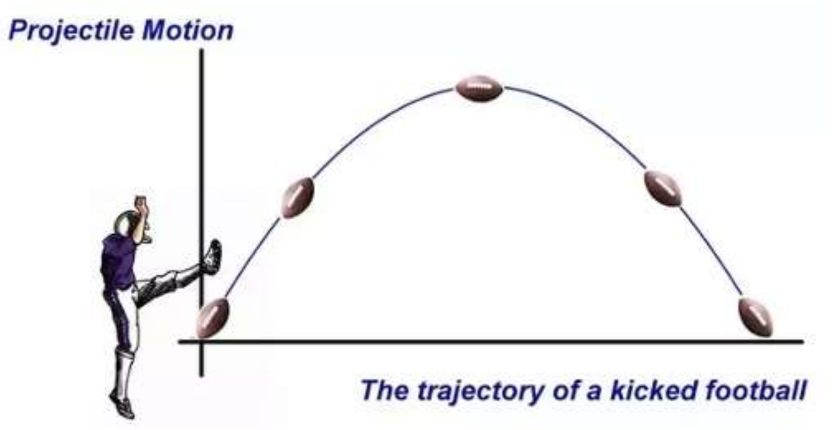


Figure 13: Trajectory of ball

Given that the time begins when he first kicks the ball and the ball lands back on the ground ten seconds later, we can use physics-based machine learning to find the projectile of the ball. The vertical displacement can be obtained by knowing the initial conditions and the partial differential equation that the motion of the ball follows.

3.3.1.1. Loss Function

The physical model that will be used is:

$$\frac{d^2x}{dt^2} = -10 \tag{7}$$

This equation is simple, non-linear and one dimensional. It assumes that a body in free-fall is subjected to a gravitational acceleration of 10m/s^2. The displacement of the body is denoted by x(t). The initial conditions set are:

1. x (0) = 0
2. x (10) = 0

This implies that when t = 0s and t = 10s, the displacement is zero. Similar to the physics-based machine learning framework encapsulated in the literature review in section 2.4.2, the loss function can be defined as:

$$Loss_{total} = \alpha * Loss_{IC} + \beta * Loss_{PDE} \tag{8}$$

where

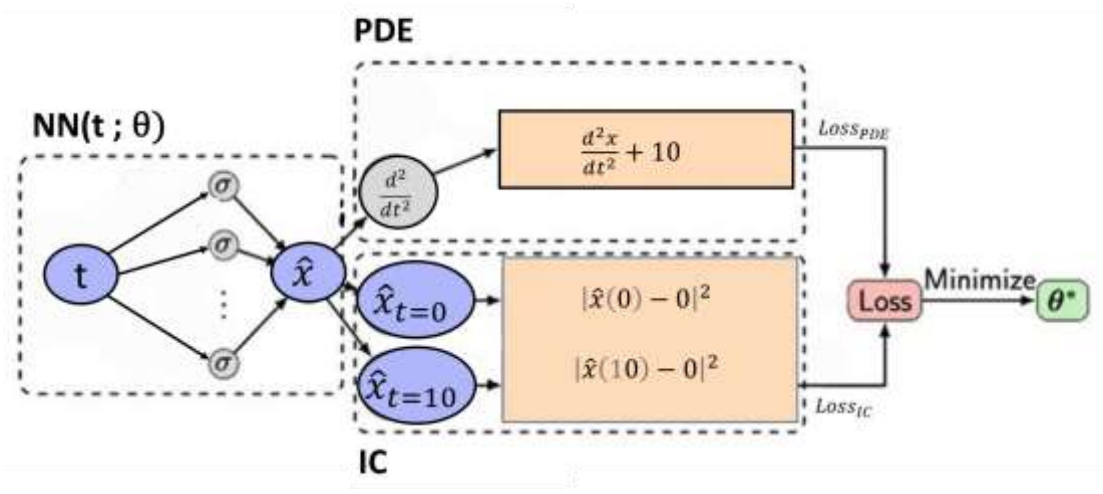
(1) $Loss_{IC} = \frac{1}{2}[|x(0) - 0|^2 + |x(10) - 0|^2]$ represents the mean square error for initial conditions (IC)

(2) $Loss_{PDE} = \frac{d^2x}{dt^2} + 10$ represent the residual term of the PDE

(3) α and β represent the hyperparameters that scale the contribution of $Loss_{IC}$ and $Loss_{PDE}$ respectively. In this case, α and β are both set as 1.

3.3.1.2. *Physics-Based Machine Learning Framework*

Given an input (denoted t), the physics-based machine learning model can be used to predict an output (denoted as $\hat{x}$), as shown in the figure below. To train the model, the loss function includes the 2 datasets at the initial condition and 50 random unlabelled training data (between t=0 and t=10) that will serve as collocation points to run through the PDE.



The aim is to infer the value of the entire displacement solution between t=0 and t=10 using the kinematics equation. A shallow neural network with only one hidden layer that has 20 neurons, and a hyperbolic tangent activation function was used. The shallow neural network was chosen due to the anticipated simplicity of the problem. Since the total training data is relatively small, the loss function is optimised using L-BFGS optimiser. It is a quasi-Newton, full-batch gradient-based optimisation algorithm. This means that in each iteration, all the data is used for training.

3.3.1.3. Results

After the model is trained, 250 random samples of time between t=0 and t=10 were input into the model and the prediction error value for the test data was reduced to 1.95×10^{-4}. The graphical result is illustrated below.

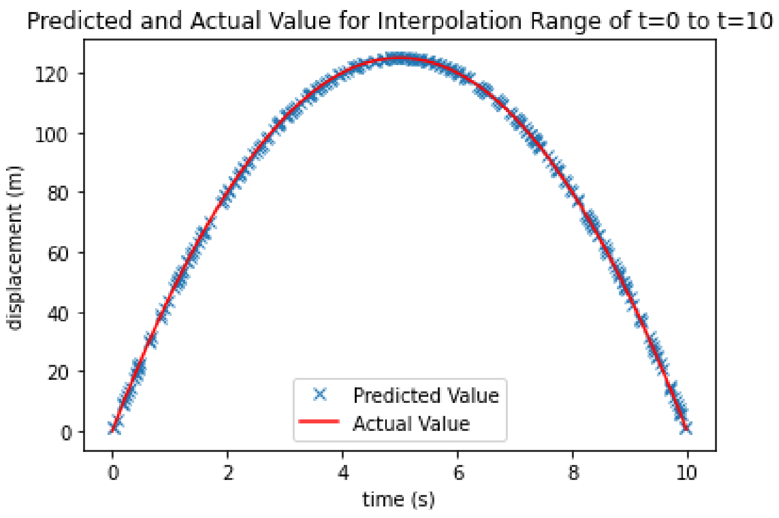


Figure 14: Results of Kinematics Example

From the results, it can be concluded that with a small training data set (2 initial conditions), physical based machine learning is able to accurately capture the simple nonlinear behaviour of the kinematics equation. This example aims to validate the robustness of physics-based machine learning methods with respect to the problem of over-fitting. Specifically, the $Loss_{PDE}$ acts as a regularisation agent to penalise solutions that do not conform to the kinematics equation.

To compare the physics-based machine learning to the normal neural network, a neural network with similar parameters was modelled. The model also used a shallow neural network that has only one hidden layer with 20 neurons and a hyperbolic tangent activation function. To compare both equivalently, 250 test data sets will be used too. Since the general guideline for splitting the training and test set is 25 percent for testing and the remaining 75 percent for training, 750 training data will be used for training [61].

After the model is trained, 250 random samples of time between t=0 and t=10 were input into the model and the prediction error value for the test data was reduced to

2.20×10^{-4}. Despite the greater number of training data, the prediction error is higher.

More interestingly, 50 test data sets (t=10 to t=15) outside the range of the training data sets (t=0 to t=10) was input into both models. The results are shown below.

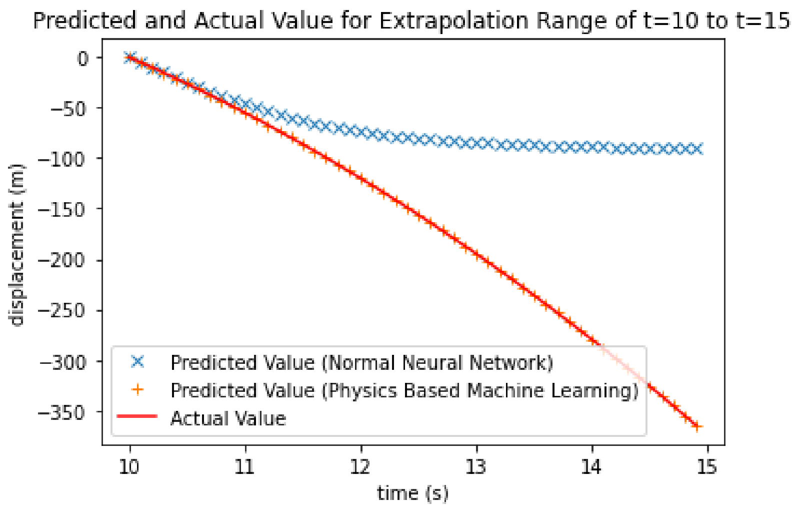


Figure 15: Comparison of results for extrapolation test data

The results corroborate that despite being out of the range of the training data set, physics-based machine learning is able to accurately extrapolate the displacement. In comparison, the neural network proves to be heavily dependent on the training data set. Specifically, the neural network is unable to capture the physical trends of the model when the test data is not within the range of the training data.

