Improved Remaining Useful Life Prediction Models For Condition-based Maintenance Planners: A Data-Driven Approach Using Transformer-Based Architectures

by

Oluwaseyi A. Ogunfowora
B.Eng., Landmark University, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTERS OF APPLIED SCIENCE

in the Department of Mechanical Engineering
University of Victoria

Improved Remaining Useful Life Prediction Models For Condition-based Maintenance Planners:
A Data-Driven Approach Using Transformer-Based Architectures

by

Oluwaseyi A. Ogunfowora
B.Eng., Landmark University, 2018

Supervisory Committee

Dr. Homayoun Najjaran, Supervisor Main, Supervisor
(Department of Mechanical Engineering)

Dr. Zuomin Dong, Departmental Member
(Department of Mechanical Engineering)

ABSTRACT

Systems and machines undergo various failure modes that result in their health degradation, so maintenance actions are required to restore them back to a state where they can perform their expected functions. Machine health degradation is inevitable, and so is the maintenance cost associated with it. However, with proper maintenance plans, maintenance costs can be reduced, machine life can be extended, and ultimately we can ensure workplace safety.
The field of prognostics is vital to systems health management and proper maintenance planning. A reliable estimation of the remaining useful life (RUL) of machines holds the potential for substantial cost savings.
Data-driven methods for predictive maintenance have been recognized as one of the most promising maintenance strategies because of their high efficiency and low cost compared to other strategies.

This work uses a sequential approach through experimentation to investigate the two main machine-learning-based methods for remaining useful life prediction, the similarity-based and direct-approximation methods. Drawing insights from existing works in the literature, the two stages of development of a similarity-based model (SBM) were optimized resulting in the development of improved similarity-based models using supervised and unsupervised machine learning methods for the health index construction. Ultimately, this work proposes a novel remaining useful life estimation model that leverages the concept of Large Language Models (LLMs) for more efficient time series data representation learning and prediction applied to the remaining useful life prediction use case. The experimental results indicate that the proposed Encoder-Transformer architecture outperforms the existing state-of-the-art models.

Other highlights of this work include the bottom-up experimental approach taken to select the best methods and make improvements. The benefits of this approach can be seen from the improved remaining useful life prediction models developed in this work compared to their other counterparts in the literature and the insights this work provides. In this work, ten separate machine-learning models were developed, trained, and tuned for experimentation purposes.

To summarize, three improved RUL prediction models: an Encoder-Transformer direct-approximation-based model, an Improved Unsupervised Learning-based Similarity-based model with Principal Component Analysis (PCA), and a Transformer-Assisted Similarity-based model were developed in this work. These models rank first, second, and fourth best amongst the twelve state-of-the-art models they were compared in the literature.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank those who made this journey easier:

I want to thank God, for giving me the grace and strength to do this work.

My supervisor, Dr Homayoun Najjaran, for providing me with the platform and resources to complete this program. I do not take your support and guidance throughout this journey lightly and I am sincerely grateful.

NTWIST Inc., through the Natural Sciences and Engineering Research Council (NSERC) Canada Alliance Grant, for funding this research.

My family, especially my mum, for their constant support and words of encouragement in my lowest moments. Even though they do not fully understand what I do, they are always ready to listen to me rejoice, complain, and cry about my achievements and setbacks.

My friend, Chukwuemeka Odiba who encouraged me to start this journey and supported me through it. I am immensely grateful to you for your contributions towards this success.

My friends and team members in the ACIS lab, for adding sprinkles of hope and laughter to this two-year journey.

DEDICATION

I dedicate this work to my mum. Your love is my light, even across continents, it shines brighter than ever.

LAY SUMMARY

The degradation of machines, especially when they are in use is inevitable and so is the need for maintenance or replacement. Just like the light bulbs in our homes, after a while, they fail and we need to replace them. For less critical components like light bulbs, the run-to-failure maintenance strategy is preferred because the cost of replacement and safety implications are reduced. On the other hand, imagine the turbo engine of an airplane fails abruptly, it can have catastrophic financial and safety implications. So for more critical machines, it is important that we use preventive maintenance strategies.

Reliable prediction of when an intervention (maintenance or replacement) is required on a machine can serve as a decision-support tool for maintenance planners to develop efficient plans, reduce maintenance costs and ensure the safety of lives and properties. Machine learning models possess remarkably potent predictive capabilities so this work uses them to predict the remaining useful life of machines.

In recent times, a sub-field of machine learning, generative AI has made groundbreaking achievements in the field of Natural Language Processing leading to the development of intelligent chatbots like ChatGPT. These intelligent language models make use of a very powerful architecture known as **transformers**. In this work, we transfer knowledge from the domain of Large Language Models (LLMs) to the time series data domain for machine and component prognostics use cases.

PREFACE

The works documented in two publications have contributed to this thesis. The research was conducted under the supervision of Dr. Najjaran in the Advanced Control and Intelligent Systems (ACIS) Laboratory at the University of Victoria in partnership with NTWIST Inc., Canada supported by the Natural Sciences and Engineering Research Council (NSERC) Canada Alliance Grant ALLRP 555220 – 20.
Specifically, the two research publications and their research contributions include the following.

Ogunfowora O, Najjaran H. Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. Journal of Manufacturing Systems. 2023 Oct 1;70:244-63.

The motivation to develop robust and efficient remaining useful life prediction models for better state representations in reinforcement learning-based maintenance planners stemmed from a gap pointed out in a literature review on Reinforcement and Deep Reinforcement Learning-based Solutions for Machine Maintenance Planning, Scheduling Policies, and Optimization that was conducted by the author of this thesis under the supervision of Dr. Homayoun Najjaran.
The author of this thesis was responsible for the conceptualization, formal analysis, investigation, data curation, visualization, the writing of the original draft, review, and editing of this manuscript. The manuscript has been submitted to the Journal of Manufacturing Systems and it is in the final stages of revision. Parts of this publication were reproduced in the motivation section in chapter 1.

Ogunfowora O, Najjaran H. A Transformer-based Framework For Multivariate Time Series: A Remaining Useful Life Prediction Use Case. arXiv preprint arXiv:2308.09884. 2023 Aug 19.

Inspired by the recent developments in the generative AI sub-field of deep learning, the author of this work adopts the transformer models used for the development of Intelligent chatbots like ChatGPT and Bard for better time series prediction and representation learning. This work was conducted under the supervision of Dr Homayoun

Najjaran and the publication is under review.

The author of this thesis was also responsible for the conceptualization, formal analysis, investigation, data curation, visualization, the writing of the original draft, review, and editing of this manuscript. The first model developed in this manuscript; the one-stage prediction task applied to the remaining useful life prediction of aircraft turbo-engines use case was reproduced in chapter 4 of this thesis.

Chapter 1

Introduction

1.1 Background

Maintenance activities take up 15%-40% of the total production costs in factories [67]. Machines/assets undergo various failure modes that result in machine health degradation, affect performance, and eventually cause failure. Degradation of machines whether under working or non-working conditions is inevitable and so is the need for maintenance.
Maintenance, as defined by [17], is a set of activities used to restore an item to a state in which it can perform its designated functions. Proper maintenance actions help to reduce machine failures, improve their reliability, and reduce the maintenance and production costs associated with unplanned downtime of machines. While the cost associated with maintenance activities cannot be totally eliminated, a proper maintenance plan can help to minimize these costs.

The maintenance planning and scheduling problem just like most planning problems is an optimization problem with the aim of developing efficient maintenance policies and adequately allocating maintenance resources and tasks to the geospatial problem. As earlier mentioned, maintenance of machines is inevitable and that is why effective maintenance planning is paramount to ensure high asset availability with minimum cost. For most industries to remain competitive, they cannot afford the short or long-term costs and effects associated with inadequate planning of production and maintenance activities which can result in not meeting customer demands and loss of sales. Developing proper maintenance policies helps to reduce the costs as-

sociated with planned and unplanned downtime of machines and maintenance costs. The field of prognostics is vital to systems health management and proper maintenance planning. The reliable estimation of remaining useful life (RUL) holds the potential for substantial cost savings. This includes avoiding unscheduled maintenance and maximizing equipment usage, serving as decision-support systems (DSS) by providing decision-makers valuable information about the condition of the machine and helping them to plan accordingly. For instance, this can inform their decision to reduce the operational loads on the machines in order to extend their life span, these estimations can also enable planners to anticipate upcoming maintenance needs and initiate a seamless logistics process, facilitating a smooth transition from faulty equipment to fully functional ones.

Maintenance strategies are broadly classified into two categories: corrective maintenance (CM) and preventive maintenance (PM). The corrective maintenance can also be referred to as breakdown maintenance, this strategy adopts the run-to-failure approach; maintenance is only performed after machine failures, it is a reactive strategy. Usually, when machines fail under this policy, a replacement or major overhaul is required to get the machine back in good condition, and the costs and maintenance duration associated with corrective maintenance are usually very high. Aside from the costs associated with maintenance actions, a major drawback of the corrective maintenance strategy is that it does not provide the decision-makers an opportunity to plan the maintenance actions. Machine failures happen abruptly and interrupt factory/running operations, which can cause significant losses due to unplanned downtime and require considerably high maintenance costs and resources for prompt maintenance actions. It can also cause accidents, for these reasons, the corrective maintenance policy is not encouraged.

The preventive maintenance strategy, on the other hand, is to proactively shut down machines for maintenance to reduce failures and enhance the reliability of the machines. The preventive maintenance strategy can be further divided into two main categories which are, scheduled and condition-based maintenance.

Scheduled maintenance is a preventive maintenance-based strategy where select maintenance activities are carried out at predefined intervals, scheduled maintenance activities are planned, and they are performed regardless of whether signs of deterioration or failures are prevalent or not. Even though this policy causes fewer abrupt failures, it can be very aggressive and eventually result in incurring unnecessary costs such as

maintenance resource costs, and spare part costs for machines or assets that are still in good condition. If the maintenance plans are not optimized, the costs associated with scheduled maintenance and industry-related costs from interventions will result in huge losses.
Condition-based maintenance also referred to as predictive maintenance suggests conducting maintenance actions based on some measurements of the machine or asset prior to failure. Predictive maintenance has been recognized as one of the most promising maintenance strategies because of its high efficiency and low cost compared to other strategies [64]. This approach helps to eliminate the unnecessary maintenance costs incurred in the scheduled maintenance approach while significantly reducing unscheduled breakdowns because the maintenance decisions are made based on the changing, real-time machine health conditions. Numerous works have been done in literature to harness the capabilities of machine-learning-based predictive models to accurately predict machine failures and make diagnoses of the failure types.
As emphasized in [10], there is a fast-growing demand for Intelligent Manufacturing Execution Systems (IMES) and one of the key functionalities of an Industry 4.0-ready MES is an intelligent maintenance planner and scheduler. By having a reliable RUL prediction model, smart maintenance planners can be developed, recent maintenance optimization research uses Artificial Intelligence (AI) to plan and schedule maintenance actions.

Reinforcement learning (RL) is a data-driven optimization algorithm that can be used to develop effective maintenance policies and there has been an upsurge in the application of RL to plan maintenance in the literature in recent years.
Reinforcement Learning is one of the three (3) main paradigms of machine learning, the others being supervised and unsupervised learning. Contrary to the other paradigms; it adopts a trial-and-error method to make decisions. The Markov state model forms a basis for the formulation of the RL paradigm that follows the notion that the available information about the current state is sufficient to predict the next state. It is based on the hypothesis that the accumulation of rewards through learning to take a sequence of optimal actions at every state in an environment is the maximization of the expected cumulative reward [42].
Figure 1.1 shows the number of RL and Deep RL (DRL) based maintenance planning and optimization publications per year over the past thirteen years, the yearly analysis shows an evident upsurge in the use of RL and DRL for maintenance plan-

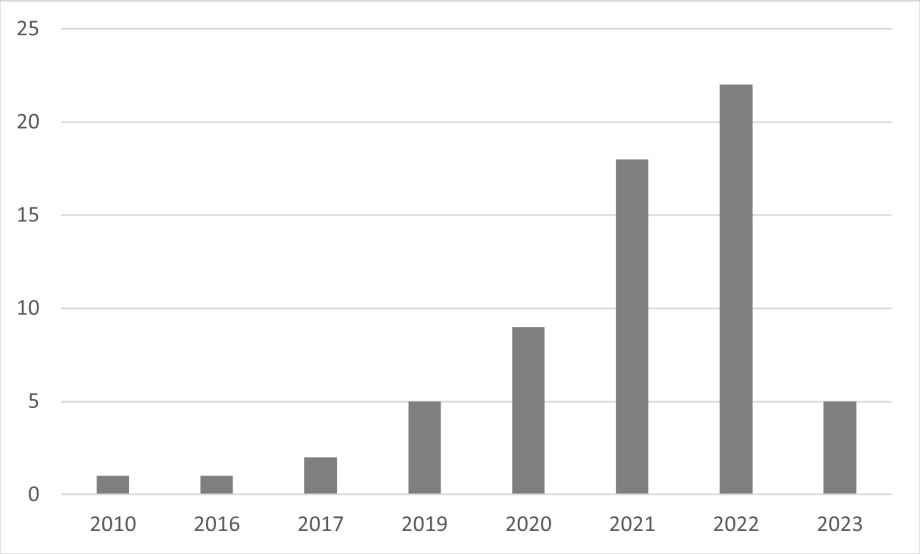


Figure 1.1: Relevant Publications/year

ning tasks between the years 2019 to 2023. There has been over an 80% increase in the number of RL and DRL-based publications for maintenance planning in the literature. This increase in the use of RL for maintenance planning is due to the increase in offline and real-time data from Internet of Things (IoT) devices and the high computing power that drives machine learning algorithms.
Reinforcement learning application to the maintenance planning problem introduces an efficient and smooth transition between data-driven, condition-based maintenance predictive models and maintenance optimization models. Reinforcement learning methods use a synergistic approach to combine the condition-based maintenance objective and the maintenance optimization objective(s) into a single problem formulation. So a condition-based maintenance planner is a combination of condition-based maintenance and maintenance optimization.

1.2 Motivation

At the beginning of this study, a literature review on RL and DRL-based Solutions for Machine Maintenance Planning, Scheduling Policies, and Optimization was conducted [42]. This literature review focused on the methodologies, findings, and well-defined interpretations of the reviewed studies while finding common ideas and concepts, identifying methodological problems, and pointing out the research gaps. It drew

insights from existing literature and defined some areas of future work.
One of the major gaps pointed out in this literature review was regarding the machine state representations. The translation of the maintenance planning problem formulation into the RL framework will involve the definition of the observable states, allowable actions, reward function, transition probabilities (model-based), or development of the simulation environment (model-free) and choosing the RL algorithm. The RL states are a representation of the environment accessible to the RL agent at any given instance, these states are constantly changing based on the interaction of the RL agent with the environment, it is very important for the state captured by the agent to have enough information required for the agent to learn good policies. For a condition-based maintenance planner where the agent is required to make optimal decisions based on the condition of the machines, it is expedient that the machine condition representations are optimal and truly depict the condition of the machine at any given time. It was observed from the reviewed publications in Appendix A that the Markov states can be represented in three major ways.

1. Health indicators: These are usually one-dimensional machine health index parameters that can be used to represent the condition of the machine. These representations are based on the single-unit maintenance policies adopted. The most common policies are age-based and failure-limit policies. The age-based policy uses the effective age(s) of the components which could be discrete or continuous state spaces to represent the machine states. The failure limit policies use the degradation paths of the machines represented by a statistical distribution for the state space representation.

2. Degradation levels: It was observed in 21% of the publications that in order to avoid making assumptions about the degradation model and its respective parameters when data is not available, the Markov discrete state degradation model is commonly used to represent the machine or asset degradation level, for instance, [64] used four degradation levels {0, 1,2,3} as the state representation. Level 0 is referred to as the best functional state and level 3 is the most degraded state. In [94], the pump states were defined to be between any of the four levels, {level 1 = No degradation, level 2 = Moderately degraded, level 3 = Severely degraded, and level 4 = Failed state}.

3. Remaining useful life: The third way of representing the machine states is to use the remaining useful life estimations of the machines/equipment. As previously

stated, prognostic information is pivotal to system health management, it gives explainable information about the decisions the RL agent is making and it is a preferred option for decision support.

Only 17% of all the reviewed publications modeled the machine degradation path from actual data while the other 83% of the publications made assumptions about the degradation paths of the machines based on the state representations they used. Figure 1.2 shows the percentage of publications and the state representation methods they used. 53% of the publications assumed that the degradation path of the

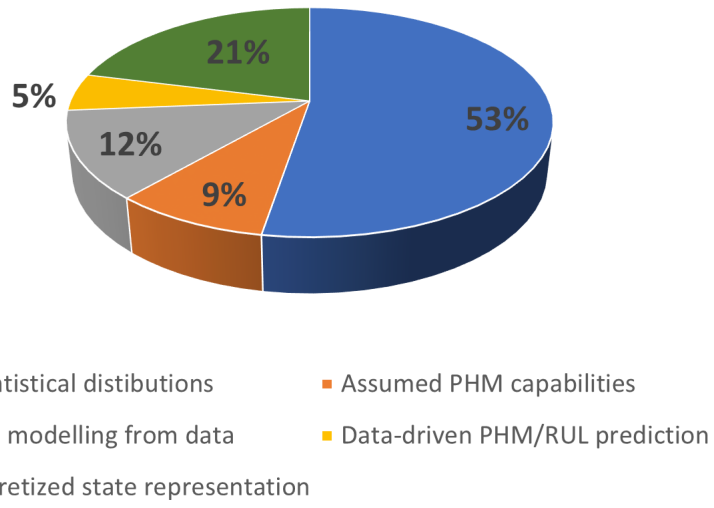


Figure 1.2: The percentages of machine state representation methods in the reviewed publications

machine follows a statistical distribution of their choice, the parameters of these statistical distributions were also assumed. The data used to train the RL agents were then drawn from these distributions. The most common statistical distributions used in the literature are the Gamma, Weibull, Poisson, and exponential distributions. These assumptions are however not practical for real-world applications, finding the statistical distribution that accurately represents the degradation path of a machine is a non-trivial task and they are prone to errors due to the strict assumptions made about the data. These errors will also affect the decisions of the RL agent. 21% of the publications used the Markov discrete states as degradation levels under the assumption that the degradation levels of the machines are known and 9% of the publications assumed the availability of prognostics and health management capabilities. 12% modeled the degradation path or levels from actual data and the other 5% made

up of only three of the publications, developed a data-driven RUL prediction model. This gap in the present research motivated this work. This work aims at using machine learning to develop efficient RUL prediction models for better machine state representations.

Scope

Preventive maintenance actions typically involve routine inspection, repairs, and replacements. The remaining useful life refers to the estimate of the remaining life (depending on a pre-defined resolution) that a machine or asset can perform its intended purpose before warranting replacement.

The RUL prediction problem is modeled as the time-to-failure prediction task for machines. Interventions such as inspection and repairs are not considered in this work, the degradation or run-to-failure data from a fleet of engines of the same type and working under similar conditions are used.

1.3 Objectives and Contributions

The objectives of this work are:

1. To categorize the two main machine-learning-based RUL prediction methods in the literature, using a systematic and sequential approach through experimentation. This includes investigating the reason for these models' performances and inadequacies and using these insights to make improvements.

2. To develop an improved similarity-based RUL prediction model through optimization of the developmental stages using a bottom-up approach which involves researching and categorizing existing methods, identifying the commonly used methods in the literature, and proposing improvements validated through experimentation.

3. And ultimately, to develop a novel model that extends the application of transformer models for more efficient time series data representation learning and prediction for an RUL prediction use case.

Contributions

1. An improved similarity-based RUL prediction model with a novel RUL fusion method (the softmax-weighted sum RUL fusion method) is proposed. This

method was evaluated on the validation dataset and the results were compared with the results from other most commonly used RUL fusion methods in the literature. The proposed method resulted in an improved performance. Also, drawing inspiration from [9], a novel supervised machine learning-based SBM, the Transformer-Assisted Similarity-based Model is proposed and developed in this work.

2. A novel direct approximation model, the Encoder-Transformer model based on the self-attention mechanism, is proposed and developed. This model leverages Large Language Model (LLM) concepts for more efficient time series data representation learning and prediction of remaining useful life. This model was compared to other state-of-the-art models in the literature and it exceeds all the models in the literature at the time this work was conducted. This makes this model the current best model on the benchmark dataset in the literature.

3. Finally, the insights drawn from these experiments are used to propose areas of improvement for existing similarity-based and direct approximation RUL prediction models. It also points out the directions of relevant future research.

1.4 Organizational Structure / Workflow

In the literature, the two main ML methods for RUL prediction are similarity-based and direct approximation methods. This work investigates these two methods to find common ideas and concepts, identify methodological problems through experimentation, and use the insights drawn to propose novel and improved methods.
For clarity, Figure 1.3 shows the workflow of the implementation stages of this work. It shows all the main processes involved in the data pre-processing, similarity-based model development, and the direct approximation method of RUL prediction performed in this work.

Chapter 1 contains the background, motivation, and contributions of this dissertation followed by an overview of the structure of the document itself.

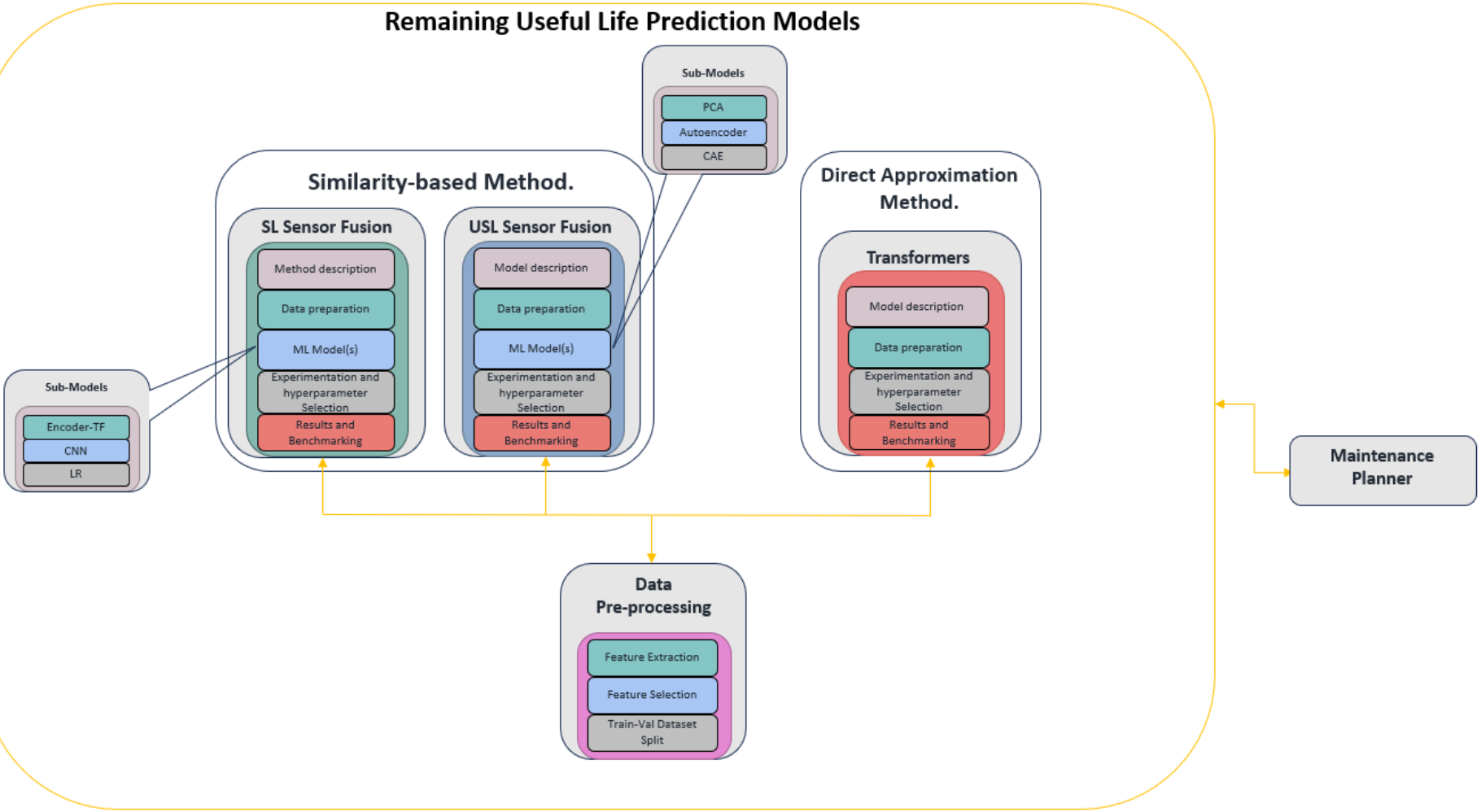
Chapter 2 contains the literature review, a statement of the claims that are proved by this dissertation followed by the proposed methods. At the end of Chapter 2, the case study, data pre-processing, and evaluation metrics for this work are presented.

Chapter 3 focuses on the similarity-based model. It describes in detail the problem which is to be tackled and its context. It presents the research question, the workflow of the chapter for better comprehension, the approach taken, the algorithms, the experiments conducted, and the experimental results. At the end of this chapter, the proposed methods are implemented. Finally, the main points about the similarity-based model are discussed.

Chapter 4 is just like Chapter 3 but it focuses on the direct approximation method. It describes in detail the problem and its context, it presents the algorithms, the experiments conducted, and the experimental results. At the end of each chapter, the proposed methods are implemented.

Chapter 5 presents the results of all the proposed models, it includes the evaluation of the results on the reference data presented above and the comparisons with other state-of-the-art models in the literature.

Chapter 6 contains the conclusion, a restatement of the claims, and the areas of future work.


