Machine Health Monitoring for Cyber-Physical Systems

by

Gürkan Aydemir

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2012

M.Sc., Electrical and Electronics Engineering, Boğaziçi University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2020

ACKNOWLEDGEMENTS

First and foremost, I would like to praise and thank Allah, the most merciful, for giving me the health, desire, ability and opportunity to start this research, to maintain and complete it satisfactorily with a great supervisor and in a comfortable environment. Peace be upon the most beloved one and all loved ones.

First of all, I would like to express my profound thanks to my supervisor Prof. Burak Acar for his valuable support and guidance throughout my PhD journey. His priceless advices, insightful discussions and generous assistance helped me in my research and writing of this thesis. It is a great pleasure to be with his guidance.

I would also like to thank Ascenix team, Deniz Eroglu, Kutluhan Erol, Abdülkadir Yazıcı and Alper Yılmaz, for their fruitful discussion and assistance. I also want to acknowledge all of the Volumetric Analysis and Visualization Group (VAVlab) members for their friendship and providing a peaceful studying environment. I would like to thank the faculty members and staff of Boğaziçi University, Electrical and Electronics Engineering department for their support.

I want to acknowledge my thesis committee, Prof. Cem Ersoy and Murat Saraçlar for their valuable discussion and feedbacks during the progress. I would also like to thank Prof. Engin Erzin and Assoc. Prof. Ayhan Bozkurt for accepting to join my defense, their helpful comments and criticism over the thesis.

I am highly indebted to Assoc. Prof. Kamran Paynabar and Prof. Nagi Gebraeel for providing a research opportunity in Georgia Institute of Technology. The guidance of Prof. Paynabar presented this thesis with a whole chapter and me with a great experience in the domain. I would like to thank the faculty members and staff of H. Milton Stewart School of Industrial and Systems Engineering in Georgia Institute of Technology for their hospitality.

This thesis is partially supported by Scientific and Technical Research Council of Turkey (TUBITAK) BIDEB 2211-E Scholarhip program. I would like to acknowledge Bursa Technical University, Faculty of Engineering and Natural Sciences for the financial support by employing me during this thesis. I would also like to thank Türkiye Fulbright Eğitim Komisyonu for their financial support in my research at Georgia Institute of Technology, USA.

I would like to express my sincere thanks to all of my friends, especially to Akif Fidanoğlu and Mustafa Yeşilyurt for their emotional support during this hard period.

My gratitude would be meaningless without thanking the biggest source of my motivation, my family. I would like to thank my father Yüksel Aydemir and my mother Zikriye Aydemir for their invaluable support through my whole life. I want to thank my sister, Büşra for her delicious cookies during the thesis writing period and being always nice to me. I want to express my gratitude and love to my wife Zeynep Hümeyra for her patience and valuable support during this stressful time. My special appreciation goes to two little heroes, my daughter Meryem and my son Abdullah Yüksel, even if they were the biggest distractions in the research and thesis writing processes. Without them, it would be worthless.

ABSTRACT

Machine Health Monitoring for Cyber-Physical Systems

Estimating the failure time of the machinery that are used in the production is crucial to achieve an efficient maintenance in Industry 4.0 era. Remaining useful life (RUL) is the term that refers to the length of time in terms of the raw time intervals or usage that a machine will continue to operate before it requires a repair or replacement. Machine learning (ML), especially deep learning, provides industry practitioners with efficient tools for estimating the RUL. However, ML is far from being fully utilized, since domain knowledge is generally ignored in current studies. This thesis focuses on three main domain specific problems in machine condition monitoring to improve the performance of ML based RUL estimation. First, RUL is ill-defined during the healthy operation period of the machinery, hence enforcing ML with respect to a fictitious true RUL during these periods adversely affects the overall RUL estimation accuracy. In this thesis, a system level anomaly detection triggered RUL estimation method is proposed to detect degradation onset point in sensor data to prevent ML models to estimate RUL in this period, and hence to increase the accuracy. Secondly, the operating conditions of the machines affect their degradation pattern and related sensor measurements. Thus, the accuracy of ML based RUL estimation models decreases when the machinery operate in varying conditions. A siamese neural network based operating condition-invariant feature extraction method is introduced to alleviate this problem. These two approaches are verified using a benchmark turbofan engine degradation data. Lastly, most of the ML models suffer from lack of data in RUL estimation. If the data are high dimensional such as image, profile, etc., the problem becomes more challenging. Two deep learning architectures are proposed to resolve curse of dimensionality in case of degradation data scarcity. Efficiency of the proposed models is demonstrated with an infrared image data.

ÖZET

Siber-Fiziksel Sistemler için Makine Sağlığı Gözleme

Endüstri 4.0 çağında, üretimde kullanılan makinelerin arıza zamanının tahmin edilmesi, verimli bir bakım elde etmek için çok önemlidir. Kalan faydalı ömür (RUL), bir makinenin onarım veya değiştirme gerektirmeden çalışmaya devam edeceği ham zaman aralıkları veya kullanım açısından zaman uzunluğunu ifade eden terimdir. Makine öğrenimi (ML), özellikle derin öğrenme, endüstri uygulayıcılarına RUL'yi tahmin etmek için etkili araçlar sağlar. Bununla birlikte, mevcut çalışmalarda alana özgün bilgiler genellikle göz ardı edildiği için ML'den tam olarak faydalanılamamıştır. Bu tez, ML tabanlı RUL tahmininin performansını artırmak için makine durumu izlemedeki alana özgü üç ana soruna odaklanmaktadır. Birincisi, RUL, makinelerin sağlıklı çalışma döneminde iyi tanımlanmamıştır, bu nedenle bu dönemde varsayılan RUL ile ilgili olarak ML'yi zorlamak, genel RUL tahmin doğruluğunu olumsuz yönde etkiler. Bu tezde, ML modellerinin bu dönemde RUL'u tahmin etmesini önlemek ve böylece doğruluğunu artırmak için sensör verilerinde bozulma başlangıç noktasını tespit etmek için sistem düzeyinde anomali saptamasıyla tetiklenmiş RUL tahmin yöntemi önerilmiştir. İkinci olarak, makinelerin çalışma koşulları bozulma düzenlerini ve ilgili sensör ölçümlerini etkiler. Bu nedenle, makine değişik koşullarda çalıştığında ML tabanlı RUL tahmin modellerinin doğruluğu azalır. Bu sorunu hafifletmek için siyam sinirsel ağ tabanlı çalışma koşulu-değişimsiz öznitelik çıkarma yöntemi tanıtılmıştır. Bu iki yaklaşım, bir referans turbofan motor bozulması verisi kullanılarak doğrulanmıştır. Son olarak, ML modellerinin çoğu RUL tahmininde veri eksikliğinden zarar görür. Veriler görüntü, profil vb. gibi yüksek boyutluysa, sorun daha da zor hale gelir. Bozulma veri kıtlığı durumunda boyutluluk belasından kurtulmak için iki derin öğrenme mimarisi önerilmektedir. Önerilen modellerin verimliliği bir kızılötesi görüntü verisiyle gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

k	Reference parameter of cumulative sum control chart
C^L	Lower cumulative sum
C^U	Upper cumulative sum
h	Control limit for anomaly detection, Hidden node
I	Image matrix
K	Kernel matrix
L	Window size
L	Loss
$\hat{L}$	Maximum likelihood
$\mathcal{N}$	Normal distribution
y	Binary indicator in contrastive loss
$u[.]$	Unit step function
$\mathbf{U}$	Trainable long short-term memory cell parameters
w_i	Gaussian mixture model's mixture weight
$\mathbf{W}$	Trainable feed-forward neural network parameters
α	Degradation coefficient, diffusion parameter
Δt	Time difference
μ	Mean
σ	Standard deviation
$\sigma_g(.)$	Activation function
Σ	Covariance matrix
τ	Degradation onset
Φ	Gaussian process coefficient
∂	Partial derivative

LIST OF ACRONYMS/ABBREVIATIONS

a.u.	arbitrary unit
ARL	Average Run Length
AT-RUL	Anomaly Triggered Remaining Useful Life
BIC	Bayesian information criterion
BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
CUSUM	Cumulative Sum
D	Dimension
GPU	Graphics processing unit
GMM	Gaussian Mixture Model
LR	Linear Regression
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multilayer Perceptron
MPCA	Multi-linear Principal Component Analysis
OCI	Operating Condition Invariant
PCA	Principal Component Analysis
PHM	Prognostics and Health Management
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root-Mean-Square Error
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
SNN	Siamese Neural Network
tanh	Hyperbolic Tangent
TTF	Time to Failure

1. INTRODUCTION

1.1. Sensor Data analytics in Industry 4.0

Industry 4.0 is the name of the concept for the high digitization of the industrial environment, and depicts the new industrial age. It can simply be defined as the evolution of industry from embedded systems to cyber-physical systems [1]. Cyber–physical systems refer to the physical or engineered entities which are monitored, organized and controlled by a computing core and connected through the internet [2]. This concept is also named as Industrial Internet of Things (IIoT) under the more general concept of Internet of Things (IoT) [3]. It is expected that Industry 4.0 will totally transform the production, logistics and service processes.

Industry 4.0 is enabled by the developments in three technologies [3]. Firstly, the improvements in the communication technology such as higher bandwidths and small wireless devices enable the industrial components to be connected to the internet. Secondly, the low-cost wired and wireless sensors can collect the data, which can be utilized for creating a more efficient and traceable production environment. Lastly, the enhancements in the computing technology such as cloud and edge computing provides a flexible and low cost tools for transforming the sensor data into the useful information.

Industry 4.0 is a broad subject that covers a lot of different concepts, however, for the sake of simplicity it is reviewed under four main headings, namely smart manufacturing, smart products, smart supply chain and smart working [4]. The smart manufacturing is the central interest in Industry 4.0. The smart manufacturing efforts can be classified into four main groups: vertical integration, flexibility, efficiency (energy and resource management) and traceability or monitorability. The vertical integration refers to the integration of all hierarchical levels of a factory from the shop floor to the management and the virtualization of all operations by an information and

communication technology (ICT) system [5]. The manufacturing processes can be more transparent, controllable and decision-making becomes faster via vertical integration.

The flexibility is another character of the smart manufacturing and it can be achieved via the modular machines and highly automated production lines. This approach facilitates producing small number of different goods efficiently [6]. Moreover, the modern consumers request unique products, and hence, the industry should be as much as flexible to fulfill these needs and still be profitable [7].

Traceability in a smart manufacturing environment can be achieved by monitoring both the products and production line from the beginning to the end of the production via the sensors [8]. The data analytics may provide necessary tools for forecasting and detecting the machine failures, overloads and product quality problems. Therefore, the production costs can be reduced and the product quality can be increased by taking the necessary maintenance and planning actions [9].

The efficiency is also a central concern in the smart manufacturing. A smart factory monitors the resource/energy suppliers and consumers around itself. Thus, the production can be planned in a more affordable and profitable fashion [10]. This leads to an efficient management of the resources for the individual manufacturers and all industry.

On the other hand, the smart connected products are also crucial for the realization of Industry 4.0. The smart products have four characteristics: monitoring, control, autonomy and optimization [11]. Manufacturers of a smart product can monitor its condition, operation and environment via the sensors on it. It can be controlled via remote commands and/or the software that is installed on it. The collected data from a smart product together with the controllability can be utilized to optimize the operation of the product. This optimization may be in numerous ways, such as the products' maintenance schedules, energy consumption, outputs, etc. These three capabilities of the smart products may enable them to achieve the autonomy, i.e., they can operate

without any human intervention.

In addition to the smart manufacturing and smart product, the smart supply chain is another aim of Industry 4.0. The smart supply chain can also be named as the horizontal integration [12]. It includes technologies that facilitates the real-time information exchange with suppliers and distributors. Moreover, the environmental conditions can also be traced via these platforms and necessary precautions can be taken. The digital platforms that fulfill this fast information exchange enable the manufacturers to optimize their organization. For instance, the warehouse costs can be reduced, the delivery dates can be accurately achieved, the individual logistic demands of the customers can be met, the supply and demand sides can be organized together, etc. [13].

The smart manufacturing is supported by the smart working which aims to increase the productivity of the workers [14]. The remote control of the production line and supporting machinery are provided to the workers by the means of the mobile devices and, the decision-making processes are supported with these instruments. Moreover, the virtual reality technology can increase the speed of the workers' training and the augmented reality devices can aid them in real-time. These two technologies may decrease the product development times by reducing the needs for the physical prototypes. The collaborative robots are also planned to be integrated in the human working environments to combine the flexibility of the human work and the efficiency of the robots [15]. This integration increases the accuracy, quality, reliability, efficiency and flexibility of the manufacturing systems. Furthermore, the health and safety of the human workers can be improved.

All of these four components of Industry 4.0 are closely related to the sensor networks and the big data created by the sensors [16]. However, the raw industrial sensory data should be analyzed with the advanced data analytics' tools in order to be transformed into the information which can provide added value. Therefore, the sensor analytics or sensor data analytics play a crucial role in the implementation of

Industry 4.0 concepts, and it is necessary to develop advanced analysis tools for the sensory data.

The industrial data analytics refers to the extraction of the useful information from the industrial big data to improve the reasoning and decision making processes [17]. There has been numerous studies that investigate the potential of the data analytics in the industry [18]. The impact of analytics in the industry can be reviewed under five phases of the production, namely product design, premanufacturing (procurement and organization), manufacturing process, distribution (and sales) and product monitoring [19].

The product design is planned to become more customer oriented and data driven in Industry 4.0. There are numerous types of data that can reflect and affect the user demands such as the user data from the e-commerce and social network platforms, the public data from the governmental organizations and service providers, the sensor data that provide the users' location, environmental conditions and usage profile. The data analytics may demonstrate the needs of the customers and the products may be designed accordingly [20]. Moreover, it may accelerate and optimize the product design processes.

The premanufacturing (procurement and organization) actions are very critical for the cost, production time and quality of a product. The environmental sensor data such as weather, road conditions, etc. and the suppliers' behavioral data can be analyzed through the advanced analytics tools. Then, the procurement and logistics of the materials that will be used in the manufacturing may be optimized based on these analyses [21]. For instance, the suppliers' may be classified by their prices, supply time and quality. The supply decisions are taken accordingly, then the costs and supply time may be reduced, and the quality of the materials can be increased. The environmental and road conditions can be utilized to increase the accuracy of the production schedules.

The industrial data analytics will also change the distribution, marketing and

sales of the products. The shopping habits of the consumers can be inferred from their usage and sensor data. The marketing and sales channels are selected and optimized based on the buyers' preferences and be more individualized [22]. Moreover, the distribution channels and processes can be more efficient if the decisions are made according to the customer behaviors. The delivery dates may be scheduled by considering the environmental conditions and may be more accurate. The logistics can be planned to reduce the delivery costs. In addition, the warehouse locations, sizes and types can be optimized.

The products in Industry 4.0 are also expected to have the sensors and internet connection. This may lead to selling services with the product instead of selling only product itself and the manufacturers may transform into the product-service providers [23]. The products are planned to be continuously monitored via the installed sensors. The data analytics help the manufacturers to optimize the operation parameters considering the environmental conditions and needs of the consumers by the remote control or installed software. Moreover, the condition of the products is assessed and, as a result, the necessary maintenance actions may be taken [24].

The manufacturing processes play vital roles in the realization of industry 4.0 aims. The term "manufacturing process" is used here to refer to all of the processes in production from the raw materials to the end product. The data analytics can improve the manufacturing processes in various ways such as the product quality, machine health management, efficiency and safety of the human workers, the energy management, etc. The product quality can be monitored at each stage of the production, then the defects can be immediately detected and the root causes can be corrected. Moreover, the machines in the production lines can be continuously traced via the sensor data analytics and their health conditions can be managed efficiently to prevent the unplanned downtimes and increase the productivity [25]. The human workers in the production environment can be also tracked via the cameras and sensors. The analytics can improve their safety and performance. Furthermore, the environmental conditions can be inferred via the sensor data in order to provide the energy savings

and stable production environment. In addition, the manufacturing can be organized by the energy prices on the daily basis and further savings can be achieved.

A typical data analytics' application can be divided into three phases: descriptive/predictive/prescriptive modeling, generation of alerts and action triggering, and reporting. The Descriptive/Predictive/Prescriptive modeling refers to the data preparation, explanation of the collected data and estimating the future behavior. It includes the data processing tools like classification, regression, summary statistics, pattern recognition, machine learning models, etc. [26]. The outputs of these models facilitate the understanding of the trends in the data and estimation of the future outcomes. The information that is generated by the descriptive/predictive/prescriptive modeling can be utilized to generate the alerts and trigger the necessary actions for the manufacturing management. This process can be made automatically or allow the human intervention. In the last stage, the performance of the applied models is reported and assessed, then, the necessary corrective actions are taken.

An additional stage, namely complex event processing (CEP), has been added to these three phases in [27]. CEP is similar to the descriptive/predictive/ prescriptive modeling, however, instead of handling the raw data, it processes the structured data which is a sequence of the predefined events. This approach has several advantages over handling the analytics only in one stage. First, the descriptive/predictive/prescriptive analysis on the raw data is generally computation intensive, and it may not be possible to implement in real time. CEP can decrease the computation time by dividing the data analytics in the sub-steps such as the detection with the summary statistics, the analysis of the occurrence rate of these events, etc. In addition to the computational efficiency, CEP can decrease the amount of data will be stored by limiting the storage only with the abnormal data.

There are various sources of the data which can be utilized via the industrial data analytics such as product, materials, prices, e–commerce, management, finance, sensor data, etc [9]. Among them the sensor data is of the special interest since continuous

monitoring with the sensors is recently started. Moreover, it may yield huge profits for manufacturers as mentioned above. However, the sensor data analytics have several serious challenges.

1.2. Data Acquisition, Algorithmic and Implementation Challenges in Industrial Sensor Analytics

The sensor data analytics is one of the key concepts to achieve smart factory. A typical application consists of three components, a data acquisition tool, an algorithm to analyze the data and a hardware to implement the algorithm. However, the unique structure of manufacturing environment imposes several challenges on data acquisition. The encountered problems in data acquisition affect the structure and quality of the data, and hence the algorithms should have several properties to handle the variety and veracity of the collected data. In addition, size of the collected data and the number of components to be monitored may induce various hardware difficulties to implement the algorithms. This section will touch on the challenges that are related to each component one by one.

There are several obstacles that prevent continuous acquisition of sensor data from the machinery in manufacturing industry. The first problem arises from the characteristics of the sensor data that are utilized in predictive maintenance applications. Vibration, image, current, temperature, sound, load, speed and pressure sensors are the most commonly used sensors to monitor industrial machinery. The information content in these types of sensor data increases with the increase in the sampling rate. However, high sampling rates increase the cost of data collection, storage and transfer, and hence this trade-off discourages the maintenance practitioners to implement high sampling rate-data acquisition set-ups in their factories. Moreover, application specific denoising and filtering schemes should be developed for proper data collection to overcome this issue which further increases data acquisition costs.

On the other hand, researchers require data that cover all life-span of the machin-

ery to model the degradation and failures/faults in the machinery. However, in many industries, complete failure of the machinery is not allowed, and the maintenance actions are taken before the complete failure, i.e., preventive maintenance strategies are already employed. Therefore, in general, it is not possible to obtain complete degradation and failure data on-site. These strategies are less efficient than the predictive maintenance, however, yet effective in preventing the costly downtimes. Furthermore, it is very costly to simulate machine operation and degradation in laboratory environment. Although one may show advantages of predictive maintenance over classical approaches using laboratory simulations, it is hard to claim that the experiments reflect the factory environment in full.

In data analytics part, the main problem is that scarcity of the complete degradation data hinders the training and usage of the state-of-art data-driven models. The advantage of the predictive maintenance arises from the estimation of the failure points using the sensor data. If the degradation can not be modeled, the accurate estimations can not be obtained and, hence the efficient maintenance plans can not be achieved. Therefore, the analytics tools that will be used in the predictive maintenance applications should have two properties to overcome the scarcity of the degradation data. First, they are needed to be trainable with the limited number of degradation samples. Second, they should have the capability of capturing information from the censored samples, i.e., the samples that have some degradation, but do not have complete failure.

The second problem is the variety of the sensor data which stems from the environmental and operational conditions. Data from the several sensors such as vibration, current, and sound are highly sensitive to operational and environmental conditions which also negatively affects the performance of the state-of-art analysis methods. This can be prevented by collecting data as profiles, which means collecting data at the special phases of the operation. However, the indicators of the faults can be observed only in some operating conditions which one can miss if the data is collected using the profiling approach. Hence, a good data analytics tool should be robust to the different operational and environmental conditions.

Implementation of analysis methods for the sensor data that is collected at high sampling rates requires a considerable amount of computational power. There is a challenging trade off between using cloud computing tools or hardware onsite for the implementation. The cost of cloud computation is far less than the one onsite, however, it requires high data rate which is also costly. On the other hand, onsite computational tools are costly but they do not require internet connection. The developments in communication technology and edge computing tools ease the implementation of predictive maintenance. However, it is still far away from continuously analyzing all of the collected sensor data. Therefore, many practitioners prefer to analyze the periodically sampled data instead of all. By doing this, the sampled data can be analyzed both onsite by edge computing devices or on cloud by transferring it to cloud storage. Moreover, there is in general no abrupt change in the health condition of the machinery, and the redundancy in the sensor data is avoided by periodically sampling.