3.3.1.4. Code

The above example is achieved using the TensorFlow library. All the codes accompanying explanations are available on the following Google Collaboratory links:

1. Physics-Based Machine Learning: https://colab.research.google.com/drive/1YgNtaZAjPpwiW7IRFTGR8FOujf1QFH7w?usp=sharing
2. Normal Neural Network: https://colab.research.google.com/drive/1CBNBY1LIENRarLGcRs2N0yFTg-tyilRy?usp=sharing

In the following section, the backbone of the code will be explained. The approach taken was to create a class that bundles important data and functionalities are together.

Code	Explanation
`class PhysicsInformedNN:`	- Create class

`  def __init__(self, t_b, x_b, t_f, layers):`
`    # Time boundary conditions at t=0 and t=10`
`    self.t_b = t_b`
`    # Displacement boundary condition x(t=0)=0 and x(t=10)=0`
`    self.x_b = x_b`
`    # Collocation time`
`    self.t_f = t_f`
`    # Neurons in each layer`
`    self.layers = layers`
`    # Initialize weights and bias`
`    self.weights, self.biases = self.initialize_NN(layers)`
 	
`    # Predict the displacement at the boundary conditions x(t=0)=0 and x(t=10)=0`
`    self.x_b_pred = self.net_x_b(self.t_b_tf)`
	- Instantiate the class to create local objects in the class
	- Instantiate the time boundary conditions
	
	- Instantiate the displacement boundary conditions
	
	- Instantiate the time collocation points
	- Instantiate the number of neurons in each layer
	- Instantiate the weights and biases of the neural network using the initialize_NN method (discussed later)
	- Input the time boundary conditions into the method net_x_b (discuss later) to predict the displacement at boundary conditions

```
    # Calculate the PDE residual
    self.f_pred = self.net_f(self.t_f_tf)

    # Formulate the loss function
    self.loss = tf.reduce_mean(tf.square(self.x_b_tf - self.x_b_pred)) + tf.reduce_mean(tf.square(self.f_pred))

    # Select L-BFGS-B as optimizer
    self.optimizer = tf.contrib.opt.ScipyOptimizerInterface(self.loss,
method = 'L-BFGS',
options = {'maxiter': 50000,
           'maxfun': 50000,
           'maxcor': 50,
           'maxls': 50,
           'ftol' : 1.0 * np.finfo(float).eps})
```

- Input the collocation time values into the method net_f (discuss later) to calculate the residual of the PDE

- Obtain the loss function which is shown in Equation 8

- Select the type of optimiser to use. In this case, the optimiser used is L-BFGS.

```
def initialize_NN(self, layers):
    weights = []
    biases = []
    num_layers = len(layers)
    for l in range(0,num_layers-1):
        W = self.xavier_init(size=[layers[l], layers[l+1]])
        b = tf.Variable(tf.zeros([1,layers[l+1]], dtype=tf.float32), dtype=tf.float32)
        weights.append(W)
        biases.append(b)
    return weights, biases
```

- The method initialize_NN is used to initialise weights and biases using Xavier initialisation

```
def neural_net(self, X, weights, biases):
    num_layers = len(weights) + 1
    H = 2.0*(X - self.lb)/(self.ub - self.lb) - 1.0
    for l in range(0,num_layers-2):
        W = weights[l]
        b = biases[l]
        H = tf.tanh(tf.add(tf.matmul(H, W), b))
    W = weights[-1]
    b = biases[-1]
    Y = tf.add(tf.matmul(H, W), b)
    return Y
```

- The method neural_net is used to produce the output for the displacement in the forward propagation method. As established in Section 2.4.1, each of the nodes in the previous layer are multiplied by their respective weights and then summed with the bias, it will form the value of the weighted sum.

`def net_x_b(self, t):`
`  x = self.neural_net(tf.concat([t],1), self.weights, self.biases)`
`  return x`
	The method net_x_b is used to predict the displacement at the boundary conditions when the time values at the boundary conditions are input

`def net_f(self, t):`
`  x = self.net_x_b(t)`
`  x_t = tf.gradients(x, t)[0]`
`  x_tt = tf.gradients(x_t, t)[0]`
`  f = x_tt + 10`
`  return f`
	The method net_f is used to output the residual of the PDE.
	$(Loss_{PDE} = \frac{d^2x}{dt^2} + 10)$

`def train(self):`
`  tf_dict = {self.t_b_tf: self.t_b, self.x_b_tf: self.x_b, self.t_f_tf: self.t_f}`
`  self.optimizer.minimize(self.sess,`
`  feed_dict = tf_dict,`
`  fetches = [self.loss],`
`  loss_callback = self.callback)`
	The method train is used to train the neural network by minimise the loss function (Equation 8). When this is achieved, the weights and biases in the neural network will be finalised.

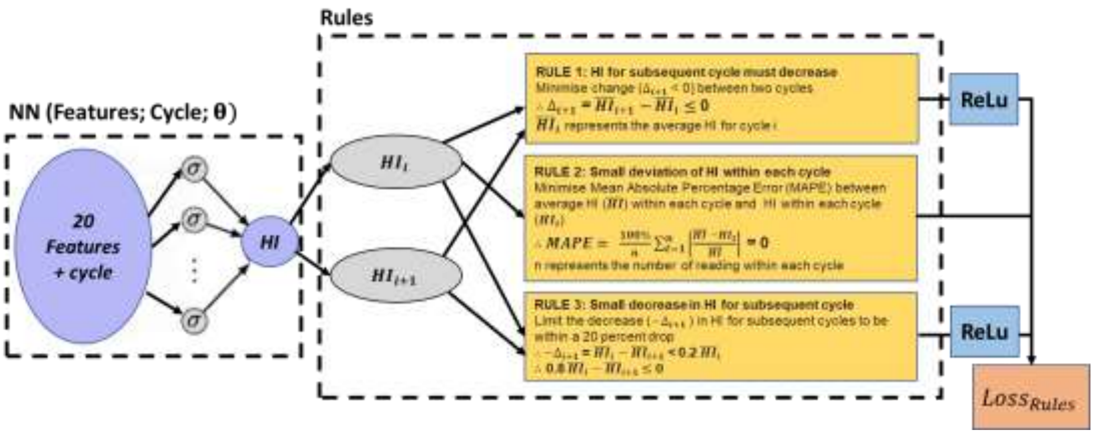
`def predict(self, t_star):`
`  x_star = self.sess.run(self.x_b_pred, {self.t_b_tf: t_star[:,0:1]})`
`  f_star = self.sess.run(self.f_pred, {self.t_f_tf: t_star[:,0:1]})`
`  return x_star, f_star`
	The method predict is used to predict the displacement when any value of time is input.

3.3.2. *Physics-based machine learning (Feature -> HI)*

The 20 most distinctive feature vectors and cycles are input into the physics-based machine learning model to produce an output of the predicted Health Index. Similar to the kinematics example previously, this proposed physics-based machine learning model framework embeds prior knowledge of HI into a neural network. However, instead of adding PDE into the loss function to function as a regularization agent, three proposed rules are added into the loss function. The rules and boundary conditions in the loss function will be further explained in this section.

3.3.2.1. *Rules*

In the forward propagation, the feature vectors and cycles are input into the neural network to predict the Health Index. The predicted Health Index for all the cycles is entered into the three rules to calculate their contribution to the loss.



Rules
NN (Features; Cycle; θ)
20 Features + cycle
σ
HI
HI_i
HI_{i+1}
RULE 1: HI for subsequent cycle must decrease
Minimise change ($\Delta_{i+1} < 0$) between two cycles
$\therefore \Delta_{i+1} = \overline{HI}_{i+1} - \overline{HI}_i \leq 0$
$\overline{HI}_i$ represents the average HI for cycle i
RULE 2: Small deviation of HI within each cycle
Minimise Mean Absolute Percentage Error (MAPE) between average HI ($\overline{HI}$) within each cycle and HI within each cycle (HI_t)
$\therefore MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{\overline{HI}-HI_t}{\overline{HI}}\right| = 0$
n represents the number of reading within each cycle
RULE 3: Small decrease in HI for subsequent cycle
Limit the decrease ($-\Delta_{i+1}$) in HI for subsequent cycles to be within a 20 percent drop
$\therefore -\Delta_{i+1} = \overline{HI}_i - \overline{HI}_{i+1} < 0.2\,\overline{HI}_i$
$\therefore 0.8\,\overline{HI}_i - \overline{HI}_{i+1} \leq 0$
ReLu
ReLu
$Loss_{Rules}$

Figure 16: Physics-Based Machine Learning Model (Rules)

As illustrated in Figure 16, the three proposed rules for physical consistency of the Health Index are (1) predicted Health Index value for the subsequent cycle should decrease (2) deviation of the predicted Health Index value within each cycle should be small (3) drop in the predicted Health Index for the subsequent cycle is small.