Remaining Useful Life Prediction Models
Sub-Models
PCA
Autoencoder
CAE
Similarity-based Method.
SL Sensor Fusion
Method description
Data preparation
ML Model(s)
Experimentation and hyperparameter Selection
Results and Benchmarking
USL Sensor Fusion
Model description
Data preparation
ML Model(s)
Experimentation and hyperparameter Selection
Results and Benchmarking
Direct Approximation Method.
Transformers
Model description
Data preparation
Experimentation and hyperparameter Selection
Results and Benchmarking
Sub-Models
Encoder-TF
CNN
LR
Maintenance Planner
Data Pre-processing
Feature Extraction
Feature Selection
Train-Val Dataset Split

Chapter 2

Literature Review, Proposed Methods, and Case Study

2.1 Literature Review

In the presence of large amounts of data, predictive maintenance with machine learning models holds the capacity to result in significant reductions in costs. There are two main ML-based approaches for remaining useful life prediction in the literature. These approaches are the similarity-based and direct approximation methods. Similarity-based models (SBM) use the similarity in the degradation profile of the training data; and unique run-to-intervention instances from similar components/machines under similar operating conditions (ensemble members) to estimate the RUL of the test component. The two main stages of development of an SBM are the health index (HI) construction stage and the similarity measurement and RUL fusion stage.

Similarity-based models gained popularity from the first Prognostics and Health Management competition held in 2008. The winners of the competition, [65] used a similarity-based model, and ever since researchers have continually found ways to improve the performance of SBMs. More research efforts have been dedicated to improving the health index construction stage of the similarity-based model. The main idea followed by the researchers is to use unsupervised learning approaches (autoencoder schemes), to convert the multi-dimensional sensor readings collected from historical run-to-failure instances to lower dimensions. These lower dimensional representations are then used to construct the one-dimensional health index (HI) values

that reflect the degradation levels of the machine at any given time. The authors of [82], used a sequence-to-sequence-based model; recurrent neural network-based autoencoders were used to construct the health indices. The novelty of this work is in how the health index representations were derived. The authors proposed an approach that is capable of dealing with several challenges in data-driven RUL estimation which includes noisy sensor readings, missing data, and lack of information about the degradation path of the machines. Rather than using the lower dimensional embeddings to represent the health index, they leveraged the information of when the machines are in normal working conditions to compute the residuals between the low dimensional embeddings outputs of the RNN autoencoder models and the existing set of embeddings corresponding to normal behaviors. The magnitudes of these residuals are used as the health indices. Their proposed approach was compared with previously developed RNN-Autoencoders and they reported better results on the real-world pump datasets.

[82] is a direct extension of the [18] described above. The authors experimented with different bi-directional recurrent neural network-based autoencoder schemes to construct the health index. Banking on the idea that compared with standard unidirectional RNNs that use only past information at every hidden state, bi-directional RNNs (BiRNNs) can use the surrounding contexts by capturing the sequential information from the time series in a forward and backward manner. It is expected that these will improve the reconstruction precision of the RNN-based autoencoder and thereby lead to improved similarity-based model performance. The authors showed that using BiRNNs improves the predictive performance of the SBM compared to uni-direction RNNs by comparing their approach with other approaches that used uni-directional RNNs and reported improved performance.
Other extensions of this main concept of using unsupervised learning approaches for health index construction have been developed in the literature. [40] used an LSTM-Autoencoder for the HI construction, [84] developed an improved version of the BiRNN-Autoencoders while [79] used a combination of LSTM and convolutional autoencoders to learn better lower dimensional embeddings for time series data (LSTM-CNN-AE).

While most research efforts have been directed toward the HI construction stages, the RUL fusion stage of SBM development has received attention from only a few researchers. The authors of [57] proposed a self-adaptive RUL fusion method that does

not rely on expert knowledge and showed that this RUL fusion method's performance surpassed the one used in [65] which won the PHM08 competition. This RUL fusion method is also generalizable to other use cases. The authors of [37] also proposed a novel similarity measure computation method that combines the Euclidean and cosine similarity measures together in order to capture the local distance and spatial direction similarities.

Even though the similarity-based model won the PHM08 competition, direct approximation models came in as the first and second runner-ups. So extensive research has also been done in improving the performance of the direct approximation methods. Direct approximation methods refer to using machine learning models to directly predict the remaining useful life of a machine at any time during its life. The ML models serve as function approximators that learn the relationship between the input and the target which in this case is the RUL.
More generally, in the direct approximation methods research works, more attention has been paid to deep learning methods because of their complexity, expressiveness, and ability to learn non-linear and complex features in data. More particularly convolutional neural networks (CNNs) and recurrent neural networks have shown impressive results on time series data. In [56] and [30] deep CNNs were used for remaining useful life prediction, [95] and [63] used a combination of fully connected neural networks and LSTMs for RUL predictions. Particularly interesting is the RBM-LSTM-FNN RUL prediction model developed by [13], this model uses a semi-supervised approach to improve the prediction accuracy. The authors investigated the effect of unsupervised pre-training in RUL predictions utilizing a semi-supervised setup. They also used a meta-heuristic algorithm, the Genetic Algorithm (GA) approach to tune the large hyperparameter space.

In sequential data modeling, resulting from the need for very long sequences in the fields of Natural Language Processing (NLP) and Large Language Models (LLMs) development, a common challenge encountered is the problem of long-term dependencies. As the input data sequence lengths increase, all the problems associated with training very deep neural networks like vanishing or exploding gradients which leads to non-convergence come to play. Attention-based networks that use attention mechanisms were introduced to improve the performance of sequence-to-sequence models like RNNs. The attention mechanisms gave rise to the self-attention mechanism which

is the fundamental building block of a transformer network.
Since time series data can be modeled as sequential data, researchers have also adopted attention-based and transformer networks for RUL prediction and these models have shown tremendous capabilities. The authors of [89] proposed a bi-directional gated recurrent unit with a temporal self-attention mechanism for RUL prediction. In [41] a transformer encoder architecture with a gated convolutional unit was developed to extract the local features and the encoder transformer to extract the global features. [35] used an encoder-decoder transformer network with CNN-based channel attention for feature extraction.

2.2 Proposed Methods

In an effort to develop improved RUL prediction models, Four claims were made and validated in this thesis.

1. In this work, a claim that a better RUL fusion method will improve the performance of the similarity-based model was raised. This is a qualitative claim and it will be validated through experiments. A novel RUL fusion method named the softmax-weighted RUL fusion method was developed in this work.

2. For the unsupervised learning approach of similarity-based model development, a claim that the complexities of the unsupervised learning models do not directly translate to better-performing similarity-based models was raised. This is also a qualitative claim and it will be validated through experiment.
   As a proof of concept, an unsupervised learning approach based on Principal Component Analysis (PCA) was developed and combined with the novel softmax-weighted RUL fusion. The results were evaluated on the test set and compared to other unsupervised learning approaches in the literature and a highly competitive performance over its counterparts in the literature was achieved.

3. A claim that a supervised learning-based health index construction method can achieve as much performance as the unsupervised learning counterparts was also raised. A research question, Is the performance of the supervised learning-based HI construction method for an SBM dependent on how well the machine

learning model can approximate the HI values on unseen datasets? was used to validate this claim through experimentation.
Also, for a proof of concept, a novel supervised learning approach for health index construction was proposed, the Transformer-Assisted SBM. This model was developed, evaluated, and compared with other models in the literature.

4. Finally, an encoder-transformer architecture was developed for RUL prediction. The results of this model on the reference dataset were compared with other state-of-the-art models in the literature and the model resulted in outstanding performance.

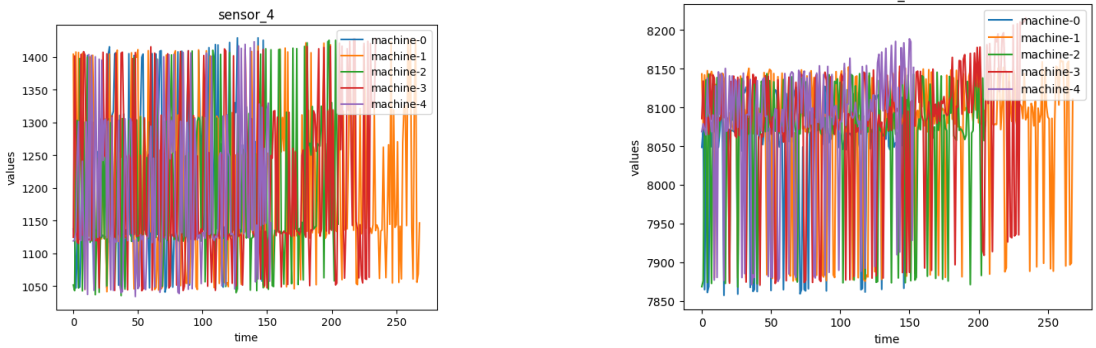
2.3 Case study and Data Pre-processing

Turbofan Engine Degradation Simulation datasets.
The turbofan engine degradation simulation dataset is one of the datasets on the NASA Prognostics Center of Excellence dataset Repository. This dataset was generated in 2008 in an effort to further research in the field of prognostics and health management by addressing the issue of the lack of common benchmark datasets that researchers can use to compare their approaches.
These datasets were generated using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS). The CMAPPS is a tool hosted on MATLAB and Simulink for simulating realistic large commercial turbofan engines. Four different sets {FD001, FD002, FD003, FD004} were simulated under different combinations of operational conditions and fault modes. These datasets are the degradation data of an aircraft turbo engine and were provided as training and test datasets to the competitors in the first prognostics challenge competition at the International Conference on Prognostics and Health Management (PHM08). The challenge is still open for the researchers to develop and compare their efforts against the winners of the challenge in 2008.
In this work, the FD002 dataset is chosen as the use case. The dataset is multivariate, 26-dimensional data from a fleet of engines of the same type, working under similar conditions. There are three operational settings that have a substantial effect on engine performance. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition. The 26-dimensional data consists of the operational settings as the first three dimensions of the dataset,

21 sensor information, and the unit ID and time cycle information are contained in the remaining two dimensions. There are 260 train trajectories and 259 test trajectories in the FD002 dataset.

Feature Extraction

Due to the different operational settings and the sensor values calibrations, the degradation trends in the data are not obvious. Figure 2.1 are the plots of run-to-failure from 5 fleets of engines for sensors 4 and 14 chosen at random and there are no clear trends showing the degradation paths of the machines. Clustering and normalization techniques are used to show these trends. For every run-to-failure machine instance,

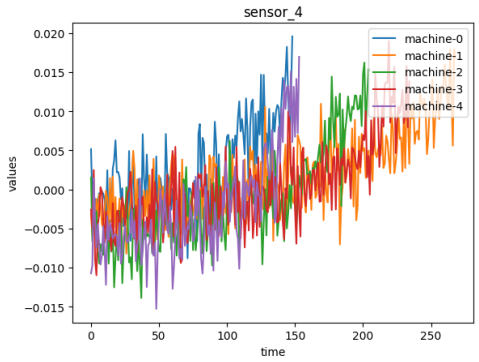


(a) Run-to-failure plots of first 5 ensemble members for sensor 4

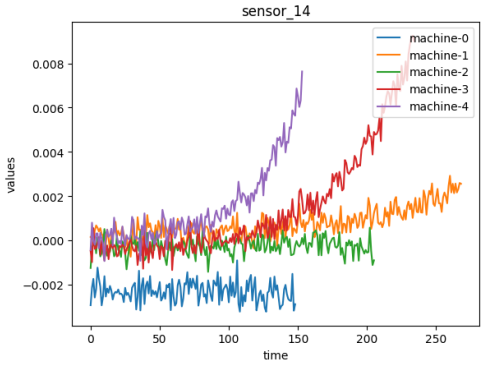
(b) Run-to-failure plots of first 5 ensemble members for sensor 14

Figure 2.1: Run-to-failure plots before feature extraction.

the operating conditions are clustered using the k-means clustering algorithm. The greedy k-means algorithm in the sci-kit-learn library was used to find the best number of clusters that resulted in the lowest cost and 6 clusters or modes of operations were found. The multi-dimensional sensor data for each time step are grouped by the clusters their corresponding operational settings belong to, these sensors are then normalized by clusters. The standardization feature scaling method was used, the mean and standard of each dimension in a cluster or group are used to normalize the sensor values belonging to that cluster by subtracting the mean and dividing it by the standard deviation. Figure 2.2 are the plots of run-to-failure from 5 fleets of engines for sensors 4 and 14 after clustering and normalization, now the degradation trends are more obvious. Sensor 4 has a linear upward trend and sensor 14 has an exponential upward trend.



(a) Run-to-failure plots of first 5 ensemble members for sensor 4



(b) Run-to-failure plots of first 5 ensemble members for sensor 14

Figure 2.2: Run-to-failure plots after feature extraction.

Feature Selection

After the working regimen clustering and normalization processes above, further data analysis was performed to select relevant sensors that give significant degradation information about the machines. It was observed that sensors 1,5,10,16,18 and 19 show no trends or irregular trends so they were not used. Also, correlation analysis shows that these sensors that show irregular trends are more correlated with the operating conditions than other sensors. They were uncorrelated with other sensors and highly correlated with themselves. More generally, these sensors also have very low correlations to the remaining useful life of the machines, unlike the other sensors that have obvious increasing or decreasing trends from run to failure.

Hence in this work, the data from the other 15 sensors were used to train the proposed model(s) and predict the remaining useful life. The sensors used in this work also follow the selected sensors in other studies in the literature [[30], [35], [74]].

Train-Validation Split

The 80-20 train-validation split was used. 80% of the dataset was used for training and the remaining 20% for cross-validation. After the hyper-parameters had been chosen, the entire training set was then used to train the final model(s), and the results were evaluated on the test dataset with 259 trajectories.

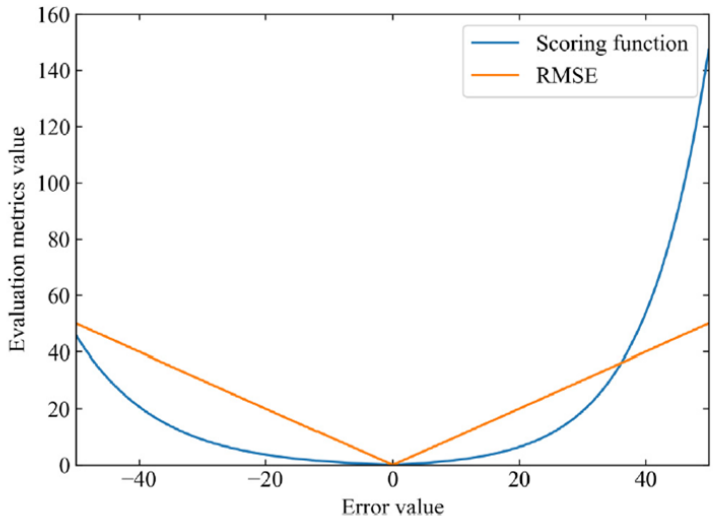


Figure 2.3: Graphical representation of the scoring function and RMSE

2.4 Evaluation Metrics

Two evaluation metrics are used in this work to evaluate the performance of the encoder transformer model against other state-of-the-art methods. The root mean square error (RMSE) and the scoring metric specifically developed for this dataset in the Prognostics and Health Management (PHM08) competition are used.
The RMSE can be expressed as follows:

$$\sqrt{\sum_{i=1}^{D}(y - y_i)^2} \tag{2.1}$$

where y_i and y are the predicted and true RUL values respectively and N represents the number of test samples.
The scoring function proposed in the PHM08 data competition penalizes late predictions more (i.e., the estimated RUL is greater than the true RUL). In the prognostic's context, the main idea is to avoid failures, it is more desirable to predict early rather than later because if the machine is predicted to fail later than when it fails, costs related to unplanned maintenance actions which are usually greater than planned maintenance costs will be incurred. But if the machine fails later than it was predicted to fail, the facility would already have maintenance plans in place. Therefore,

the scoring algorithm for this challenge was asymmetric around the true time of failure such that late predictions were more heavily penalized than early predictions. In either case, the penalty grows exponentially with increasing errors between the true and predicted RULs. The asymmetric preference is controlled by parameters $a1$ and $a2$ in the scoring function given in equation 2.2 and Figure 2.3 show the score and the RMSE functions as a function of the error. Compared to the score function, the RMSE score grows linearly with increasing errors between the true and predicted RULs.

$$s = \begin{cases} \sum_{i=1}^{n} e^{-(\frac{d}{a1})} - 1 & \text{for d} < 0 \\ \sum_{i=1}^{n} e^{(\frac{d}{a2})} - 1 & \text{for } d \geq 0 \end{cases} \tag{2.2}$$

where, s is the computed score,
n is the number of machine instances,
d = Estimated RUL - True RUL,
$a_1 = 10$, a_2, $= 13$.

Chapter 3

Similarity-Based Models

The two main stages of development of a similarity-based model (SBM) are the health index construction and the remaining useful life (RUL) fusion stages. Figure 3.1 shows the development framework for similarity-based models, it also highlights all the methods involved in every stage. This work aims at developing an improved similarity-based model for RUL prediction by building upon existing knowledge and making improvements.
To achieve this, the approach of optimizing every stage of the development process is taken. This involves identifying the well-established and commonly used methods in the literature, analyzing the performance and shortcomings of these methods through a well-defined experimentation process, and drawing insights from the analysis to make improvements at every stage of development.

In section 3.1, the main stages of the SBM development process are discussed while presenting the most common methods in the literature. The base model is also introduced and developed in this section. The base model was developed mainly for tuning and selecting the best methods and hyperparameters to optimize the second stage of the SBM development process. In this work, the second stage, the similarity measure and RUL fusion stage of the SBM process were chosen to be optimized first. This is because it is made up of many sub-methods and these sub-methods can be highly sensitive to the output of the first stage, the HI construction. For an efficient and unbiased experimentation process, a constant, reliable, and verifiable HI construction method was developed for the base model, these HI values were then used to select the best methods for the second stage through experimentation. Asides from using the best methods across all the sub-stages, a novel RUL fusion method

was also proposed, developed, and evaluated against other RUL fusion methods in the literature.
At the end of this section, a new workflow for a similarity-based model using the best methods was presented, developed and the results were analyzed. Finally, the insights drawn from these experiments that will inform our decisions in the next sections were presented.

The two ML-based methods of HI construction, the supervised and unsupervised learning methods were discussed and optimized in sections 3.2 and 3.3 respectively. After optimizing the second stage, sections 3.2 and 3.3 focus on optimizing the first stage of an SBM.
In section 3.2, the supervised learning method was optimized by formulating a research question following the works of [65] that won the PHM08 competition. In Section 3.2 the research question, is the performance of the unsupervised learning-based HI construction method for an SBM dependent on how well the machine learning model can approximate the HI values on unseen datasets or the expressiveness of the machine learning model? is answered.
Different ML models with palpable levels of complexities were introduced, developed, and experimented with to answer this research question. The supervised learning approach for HI construction has not been explored greatly in the literature, this work contributes to the literature by using well-defined experiments to investigate the effectiveness of this method of HI construction by identifying the drawbacks of this method, providing insights on how it can be improved, and comparing it with its unsupervised learning counterpart. At the end of this section, a novel and optimized supervised learning-based HI-constructed SBM was developed. The results were analyzed in the evaluation, analysis, and comparisons chapter of this work.

Section 3.3 follows the same pattern as section 3.2, a research question based on the trends that have been observed in the literature was raised and the performance of the SBMs based on the complexities of the unsupervised models was also investigated. At the end of this section, an optimized unsupervised learning-based HI-constructed SBM was also developed. The results were analyzed in Chapter 5.
The major highlight of this chapter is the bottom-up experimental approach taken to experiment and select best the methods. The benefits of this approach can be seen in the results achieved from the optimized similarity-based models developed when

compared to their other counterparts in the literature. The insights drawn from the experiments provide ways to improve existing methods and the research directions for the future. Another highlight of this work is that ten separate ML models were developed, trained, and tuned in this work to perform the experiments.

At the end of this section,

1. The methods that resulted in the best performances for the second stage of the SBM development process are highlighted.
2. A novel RUL fusion method **(softmax-weighted RUL fusion method)** is proposed and evaluated against other most commonly used RUL fusion methods in the literature.
3. The research question, Is the performance of the supervised learning-based HI construction method for an SBM dependent on how well the machine learning model can approximate the HI values on unseen datasets/ the expressiveness of the machine learning model? is answered.
4. Finally, drawing inspiration from [65], a novel supervised machine learning-based similarity-based model **(Transformer-Assisted SBM)** is developed.

3.1 General Concepts and the Base Model

The remaining useful life of a component refers to the usage time remaining before an intervention e.g., repair or replacement is required. Similarity-based models use the similarity in the degradation profile of the training data; and unique run-to-intervention instances from similar components/machines under similar operating conditions (ensemble members) to estimate the RUL of the test component.
The most used similarity-based model in literature is the residual similarity model. Generally, the historical data of run-to-intervention of a component from different unique instances operated under the same conditions fit with a model of identical structure. The degradation data from the test component is then used to compute the residuals between each component/ensemble member, and the test data. The magnitude of these residuals indicates how similar the test component is to the corresponding ensemble members.
There are so many factors involved in the development of SBM models. To give

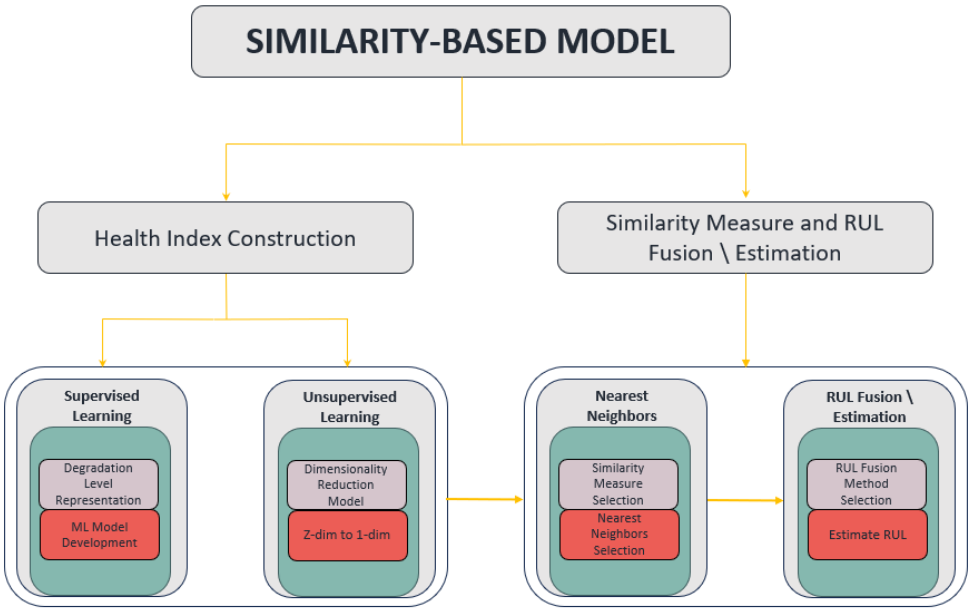


Figure 3.1: The framework of a similarity-based RUL prediction method

this section a smooth and logical flow; the underlying concepts in each stage of the development of a similarity-based model are captured in the underlying concepts subsection of section 3.1.

The main factor that differentiates one SBM model from the other is the health index construction method applied. There are two machine learning-based health index construction methods; supervised and unsupervised learning methods. In this work, a base model is developed. This model serves two main purposes. It serves as a benchmark to validate the performance of the SBM models developed in this work. Secondly, it is used for tuning and selecting the best methods and hyperparameters for the similarity measure and RUL fusion stages.

Since the distinguishing factor of SBM models is the HI construction method, it is reasonable to use a naïve base model to tune the second stage of the SBM development process. The health index construction method used in the base model is distinct; known RUL values from the validation set are used to construct the health index for the base model inspired by the work of [65]. Simply put, after the base model development process, the best methods and hyperparameters for the similarity measure and RUL fusion stages of an SBM model are tuned, chosen, and kept constant. Different health index construction methods were then developed, validated, and strategically

eliminated while drawing insights from the shortcomings of the previous methods to make improvements in subsequent methods.

3.1.1 A Similarity-based Model Underlying Concepts

The Health Index Construction Development Stage.
Health index construction is the process of finding meaningful low-dimensional (usually one-dimensional) representations of the multi-variate time-series data at any time, t. This low-dimensional representation, usually a single value is referred to as the health index or health indicator that describes the state of a component/machine at any time during the operation. These health indices or indicators can be used to monitor the condition of machines in real-time.
The HI should reflect the degree of degradation or current condition of the component/machine, and it is used as a basis for comparison between the degradation profiles of the train datasets and the test component. The health index construction stage can simply be referred to as a multi-dimensional sensor fusion process. Traditional sensor fusion techniques like the Kalman Filter can also be adopted but this work explores ML-based sensor fusion techniques, so, only ML-based methods are considered. ML-based HI construction methods as earlier mentioned are either supervised or unsupervised learning methods.

The Supervised Learning Method
The supervised learning method is inspired by [65]. In this method, the health index is generated by using a fixed set of intuitive rules and assigned to the training samples across all the ensemble members. In [65], the health index was a representation of the machine's condition based on how long it operated before failure. A linear model was used to represent the condition of the machines.
All the run-to-failure data was assumed to start with a good health condition. The health condition at the beginning is assigned a value of 1 and the health condition at failure is assigned a value of 0. The health condition was assumed to linearly degrade from 1 to 0 over time. This linear degradation can be referred to as a single-value and fused representation of the multi-dimensional sensor values over time.
This health index can be computed as follows:
For any given ensemble member i at the time t,

$$HI_i(t) = rul_i(t)/max\_rul_i \tag{3.1}$$

where, $HI_i(t)$ is the health index for ensemble unit (i) at time t, $rul_i(t)$ is the remaining useful life of ensemble unit (i) at time t, and max_rul_i is the total life span of ensemble unit (i).