There is a drawback of periodically sampling the sensor data. The indicators of a fault can show itself as a sequence of randomly distributed events. If these events are rare in the initial phase of the degradation, they can be missed. To prevent this, one can use to sample the sensor data according to the information content/novelties that it consists of and, hence preserves the critical information. One also requires computationally lightweight monitoring/analysis tools to achieve such a sampling.

In this thesis, we focus on developing sensor analytics tools to overcome the challenges that are imposed by the nature of industrial sensor data. The next section briefly introduces the challenges that are addressed by this thesis in the predictive analytics for industrial machinery.

1.3. Challenges in Predictive Maintenance

Maintenance costs constitute more than 15% of operational budget in industry [28]. Therefore, reducing the maintenance costs is very crucial for the manufacturers. Industry 4.0 offers several opportunities to develop efficient and effective maintenance

strategies and hence, reduce the maintenance costs [29]. The availability of the sensor data, high speed communication and low-cost computational power enable continuous monitoring of machinery and components in a manufacturing environment. Analyses of the sensor data can give a better insight of the current condition and more accurate estimation of the future health state of the engines and components in the production line. The maintenance actions may be scheduled to avoid the unexpected downtimes and to optimize the maintenance expenditure. Moreover, it may also reduce the risks in the critical systems such as aviation, automobiles, health services, etc.

Predictive maintenance is a generic term that contains all methods and technologies that is used to monitor the reliability of a system and estimate the future failures/faults and system risks [30]. Predictive maintenance studies can be categorized into *detection of anomalies*, *fault diagnosis* and *estimation of RUL* [31]. Anomaly detection refers to capturing the state change of the machines and abnormal events. A baseline representation is obtained through modeling the data acquired during the healthy operation under the normal operating conditions. Then, the collected data on test is compared with the baseline data. The anomaly detection problem can be handled both in supervised and unsupervised manners. In the unsupervised anomaly detection, the detector is designed to catch any type of the abnormality with respect to baseline. Whereas in the supervised anomaly detection, only a subset of the abnormalities which correspond to the predefined events are captured.

Diagnosis means revealing the mode or root cause of a fault [32]. It also includes the determination of the size and severity of the fault. In general, it is a supervised classification problem. The type and severity of the fault is classified into the predefined severity and fault classes which are available in the training data.

Prognostics, on the other hand, focuses on predicting the RUL of the machinery [33]. It is in general defined as a regression problem. Evolution of the failures are modeled utilizing the past data. Then, at a given time of operation, the evolution until that time is analyzed via the same model and the RUL is estimated accordingly.

There is a large body of literature on detection, diagnostics and prognostics [34–37]. Nonetheless, the increase of the available sensor data and the computational power accelerated PHM studies, especially the data driven approaches [31, 34]. Deep learning and several statistical learning models such as graphical probabilistic models, convolutional and recurrent neural networks, decision and regression trees, outperform the others in general [38]. However, there are several challenges that negatively affect the performance of learning models:

- The proposed models in the literature focus on one of the three problems, i.e., detection, diagnostics and prognostics. This may prevent full utilization of the algorithms since these problems are nested in most of the cases. More specifically, diagnostics and prognostics models are in general redundant in the healthy period of the machine life cycle. Enforcing them to operate in that period may lead to sub-optimal models and decrease their performance. In addition, they may cause unnecessary computational burden. Therefore, it is also necessary to come up with system level solutions to merge and organize these separate blocks by including the domain knowledge and improve the performance of the individual blocks and overall system.
- The previous studies are far away from reflecting the real cases in industry. Much of the literature did not consider the varying operating and environmental conditions. Therefore, the proposed architectures are not robust to the operating and environmental conditions, which may impose several obstacles in their practical applications.
- The models are very complex, i.e., the number of the parameters that will be learned is high, in order to obtain better detection, diagnosis and prognosis performance. Therefore, the learning models are suitable for the cases where the training sets include relatively high number of samples. However, this condition might not be met in predictive maintenance applications since data acquisition in the complete lifespan of the machinery is very difficult in the field and costly in the laboratory environment. This may cause the failure of popular learning models in the predictive maintenance domain. Moreover, some of the available

degradation data is high dimensional and contains redundant data such as the image and profile data which may impose an extra difficulty in the training of such models. Therefore, it is very critical to design models that are robust to the scarcity of the training data and high dimensionality of the sensor data.

1.4. Problem Statement

This thesis focuses on three main subjects to handle aforementioned challenges in predictive maintenance domain:

In the first part of the thesis, a system level solution is proposed to merge and organize the separate detection and RUL estimation blocks. More specifically, anomaly detection triggered RUL estimation model is proposed to improve the RUL estimation performance of a given RUL estimator. The system consists of two main blocks: The first block is an individualized (machine specific) anomaly detection block which is computationally lightweight and trained in an unsupervised way. The second block is a machine learning based RUL estimator. The machine specific anomaly detection block continuously monitors the sensor data (and/or extracted features from the sensor data). When it detects an anomaly, the anomaly detected point is marked as the degradation onset and RUL estimation is initiated. In the training phase, each degradation data is clipped from the individual degradation onset point and the RUL estimator is trained only on the recordings from the degradation onset to the complete failure. The performance of the RUL estimator that is trained only with the recordings under degradation is expected to be better, since the RUL is ill-defined while the machine is healthy. Moreover, the computational cost is decreased by avoiding RUL estimation in the healthy period.

In the second part of the thesis, a solution is proposed to estimate the RUL of the machinery with varying operating conditions. More specifically, a siamese neural network (SNN) based feature extraction is proposed to obtain an operating condition invariant feature set. The machine degradation is a continuous process, whereas the

machine operating conditions which directly affect the sensor readings are in general discrete and finite. It is assumed that the degradation levels are close in the consecutive time steps over the operating condition change points, and different between the healthy and faulty period of the engines' lifespan recordings. A multilayer perceptron (MLP) is trained using SNN architecture to minimize the change in the feature values between the consecutive time steps over the operating condition change points and maximize the difference between the healthy and faulty periods of the individual engines' lifespan recordings.

In the third part of the thesis, two different solutions are proposed to address a RUL estimation problem with high dimensional data, which is a machine infrared image data, in the case of the limited training set. First, a convolutional long short–term memory network (LSTM) model is proposed. The convolutional part extracts spatial features from the individual images, and the LSTM part is run over the extracted features and captures the temporal information from the sequence. In the second model, an autoencoder is utilized to extract spatial information from the individual images. Encoder part of autoencoder is used as feature extractor and a separate LSTM network regresses the RUL from these features.

The rest of the thesis is organized as follows: Chapter 2 briefly reviews the literature and points out the gaps in the predictive maintenance domain. The anomaly detection triggered RUL estimation model is introduced in Chapter 3. The efficiency of the proposed system is demonstrated using a simulation and C-MAPSS turbofan engine degradation data [39]. Chapter 4 presents SNN based operating condition invariant feature extraction method. The model is verified using the same turbofan engine degradation data [39]. The two deep learning models for image prognostics are introduced in Chapter 5. The performance of the proposed models are demonstrated using a simulation data and infrared degradation image stream data collected from the rotational element thrust bearing [40]. Chapter 6 concludes the thesis with some remarks on the findings and future research directions.

2. BACKGROUND

Maintenance is a critical part of the manufacturing since the invention of the steam engines near 1850s. At first, the common maintenance approach was reactive and it refers to fixing the machine/component when it breaks. However, the cost of the reactive maintenance is high in complex systems since the individual breaks may cause failure of the other components or even the whole system. Moreover, the failures may lead to the unplanned downtimes and hence, the extra costs due to the delays in the delivery [28].

Later in 1950s, the preventive maintenance was proposed to prevent the aforementioned problems. It means maintaining the machines in the predetermined time intervals which are based on the failure frequencies. However, the reliability of the individual components may vary excessively and this may lead to an inefficient maintenance planning [30].

Predictive maintenance was proposed to improve the efficiency of the manufacturing and maintenance systems by estimating the reliability of the machinery in future [31]. It is part of a broader concept which is prognostics and health management (PHM) that includes all the actions in planning of the maintenance. The schema of a typical PHM application is illustrated in Figure 2.1. The data is collected from the individual machines via different types of sensors. The collected sensor data is preprocessed and features that will be analyzed are extracted from it. The condition of the machinery is continuously monitored for the early detection and diagnosis of the machine failures, as well as the prognosis of the future health status of the machinery through the sensor readings. The maintenance actions are planned and scheduled according to the output of the detection, diagnosis and prognostics blocks.

The monitoring/detection, diagnostics and prognostics blocks play vital roles in the predictive maintenance applications and widely studied in the literature [32–34,37].

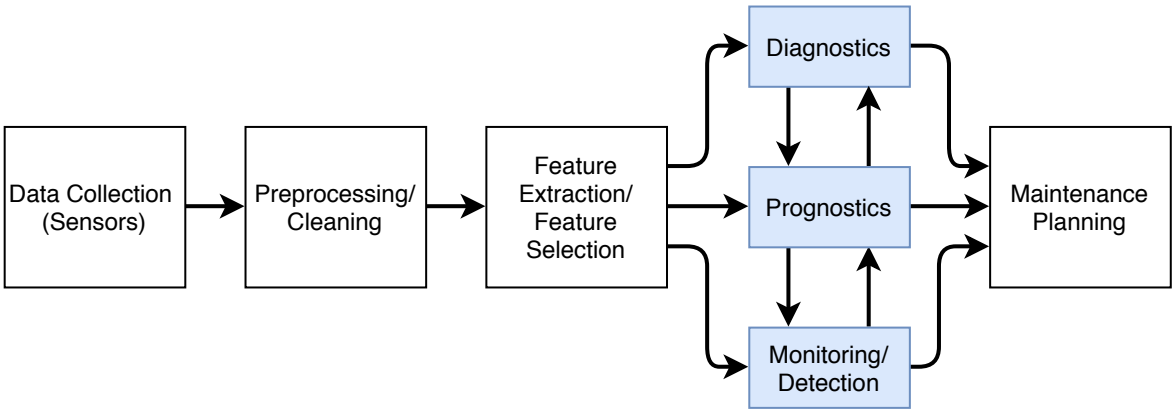


Figure 2.1. The schema of a PHM system. The collected data is transformed into the information through preprocessing, feature extraction, monitoring, diagnostics and prognostics blocks. The extracted information is used for the maintenance planning.

The predictive maintenance approaches can be classified into three groups: the physical model based, knowledge based and data driven methods. In the physical model based methods, a mathematical model of the system is developed. The failure and degradation modes have also corresponding mathematical representations and the sensor data is assessed based on these representations [36]. This approach requires the complete understanding of the machine and its responses under different operational, environmental conditions and failure modes. This technique can not be employed if the mathematical model of the machine/component is unavailable. Knowledge-based methods (a.k.a. expert systems) refer to the models based on the rules that are predefined by the practitioners in the domain. Such models consist of the conditions that define the relation between the failures, degradations and their symptoms [41]. However, the development of those models are expensive and time consuming.

On the other hand, the data-driven approaches learn the behavior of the machine only on its monitored and historical data. The data-driven models do not require the physical model of the system or the system specific knowledge. The development and implementation of the data-driven models are low-cost, quicker and easier as compared to the knowledge and physical model based methods [31, 36]. Therefore, in recent

years there is a growing trend towards to the data-driven models in the predictive maintenance studies.

A large and growing body of literature has studied data-driven models in the predictive maintenance domain. The data-driven models can be classified into two groups: statistical methods and machine learning approaches [31,42]. Statistical models assume an underlying distribution or statistical property for the data. If an observation contradicts with the assumed model, it is taken as an anomaly, failure or degradation. Hypothesis testing, extreme value theory, Gaussian mixture models and nonparametric density estimators are several examples of the statistical approaches that are employed in condition monitoring of the machinery [43–47].

The machine learning methods, on the other hand, are utilized not only in the anomaly detection but also the diagnostics and prognostics tasks [31, 34]. Diagnostics means the identification of the mode of degradation or failure, whereas prognostics refers to estimation of the machine/component status in the future. The popular learning algorithms that are employed in machine health monitoring can be listed as follows: artificial neural networks [38,48], support vector machines [49,50], linear/nonlinear regression [51, 52], Bayesian networks [53, 54], hidden Markov models [55, 56], principle component analysis [57, 58], Kalman filter [59, 60] and particle filter [61, 62].

Although most of the popular machine learning models are employed for the predictive maintenance, namely monitoring, diagnostics and prognostics tasks, the studies are focused only one of those three tasks. However, the monitoring, diagnostics and prognostics should be implemented together in the real life scenarios. A systematic understanding of how monitoring contributes to the prognostics and diagnostics is still lacking. In addition, the industrial machinery operates under varying environmental and operational conditions. The collected sensor data vary with the operating conditions. Moreover, in many cases, the features extracted from the sensor data are specific to operating conditions. Therefore, the performance of machine learning models, especially prognostics are worse for the machinery with varying conditions than for the

machinery with single operating condition. Tackling the adverse effect of varying operating conditions is still missing in the literature. Furthermore, another major problem in predictive maintenance is the scarcity of the annotated sensor data spanning the full operational life of machinery, from the initial installment to the complete failure. To develop a scientific understanding of the failures and degradations in machine health assessment, a complete failure and degradation from the start of the operation to the complete failure is required. However, it is in general expensive and not feasible to obtain the complete failure of the machinery in the field. It is also very expensive to generate the degradation in laboratory environment, and hence, only limited number of experiments is available for the studies. This induces an extra problem to training of the machine learning models. Furthermore, if dimensionality of the data is high such as image and profile data, the problem is more aggravated. In the following sections, the literature that focuses on such problems is briefly reviewed.

2.1. Anomaly Detection for Prognostics

Estimating the remaining useful life (RUL) of a system is the core of predictive maintenance applications [63]. Machine learning (ML) is the leading paradigm in the RUL estimation compared to the model based and statistical approaches. ML methods are capable of learning complex models on rich training data [38]. Random forest [64], support vector regression [65], recurrent [66] and convolutional neural networks [67] are several examples of the machine learning methods that are commonly employed in the RUL estimation.

One of the challenging problems in creating a machine learning model for the RUL estimation is the selection of the ground truth RUL for the available data. By definition, RUL is a linearly decreasing function with time and is equal to zero at the time of failure. However, RUL is ill-defined in the absence of degradation. State-of-art methods address this problem by setting a maximum RUL value [68] and assuming that the RUL is constant beyond this maximum. This corresponds to assuming a piecewise linear true RUL clipped at the maximum allowed RUL. Exploiting computational geometry

tools and using such clipped true RUL training data, Ramasso selected training data similar to the test data in terms of geometric similarity and used them to estimate the test RUL [69]. Zheng et al. proposed to use Long Short-Term Memory (LSTM) networks for the same data set but with a different maximum RUL value [70]. Li et al. utilized deep convolutional neural networks (CNNs) for the RUL estimation again with piecewise linear RUL ground truth [71]. Al-Dulaimi et al. introduced a hybrid model and reported better prediction results over the individual CNN and LSTM [72]. Despite its wide usage in the literature, clipping the true RUL at a predetermined maximum distorts the training of the machine learning algorithms as it implies low variance observations during constant RUL regions. Further the maximum RUL is an arbitrary choice and not necessarily the same for different systems [68, 69, 71].

Nouri et al. [73] employed a statistical method, namely Cumulative Sum (CUSUM) control chart, to detect the transition of a tool from gradual region to failure region. Yuan et al. [74] proposed to use an anomaly detector to set the maximum RUL for each system but they also used constant RUL prior to the anomaly during training. Trying to estimate constant RUL values distorts the RUL estimations in true degradation part because the constant RUL values are not correlated with the sensor measurements but rather the anomaly detection. Furthermore, the proposed support vector machine based anomaly detection method is not machine specific and ignores the environmental conditions and the properties of the individual machines. Thus, it is not suitable for most of the prognostics applications in the industry.

2.2. The Effect of Varying Operational Conditions

Varying operating conditions have an adverse effect on machine health monitoring applications, especially on prognostics, since the sensor measurements and machine/component degradation have a complex response to the varying operating conditions. Deep learning is becoming dominant among the data driven models because of its ability of modeling complex systems and hence is widely studied for the machine prognostics [38]. RUL estimation plays crucial role in prognostics, and so, the RUL

estimation studies that also consider the varying operating conditions are reviewed in this section.

First deep learning models that are employed for the RUL estimation are the multilayer perceptrons (MLPs) [48, 75]. Zhang et al. utilized an ensemble of deep belief networks to estimate RUL [76]. Babu et al. [67] proposed a deep convolutional neural network (DCNN) based approach to estimate RUL with a sliding window approach since CNN networks can not accept the inputs with varying sizes. In [71], the authors proposed a new DCNN architecture on the same data set and improved the performance. On the other hand, a long short-term memory (LSTM) network was reported to have a better performance than DCNN on a benchmark data set [70]. Liao et al. [77] introduced an ensemble of LSTM networks using bootstrap technique to estimate the RUL and its confidence interval. Feature attention method, based on bidirectional gated recurrent units and CNNs, was utilized to estimate RUL [78]. However, none of these models consider the multiple operating conditions which is the case in real-life. Therefore, their performance on multiple operating condition data were far worse than on the data with single operating condition set-up [67, 70, 71, 77, 78]. This hinders their practical applicability.

In general, deep learning models ignore the domain knowledge and approach the problem as a black box from the input to the output. However, as current literature suggests this may affect the performance negatively. As a first approach, increasing the complexity of the models, at the cost of harder training (due to computation load and need for larger data sets), can potentially increase the accuracy of RUL estimation under multiple operating conditions. Huang et al. employed a bidirectional LSTM (biLSTM) network to achieve better RUL estimation performance in multiple operating conditions [79]. Al-Dulaimi et al. proposed a hybrid of DCNN and LSTM networks [72]. Although these studies reported improvement in RUL estimations under varying operating conditions, yet the performance was significantly inferior to non-varying conditions. On the other hand, Al-Dahidi et al. modeled the different operating conditions with hidden Markov models (HMM) [80]. However, the HMMs

assumed discrete and finite number of operating conditions, which is not the case in reality, as for example, the operating conditions (such as load) may change continuously. Li et al. proposed a wiener process model to capture variability in different units (engines/components), yet the variability in operating condition was not considered [81]. Therefore, it may be useful to consider the varying operating conditions and generate an operating condition invariant features from the raw features that are not robust to the varying operating conditions.

2.3. High Dimensionality of the Degradation Data

Several different types of sensors such as temperature, vibration, current, voltage sensors and cameras have been used in machine health monitoring domain [82]. Imaging devices are of special interest in machine condition and structure health monitoring due to the rich information that image data contain. The main superiority of imaging over the other sensors is that their implementation is easier since the cameras are in general non-contact and do not require to be permanently installed. Besides, the image data provides crucial information about the fault status of the photographed object. However, the high dimensionality of the image data together with the scarcity of the industrial degradation data makes it difficult to use it.

Various types of imaging are used in different health monitoring applications. Charge-coupled device images are utilized for evaluating the surface quality [83]. X–ray imaging is proposed to model the evolution of cracks in composite materials [84]. Infrared thermography is used for diverse applications such as civil structure monitoring [85], fatigue dissipation in steel sheets [86], tool condition monitoring [87] and machinery inspection [88, 89]. Even though many researches focused on imaging for condition monitoring/diagnostics, only few of them studied the prognostics using image data.

A spatio-temporal process was employed to model the degradation image streams [90]. The time-to-failure (TTF) of the system was estimated by using the parameters

of this spatio-temporal process. However, the physical degradation can be modeled using a spatio-temporal process under several special conditions. In addition, the model requires a preset threshold for the interested fault which is difficult to select for the image degradation data. In a recent work, a tensor regression method was proposed to estimate TTF of machinery from their degradation image streams [51]. The main problem of their approach is that the tensor decomposition techniques utilized in the study, which are CANDECOMP/PARAFAC (CP) and Tucker, have exponential computational time with the time and dimensions of the images, albeit they solved the high dimensionality problem via these tensor decomposition techniques and reported good performance on a infrared image data.

Numerous studies focused on deep learning models in the predictive maintenance for the diagnostics and prognostics task recently. Stacked denoising autoencoders were used to generate features from the raw sensor measurements for motor fault diagnosis [91, 92]. Convolutional neural networks (CNNs) in 1-D [93] and 2-D [94] were utilized to detect and classify motor faults. The gradual evolution of the faults, which is the key point behind the prognostics, was not considered in these studies. A version of restricted Boltzmann machines was employed in estimation of the remaining useful life (RUL) of the rotating engines [95]. Recurrent neural networks (RNN) were proposed to estimate the RUL of aero engine [74]. The superiority of LSTM over gated recurrent unit (GRU) networks and Vanilla RNN was demonstrated on a benchmark data set. However, high dimensionality was not a problem in the aforementioned studies as opposed to the image data, since the number of measured quantities are comparatively small. The authors offered a model that combines convolutional and LSTM neural networks to diagnose the health status of a system from its profile data which is also high dimensional [96]. In this model, the localized features in short time intervals were extracted by convolutional layers from the raw time series data and bidirectional LSTM layer captured the temporal information in the feature series. However, the prognostics using profile data were not studied. A novel autoencoder architecture based on LSTM was employed to model multi-sensor data in healthy period of the machinery and the reconstruction error was exploited as the health indicator [97]. However, the

dimension of the sensor data was lower than the image and profile data, and thus high dimensionality was not challenging. A deep convolutional neural network (CNN) model was proposed recently to estimate RUL of turbofan engines [71]. The sensor data were restructured using a sliding time window approach, since the CNNs can not operate with varying length sequences. The deep CNN with the sliding time window prevents the complete use of the input sequences by disregarding the inputs beyond the sliding time window.