3.3.2.1.1. Rule 1: Predicted Health Index value for subsequent cycle should decrease

The first physical knowledge is that the Health Index of a motor decreases according to time. Therefore, the prediction of Health Index for the subsequent cycle must decrease in order to satisfy the physical law. Logically, if the predicted Health Index of the subsequent cycle increase, it would be physically inconsistent. To enforce this rule in the neural network, a Loss $_{\text{Rule 1}}$ term will be added into the neural network.

$$Loss_{Rule\ 1} = \frac{1}{N}\left[ReLu(\sum_{i=1}^{N-1} \overline{HI}_{i+1} - \overline{HI}_i)\right]^2 \qquad (9)$$

Where *i* represents the cycle no, *N* represents the last cycle and ReLu represents rectified linear activation function. It is a piecewise linear function that will produce the input if it is positive, otherwise, it will output zero as illustrated below.

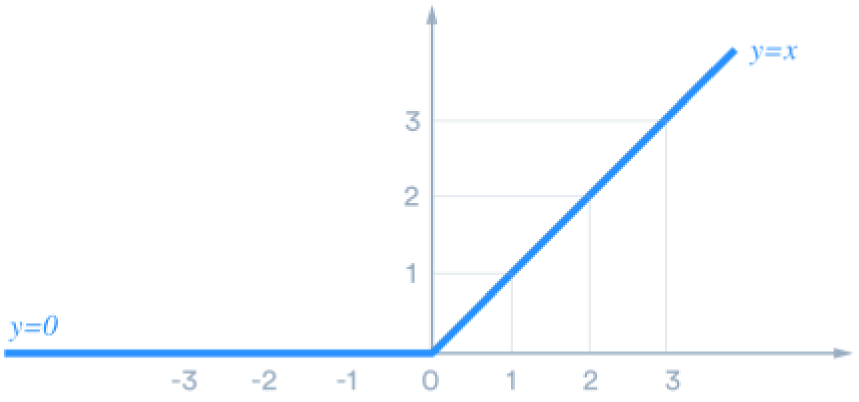


Figure 17: ReLu function

Ideally, if $\overline{HI}_{i+1}$ (the mean of HI value for cycle i+1) is less than $\overline{HI}_i$ (the mean of HI value for cycle i), $\overline{HI}_{i+1} - \overline{HI}_i$ will be negative. When a negative value passes through the ReLu function, the output will be zero. Therefore, when the mean of the Health Index decreases in the subsequent cycle, the loss term ($Loss_{Rule\ 1}$) would zero.

Conversely, if $\overline{HI}_{i+1}$ (the mean of HI value for cycle i+1) is more than $\overline{HI}_i$ (the mean of HI value for cycle i), $\overline{HI}_{i+1} - \overline{HI}_i$ will be positive. When a positive value passes through the ReLu function, the output will be equal to the input. Therefore, when the increase in the mean of the Health Index for the subsequent cycle is larger, the loss term ($Loss_{Rule\ 1}$) would also be larger. The loss term ($Loss_{Rule\ 1}$) will be the mean square error of the output of the ReLu function. Hence, an increase in Health Index will punish the loss function. The pseudocode with explanations for Rule 1 is written below.

```
# Create an array which appends all the changes
total_change = []
# Create a loop to run from first to last cycle
for i in range(1, max_cycle):
  # Obtain features for cycle i
  feature_cycle1 = feature_f_train[cycle_f_train[:,0] == i]
  # Obtain cycle for cycle i
  cycle_cycle1 = cycle_f_train[cycle_f_train[:,0] == i]
  # Obtain features for cycle i+1
  feature_cycle2 = feature_f_train[cycle_f_train[:,0] == i+1]
  # Obtain cycle for cycle i+1
  cycle_cycle2 = cycle_f_train[cycle_f_train[:,0] == i+1]
  # Input features and cycle for cycle i into model to obtain HI
  hi_cycle1 = model(feature_cycle1, cycle_cycle1)
  # Input feature and cycle for cycle i+1 into model to obtain HI
  hi_cycle2 = model(feature_cycle2, cycle_cycle2)
  # Obtain the difference between mean HI of adjacent cycles
  change = mean(hi_cycle2) - mean(hi_cycle1)
  # Append different to array
  total_change.append(change)
# Reshape the array into size (nx1)
total_change = stack(total_change)
# Pass the array through a ReLu function
cost1 = relu(total_change)
# Obtain the mean square error of the array
loss1 = mean(square(cost1))
```

3.3.2.1.2. Rule 2: Deviation of the predicted Health Index value within each cycle should be small

The second physical knowledge is that the predicted Health Index of a motor within the same cycle should be similar. Therefore, predictions for Health Index of the same cycle should be close together. To enforce this rule in the neural network, a Loss $_{\text{Rule2}}$ term will be added into the neural network. The formula for $Loss_{Rule\ 2}$ is similar to the mean absolute percentage error between the mean predicted Health Index for one cycle (denoted as $\overline{HI}$) and each of the reading within that cycle (denoted as HI_t).

$$Loss_{Rule\ 2} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{\overline{HI} - HI_t}{\overline{HI}}\right| \tag{10}$$

Where *t* represents the data number within a cycle, *n* represents the last data set in the cycle.

Based on the formula, a larger deviation in the predicted Health Index of the same cycle, will reflect a larger loss term ($Loss_{Rule\ 2}$). Hence, a large deviation will punish the loss function. The pseudocode with explanations for Rule 2 is written below.

```
# Create an array which appends all the MAPE
total_mape = []
# Create a loop to run from first to last cycle
for i in range(1, self.max_cycle+1):
  # Obtain features for cycle i
  feature_cycle1 = feature_f_train[cycle_f_train[:,0] == i]
  # Obtain cycle for cycle i
  cycle_cycle1 = cycle_f_train[cycle_f_train[:,0] == i]
  # Input features and cycle for cycle i into model to obtain HI
  hi_pred = model(feature_cycle1, cycle_cycle1)
  # Obtain the mean predicted Health Index for one cycle
  hi_mean = mean(hi_pred)
  # Obtain the MAPE
  mape = mean_absolute_percentage_error(hi_true, hi_pred)
  # Append MAPE to array
    total_mape.append(mape)
# Reshape the array into size (nx1)
loss_2 = total_mape = stack(total_mape)
```

3.3.2.1.3. Rule 3: Drop in the predicted Health Index for the subsequent cycle is small

The third physical knowledge is that the Health Index of a motor decreases gradually according to time and there are no sharp drops in Health Index between cycles. Therefore, the prediction of Health Index for the subsequent cycle must decrease by a small proportion in order to satisfy the physical law. Logically, if the predicted Health Index of the subsequent cycle drops drastically, it would be physically inconsistent. Therefore, we would like to punish the loss function if $\overline{HI}_i$ (the mean of HI value for cycle i) drops by more than 20 percent. Mathematically, there will be inconsistency if

$$\overline{HI}_i - \overline{HI}_{i+1} > 0.2\,\overline{HI}_i \Rightarrow 0.8\,\overline{HI}_i - \overline{HI}_{i+1} > 0$$

To enforce this rule in the neural network, a $Loss_{Rule\ 3}$ term will be added into the neural network.

$$Loss_{Rule\ 3} = \frac{1}{N}\left[ReLu\left(\sum_{i=1}^{N-1} 0.8\,\overline{HI}_i - \overline{HI}_{i+1}\right)\right]^2 \qquad (11)$$

Where *i* represents the cycle no, *N* represents the last cycle.

Ideally, $0.8\,\overline{HI}_i - \overline{HI}_{i+1}$ will be negative. When a negative value passes through the ReLu function, the output will be zero. Therefore, the mean of the Health Index decreases in the subsequent cycle, the loss term ($Loss_{Rule\ 3}$) would zero.

Conversely, $0.8\,\overline{HI}_i - \overline{HI}_{i+1}$ will be positive. When a positive value passes through the ReLu function, the output will be equal to the input. Therefore, when drop in the mean of the Health Index for the subsequent cycle is larger and exceeds 20 percent, the loss term ($Loss_{Rule\ 3}$) would also be larger. The loss term ($Loss_{Rule\ 3}$) will be the mean square error of the output of the ReLu function. Hence, a huge drop in Health Index will punish the loss function.

The pseudocode with explanations for Rule 3 is written below.

```
# Create an array which appends all the drops
total_drop = []
# Create a loop to run from first to last cycle
for i in range(1, max_cycle):
  # Obtain features for cycle i
  feature_cycle1 = feature_f_train[cycle_f_train[:,0] == i]
  # Obtain cycle for cycle i
  cycle_cycle1 = cycle_f_train[cycle_f_train[:,0] == i]
  # Obtain features for cycle i+1
  feature_cycle2 = feature_f_train[cycle_f_train[:,0] == i+1]
  # Obtain cycle for cycle i+1
  cycle_cycle2 = cycle_f_train[cycle_f_train[:,0] == i+1]
  # Input features and cycle for cycle i into model to obtain HI
  hi_cycle1 = model(feature_cycle1, cycle_cycle1)
  # Input features and cycle for cycle i+1 into model to obtain H
I
  hi_cycle2 = model(feature_cycle2, cycle_cycle2)
  # Obtain the drop for the mean HI of next cycle
  drop = 0.8 * mean(hi_cycle1) - mean(hi_cycle2)
  # Append different to array
  total_drop.append(drop)
# Reshape the array into size (nx1)
total_drop = stack(total_drop)
# Pass the array through a ReLu function
cost3 = relu(total_drop)
# Obtain the mean square error of the array
loss3 = mean(square(cost3))
```

3.3.2.2. Boundary Condition

In the forward propagation, the feature vectors and cycles are input into the neural network to predict the Health Index. The predicted Health Index for the first and last cycles are entered into the boundary conditions (BC) to calculate their contribution to the loss.