While this method of health index construction might seem simple, it is very intuitive and applies well to the problem in consideration. Recall that the main premise is to find a single value representation of the degradation levels of the components/machines at any given time for all the ensemble members, this linear degradation model does this. It uses the survival duration of each machine to influence the health index generation process and shows the degradation degrees. If a machine has a short or long survival duration, it still passes through all these stages of failure and can be compared to another machine.

Figure 3.2 shows the health index values of some ensemble members constructed with the linear degradation model. It shows how the degradation rates change linearly from one ensemble member to another based on the survival duration of the machine or ensemble member. This health index construction method since it was introduced

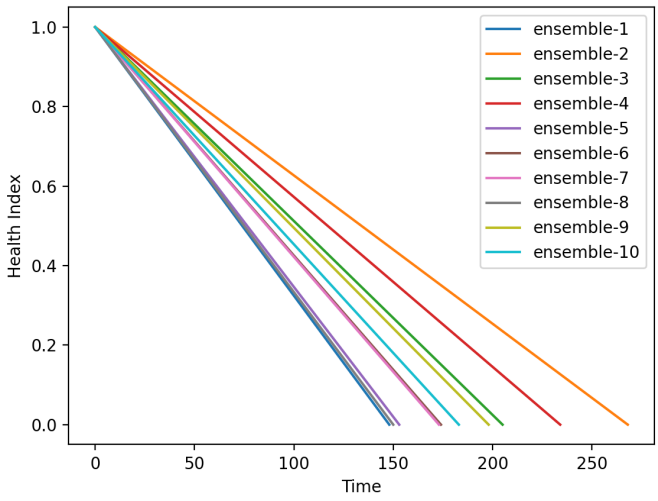


Figure 3.2: An image of the health condition of some ensemble members changing from 1 to 0 with varying degrading speeds.

by [8] has since been used, adapted, and improved continuously in literature. The shortcoming of this method however is that the health index computation method

requires the max RUL which is equivalent to the survival duration of the machine. Recall, however, that, this is what we are trying to find, if we know the survival duration of the machine then we do not need to develop an RUL estimation or prediction model for the test components/machines. To address the issue, a supervised learning ML model is used to learn the relationship between the multi-dimensional sensor values and the target which is the manually constructed HI with the linear degradation model.

The idea is to develop an ML model that takes the multi-dimensional sensor values from the train datasets as inputs, use the constructed health indicators (which can be easily generated for the training datasets because the survival duration is known) as the targets, and learn to predict the health index on unseen datasets (test data). Subsequently, the health indices that will be used for the similarity-based model will not be the ones constructed from the linear degradation model, instead, the trained ML model will be used to predict the health index on both the train, validation, and test datasets. This is why this method of health index construction is referred to as the supervised learning method or health index regression prediction approach.

Generally, in literature, unsupervised learning approaches have been a preferred method of sensor fusion because assumptions about the degradation path of the machine need not be made. This work however explores the capabilities of the supervised learning approaches and compares how well the unsupervised learning methods perform over the supervised learning methods.

Model-specific details for the supervised learning models developed in this work are discussed further in the corresponding sub-sections in 3.2 To summarise, the linear degradation model is used to generate the targets for the ML models, and the ML models are used to generate the health indices that will be used in the next stages of the similarity-based model development.

The Unsupervised Learning Method

This is the process of mapping the data features to lower-dimensional embeddings that still preserve enough information about the degradation profiles of the components/machines. This lower-dimensional representation serves as the health index.

Unsupervised traditional ML and deep learning dimensionality reduction models like Principal Component Analysis (PCA), and deep Autoencoders (AE) can be used to map the sensor values to a latent space of lower dimensions. In the literature, different variants of autoencoders like recurrent neural networks autoencoders have been

used to learn better low-dimensional representations.
It was observed that researchers try to find more complex representations of the health conditions of the machines by using deeper neural network autoencoders with the claim that it improves the performance of the similarity-based models. This works investigates this claim by developing a good basis for comparison and observing the performance of the similarity-based models as the expressiveness of the unsupervised learning models increases.
Model-specific details for the unsupervised learning models developed in this work are discussed further in the corresponding sub-sections in 3.3.

Health Index Processing Details

ML models are used to predict the health index through regression or dimensionality reduction techniques. The ML model predictions can have very large fluctuations due to noise; a filtering process can be performed on the predicted health index before they are used. Also, following the works in [65], researchers perform curve-fitting on the predicted training data health indices to ensure that all the ensemble members have similar degradation models. This was used to smoothen the approximated health indices for all run-to-failure instances and help generate less noisy or biased residuals between the test and training ensemble members.
Some researchers also normalize the health index values after the curve-fitting and filtering process to values between 0 and 1 to get good residual values. The effectiveness of the curve fitting and filtering processes on the predicted health indices is accessed with the base model.

3.1.2 The Similarity Measure and RUL Fusion Stages

The health indices generated in the first stage of the SBM development process are passed on to the second stage. As severally stated, similarity-based models try to find the ensemble members with similar degradation profiles to the test component's degradation profile and use the RUL values of the corresponding most similar ensemble members to estimate the RUL of the test component at any given time. This method raises three questions which are:

1. How do we select the ensemble members that are most similar – Measures of similarity?

2. How many of the most similar ensemble members do we use for RUL estimation – Number of nearest neighbors or candidate selection methods?

3. Finally, how do we estimate the RUL for the test component from the RULs of the nearest neighbors at any given time – RUL fusion methods?

The above questions are discussed in this section and answered with the base model. Many methods have been used in the literature to answer the questions above. There are no rules in the literature that specifies the use of one method over the other. Usually, researchers follow an empirical process to select their preferred methods. Another contribution of this work is using a sequential elimination process alongside a verifiable correct base model to experiment with the most common methods and select the methods that result in the best performances.

The Measure of Similarity

Measures of similarity provide a numerical value that shows the strength of associations between variables. In this work, it is a way of identifying the ensemble members that have similar degradation profiles to the test component/machine. A residual similarity-based model is used, residual-similarity-based models use the distances between variables as a measure of similarity or dissimilarity.

A residual function is used to determine the cumulative one-step distances between the health index of each ensemble member x and the health index of the test component y over the life duration of the test component. The magnitudes of these distances indicate how similar the test component is to the corresponding ensemble member, the smaller the distance, the higher the similarity.

Many measures of similarity have been used in the literature, this works collects a list of the most used distance-based measures of similarity for SBM. The base model is used to investigate the performance of these measures and choose the best measure for this application. The most used measures of similarity are Euclidean distance (L2 Norm), Manhattan distance (L1 Norm), and cosine distance.

The Euclidean distance between two points x and y is computed as:

$$Distance(x, y) = sqrt(\sum_{i=1}^{D}(x_i - y_i)^2) \tag{3.2}$$

The Manhattan distance is computed as:

$$Distance(x, y) = \sum_{i=1}^{D} |x_i - y_i| \tag{3.3}$$

And the cosine distance is computed as:

$$Distance(x, y) = 1 - \frac{x.y}{|x|.|q|} = 1 - \frac{\sum_{i=1}^{D}(x_i.y_i)}{\sqrt{\sum_{i=1}^{D} x_i^2}.\sqrt{\sum_{i=1}^{D} y_i^2}} \tag{3.4}$$

The Nearest Neighbours or Candidate Selection

The most similar ensemble members can be referred to as the nearest neighbors or candidates. All the RULs of the ensemble members are sorted in ascending order by the distance scores estimated. Three main methods have been used in the literature to select the nearest neighbors.

1. The first method requires choosing a constant, k number of nearest neighbors. If k is 10, the first 10 ensemble members (sorted by the distance scores from the smallest to the largest scores) are selected.

2. The second method inspired by [65] is the use of a constraint function to select a cut-off distance score. The cut-off distance score in [65] was set to a 25% increase of the smallest score using constraint $D_i \leq 1.25D_i$ where D_i is the smallest score. This threshold or cut-off score can vary from one dataset or machine to another.

3. The last method inspired by [57] uses all the ensemble members. In [57], weights were assigned to each ensemble member based on their degree of similarity d_i, most similar members get very large weights and dissimilar members are assigned low weights. In this method, the candidate selection process is not separated from the RUL fusion process and the RULs of all the ensemble members are used to estimate the RUL of the test component. The similarity degrees or weights are computed as the inverse of the similarity scores for every ensemble member.

   In this work, the first case of using a softmax function over the similarity scores to estimate the similarity weights was also experimented with.

The RUL Fusion Method

The RUL fusion method as the name suggests are methods on how to fuse the RUL values of the selected ensemble members effectively to estimate the RUL for the test component.

A naïve approach is to use the central tendency measures such as the mean and median of the RUL values of the candidate member. For clarity, the RULs of the candidate members can be computed as the survival duration minus the length of the test component or the discrete time steps that the test component has been operated for.

A second method of sensor fusion is to use a weighted sum. Rather than using the ordinary mean where all the candidate members are given the same weight, a weight vector is generated. The weight vector contains values that represent the similarity degrees of the candidate members with the test component.

The similarity degrees are values between 0 and 1 computed by getting the inverse of the distance scores. In [57] a generalizable RUL fusion method was proposed. It can be computed as shown in equation 3.5 below:

$$RUL = \sum_{i=1} \overline{w_i}.RUL_i, \sum_{i=1} \overline{w_i} = 1, \overline{w_i} = \frac{w_i}{\sum_k w_k} \tag{3.5}$$

where w_i and w_k are the respective weights estimated by the inverse of the similarity scores of d_i and d_k. The weights $\overline{w_i}$ represent the percentage of similarity between a testing unit j with a unit i ensemble member. The sum of weights for all considered units i is equal to 1 (i.e. 100%). It means that a unit with a high degree of similarity will have a weight value near 1 and a low similarity will have a weight close to 0.

The last method of RUL fusion influenced by [65] is application-specific and rule-based. This method requires expert knowledge, and an in-depth understanding of the data through data analysis. It is also very sensitive to changes in the dataset.

[57] showed that their generalizable RUL fusion method resulted in a better performance than the rule-based RUL method used in [65] which won the PHM08 challenge in 2008. So, in this work, the rule-based fusion method was not used in the experiment.

Proposed RUL Fusion Method: Softmax-Weighted Sum

The special characteristics of the generalizable RUL fusion method introduced in [57] is, the weights are similarity probabilities with values between 0 and 1 and the sum of

all the weights equals 1 or 100%. These characteristics are consistent with attributes of the weight vectors generated by a softmax function when applied to real-valued vectors. Drawing inspiration from the work in [57], a novel RUL fusion method was introduced where the softmax function is used to generate the similarity probabilities. Equations 3.6 and 3.7 below show the proposed RUL fusion method.

$$RUL = \sum_{i=1} \overline{w_i}.RUL_i, \tag{3.6}$$

$$\sum_{i=1} \overline{w_i} = 1, \overline{w_i} = \sigma(d_i) = \frac{e^{d_i}}{\sum_{i=1}^{k} e^{d_i}} \quad for\ i = 1, 2, \ldots, k \tag{3.7}$$

where d_i is the respective residuals or similarity scores estimated by the similarity measure for every ensemble member i, and k is the number of selected ensemble members. The weights $\overline{w_i}$ represent the percentage of similarity between a testing unit j with a unit i ensemble member. The sum of weights for all considered units i is equal to 1 (i.e. 100%). The results are evaluated on the validation dataset. The methods that resulted in the best RUL predictions were identified and used downstream for the final similarity-based models.

3.1.3 The Base Model

In previous sections, the base model has been briefly introduced. This section provides a detailed explanation of the significance of the base model, how the base model was developed, and the different experiments carried out to select optimal hyperparameters used in the final similarity-based models developed in this work.
The base model serves two major purposes, it serves as a benchmark for comparison with other similarity-based models, and it was used specifically to find the best methods and hyperparameters for the similarity measure and RUL fusion stage of the SBM development process.

Base Model for Experimentation

A benchmark model is a model that serves as a reference or standard against which other models are compared. As previously pointed out, the main difference between one SBM model and the other is in how the health indices are constructed. In the supervised learning method of HI construction, if the survival duration is known, we can use the linear degradation model to generate the health indices. However, due to

the unavailability of this information on the test component, we used ML models to construct the final HI.
While this information is unavailable for the test dataset, we have this information on the validation set. So, equation 3.1 was used to construct the base model's health index. Here, we use the actual values of the health indices rather than the approximate predictions from the ML models.

So we can confidently assume that, given that the linear HI construction method is a good representation of the degradation path from run to failure, and the optimal methods and hyperparameters for the second stage SBM development process have been tuned, the RUL prediction scores on the test set is not just a benchmark for other models but it is also the best result we can achieve with a supervised learning HI construction similarity-based model. After the exact HI construction process, the validation set was used to tune and select the best methods and hyperparameters for stage two, then the final base model was developed. Various experiments were carried out using cross-validation on the validation dataset. The experiments conducted were:

1. Different similarity measures: Euclidean distance, Manhattan distance, cosine distance, and similarity degrees.

2. Candidate selection methods.

3. RUL Fusion methods: Median and, weighted sum introduced in [57] and proposed weighted sum method in this work.

4. Finally, experimenting with the effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) and different HI approximation techniques.

Before discussing the experiments, the result analysis method on the validation data follows a slightly different approach. 50%, 70%, and 90% of sample validation data are used to predict the RUL. This method is used to evaluate the performance of the similarity-based models at different life spans of the machine.

Experimenting with different similarity measures

The most used similarity measures in literature are the Euclidean, Manhattan, and cosine distances. To make the similarity scores scale invariant to the number of nearest neighbors selected, the mean squared error (MSE), mean absolute error (MAE)

and mean cosine error which is simply the mean of the Euclidean, Manhattan, and cosine similarity error values respectively were used.
The evaluation metrics, RMSE, and score function that quantifies the errors between the predicted and true RUL values on the validation set are shown in Table 3.1 at 50,70, and 90% of the machine lifetimes. Across all the lifetime percentages,

Table 3.1: Experimenting with different similarity measures results table.

Similarity Measure	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
Euclidean	24.3	25.5	26.9	29.4	35.9	43.8
Manhattan	24.3	25.5	26.9	29.4	35.9	5.1
Cosine Similarity	27.9	29.1	30.5	33.0	39.5	43.8

the Euclidean and Manhattan similarity measures gave very similar results with the Manhattan distance measure resulting in a better score at the 90% mark. The cosine distance however resulted in higher errors on both evaluation metrics.
From this experiment, it can be concluded that the Euclidean and Manhattan similarity measures are good similarity measures and either of them can be chosen as simple go-to heuristics for measuring residual similarities for time-series data.

Experimenting with different candidate selection methods and the number of nearest neighbors hyperparameter tuning
To avoid using a biased evaluation of the SBM model when selecting the best candidate selection method between the k, constant nearest neighbors, and the cut-off distance score method, we need to first tune the k hyperparameter. In this work, different k values with a minimum value of 3 and a maximum value of 70 with a step size of 10 after $k = 10$ were used.

Table 3.2 shows the results of using different numbers of nearest neighbors. For the base model where the HI is constructed with a linear degradation model, $k = 50$ resulted in the lowest error between the true and estimated RULs.
To test the sensitivity of the number of nearest neighbors to the HI construction method, the k values were not only tuned for the base model but the simple linear regression (LR) model was used to approximate the HI values and experimented with. It can be seen from Table 3.3 that the number of nearest neighbors is highly sensitive to the HI construction model used.
The k values are very sensitive to the model type and complexity because the re-

Table 3.2: The number of nearest neighbor experimentation result table for the base model.

k	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
3	24.3	25.5	26.9	29.4	35.9	43.8
5	29.2	30.5	31.8	59.8	73.0	89.1
10	19.1	20.3	26.2	13.6	16.6	39.5
20	12.2	13.3	14.5	4.7	5.6	6.9
30	6.4	7.0	5.7	1.6	1.9	1.3
40	4.8	5.1	6.1	1.0	1.2	1.4
50	**4.5**	**4.7**	**5.7**	**1.0**	**1.1**	**1.3**
60	7.9	8.7	9.9	2.2	2.6	3.1
70	5.3	5.7	6.8	1.2	1.4	1.7

lationships between the inputs and the targets learned by each model are different. Even though the targets are the same, the hidden representations or feature representations and mappings learned differs from one model to another based on the model complexities and architecture. It is advisable to tune the number of nearest neighbors, the k parameter for any HI construction model used. There is no heuristic for how k should be selected, so it should be selected empirically.

Table 3.3: The number of nearest neighbor experimentation result table for the LR model.

k	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
3	6.67	28.36	61.52	1.69	53.86	6001.99
5	6.67	6.71	61.52	1.69	1.26	6001.99
10	3.95	8.75	19.10	0.75	1.78	13.96
20	5.26	3.00	9.57	1.00	0.52	2.94
30	3.55	4.12	5.00	0.52	0.91	1.09
40	5.00	6.71	2.57	1.12	1.72	0.50
50	**3.61**	**3.26**	**1.00**	**0.55**	**0.68**	**0.19**
60	6.08	2.15	0.71	1.48	0.31	0.08
70	3.61	4.74	1.90	0.57	1.08	0.36

A very interesting observation and insight can be drawn from the above sensitivity experiment. It can be observed that the LR HI constructed model had better results than the base models for the tuned k values on the validation set.

This improved performance shows that the assumption that was made in [65] about the degradation path of the machine being linear is not accurate. While the linear regression model was not able to learn the target, the linear degradation model did well, it however tried to learn some representations within its capacity.
This shows that using models that best represent the actual degradation path of the machine can improve the performance of SBM models. So rather than making assumptions blindly about the degradation paths of the machine and using a regression model to approximate the HI values, unsupervised learning models shine here. Since no assumptions need to be made, lower dimensional representations that retain enough information about the life cycle of the machines are preferred. This makes concrete the reason why most researchers in the literature prefer to use unsupervised learning methods for the health index construction.
This result also shows that the linear degradation model is not the best representation of the degradation path. It shows that the degradation path can be represented with better models or statistical distributions.

Table 3.4 shows the results of the different candidate selection methods using the median as the RUL fusion method. In Table 3.2, $k = 50$ gave the best results on the validation data. The cut-off score percentage needs not to be tuned because it was adapted from [65]. The k-nearest neighbor selection method performed better on the validation set and was therefore chosen as a preferred candidate selection method.

Table 3.4: Experimenting with the k-nearest neighbors and cut-off score candidate selection methods table of results.

Candidate Selection Methods	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
k-nearest neighbours	**4.5**	**4.7**	**5.7**	**1.00**	**1.10**	**1.30**
Cut-off score	17.33	18.51	19.81	10.49	12.72	15.52

Experimenting with different RUL fusion methods

The median, the generalizable weighted sum method proposed in [57] (shown in equation 3.5) and the weighted sum method with softmax proposed in this work (shown in equations 3.6 and 3.7) were experimented with. The Euclidean similarity measure was used for this experiment.
The k-nearest neighbors with median RUL fusion method, k-weighted sum RUL fusion method which refers to using a subset of the ensemble members with the weighted

sum methods (softmax or inverse) rather than using all the ensemble members as proposed in [57] was experimented with. All the ensemble members with the weighted sum methods (softmax and inverse) were also experimented with. Note, the inverse weighted sum (IWS) RUL fusion method refers to the RUL fusion method introduced in [57] and the softmax weighted sum (SWS) RUL fusion method is the RUL fusion method introduced in this work.

Table 3.5: Experimenting with different RUL fusion methods.

RUL Fusion Methods	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
k-nearest neighbours (k=50) + Median	4.53	4.74	5.70	0.96	1.08	1.31
k-nearest neighbours (k=50) + IWS	4.37	6.09	1.32	0.79	1.50	0.25
k-nearest neighbours (k=50) + SWS	**3.57**	**5.20**	**4.42**	**0.54**	**1.22**	**0.93**
All ensemble members (k=full) + IWS	4.60	4.85	4.19	0.99	1.11	0.87
All ensemble members (k=full) + SWS	**3.76**	**4.61**	**3.35**	**0.67**	**1.04**	**0.67**

From Table 3.5, we can observe that the softmax method of generating similarity weights or probabilities proposed in this work has better results and lower errors when used both with the k-nearest neighbors method and all ensemble members method. From this experiment, the k-nearest neighbor weighted sum with softmax and the softmax weighted sum with all ensemble members RUL fusion methods gave the best results. It can be observed that using the k-nearest neighbors weighted sum with softmax RUL fusion method gave better results at the early stages of the machine life while the all-members weighted sum with softmax method gave better results at the 70% and 90% marks.

With this experiment, this work was able to verify the RUL fusion method proposed in [57] and propose a better RUL fusion method using the k-nearest neighbors and weighted sum with softmax methods. The RUL fusion method is referred to as the **softmax-weighted sum RUL fusion method**, it can be used with the k-nearest neighbors method or with all ensemble members. This is another notable contribution of this work.

Experimenting with the effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) and different HI approximation techniques

This experiment differs a little from the other experiments discussed above. Generally, given that a similarity metric of choice has been chosen, the one-step errors

between the test data and each training data from time $t = 0$ up until $t =$ length of the test data are used to compute the similarity score or residuals. All the time steps before the current time step are used to compute the similarity score. This experiment investigates using some fractions of the time step rather than the full length. Under the notion that a closer time step to the current time step might give a better idea of the degradation path of the test components at the current time, t. It might be beneficial to tune the time lag or fraction of the time steps to look backward from the current time step.

It is possible that at different percentages of the component life, the window length that will result in better RUL estimations can vary.
For instance, at 90% of the lifetime of the validation data for a given ensemble member, it might not be a good idea to look further back into the past (i.e., look at the earlier stages HI values) to compute the residuals. Looking at the most recent degradation profiles where we can vividly see the degradation path of the machines might give better results.
To perform this experiment, the fraction of the length of the validation data at different life percentages is used. For instance, if the length at 50% of the lifetime of a unit in the validation set is 30, and a lifespan fraction (L) of 2 is experimented with, $20/2 = 10$ timesteps behind from the current timestep for which the RUL should be estimated will be used to compute the residuals. So the larger the lifespan fraction, L, the smaller the number of timesteps that will be used to compute the residuals.
This experiment was carried out with the base model to investigate this hypothesis. See Appendix B for the experimental results of using different lifespan fractions L at different percentages *pct* for different HI construction methods; supervised learning HI construction method with a feed-forward neural network and unsupervised learning method with PCA.

The effect of experimenting with different window lengths on the base model is close to nothing. From this experiment, it can be observed that the lifetime fraction parameter L is sensitive to the HI construction method used. By changing L and using the linear degradation model for HI construction, the effect is close to nothing or very minimal while the effect is more obvious when we use approximate HI's. So if we decide to use the approximate HI methods, this is a hyperparameter that should be tuned.

More specifically, the experiment carried out with approximate HI with feed-forward NN shows that using the full length gave the best results. And with PCA, $L = 1.5$ gave better results for 50%, $L = 5$ gave better results for 70%, and $L = 4$ for 90%. Generally, for a more robust model and to reduce the hyperparameter space, the effect of changing the lifespan fraction L can be neglected and/or compensated by using more sophisticated models.

3.1.4 Final Base Model Workflow and Results On The Validation dataset

In section 3.1, we have selected the best methods and hyperparameters for the second stage of the SBM development process. These methods and the exact health index construction method (linear degradation model) were used to develop the final base model.

Figure 3.3 shows the final workflow of the similarity-based model based on the best methods and hyperparameter values.

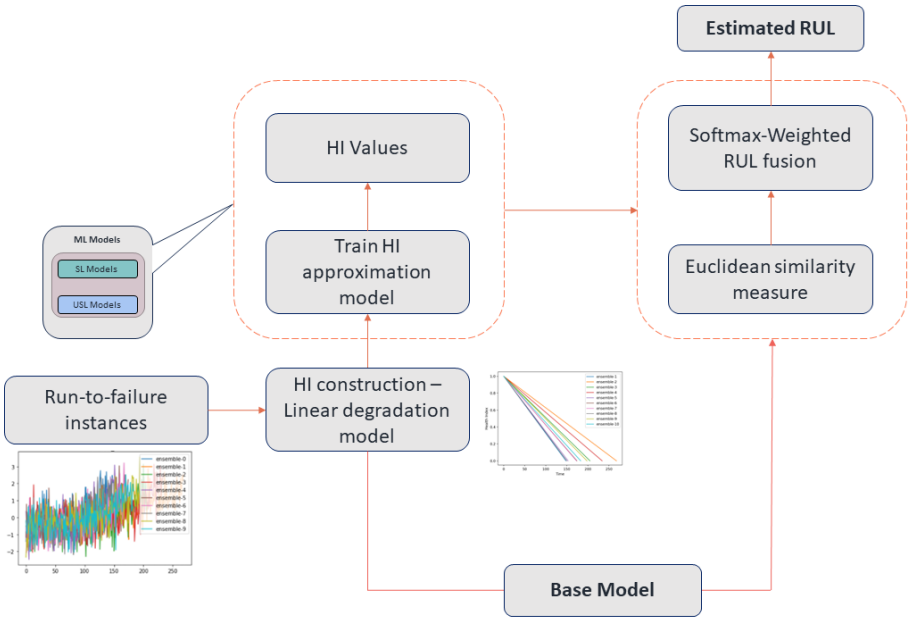


Figure 3.3: Proposed similarity-based model for RUL estimation.