Deep learning models can be utilized to process degradation image data since they achieve good performance in other domains, such as video segmentation, object tracking, etc. However, the high dimensionality together with the limited data size should have been conveniently addressed.

3. ANOMALY TRIGGERED REMAINING USEFUL LIFE ESTIMATION

3.1. Proposed Approach

Estimating the remaining useful life (RUL) of a machine is the core of the predictive maintenance applications. By definition, the RUL is a linearly decreasing function with time and is equal to zero at the time of failure. In the data-driven RUL estimation models, the model regresses the RUL values from its raw sensor recordings or the extracted features from this raw data. A brand new machine operates in a healthy condition for a long period and hence there is little or no degradation that can be observed in the sensor data in this period.

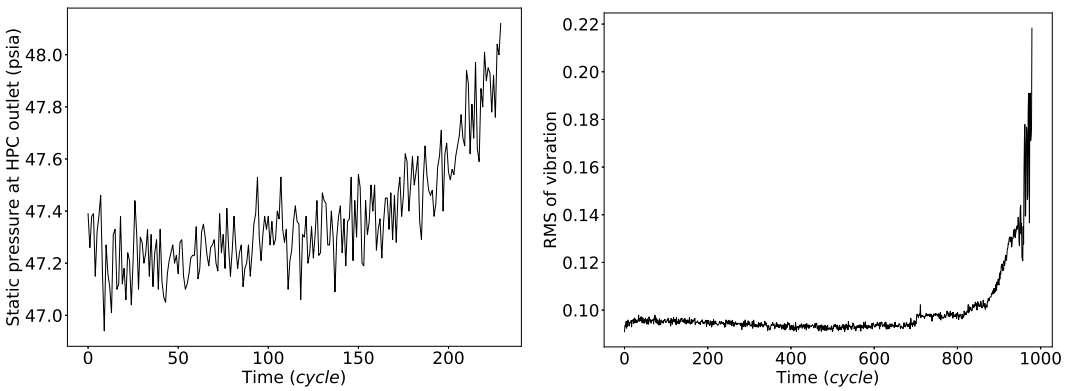


Figure 3.1. Examples of degradation signals. (Left) A pressure sensor measurement from turbofan engine. (Right) RMS of vibration signal from a roller bearing.

Figure 3.1 demonstrates two samples of the degradation signals. The depicted features are critical to assess the health condition of the machinery of interest and hence to estimate the RUL. As seen in Figure 3.1, there is little or no degradation in almost more than half of the overall lifespan. The RUL is ill-defined in such a case and performance of the RUL estimators that are trained on overall lifespan may be adversely affected.

As stated in Section 2.1, this problem was also addressed in the literature and a modified ground truth RUL was employed as shown in Figure 3.2 [68]. In this piecewise RUL ground truth, an arbitrary maximum RUL value is set and the RUL is assumed to be constant beyond this value. Adoption of such an approach is based on the assumption that time to failure of each machine is the similar once degradation is started to be observed. However, the time from degradation onset to the complete failure is different for the different units because of the imperfections in the manufacturing processes and varying operating/environmental conditions. Therefore, this choice of RUL ground truth may not reduce the adverse effect of compelling the RUL estimators to estimate RUL in the healthy period.

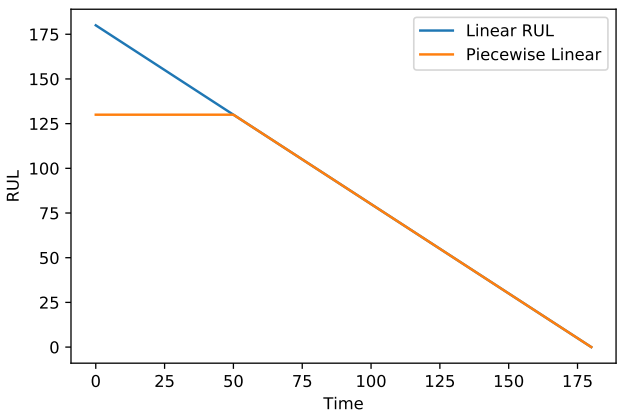


Figure 3.2. Modified piecewise linear RUL ground truth.

To address the aforementioned problems, we propose anomaly detection methodology with which the RUL estimation is initiated [98]. The proposed Anomaly Triggered-RUL (AT-RUL) estimation model increases the accuracy in the critical region, decreases the computational cost by skipping the feature extraction and RUL estimation prior to anomaly detection. The efficiency of the suggested model has been demonstrated using simulation and a popular benchmark data for three different ML models, namely, linear regression (LR), random forest (RF) and deep LSTM network.

Two different anomaly detection algorithms, namely cumulative sum (CUSUM)

control charts and Gaussian mixture models (GMM), are analyzed and compared for the defined problem in the following section. The AT-RUL estimation is presented in Section 3.3. Section 3.4 presents the simulation experiments and reports the results of the proposed method in comparison to the conventional approaches. The performance on the C-MAPSS turbofan engine degradation data is demonstrated in Section 3.5. Section 3.6 discusses the results with the pros and cons of the proposed approach.

3.2. Anomaly Detection Models

3.2.1. Anomaly Detection by Cumulative Sum

Normal distribution is commonly employed to model the data for healthy periods [99]. A significant deviation in the tracked feature from its mean in the healthy period is a good estimate for the onset of degradation. CUSUM control charts offer an efficient way of monitoring mean of a process based on its samples [100].

Lower and upper CUSUMs for a monitored time series, $x[n]$, are defined as,

$$C^L[n] = max(0, -\frac{x[n]-\mu}{\sigma} - k + C^L[n-1]) \quad (3.1)$$

$$C^U[n] = max(0, \frac{x[n]-\mu}{\sigma} - k + C^U[n-1]) \quad (3.2)$$

where μ is the average, σ is the standard deviation of the time series $x[n]$ in healthy period and k is a preset reference value. Lower and upper CUSUMs are initialized as $C^L[0] = C^U[0] = 0$. As soon as, $C^L[n] > h$ or $C^U[n] > h$, for a preset threshold h, a detected anomaly is flagged.

The reference value k and the threshold h are two critical hyper-parameters of CUSUM. Sensitivity to shifts in the mean is determined by k and k should be the half of the mean difference between regions before and after the degradation onset. The control limit h is selected to achieve a target false alarm rate for a given k.

3.2.2. Anomaly Detection by Gaussian Mixture Models

Gaussian mixture models (GMMs) can also be utilized for anomaly detection. The distribution of monitoring statistics, which can be both the raw sensor measurements and the features that are extracted from the raw sensor measurements, is modeled with a GMM. The sample that has low probability in the modeled GMM is taken as the anomaly.

Let $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_{2,\cdots}, \mathbf{x}_N\}$ be a training set of N samples that is collected from an engine in a normal condition where each sample $\mathbf{x}_r$ consists of d features. $\mathbf{x}_r$ is assumed to follow a Gaussian mixture model, i.e., the probability density function of x_r can be represented as a weighted sum of finite Gaussian components as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{K} w_i \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \tag{3.3}$$

where K denotes the number of Gaussian components. $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ and w_i are the mean, the covariance matrix and the mixture weight of the ith Gaussian component, respectively. Probability density of each component is represented as follows:

$$\mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) = \frac{1}{\sqrt{(2\pi)^K \left|\boldsymbol{\Sigma}_i\right|}} \exp\left(-\frac{1}{2}\left(\mathbf{x} - \boldsymbol{\mu}_i\right)^{\mathrm{T}} \boldsymbol{\Sigma}_i^{-1}\left(\mathbf{x} - \boldsymbol{\mu}_i\right)\right) \tag{3.4}$$

To normalize total probability distribution to 1, w_i should satisfy $\sum_{i=1}^{K} w_i = 1$.

Obtaining maximum likelihood solution for the parameters $\boldsymbol{\mu}_i$, $\boldsymbol{\Sigma}_i$ and w_i analytically is almost impossible. Therefore, they are learned using well-known Expectation-Maximization (EM) algorithm in which initial values can be obtained using K-means clustering algorithm. In the expectation step, the posteriori probability of a sample in each Gaussian component is calculated as follows:

$$P_{ri}\left(\mathbf{x}_r|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) = \frac{w_i \mathcal{N}_i\left(\mathbf{x}_r|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)}{\sum_{j=1}^{K} w_j \mathcal{N}_j\left(\mathbf{x}_r|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)} \tag{3.5}$$

Then, the posteriori probability of all samples in each Gaussian component is obtained as:

$$P_i = \sum_{r=1}^{N} P_{ri}\left(\mathbf{x}_r|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right) \tag{3.6}$$

In the maximization step the parameters $\boldsymbol{\mu}_i\boldsymbol{\Sigma}_i$ and w_i are updated according to the following equations:

$$w_i = P_i/N \tag{3.7}$$

$$\mu_i = \frac{\sum_{r=1}^{N}\left(P_{ri}\left(\mathbf{x}_r|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\mathbf{x}_r\right)}{P_i} \tag{3.8}$$

$$\boldsymbol{\Sigma}_i = \frac{\sum_{r=1}^{N}\left(P_r\left(\mathbf{x}_r|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)\left(f_r - \boldsymbol{\mu}_i\right)\left(\mathbf{x}_r - \boldsymbol{\mu}_i\right)^T\right)}{P_i} \tag{3.9}$$

EM algorithm guarantees decrease in log likelihood of GMM algorithm, however, global convergence is not guaranteed. The algorithm stops when improvement in the log-likelihood of GMM is less than a predefined threshold ϵ.

There is no analytical way to select the number of parameters, but one can use Bayesian information criteria to choose K which is defined as:

$$\mathrm{BIC} = \ln(N)K - 2\ln(\widehat{L}) \tag{3.10}$$

where N, K, and $(\widehat{L})$ are the number of samples, the number of components and maximum likelihood value for the model with K components.

The GMM may be used as an anomaly detector in the following manner: First, a GMM distribution is fitted to the healthy data. Then, a target average run length for the false alarms is selected. A probability density threshold that provides the selected false alarm rate is approximated using Monte Carlo simulation methods and regarded as control limit h. Any sample that has probability density value below h is considered as anomaly.

3.2.3. Assessment of CUSUM and GMM for Degradation Detection

The performance of CUSUM and GMM models in degradation detection are evaluated and compared in this section. Normal distribution is commonly employed to model the data for the healthy period [99]. Exponential and linear models are two popular models that are used to model the degradation in industrial machinery [99, 101, 102].

In this section, to compare the performance of modeling healthy state of machinery, two different healthy condition behaviors are simulated. In the first model, the healthy case is assumed to be from a uni-modal normal distribution. Then, it is considered to be from a mixture of two different normal distributions in the second. Linear and exponential degradation signals are added and the detection performance of each model is reported. For the simulation purposes, the signal is generated univariate. The generated signal has the following form

$$\hat{s}[n] = r[n] + u[n-\tau]f(t-\tau) \tag{3.11}$$

where n is an arbitrary time unit (a.u.), $r[n]$ is the additive noise, $u[n]$ is the unit step, τ is the degradation onset point and f is the degradation signal. In the exponential degradation, f is defined as $(e^{\alpha(t-\tau)} - 1)$ and it is $\alpha(t-\tau)$ in linear one.

For the first scenario, the healthy samples were generated as a zero-mean and unit-variance Gaussian noise. Half of the samples were separated to determine anomaly detection threshold, say h. Linear and exponential degradation components were separately added after a 50 $a.u.$s to all of the other samples. A GMM distribution was fitted to the data in healthy period which was 50 $a.u.$s in this case. The control limit h was selected as 0.0044 to provide an average run length for false alarms as $ARL_0 = 370$. CUSUM model's parameters, namely μ and σ, were computed using the data in the first 50 $a.u.$s. Then, three candidate k values, $k = \{0.5, 0.75, 1\}$ which are commonly used in the literature, are selected and the hs are set to $\{4.72, 3.33, 2.30\}$ for each k

respectively to achieve same ARL_0 performance.

Table 3.1. The degradation detection performance of CUSUM and GMM in terms of ARL_1, i.e., speed of detecting degradation onsets when distribution of the healthy data is modeled using unimodal Gaussian distribution.

	Exponential (α)				**Linear (α)**		
MODEL	0.005	0.01	0.05	0.1	0.01	0.05	0.1
CUSUM (k=0.5, h=4.72)	75	47	21	10	55	20	13
CUSUM (k=0.75, h=3.33)	82	51	16	10	60	21	13
CUSUM (k=1, h=2.30)	89	56	17	10	66	22	14
GMM (h=0.0044)	113	71	20	12	90	30	18

ARL_1 performance of CUSUM and GMM anomaly detectors are summarized in Table 3.1. The CUSUM algorithm with all chosen (k, h) parameters outperforms GMM in linear and exponential degradation detection if the data have a uni-modal normal distribution in the healthy operating period of the engine. The less degradation coefficient means late detection and the performance advantage of CUSUM increases for all (h, k) candidate pairs when the degradation coefficient decreases.

Secondly, healthy data was assumed to be from a mixture of two Gaussian noises with means $\mu_1 = 0.5$, $\mu_2 = -0.25$ and standard deviations $\sigma_1 = 2$, $\sigma_2 = 1$, respectively. Weights of the Gaussian models are taken as equal, i.e., 0.5. As in the previous case, half of the samples were separated to determine h and linear and exponential degradations were separately added to other half at the point where time equals to 50 $a.u.$. A GMM with 2 components are fitted to data in healthy period. The control limit h was set to 0.00204 which provides $ARL_0 = 370$. The CUSUM parameters, namely μ and σ, were also computed using the data in first 50 $a.u.$s. The candidate k parameters were also chosen as $\{0.5, 0.75, 1\}$ for CUSUM and the corresponding hs were set to $\{0.5, 0.75, 1\}$ to provide $ARL_0 = 370$.

Table 3.2 depicts the ARL_1 performance of CUSUM and GMM anomaly detectors

Table 3.2. The degradation detection performance of CUSUM and GMM in terms of ARL_1, i.e., speed of detecting degradation onsets when distribution of the healthy data is modeled using mixture of two Gaussian distributions.

	Exponential (α)				**Linear (α)**		
MODEL	0.005	0.01	0.05	0.1	0.01	0.05	0.1
CUSUM (k=0.5, h=5)	98	63	21	13	75	28	18
CUSUM (k=0.75, h=3.72)	106	69	22	13	85	30	19
CUSUM (k=1, h=2.99)	117	75	23	14	94	34	21
GMM (h=0.00206)	163	95	31	17	128	52	33

for the second scenario. The CUSUM algorithm with all chosen (k, h) parameters outperforms GMM in linear and exponential degradation detection when the data in the healthy operating period of the engine have a distribution of two Gaussian mixtures. The performance advantage of CUSUM increases for all (h, k) candidate pairs when the degradation coefficient decreases as in the previous case.

The CUSUM algorithm is adopted as anomaly detector in the rest of this thesis, since it performs better in the detection of exponential and linear degradations.

3.3. Anomaly Triggered RUL Estimation (AT-RUL)

AT-RUL estimation is a system level approach to estimate RUL (time to failure) of a machinery. In AT-RUL, CUSUM algorithm is used for continuous monitoring and change (anomaly) detection. Once an anomaly is detected, it is regarded as the onset of degradation. RUL is estimated starting from the degradation onset point. The proposed AT-RUL estimation system is illustrated in Figure 3.3.

The system consists of two main blocks, which are anomaly detection and RUL estimation blocks. The main objective of anomaly detection block is to determine the degradation onset point in individual systems. The RUL estimation block is trained

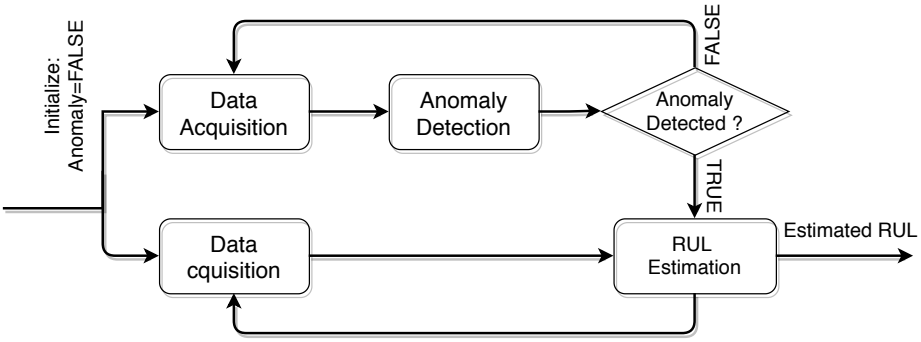


Figure 3.3. Anomaly Triggered Remaining Useful Life estimation model. Anomaly detection block determines the degradation onset point in individual systems. The RUL estimation block starts to estimate RUL as soon as an anomaly is detected.

only on active degradation period, i.e., from the detected degradation onset up to the complete failure point of the training samples. RUL estimation during testing starts as soon as an anomaly is detected on the corresponding sensor measurements.

The proposed model is convenient for any type of time series data with a normal distribution in the healthy period. It can be both raw sensor measurements or features extracted from them.

3.3.1. Cumulative Sum Model Design

Cumulative sum algorithm is adopted for the anomaly detection in the proposed architecture, since it performs better in the detection of exponential and linear degradations as stated in Section 3.2.3. In the industry, most of the machinery is monitored via multiple sensors or features. To monitor the multiple features simultaneously, assuming independence, we employed individual (Bernoulli) CUSUM charts rather than a multi-variate one. The individual charts perform better in the lack of the prior information about dependencies [103, 104].

The first step for the design of a CUSUM control chart is to characterize the

healthy operation, i.e., to determine the mean (μ) and standard deviation (σ) (see Equations (3.1) and (3.2)) prior to degradation onset. Since the sensor measurements are affected by external factors such as environmental noise, sensor position, operation conditions (e.g. load), etc., each machine needs to be characterized individually. Hence, the mean and standard deviation for each system are estimated during a fixed number of average life time after installment, during which the system is assumed to be healthy.

Secondly, the reference value k and the control limit h need to be selected based on the training samples. The reference value k is chosen empirically from $\{0.5, 0.75, 1\}$ based on training data [105]. Once k is fixed, the control limit h is empirically determined to provide a high average run length (ARL) values for false alarms and low ARL values for true alarms.

3.3.2. RUL Estimator Design

The main idea behind AT-RUL is to avoid RUL estimation in healthy operation during which the RUL is ill-defined. To this end, the AT-RUL scheme trains and tests RUL estimators after the degradation onset, which is marked/flagged with the anomaly detector. The CUSUM anomaly detector is used in this thesis as a viable choice, yet other choices of detectors can also be applied, esp. when the gaussianity assumption during the healthy period may not be valid.

All sensor sequence samples are initially processed with the anomaly detector whose hyper parameters are adjusted using a validation set separate from both training and test sets. The CUSUM detector employed here is rather insensitive to small changes in its hyper-parameters, hence direct use of the parameters suggested in literature may also be suggested as a viable option in case needed. Each recording's end point is taken as the complete failure point, which adheres with the C-MAPSS data description and, in case of simulation data, is determined by a threshold described in Section 3.4.1. Each training sample is clipped from the degradation (anomaly) onset point and used to train the RUL estimator.

The AT-RUL scheme is applicable with any machine learning based RUL estimator model. These models can broadly be classified in three groups: The memoryless estimators that only use the instantaneous sensor measurements, the estimators with explicit fixed term memory and the ones with implicit memory. The first group is not suitable for many industrial applications due to low SNR of sensor measurements which makes trend tracking (i.e., memory) a necessity for reliability. The second group uses a fixed length past data (usually in the form of a sliding time window) to estimate the RUL. Linear regression (LR) and random forest regression (RF) models are popular examples of this group. They can compute the RUL estimates with a fixed delay in time which is based on the sliding window size. The third group is mainly composed of the deep long-short-term-memory (LSTM) networks. LSTM networks are not only capable of remembering past data but also provide instantaneous RUL estimations without any delay but with a burn-in time. We used LR, RF and LSTM RUL estimators with and without AT-RUL scheme for comparative assessment.

The clipped training samples are used to train the LR, RF and LSTM RUL estimators, with respect to the true RULs, defined as described above. No data preprocessing is applied for the LR and RF estimators. The raw sensor measurements and extracted features are directly fed into the estimators using a sliding window. The LR estimator is trained with respect to the least squares error criteria. The RF estimator is an ensemble of regression trees in which only random subsets of the input dimensions are used for the candidate splits during the training of individual trees. The variance reduction method is used for RF training [106].

3.4. Simulation Experiments

The effect of anomaly detection and triggering on ML based RUL estimation is assessed on simulation data in comparison to the conventional approach. The assessment is based on root mean square error (RMSE) between estimated and (linearly decreasing) true RULs.