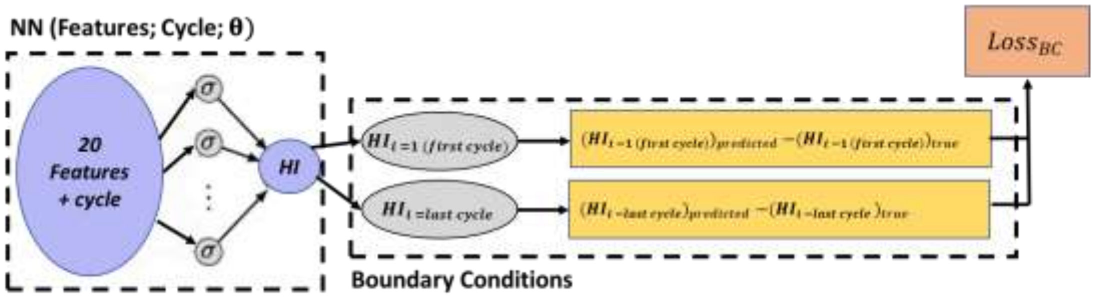

NN (Features; Cycle; θ)
20 Features + cycle
σ
HI
$HI_{i=1\,(first\,cycle)}$
$HI_{i=last\,cycle}$
$(HI_{i=1\,(first\,cycle)})_{predicted} - (HI_{i=1\,(first\,cycle)})_{true}$
$(HI_{i=last\,cycle})_{predicted} - (HI_{i=last\,cycle})_{true}$
$Loss_{BC}$
Boundary Conditions

Figure 18: Physics-Based Machine Learning Model (Boundary Conditions)

Similar to a traditional neural network, the first and last cycles are used as training set as seen in Figure 18. The true value of Health Index for the first cycle is set to be 1 (100%) as this represents when the motor is fully healthy. As the motor's health degrades to the last cycle, the true value of the Health Index of the last cycle is set to be 0.1 (10%) as this represents when the motor is faulty. Hence, the mean square error (Loss $_{BC}$) between the predicted HI value and true HI value for the first and last cycles must be minimised.

$$Loss_{BC} = \frac{1}{2}\left[\begin{array}{l} \sum[\,(HI_{i=1\,(first\,cycle)})_{predicted} - (HI_{i=1\,(first\,cycle)})_{true}\,]^2 \\ \quad + \sum[\,(HI_{i=last\,cycle})_{predicted} - (HI_{i=last\,cycle})_{true}\,]^2 \end{array}\right] \tag{12}$$

3.3.2.3. Overall physics-based machine learning framework

Combining the loss of the three rules and boundary conditions, the overall loss function can be defined as:

$$Loss_{total} = \alpha_0 * Loss_{BC} + \alpha_1 * Loss_{Rule\ 1} + \alpha_2 * Loss_{Rule\ 2} + \alpha_3 * Loss_{Rule\ 3} \tag{13}$$

Where: $\alpha_0, \alpha_1, \alpha_2$ and α_3 represent the hyperparameters that scale the contribution of the individual losses. It is important to adjust the relative importance of each regularisation parameters since the individual losses are all of different scales. The method for adjusting exceeds the scope of the present work and will be investigated in further studies. In this problem, $\alpha_0, \alpha_1, \alpha_2$ and α_3 were all kept at 1.

The overall physics-based machine learning framework can be derived as illustrated.

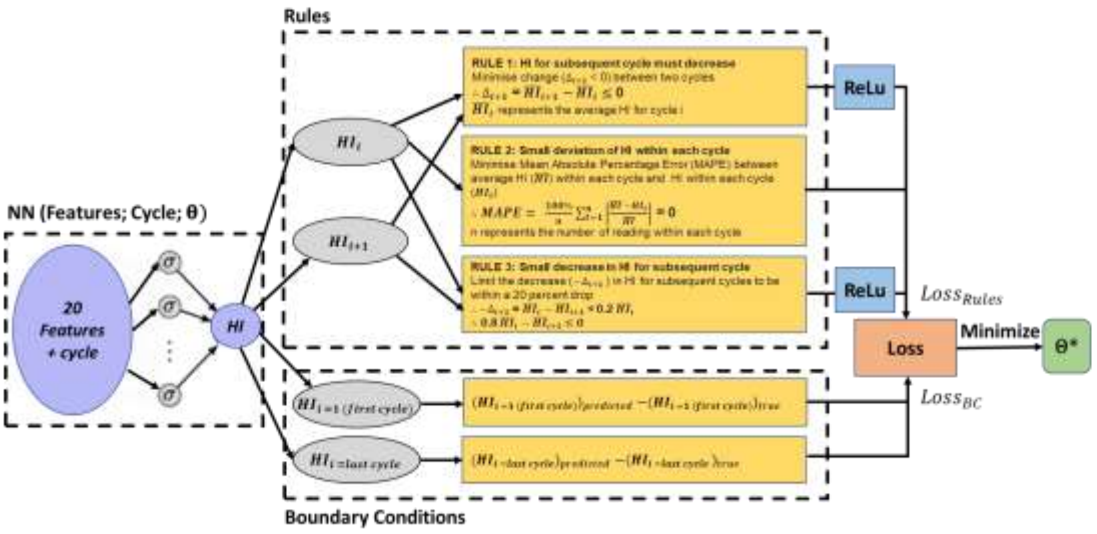

Rules
NN (Features; Cycle; θ)
20 Features + cycle
σ
HI
HI_i
HI_{i+1}
RULE 1: HI for subsequent cycle must decrease
Minimise change ($\Delta_{i+1} < 0$) between two cycles
$\therefore \Delta_{i+1} = HI_{i+1} - HI_i \leq 0$
HI_i represents the average HI for cycle i
RULE 2: Small deviation of HI within each cycle
Minimise Mean Absolute Percentage Error (MAPE) between average HI ($\overline{HI}$) within each cycle and HI within each cycle (HI_i)
$\therefore MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{\overline{HI}-HI_t}{\overline{HI}}\right| = 0$
n represents the number of reading within each cycle
RULE 3: Small decrease in HI for subsequent cycle
Limit the decrease ($-\Delta_{i+1}$) in HI for subsequent cycles to be within a 20 percent drop
$\therefore -\Delta_{i+1} = HI_i - HI_{i+1} < 0.2\,HI_i$
$\therefore 0.8\,HI_i - HI_{i+1} \leq 0$
ReLu
ReLu
$Loss_{Rules}$
Loss
Minimize
θ*
$Loss_{BC}$
$HI_{i=1\ (first\ cycle)}$
$HI_{i=last\ cycle}$
$(HI_{i=1\ (first\ cycle)})_{predicted} - (HI_{i=1\ (first\ cycle)})_{true}$
$(HI_{i=last\ cycle})_{predicted} - (HI_{i=last\ cycle})_{true}$
Boundary Conditions

Figure 19: Overall Physics-Based Machine Learning Model

A shallow neural network with only one hidden layer that has 50 neurons, and a hyperbolic tangent activation function was used. The shallow neural network was chosen due to the anticipated simplicity of the problem. In general, the neural network should provide sufficient approximation capacity to provide for the complexity of Health Index prediction.

Although more systematic optimisation techniques such as Bayesian optimisations have been employed to fine-tune the design of the neural network, the limited theoretical knowledge about the interplay between the neural architecture/training procedure and the complexity of the underlying rules is still poorly understood. Therefore, the widely adopted Adam optimiser is selected. Adam is an efficient optimisation technique for gradient descent and is frequently adopted for large problem involving a lot of data or parameters as it requires less memory and is more efficient.

Each of the 8 induction motors trains its own neural network. Features from each of the motors are processed through all 8 neural networks. When features from all 8 motors run through its own neural network, it shows a similar trend. Figure 20 illustrates the trend of the Health Index with features from Motor 1 is processed through the neural network trained by Motor 1.

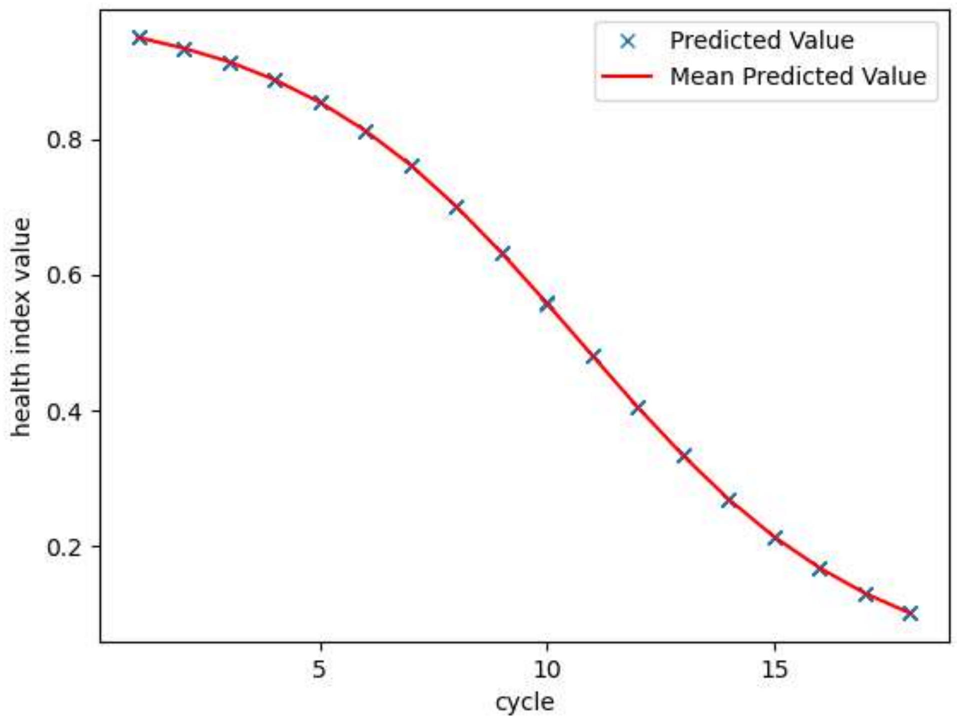


Figure 20: Health Index against Cycle for Motor 1

From Figure 20, it can be concluded that with a small, labelled training data set (first and last cycle), physical based machine learning is able to produce predictions of the Health Index that adheres to the rules imposed. Figure 1 accurately captures the gradual decrease for subsequent cycles, the small deviation in Health Index predictions for the same cycle and the boundary conditions (first cycle is 100% healthy and last cycle is 10% healthy).