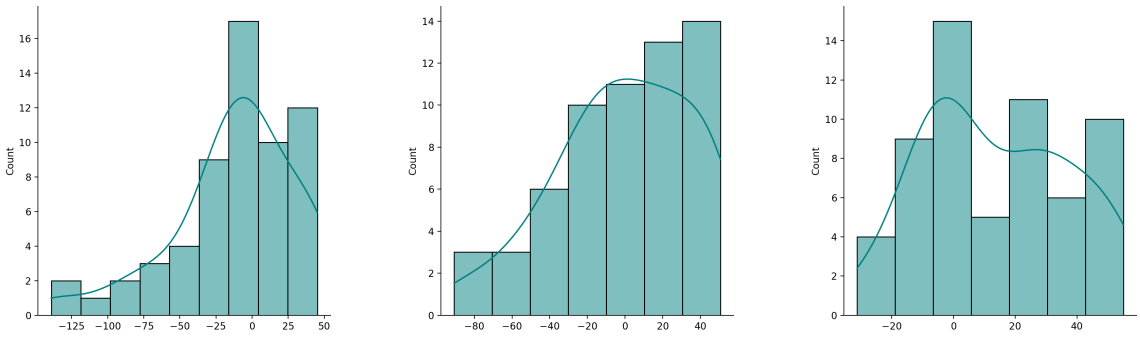
The Final Base Model Result Analysis

In this sub-section, some descriptive statistics were used to analyze the errors between

the true and predicted RULs. Table 3.6 shows the mean, median, and standard deviations of the errors between the true and estimated RULs on the validation dataset using the above SBM workflow for the base model.

Table 3.6: RUL errors at different life stages of the machine instances in the validation dataset for the base model.

SBM Model	**Error mean**			**Error median**			**Error std**		
	50%	70%	90%	50%	70%	90%	50%	70%	90%
Weighted k-nearest neighbor	-12.6	-0.46	13.8	-9.73	-0.98	10.1	42.5	36.3	23.4



(a) Probability distribution of the RUL errors at 50%.

(b) Probability distribution of RUL the errors at 70%.

(c) Probability distribution of the RUL errors at 90%.

Figure 3.4: RUL errors probability distributions at different lifetime percentages for the base model.

Figures 3.4 and 3.5 show the probability distributions and box plots of the errors at 50, 70, and 90% of the lifetime. At 50 and 70% of the machine life cycle, i.e., the machine is in an intermediate health stage, the error probability distributions are left-skewed which means there are probably outliers on the lower bound of the data distribution. These outliers are shown on the box plot. The mean of the RUL error distribution is shifted to the right and the variance is higher at 50 and 70% of the lifetimes due to the outliers.

These RUL error outliers were manually examined and it was observed that at the intermediate health stages of the machines when fewer data have been observed, there tends to be great variance in the RULs of the chosen nearest neighbors which leads to unreasonably long or short RUL estimations. At 90% of the machine lifetimes, when more data has been observed, the variance in the error distribution is less, and better RUL estimations are made.

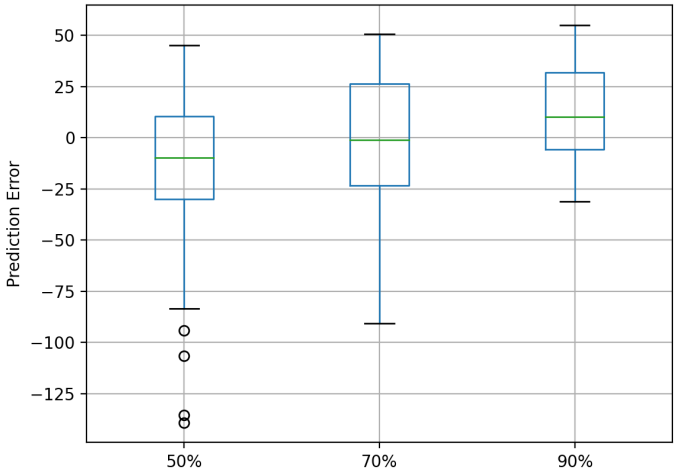


Figure 3.5: Box plots of RUL errors at different lifetime percentages.

This high variance in the RULs of the chosen nearest neighbors is because, for the turbo-engine machine examined in this problem, through data analysis, it was observed that on average, the machines operate in a healthy condition for about 125-time steps before it starts to degrade. So, at the early stages when the machine's degradation paths have not been formed yet, it is difficult to know which members of the ensemble would have similar degradation paths to the test machine leading to high RUL estimation errors.
This is the reason why the standard deviation of the error distributions is higher at the intermediate stages.

This behavior is common with mechanical components; they don't begin to degrade till after some usage period and this is considered when predicting the RULs. Also, we desire better predictions when the machines are close to failure and the similarity-based models make very good predictions at the end of the life of the machines. A common strategy used in literature to address this issue as seen in [[83], [66], [23]] is by limiting the maximum value of RUL estimation for any test instance to a constant RUL_max. In this case, when the machine is still in the early life stages, the RUL will be fairly large, so a constant threshold is chosen such that if the RUL is greater than the RUL_max, then the machine is considered to be in a good condition. The RUL_max used by other researchers for the CMAPSS dataset 2, FD002 is from

a set {160,165,170,175,180,185,190}. Note however that the maximal true RUL value of test engines in dataset no 2 is 200.

3.1.5 Insights and Main Points

Some insights were drawn from the above experiments which inform the next steps of the similarity-based model development.

1. The assumption that the machine degradation profile is linear is not good enough. The improved performance of the linear regression approximated HI model over the base model shows that the linear regression model learned a degradation path that more accurately represents the machine's degradation profile than the linear degradation model. The visual representations of the approximated HI values in section 5.1 also support this. The unsupervised learning representations learned from the data show that the machine does not start degrading immediately from the early stages.

2. The performance of the similarity-based model can be improved if the health indices used are a good representation of the degradation path.

3. The performance of the supervised learning HI construction similarity-based model might be improved if the ML model used to approximate the relationship between the sensor values and the linear degradation HI values is more complex. i.e., the performance of this model is based on how well the ML model can approximate the HI values on unseen datasets.
   Even though the linear degradation model is not an accurate representation of the degradation profile, the base model showed a very good performance on the test dataset so if a more expressive and complex model is used, and the model is able to learn this linear model, we should expect a result as good as that of the base model on the test dataset.

4. Curve fitting is a very important step. It helps to smoothen the encoded representations and reduce noise and fluctuations in the output of the ML models that can affect the computation of the residual. However, it has been shown to result in higher prediction errors from the experiments carried out in this work. Filtering also did not show much improvement in the SBM performance. In fact, in some cases, it caused more errors. Finally, normalizing the HI values after also prediction resulted in worse performance and it is not encouraged.

5. The number of neighbors k, for a k-nearest-neighbor's SBM is very sensitive to the HI approximation model type, architecture, and complexity. For every HI construction model used and dataset, it is advisable to tune the k parameter.

6. The SBM model benefits greatly from more ensemble members in the training set. The more time steps observed when computing the residuals, the better the result of the SBM model.

7. The model is greatly sensitive to outlier ensemble members.

3.2 Supervised Learning HI Construction Methods

In this section, the supervised learning HI construction methods were experimented with. Recall that a very important insight drawn from the base model is that the performance of the supervised learning-based HI construction method for an SBM might be based on how well the ML model can approximate the HI values on unseen datasets. Using a bottom-up approach through experimentation, the performance of the similarity-based models based on the complexity of the supervised machine learning model used to approximate the HI values was investigated.
This section answers the research question, Is the performance of the supervised learning-based HI construction method for an SBM dependent on how well the machine learning model can approximate the HI values on unseen datasets?

A strategic and sequential approach was taken to evaluate the performance of different ML models. Since this work investigates how the complexity of the machine learning model used for HI values approximation affects the performance of the similarity-based model, ML models with different levels of complexity were considered. The simple linear regression, four deep learning models, and an encoder transformer model for regression tasks were experimented with.
These models were chosen hierarchically. The linear regression model is the simplest ML model, and this was the model used in [65], the similarity-based model that won the PHM08 competition. The four deep learning models used were the feed-forward neural network, multi-tail feed-forward NN, recurrent neural networks (LSTM), and convolutional neural networks. For efficiency, the deep learning methods were evalu-

ated with the r2-score metric, the r2-score is a scale-invariant regression score function used to evaluate how well a model behaves on the dataset. The higher the r2-score, the better the performance of the model. Rather than developing separate similarity-based models for all the deep learning methods, the r2-score metric was used to choose the deep learning model that approximated the HI values the best and this ML model was used to develop a similarity-based model.

Amongst all the deep learning methods experimented with in this work, the convolutional neural network had the highest r2-score of 0.77 and it was chosen and used to develop the deep learning similarity-based model. Finally, an encoder transformer model was used to approximate the HI values. A transformer model is a highly expressive model that has shown amazing predictive capabilities in Large Language Models (LLM) applications, the transformer model used in this work is discussed in detail in Chapter 4.
The ML models were developed, trained, and tuned to achieve good health index predictions. This section introduces the machine learning methods and the best hyperparameters tuned during training. The r2-scores are presented in tables and are shown in Appendix C. Finally, the Transformer-Assisted Supervised Learning-based SBM is proposed and developed.
The r2-score function can be computed as:

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} \tag{3.8}$$

where, R^2 is the coefficient of determination, RSS is the sum of squares of residuals and TSS is the total sum of squares.

Linear Regression

Linear regression is the simplest ML model. It is a parametric model that assumes a linear relationship between the dependent(target/output) and independent variables(input/predictor). The relationship between the output variable Y and the predictor variable X can be described by a straight line.

$$Y = b0 + b1 * X \tag{3.9}$$

where, $b0$ is the intercept and $b1$ the slope.

The learning rate and regularization parameters were tuned using a grid search for values in the set {0.001, 0.003, 0.01, 0.03, 0.1, 0.3} and the RidgeCV class in the scikit-learn machine learning library. The values 0.01 and 0.03 for the learning rate and regularization parameters respectively resulted in the best r-score of 0.65.

Fully Connected Artificial Neural Network

A simple artificial neural network (ANN) is a fundamental building block for other deep learning networks which have become extremely powerful tools applied to a wide range of applications including image recognition and natural language translation. The artificial NN is simply a mathematical function that can take in a vector as input and output a vector. The effectiveness of ANNs comes from the presence of activation functions, the activation functions help the ANNs to find non-linear relationships in the data, and due to the network structure, they can learn complex, non-linear relationships during training on their own.

The fully connected ANN is an artificial neural network where all the nodes from the previous layers are connected to all the nodes at the next layer. Figure 3.6 shows the structure of a simple single-layer artificial neural network.

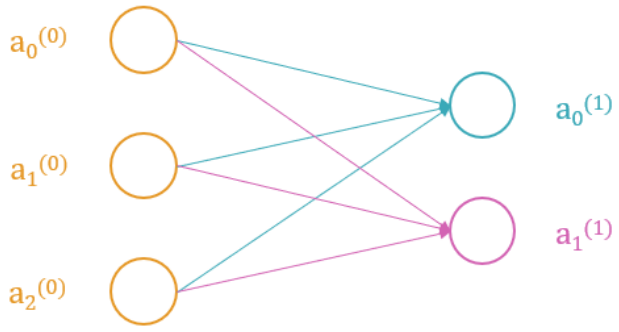


Figure 3.6: The structure of a simple single-layer artificial neural network.

The output of each node can be expressed as:

$$a_0^{(1)} = \sigma(w_0.a^{(0)} + b_0) \tag{3.10}$$

$$a_1^{(1)} = \sigma(w_1.a^{(0)} + b_1) \tag{3.11}$$

where,
$a^{(0)}$ is a vector of the activation units in layer 0,
$w^{(0)}$ is a vector of the weights of every activation unit from layer 0 to the first activation unit in layer 1, $a_0^{(1)}$,
$w^{(1)}$ is a vector of weights for every activation unit in layer 0 to the second activation unit in layer 1, $a_1^{(1)}$,
$a_0^{(1)}$ is the first activation unit in layer 1,
$a_1^{(1)}$ is the second activation unit in layer 1,
b_0 and b_1 are the bias values for the corresponding activation units to layer 1.
The equations 3.10 and 3.12 can be written in a vectorized format as:

$$a_1^{(1)} = \sigma(W_1.a^{(0)} + b_1) \tag{3.12}$$

where, $a_1^{(1)}$, $a^{(0)}$ and b_1 are vectors and W_1 is a matrix of weights.
The convention in deep learning is to stack multiple base layers over one another, so multiple hidden ANN layers and more neurons can be added to the simple ANN shown above to get a more expressive model that can be used to learn more complex relationships. A fully connected ANN was trained to predict the HI values. Table C.1 in Appendix C shows the tuned hyperparameters that performed well on the dataset and the r-score.

Multi-Tail ANN

The multi-tail ANN is a variant of the fully connected ANN. The outputs of multiple fully connected NN layers are concatenated together. Figure 3.7 shows the structure of the multi-tail ANN. A multi-tail ANN model was also built, trained, and tuned in this work to approximate the HI values. Table C.2 shows the best hyperparameter values for the dataset. For the multi-tail, RNN, and CNN models, sliding window lengths of {5,10,20} were tried and a window size of 10 worked best for the dataset.

Convolutional Neural Networks

Convolutional NNs are a type of deep-learning neural network that uses convolutions to find patterns in data. The idea behind convolution is to use kernels to extract features from input data, the concept of using kernels for an image or signal processing is not new. In the past, kernels of known weights e.g., the Gaussian Blur Filter were used to blur images or used on time series data for smoothing by reducing the level of

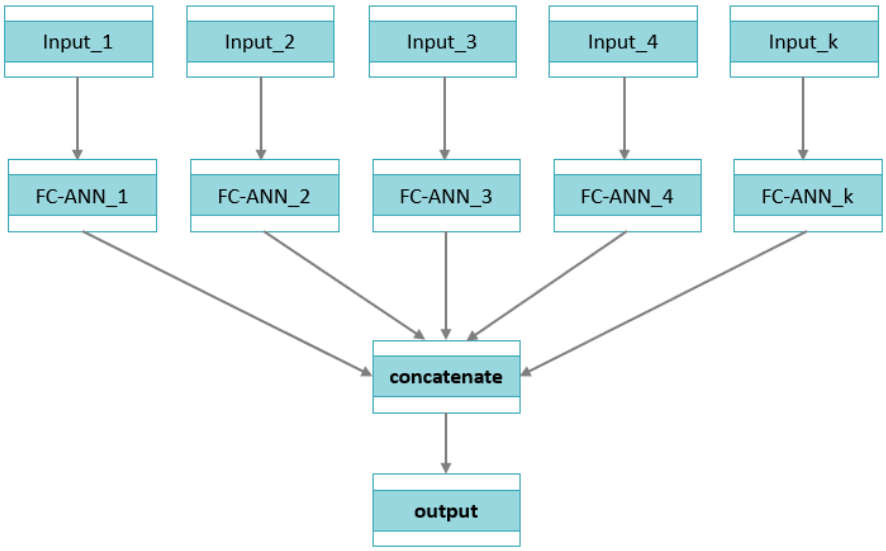


Figure 3.7: Structure of a multi-tail ANN.

noise in them. What makes CNNs special is the fact that they use learnable kernels which is a common characteristic of deep neural networks, the features are learned.

CNNs are composed of three main layers: the convolutional layer, the pooling layer, and the fully connected layer. The convolutional layer(s) parameters use learnable kernels. These kernels, usually two-dimensional for image recognition tasks or one-dimensional for time series data, glide over the entire depth of the input while calculating the scalar product for each kernel. Then an activation function is used to enhance the nonlinear expression of the convoluted features. The process is shown in equation 3.13, where x is the signal, f_k is the kernel filter, b_k is a bias, and σ is the activation function. After each convolution and pooling layer, the image or feature maps shrink and we lose spatial information, but this makes the network generalize better because while the size of the feature maps shrinks, generally the number of feature maps increases. Where the features are found matters less, and the network cares more about the fact that the features were found.
The most common pooling method in CNN is max-pooling max, which calculates the maximum value in a range w, as shown in equation 3.14. The fully connected layer consists of neurons directly connected to the neurons in the two adjacent layers, similar to a traditional ANN.

$$h_k = \sigma(x * f_k + b_k) \tag{3.13}$$

$$h_p k = max(h_k, w) \tag{3.14}$$

Figure 3.8 is a schematic representation of CNN with convolution, batch normalization, pooling, activation function, and fully connected layers.

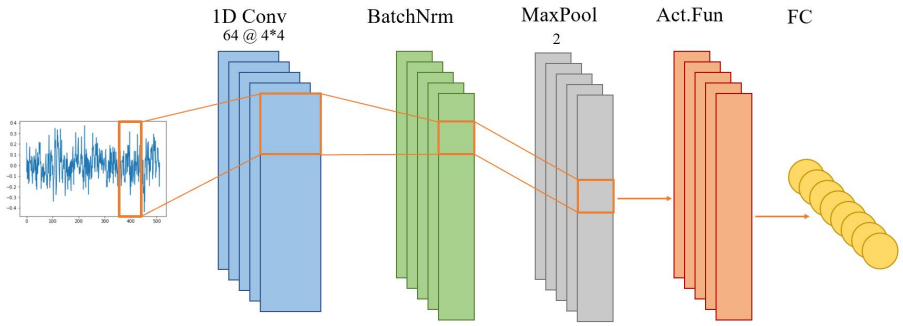


Figure 3.8: A schematic representation of the main CNN layers.

While CNNs have been primarily used to solve image-driven pattern recognition tasks, they have also shown great potential in their applications to time series data prediction. This is because we can use the autocorrelations in time series data to enhance the predictive abilities which is the fundamental building block of autoregressive (AR(p)) models (linear models), so autoregressive CNNs can be thought of as an extension of AR(p) models.
A CNN was trained to predict the HI values, and a sliding window length of 5 gave the best results. Table C.3 shows the best hyperparameters for the dataset and the r-score. A big improvement in the r-score was observed with CNN. In this work, the CNN model resulted in better HI predictions than other models including the LSTM model.

Recurrent Neural Networks

RNNs are one of the most used deep learning models for time series data modeling. The output of RNNs at a given time step t, is not only influenced by the input at that time step but also by the hidden state representation which contains a memory of all the states that have been visited before the current state. This is why RNNs

are said to have memory.

A common challenge encountered by the naïve RNNs, the Elman units is the problem of long-term dependencies. As the input data sequence lengths increase, the more deeply nested and further back in the sequence t, due to the compounding multiplicative effect when performing gradient descent, the more likely it is for the gradients to vanish. This was the issue faced by the simple RNN(s); long-short-term memory (LSTM(s)) that use gates to decide how much of the hidden representation from previous layers should be remembered in the current time step was developed to fix this issue.

LSTMs use three gates, input, forget, and output gates. The gates can be considered as switches that control how much information passes through them but mathematically, each gate can be considered as a single neuron with values between 0 and 1. An extra state known as the cell state (Ct) is introduced in LSTM. The forget gate $f(t)$ controls how much of the cell state should be forgotten, and the input gate $i(t)$ controls how much of the previous hidden state should be remembered in the current cell state. The output gate $o(t)$ controls how much of the previous cell state is to be forgotten.

$$f(t) = \sigma(W_{xf}^T.x_t + W_{hf}^T.h_{t-1} + b_f) \tag{3.15}$$

$$i(t) = \sigma(W_{xi}^T.x_t + W_{hi}^T.h_{t-1} + b_i) \tag{3.16}$$

$$o(t) = \sigma(W_{xo}^T.x_t + W_{ho}^T.h_{t-1} + b_o) \tag{3.17}$$

$$c(t) = f_t(\odot)c_{t-1} + i_t(\odot)f_c(W_{xc}^T.x_t + W_{hc}^T.h_{t-1} + b_c) \tag{3.18}$$

$$h_t = o_t(\odot)f_h(c_t) \tag{3.19}$$

where,

$f(t)$, $i(t)$ and $o(t)$ are neurons and are computed just like the way the feed-forward NN neurons or activation units are computed shown in equation 3.12.

Figure 3.9 is the basic structure of LSTM-RNN. A many-to-one LSTM model was trained to predict the HI values, Table C.4 shows the best hyperparameters for the dataset and the r-score. Contrary to the general intuition that LSTM works best for time series data, in this work, CNN showed a better performance than the LSTM model. The LSTM model did not result in significantly better performance than the fully connected ANN.

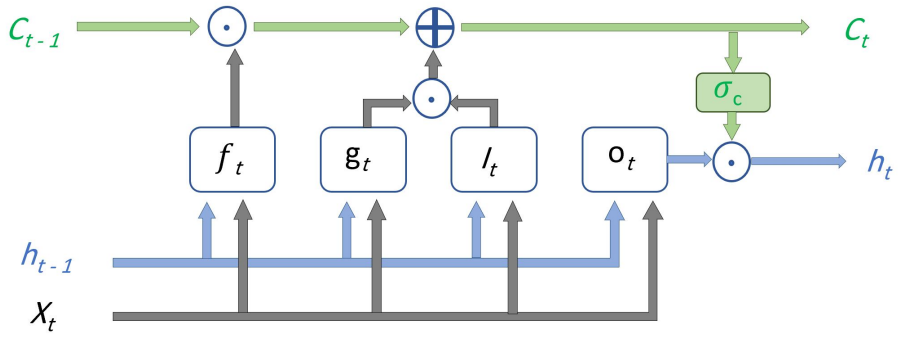


Figure 3.9: The basic structure of an LSTM network.

Supervised Learning-based HI-Constructed Similarity-based Models Development

The CNN model achieved the highest r-score on the dataset, so it was selected and used to approximate the HI values for the similarity-based model.

The Transformers Assisted Similarity-based Model

For the smooth flow of the work, the encoder-transformer model is briefly introduced here. As earlier mentioned this work introduces a novel encoder-transformer model for remaining-useful-life prediction and this model's performance surpassed every state-of-the-art model that it was benchmarked against in literature. Using our knowledge of the high performance of this model for time series data prediction, we developed a transformer-assisted similar-based model. Based on the claim that the better the approximation of the HI values, the better the performance of the SBM model, we also used the encoder-transformer model which is discussed in detail in Chapter 4 to approximate the HI values. An r2-score of 0.9 was achieved with the encoder-transformer model.

Finally, the linear regression, CNN, and transformer models were used to develop similarity-based models. Their performances were evaluated on the test dataset.

The results were evaluated using different RUL_max values, {160,170,180,190,200}. Table 3.7 below shows RMSE and score values for the different RUL_max values considered.

Does the complexity of the Supervised Learning HI construction models influence the performance of the similarity-based models?

From the results in Table 3.7, a huge improvement expressed by lower RMSE and

Table 3.7: Evaluation results on test dataset for supervised learning models-based SBM.

RUL_max	**SL Model**	**RMSE**	**SCORE**
RUL_max = 160			
	Linear regression	13.3	1445.1
	CNN	14.7	1489.3
	Encoder-transformer	13.4	1392.3
RUL_max = 170			
	Linear regression	16.7	3554.5
	CNN	15.5	1558.5
	Encoder-transformer	14.7	1532.1
RUL_max = 180			
	Linear regression	19.0	5060.7
	CNN	16.8	1778.2
	Encoder-transformer	16.3	1803.9
RUL_max = 190			
	Linear regression	19.2	5086.2
	CNN	17.5	1887.4
	Encoder-transformer	17.1	1892.2
RUL_max = 200			
	Linear regression	20.2	5278.2
	CNN	18.2	1959.7
	Encoder-transformer	17.8	1971.3

score values can be seen between the linear regression and the CNN model across all RUL_max values, this clearly shows that a better approximation of the health indices improves the performance of the similarity-based model. The improvement in the performance of the SBM model between the CNN and transformer model is not as conspicuous as that of the CNN and linear regression models because of the extreme simplicity of the linear regression model and because CNNs are very good deep learning models for time series data.

This decline in performance from a more expressive to a less expressive model grows exponentially as the RUL_max values increase. That is, as the model tries to predict the RULs of machines at earlier stages of the machine life cycle, due to the score function formulation which increases exponentially as the prediction errors grow and the inherent inability of similarity-based models to make very accurate RUL predictions at the early stages of the machine life. The RMSE score is a better metric to evaluate these results. Using the RMSE score, the transformer model generally performs the

best across all the RUL_max values. This experiment performed above shows that indeed a better approximation of the HI values by using more complex and expressive models improves the performance of supervised learning-similarity-based models.

3.3 Unsupervised Learning HI Construction Methods

The unsupervised learning sensor fusion methods for HI construction have been explored greatly in literature. Many unsupervised learning methods from linear models like Principal Component Analysis (PCA) to deep learning models like the Bi-directional Recurrent Neural Networks Autoencoders (Bi-RNN AE) have been explored greatly in literature. Researchers have used more sophisticated representation learning networks like DNN-based autoencoders with the aim of improving the performance of the similarity-based models.

Most publications have however compared these unsupervised learning models for SBM from one to the other with different RUL fusion methods. This isn't a good basis for comparison to assert the claim that using more sophisticated unsupervised learning models will automatically result in better similarity-based models. To assert this claim, this work investigates the performance of SBMs with a focus on the improvement rate of the SBM from simpler unsupervised learning models to more sophisticated models while keeping the downstream tasks (Stage 2 of the SBM development process) constant. This sets a good basis for comparison.
Three models with palpable differences in the degree of complexities were used to perform this experiment, Principal Component Analysis (PCA), Fully connected Autoencoder (AE), and Convolutional Autoencoder (CAE).
For clarity, this section aims at investigating the performance improvement of unsupervised learning method HI constructed - SB models as the unsupervised learning model complexities increase.

A critical detail about each model that will be mentioned in this section is how the lower dimensional representations from the unsupervised learning models are mapped to one-dimensional HI values and/or used directly to compute the similarity measures.

This detail varies from one model to the other due to the dimension and type of outputs each model returns.
More generally, unsupervised dimensionality reduction models are usually referred to as autoencoders and can be demonstrated in the Figure 3.10 below.