3.4.1. Data

We used an exponential degradation model for the simulations [101, 102]. The simulation data ($\hat{s}[n]$) is generated by adding an exponential degradation signal ($e^{\alpha n}$) to a zero-mean and unit-variance Gaussian noise, starting from a randomly selected (from a uniform distribution of $U[50, 100)$) onset point. Time of failure is marked as the first point when the simulated signal reaches a predefined threshold ($\hat{s}[n] = 15$). The true RUL is a unit slope linearly decreasing function from the onset of the degradation to the failure point. The final signal has the following form,

$$\hat{s}[n] = r[n] + u[n - \tau] \left(e^{\alpha(n-\tau)} - 1\right) \tag{3.12}$$

where n is an arbitrary time unit (a.u.), $r[n]$ is the additive white Gaussian signal, $u[n]$ is the unit step, τ is the degradation onset point and α is the degradation coefficient which is randomly selected from $U[1, 2)$. We generated 400 training and 100 test sample signals for the experiments.

3.4.2. Training

First 30 data points of each signal are assumed to be in healthy period and are used to characterize the healthy operation (i.e. to estimate μ and σ in Equations (3.1) and (3.2)). The training set is divided into training and validation sets to choose reference value k and threshold h. 0.5, 0.75 and 1 are used as candidate k's [105]. For each k, we set ARL_0 (Average run length for false alarms) to be 600, which corresponds to 5% false alarm rate during the assumed healthy period. The (h, k) pairs thus selected are $(8, 0.50)$, $(4.2, 0.75)$ and $(3.4, 1)$. These values are tested on the validation set, and their response time to degradation (ARL_1) is evaluated. $(4.2, 0.75)$ exhibits the best performance where $ARL_1 = 12$. h is fine-tuned to $h = 3.57$ which provides $ARL_1 = 8$ and 6% Type 1 error.

All simulation signals are preprocessed with CUSUM and degradation onset

points are determined. The sections prior to the detected degradation onsets are discarded (cropped). Each sample is assigned a true RUL value starting from the degradation onset in a linearly decreasing fashion such that RUL is zero at the time of failure (defined as $\hat{s}[n] \geq 15$). LSTM RUL estimator takes $\hat{s}[n]$ as input and generates a RUL estimation at each time instant n. However, RF and LR estimators use the last L points to make an estimation.

For comparison purposes, the same estimators are trained and run on the complete (non-cropped) sequences. In that case, the true RUL used for the training is assumed fixed beyond a predefined maximum RUL as described in [68]. The maximum RUL value is empirically set to 114, which gives the best performance for conventional approaches on our data set. Comparisons between conventional approach and AT-RUL are performed on the sections after the detected degradation onsets. As the degradation onset points may not be aligned throughout the data set, we also reported the number of simulated signals (test samples) for which degradation has been detected (hence RUL is estimated) together with the actual number of samples for which the exponential degradation signal was onset in the simulated signals.

Hyperparameters of RUL estimators are optimized with respect to the conventional piecewise linear true RUL by employing cross validation in the training set. LSTM RUL estimator consists of two LSTM layers each with 64-D state vectors and two fully connected layers with 16 and 8 neurons, respectively. On the other hand, least square LR estimator is utilized. The window size is selected as $L = 20$ which provides the best RUL estimation accuracy with least squares LR estimator. The same window size is also used for the RF estimator. The number of trees is set to 100, which is optimized in the training set via cross-validation. All other parameters are left at their default values [107]. Identical estimators are used for the conventional and AT-RUL estimation to facilitate fair comparison.

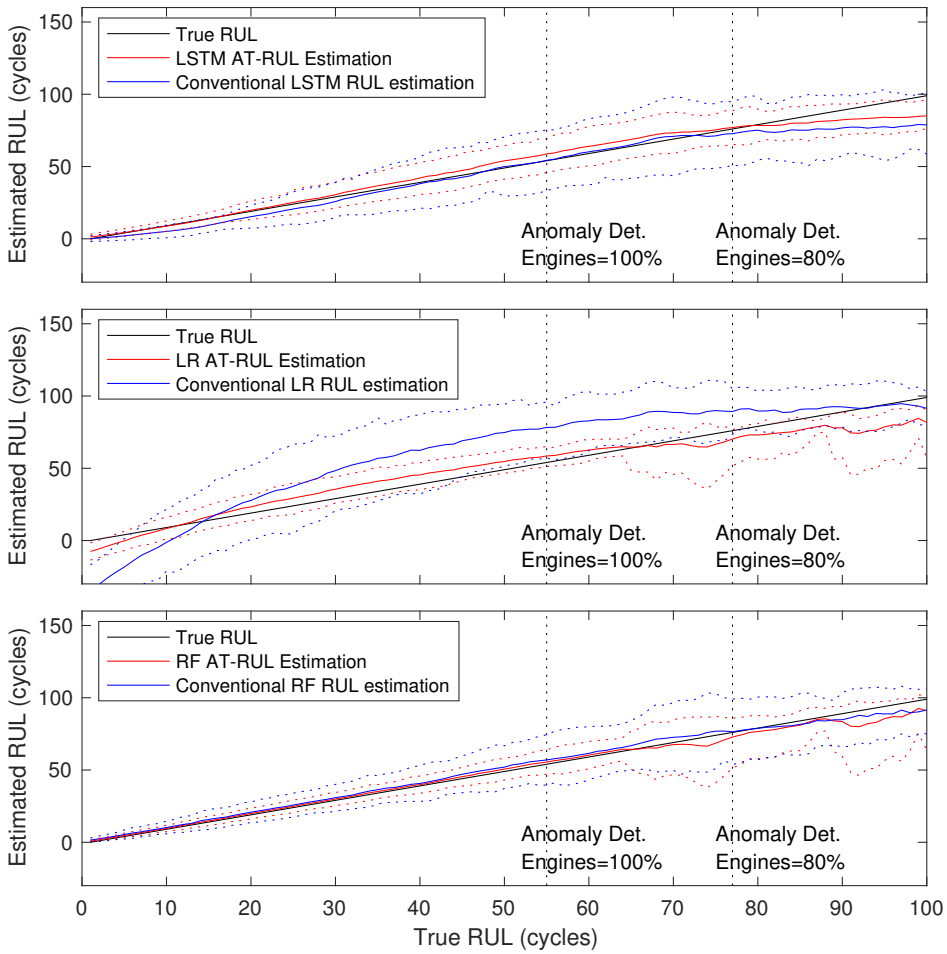


Figure 3.4. The RUL values estimated with LR, RF and LSTM estimators for AT-RUL and conventional approaches on simulation data. The (mean $\pm$ 2std) plots across all test samples are given as functions of true RUL.

3.4.3. Results

The CUSUM anomaly detector detected anomalies, on average, seven time points after the true degradation onset in the simulation data. The latest detection of a degradation is 17 points which corresponds to 20% of the lifespan of the fastest degrading engine. A degradation onset was detected for all test engines as soon as the true RUL is less than 55 (a.u.), while no detection was done for 20% of the engines when true RUL is larger than 77 (a.u.), as marked on Figure 3.4.

Figure 3.4 depicts the estimated RUL (mean $\pm$ 2std) versus the true RUL, for

AT-RUL and conventional estimation approaches, using LSTM, RF and linear RUL estimators. All three estimators show distinct characteristics with the conventional approach. Among all, the LR estimator is the worst performing one. The RF and LSTM estimators have comparable performances, with the LSTM estimator slightly deviating from the true RUL for low (< 30(a.u.)) and high (> 80(a.u.)) RUL. Nevertheless, all estimators' mean performance, across all engines, is either significantly improved or slightly degraded for AT-RUL approach, for all true RUL values. The improvement is clearer in the decreased standard deviations of the RUL estimations across the samples. The AT-RUL approach is not only capable of improving even the poorly performing LR estimator but also provides robustness. Note that the means and standard deviations for the AT-RUL approach are computed only over the engines that are flagged with a degradation onset. It is fair to say that the RF and LSTM estimators with AT-RUL approach have comparable and high performance for all true RULs, while the latter is more robust across the test samples for higher true RULs.

3.5. C-MAPSS Data Experiments

The effect of anomaly detection and triggering on ML based RUL estimation is assessed on a popular benchmark data in comparison to the conventional approach. The assessment is based on root mean square error (RMSE) between estimated and (linearly decreasing) true RULs. The C-MAPSS data set also includes a partially recorded test set. The single point RUL estimation performance of AT-RUL estimation on this test set is also presented.

3.5.1. Data

We used the publicly available NASA C-MAPSS turbofan engine simulation data which is commonly used for benchmarking prognostic algorithms [39]. The C-MAPSS is a commercial software to simulate the transient operation of the turbofan engines. The benchmark data is created using the simulation software by imposing a degradation as explained in [39]. There are 4 subsets of the benchmark data for the different fault

Table 3.3. The sensor readings that are used for RUL estimation in this thesis.

Symbol	**Description**	**Units**
T50	Total temperature at LPT (low-pressure turbine) outlet	°R
P30	Total pressure at HPC (high-pressure compressor) outlet	psia
Ps30	Static pressure at HPC outlet	psia
phi	Ratio of fuel flow to Ps30	pps/psi

modes and operating conditions. $FD001$ subset which includes single fault mode single operating condition samples is used in the experiments..

The FD001 subset includes measurements from 21 sensors and 100 jet engines spanning the full range from healthy operation to complete failure. The life of engines are assessed in terms of their usage time instead of raw time since the engines' degradation is negligible when they do not operate. Moreover, the sensor measurements exhibits oscillatory behavior in operation, and hence their average values are recorded and provided in the data set. Therefore, the time series data include multiple sensor measurements that are recorded periodically at each operation *cycle*, and the RUL is estimated in terms of the cycles. Few sensor measurements have binary or constant values during the lifespan. Table 3.3 summarizes the four sensor measurements with engine independent responses that are included in the experiments. These are the ones that respond to degradation either with an increasing or decreasing trend across all engines. The complete failure point of an engine is defined with respect to a predefined engine wear threshold. However, the wear threshold does not correspond to a threshold in the sensor measurements because the response observed in sensor readings is a complex function of the wear and other parameters.

The C-MAPSS data set also includes a partially recorded test set. This test set is composed of 100 engines' sensor data which are recorded up to an arbitrarily chosen point in time prior to complete failure. The recordings are labeled with the true RUL at the end of each recording, which can be linearly extrapolated to the recorded data. This test data set is commonly used to assess the accuracy of single point RUL estimation

at the end of each recording [67–69].

3.5.2. Training

The average time-to-failure value for these 100 engines is 207 cycles. First 40 cycles (20% of the average duration of recordings) are assumed *healthy*, i.e. prior to true degradation onset, from which the μ and σ that characterize the healthy operation are estimated. We adopted voting for sensor fusion, namely, an anomaly is flagged when at least two of the four monitored sensors make a detection.

The k parameter is fixed to 0.75 and h is initially selected such that the false alarm rate in the first 20% of the recordings is 5%. Thus selected h values are 3.47, 3.55, 3.69, 3.12 and 3.02 for 5 folds of the cross-validation. These h values are close to the fixed setting $h = 3.34$ which is commonly used in the literature [105]. To facilitate the homogeneity of cross-validation experiments and for fair comparison, we also have fixed $h = 3.34$ for all, which corresponds to $ARL_0 = 367$.

We used two LSTM layers with 64 states and two densely connected layers with 8 neurons in each [70]. Sliding window lengths for RF and LR estimators are fixed to 20. The number of estimator trees in the RF is set to be 100.

3.5.3. Results

The C-MAPSS data experiments were run following the experimental scheme explained in the previous section. As the true degradation onset points are not known for C-MAPSS data, it is not possible to assess the performance of anomaly detection point accuracy, yet the CUSUM flagged 100% (80%) of the test engines before the true RUL reached 60 (80) cycles.

Figure 3.5 depicts the RUL estimations by different estimators for both AT-RUL and conventional estimation approaches, as a function of true RUL. The AT-RUL

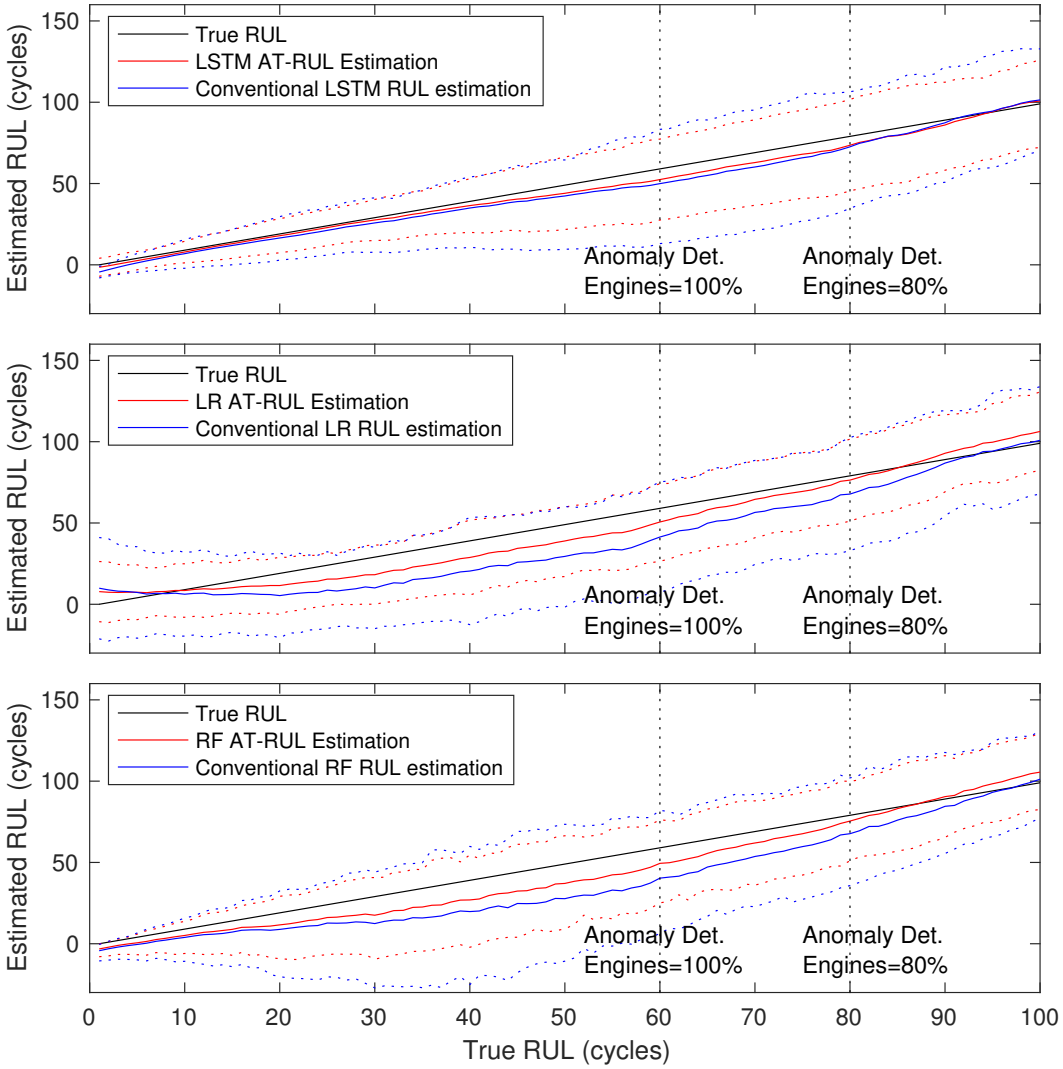


Figure 3.5. The RUL values estimated with LR, RF and LSTM estimators for AT-RUL and conventional approaches on C-MAPSS data. The (mean $\pm$ 2std) plots across all engines are given as functions of true RUL.

approach's improvement over the conventional one, in both means and standard deviations of RUL estimations, is clearly observed for all estimator types. The LR estimator is still the worse performing one compared to other estimators, though the AT-RUL improved its accuracy significantly as well. Nevertheless, the difference between the LSTM and RF estimators is more pronounced compared to the simulation experiments, though both are improved by the AT-RUL approach. More specifically, the LSTM estimator with AT-RUL is the best performing one in terms of both mean and standard deviation across a wide range of true RUL values.

3.5.4. Single Point RUL Estimation Experiments

The jet engine data set also includes a partially recorded test set. This test set is composed of 100 engines' sensor data which are recorded up to an arbitrarily chosen point in time prior to the complete failure point. The recordings are labeled with the true RUL at the end of each recording, which can be linearly extrapolated to the recorded data. This test data set is commonly used to assess the accuracy of single point RUL estimation at the end of each recording [67–69]. Our CUSUM based anomaly detector could detect an anomaly in 79 out of 100 recordings. It is very likely that degradation might not have been onset before the end of the recording in the remaining 21 recordings (Mean true RUL at the end of these 21 engines' sensor recordings was 112 cycles). In Table 3.4, we report the mean RMSE in single point RUL estimation at the end of 79 engines' recordings as well as the score defined as [39],

$$\text{Score} = \begin{cases} \sum_{i=1}^{n} e^{-\left(\frac{RUL_{est}-RUL_{true}}{a_1}\right)} - 1 \text{ for } (RUL_{est} - RUL_{true}) < 0 \\ \sum_{i=1}^{n} e^{\left(\frac{RUL_{est}-RUL_{true}}{a_2}\right)} - 1 \text{ for } (RUL_{est} - RUL_{true}) \geq 0 \end{cases} \tag{3.13}$$

where $\alpha_1 = 13$ and $\alpha_2 = 10$. The score was designed to penalize the late predictions (i.e. estimated RUL > true RUL) more than the early predictions The AT-RUL estimator has significantly lower scores than conventional RUL estimation for all 3 types of estimators. The RMSE values are close for both cases with AT-RUL having a lower RMSE only for LSTM estimator. Lower scores and close RMSE values indicate that AT-RUL tends to underestimate RUL in this data set, staying on the safe side.

3.6. Discussion

The results reveal that the proposed AT-RUL estimation model provides better performance than the conventional ML methods. It generates more accurate predictions as time progresses, which is a desired property because in general the points that are close to the failure are much more critical for the industry practitioners and maintenance teams. This is achieved by detecting and discarding the redundant(healthy)

Table 3.4. The mean and standard deviation of RMSE and Score for different ML models used with AT-RUL and the conventional approach on C-MAPSS FD001 test set. (Note that LR has unique solution.)

ML Model	**AT-RUL estimator**		**Conventional RUL est.**	
	RMSE (mean ± std)	SCORE (mean ± std)	RMSE (mean ± std)	SCORE (mean ± std)
LSTM	17.15±0.24	392±23	19.01±1.7	491±54
RF	21.1±0.2	520±21	18.3±0.15	740±37
LR	22.44	580	19.4	1171

part of the data which does not consist of reliable degradation information. On the other hand, it has been shown that CUSUM is a useful anomaly detection tool for the industrial degradation data. It is in fact a tool to detect the change in the mean of a random normal process, and assumes that data under degradation (fault) is also a random normal process with a different mean value. However, the results suggest that it can be also employed for detecting a gradual change.

The ground truth RUL is ill-defined in the healthy period. Therefore, the conventional ML models are trained by assigning a piecewise RUL ground truth to overcome this problem. However, this strategy distorts the performance of ML models by forcing them to estimate the RUL in ill-defined periods, i.e., the model is compelled to regress the same maximum RUL value for non-degrading (healthy) period even if the observations are not similar. The proposed model prevents this distortion caused by ill-defined periods by eliminating the data before the degradation onset points from the training of ML models as well as from the test data.

Moreover, the maximum fixed RUL value is an arbitrary choice, and does not hold for all systems. When the degradation onset point in the sensor data is earlier or later than the chosen maximum RUL, it takes longer time for the RUL estimator to converge the true RUL value. Figure 3.6 shows an example where the degradation

onset point does not coincide with the chosen maximum RUL. The sensor data is composed of 180 cycles. The conventional approach assumed constant RUL until the true RUL is 125 cycles (corresponds to 55^{th} cycle in the sequence). AT-RUL detected a degradation onset at 70^{th} cycle. It can be easily deduced from the RUL estimation of conventional LSTM RUL estimator that the degradation starts at 75^{th} cycle where the AT-RUL method also detects an anomaly in the sensor measurements. It takes almost $50 - 60$ cycles to generate acceptable RUL estimation performance for conventional LSTM estimator after the degradation onset point.

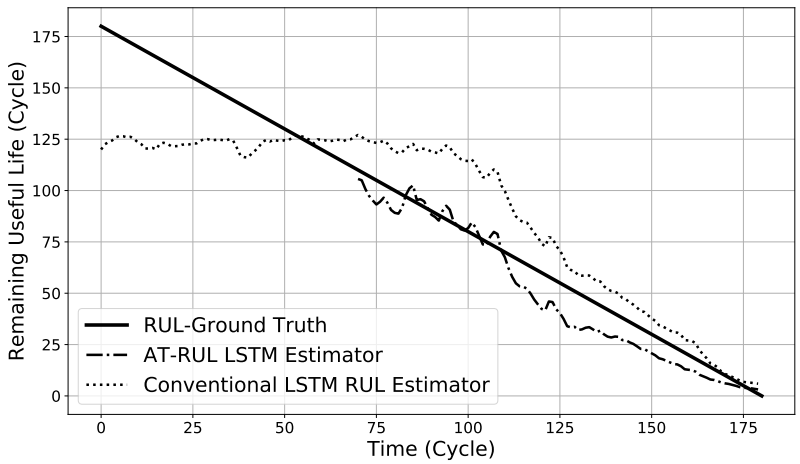


Figure 3.6. AT-RUL and conventional RUL estimation with LSTM for a sample test engine from jet engine data set.

It can be concluded based on our results that LSTM outperforms random forest and linear regression at the points that are close to the failure. Moreover, LSTM model starts to estimate RUL at the time of anomaly detection without loosing any time, whereas LR and RF have to wait for an extra time period to estimate RUL.