3.3.2.4. Code

The above example is achieved using the TensorFlow library. All the codes accompanying explanations are available on the following Github repository:

https://github.com/nicholassung97/PBML/blob/main/engine%20problem_trial6.py

In the following section, the backbone of the code will be explained. The approach taken was to create a class that bundles important data and functionalities are together.

Code	Explanation
`class PhysicsInformedNN:`	- Create class
`def __init__(self, feature_b_train, cycle_b_train, hi_b_train, feature_f_train, cycle_f_train, layers):`	- Instantiate the class to create local objects in the class

`# Features for first and last cycle`
`self.feature_b_train = feature_b_train`
	- Instantiate the features for first and last cycle

`# Cycles for first and last cycle`
`self.cycle_b_train = cycle_b_train`
	- Instantiate the cycles for first and last cycle

`# HI value at first(HI=1) and last(HI=0.1) cycle`
`self.hi_b_train = hi_b_train`
	- Instantiate the HI value for the first and last cycle

`# Features for collocation points`
`self.feature_f_train = feature_f_train`
	- Instantiate the features for the collocation points

`# Cycles for collocation points`
`self.cycle_f_train = cycle_f_train`
	- Instantiate the cycles for the collocation points

`# Neurons in each layer`
`self.layers = layers`
	- Instantiate the number of neurons in each layer

`# Initialize weights and bias`
`self.weights, self.biases = self.initialize_NN(layers)`
	- Instantiate the weights and biases of the neural network using the initialize_NN method (discussed later)

`# Predict the HI value for the first and last cycle`
`self.hi_pred = self.net_hi(self.feature_b_train_tf, self.cycle_b_train_tf)`
	- Input the feature and cycle for the first and last cycle into the method net_hi (discuss later) to predict the HI value for the first and last cycle

`# Rule 1 residual`
`self.cost1_pred = self.net_cost1(self.feature_f_train_tf, self.cycle_f_train_tf)`
	- Input the collocation features and cycles values into the method net_cost1 (discuss later) to calculate the residual of Rule 1

```
  # Rule 2 residual
  self.cost2_pred = self.net_cost2(self.feature_f_train_tf, self.cycle_f_train_tf)

  # Rule 3 residual
  self.cost3_pred = self.net_cost3(self.feature_f_train_tf, self.cycle_f_train_tf)


# Formulate the loss function
  self.loss0 = tf.reduce_mean(tf.square(self.hi_b_train_tf - self.hi_pred))
  self.loss1 = tf.reduce_mean(tf.square(self.cost1_pred))
  self.loss2 = tf.reduce_mean(tf.square(self.cost2_pred))
  self.loss3 = tf.reduce_mean(tf.square(self.cost3_pred))
  self.loss = self.loss0 + self.loss1 + self.loss2 + self.loss3
  # Select type of optimiser
  self.optimizer = tf.contrib.opt.ScipyOptimizerInterface(self.loss,method='Adam')
```

- Input the collocation features and cycles values into the method net_cost2 (discuss later) to calculate the residual of Rule 2
- Input the collocation features and cycles values into the method net_cost3 (discuss later) to calculate the residual of Rule 3

- Obtain the loss function which is shown in Equation 13

- Select the type of optimiser to use. In this case, the optimiser used is Adam.

```
def initialize_NN(self, layers):
    weights = []
    biases = []
    num_layers = len(layers)
    for l in range(0,num_layers-1):
        W = self.xavier_init(size=[layers[l], layers[l+1]])
        b = tf.Variable(tf.zeros([1,layers[l+1]], dtype=tf.float32), dtype=tf.float32)
        weights.append(W)
        biases.append(b)
    return weights, biases
```

- The method initialize_NN is used to initialise weights and biases using Xavier initialisation

```
def neural_net(self, X, weights, biases):
    num_layers = len(weights) + 1
    H = 2.0*(X - self.lb)/(self.ub - self.lb) - 1.0
    for l in range(0,num_layers-2):
        W = weights[l]
        b = biases[l]
```

- The method neural_net is used to produce the output for the displacement in the forward propagation method. As established in Section 2.4.1, each of the nodes in the previous layer are multiplied by their respective weights and then summed with the

`        H = tf.tanh(tf.add(tf.matmul(H, W), b))`
`    W = weights[-1]`
`    b = biases[-1]`
`    Y = tf.add(tf.matmul(H, W), b)`
`    return Y`
	bias, it will form the value of the weighted sum.

`def net_hi(self, feature, cycle):`
`  hi = self.neural_net(tf.concat([feature, cycle], 1), self.weights, self.biases)`
`  return hi`
	The method net_hi is used to predict the HI value at the first and last cycle given the input is features and cycles of the first and last cycle

`def net_cost1(self, feature, cycle):`
`total_change = []`
`for i in range(1, self.max_cycle):`
`feature_cycle1 = feature[cycle_f_train[:,0] == i]`
`cycle_cycle1 = cycle[cycle_f_train[:, 0] == i]`
`feature_cycle2 = feature[cycle_f_train[:,0] == i+1]`
`cycle_cycle2 = cycle[cycle_f_train[:, 0] == i+1]`
`hi_cycle1 = self.net_hi(feature_cycle1, cycle_cycle1)`
`hi_cycle2 = self.net_hi(feature_cycle2, cycle_cycle2)`
`change = tf.math.reduce_mean(hi_cycle2) - tf.math.reduce_mean(hi_cycle1)`
`total_change.append(change)`
`total_change = tf.stack(total_change)`
`cost1 = tf.keras.activations.relu(total_change)`
`return cost1`
	The method net_cost1 is similar to the pseudocode presented in section 3.3.2.1.1 and is used to output the residual of the Rule 1 (Equation 9).

`def net_cost2(self, feature, cycle):`
`total_mape = []`
`for i in range(1, self.max_cycle+1):`
`feature_cycle1 = feature[cycle_f_train[:, 0] == i]`
`cycle_cycle1 = cycle[cycle_f_train[:, 0] == i]`
`hi_pred = self.net_hi(feature_cycle1, cycle_cycle1)`
`hi_true = tf.math.reduce_mean(hi_pred)`
`mape = tf.stack(tf.keras.metrics.mean_absolute_percentage_error(hi_true, hi_pred))`
	The method net_cost2 is similar to the pseudocode presented in section 3.3.2.1.2 and is used to output the residual of the Rule 2 (Equation 10).

`total_mape = tf.concat([total_mape, mape],0)`
`cost2 = tf.stack(total_mape)`
`return cost2`	

`def net_cost3(self, feature, cycle):`
`total_drop = []`
`for i in range(1, self.max_cycle):`
`feature_cycle1 = feature[cycle_f_train[:, 0] == i]`
`cycle_cycle1 = cycle[cycle_f_train[:, 0] == i]`
`feature_cycle2 = feature[cycle_f_train[:, 0] == i + 1]`
`cycle_cycle2 = cycle[cycle_f_train[:, 0] == i + 1]`
`hi_cycle1 = self.net_hi(feature_cycle1, cycle_cycle1)`
`hi_cycle2 = self.net_hi(feature_cycle2, cycle_cycle2)`
`drop = 0.8 * tf.math.reduce_mean(hi_cycle1) - tf.math.reduce_mean(hi_cycle2)`
`total_drop.append(drop)`
`total_drop = tf.stack(total_drop)`
`cost3 = tf.keras.activations.relu(total_drop)`
`return cost3`
	The method net_cost2 is similar to the pseudocode presented in section 3.3.2.1.3 and is used to output the residual of the Rule 3 (Equation 11).

`def train(self):`
`  tf_dict = {self.feature_b_train_tf: self.feature_b_train, self.cycle_b_train_tf: self.cycle_b_train, self.hi_b_train_tf: self.hi_b_train, self.feature_f_train_tf: self.feature_f_train,self.cycle_f_train_tf: self.cycle_f_train}`
`  self.optimizer.minimize(self.sess, feed_dict=tf_dict, fetches=[self.loss], loss_callback=self.callback)`
	The method train is used to train the neural network by minimise the loss function (Equation 13). When this is achieved, the weights and biases in the neural network will be finalised.