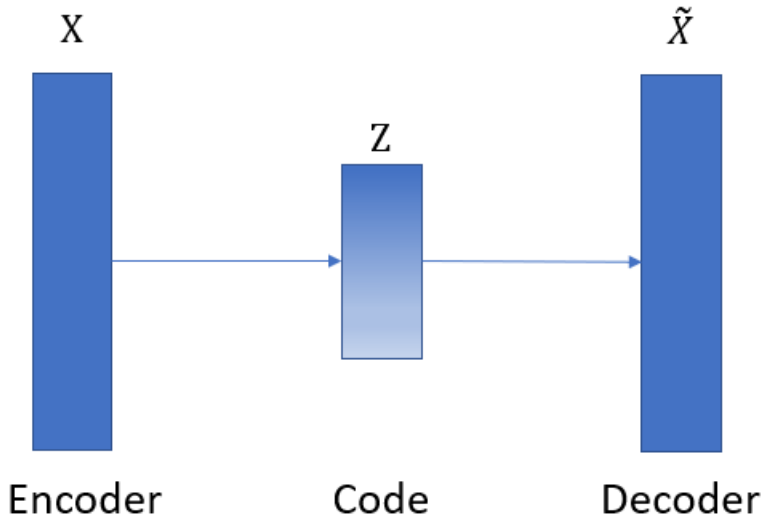


Figure 3.10: Autoencoders basic architecture.

An autoencoder encodes a data point X and tries to decode it to something similar. The mapping from the data to the code is called an encoder and the mapping from the code to the reconstructed data is called the decoder. The objective is to find the parameters such that the reconstruction error between input data X and the reconstructed data $\tilde{X}$ is minimized. The code, Z is the low-dimensional representation. If the mapping from the encoder to the decoder is linear mapping, then we get the PCA. So PCA can be thought of as a linear autoencoder. Deep neural network autoencoders simply have non-linear mappings between the encoder and the decoder.

Principal Component Analysis (PCA)

PCA is a dimensionality reduction method that uses a linear projection from higher-dimensional subspaces to lower-dimensional spaces while retaining as much variance or information in the data as possible. The fundamental concept of the PCA algorithm

is orthogonal projections. Figure 3.11 shows an illustration of orthogonal projections of vectors onto 1D subspaces.

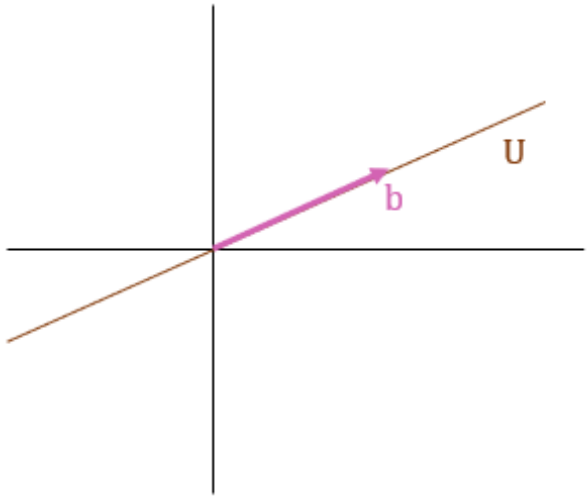


Figure 3.11: Orthogonal projection unto 1-D subspace.

Given a vector x in 2 dimensions and a 1-dimensional subspace u with a basis vector b all vectors in u can be represented as $\lambda * b$ for some λ. The vector in u that is closest to x can be found by an orthogonal projection of x unto u which is denoted by $\Lambda_u(x)$.
The important properties of projection are:

$$\Lambda_u(x) = \lambda * b \tag{3.20}$$

$$< b, \Lambda_u(x)) >= 0 \tag{3.21}$$

Combining equations 3.20 and 3.21 together, $\Lambda_u(x)$ can be computed as:

$$\Lambda_u(x) = \frac{bb^T}{||b||^2} * X, if||b|| = 1; \lambda_u(x) = b^T X \tag{3.22}$$

$$\Lambda_u(x) = bb^T X \tag{3.23}$$

In PCA, the objective is to find the basis vectors that exist in the m-dimensional lower subspace that minimizes the reconstruction error $||X - \Lambda_u(x)||$ between the n-dimensional data points and the orthogonal projection of point X unto subspace m.

J can be re-written as,

$$J = b_j^T S b_j N \tag{3.24}$$

Where S is $\frac{1}{N}\sum_{n=1} X_n X_n^T$ which is equivalent to the data covariance matrix.
The eigenvectors of the data covariance matrix are the basis vectors. The eigenvectors that belong to the smallest eigenvalues are the basis of the subspace to be ignored and the principal subspace is spanned by the eigenvectors belonging to the m largest eigenvalues of the data covariance matrix.
So, B is the eigenvectors belonging to the m largest eigenvalues of the data covariance matrix.

$$\Lambda_u(x) = \lambda * B = BB^T X \tag{3.25}$$

Where λ = code = $B^T X$ which is the lower dimensional representation that is used.

PCA Implementation Model Development Details

In this work, m was chosen to be the smallest value so that 80% of the variance in the data is retained. m = 3, so the multi-dimensional training datasets were projected onto a 3-dimensional subspace.
To avoid data spillage, the PCA algorithm was trained only on the training dataset then the transformations from X_i to Z_i on the training datasets were then mapped onto the validation and test datasets to reduce their dimensions.
PCA 3D to 1D Fusion: Finally, to convert the 3D code to 1D HI values, a linear combination weighted by the percentage of contribution of the chosen principal components and the 3D points was performed. Simply put, the dot product between the PCA elements (matrix) and the contribution (vector) was taken.

Fully-connected Autoencoder (FAE) Implementation and Model Development Details

The fully connected deep neural network architecture has already been discussed in detail in section 3.2. The major difference between a fully-connected NN and the autoencoder is in the target, the inputs are also the targets for autoencoders. Autoencoders are a type of neural network that is trained to replicate its input to the output. Deep Learning-based autoencoder architectures are usually symmetric in nature just as shown in Figure [encoder], the number of layers required to map the inputs to the code is also required to transform the code to the output which is a

replica of the input.

AE Code Fusion: The encoded representation (code) dimension chosen for this work is m = 3. This was chosen empirically using cross-validation and using the PCA dimension as a basis. The first dimension of the code was chosen as the HI values. Table C.5 below shows the FAE model hyperparameters.

Convolutional Autoencoder (CAE) implementation and model development details

The only difference between the AE and CAE is the base architecture chosen. In fact, the only difference between most deep learning autoencoder architectures is the base architecture. So, if the base model is an LSTM, then it is referred to as LSTM AE. 1D convolutions are used on the time series data with a constant window length of 5 which have been previously optimized. This autoencoder model leverages the good predictive abilities of convolutional neural networks on time series data.

CAE Code Fusion: The encoded representation (code) dimension chosen was also m = 3. The first feature map was chosen and a global max pooling layer was applied over the encoded depth to get a 1D representation of the HI values. Table C.5 shows the CAE model hyperparameters.

Unsupervised Learning HI-Constructed Similarity-based Models

The CAE model achieved the highest r-score on the true X values versus the reconstructed $\tilde{X}$ values. Just like in section 3.2, the results were evaluated using different RUL_max values, {160,170,180,190,200}. Table 3.7 below shows RMSE and score values for the different RUL_max values considered.

Does the complexity of the unsupervised learning HI construction models influence the performance of the similarity-based models?

From the results in Table 3.8, a huge decline expressed by higher RMSE and score values can be seen between the PCA and FAE models across all RUL_max values, with the PCA model outperforming the FAE model at every RUL_max value. This decline in performance grows exponentially as the RUL_max values increase.

Using the RMSE as a basis for evaluation, the PCA RMSE values are lower than that of the FAE and CAE models across all the RUL_max values except at RUL_max = 160. This experiment shows that the complexity of the unsupervised learning ML models does not directly translate to a better similarity-based model.

Table 3.8: Evaluation results on test dataset for unsupervised learning models-based SBM.

RUL_max	**SL Model**	**RMSE**	**SCORE**
RUL_max = 160			
	PCA	14.9	1193.1
	FAE	13.7	1317.7
	CAE	13.2	1272.3
RUL_max = 170			
	PCA	16.3	1338.7
	FAE	16.5	1850.5
	CAE	16.2	1392.3
RUL_max = 180			
	PCA	17.5	1460.5
	FAE	18.4	2088.4
	CAE	18.1	1950.3
RUL_max = 190			
	PCA	18.2	1530.7
	FAE	20.0	2781.13
	CAE	17.8	1562.3
RUL_max = 200			
	PCA	18.8	1556.3
	FAE	21.3	3026.2
	CAE	19.2	2560.3

Chapter 4

Direct Approximation Method

4.1 Transformers For Multi-variate Time-series Data Prediction

In this chapter, the direct approximation method implementation details, problem formulation, modeling, and experimental results are discussed.
In this work, direct approximation models refer to using ML models to directly predict the remaining useful life (RUL) of a machine at any time during its life, the ML models serve as function approximators that learn the relationship between the input and the target which in this case is the RUL. Since we have numerous amounts of data from run-to-failure instances from several machines, we can formulate this problem as a prediction task that is native to the supervised learning ML paradigm. Supervised learning is one of the three main paradigms of ML that leverages the abundance of past data and learns from it to perform two main tasks: regression, or classification. While the domain applications of ML may vary largely, ranging from predictive modeling and analytics to computer vision or generative AI applications; the fundamental tasks performed remain either a regression or classification task. Regression is simply used to predict continuous variables and classification is used to predict discrete variables.

The basic pipeline for ML tasks follows the schematic below, the pre-processed input data is fed to a model of choice and the model is trained to learn the relationship between the dependent (target) and independent variables (input) and use the learned model to make predictions of the target on previously unseen data points.

The RUL prediction problem can be modeled as a regression task by naively repre-

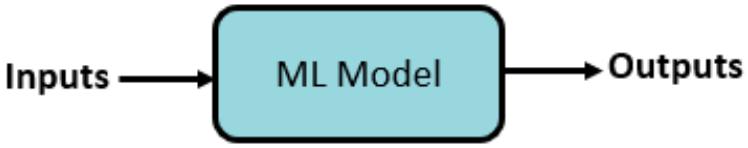


Figure 4.1: Basic machine learning pipeline.

senting the target as the actual time-to-failure values. That is, depending on how the successive data points are collected in time, given that we are at timestep t, if the machine will fail at timestep $t + 10$, then the target (RUL) at timestep t will be $t + 10 - t$.

This work focuses on the use of transformer networks for making predictions on multi-variate time series data. Before going into details about the architecture of transformer networks, it is expedient to highlight the characteristics of time series data, why they are so hard to predict, and why transformers work for them. Machine models can be divided into four main categories, Classical ML models, Ensemble models, Deep Learning models, and Attention and Transformer-based networks. Classical ML models such as linear and logistic regression models learn the linear relationship between the dependent and independent variables. Classical ML models are simpler models, so their applications are very limited. Ensemble models combine multiple weak or simpler models together to create a single strong model using two main techniques; bagging and boosting techniques. Deep learning models stack multiple layers of the base models of choice such as the fully connected feed-forward, convolution layers, or recurrent neural network layers and activation functions to generate models that can learn non-linear relationships between the inputs and the targets.

Most real-world applications of ML such as time-series data forecasting or image recognition require sophisticated models such as deep neural networks to make good predictions. The major thing to point out is that succeeding ML model categories were developed to improve the existing models. For instance, the ensemble models

were developed by combining multiple classical models like decision trees to form strong models that can learn more complex representations and deep learning models were also developed to improve existing ML models.

Time series data prediction and/or forecasting is one of the foremost domain applications of Predictive modeling. A variety of modeling approaches have been developed in the literature for univariate and multivariate time series data. Statistical methods such as auto-regressive models and their variates (ARIMA, VARMA, VARIMA, etc.) or deep, and non-deep learning ML methods such as TS-CHIEF [59] have been used in literature to make predictions. However, unlike in Computer Vision or Natural Language Processing (NLP) domains, deep learning models have yet to gain dominance or result in incomparable results compared to non-deep learning models. According to [86], non-deep learning methods such as TS-CHIEF [59], HIVE-COTE [34], and ROCKET [10] currently hold the record on time series regression and classification dataset benchmarks, closely matching or even outperforming sophisticated deep learning architectures like InceptionTime and ResNet. Time series data prediction is a non-trivial task and machine learning models might find it hard to learn to make predictions of time series data because of the following reasons:

1. Machine learning models are developed under the assumption that the data points are independent and identically distributed random variables, but time series data don't obey this assumption. Time series data is a sequence of measurements taken at successive equally spaced intervals over time and the data points or observations are not mutually independent, there is autocorrelation between the data points. Also, they are not identically distributed random variables, as the name implies time series data changes over time so distribution (mean and standard deviation) changes over time, they are non-stationary. While different methods such as differencing or detrending have been developed in the literature to make time series data more stationary, these underlying characteristics of time series data make it difficult for machine learning models to make very good predictions.

2. Other distinct characteristics particular to time series data are trends, seasonality, cycles, special events detection, and classification, or a combination of all these characteristics. Changes in the trends or seasonality patterns in the data make time series data very difficult to predict. These distinctive traits observed in univariate time series data even become more difficult to handle

in multi-variate time series data. The complex, non-linear relationship between the input variables and the target in time series data makes it difficult to predict.

3. Finally, unlike language, which is universal, and has led to the widespread success of Natural Language Processing (NLP) because of their ability to learn encoded representations of natural language through unsupervised learning approaches, time series data comes in different kinds and forms. This makes it difficult to generalize the encoded representation of a particular data/use case to another data/use case. The good news is, however, difficult time series modeling gets, researchers keep developing ways to make better predictions with the available tools and knowledge about the data.

   As we have shown above, time series data modeling and predictions are non-trivial tasks. Multi-dimensional time series modeling is even more difficult; more sophisticated models are required to make better predictions. With this understanding in mind, this work aims at achieving better predictions for multi-variate time series data by exploring how we can transfer the concepts of Large Language Models (LLM) to the time series domain. Since time series data can be modeled as sequential data, it is only reasonable to transfer the concept of how sequential data has been modeled in other domains like natural language processing to the time series data prediction and modeling domain. The base architecture for large language models is **transformers** and this work adopts it for better time-series data predictions and more efficient time-series data representation learning.

   The base model used in LLM models such as BERT and GPT models is **transformers**. Transformers are highly expressive models that are excellent for sequential data modeling, they are built upon the foundation of **attention** mechanisms. Attention is a resource allocation mechanism that mimics the brain by recognizing what or where to pay more attention to and assigning larger weights to them. Attention mechanisms were developed to solve the problem of long-term dependencies experienced in recurrent neural networks due to very long data sequences.

   Sequential data is data in which the ordering matters. Time series data are evenly spaced data points and can be modeled as sequential data with sequences taken at successive equally spaced points in time. Since time series data is sequential data, we can use sequential modeling approaches like transformers and

attention mechanisms to model it.

4.2 Attention-based Models and Transformer Architecture

Attention-based Models and Mechanism

Attention-based models are models that use attention mechanisms. The attention mechanism allows the base models, usually neural networks (ANN, CNN, or RNNs) to focus on the most important features of the input that produces better results at the output. The presence of this attention mechanism helps to improve the performance of the base models. Attention-based models have gained more traction in sequence-to-sequence models like recurrent neural networks (RNNs).

RNNs are one of the most used deep learning models for sequential data modeling. The output of RNNs at a given time step t, is not influenced only by the input at that time step but also by the hidden state representation which contains a memory of all the states that have been visited before the current state. This is why RNNs are said to have memory. A common challenge encountered by the naïve RNNs, the Elman units is the problem of long-term dependencies. As the input data sequence lengths increase, the more deeply nested and further back in the sequence a hidden unit is, and due to the compounding multiplicative effect when performing gradient descent and learning the weights, the more likely it is for the gradients to vanish. This was the issue faced by the simple RNN(s); long-short-term memory (LSTM(s)) and gated-recurrent neural networks (GRU(s)) that use gates to decide how much of the hidden representation from previous layers should be remembered in the current time step were developed to fix this issue. However, in most applications, the sequence lengths are required to be very long, and this brings out all the problems associated with training very deep neural networks like vanishing or exploding gradients which leads to non-convergence, so attention networks are used to improve the performance of RNNs.

Attention Mechanism

To focus on the most important parts of the input to get better outputs, context vectors ($C(t)$) which is a dot product between the hidden representation and learnable weights to all outputs at every time step in the sequence are used. They determine

which portions of the hidden representations are important to generate the output at any time instance, t. So, at every time step of the decoder, the context vector is different.

The attention mechanism was introduced by [4]. The computation of the attention mechanism is divided into three main steps, the computation of the alignment scores, the weights, and finally the context vector.

1. Compute alignment scores: The alignment score is a function of the hidden states of the encoder, h_j, and the decoder output at $t-1$, s_{t-1}. The alignment model can be implemented by a feed-forward neural network. The alignment score as shown in equation 4.1 can be generally computed with three methods shown in equation 4.2.

$$score(s_{t-1}, h_j) = s_{t,j} = \tag{4.1}$$

$$score(s_t, j) = \begin{cases} (s_t^T)h_j & \text{Dot-product Attention} \\ (s_t^T)W_a h_j & \text{General Attention} \\ V_a^T tanh(W_a[s_t; h_j]) & \text{Additive Attention} \end{cases} \tag{4.2}$$

2. Compute attention weights by applying softmax operation on the alignment scores as seen in equation 4.3.

$$\alpha_{(s_{t,j})} = \frac{\exp(s_{t,j})}{\sum_j \exp(s_{t,j})} \tag{4.3}$$

3. Compute context vector: As mentioned above, the context vector is a weighted sum of the attention weights and the hidden state representations for every output. The equation 4.4 below shows the computation of the context vector for a single output node.

$$C_t = \sum_{j=1}^{T} \alpha_{(s_{t,j})} h_j \tag{4.4}$$

More generally, the attention mechanism can be computed using three main components, the queries Q, the keys K and, the values V. The query can be compared with the decoder output from the previous layer s_{t-1} and the key with the encoder inputs h_j of the Bahdanau attention mechanism.

Self-attention Mechanism and Transformer Network

The self-attention mechanism is the fundamental building block of a transformer network. The self-attention mechanism finds the similarity between the input sequence. The idea is to create context-aware vectors by transforming the input vectors into their own queries, keys, and values and applying the generalized attention mechanism. Simply put, the queries, keys, and values are computed **on** in the same sequence unlike in the initially introduced attention mechanisms where the query is computed on the decoder values and keys on the encoder values.

Self-attention-based networks are superior to RNN(s) or attention-based networks with RNN(s) because they eliminate the need for a base seq2seq model like RNN(s) before computing attention. The attention mechanism can be performed directly on the input data sequence so parallelization is possible while eliminating the long-term dependence problems of RNNs and other problems that arise from long sequences and it can also effectively manage variable length input sequences.

A basic transformer block is made up of four main sub-modules which are: multi-head attention (MHA), positional encoding, layer norm and skip connections, and feed-forward neural network layers. Figure 4.2 shows the modules that make up a single transformer block.

1. **The multi-head attention:** The multi-head attention is concatenated multiple attention heads so that each attention head can pay attention to different parts of the input sequence. Single-head attention can also be computed in two (2) major steps; compute the queries, keys, and values from the input sequence, and use the scaled-dot product attention mechanism. The scaled-dot product is done by computing the dot products of the query with the keys, dividing each by $\sqrt{dk}$, and applying the softmax function to obtain the attention weights.
   The queries, keys, and values can be computed as:

$$\begin{cases} q_i = W^{(Q)T} x_i \\ k_i = W^{(K)T} x_i \quad \text{for } i = 1...T \\ v_i = W^{(V)T} x_i \end{cases} \tag{4.5}$$

   where, q_i is the query, k_i the key and v_i the value at a given time step and $W^{(Q)T}$, $W^{(K)T}$, $W^{(V)T}$ are trainable parameters.
   With $W^{(Q)T}$, $W^{(K)T}$, $W^{(V)T}$, the scaled dot product equation can be computed

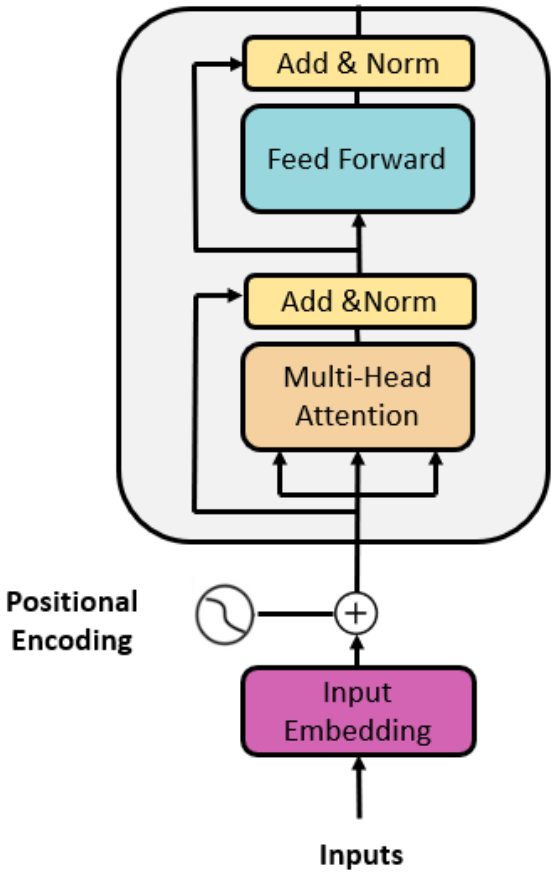


Figure 4.2: A Single Transformer Block.

as,

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4.6}$$

where $\sqrt{d_k}$ is the dimension of the key vector k and query vector q .
Multi-head attention (MHA) can be computed as,

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^A \tag{4.7}$$

2. **Positional Encoding**: As earlier stated, sequential data is a type of data in which order matters. However, after performing self-attention on the input sequence, the sequence is randomly rearranged commonly referred to as

"permutation invariant", so, to keep the data sequence in order, position encodings are used to add positional information to the input.

3. **Layer normalization and skip connections:** The output of the multi-head attention (MHA) is fed to a layer normalization layer after adding skip connections to the MHA output. The skip connections are used to help the network learn better the same way it was applied in convolutional neural networks like RESNET. It was observed that merely adding more layers to a neural network does not improve the performance instead it degrades it, so, explicitly adding identity layers such that the networks need to learn only residual representations makes the network perform better.

4. **Fully connected layer:** Many fully connected layers can be stacked together after the MHA block with skip connections and layer norm.

Finally, three (3) main architectures can be developed from basic transformer blocks. Just like in other seq2seq models, these architectures are generally referred to as the encoder, decoder, and encoder-decoder architectures.
While unfortunately, the architecture nomenclature for LLMs is somewhat confusing, the encoder-only architecture is an encoder and a decoder just not an auto-regressive decoder, and the decoder architecture is an auto-regressive encoder-decoder and the encoder-decoder network really means an encoder with an auto-regressive decoder. In this work, an encoder architecture is developed and used for multi-variate time series data prediction. **The encoder architecture is simply a stack of transformer blocks with a feed-forward neural network decoder as seen in Figure 4.3 below.**

4.3 Data Preparation

In Chapter 2 above, the feature extraction, feature selection, and working regimen data normalization data pre-processing stages have been extensively discussed. In this section, data preparation refers to how the data is formatted, shaped, and fed to the model for training and inferencing.
The way data is fed into a model can vary from one domain, application, or model to another. This can also impact the performance of the model greatly. For instance, if the data is independently and identically distributed (iid), the input is prepared in a

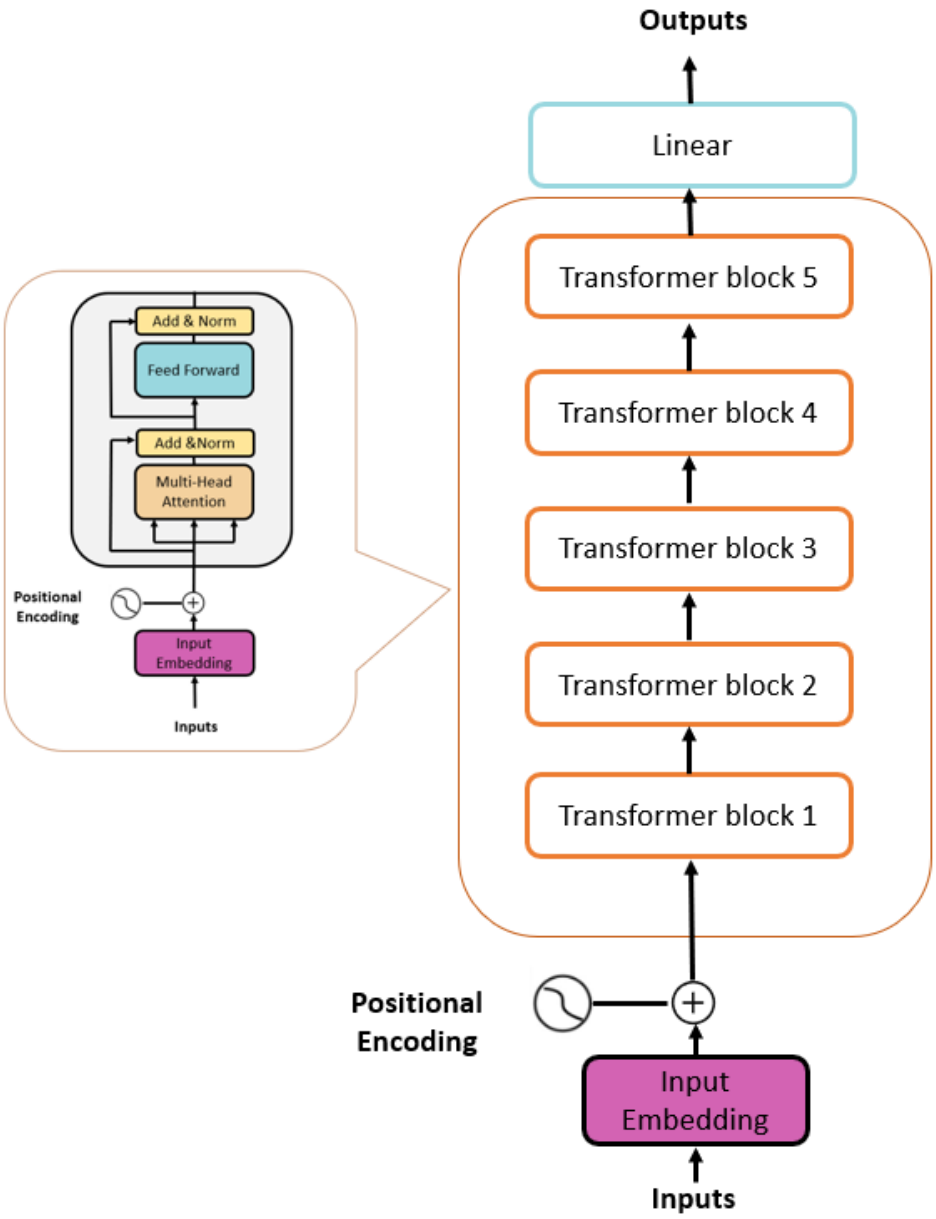


Figure 4.3: An Encoder Architecture.

tabular manner with two dimensions, $N * D$ where N is the number of data points and D is the number of features. Time series data input X can also be conveniently formatted to have shape $N * D$. However, this will not usually result in a good performance because one of the ways of enforcing stationarity in time series data is to use a sliding window to wrap portions of the data points for a constant time window. The most preferred format and effect of this way of formatting time series data are further discussed below.