Furthermore, the proposed model decreases the computational time by avoiding unnecessary computations of the RUL estimation during the healthy period. All of the benchmark ML models and CUSUM have computational complexities that are linear with time. However, the number of computations at each time is only 5 for CUSUM

for a single monitoring statistics and more than 100 for all benchmark RUL estimators.

4. OPERATING CONDITION INVARIANT FEATURE LEARNING

4.1. Proposed Approach

Remaining useful life (RUL) estimation is very critical for planning the maintenance of machinery in various industries. Deep learning models have gained popularity as the key tools in estimating RUL. However, variable operating conditions have adverse effects on the performance of these deep learning models, since the sensor readings or extracted features are varying with operating conditions.

As stated in Chapter 2, many studies focused only on machinery with single operating condition. Moreover, the studies that focused on both single and multiple operating conditions reported incomparably better performance in single operating condition even if the complexity of the model is increased for multi-operating conditions. However, in industry, many machinery operate in different operating conditions. Therefore, we propose an operating condition invariant (OCI) feature extraction method based on siamese neural networks (SNN) to tackle this problem.

The model is introduced in the following section. Section 4.3 presents the experiments and results. The chapter is concluded with the discussion of the advantages and the disadvantages of using OCI features.

4.2. Operating Condition Invariant Feature Extraction

We train a multilayer perceptron (MLP) model as the OCI feature extractor from raw features which vary with operating conditions, using a siamese (SNN) architecture. Pairs of raw features is fed into the SNN, with twin MLP branches, which is trained so as to minimize the difference in the output (features) for consecutive time windows over the operating condition change points and to maximize the difference between

time windows from the start and end of individual recordings. This assumes that the training sensor recordings span the full range of machine lifetime, i.e. the machinery is healthy at the start of the sensor data recordings and is completely failed at the end.

4.2.1. Siamese Architecture with MLP Branches

SNNs are the special types of neural networks that contain twin sub-networks (with identical weights) connected through an output layer, which may simply be a loss function. The training data is prepared in pairs where each pair is labeled/tagged for supervised training. The pairs are fed into twin networks and the specially designed loss functions compute a loss over the outputs of twin networks. Figure 4.1 depicts the general architecture of an SNN with MLP twins and loss function as the output layer, that we have employed in this thesis. Each of the networks includes an input layer, two hidden layers with hyperbolic tangent (tanh) activation function and an output layer that is linearly activated. The MLP twins are trained using a contrastive loss function as described in Section 4.2.2, as the OCI feature extractor from input raw features.

4.2.2. Training with Contrastive Loss

SNNs were first proposed to learn a similarity metric which can be used for verification tasks such as signature and face verification [108, 109]. The known sample is given to the first sister network and the sample that is wanted to be verified is given to the second one. The loss function measures the similarity of this pair. It is desired that the sister networks generate representations that have low loss value for genuine, i.e., *like* pairs, and high loss value for fake i.e., *unlike* ones. The training of SNN requires like and unlike pairs which should be created from the individual degradation sensor recordings. In our case, the *like* and *unlike* pairs refer to data from the same and different health conditions, respectively.

For the like pairs, it is assumed that the health status of a system at consecutive time instances remain unchanged in a degradation sensor recording, hence the succes-

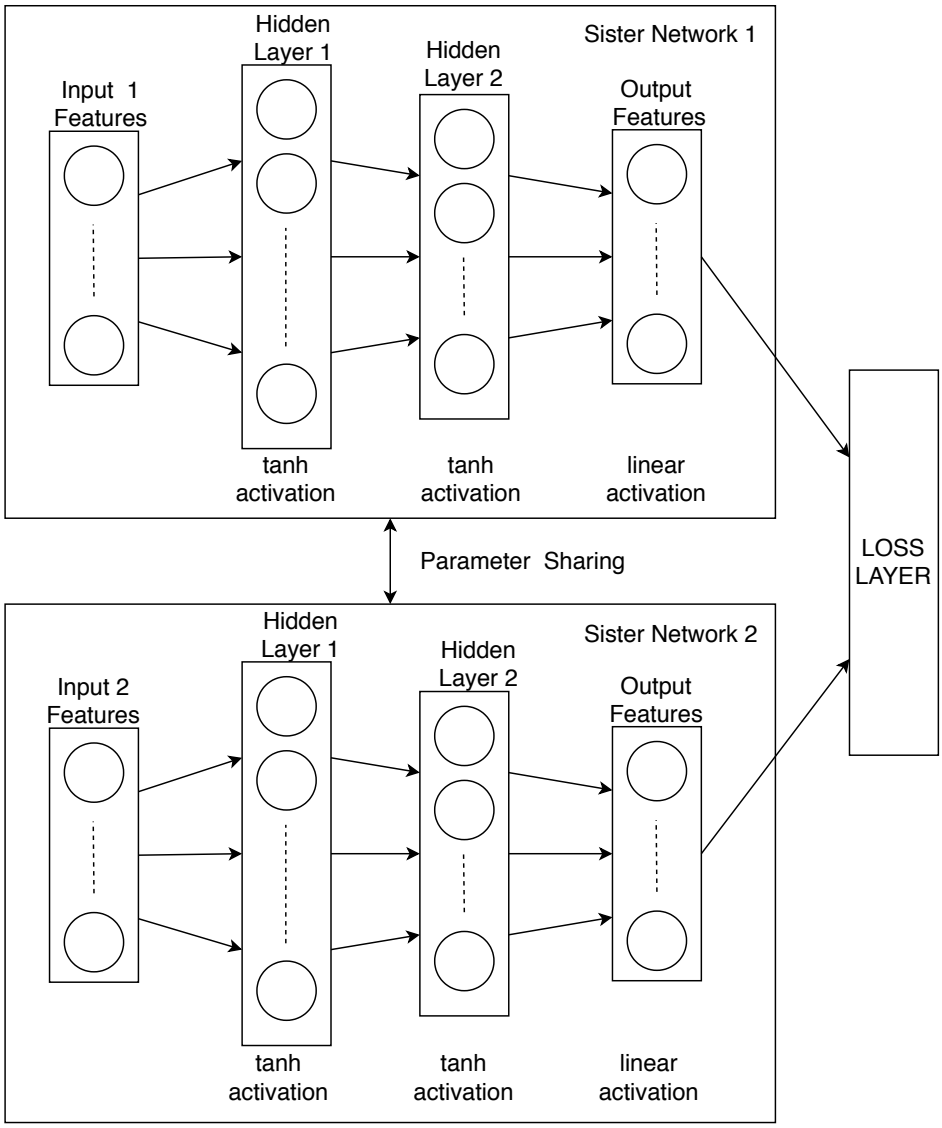


Figure 4.1. The architecture of an SNN with MLP twins and loss function as the output. Each sister MLP consists of two hidden layers with hyperbolic tangent (tanh) activation, and serves as the operating condition invariant (OCI) feature extractor.

sive recordings are used as the like pairs. We specifically used consecutive recordings across operating condition changes to force the training on generating similar features across varying operating conditions. For the unlike pairs, it is assumed that very early measurements are from the healthy period and the very late measurements are from the faulty period. Hence pairs of recordings, one from the assumed healthy period and one from the faulty period, are utilized to generate the unlike pairs. We used equal number of recordings from both periods and used all pairs of these two sets as unlike

1: $x_{1,1}, \ldots, x_{1,T_1},\ x_{2,1}, \ldots, x_{1,T_2},\ \ldots x_{k,1} \ldots x_{k,T_k}$
2: $TrSet$ (Training Set for SNN)
3: **for** $e = 1, ..., k$ {For each engine} **do**
4: $count \leftarrow 0$
5: **for** $n = 1, ..., T_e - 1$ {For each measurement} **do**
6: **if** OC_n not equals to OC_{n+1} {If Operating Conditions are different} **then**
7: $TrSet.insert([x_{e,n}, x_{e,n+1}, 0])$ {Add consecutive pair to training set with label 0}
8: $count \leftarrow count + 1$
9: **end if**
10: **end for**
11: $w \leftarrow \lceil \sqrt{count} \rceil$
12: **for** $i = 1, ..., w$ **do**
13: **for** $j = 1, ..., w$ **do**
14: $TrSet.insert([x_{e,i}, x_{e,T_e-j}, 1])$ {Add pair to the training set with label 1}
15: **end for**
16: **end for**
17: **end for**

Figure 4.2. Generation of Training Set Pairs for SNN.

training pairs. The procedure of generating like and unlike pairs for the training of the SNN is summarized in the algorithm that is depicted in Figure 4.2.

The other critical issue is the loss function that is used in the training of the SNN model. Let $(\mathbf{x}_i^1, \mathbf{x}_i^2)$ be the i^{th} pair of raw features and $f(.; \mathbf{W})$ be the function that transforms raw features to OCI features, $\mathbf{r}_i^1$ and $\mathbf{r}_i^2$, as,

$$\mathbf{r}_i^1 = f(\mathbf{x}_i^1; \mathbf{W}) \tag{4.1}$$

$$\mathbf{r}_i^2 = f(\mathbf{x}_i^2; \mathbf{W}) \tag{4.2}$$

where $\mathbf{W}$ are the parameters to be learned. The contrastive loss is defined as,

$$L(y_i, \mathbf{r}_i^1, \mathbf{r}_i^2) = (1 - y_i)d(\mathbf{r}_i^1, \mathbf{r}_i^2) + y_i \max(0, m - d(\mathbf{r}_i^1, \mathbf{r}_i^2)) \tag{4.3}$$

where y_i is binary indicator which is equal to 0 for the like pairs and 1 for the unlike pairs, $d(\mathbf{r}_i^1, \mathbf{r}_i^2) = ||(\mathbf{r}_i^1 - \mathbf{r}_i^2)||_2^2$ and m is the margin. Minimization of L corresponds to minimization/maximization of the distance between like/unlike pairs. The margin m limits the intra-distance of unlike pairs. The loss is minimized with respect to $\mathbf{W}$ using stochastic gradient descent optimization algorithms.

This choice of like pairs assumes that the operating conditions are available or sensible which may not be the case in industrial applications. To address this problem, the like pairs can be constructed by using all consecutive pairs in the degradation sequences. But in this case the number of unlike pairs should be increased too, since the number of like pairs increases. To increase the number of unlike pairs, the samples that are close in time may need to be included. This may lead to fail of our assumption that unlike pairs are constructed from very early (healthy) and very late (faulty) measurements. Therefore, for such a case the binary indicator y_i in contrastive loss is modified as follows:

$$y_i = \frac{\Delta t_i}{TTF_i} \tag{4.4}$$

where TTF_i is the time to failure of the engine from start of the operation to the complete failure and Δt_i is the time difference between the selected pairs.

4.3. Experiments

The efficiency of the proposed OCI feature extraction method is assessed using NASA C–MAPSS data which is commonly used as a benchmark data set for RUL estimation [39]. The data set is summarized in Section 4.3.1. The OCI features are generated by an MLP trained using an SNN architecture with contrastive loss. The

new features are compared with conventional features, which are the raw sensor measurements in the given data set, using two popular deep RUL estimation architectures, namely DCNN and LSTM [70, 71]. The RUL estimation results are reported in terms of root mean square estimation error (RMSE) and the score function that punishes the positive errors (i.e. the true RUL is more than the estimated RUL) more than the negative errors as,

$$\text{Score} = \begin{cases} \sum_{i=1}^{n} e^{-\left(\frac{\hat{\mathrm{T}}_i - \mathrm{T}_i}{a_1}\right)} - 1 \text{ for } (\hat{\mathrm{T}}_i, -\mathrm{T}_i) < 0 \\ \sum_{i=1}^{n} e^{\left(\frac{\hat{\mathrm{T}}_i - \mathrm{T}_i}{a_2}\right)} - 1 \text{ for } (\hat{\mathrm{T}}_i - \mathrm{T}_i) \geq 0 \end{cases} \tag{4.5}$$

where T is the true RUL, $\hat{\mathrm{T}}$ is the estimated RUL, free-parameters $\alpha_1 = 13$ and $\alpha_2 = 10$ [39]. Lower scores indicate better performance.

4.3.1. C–MAPSS Turbofan Engine Degradation Simulation Data Set

NASA C–MAPSS is a tool that is designed to simulate operation of large turbofan engines. The data set used in this thesis is generated using NASA C–MAPSS simulator [39]. 21 different sensor measurements are recorded besides three different values which define the operating conditions for the engines. Samples are collected periodically at each operation *cycle* and RUL is defined in terms of cycles. The data set is divided into 4 subsets which consist of single operating condition–single fault mode (FD001), multiple operating conditions–single fault mode (FD002), single operating condition–multiple fault modes (FD003) and multiple operating conditions–multiple fault modes (FD004). FD002 consists of the records that are collected from the engines with one fault mode–six operating conditions while FD004 contains the engines with two fault modes–six operating conditions.

Each subset is divided into training and test sets. The training set includes the sensor recordings of engines from a healthy condition to the failure. The sensor

measurements in test engines are available until a specific cycle and RUL labels are given at this cycle in terms of the number of cycles that remain to the failure. The number of available training and test samples are 260 and 259 for FD002, 248 and 249 for FD004.

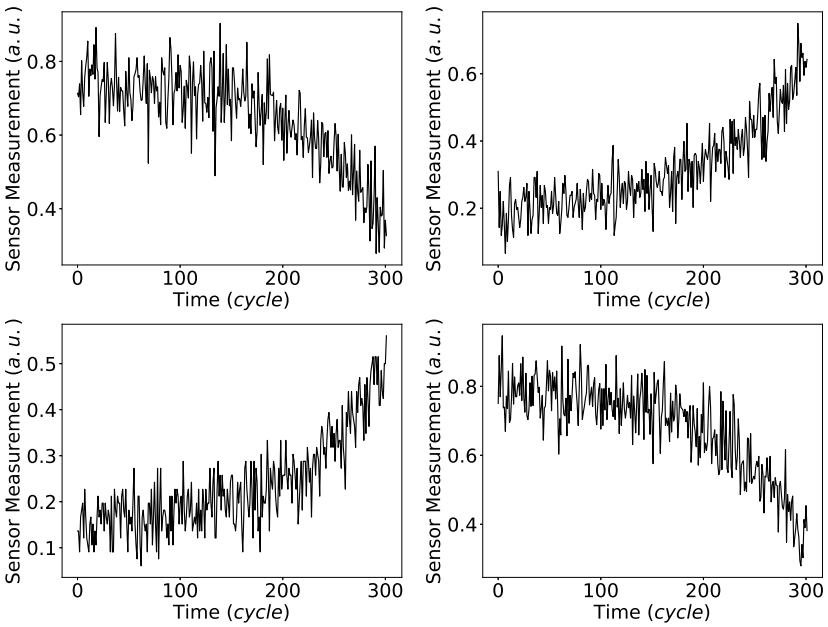
4.3.2. Operation Condition Invariant (OCI) Features

Operating condition invariant feature extractor MLP model has two hidden layers with 64 and 128 neurons, respectively. Both input and output layers have 21 neurons in order to employ the same RUL estimators for both feature sets. All inputs (raw features) are normalized to $[0, 1]$ using min–max normalization [110]. The training set is generated by applying Algorithm 4.2 on the training sets of FD002 and FD004 subsets. Siamese (SNN) architecture with contrastive loss defined in (4.3) and a margin value $m = 1$ is used for the training. AdaGrad algorithm with a learning rate of 0.01 is employed [111]. A randomly selected 10% of the training set is used as the validation set. The maximum number of epochs is set to be 100 and the model that provides the best performance on the validation set is selected as OCI feature extractor. The Keras library is used for implementation and simulations are run on GPU at Google Colaboratory[1] [112].

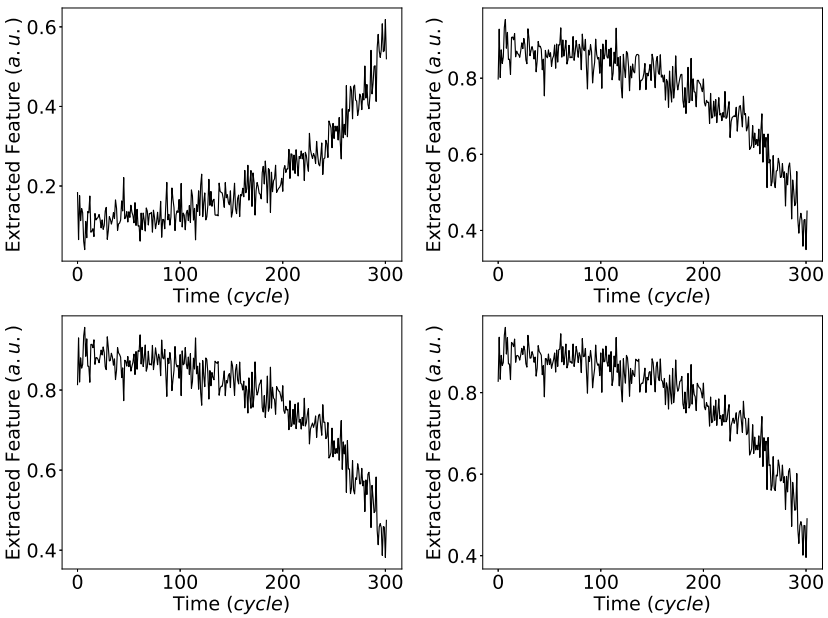
Figure 4.3 and Figure 4.4 depict four raw features and four MLP generated features out of 21 OCI features, for two sample engines with single and varying operating conditions. The features with high Pearson correlation coefficients with true RUL are presented for both, as they reflect the degradation process better. The comparison of the raw features from both engines clearly shows the destructive effect of varying operating conditions on sensor measurements, which has adverse effects on RUL estimation. On the other hand, the new MLP generated OCI features clearly demonstrate the degrading health of the engines both under single and varying operating conditions.

Appendix A presents more raw sensor measurements and OCI features for a

[1] https://colab.research.google.com/



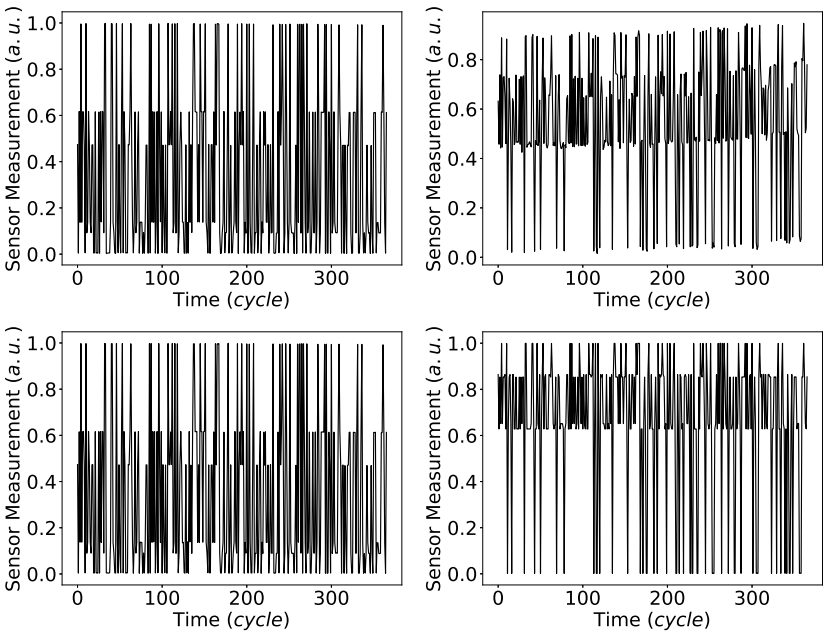
(a)



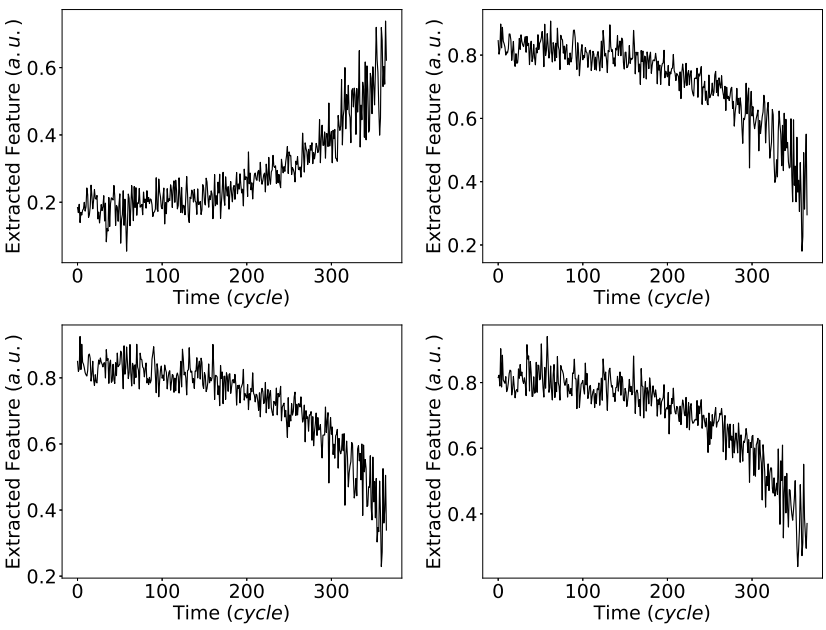
(b)

Figure 4.3. The raw sensor measurements and operating condition invariant features for a sample engine with single operating condition (Engine 48 in the test set of FD001): (a) Normalized raw features. (b) Operating condition invariant features.

sample engine in each subset of the C-MAPSS data set.