`def predict(self, feature, cycle):`
`hi_star = self.sess.run(self.hi_pred, {self.feature_b_train_tf: feature, self.cycle_b_train_tf: cycle})`
`cost1_star = self.sess.run(self.cost1_pred, {self.feature_f_train_tf: feature, self.cycle_f_train_tf: cycle})`
	The method predict is used to predict the HI value when feature and cycle is input.

```
cost2_star = self.sess.run(self.cost2_pred, {self.feature_f_train_tf: feature, self.cycle_f_train_tf: cycle})
cost3_star = self.sess.run(self.cost3_pred, {self.feature_f_train_tf: feature, self.cycle_f_train_tf: cycle})
return hi_star, cost1_star, cost2_star, cost3_star
```

3.3.3. Ensemble Approach (HI -> RUL)

When features from all 8 motors run through its own neural network, the error is low as tabulated below.

Table 8: Training loss value of each motor

Motor No	**Loss**
1	2.91×10^{-5}
2	7.02×10^{-5}
3	1.61×10^{-5}
4	1.78×10^{-5}
5	2.75×10^{-5}
6	1.04×10^{-5}
7	1.81×10^{-5}
8	2.73×10^{-5}

Since the error is low when features from all 8 motors run through its own neural network, the HI degradation curve of the training motor can be assumed as the true. Hence, the estimated RUL ($\widehat{RUL}$) can be obtained by mapping the predicted HI value of each cycle for the testing motor to the HI degradation curve of the training motor. This process can often come in the form of both interpolation and extrapolation.

Case 1: Interpolation

If the predicted HI value from each cycle of the test motor is within the range of the training HI degradation curve, this HI value will be mapped onto the training HI degradation curve to obtain the estimated cycle ($\hat{t}$). The estimated remaining useful life can then be obtained by:

$$\widehat{RUL}(t) = T - \hat{t} \tag{14}$$

Where T is the total lifetime of the training motor and $\widehat{RUL}(t)$ is the predicted RUL of the testing motor.

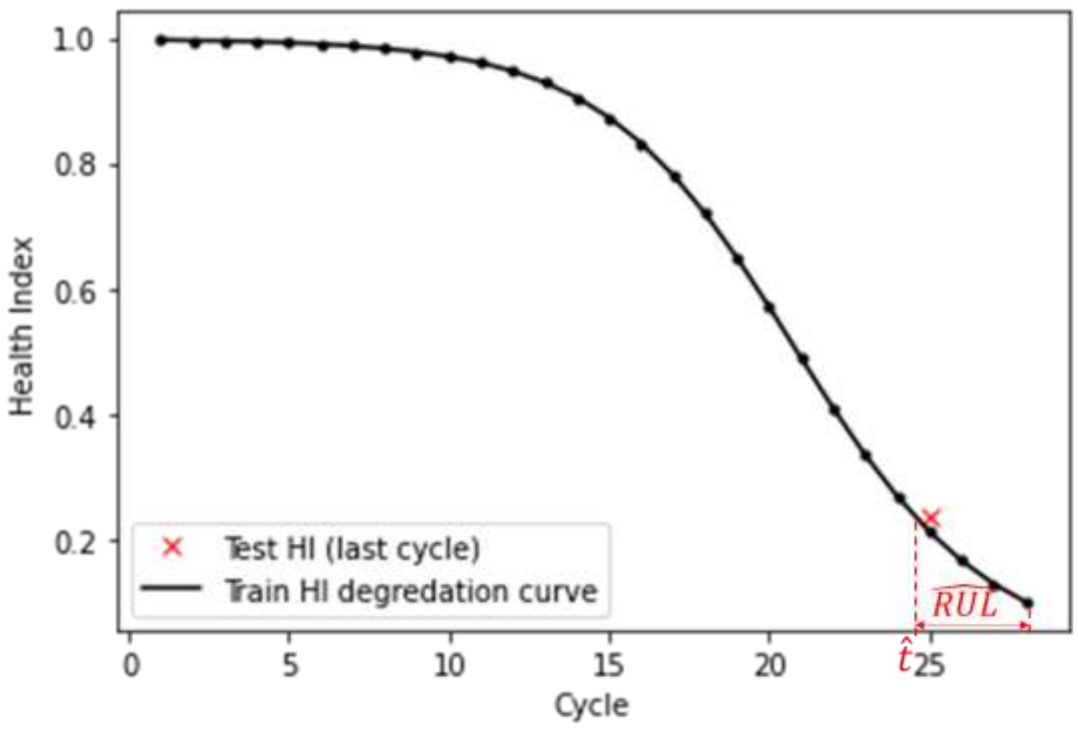


Figure 21: Interpolation of HI to obtain RUL

In the above example, the features in the 25th cycle from motor 8 (test) was input into the physics-based machine learning model trained by motor 2 (lifetime of 28 cycles). Since the predicted Health Index value of 0.234 (represented as the red cross) falls within the range of the training HI degradation curve (represented as the black curve), the estimated cycle ($\hat{t}$) can be interpolated as 24.7. Hence the estimated remaining useful life for motor 8 at the 25th cycle when passed through motor 2's neural network at will be:

$$\widehat{RUL}(t) = 28 - 24.7 = 3.3$$

Case 2: Extrapolation

If the predicted HI value from each cycle of the test motor is outside the range of the training HI degradation curve, this HI value will be mapped onto an extended line (formed by the last two cycles) to obtain the estimated cycle ($\hat{t}$). The estimated remaining useful life can then be obtained by Equation 14.

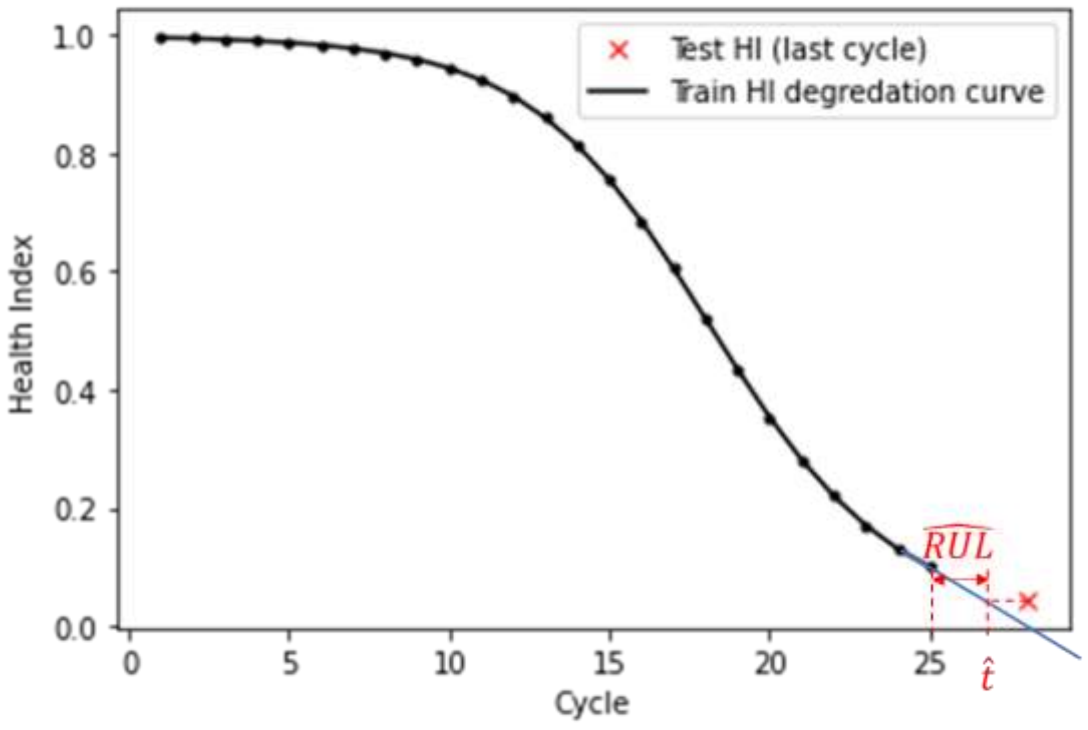


Figure 22: Extrapolation of HI to obtain RUL

In the above example, the features in the 28th from motor 2 (test) was input into the physics-based machine learning model trained by motor 8 (lifetime of 25 cycles). Since the predicted Health Index value of 0.0444 (represented as the red cross) falls outside the range of the training HI degradation curve (represented as the black curve), an extended line formed by the last two cycles (represented by the blue line) is used to extrapolate the estimated the estimated cycle ($\hat{t}$) to be 26.8.

Hence the estimated remaining useful life for motor 2 at the 28th cycle when passed through motor 8's neural network at will be:

$$\widehat{RUL}(t) = 25 - 26.8 = -1.8$$

Each motor will train its individual HI Prediction Model using the physics-based machine learning model. Hence, there will be 8 HI prediction models in total. The features from one test motor will run through the other 7 HI prediction models and the final estimated RUL was averaged to obtain the final remaining useful life prediction. This process is illustrated in the figure below.