As earlier mentioned, because time series data is not iid, it does not have strong-sense stationarity (i.e., the distribution in your stochastic process does not change over time). This affects the way time series data is formatted for use in ML models. To obey the strict assumption that a model can only learn if the data points are iid, non-stationary data are often transformed to become stationary. This is achieved by using a weaker form of stationarity commonly employed in signal processing and time series data analysis known as **weak-sense stationarity or covariance stationarity**.
The weak-sense stationarity implies that the mean and autocovariance don't change over time. The mean at time t should be the same as the mean at time $t+\tau$ for all τ, and the autocovariance for some random variable Y at time t_1 and another random variable Y at time t_2 is only a function of the difference between t_1 and t_2 and only needs to be indexed by one variable, τ rather than two variables. This equivalently implies that the autocorrelation depends only on $\tau = t_1 - t_2$. That is, suppose I take a window of constant length, say $t = 20$ and I pick any two time points in the series, if the time difference, τ is the same, and the covariance is the same, then the data is weak-sense stationary. The relationship between these points in the time series remains constant no matter where I look as long as they are the same distance apart. The weak sense stationarity mean and autocovariance equations can be computed as equations 4.8 and 4.9 respectively.

$$\mu_y(t) = \mu_y(t+\tau) \text{and for all } \tau \tag{4.8}$$

$$K_{yy}(t_1, t_2) = K_{yy}(t_1 = t_2, 0) \text{and for all } t_1,\, t_2 \tag{4.9}$$

In time series data analysis, to make the data stationary, the choice of τ is very important, and it is usually chosen empirically in ML applications. In this work, two input data formatting methods were experimented with. 1) Constant time window length. 2) Increasing time window length.

1. **Constant Time Window Length:** A sliding window with a constant length and a step size of 1, is used to wrap subsets of the datasets. The time series data input into the transformer is formatted as 3-dimensional data of shape $N * T * D$, where N is the number of data points, T is the sequence length, and D is the data dimension or size of the feature space. Different window lengths were tried, {2,5,10,20,30} and widow length, $T = 20$ was chosen because it gave the lowest loss on the validation set.

A detail specific to sequence-to-sequence (seq2seq) modeling is how to deal with varying-length sequences. When datasets of different sequence lengths are considered, the shorter sequences are padded with zeros at the ends to match the length of the longest pre-defined sequence. To ensure that these sequences padded with zeros do not pose as a bottleneck when computing the attention weights, padding masks are used to eliminate and remove their effect.
Figure 4.4 shows a schematic of how the sliding window works on time series data.

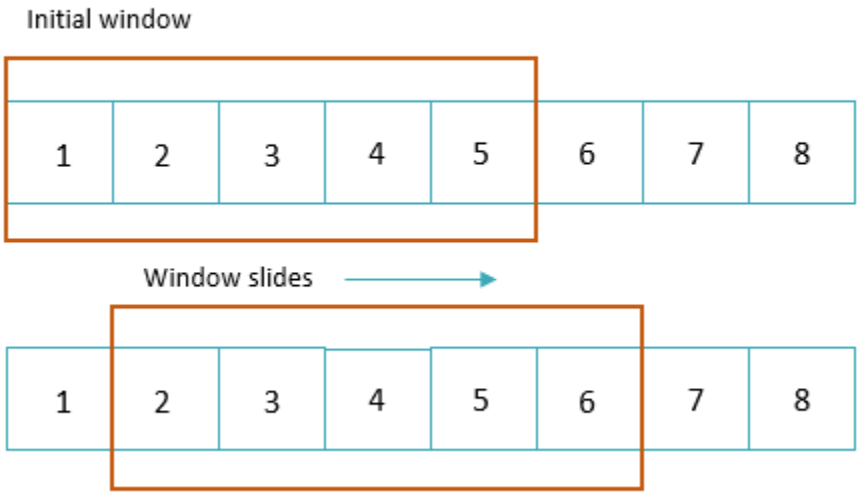


Figure 4.4: Sliding window

2. **Increasing time window length:** In this method, a window of increasing length is adopted. An initial minimum window length for every machine instance is set to 5 for the initial data point, then, successive data points are selected by using a window length that is constantly incremented by 1 step size at every time step. Figure 4.5 shows how the data points are wrapped by the expanding window.

The expanding window length method resulted in better prediction performance of the model than the sliding window method of constant length as shown in Table 4.1 below. While keeping other hyperparameters constant, the transformer models were trained from scratch five times, and the average RMSE and MAE errors from each

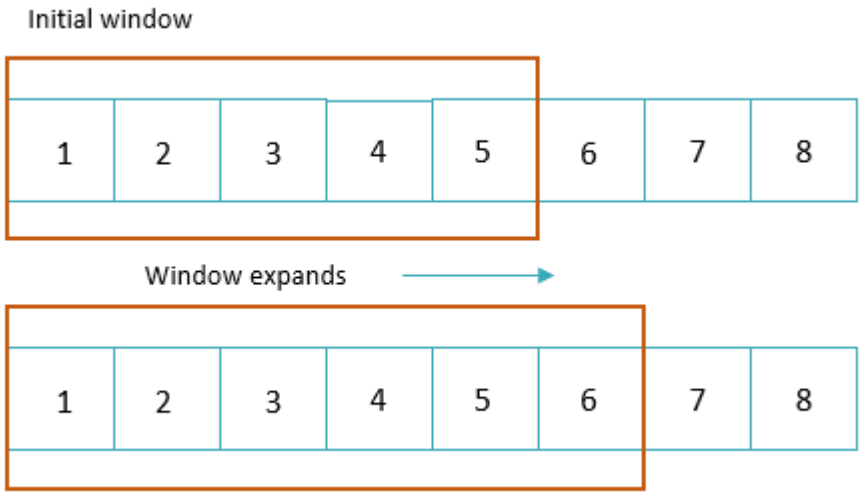


Figure 4.5: Expanding window

model on the test set show that the expanding window method surpasses the constant window length method for this given dataset by over 20% on both the RMSE and MAE metrics.

Table 4.1: Experimenting with different window lengths table of Results.

Experiment	Average RMSE	Average MAE
Sliding Window	22.07	17.15
Expanding Window	**18.027**	**14.17**
Percentage Improvement (%)	22.44	21.05

This palpable improvement in the performance of the expanding window method over the sliding window method can be tied to the nature of the degradation paths of machines. One of the most used and well-accepted statistical distributions in reliability engineering, the Weibull distribution introduced in 1939 by Waloddi Weibull divides machine life stages into three, the early failure stage, the constant failure stage, and the wear-out failure stage. With this understanding, the model's awareness of the initial stages of the machine's life will help it to make better predictions and that is what the expanding window method provides. Using a sliding window makes the model lose sight of the initial stages of the machine degradation, it confines the model's decisions to only t-time steps behind at every point. Also, gradually feeding an incremented sequence length to the model forces it to learn the underlying degradation path of the machines while taking into consideration the initial or early

failure stages if there be any for any given machine instance.

This method also makes predictions during inferencing more straightforward and built into the model. For the prognostics use case, the test data also has varying sequence lengths, this is because we should be able to determine the remaining useful life of the machine at any time before its failure, be it after 2 days of operation or after 50 days of operation. Using the expanding window length has helped the model to learn to make good predictions with varying length input.
The input data also has the format $N*T*D$ with T increasing from one data point to the other for a given machine instance. To handle the varying length inputs into the transformer model, the inputs with sequence lengths that are less than the maximum sequence length which is 378 (the life length of the machine with the longest lifespan in the training dataset) are padded with zeros to make all sequences in a batch have same sequence length, T.

Finally, attention masks of shape $N * T$ are also passed as inputs to the encoder transformer model. The attention masks are required because we do not want our self-attention mechanism to pay any attention to the padded tokens/inputs as they do not add any relevant information to the model. This is achieved by creating an $N * T$ padding array mask for every data point and setting the sequences we want the model to pay attention to one and the padded sequences in the input X to zero. Figure 4.6 adapted from the paper "Attention is all you need" by [61], shows how the self-attention mechanism is computed with an optional choice of attention masks when working with varying-length input sequences. Finally, the target, Y must also be considered and effectively passed into the model. In this model, the RUL is the target or label. General machines don't just fail abruptly, they usually first operate normally if they are not subjected to early-stage failures, and they deteriorate in some manner. So, following previous works [[74], [33], [32], [29], [35]] in literature, a piece-wise RUL was employed rather than the actual RUL as seen in Figure 4.7.
Using the degradation profile and the lifetime distributions of all the machines in the datasets, it was observed that the actual RUL label values RUL_early ≥ 125 and remain constant at the beginning and are assumed to degrade linearly until it fails.

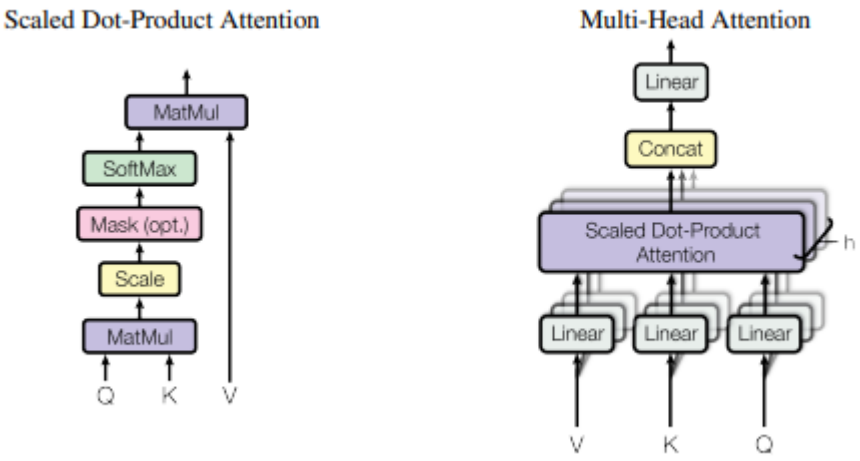


Figure 4.6: Scaled-Dot Product Attention Mechanism. *Adapted from [61]*

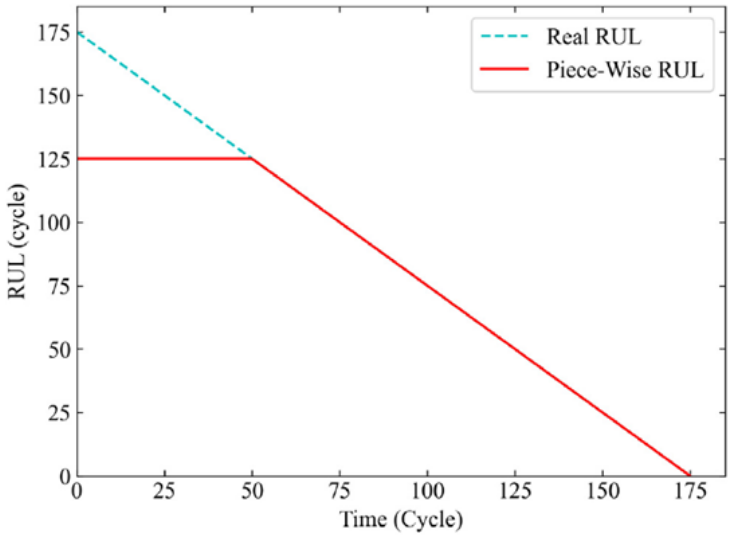


Figure 4.7: Rectified piecewise remaining useful life function.

4.4 Encoder-Transformer Model Experimentation and Hyperparameter Selection

Recall that in the "Attention is all you need" in [61], where transformers based solely on self-attention mechanisms were introduced to dispense the need for recurrent and convolution neural networks; the experiments were performed on machine translation tasks. However, in this work, we aim to find ways to efficiently leverage transformer models to make better predictions on time series data. To achieve this and transfer knowledge from the NLP domain, it is expedient that while following the base transformer architecture like using a self-attention mechanism and skip connections, we also need to tailor some of the transformer modules to time series data. In this section, we discuss the model-specific experimentations and hyperparameter tuning process.

Model-specific Experimentation

Three model-specific experiments were conducted, layer normalization versus batch normalization layers, and fixed versus learnable positional encodings. Three input data transformation methods were also tested. After these experiments, the layers and/or hyperparameters that gave the best results on the validation set based on the above-mentioned evaluation metrics were used to train the final transformer network. These experiments were carried out using an elimination method i.e. while keeping the other layers of the encoder network and hyperparameters constant, the layers of interest were tuned.

1. **Layer vs Batch Normalization:** For NLP applications as proposed in [61], layer normalization is used after the multi-head attention and feed-forward network layers leading to significant performance improvement over batch normalization. But in this work, consistent with the results from [86], batch normalization leads to better performance on time series data. Table 4.2 shows the experimental results of batch normalization vs layer normalization on the validation set.

   There is an 18.6% performance improvement of the transformer model on the scoring function metric and a 2.1% improvement on the RMSE metric on the validation dataset. This improvement supports the claim of the authors in [86] that batch normalization supersedes layer norm on time series data based on

Table 4.2: Layer vs batch normalization experiment table of results.

Experiment	Average RMSE	Average MAE
Layer Normalization	331.32	18.16
Batch Normalization	**279.37**	**17.79**
Percentage Improvement (%)	18.60	2.08

the time series benchmark datasets they experimented upon.

2. **Fixed vs Learnable Positional Encodings:** The positional encodings can be fixed or learnable parameters. The results in Table 4.3 show that fixed positional encodings give better results than learnable encodings which is also consistent with the results from the paper [4].

Table 4.3: Fixed vs learnable encodings experiment table of results.

Experiment	Average RMSE	Average MAE
Learnable Positional Encodings	304.23	18.51
Fixed Positional Encodings	**270.47**	**16.96**
Percentage Improvement (%)	12.48	9.12

There is a 12.5% performance improvement of the transformer model on the scoring function metric and a 9.12% improvement on the RMSE metric on the validation dataset when fixed encoding is used vs learnable encodings.

3. **Input Data Transformations:** In NLP, the input embedding layer converts each token into embeddings, commonly referred to as embedding vectors. This type of representation allows similar words to have similar embeddings. It is a technique that allows individual tokens to be represented as real-valued vectors and this technique is implemented in a way like neural networks, and hence the technique is often lumped into the field of deep learning.
   This approach of representing words as embedding vectors is considered one of the key breakthroughs of deep learning on challenging NLP problems. For time series problems, researchers try to adopt this idea by using linear projections or 1D convolution layers on the input data. According to [86], linearly projecting the input data onto a d-dimensional vector space where d represents the dimension of the transformer model D_{model} rather than using the original input data $D_{features}$ will result in a better performance.
   In this work, experiments that involve using a linear transformation, convo-

lution layers, and no initial transformations on the input data (i.e., $D_{model} = D_{features}$) were carried out and results show that using no transformations on the input data gave significantly better results.

This result is intuitively reasonable because of the self-attention mechanisms used in the encoder-transformer network. The self-attention mechanism is essentially a combination of dot products on sequences from the same data. The premise is, the dot product is a measure of similarity, the values of the dot product are large when two vectors are similar and low when the vectors are dissimilar. In time series data prediction, we leverage the autocorrelation between the features to make predictions and the dot product is analogous to correlation, so using the multi-head self-attention mechanism directly on the input data can find the autocorrelations in the multi-dimensional vector space resulting in the model learning better from the data directly.
Table 4.4 shows the result of these experiments on the validation set.

After performing these experiments, batch normalization, fixed positional encoding and not performing any initial transformations on the input data resulted in better model predictions.

Table 4.4: Different input data transformation methods experiment table of results.

Experiment	Average RMSE	Average MAE
1D-CNN	394.028	19.53
Linear Transformation	349.19	18.57
No Input Data Transformation	**331.32**	**18.16**

The results from the model-specific experiments also validate the recommendations in [86] on using batch normalization instead of layer normalization for time-series data but contrary to the results from [86] about performing a linear projection of the input data before the multi-head attention layer, it did not result in a better performance on this dataset.

Hyperparameter Selection

The main hyperparameters in an encoder transformer model are the number of transformer blocks in the encoder, the number of self-attention heads in each transformer block, the dropout rate, and the loss functions. These were tuned and Table 4.5 shows the best hyperparameters selected through cross-validation on the validation set for this work.

Table 4.5: Best hyperparameters for the Encoder-Transformer architecture.

Parameter	**Value**
dim model	14
dim FFW	256
num heads	7
num transformer blocks	3
dropout rate	0.4

Chapter 5

Evaluation, Analysis, and Comparisons

The proposed supervised learning-based SBM model, the Transformer-Assisted SBM, the improved unsupervised learning-based SBM model with the PCA algorithm, and The Encoder-Transformer models are evaluated on the test dataset using the predefined evaluation metrics in this chapter. All the models were developed using the NVIDIA Tesla P100 GPU with 16GB memory.
The predicted RUL values are analyzed and compared with the true RUL values. Section 5.1 focuses on the evaluation and analysis of the similarity-based models that are developed in this work while section 5.2 focuses on the direct approximate method. At the end of this chapter, the RMSE and score function values of these models on the test dataset are compared with other state-of-the-art models in the literature.

5.1 Similarity-based Models Evaluation and Analysis

5.1.1 The Transformer-Assisted SBM.

In the experiments carried out, the curve fitting process was not executed. The approximated HI-values were used directly to develop the similarity-based models. The proposed Transformer-Assisted SBM, based on the transformer HI-construction model was developed and evaluated on the test dataset. The RMSE and score values were evaluated on the test dataset with the maximal RUL_max, 200. Table 5.1 shows

the RMSE and score values for this model.

Table 5.1: RMSE and score values for the Transformer-Assisted SBM.

Model	RMSE	SCORE
Transformer-Assisted SBM	18.15	1959.65

5.1.2 An Improved Unsupervised Learning SBM with PCA

Just like in the supervised learning SBM, the curve-fitting process was excluded in the final model. The final unsupervised learning model used to approximate the HI values is the PCA, this is because it resulted in the best RMSE values across all the RUL_max values. The results are shown and analyzed below.
The RMSE and score values were evaluated on the test dataset with the maximal RUL_max, 200. Table 5.2 shows the RMSE and score values for this model.

Table 5.2: RMSE and score values for the unsupervised learning-based SBM with PCA.

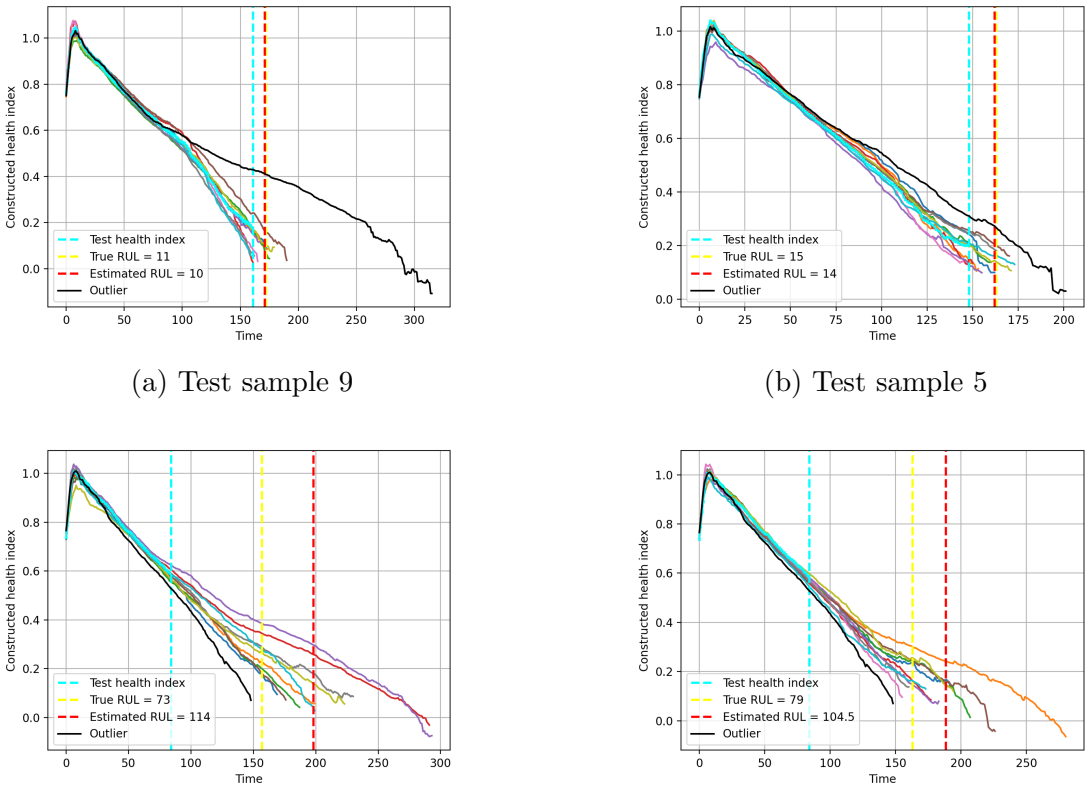
Model	RMSE	SCORE
Transformer-SBM	18.77	1556.28

5.1.3 Discussions

Visual Representation of the Transformer-Assisted SBM Performance On Test Samples

Figure 5.1 shows the visual representations of the similarity-based model predictions, the degradation profiles of the selected ensemble members using the Transformer-Assisted SBM, and the test components' degradation profile for some data samples chosen at random.

The cyan line is the test component's approximated HI values, the black is the approximated HI values for an outlier ensemble member with a dissimilar degradation profile to the nearest neighbors. The other lines are the approximated HI values for the k-nearest neighbors where k=10.
As seen it is in Figure 5.1, at the late-life stages of the machines which are depicted by smaller true RUL values, the SB model's predictions are very accurate. For the test sample 9 shown in Figure 5.1a, the true RUL is 11 and the estimated RUL is 10,



(a) Test sample 9

(b) Test sample 5

(c) Test sample 12

(d) Test sample 10

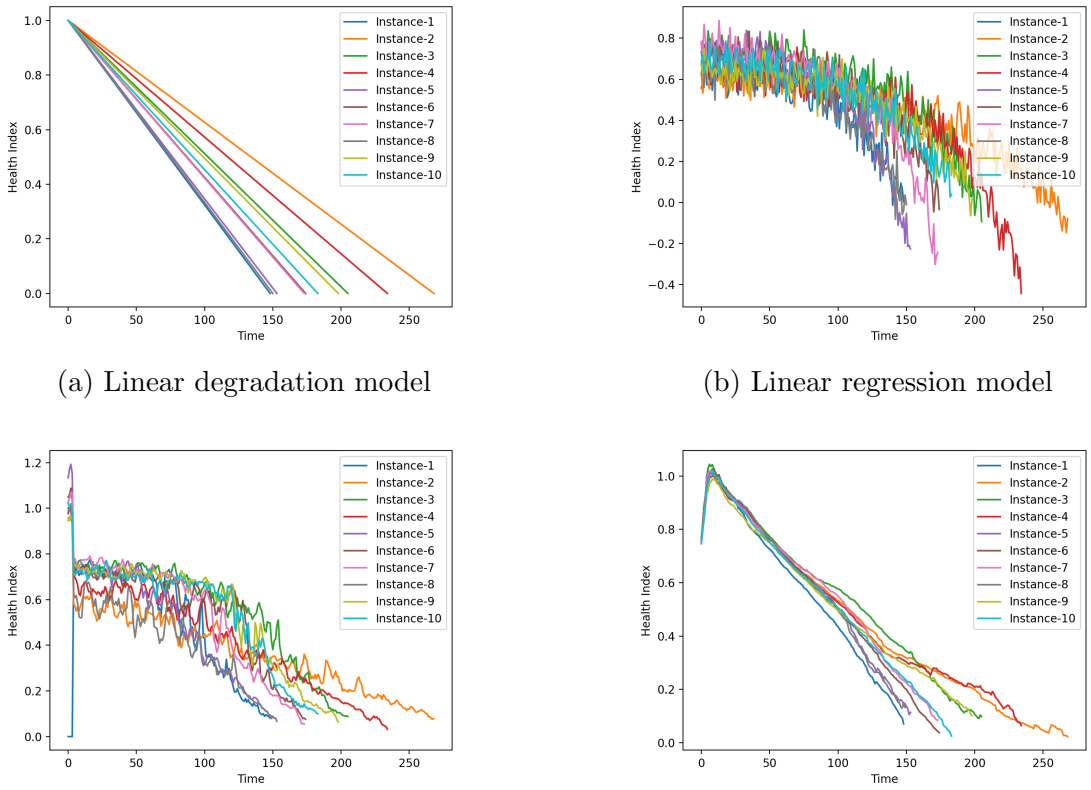
Figure 5.1: Visual representations of the SBM predictions.

this same behavior can be observed in Figure 5.1b where the predicted RUL tightly tracks the true RUL values with a prediction error of only 1 day. Also, the SB model favors early predictions more than late predictions which is the preferred behavior for a prognostics model.