(a)



(b)

Figure 4.4. The raw sensor measurements and operating condition invariant features for a sample engine with multiple operating conditions (Engine 64 in the test set of FD002): (a) Normalized raw features. (b) Operating condition invariant features.

4.3.3. Effects on RUL Estimation

We trained and used identical LSTM and DCNN RUL estimators on raw and MLP generated features separately, to comparatively assess the effect of operating

condition invariant features on RUL estimation. The former model includes two LSTM layers that have 32 and 64 neurons, respectively, with two fully connected layers that have 16 neurons with hyperbolic tangent activation. Output layer is a linearly activated layer which regresses the estimated RUL value. The hyperparameters (namely, the architecture) that provide the best testing results for the raw features are used also with the OCI features to facilitate fair comparison.

The DCNN model consists of four identical convolutional layers with 10 kernels each have a size of 10×1 and an extra convolutional layer with only 1 kernel of size 3×1. Rectified linear unit (ReLU) activation is used in convolutional layers. Then, the extracted $2-D$ feature array is flattened and fed into a fully connected layer with 100 neurons. The output layer with a linear activation regresses the estimated RUL. The raw features are normalized to $[0, 1]$, as always. A sliding window approach is used to generate $2-D$ input array. Window lengths are 20 and 15 for FD002 and FD004 data sets, since the shortest test sequences include only 20 and 15 cycles, respectively. These hyperparameters had the best performance for the raw features and hence the same architecture is also used for the OCI features.

Both RUL estimators are trained using root mean square error as loss function and Adam optimizer [113]. Keras Python library is used for implementation and run on Google Colaboratory [112].

Table 4.1 illustrates the performance of LSTM and DCNN RUL estimators with the raw features and the new OCI features. The OCI features improve the RMSE and score performance of DCNN RUL estimator for FD002 by 32% and 83%, respectively. The RMSE and score improvements in DCNN RUL estimation in FD004 are 25% and 75%. The OCI features improve the RMSE and score performance of LSTM RUL estimator for FD002 by 40% and 79%, respectively. The RMSE and score improvements in LSTM RUL estimation in FD004 are 41% and 91%. Table 4.1 also includes the RMSE and score of the hybrid DCNN–LSTM model [72]. To the best of our knowledge, this model provides the best performance for the C-MAPSS FD002 and FD004 data

Table 4.1. RMSE and Score Performance of DCNN, LSTM and hybrid DCNN–LSTM networks on FD002 and FD004 subsets with raw features and OCI features.

		DCNN Raw Features	**DCNN OCI Features**	**LSTM Raw Features**	**LSTM OCI Features**	**Hybrid Model [72] Raw Features**
FD002 (Multi OC-Multi Fault)	**RMSE (mean±std)**	26.82 ± 1.23	18.28 ± 0.52	21.78 ± 2.41	13.07 ± 1.04	15.24 ± 2.65
	Score (mean±std)	11286 ± 3675	1966 ± 239	3694 ± 2067	776 ± 134	1282 ± 260
FD004 (Multi OC-Multi Fault)	**RMSE (mean±std)**	27.65 ± 1.69	20.73 ± 0.29	25.54 ± 3.14	15.17 ± 0.90	18.16 ± 2.17
	Score (mean±std)	12644 ± 6428	3134 ± 571	11527 ± 12341	1020 ± 135	1527 ± 322
FD001 (Single OC-Multi Fault)	**RMSE (mean±std)**	15.97 ± 0.6	15.95 ± 0.34	12.88 ± 0.68	13.86 ± 0.75	13.07 ± 0.72
	Score (mean±std)	507 ± 110	515 ± 54	293 ± 40	320 ± 36	245 ± 23
FD003 (Single OC-Multi Fault)	**RMSE (mean±std)**	17.10 ± 2.80	17.99 ± 0.82	14.5 ± 2	14 ± 1.4	12.22 ± 0.54
	Score (mean±std)	1420 ± 1003	1332 ± 513	427 ± 217	523 ± 203	287 ± 40

sets, reported in literature. LSTM RUL estimator with the OCI features enhances the RMSE and score performance of the best approach in the literature in FD002 by 14.5% and 39.5%, respectively [72]. The RMSE and score improvements for FD004 data set are 16.5% and 33.2%. The improvement is more for LSTM estimators, while the LSTM estimator consistently performs better than the DCNN. Nevertheless, even the DCNN estimator using the new operation condition invariant features, performs better than the LSTM with raw features.

Table 4.1 indicates that the OCI features worsen the RUL estimation score under single operating condition, though at a smaller level (9.2% in FD001 and 22% in FD003) compared to the improvement in the multiple operating conditions. The RMSE for

single operating condition, on the other hand, showed a slight decrease (14.5 vs 14) in FD003 (multiple faults) and a slight increase (12.88 and 13.86) in FD001 (single fault) with OCI features compared to raw features. Hence, the OCI features do not have a significant adverse effect under non-varying operating condition.

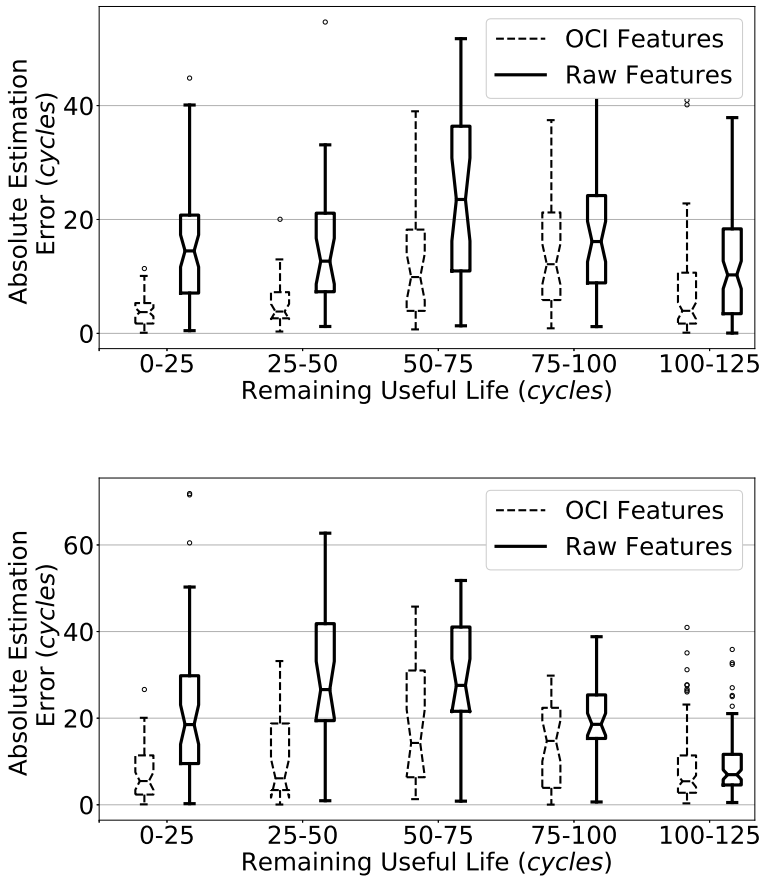


Figure 4.5. RUL estimation errors for the last recorded data points for LSTM RUL estimator: (Top) Test set of FD002 (single fault type-varying operating conditions.) (Bottom) Test set of FD004 (multiple fault types-varying operating conditions.)

Figure 4.5 depicts estimated RUL RMSEs in relation to the true RUL, for the better performing LSTM estimator, and for test sets of FD002 and FD004. The box-plots clearly demonstrate the better RUL estimation using OCI features for all true RUL ranges. Note that, for each test engine, the RUL estimations at the end of the sensor recording (which 0-125 cycles before the actual failure) are used.

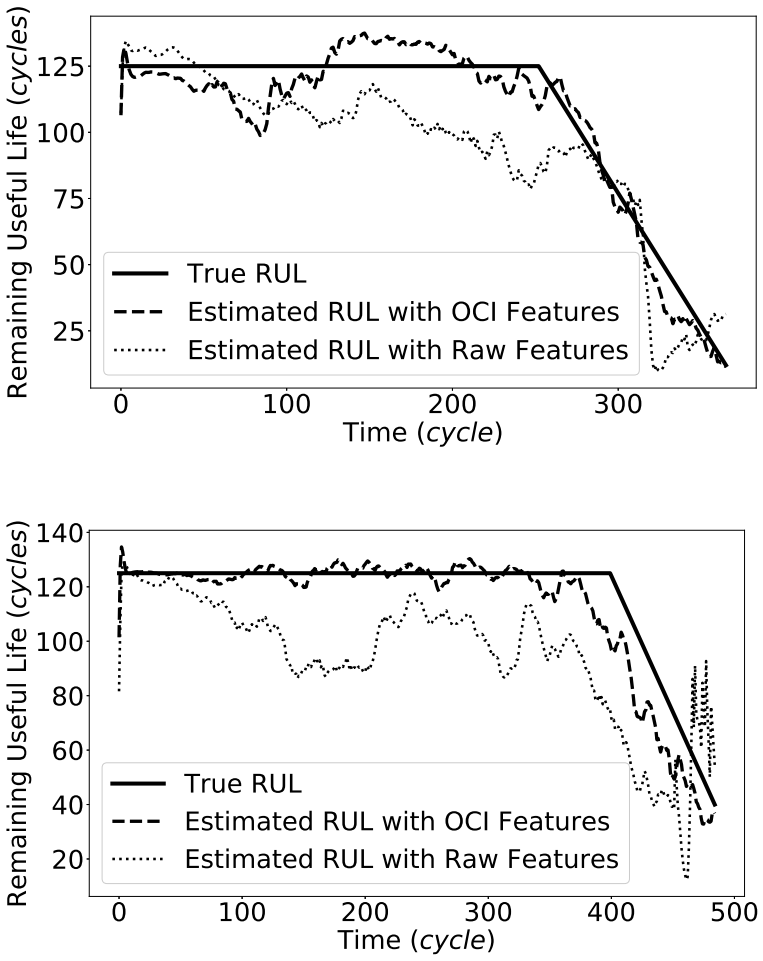


Figure 4.6. Actual and estimated RUL values for two engines. (Top) Engine 64 in FD002 (single fault type-varying operating conditions.) (Bottom) Engine 24 in FD004 (multiple fault types-varying operating conditions.)

Figure 4.6 depicts the true and estimated RUL values for two sample test engines as a function of time-to-failure[2] , from the sets summarized in Figure 4.5. These sample plots demonstrate the higher accuracy of RUL estimation using OCI features, which are almost always better than the estimates based on raw features for these two sample engines. Furthermore, the OCI features based RUL estimations are more stable over the course of the recordings (machine health observations).

[2]Note that the true RUL and time-to-failure are not equal for time-to-failure larger than 125 cycles, which is a common standard employed in RUL estimation literature.

To further show the effectiveness of the OCI features, we have combined the training set of FD001 with FD002 and the training set of FD003 with FD004 since in this combination the fault modes are same and operating conditions are different. The performance of the LSTM RUL estimator is depicted in Table 4.2. As seen in the table, better RMSE performance is obtained in all subsets, because of the increase in the size of the training data. The score performance is also better for all subsets except FD002.

Table 4.2. RMSE and Score Performance of LSTM RUL estimator that is trained on combined training sets on FD001-FD002 and FD003- FD004 subsets with OCI features.

	RMSE (mean±std)	**Score (mean±std)**
FD001	11.75 ± 0.3	240 ± 22
FD002	12.61 ± 0.81	813 ± 84
FD003	13.12 ± 1.27	300 ± 45
FD004	13.33 ± 0.63	874 ± 126

4.3.4. Results with Unknown Operating Conditions

The experiments above assume that the operating conditions are available in the training measurements. However, this assumption may not hold in real life. In this section, we assume that the operating conditions are not available and all consecutive pairs are used as like pairs and the unlike pairs are chosen as stated in Section 4.2.2. The modified binary indicator in (4.4) is employed.

The performance of OCI features with unknown operating conditions on the same data set is illustrated in Table 4.3. The score performance slightly decreases for both single ($FD002$) and multi ($FD004$) fault modes. However, the RMSE performance is enhanced for multiple fault modes whereas it is worsen for the single fault mode.

Table 4.3. RMSE (cycles) and Score Performance of LSTM RUL estimator on FD002 and FD004 subsets with OCI features where the operating conditions are assumed to be unknown.

	RMSE (mean±std)	**Score (mean±std)**
FD002	13.32 ± 1.43	833 ± 125
FD004	14.13 ± 1.73	1148 ± 150

4.4. Discussion

The experimental results show that the MLP based OCI feature extraction (trained with an SNN architecture) improves the performance of deep learning based RUL estimation significantly, in terms of not only RMSE but also the stability and performance scoring. The OCI feature generator MLP is trained to minimize the feature changes in consecutive samples and to maximize the feature differences between starting and ending points of *complete lifespan* sensor measurements. In addition to enhancing performance of individual models, to the best of our knowledge, the proposed OCI feature extraction, together with LSTM RUL estimator, gives the best performance on the commonly used C–MAPSS data set, among the models proposed in the literature so far.

The OCI features were shown to be robust against the destructive effect of operating conditions variations on RUL estimation. This robustness is not only observed in the improvement of RUL estimations but also via direct observation of OCI features in comparison with the raw features as depicted in Figure 4.3 and Figure 4.4. The clear degradation trends observed in OCI features follow the same trends as raw features under single operating condition. Hence, the OCI features look promising as a direct way for monitoring the machinery health, even without an RUL estimator, which would be invaluable in practical applications where the training data (for RUL estimation) is scarce, if not missing. Note that the training of OCI extractor uses *like* and *unlike*

pairs of measurements. Collecting the former is straight forward and does not require to wait until complete failure, where as the latter needs data collected during (or in the vicinity of) complete failure. Hence, the scarcity of *unlike* pairs of measurements, in practice, may pose a risk to OCI features' application. The true effect of having less (or no) *unlike* pairs is yet to be assessed in future work.

The results also revealed that the proposed OCI feature extraction method can be exploited in the absence of operating condition information. Even if a slight decrease was observed in the performance, it was still much better than the usage of raw features. Therefore, OCI features can be used with known and unknown operating conditions.

Our results show that the OCI features significantly improve the RUL estimation under varying operating conditions, for which they were designed. The slight decrease in performance under single operating condition is not expected to hinder the use of OCI features, though under the condition that the machinery will be operated in a single mode, the raw sensor measurements may be preferred.

As expected, the RUL estimators perform better with single fault type than with multiple fault types for all setups, including the raw and OCI features. The OCI feature generator MLP is trained to suppress the destructive effect of variable operating conditions, hence OCI features are not expected to provide a robustness to RUL estimators, against multiple faults. It can be conjectured that training fault type specific OCI generators to be used with fault specific RUL estimators would facilitate improved RUL estimation even in multiple fault cases. It is possible to train and deploy a consortium of OCI generator and RUL estimator pairs to get estimate not only the RUL but also the root cause for failure (diagnostics). Such research requires much richer training data set, yet the training approach would be as proposed in this work.

With regard to the RUL estimator models, our results suggest that the LSTM model is a better fit to the problem than the DCNN model. This is expected as the LSTM networks are more suitable for time series data in the sense of capturing

dependencies in time. The DCNN uses only limited number of past data points, hence has limited memory.

5. IMAGE PROGNOSTICS USING DEEP LEARNING

5.1. Proposed Approach

Popularity of image analytics in predictive maintenance domain is increasing because of the fact that image data provide rich information. The main superiority of imaging is the ease of implementation because the cameras are in general non-contact and do not require permanent installation. Image data, esp. infrared, contains critical information about the health condition of the photographed object. However, the high dimensionality of image data makes it difficult to use it.

The deep learning models can be utilized for this task. However, they are very complex, i.e. the number of the parameters that will be learned is high. Therefore, they require rich training sets to be trained. However, this condition might not be met in prognostics applications since data acquisition in the complete lifespan of the machinery is very difficult in the field and costly in the laboratory environment. This may cause the failure of popular deep learning methods such as multilayer perceptrons, convolutional and recurrent neural networks in generalization. Moreover, the image data is high dimensional and contains redundant data which may impose an extra difficulty in the training.

To overcome the stated problem, we suggested two deep learning based methods for image prognostics [114]. The proposed image prognostics models are depicted in Figure 5.1. In the first model, the spatial information is extracted by two convolutional layers with two pooling layers from the individual images in the degradation streams. The temporal information is captured by an LSTM layer from the outputs of the convolutional part. Then, TTF is regressed as weighted sum of the output states of LSTM cell at each individual time points. This approach addresses the problem of high-dimensionality by avoiding the redundancy in the individual images due to the correlation between neighbor pixels by sharing the same convolutional layers at each

time instant. Then, the LSTM layer captures short and long term temporal information from the lower dimensional space. Moreover, the decrease in the number of parameters may overcome the overfitting where the scarcity of degradation data is an issue.

A deep autoencoder model is designed to decrease the dimension of the individual images in an unsupervised manner in the second method. This helps the overall system to handle high dimensionality problem. Encoder part of the autoencoder transforms the individual image into a lower dimensional space. An LSTM network is trained on this lower dimensional space.

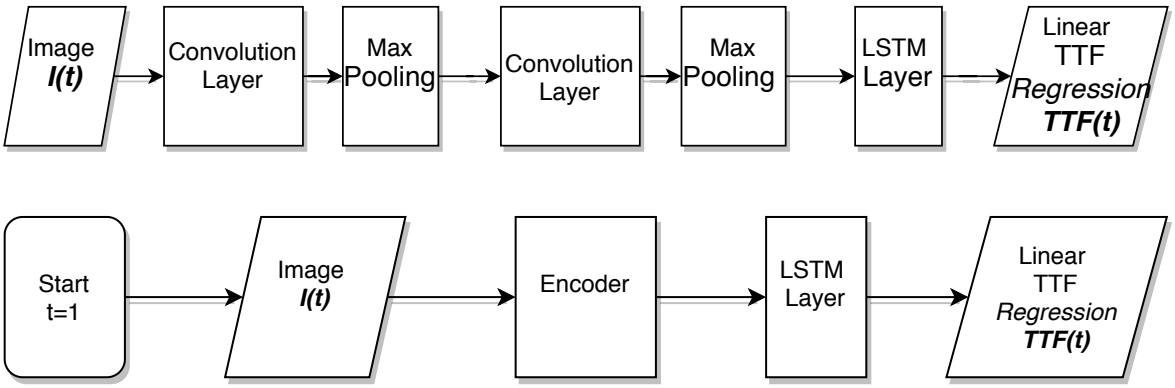


Figure 5.1. Two methods for image prognostics: (Top) Method 1: CNN extracted feature based LSTM TTF estimator. (Bottom) Autoencoder based LSTM TTF estimator.

It is worth to mention that although CNNs gave the best performance in object recognition and tracking tasks [110, 115], they may be efficient in prognostics. The characteristic of a typical infrared degradation image stream is different than the video data, where the effect of the fault is started to be seen at the outer boundaries of the object, and then, diffuses towards the center. Therefore, a fully connected autoencoder architecture may also provide a compact representation of the individual images.

The preliminaries of the utilized deep learning architectures are presented in Section 5.2. Section 5.3 explains the deep learning based image prognostics in detail. The experiments and results are given and discussed in the last section.

5.2. Preliminaries

In this section, the autoencoders, convolutional neural networks and LSTM networks are briefly introduced.

5.2.1. Autoencoders

A compact representation of the samples in a data set can be obtained by using autoencoders. It is trained to copy its inputs to the outputs. The autoencoder may be divided into two parts: an encoder function that maps the input to a code space **h** as:

$$\mathbf{h} = f(\mathbf{x}), \tag{5.1}$$

and a decoder function that reconstructs the input from the mapping or code as:

$$\hat{\mathbf{x}} = g(\mathbf{h}). \tag{5.2}$$

It is hard to select a candidate function and learn its parameters, and hence these two functions are approximated by using an artificial neural network model. Node values in the hidden layers are calculated as a nonlinear activation of linear combination of the values in the preceding layers as:

$$h_{ij} = \sigma\big(\sum_{k=1}^{n} W_{ijk} h_{(i-1)k}\big) \tag{5.3}$$

where h_{ij} represents jth hidden node value in layer i, W_{ijk} represents trainable weight parameter between nodes h_{ij} and $h_{(i-1)k}$, n represents the number of nodes in layer $i-1$. $\sigma(.)$ is the nonlinear activation function. Rectified Linear Unit (ReLU), that is defined as $\sigma(x) = \max(0, x)$, is recommended as the initial candidate for the activation function in convolutional layers [110]. However, ReLU has a drawback when it is learned via the gradient based methods. When a neuron equals to 0, it can not be recovered again from this 0 state because the slope of activation function is 0 when $x < 0$. Leaky ReLu

is proposed to overcome this problem in ReLU activation [116]. LeakyReLU is defined as:

$$\sigma(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x & \text{else} \end{cases} \tag{5.4}$$

where α is a constant much smaller than 1. Autoencoders are an effective nonlinear extension of principal component analysis (PCA) when both of the encoder function f and the decoder function g are nonlinear. For this reason deep autoencoders are mostly constructed with a set of hidden layers with nonlinear activation both in encoder and decoder parts.