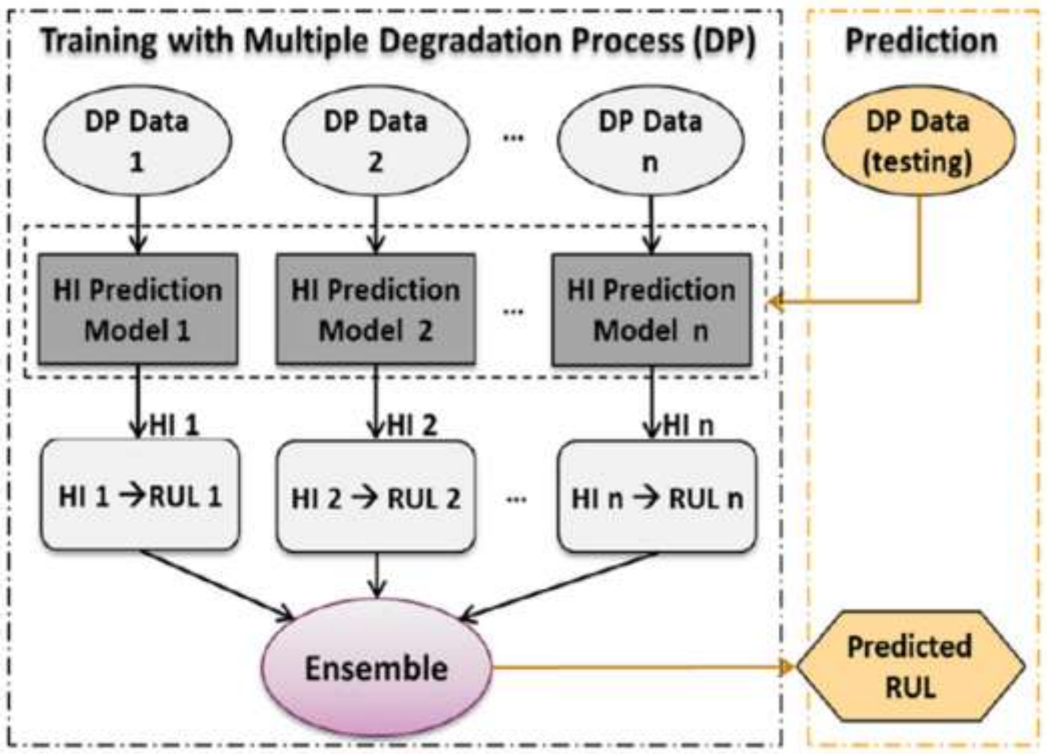


Figure 23: Ensemble approach

3.4. Performance Evaluation

To assess the performance of the physics-based machine learning model in RUL prediction, a cross-validation method was used. This means that one motor is used as the testing test whilst the others are used for training. The two main metrics that will be used is the Root Mean Square Error (RMSE) and Mean Absolute Percentage Error (MAPE). Given the predicted RUL ($\widehat{RUL}(t)$) and the true RUL ($RUL(t)$) at the cycle t (t = 1, 2, …., T), the metrics are defined as follows.

(1) Root Mean Square Error (RSME):

$$RSME = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\epsilon(t))^2} \tag{15}$$

Where $\epsilon(t) = RUL(t) - \widehat{RUL(t)}$ is the difference between the true RUL and the predicted RUL at cycle t

(2) Mean Absolute Percentage Error (MAPE): to quantify the mean error in percentage

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{100 \cdot \epsilon(t)}{RUL(t)}\right| \tag{16}$$

4. Results

Table 9 tabulates the RMSE and MAPE metrics and compares them with the conventional direct way of RUL predictions collected by Yang et.al [58]. The direct method does not utilise a two-stage mapping process with Health Index as the intermediary and the features are directly mapped to the RUL during training. The last row in the table shows the averaged RMSE and MAPE in predicting all of the 8 motors. The smaller RMSE and MAPE value for each motor is bolded for the sake of easier comparison.

Table 9: Performance Evaluation of RUL predictions

Motor	*Method*	*RMSE*	*MAPE*
1 (18 cycles)	Physics-Based	9.23	**80.07**
	Direct	**7.54**	150.74
2 (28 cycles)	Physics-Based	**1.63**	**36.79**
	Direct	3.91	38.55
3 (26 cycles)	Physics-Based	**1.83**	**33.25**
	Direct	2.71	51.67
4 (29 cycles)	Physics-Based	**2.42**	51.08
	Direct	6.32	**24.10**
5 (28 cycles)	Physics-Based	**1.64**	**36.78**
	Direct	3.37	82.33
6 (27 cycles)	Physics-Based	**2.35**	31.64
	Direct	2.36	**24.18**
7 (27 cycles)	Physics-Based	**1.36**	**31.39**
	Direct	4.93	95.52
8 (25 cycles)	Physics-Based	2.68	**39.34**
	Direct	**2.58**	45.76
Average	Physics-Based	**2.89**	**42.54**
	Direct	4.22	64.11
	Percentage Reduction	31.5%	33.6%

The results clearly show that the RUL prediction using physics-based machine learning out-performs the direct method for most engines. The superiority of the physics-based machine learning is further depicted by the average RMSE and MAPE values whereby RMSE always outperformed the direct method. The average performance improves by 31.5% (2.89 vs 4.22) using RMSE and 33.6% (42.54 VS 64.11) using MAPE. Further comparisons on the prediction performance of some individual motors showed significant improvement with the physics-based machine learning degradation model, in some cases, exceeding 200%. From the results in Table 10, in predicting 8 motors, the RMSE of 6 motors has been improved, ranging from 0.4% (Motor 6, 2.35 vs 2.36) to 262% (Motor 7, 1.36 vs 4.93).

It is also observed that there were a few cases where the RMSE has not improved. Using RMSE, motor 1 and motor 8 performance decreased by 22.4% (9.23 vs 7.54) and 3.9% (2.68 vs 2.58). However, just by looking at the overall performance, the physics-based machine learning method definitely improved predictions.

5. Conclusions

This paper investigated the feasibility of using physics-based machine learning for predicting the Remaining Useful Life. The principles and effectiveness of physics-based machine learning was first showcased in a simple kinematics problem. The results highlighted the robustness of physics-based machine learning as it required less training data whilst making predictions that adhered to the physical laws. Next, using the proposed prognostics framework with Health Index formulation and realization with multi-model ensemble, this paper investigated physics-based machine learning for predicting the Remaining Useful Life. Utilizing the imposition of rules on the loss function as a regularization agent, the results showed that the proposed approach achieved good performance in predicting the RULs and found it to be more than 30% effective than the direct method.

6. Future Work

In subsequent studies, the initialization of weights and biases of the neural network could be explored to improve the optimization process. Additionally, an efficient method of adjusting the hyperparameters for each of the loss function components could be investigated. Lastly, the application of the proposed prognostics method to other real-world problems will be explored.

References

[1] K. D. a. K. S. D. Tran Anh, “The Predictive Maintenance Concept in the Maintenance Department of the “Industry 4.0” Production Enterprise,” *Foundations of Management,* vol. 10, no. 1, pp. 283-292, 2018.

[2] A. J. Clark, “Prognostics and Health Management,” [Online]. Available: https://calce.umd.edu/prognostics-and-health-management.

[3] M. M. a. A. K. V. Roblek, “"A Complex View of Industry 4.0,” *SAGE Open,* vol. 6, no. 2, 2016.

[4] F. C. a. H. H. Y. Chiu, “Developing a factory-wide intelligent predictive,” *Journal of the Chinese Institute of Engineers,* vol. 40, no. 7, pp. 562-571, 2017.

[5] Y. M. L. L. a. L. L. J. Yan, “Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance,” *IEEE Access,* vol. 5, pp. 23484-23491, 2017.

[6] K. G. Matthew J. Daigle, “Model-Based Prognostics With Concurrent Damage Progression Processes,” *IEEE Xplore,* pp. 535-546, 2013.

[7] R. Analytics, “Model-Driven vs Data Driven methods for Working with Sensors and Signals,” 20 Jul 2016. [Online]. Available: https://medium.com/@RealityAI/model-driven-vs-data-driven-methods-for-working-with-sensors-and-signals-98fc6ac8cc92. [Accessed 2022].

[8] M. &. G. K. Schwabacher, “Survey of artificial intelligence for prognostics.,” *AAAI Fall Symposium - Technical Report.* , p. 2, 2007.

[9] F. K. Linxia Liao, “Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction,” *IEEE Transactions of Reliability,* vol. 63, no. 1, pp. 192-194, 2014.

[10] O. Fink, “Data-driven intelligent predictive maintenance of industrial assets,” *Women in industrial and systems engineering,* p. 589–605, 2020.

[11] L. Swanson, “Linking maintenance strategies to performance.,” *Int. J. Prod. Econ.,* p. 237–244, 2001.

[12] D. J. Wilkins, “The Bathtub Curve and Product Failure Behavior,” *Reliability Engineering Resources,* no. 21.

[13] Y. Z. X. L. P. W. Y. a. D. R. Ran, “A survey of predictive maintenance: systems, purposes and approaches.,” ArXiv, 2019.

[14] Q. M. G. a. F. O. Wang, “Domain adaptive transfer learning for fault diagnosis,” in *2019 prognostics and system health management conference,* Paris, France, 2019.

[15] S. Anunaya, “Analytics Vidhya,” 10 Auguest 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/data-preprocessing-in-data-mining-a-hands-on-guide/.

[16] “Feature extraction for machine learning and deep learning,” MathWorks, [Online]. Available: https://www.mathworks.com/discovery/feature-extraction.html#:~:text=Feature%20extraction%20refers%20to%20the,directly%20to%20the%20raw%20data..