However, the typical behavior of SBM models at the early life stages, depicted by the large true RUL values, is shown in Figures 5.1c and 5.1d for test samples 10 and 12 respectively. The error band is loose and the predicted RUL values do not track the true RUL values as tightly as it does in the late-life stages.

Model Approximated HI Values Representations

To visualize the approximated HI values and access the efficacy of each supervised and unsupervised machine learning model used in this work, the approximated health indices for the first ten training instances were plotted.



(a) Linear degradation model

(b) Linear regression model

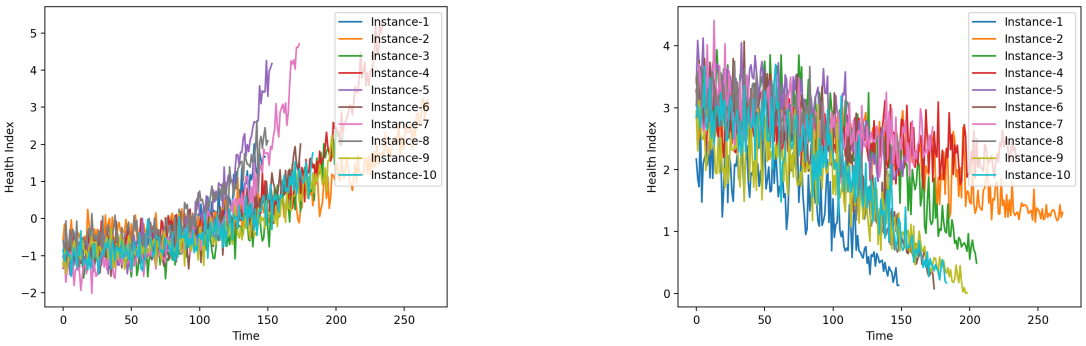
(c) CNN model

(d) Encoder-transformer model

Figure 5.2: Approximated HI values for the supervised learning models.

Figure 5.2a shows the original linear degradation model that was approximated by the supervised learning models. Figure 5.2 shows the approximated HI values for the different supervised learning models and figure 5.3 for unsupervised learning models. From the sub-figures of 5.2, we can immediately see how well each model was able to approximate the linear degradation model. The linear regression model was not able to approximate the linear degradation model well, however, it learned a representation that actually resembles the degradation profile of the machine more, where the degradation does not begin instantly. This visualization also gives insights into why the winning model in the PHM08 competition was able to still perform exceptionally even though they used a linear regression model for the HI values approximation. The CNN model approximations have a more meandering curve but it follows a more linear path than the linear regression model. As mentioned earlier, transformers are very complex models that have achieved tremendous results in LLMs and the transformer model plot shows the efficacy of this model in learning complex representations.

The approximated HI values produced by the transformer model show the closest approximation to the linear model. So even though a linear degradation model is not the best assumption for the degradation path of the machine(s), being able to accurately approximate the linear model resulted in exceptional behavior.



(a) PCA model

(b) FAE model

Figure 5.3: Approximated HI values for the unsupervised learning models.

The sub-figures of 5.3 for unsupervised learning representations show very interesting plots. The PCA plot has an upward trend while the fully-connected autoencoder plot has a downward trend. This difference in trend directions can be attributed to the type of model used. PCAs are linear models and the best dimensions that retain the highest variance that can be modeled by a linear model are learned. Another interesting part of the PCA plot is its similarity with the linear regression plot just with an upward trend, this similarity in trend is linked to the fact that they are both linear models.

The FAE model HI value representations for the first neuron of the latent space show a very interesting result. It can be observed from the FAE plot that just as we observed through data analysis, the machine does not follow a linear degradation path. The degradation does not begin at the early stages of the machine's life and the difference in the degradation profiles can only really be observed towards the end of life. This attribute makes it difficult for prognostics models, both similarity-based and direct approximation models to make good predictions at the early life stages of the machines.

Main Points - Similarity-base Models

1. The similarity-based model is a combination of a prognostics and a condition monitoring model. The health indices can be used as a single value representation of the machine's condition at any given time and this value can be used to monitor the machine in real-time.

2. The health index representations learned by unsupervised learning models can give us a better idea of the degradation path of the machine.

3. Some of the benefits of the similarity-based model over the direct approximation model are:

   (a) Similarity-based models can be used as condition monitoring and degradation modeling tool.

   (b) The health indices are simply the conditions of the machines. They represent the degradation levels and these values can be used directly as the reinforcement learning model state representations.

   (c) It is explainable: One of the major drawbacks of the direct approximation prognostics model is its unexplainability because they use black box models. SBMs are easily explainable and the reasoning behind the model estimations can be easily accessed.

   (d) Finally, SBMs can still be used even when there are no targets. Similarity-based models are very intuitive.

4. Similarity-based model for prognostics also has some disadvantages:

   (a) It has so many methods and stages of development. The hyper-parameter space can quickly grow exponentially.

   (b) The space requirement is high. Because the training instances library is required during evaluation, unlike direct approximation models.

   (c) Time complexity: As the number of training samples or ensemble members increases, the space requirement also increases. It has a linear time complexity of 0(N). This also affects the prediction speed, as the samples grow, the prediction time gets longer.

5.2 Encoder-Transformer-based Model Results

Visual Representation of Predicted vs True RULs

To evaluate this model, the RUL predictions on a set of engine units picked randomly from the validation set are plotted in Figure 5.4. Figure 5.5 shows the smoothed versions of these plots, we can see that the estimated RULs also have the same downward trend as the true RULs after RUL_early till failure. The predicted RUL tracks the true RUL very closely at every time step. Also, the predicted RULs at the early life are near to the constant RUL_early at the beginning of the unit life.

True vs Predicted RUL Error Statistical Analysis

Finally, some descriptive statistics of the RUL errors (true minus predicted RULs) were conducted. Histogram, box, and probability distribution plots for the errors between the true and the predicted RULs were plotted in Figures 5.6a and 5.6b.

The mean, median, and mode are represented with cyan, red, and yellow dashed lines on the histogram plot while the maximum and minimum RUL values are represented with grey dashed lines. Table 5.3 below shows the mean, mode, median, skewness coefficient, and standard deviation of the RUL errors.

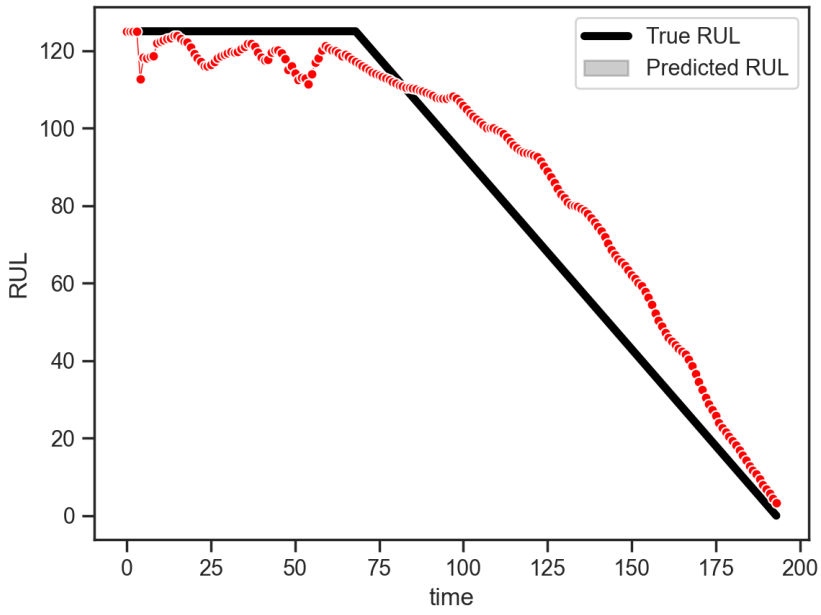
From the histogram and probability distribution plots, we can clearly see that the errors are left skewed which means there are probably outliers on the lower bound of the data distribution. The box plot below clearly shows that. Due to the outliers on the lower bound, the mean is shifted to the left of the median or Q3-Q2 < Q2-Q1.

The intensity of skewness shows how much deviation the probability distribution is from a normal probability distribution and if the distribution can be adequately represented with the mean and standard deviation. The Pearson coefficient of skewness was used to check the degree of skewness the coefficient of skew is -0.74 which means it is moderately left skewed. It is > -1 so the distribution is not badly skewed.

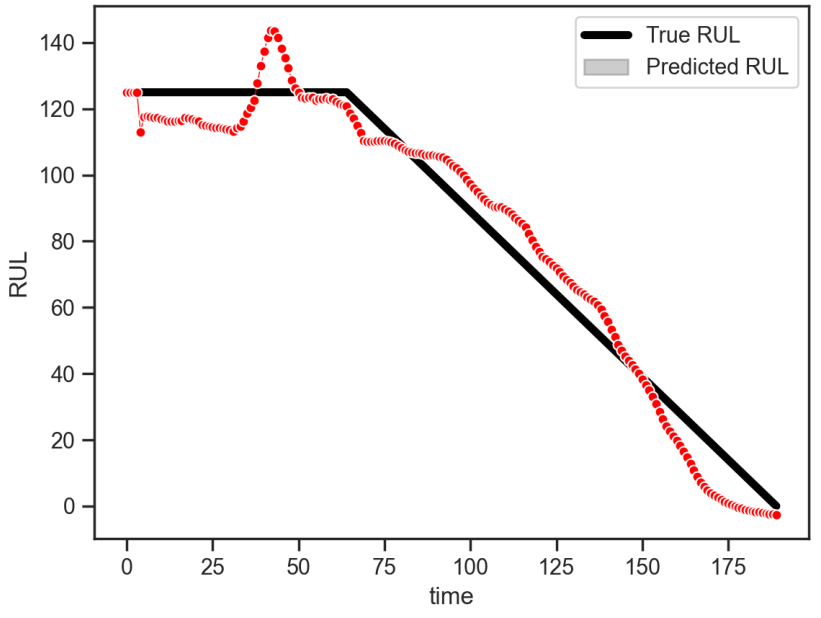
Because the data is moderately skewed, the error population is represented with the median and standard deviation of the RUL errors.

5.3 Comparison of Results

Finally, the performance of the proposed models was evaluated on the test dataset with the RMSE, and score function metrics were compared with the results from other state-of-the-art models in the literature. The proposed Encoder-Transformer model

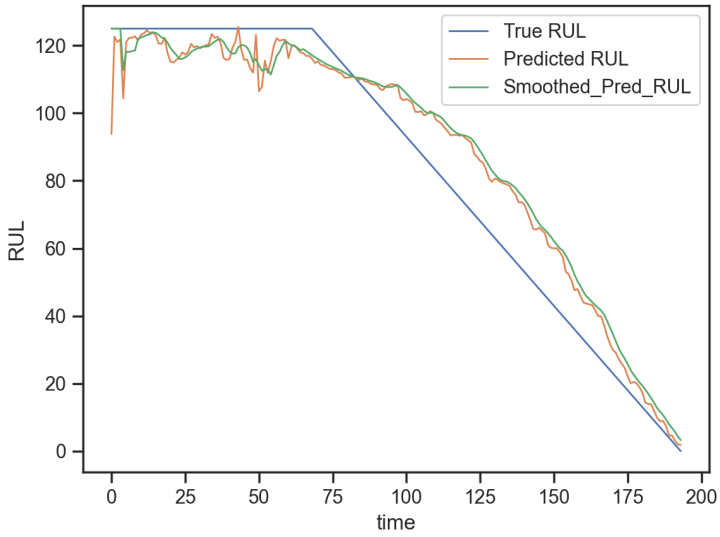


(a) Prediction result of engine 205

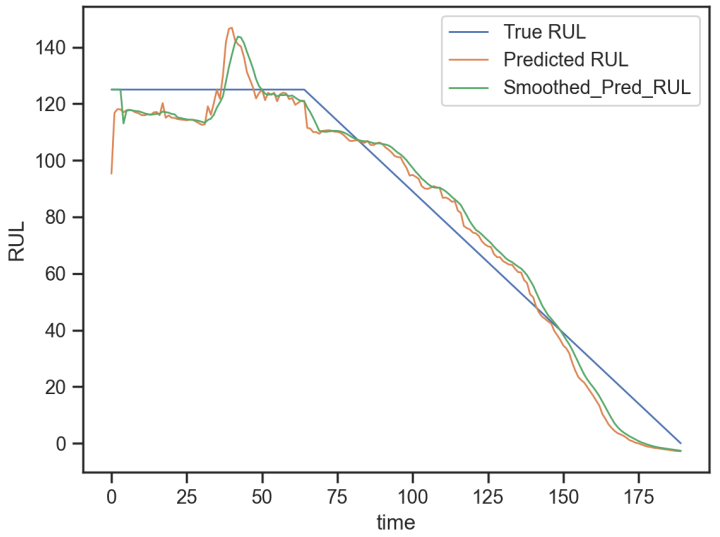


(b) Prediction result of engine 246

Figure 5.4: Prediction results of two engines in the validation datasets.



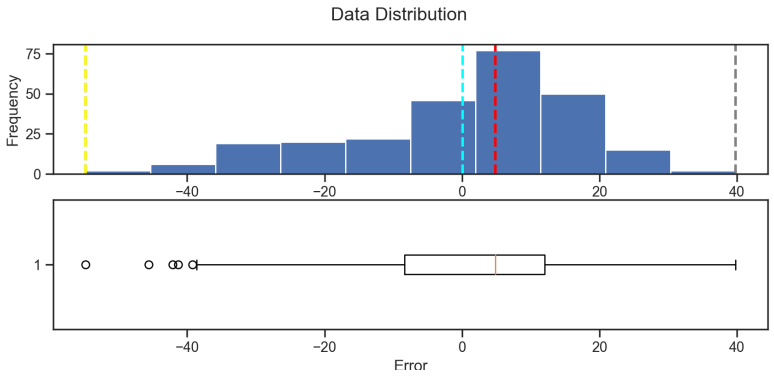
(a) Prediction result of engine 205b



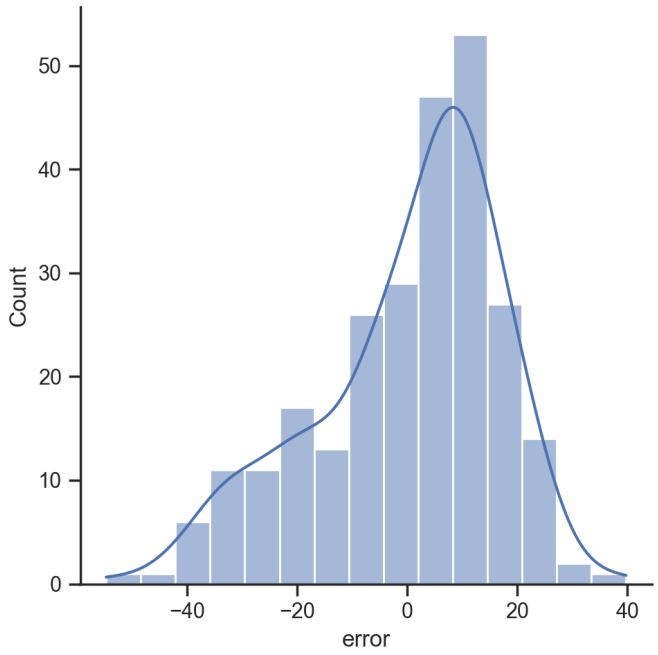
(b) Prediction result of engine 246b

Figure 5.5: Prediction results of two engines in the validation datasets + smoothing.

using the direct approximation method in this work surpasses all the other models it was compared with. It surpassed the improved similarity-based models developed in this work and the other 14 state-of-the-art models it was compared within the literature.



(a) Histogram and Box plots of RUL errors.



(b) Probability distribution of RUL errors.

Figure 5.6: Descriptive statistics table for RUL error plots.

Table 5.3: Descriptive statistics table for RUL errors.

Statistics	**Values**
Mean	0.09
Median	4.85
Mode	-54.76
Std	16.9
Skew coefficient	-0.75

A percentage performance increase of 45.5% on the score function was achieved over the second-best model in literature [74]. More importantly, it should be noted that this is not the first transformer model that has been used on this dataset. Larger and more complex transformer models such as the encoder-decoder model with inputs that have been transformed with convolutional layers in [74] and encoder-decoder + CNN with channel attention models in [35] have been used and from the results this did not greatly improve the performance of the network.

This observation lays emphasis on the experimental results from using different input data transformation methods that using the time series data as it is without any transformations gives better results, it also proves that having a larger model does not necessarily guarantee a better result on the unseen dataset. However, this result is also a precursor to future improvements that can be made to this model, a larger and more complex encoder-decoder network containing more parameters can be used in the future with the current setup, and better results are hypothetically expected.

The exceeding performance of the encoder-transformer model does not completely rule out the improved similarity-based models developed in this work. As it is seen in Table 5.4, the improved unsupervised learning-based SBM with PCA developed in this work surpassed the other state-of-the-art counterparts in the literature with more sophisticated autoencoder models. More interestingly, it ranks second amongst the other 14 state-of-the-art models it was compared with. This shows the effectiveness of similarity-based models for RUL prediction tasks and other domains it can be applied to.
This result also shows that the complexity of the autoencoder model does not directly translate to a better similarity-based model.

The result of the supervised learning-based model proposed in this work, even though it does not surpass the result of the improved unsupervised learning model with PCA also developed in this work, shows a very competitive result. It ranks fourth amongst all the state-of-the-art models it was compared with. This asserts the claim that the complexity of the supervised learning model and how well it behaves on unseen datasets can improve the performance of the similarity-based model. This also suggests that this is a method worth researching further, researchers have not

paid attention to the supervised learning method of HI construction but these results show that it is an area worth researching.

To summarize, the models developed in this work rank first, second, and fourth best amongst the other 14 state-of-the-art models they were compared with including the other transformer models in the literature.

Table 5.4: Performance comparison with other models.

Category	Models	RMSE	Score
SBM	RNN Autoencoders [18]	2650	19.59
	BI-RNN Autoencoders [82]	3099	22.07
	USL-Best (PCA-SBM)	**1556.28**	**18.77**
	SL-Best (Transformer-Assisted SBM)	**1959.65**	**18.15**
Direct Apprx	HMC [5]	19400	20.74
	CNN [56]	13570	30.29
	DCNN [30]	10412	22.36
	MODBNE [88]	5585	25.05
	LSTM-FNN [95]	4450	24.49
	CatBoost [11]	3493.2	15.8
	RBM-LSTM-FNN [13]	3366	22.73
	DFC-LSTM [63]	3296.3	20.3
	Auto-encoder [85]	2650	19.59
Transformers	GCU-Transformer [89]	N/A	22.81
	BI-GRU-TSAM [41]	2264	18.94
	Double-Attention-based architecture [35]	1575	17.08
	Encoder transformer (this work)	**1081.878**	**16.96**

Finally, the true versus predicted RUL values on the test dataset are plotted in Figure 5.7.

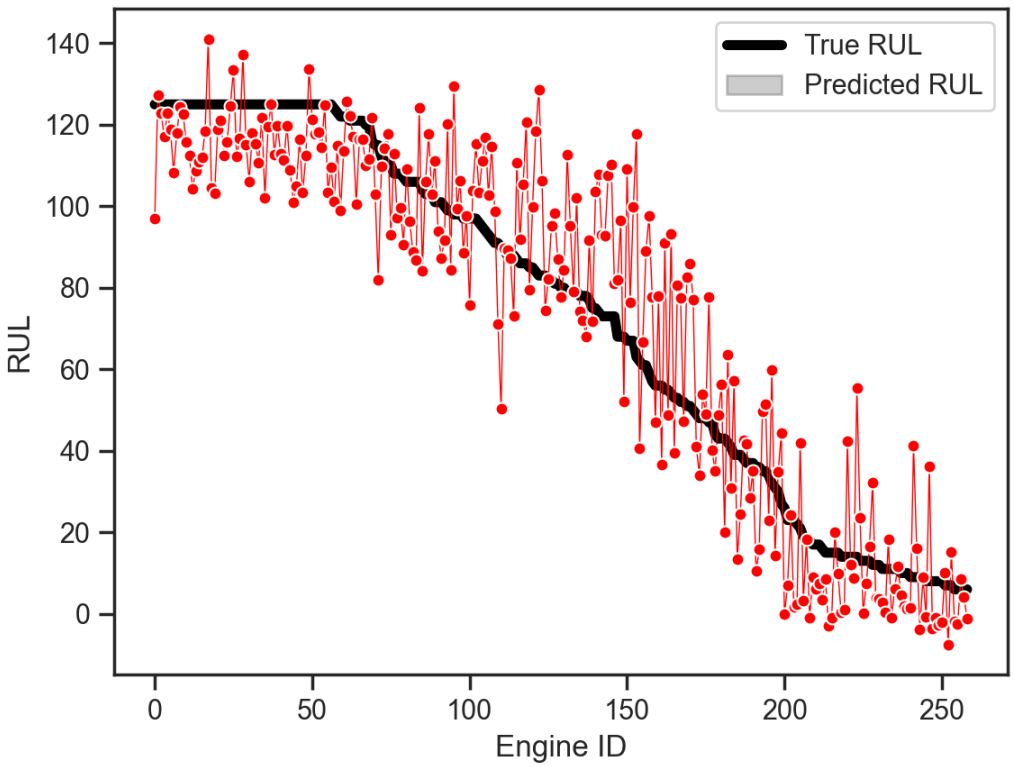


Figure 5.7: True RUL vs Predicted RUL on test dataset.

Chapter 6

Conclusions and Future Works

In this thesis, the two main machine-learning-based methods of remaining useful life (RUL) prediction, the similarity-based and direct approximation methods were investigated, optimized, developed, and compared against each other and other state-of-the-art models in the literature.
A sequential approach through experimentation was used to optimize every stage of the development process of similarity-based models (SBM) for remaining useful life prediction. Using a bottom-up approach, the best methods amongst the most commonly used methods in the literature for the second stage of an SBM development process were found. Also, drawing ideas from a scalable, high-performing RUL fusion method in the literature, the softmax-weighted sum RUL fusion method was proposed, developed, and evaluated on the validation dataset. Compared to the RUL fusion method proposed in [57] it resulted in lower remaining useful life errors.

For the health index construction stage, a novel supervised learning-based Transformer-Assisted SBM and an improved unsupervised learning-based model with PCA were developed. Finally, a novel direct approximation model, the Encoder-Transformer model based on the self-attention mechanism is proposed and developed. This model leverages Large Language Model (LLM) concepts for more efficient time series data representation learning and prediction of remaining useful life.
These models were compared with twelve other state-of-the-art models in the literature on the reference dataset and the Encoder-Transformer models it was compared with and ranking close to the best is the improved unsupervised learning-based model with PCA model also developed in this work. The novel supervised learning-based Transformer-Assisted SBM model ranked fourth position.

The Encoder-Transformer model outranks the next best model it was compared in the literature. It should be noted that this model also used a transformer architecture with channel attention-based CNN. This exceeding performance of the proposed model reinforces the benefit of the sequential approach of experimentation used in this work.
Through experimentation, it was observed that using deep learning models to create vector embeddings or transforming the inputs passed to the transformer model for time series data resulted in a depreciated performance, so the performance witnessed in this Encoder-Transformer model validates the result of this experiment. It also proves that having a larger model does not necessarily guarantee a better result on unseen datasets. This is because, with just the encoder architecture used in this model, a very good performance was achieved. However, this result also shows how future improvements can be made to this model. A larger and more complex encoder-decoder network can be used in the future with the current setup, and better results are expected.

The sequential approach used in this work also helps to acquire valuable insights on how to improve similarity-based models. These insights were used to develop improved and novel similarity-based models. More importantly, the insights drawn from the experiments suggest areas of relevant future work

The results show the efficacy of similarity-based models. However, it can be observed that because the SBM development stage where the estimated RUL values are generated is separated from the function approximation stage, the model is not specifically trained to minimize the RUL prediction errors. The cost function is not formulated and minimized based on the actual output. To cater to this, reinforcement learning-assisted deep learning models which include the actual RUL output in the reward formulation can be used to improve similarity-based models. This is an area of future work that has not yet been explored in the field of prognostics and health management before.

Also, the results from using different unsupervised-learning models of varying complexities for the HI construction do not support the claim that more complex models result in better similarity-based models. From the experiment, the PCA, a linear

autoencoder model resulted in lower RUL errors compared with the fully-connected and convolutional autoencoder. This also calls for more research into the reason behind this behavior. How these encoded embeddings can be efficiently translated to HI values that result in better performance of similarity-based models is an area of future work.

Finally, the supervised learning-based HI construction method has not been explored in literature, aside from [65] which motivated the approach. This work is the first to explore the supervised learning HI construction method to the best knowledge of this author at the time this work was done. The results obtained from the Transformer-Assisted similarity-based model show tremendous potential and it is an area worth researching further.

Appendix A

Additional Information

Reference table containing a summary of the state representations of all reviewed publications in the Reinforcement and Deep Reinforcement Learning-based Solutions for Machine Maintenance Planning, Scheduling Policies, and Optimization Manuscript.