However, if the number of parameters is high and the training set contains the limited number of samples, the autoencoder may overfit the training data [110]. Therefore, in some applications it is more reasonable to adopt a nonlinear deep encoder and a linear one-layer decoder. In this case, overall network can be seen as kernel PCA since x is regenerated as linear combination of h which is a nonlinear function of the input [110].

Autoencoders are generally trained with gradient descent based algorithms with mean squared error loss function that is defined as,

$$C(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||_2^2. \tag{5.5}$$

Backpropagation is utilized for the computation of the gradients of the parameters [117].

5.2.2. Convolutional Neural Networks

Convolutional neural network (CNN), which is first suggested in [118], is by far the most used approach in machine vision. It is named as convolutional because of

the *convolution* operation that is adopted instead of the matrix multiplication in other feed-forward neural networks. The convolution for discrete functions is described as:

$$s[n] = (x * w)[n] = \sum_{k=-\infty}^{\infty} x[n-k]w[k] \tag{5.6}$$

where x is input and w is weight vector or kernel. The convolution may be expanded to 2-D by using 2-D kernel K as:

$$S[i,j] = (I * K)[i,j] = \sum_{m}\sum_{n} I[i-m, j-n]K[i,j]. \tag{5.7}$$

There are two principal ideas behind the CNNs: 1) Sparse interactions: each output unit only interacts with a subset of the input neurons in contrast to classical fully connected networks. 2) Parameter sharing: Same kernels are used for different parts of the input(image) which reduces the number of parameters to be learned.

The convolution is followed by a nonlinear activation (the Leaky ReLU that is defined in (5.4) is preferred in this thesis). The nonlinear activation is also known as detector, because when the convolution output of a kernel for a region in the input generates a nonzero value on this activation function it is interpreted as an information content is available in this region.

Another key layer in the CNN is pooling function. Pooling layer in a CNN replaces the outputs of the neurons at a certain location of the preceding layer with the summary statistics of these outputs. Maximum pooling is predominantly preferred in machine vision because the accurate localization of the object is not of interest and strong computational efficiency is achieved in the training. Moreover, the network becomes invariant to small translations in the input with the pooling. However, in some image processing tasks, each activation is important, and thus the average pooling is preferred instead of maximum pooling, in order to get better summary of preceding convolutional layers.

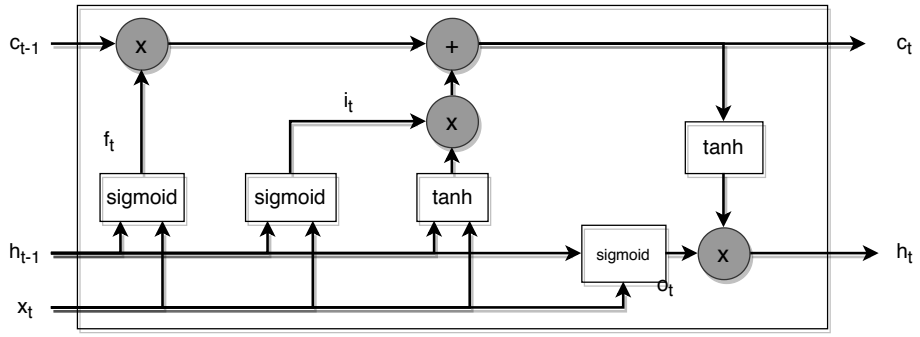


Figure 5.2. LSTM Cell.

5.2.3. Long-Short Term Memory Networks

The TTF is generated based on the output states of the LSTM layer. It is a subclass of recurrent neural networks (RNNs) and RNN is a special type of feed-forward neural network that is used for sequence processing. RNN has a state parameter, namely memory unit, to remember previous inputs, and hence is capable of capturing temporal patterns from the sequential input.

Among other types of RNN models, LSTM networks are of special interest because of their capability of modeling short and long term dependencies [110, 119]. An LSTM cell with a forget gate is illustrated in Figure 5.2.

In an LSTM cell, there are two separate state vectors, $c(t)$ and $h(t)$ which are known as cell state vector and output state vector, respectively. The states are calculated with the following equations:

$$
\begin{aligned}
f(t) &= \sigma_g(W_f x(t) + U_f h(t-1) + b_f) \\
i(t) &= \sigma_g(W_i x(t) + U_i h(t-1) + b_i) \\
o(t) &= \sigma_g(W_o x(t) + U_o h(t-1) + b_o) \\
c(t) &= f(t) \circ c(t-1) + i_t \circ \sigma_c(W_c x(t) + U_c h(t-1) + b_c) \\
h(t) &= o(t) \circ \sigma_h(c(t))
\end{aligned} \tag{5.8}
$$

where $\circ$ is element-wise product and t is time index. f, i, o are known as forget gate's, input gate's and output gate's activations, respectively. In general, sigmoid function is used for σ_g and hyperbolic tangent is preferred for σ_c and σ_h. The cell and output states are initialized as $c(0) = 0$ and $h(0) = 0$. If the data of interest include sequences that have complex temporal information with short term and long term dependencies, it can be modeled by LSTM networks. LSTM cell can manage both short-term and long-term relations by its input and forget gates.

LSTMs or in general RNNs, are trained by gradient descent based methods as the other neural networks. However, because of its recursive structure, back-propagation algorithm is applied through time, and it is named as backpropagation through time (BPTT) [120]. At first, the network is unrolled through time, and then partial derivatives of the loss function with respect to trainable parameters at each time index are calculated separately with BPTT. Parameters are updated using aggregating those gradients by summation or averaging in mini batches.

5.3. Deep Learning Based Image Prognostics

The aim of our thesis is to estimate and update the TTF value of a system from its degradation image stream. Two different approaches are proposed to estimate TTF of industrial systems from its image sequence:

5.3.1. Model 1: Integrated CNN-LSTM Based TTF Prediction

First model is a special type of neural network in which two time distributed convolutional layers precede an LSTM layer as shown in Figure 5.3. The LSTM layer outputs are fed into a fully connected linearly activated layer for TTF calculation. A TTF value is produced at each time instant t after a burn-in period.

All of the convolutional, LSTM and fully connected layer parameters are learned by using BPTT algorithm. TTF estimation is a regression problem rather than a

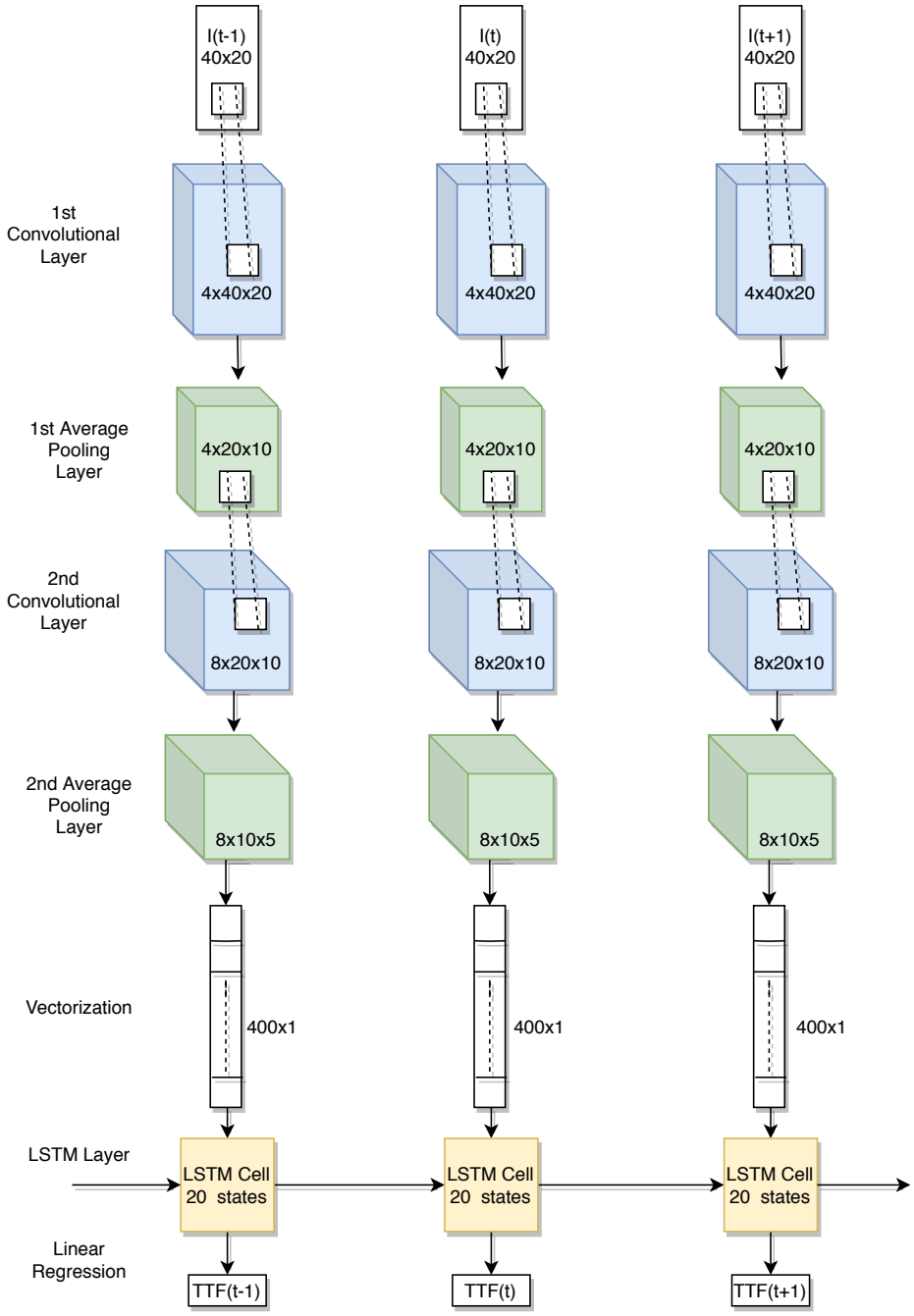


Figure 5.3. Convolutional LSTM Network.

classification problem, and mean squared error is selected as cost. However, in our TTF prediction problem, a weighted average of squared error is adopted as the training

cost function that is calculated for a sample as follows:

$$C = \sum_{k=t_b}^{t_f} \left| k(TTF_i - \hat{TTF}_i) \right|^2 \tag{5.9}$$

where t_b and t_f are burn-in and failure time respectively. The principal idea behind this cost function is that the number of samples, and hence the amount of information that is used in TTF estimation, increases with time. Moreover, the degradation becomes more pronounced as time progresses. This cost makes TTF estimations to be biased for being more accurate as the time progresses, i.e., the failure approaches.

The most challenging task in deep learning is that there is no systematic way of tuning hyperparameters of the models. A heuristic approach is adopted in this research to select hyperparameters. First, kernel sizes for convolutional layer and pooling layer are intuitively set to 3×3 and 2×2, respectively, because the images in our data set are 40×20. As opposed to the image recognition, in our application, CNN layers are expected to summarize the information in all areas of the individual images, and larger kernel sizes for convolutional and pooling layers may adversely affect the performance. Then, a network is created with a single convolutional and a single LSTM layer. Number of kernels and cell state size in the LSTM layer are optimized iteratively by updating one of them and fixing the other. The initial values are selected so large that the model overfits the training data, and decreased down to a point where the performance of the model decreases.

Once the number of kernels and cell state size are selected, an extra CNN layer is inserted to the model but the number of kernels is halved. It is observed that this extra convolutional layer significantly enhances the performance, however, adding a third convolutional layer decreases the performance even below the single convolutional layer. So, the number of layers is selected as 2 for the CNN part. Furthermore, after optimizing the CNN parameters, a second LSTM layer is added to the model. Adding an LSTM layer does not improve the results, and hence it is decided to employ a single LSTM layer in the model. The overall model that is obtained has 2 convolutional

layers that has 4 and 8 3×3 convolution kernels, respectively, and 2×2 average pooling kernel. It also has a standard LSTM layer that has 5 neurons in its cell state. Adam optimization technique is used in the training with the parameters as $learningrate = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

5.3.2. Model 2: Integrated Autoencoder-LSTM TTF Prediction

In the second model, two separate neural networks are designed. First, dimension of the images is decreased using a multilayer perceptron which is trained as an autoencoder. The autoencoder is trained with the individual vectorized images in the training set, i.e., temporal information is totally ignored for this part. The redundancy in the spatial domain is aimed to be eliminated. Then, the encoder part of the autoencoder is utilized for transforming the images both in the training and test sets to a lower dimensional space. A single layer LSTM network is used to predict TTF values of the systems using feature streams generated by the autoencoder at each time instant after a burn in period. The TTF estimation using this method is presented in Figure 5.4.

The loss function defined in (5.9) is used in the training of the LSTM.

The hyperparameters of autoencoder and LSTM models are chosen by a heuristic approach that is similar to the one in Section 5.3.1. First, a single hidden layer autoencoder is designed, and the number of neurons in the hidden layer is decreased from a large number where the model overfits the data. Then, an extra hidden layer is added to the encoder part, while the number of neurons in the layers are selected to be half of the value in the previous model. The reconstruction error is reduced to almost half by addition of the extra layer. However, adding more layers in both coding and decoding parts, while fixing the number of feedforward operations leads to an increase in the reconstruction error. Thus, a model with two encoding layers and one decoding layer is preferred. Moreover, the number of neurons in the hidden layers are optimized in an iterative way as discussed in Section 5.3.1. The trained model has 200 and 20 neurons in its hidden layers, respectively. The LSTM cell state is fixed exactly to the one in

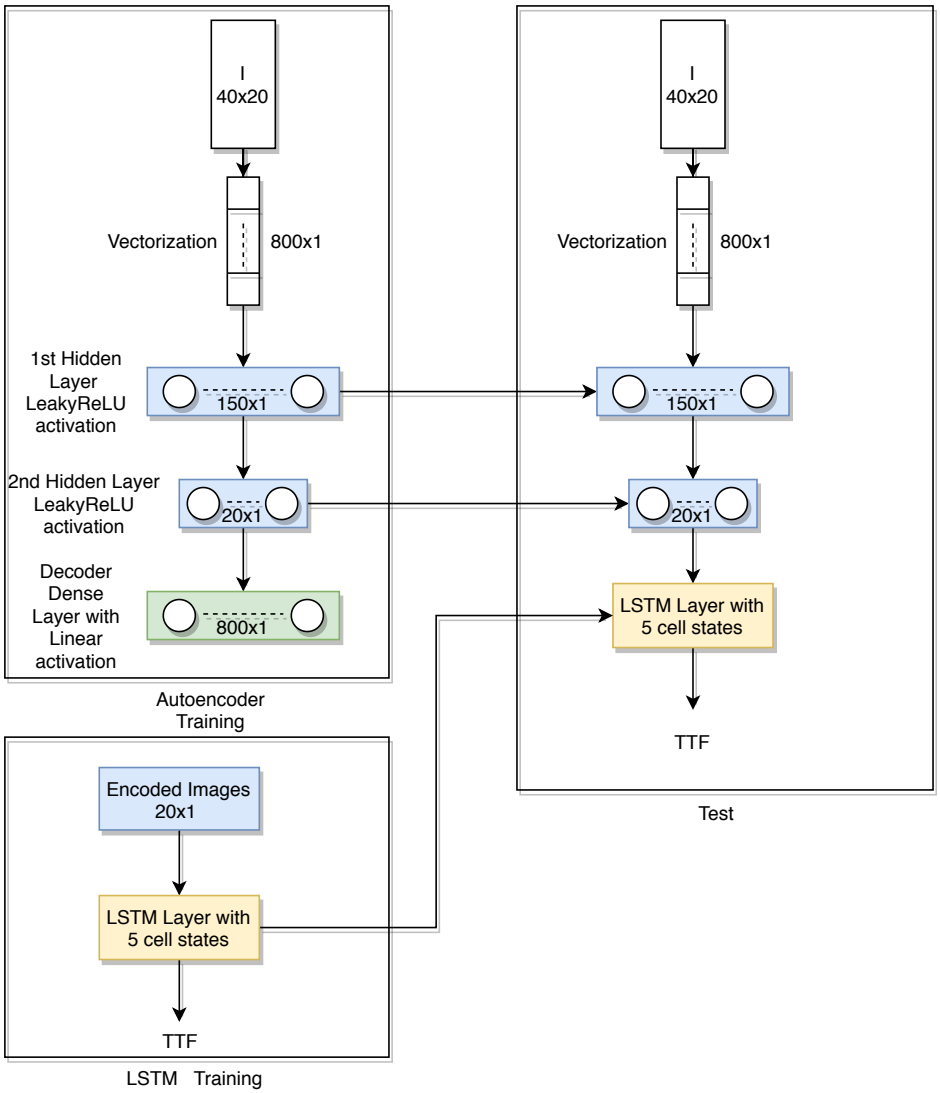


Figure 5.4. Autoencoder based feature extraction and LSTM based TTF estimation.

integrated CNN-LSTM model (i.e., 5) for a fair comparison. Adam algorithm is used in the training with the parameters $learningrate = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$.

5.4. Experiments

In this section, the proposed methods are verified by using both a simulated degradation image stream data and a case study.

5.4.1. Simulation Study

In this part, degradation image streams are generated based on a heat transfer process as in [51]. Let $I(x, y, t)$ represent the pixel value at position (x, y) at time t. The evaluation of the pixel values is governed by the following equation:

$$\frac{\partial I}{\partial t} = \alpha\Big(\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}\Big) \tag{5.10}$$

where α is the diffusivity coefficient. The diffusivity constant α defines the rate of degradation, and, so determines the TTF value. Coordinate values are set to $0 \leq x, y \leq 40$, and the images are recorded at integer values of (x, y). Initially, pixel values are set to 0, other than the boundary conditions which are set to 1 for all t. Two types of noise are injected to images. First, a noise that is generated by a spatial Gaussian process with mean $\mu = 0$ and covariance function $K((x_1, y_1), (x_2, y_2)) = \sigma^2 exp\big(-\phi\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}\big)$ where $\sigma^2 = 0.01$ and $\phi = 0.25$. The second type of noise is independent and identically distributed Gaussian noise with variance $\sigma^2 = 0.02^2$. This leads to images of size 41×41. 500 samples of length 100 are constructed where α is randomly generated from a uniform distribution $U(0.5 \times 10^{-4}, 10^{-4})$. An example subset of images (at time instants $t = 20, 40, 60, 80, 100$) with and without noise is illustrated in Figure 5.5.

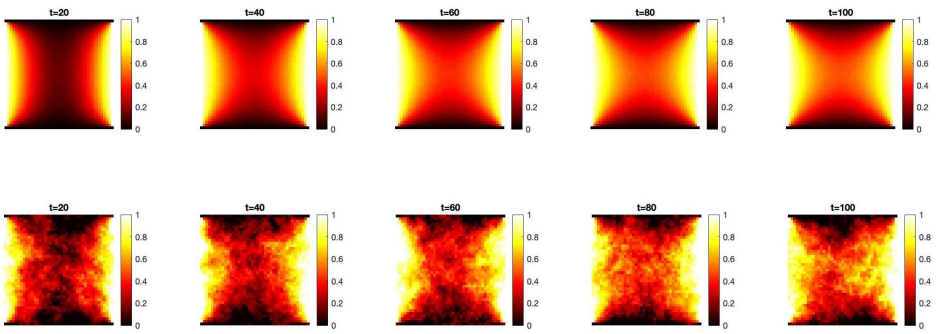


Figure 5.5. Simulated image streams: (Top) Without noise. (Bottom) With noise.

To show the generalization capability of our methods, TTF values are assigned in two different manners. In the first case, diffusivity constant α is set as TTF. In

the second case, the TTF of system i is defined as a nonlinear function of diffusivity constant as:

$$TTF_i = 10 - k\alpha_i^2 \tag{5.11}$$

where k is a constant such that $TTF_i > 0$.

To assess the efficiency of our methodologies, we present their performance in comparison to various approaches. First approach that is used as benchmark is the log-linear scale tensor regression algorithm that have been studied in [51]. The authors used two tensor decomposition techniques, CANDECOMP/PARAFAC and Tucker decompositions with multi-linear principal component analysis (MPCA) to reduce the dimensionality of the tensor regression. For the sake of simplicity, we only replicate their algorithm with Tucker decomposition using MPCA, since it performs slightly better than CANDECOMP/PARAFAC in the simulation data.

Secondly, a deep LSTM network is used for benchmarking. However, the LSTM network has been overfitted to the training data because of the high dimensional input streams. To avoid the overfitting, the LSTM network is trained with 90% dropout between consecutive LSTM layers and between the input and first LSTM layer in the training [121]. Dropout simply means randomly setting some input, state or output values of the LSTM layers to zero in the forward pass. This provides regularization for the network, and prevents overfitting.

Finally, we used a deep CNN as a benchmark model. However, a standard CNN does not work with varying length sequences. Moreover, its computational time increases quadratically with the length of the sequence, and this may lead to a non-scalable network. These issues are solved by adopting a sliding time window approach as in [71]. At each time t, $TTF(t)$ is calculated by feeding the last L images to the network. In addition, CNN is employed in 3-D since our image streams are 3-D tensors.

The parameters of the proposed networks are directly set to values represented in Figure 5.3 and Figure 5.4. Burn-in period is defined as 20% of length of the stream, i.e., 20.