[17] I. G. S. N. M. a. Z. L. A. Guyon, “Feature extraction: foundations and applications (studies in fuzziness and soft computing),” in *Heidelberg: Springer-Verlag,* Berlin, 2006.

[18] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Comput. Vis. ,* vol. 60, pp. 91-110, 2004.

[19] H. E. A. T. T. a. V. G. L. Bay, “Speeded-up robust features (surf).,” *Comput. Vis. Image Understand,* vol. 110, p. 346–359, 2008.

[20] S. a. M. P. Davis, “ Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Trans. Acoust. Speech Signal Process.,* vol. 28, pp. 357-366, 1980.

[21] S. K. a. L. M. Kopparapu, “Choice of mel filter bank in computing mfcc of a resampled speech,” in *10th international conference on information science, signal processing and their applications (ISSPA 2010),* Kuala Lumpur, Malaysia, 2010.

[22] B. &. L. C. Soro, “ A wavelet scattering feature extraction approach for deep neural network based indoor fingerprinting localization.,” *Sensors 19,* vol. 8, 2019.

[23] Y. Y. B. J. X. J. F. L. N. a. N. A. K. Lei, “Applications of machine learning to machine fault diagnosis: a review and roadmap.,” *Mech. Syst. Signal Process.,* p. 138, 2020.

[24] A. a. Y. B.-S. Widodo, “ Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors,” *Expert Syst. Appl.,* vol. 33, pp. 241-250, 2007.

[25] D. a. M. J. Logan, “Using the correlation dimension for vibration fault diagnosis of rolling element bearings-i. Basic concepts.,” *Mech. Syst. Signal Process.,* vol. 10, pp. 241-250, 1996.

[26] I. G. S. N. M. a. Z. L. A. Guyon, “Feature extraction: foundations and applications (studies in fuzziness and soft computing,” in *Heidelberg: Springer-Verlag.*, Berlin, 2006.

[27] N. C. Q. Z. Y. H. W. W. K.L. Tsui, “Prognostics and health management: a review on data driven approaches,” *Math. Problems Eng,* pp. 1-7, 2015.

[28] W. Q. E. L. A. &. P. F. G. Meeker, Statistical methods for reliability data, John Wiley & Sons, 1998.

[29] Y. G. &. N. P. Li, “Gas turbine performance prognostic for condition-based maintenance,” *Applied energy,* vol. 10, no. 86, pp. 2152-2161, 2009.

[30] T. K. S. L. Y. Li, “Stochastic prognostics for rolling element bearings,” *Mechanical Systems and Signal Processing,* no. 14 , pp. 747-762, 2000.

[31] A. a. J. W. H. Usynin, “Use of linear growth models for remaining useful life prediction,” in *The Maintenance and Reliability Conference (MARCON 2007),* May 2007.

[32] C. J. a. W. O. M. Lu, “Using degradation measures to estimate a time-to-failure distribution.,” *Technometrics,* vol. 35, pp. 543-559, 1993.

[33] M. L. R. L. J. R. N.Z. Gebraeel, “Residual-life distributions from component degradation signals: A Bayesian approach,” *IIE Transactions,* vol. 37, pp. 543-557, 2005.

[34] H. M. D.R. Cox, The Theory of Stochastic Processes, London: Methuen and Company, 1965.

[35] M. Abdel-Hameed, “A Gamma wear process,” *IEEE Transactions on Reliability,* vol. 24, pp. 152-153, 1975.

[36] W. W. C. H. H. D. H. Z. Xiao-ShengSi, “Remaining useful life estimation – A review on the statistical data driven approaches,” *European Journal of Operational Research*, vol. 213, no. 1, pp. 1-7, 2011.

[37] Y. a. N. K. Shao, “Prognosis of remaining bearing life using neural networks,” *Proc. IME J. Syst. Contr. Eng,* vol. 214, p. 217–230, 2000.

[38] P. &. V. G. Wang, “Fault prognostics using dynamic wavelet neural networks,” *Artificial Intelligence forEngineering Design and Manufacturing,* vol. 15, pp. 163-171, 2001.

[39] M. H. L. M. J.Z. Sikorska, “Prognostic modelling options for remaining useful life estimation by industry,” *Mechanical Systems and Signal Processing,* vol. 25, no. 5, pp. 1803-1836, 2011.

[40] H.-Z. W. H.-K. L. Y.-F. Z. L. a. L. Z. Huang, “Support vector machine based estimation of remaining useful life: current research status and future trends,” *J. Mech. Sci. Technol,* vol. 29, p. 151–163, 2015.

[41] C. Z. Z. a. H. Z. Sun, “Research on bearing life prediction based on support vector machine and its application.,” in *J. Phys.: Conf. Ser.*, 2011.

[42] R. C.-M. B. M. S. L. E. F. F. a. Z. N. Khelif, “Direct remaining useful life estimation based on support vector regression,” *IEEE Trans. Ind. Electron,* vol. 64, p. 2276–2285, 2017.

[43] C. S. L. F. R.-P. J. a. J. F. J. d. C. Ordóñez, “A hybrid ARIMA-SVM model for the study of the remaining useful life of aircraft engines.,” *J. Comput. Appl. Math. ,* vol. 346, p. 184–191, 2019.

[44] R. a. S. Satishkumar, “Remaining life time prediction of bearings through classification using decision tree algorithm,” *Int. J. Appl. Eng. Res.,* vol. 10, p. 34861–34866., 2015.

[45] Z. P. J. D. K. G. K. L. H. C. B. e. a. Zheng, “A novel method for lithium-ion battery remaining useful life prediction using time window and gradient boosting decision trees,” in *2019 10th international conference on power electronics and ECCE Asia (ICPE 2019 - ECCE Asia),* Busan, South Korea, 2019.

[46] V. T. T. S. V. R. B. M. a. K. M. G. Mathew, “Prediction of remaining useful lifetime (rul) of turbofan engine using machine learning,” in *2017 IEEE international conference on circuits and systems (ICCS),* Thiruvananthapuram, 2017.

[47] M. K. L. M. S. J.Lou, “Model-based prognostic techniques [maintenance applications],” in *International Automatic Testing Conference, AUTOTESTCON*, 2013.

[48] N. A. a. S. Administration, “Water Recycling System Prognostics,” [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/water-recycling-system-prognostics/modeling/nominal-system-modeling/.

[49] W. K. Yu and Harris, “A new stress-based fatigue life model for ball bearings,” *Tribology Transactions*, vol. 1, no. 44, pp. 11-18, 2001.

[50] Y.-H. L. C. H. S. L. S. H. A. Downey, “Physics-based prognostics of lithium-ion battery using non-linear least squares with dynamic bounds,” *Reliab. Eng. Syst. Saf*, vol. 182, pp. 1-12, 2019.

[51] P. Paris and F. Erdogan, “Closure to "Discussions of 'A Critical Analysis of Crack Propagation Laws,” *Journal of Basic Engineering*, vol. 85, no. 4, pp. 528-534, 1963.

[52] K. G. M.J. Daigle, “Model-based prognostics with concurrent damage progression processes,” *IEEE Trans. Syst., Man, Cybern.: Syst*, vol. 43, no. 3, pp. 535-546, 2013.

[53] A. S. S. S. B. S. J. C. Kai Goebel, Prognostic Performance Metrics, Chapman and Hall/CRC, 2012.

[54] A. &. S. G. Bonnett, “Cause and Analysis of Stator and Rotor Failures in Three-Phase Squirrel-Cage Induction Motors,” *IEEE Transactions on Industrial Electronics*, vol. 4, no. 28, pp. 921-937, 1992.

[55] B. F. W. B. F. &. H. Y. C. P. It. [Online]. Available: https://www.ibtinc.com/causes-of-bearing-failure/.

[56] S. Soergel and P. Rastgoufard, “An Analysis of Induction Motor Predictive Maintenance Techniques,” in *28th Southeastern Symposium on System Theory*, Los Alamitos, CA, United States, 1996.

[57] P. G. M. Raissi, “Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving non-linear partial differential equations,” *Journal of Computational Phyiscs,* vol. 378, pp. 686-707, 2019.

[58] M. S. H. T. Z. Z. X. P. L. a. S. N. F. Yang, “Health Index-Based Prognostics for Remaining Useful Life Predictions in Electrical Machines,” *IEEE Transactions on Industrial Electronics,* vol. 63, no. 4, 2016.

[59] M. Sharp, “Prognostic approaches using transient monitoring methods,” University of Tennessee, Knoxville, 2012.

[60] X. Y. Q. Q. Wu, “Pattern recognition and its application in fault diagnosis of electromechanical system,” *J. Inf. Comput. Sci.,* vol. 9, no. 8, pp. 2221-2229, 2012.

[61] J. P. M. a. L. Massaron, “Training, Validating, and Testing in Machine Learning,” 10 06 2016. [Online]. Available: https://www.dummies.com/article/technology/information-technology/ai/machine-learning/training-validating-testing-machine-learning-226757/.

[62] D. S. M.-P. B. R. S. S. K. S. S. M. Bruckner, “An introduction to opc ua tsn for industrial communication systems,” *Proc. IEEE,* vol. 107, p. 1121–1131, 2019.

[63] L. Hadjileontiadis, “Lung Sounds: An Advanced Signal Processing Perspective,” *Morgan & Claypool,* 2008.