Table A.1: Reference Table containing a summary of all reviewed papers

Ref No	System	Degradation Model/State Representation
[69]	Multi unit	Gamma distribution
[73]	Single unit	Exponential distribution
[62]	Multi unit	Assumed PHM capabilities (RUL)
[25]	Multi unit	Weibull distribution
[21]	Multi unit	Data-driven relablilty model
[90]	Multi unit	Gamma and Poisson distribution
[38]	Multi unit	Weibull distribution
[80]	Multi unit	Gamma and Poisson distribution
[45]	Single unit	Exponential distribution
[1]	Single unit	Gaussian distribution
[3]	Multi unit	Assumed failure probability
[91]	Single unit	Weiner process and Prognostics(RUL)
[48]	Multi unit	Exponential distribution
[60]	Multi unit	Weibull distribution
[81]	Multi unit	Gamma and Poisson distribution
[94]	Single unit	Gamma distribution
[14]	Multi unit	Markov discretized process
[53]	Multi unit	Weibull distribution
[75]	Multi unit	Randomly triggered
[76]	Single unit	Linear function
[64]	Single unit	Markov discretized process
[96]	Multi unit	Gamma distribution
[19]	Multi unit	Assumed failure probability

Table A.1 – *Continued from previous page*

Ref No	System	Degradation Model/State Representation
[52]	Multi unit	Assumed PHM capabilities (RUL)
[8]	Multi unit	Exponential distribution
[70]	Multi unit	Markov discretized process
[9]	Multi unit	Assumed PHM capabilities (RUL)
[58]	Multi unit	Known failure rates
[39]	Single unit	Poisson distribution
[78]	Multi unit	Reliability model
[44]	Multi unit	Equipment degradation behaviour from data
[49]	Multi unit	Assumed PHM capabilities (RUL)
[47]	Single unit	Weiner process
[43]	Multi unit	Equipment degradation behaviour from data
[20]	Single unit	Assumed PHM capabilities (RUL)
[55]	Multi unit	Markov discretized process
[2]	Multi unit	Equipment degradation behaviour from data
[51]	Single unit	Not specified
[31]	Single unit	Markov discretized process
[72]	Multi unit	Capacity degradation model
[77]	Single unit	Markov discretized process
[46]	Multi unit	Markov discretized process
[28]	Multi unit	Markov discretized proces
[22]	Multi unit	Data driven relablilty model
[24]	Single unit	Uniformly distribution & Linear function
[16]	Multi unit	Markov discretized process
[26]	Multi unit	Markov discretized process
[92]	Multi unit	Predefined discrete state markov process and gamma distribution
[71]	Single unit	Data-driven prognostics algorithm - supervised learning regressor
[54]	Multi unit	Not required
[27]	Single unit	Data-driven prognostics algorithm-CNN with Monte Carlo dropout
[7]	Multi unit	Poisson distribution
[50]	Single unit	Markov discretized process
[15]	Multi unit	Weibull distribution
[6]	Multi unit	Random process
[68]	Multi unit	Gamma distribution
[87]	Multi unit	Weibull and Gamma distribution
[36]	Multi unit	Weibull distribution
[93]	Single unit	Not required
[12]	Multi unit	RUL prediction

Appendix B

Additional Information

Experimenting with the effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) and different HI approximation techniques

Table B.1: The effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) on the base model experimentation results.

L	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
1.5	3.26	2.26	0.35	0.77	0.51	0.05
3	3.26	2.26	0.35	0.77	0.51	0.05
Full	3.26	2.26	0.35	0.77	0.51	0.05

Table B.2: The effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) on the HI constructed with feed-forward NN experimentation results.

L	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
1.5	21.3	14.2	4.60	9.40	3.90	0.60
2	21.3	12.8	7.40	9.40	3.20	1.20
3	17.8	12.1	6.40	6.20	2.90	1.00
4	20.6	12.1	6.70	8.70	2.90	1.10
5	20.6	10.4	6.00	8.70	2.30	0.90
Full	**17.8**	**7.20**	**4.20**	**6.20**	**1.40**	**0.60**

Table B.3: Experimenting with the effect of changing the time window length, L when estimating the RUL at different lifetime percentages (pct) on the HI constructed with PCA.

L	**RMSE**			**SCORE**		
	50%	70%	90%	50%	70%	90%
1.5	21.7	19.5	10.3	9.80	7.50	2.10
2	19.6	19.5	9.20	7.60	7.50	1.70
3	23.1	19.8	8.80	11.5	7.80	1.60
4	22.0	16.3	8.80	10.2	5.10	1.60
5	22.0	15.6	9.20	10.2	4.70	1.70
Full	18.9	14.6	10.3	7.00	4.10	2.10

Appendix C

Additional Information

Developed ML Model Hyperparameters Tables

Table C.1: Tuned hyperparameters for the fully-connected ANN model.

Hyperparameters	**Value**
No of layers	2
No of neurons in layer 1	32
No of neurons in layer 2	32
Activation function	relu
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
Regularization rate layer 1 , 2	0.00, 0.003
r2-score	**0.73**

Table C.2: Tuned hyperparameters for the multi-tail ANN model.

Hyperparameters	**Value**
No of tails	8
No of layers/tail	1
No of neurons in all layer	5
No of neurons in output layer	16
Activation function	relu
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
r2-score	**0.52**

Table C.3: Tuned hyperparameters for the CNN model.

Hyperparameters	**Value**
No of convolutional layers	2
No of filters in layer 1	32
No of filters in layer 2	64
Filter size	3
No of neurons in output layer	16
Activation function	relu
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
Dropout rate	0.4
r2-score	**0.77**

Table C.4: Tuned hyperparameters for the LSTM model.

Hyperparameters	**Value**
No of LSTM layers	3
No of neurons in layer 1	32
No of neurons in layer 2	24
No of neurons in layer 3	16
Activation function	tanh
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
r2-score	**0.55**

Table C.5: Tuned hyperparameters for the fully-connected autoencoder model.

Hyperparameters	**Value**
Latent dim	3
No of encoder layers	1
No of decoder layers	1
No of neurons in encoder and decoder	5
Activation function	relu
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
r2-score	**0.83**

Table C.6: Tuned hyperparameters for the CAE model.

Hyperparameters	**Value**
Latent dim	3
No of encoder conv layers	2
No of decoder conv layers	2
No of filters in encoder and decoder layer 1	32
No of filters in encoder and decoder layer 2	64
Filter size	3
Activation function	relu
Optimizer	ADAM
Loss function	MSE
Learning rate	0.001
Dropout rate	0.4
r2-score	**0.94**

Bibliography

[1] Aniket Adsule, Makarand Kulkarni, and Asim Tewari. Reinforcement learning for optimal policy learning in condition-based maintenance. *IET Collaborative Intelligent Manufacturing*, 2(4):182–188, 2020.

[2] Pedro Andrade, Catarina Silva, Bernardete Ribeiro, and Bruno F Santos. Aircraft maintenance check scheduling using reinforcement learning. *Aerospace*, 8(4):113, 2021.

[3] CP Andriotis and KG Papakonstantinou. Deep reinforcement learning driven inspection and maintenance planning under incomplete information and constraints. *Reliability Engineering & System Safety*, 212:107551, 2021.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Maximilian Benker, Lukas Furtner, Thomas Semm, and Michael F Zaeh. Utilizing uncertainty information in remaining useful life estimation via bayesian neural networks and hamiltonian monte carlo. *Journal of Manufacturing Systems*, 61:799–807, 2021.

[6] Joyjit Chatterjee and Nina Dethlefs. Deep reinforcement learning for maintenance planning of offshore vessel transfer. In *Developments in Renewable Energies Offshore*, pages 435–443. CRC Press, 2020.

[7] Jiahao Chen and Yu Wang. A deep reinforcement learning approach for maintenance planning of multi-component systems with complex structure. *Neural Computing and Applications*, pages 1–14, 2023.

[8] Yiming Chen, Yu Liu, and Tangfan Xiahou. A deep reinforcement learning approach to dynamic loading strategy of repairable multistate systems. *IEEE Transactions on Reliability*, 71(1):484–499, 2021.

[9] Michele Compare, Luca Bellani, Enrico Cobelli, Enrico Zio, Francesco Annunziata, Fausto Carlevaro, and Marzia Sepe. A reinforcement learning approach to optimal part flow management for gas turbine maintenance. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 234(1):52–62, 2020.

[10] Angus Dempster, François Petitjean, and Geoffrey I Webb. Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, 34(5):1454–1495, 2020.

[11] Hongbin Dong, Tao Li, Rui Ding, and Jing Sun. A novel hybrid genetic algorithm with granular information for feature selection and optimization. *Applied Soft Computing*, 65:33–46, 2018.

[12] Mohaiad Elbasheer, Francesco Longo, Giovanni Mirabelli, Antonio Padovano, Vittorio Solina, and Simone Talarico. Integrated prescriptive maintenance and production planning: a machine learning approach for the development of an autonomous decision support agent. *IFAC-PapersOnLine*, 55(10):2605–2610, 2022.

[13] André Listou Ellefsen, Emil Bjørlykhaug, Vilmar Æsøy, Sergey Ushakov, and Houxiang Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183:240–251, 2019.

[14] Maomao Feng and Yang Li. Predictive maintenance decision making based on reinforcement learning in multistage production systems. *IEEE Access*, 10:18910–18921, 2022.

[15] Waldomiro Ferreira, Cristiano Cavalcante, and Phuc Do Van. Deep reinforcement learning-based maintenance decision-making for a steel production line. In *31st European Safety and Reliability Conference, ESREL 2021*, 2021.

[16] Jakob Giner, Raphael Lamprecht, Viola Gallina, Catherine Laflamme, Lennard Sielaff, and Wilfried Sihn. Demonstrating reinforcement learning for maintenance scheduling in a production environment. In *2021 26th IEEE International*

Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8. IEEE, 2021.

[17] Bernard Grabot, Bruno Vallespir, Gomes Samuel, Abdelaziz Bouras, and Dimitris Kiritsis. *Advances in Production Management Systems: Innovative and Knowledge-Based Production Management in a Global-Local World: IFIP WG 5.7 International Conference, APMS 2014, Ajaccio, France, September 20-24, 2014, Proceedings, Part II*, volume 439. Springer, 2014.

[18] Narendhar Gugulothu, Vishnu Tv, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Predicting remaining useful life using time series embeddings based on recurrent neural networks. *arXiv preprint arXiv:1709.01073*, 2017.

[19] Jueming Hu, Yuhao Wang, Yutian Pang, and Yongming Liu. Optimal maintenance scheduling under uncertainties using linear programming-enhanced reinforcement learning. *Engineering Applications of Artificial Intelligence*, 109:104655, 2022.

[20] Yang Hu, Xuewen Miao, Jun Zhang, Jie Liu, and Ershun Pan. Reinforcement learning-driven maintenance strategy: A novel solution for long-term aircraft maintenance decision optimization. *Computers & Industrial Engineering*, 153:107056, 2021.

[21] Jing Huang, Qing Chang, and Jorge Arinez. Deep reinforcement learning based preventive maintenance policy for serial production lines. *Expert Systems with Applications*, 160:113701, 2020.

[22] Jing Huang, Qing Chang, and Nilanjan Chakraborty. Machine preventive replacement policy for serial production lines based on reinforcement learning. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 523–528. IEEE, 2019.

[23] Racha Khelif, Simon Malinowski, Brigitte Chebel-Morello, and Noureddine Zerhouni. Rul prediction based on a new similarity-instance based approach. In *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pages 2463–2468. IEEE, 2014.

[24] Michael Knowles, David Baglee, and Stefan Wermter. Reinforcement learning for scheduling of maintenance. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 409–422. Springer, 2010.

[25] Andreas Kuhnle, Johannes Jakubik, and Gisela Lanza. Reinforcement learning for opportunistic maintenance optimization. *Production Engineering*, 13(1):33–41, 2019.

[26] Raphael Lamprecht, Ferdinand Wurst, and Marco F Huber. Reinforcement learning based condition-oriented maintenance scheduling for flow line systems. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–7. IEEE, 2021.

[27] Juseong Lee and Mihaela Mitici. Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics. *Reliability Engineering & System Safety*, 230:108908, 2023.

[28] Bangcheng Li and Yifan Zhou. Multi-component maintenance optimization: An approach combining genetic algorithm and multiagent reinforcement learning. In *2020 Global Reliability and Prognostics and Health Management (PHM-Shanghai)*, pages 1–7. IEEE, 2020.

[29] Han Li, Wei Zhao, Yuxi Zhang, and Enrico Zio. Remaining useful life prediction using multi-scale deep convolutional neural network. *Applied Soft Computing*, 89:106113, 2020.

[30] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018.

[31] Zhen Li, Shisheng Zhong, and Lin Lin. An aero-engine life-cycle maintenance policy optimization algorithm: Reinforcement learning approach. *Chinese Journal of Aeronautics*, 32(9):2133–2150, 2019.

[32] Zhixiong Li, Dazhong Wu, Chao Hu, and Janis Terpenny. An ensemble learning-based prognostic approach with degradation-dependent weights for remaining useful life prediction. *Reliability Engineering & System Safety*, 184:110–122, 2019.

[33] Yu-Lin Liao, Che-Cheng Kuo, and Ya-Fu Peng. Prediction and identification using recurrent wavelet-based cerebellar model articulation controller neural networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–6. IEEE, 2010.

[34] Jason Lines, Sarah Taylor, and Anthony Bagnall. Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles. *ACM transactions on knowledge discovery from data*, 12(5), 2018.

[35] Lu Liu, Xiao Song, and Zhetao Zhou. Aircraft engine remaining useful life estimation via a double attention-based data-driven architecture. *Reliability Engineering & System Safety*, 221:108330, 2022.

[36] Yilai Liu and Xinbo Qian. Selective maintenance optimization with stochastic break duration based on reinforcement learning. *Eksploatacja i Niezawodność*, 24(4), 2022.

[37] Yingchao Liu, Xiaofeng Hu, and Wenjuan Zhang. Remaining useful life prediction based on health index similarity. *Reliability Engineering & System Safety*, 185:502–510, 2019.

[38] Yu Liu, Yiming Chen, and Tao Jiang. Dynamic selective maintenance optimization for multi-state systems over a finite horizon: A deep reinforcement learning approach. *European Journal of Operational Research*, 283(1):166–181, 2020.

[39] Zahra Mahmoodzadeh, Keo-Yuan Wu, Enrique Lopez Droguett, and Ali Mosleh. Condition-based maintenance with reinforcement learning for dry gas pipeline subject to internal corrosion. *Sensors*, 20(19):5708, 2020.

[40] Pankaj Malhotra, Vishnu Tv, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder. *arXiv preprint arXiv:1608.06154*, 2016.

[41] Yu Mo, Qianhui Wu, Xiu Li, and Biqing Huang. Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit. *Journal of Intelligent Manufacturing*, 32:1997–2006, 2021.

[42] Oluwaseyi Ogunfowora and Homayoun Najjaran. Reinforcement and deep reinforcement learning-based solutions for machine maintenance planning, scheduling policies, and optimization. *Journal of Manufacturing Systems*, 70:244–263, 2023.

[43] Kevin Shen Hoong Ong, Wenbo Wang, Nguyen Quang Hieu, Dusit Niyato, and Thomas Friedrichs. Predictive maintenance model for iiot-based manufacturing: A transferable deep reinforcement learning approach. *IEEE Internet of Things Journal*, 2022.

[44] Kevin Shen Hoong Ong, Wenbo Wang, Dusit Niyato, and Thomas Friedrichs. Deep-reinforcement-learning-based predictive maintenance model for effective resource management in industrial iot. *IEEE Internet of Things Journal*, 9(7):5173–5188, 2021.

[45] Panagiotis D Paraschos, Georgios K Koulinas, and Dimitrios E Koulouriotis. Reinforcement learning for combined production-maintenance and quality control of a manufacturing system with deterioration failures. *Journal of Manufacturing Systems*, 56:470–483, 2020.

[46] Panagiotis D Paraschos, Georgios K Koulinas, and Dimitrios E Koulouriotis. Parametric and reinforcement learning control for degrading multi-stage systems. *Procedia Manufacturing*, 55:401–408, 2021.

[47] Shenglin Peng et al. Reinforcement learning with gaussian processes for condition-based maintenance. *Computers & Industrial Engineering*, 158:107321, 2021.

[48] Luca Pinciroli, Piero Baraldi, Guido Ballabio, Michele Compare, and Enrico Zio. Deep reinforcement learning based on proximal policy optimization for the maintenance of a wind farm with multiple crews. *Energies*, 14(20):6743, 2021.

[49] Luca Pinciroli, Piero Baraldi, Guido Ballabio, Michele Compare, and Enrico Zio. Optimization of the operation and maintenance of renewable energy systems by deep reinforcement learning. *Renewable Energy*, 183:752–763, 2022.

[50] Hasan Rasay, Farnoosh Naderkhani, and Amir Mohammad Golmohammadi. Reinforcement learning based on stochastic dynamic programming for condition-based maintenance of deteriorating production processes. In *2022 IEEE Inter-*

national Conference on Prognostics and Health Management (ICPHM), pages 17–24. IEEE, 2022.

[51] Jorge Ribeiro, Pedro Andrade, Manuel Carvalho, Catarina Silva, Bernardete Ribeiro, and Licínio Roque. Playful probes for design interaction with machine learning: A tool for aircraft condition-based maintenance planning and visualisation. *Mathematics*, 10(9):1604, 2022.

[52] Roberto Rocchetta, L Bellani, M Compare, Enrico Zio, and E Patelli. A reinforcement learning framework for optimal operation and maintenance of power grids. *Applied energy*, 241:291–301, 2019.

[53] Marcelo Luis Ruiz Rodríguez, Sylvain Kubler, Andrea de Giorgio, Maxime Cordy, Jérémy Robert, and Yves Le Traon. Multi-agent deep reinforcement learning based predictive maintenance on parallel machines. *Robotics and Computer-Integrated Manufacturing*, 78:102406, 2022.

[54] Pegah Rokhforoz, Mina Montazeri, and Olga Fink. Safe multi-agent deep reinforcement learning for joint bidding and maintenance scheduling of generation units. *Reliability Engineering & System Safety*, page 109081, 2023.

[55] JH Ruan, ZX Wang, Felix TS Chan, S Patnaik, and Manoj Kumar Tiwari. A reinforcement learning-based algorithm for the aircraft maintenance routing problem. *Expert Systems with Applications*, 169:114399, 2021.

[56] Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Database Systems for Advanced Applications: 21st International Conference, DASFAA 2016, Dallas, TX, USA, April 16-19, 2016, Proceedings, Part I 21*, pages 214–228. Springer, 2016.

[57] Sébastien Schwartz, Juan José Montero Jiménez, Rob Vingerhoeds, and Michel Salaün. An unsupervised approach for health index building and for similarity-based remaining useful life estimation. *Computers in Industry*, 141:103716, 2022.

[58] Chandran Senthil and Ranjitharamasamy Sudhakara Pandian. Proactive maintenance model using reinforcement learning algorithm in rubber industry. *Processes*, 10(2):371, 2022.

[59] Ahmed Shifaz, Charlotte Pelletier, François Petitjean, and Geoffrey I Webb. Ts-chief: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, 34(3):742–775, 2020.

[60] Jianyu Su, Jing Huang, Stephen Adams, Qing Chang, and Peter A Beling. Deep multi-agent reinforcement learning for multi-level preventive maintenance in manufacturing systems. *Expert Systems with Applications*, 192:116323, 2022.

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[62] Kim Verbert, Bart De Schutter, and R Babuška. Timely condition-based maintenance planning for multi-component systems. *Reliability Engineering & System Safety*, 159:310–321, 2017.

[63] Cunsong Wang, Zhenghong Zhu, Ningyun Lu, Yuehua Cheng, and Bin Jiang. A data-driven degradation prognostic strategy for aero-engine under various operational conditions. *Neurocomputing*, 462:195–207, 2021.

[64] Hongfeng Wang, Qi Yan, and Shuzhu Zhang. Integrated scheduling and flexible maintenance in deteriorating multi-state single machine system using a reinforcement learning approach. *Advanced Engineering Informatics*, 49:101339, 2021.

[65] Tianyi Wang, Jianbo Yu, David Siegel, and Jay Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE, 2008.

[66] Tianyi Wang, Jianbo Yu, David Siegel, and Jay Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE, 2008.

[67] Wenbin Wang. An overview of the recent advances in delay-time-based maintenance modelling. *Reliability Engineering & System Safety*, 106:165–178, 2012.

[68] Xiao Wang, Chao Qi, Hongwei Wang, Qingmin Si, and Guowei Zhang. Resilience-driven maintenance scheduling methodology for multi-agent production line sys-

tem. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 614–619. IEEE, 2015.

[69] Xiao Wang, Hongwei Wang, and Chao Qi. Multi-agent reinforcement learning based maintenance policy for a resource constrained flow line system. *Journal of Intelligent Manufacturing*, 27(2):325–333, 2016.

[70] Xiao Wang, Guowei Zhang, Yongqiang Li, and Na Qu. A heuristically accelerated reinforcement learning method for maintenance policy of an assembly line. *Journal of Industrial and Management Optimization*, 2022.

[71] Kevin Wesendrup and Bernd Hellingrath. Post-prognostics demand management, production, spare parts and maintenance planning for a single-machine system using reinforcement learning. *Computers & Industrial Engineering*, 179:109216, 2023.

[72] Qilong Wu, Qiang Feng, Yi Ren, Quan Xia, Zili Wang, and Baoping Cai. An intelligent preventive maintenance method based on reinforcement learning for battery energy storage systems. *IEEE Transactions on Industrial Informatics*, 17(12):8254–8264, 2021.

[73] AS Xanthopoulos, Athanasios Kiatipis, Dimitris E Koulouriotis, and Sepp Stieger. Reinforcement learning-based and parametric production-maintenance control policies for a deteriorating manufacturing system. *IEEE Access*, 6:576–588, 2017.

[74] Dan Xu, Xiaoqi Xiao, Jie Liu, and Shaobo Sui. Spatio-temporal degradation modeling and remaining useful life prediction under multiple operating conditions based on attention mechanism and deep learning. *Reliability Engineering & System Safety*, 229:108886, 2023.

[75] Qi Yan, Hongfeng Wang, and Fang Wu. Digital twin-enabled dynamic scheduling with preventive maintenance using a double-layer q-learning algorithm. *Computers & Operations Research*, 144:105823, 2022.

[76] Qi Yan, Wenbin Wu, and Hongfeng Wang. Deep reinforcement learning for distributed flow shop scheduling with flexible maintenance. *Machines*, 10(3):210, 2022.

[77] Hongbing Yang, Wenchao Li, and Bin Wang. Joint optimization of preventive maintenance and production scheduling for multi-state production systems based on reinforcement learning. *Reliability Engineering & System Safety*, 214:107713, 2021.

[78] Yanhua Yang and Ligang Yao. Optimization method of power equipment maintenance plan decision-making based on deep reinforcement learning. *Mathematical Problems in Engineering*, 2021, 2021.

[79] Zhuang Ye and Jianbo Yu. Health condition monitoring of machines based on long short-term memory convolutional autoencoder. *Applied Soft Computing*, 107:107379, 2021.

[80] Nooshin Yousefi, Stamatis Tsianikas, and David W Coit. Reinforcement learning for dynamic condition-based maintenance of a system with individually repairable components. *Quality Engineering*, 32(3):388–408, 2020.

[81] Nooshin Yousefi, Stamatis Tsianikas, and David W Coit. Dynamic maintenance model for a repairable multi-component system using deep reinforcement learning. *Quality Engineering*, 34(1):16–35, 2022.

[82] Wennian Yu, II Yong Kim, and Chris Mechefske. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. *Mechanical Systems and Signal Processing*, 129:764–780, 2019.

[83] Wennian Yu, II Yong Kim, and Chris Mechefske. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. *Mechanical Systems and Signal Processing*, 129:764–780, 2019.

[84] Wennian Yu, II Yong Kim, and Chris Mechefske. An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering & System Safety*, 199:106926, 2020.

[85] Wennian Yu, II Yong Kim, and Chris Mechefske. An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering & System Safety*, 199:106926, 2020.

[86] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. A transformer-based framework for multivariate time series

representation learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2114–2124, 2021.

[87] Chen Zhang, Yan-Fu Li, and David W Coit. Deep reinforcement learning for dynamic opportunistic maintenance of multi-component systems with load sharing. *IEEE Transactions on Reliability*, 2022.

[88] Chong Zhang, Pin Lim, A Kai Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE transactions on neural networks and learning systems*, 28(10):2306–2318, 2016.

[89] Jiusi Zhang, Yuchen Jiang, Shimeng Wu, Xiang Li, Hao Luo, and Shen Yin. Prediction of remaining useful life based on bidirectional gated recurrent unit with temporal self-attention mechanism. *Reliability Engineering & System Safety*, 221:108297, 2022.

[90] Nailong Zhang and Wujun Si. Deep reinforcement learning for condition-based maintenance planning of multi-component systems under dependent competing risks. *Reliability Engineering & System Safety*, 203:107094, 2020.

[91] Ping Zhang, Xiaoyan Zhu, and Min Xie. A model-based reinforcement learning approach for maintenance optimization of degrading systems in a large state space. *Computers & Industrial Engineering*, 161:107622, 2021.

[92] Yiming Zhang, Dingyang Zhang, Xiaoge Zhang, Lemiao Qiu, Felix TS Chan, Zili Wang, and Shuyou Zhang. Guided probabilistic reinforcement learning for sampling-efficient maintenance scheduling of multi-component system. *Applied Mathematical Modelling*, 119:677–697, 2023.

[93] Fang Jian Zhao and Yifan Zhou. Wind farm maintenance scheduling using soft actor-critic deep reinforcement learning. In *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, pages 1–6. IEEE, 2022.

[94] Yunfei Zhao and Carol Smidts. Reinforcement learning for adaptive maintenance policy optimization under imperfect knowledge of the system degradation model and partial observability of system states. *Reliability Engineering & System Safety*, 224:108541, 2022.

[95] Shuai Zheng, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta. Long short-term memory network for remaining useful life estimation. In *2017 IEEE international conference on prognostics and health management (ICPHM)*, pages 88–95. IEEE, 2017.

[96] Yifan Zhou, Bangcheng Li, and Tian Ran Lin. Maintenance optimisation of multicomponent systems using hierarchical coordinated reinforcement learning. *Reliability Engineering & System Safety*, 217:108078, 2022.