5-fold cross validation is used for the tests, where we have 400 training and 100 test samples in each fold, and mean of the results are presented. The performance is presented in terms of absolute estimation error which is calculated as:

$$\text{Prediction Error} = \frac{|\text{Estimated TTF} - \text{Real TTF}|}{\text{Real TTF}}. \tag{5.12}$$

Figure 5.6 demonstrates the performance comparison of LSTM network with dropout, 3-D CNN with sliding time window, tensor regression approaches and our methods, namely convolutional LSTM network and autoencoder code based LSTM network for the first case where $TTF = \alpha$. Median absolute prediction errors (and interquartile range) are 1.5%(2%) and 1.7%(1.6%) for convolutional LSTM and autoencoder based LSTM networks, and 2.2%(2%), 3%(3.4%) and 9.9%(9.8%) for tensor regression, LSTM with dropout and 3-D deep CNN methodologies, respectively. Our methods outperform LSTM with dropout and 3-D deep CNN methods, and performs slightly better than tensor regression methods.

Figure 5.7 illustrates the performance comparison of the benchmark and our methods for the second case where TTF value are set as a nonlinear function of the diffusion parameter, α. Median absolute prediction errors (and interquartile range) are 3.7%(5.6%) and 2.2%(4%) for convolutional LSTM and autoencoder based LSTM networks, and 12%(16%), 4%(9.7%) and 20%(20%) for tensor regression, LSTM with dropout and 3-D deep CNN methodologies, respectively. Our methods perform much better than the tensor regression in the nonlinear TTF case as well as 3-D CNN and LSTM with dropout in contrast to first case.

It is observed that LSTM networks capture the temporal information in image

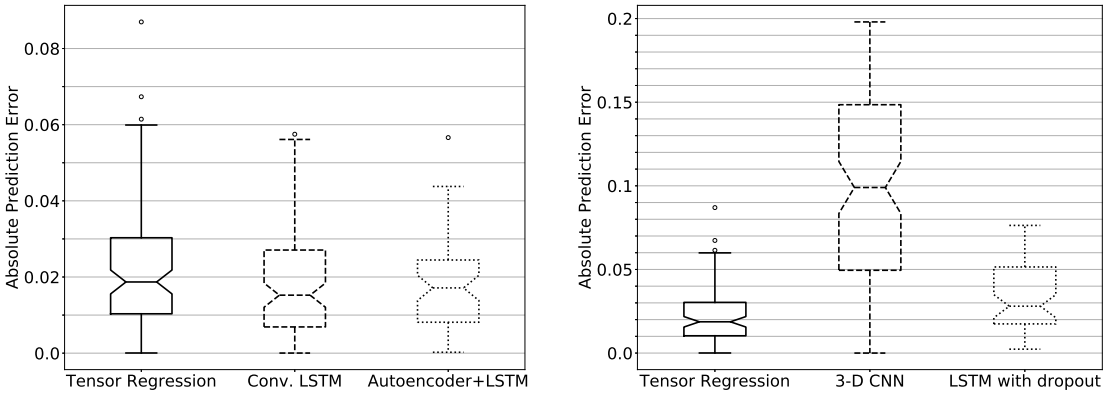


Figure 5.6. Prediction errors of the proposed and benchmark methods on simulated data: (Left) Comparison of convolutional LSTM with 3D deep CNN and LSTM with dropout. (Right) Comparison of proposed methods with tensor regression.

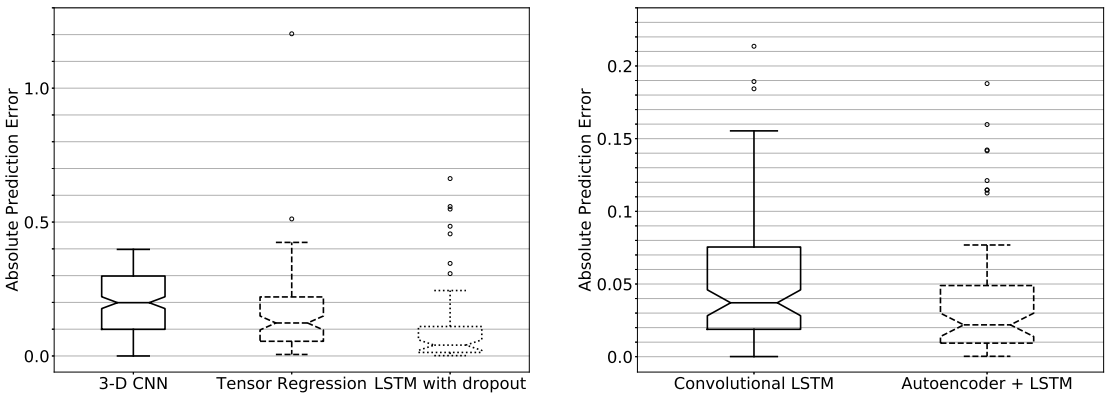


Figure 5.7. Prediction errors of the proposed architectures and benchmark methods on simulated data with nonlinear TTF: (Left) Tensor regression, 3-D deep CNN and LSTM with dropout. (Right) Convolutional LSTM and Autoencoder based LSTM.

streams without making any assumptions on the distribution of the TTF value. However, if the dimension of the input is high and the number of samples is low, the model overfits the data. Although the high dropout rates (90% in the simulation study) may prevent overfitting as demonstrated in Figure 5.6 and Figure 5.7, they may lead to underfitting. Dimension reduction techniques which can model the spatial correlation of

the image pixels improve the results substantially. The proposed architectures model the individual images in two different ways. It is also shown that deep CNN with a sliding time window cannot model the complex temporal relations in the image sequences. Tensor regression also failed in TTF prediction for the nonlinear case, because the model assumes a log-linear relation between image streams and TTF values.

The computational complexity of the proposed models in the test is linear with time and input dimension which makes them practically applicable in industry. Experiments are conducted using Tensorflow-GPU [122] on NVIDIA Quadro K4000 graphics board. Their total test time on simulation data for one sample $t = 1, 2, ..., 100$ are presented in comparison to the benchmark methods in Table 5.1.

Table 5.1. Computational time of the proposed image prognostics models and benchmarking methods.

Method	**Time (milisecond)**
Convolutional LSTM	7.7
Autoencoer based LSTM	7
LSTM with dropout	6.2
Deep CNN	104
Tensor regression	54000

5.4.2. Case Study

In this section, we evaluate the performance of the proposed techniques in a case study. The data used in this article is collected from rotational element thrust bearing by using the test bed presented in [40]. Four experiments are conducted from the brand new state to failure. The degradation streams consist of 375, 611, 827 and 1478 images, respectively. The number of samples is increased by resampling the original image streams accordingly, and a total of 284 image data streams are generated as suggested in [51] for a fair comparison. The lengths of generated sequences vary between 17 and 55. An example of the image streams are illustrated in Figure 5.8.

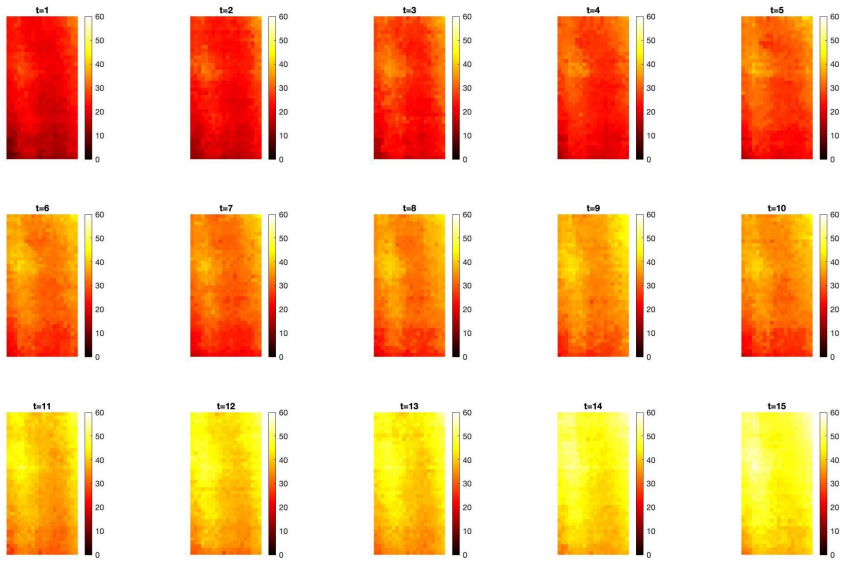


Figure 5.8. An example of degradation image stream.

The parameters of the suggested neural networks are set as in Figure 5.3 and Figure 5.4. The burn in time is set to 20% of the shortest sequence length which is 3. Absolute percentage error that is described in (5.12) is used as performance metric.

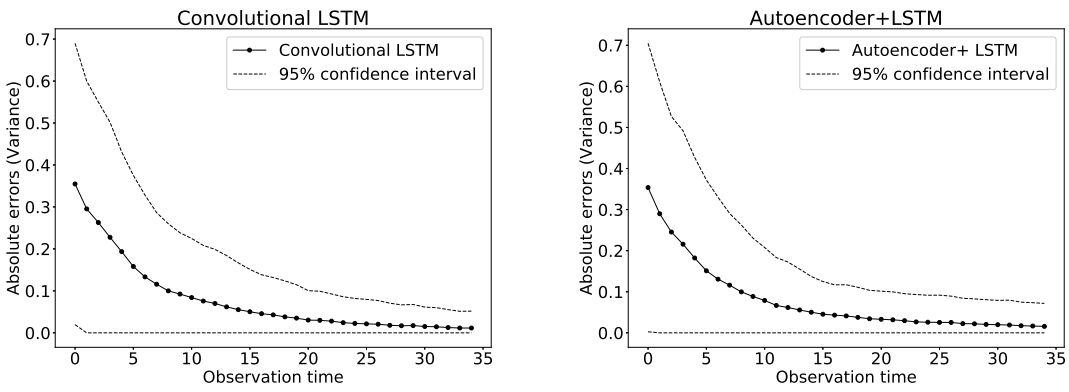


Figure 5.9. Prediction errors of with 95% confidence interval: (Left) Convolutional LSTM. (Right) Autoencoder based LSTM.

Figure 5.9 shows the performance of our methods, namely convolutional LSTM

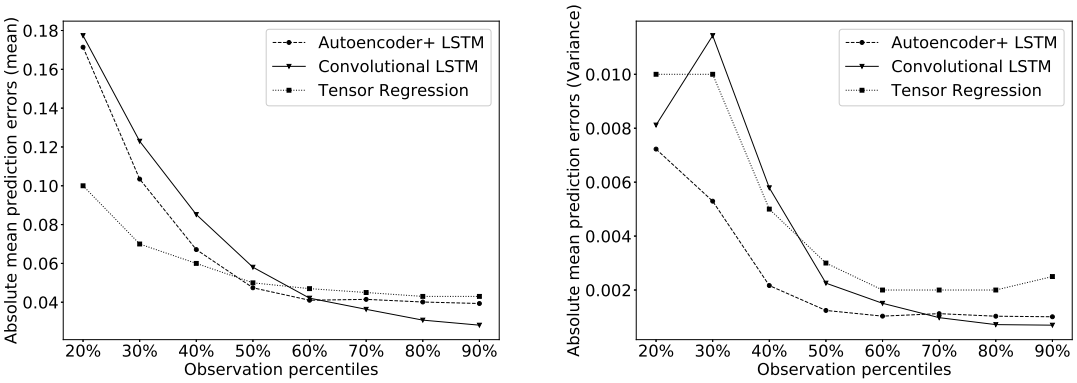


Figure 5.10. Absolute prediction errors. (Left) Mean. (Right) Variance.

network and autoencoder code based LSTM network with 95% confidence interval with respect to the number of observations available. The results show that the mean of the proposed architectures increases as time progresses, as expected. Moreover, the error interval also decreases with time.

Figure 5.10 compares the performance of our methods with the tensor regression with respect to the observation percentiles. Only the tensor regression method is used as a benchmark, because deep CNN and LSTM with dropout methodologies perform far below the proposed methods in the case study data. The mean (variance) of absolute prediction errors for tensor regression, convolutional LSTM and autoencoder based LSTM networks are 0.05%(0.003%), 0.058%(0.002%) and 0.047%(0.001%) at 50^{th} percentile and 0.047%(0.003%), 0.042%(0.002%) and 0.041%(0.001%) at 60^{th} percentile. Tensor regression is slightly better at lower percentiles and worse at higher percentiles than the proposed approaches.

5.5. Discussion

Two novel deep learning based image prognostics models are proposed in this chapter. Both models divide the problem into the spatial and temporal parts to cope with the high dimensionality problem. In the first model, spatial information is ex-

tracted by a convolutional network, whereas an MLP which is trained as autoencoder is used for the same purpose in the second. Both models have an LSTM layer which estimates TTF of the component.

Both simulation and real case experiment results suggest that the proposed architectures can be used in the industrial applications. Both models perform better than the conventional approaches after the 40^{th} percentiles of the machinery lifespan. The main disadvantage of the proposed methods is that when the number of samples/observations is low, they perform worse than the conventional approaches. On the other hand, complex temporal information in the long sequences with high dimensional inputs can be modeled with the models that we propose.

Moreover, the computational complexity of the proposed methodologies in the test part is scalable, hence they are suitable for real life applications. However, the training time of the proposed deep learning models are far worse than the conventional approaches such as tensor regression.

In the simulation experiments, it is also observed that the proposed models can capture not only the linear/log-linear correlation between image stream and TTF but also the nonlinear ones. The conventional approaches such as $3-D$ CNN and tensor regression, fail to generate accurate TTF estimations in the nonlinear case. This property may lead to the usage of the proposed models in the other types of high dimensional sensor data, such as profile data.

Another interesting result is that the autoencoder based LSTM is better at lower percentiles and convolutional LSTM is better at higher percentiles. The autoencoder's denoising effect on the individual image may lead to this behavior. In the lower percentiles LSTM with denoised lower dimensional space capture temporal structure more easily. On the other hand, the convolutional LSTM outperforms autoencoder based strategy in higher percentiles because of its supervised feature extraction design.

6. CONCLUSION

This thesis focuses on three main problems in the domain of industrial machinery prognostics and health management. First, a novel anomaly triggered remaining useful life (AT-RUL) estimation scheme is proposed in this thesis. The CUSUM control chart is employed as anomaly detector, and various different ML models are used as the RUL estimator. The positive impact of incorporating anomaly detection to RUL estimation is demonstrated using in-house simulations and publicly available C-MAPSS turbofan engine degradation data. However, AT-RUL estimation scheme is only employed for the engines which operate under non-varying operating conditions. The anomaly detection for the sensor data of the engines with varying operating conditions is a more challenging problem. The state-of-art anomaly detection algorithms fail to provide the degradation onsets in the degradation data. Therefore, we plan to develop an anomaly detection model that is suitable for varying operating conditions and demonstrate the effectiveness of anomaly triggering in RUL estimation also under varying operating conditions.

Secondly, a novel deep learning based operating condition invariant (OCI) feature extraction method is proposed in this thesis. The proposed OCI feature extractor is trained using a siamese neural network architecture. The effectiveness of OCI features in RUL estimation is illustrated using C-MAPSS turbofan engine degradation data set. Degradation recordings from a healthy situation to the complete failure are used for the training of OCI feature extractor. However, this model lacks of exploiting censored samples, i.e., the samples with a limited degradation, not a complete failure. Extension of the OCI feature extraction to the censored samples is reserved for the future research. Moreover, the training procedure assumes that the operating conditions may change at discrete time points. The performance under continuous change of conditions needs to be assessed. Extension of the proposed approach to the machinery with continuously varying operating conditions is another future research topic.

Lastly, two deep learning models are proposed for the image prognostics. First architecture has two convolutional layers which extract the spatial information from the individual images, and an LSTM layer which traces the temporal information from the outputs of the last convolutional layer. A deep autoencoder is used to obtain the representation of the individual images instead of convolutional layers, and an LSTM network is used to estimate TTF of the system using this representation. The proposed image prognostics models are verified using a simulation and a case study data.

Different types of degradation data are utilized in all three parts of the thesis. A wholistic approach that utilizes different types of sensors such as image, vibration, pressure, temperature, etc. at the same time may improve the performance. However, when the properties of the sensor data such as sampling frequency, probability distribution, etc. are different, fusion of the degradation data may be challenging. We plan to develop a wholistic approach that combines the sensor data with different characteristics to estimate the RUL of the industrial machinery as a future work.

We believe that the proposed AT-RUL estimation model together with OCI features encourages the practitioners to employ them in industry by providing them a scalable, computationally lightweight and environment/operating conditions-free prognostic tool. In addition, suggested image prognostics model can be employed in any high dimensional data, such as profile data, and it can be efficiently implemented in industry. One can implement the proposed RUL estimation models on cloud or edge computing devices. In the test phase, all of the architectures are computationaly lightweight and scalable with time and number of monitored devices, and hence they can be run on commercial inexpensive microcontroller boards. However, their training time with such devices may be time consuming. Therefore, the models can be trained on onsite CPUs/GPUs or cloud computing platforms. Moreover, with the help of a small storage designed as ring memory unit, some of the retrospective data can be recorded. The samples in which the proposed models perform poorly can be sent to the computing platform for further training and fine tuning of the models.

REFERENCES

1. *INDUSTRIE 4.0–Smart Manufacturing for the Future*, GTAI (Germany Trade &Invest), Berlin, 2014.

2. Brettel, M., N. Friederichsen, M. Keller and M. Rosenberg, "How Virtualization, Decentralization and Network Building Change the Manufacturing Landscape: An Industry 4.0 Perspective", *International Journal of Information and Communication Engineering*, Vol. 8, No. 1, pp. 37–44, Nov. 2014.

3. Xu, L. D., E. L. Xu and L. Li, "Industry 4.0: state of the art and future trends", *International Journal of Production Research*, Vol. 56, No. 8, pp. 2941–2962, Apr. 2018.

4. Frank, A. G., L. S. Dalenogare and N. F. Ayala, "Industry 4.0 technologies: Implementation patterns in manufacturing companies", *International Journal of Production Economics*, Vol. 210, pp. 15–26, Apr. 2019.

5. "Industrie 4.0 maturity index", G. Schuh, R. Anderl, G. J., M. ten Hompel and W. Wahlster (Editors), *In: Managing the Digital Transformation of Companies (Acatech Study) Herbert Utz Verlag*, Munich, 2017.

6. Balogun, O. O. and K. Popplewell, "Towards the integration of flexible manufacturing system scheduling", *International Journal of Production Research*, Vol. 37, No. 15, pp. 3399–3428, Oct. 1999.

7. Roblek, V., M. Meško and A. Krapež, "A Complex View of Industry 4.0", *SAGE Open*, Vol. 6, No. 2, pp. 1–11, Apr. 2016.

8. Wang, S., J. Wan, D. Li and C. Zhang, "Implementing Smart Factory of Industrie 4.0: An Outlook:", *International Journal of Distributed Sensor Networks*, Vol. 12, pp. 1–10, Jan. 2016.

9. Tao, F., Q. Qi, A. Liu and A. Kusiak, "Data-driven smart manufacturing", *Journal of Manufacturing Systems*, Vol. 48, pp. 157–169, Jul. 2018.

10. Jeschke, S., C. Brecher, T. Meisen, D. Özdemir and T. Eschert, "Industrial Internet of Things and Cyber Manufacturing Systems", S. Jeschke, C. Brecher, H. Song and D. B. Rawat (Editors), *Industrial Internet of Things: Cybermanufacturing Systems*, Springer Series in Wireless Technology, pp. 3–19, Springer International Publishing, Cham, 2017.

11. Porter, M. E. and J. E. Heppelmann, "How Smart, Connected Products Are Transforming Competition", *Harvard Business Review*, Nov. 2014.

12. Lu, Y., "Industry 4.0: A survey on technologies, applications and open research issues", *Journal of Industrial Information Integration*, Vol. 6, pp. 1–10, Jun. 2017.

13. Pfohl, H.-C., B. Yahsi and T. Kurnaz, "Concept and Diffusion-Factors of Industry 4.0 in the Supply Chain", M. Freitag, H. Kotzab and J. Pannek (Editors), *Dynamics in Logistics*, Lecture Notes in Logistics, pp. 381–390, Springer International Publishing, Cham, 2017.

14. Vaidya, S., P. Ambad and S. Bhosle, "Industry 4.0 – A Glimpse", *Procedia Manufacturing*, Vol. 20, pp. 233–238, Jan. 2018.

15. Stock, T. and G. Seliger, "Opportunities of Sustainable Manufacturing in Industry 4.0", *Procedia CIRP*, Vol. 40, pp. 536–541, Jan. 2016.

16. Lasi, H., P. Fettke, H.-G. Kemper, T. Feld and M. Hoffmann, "Industry 4.0", *Business & Information Systems Engineering*, Vol. 6, No. 4, pp. 239–242, Aug. 2014.

17. Yin, S. and O. Kaynak, "Big Data for Modern Industry: Challenges and Trends [Point of View]", *Proceedings of the IEEE*, Vol. 103, No. 2, pp. 143–146, Feb. 2015.

18. James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, and Angela Hung Byers, "Big data: The next frontier for innovation, competition, and productivity | McKinsey", Library Catalog: www.mckinsey.com.

19. Qi, Q. and F. Tao, "Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0: 360 Degree Comparison", *IEEE Access*, Vol. 6, pp. 3585–3593, 2018.

20. Nedelcu, B., "About Big Data and its Challenges and Benefits in Manufacturing", *Database Systems Journal*, Vol. 4, No. 3, pp. 10–19, 2013.

21. Addo-Tenkorang, R. and P. T. Helo, "Big data applications in operations/supply-chain management: A literature review", *Computers & Industrial Engineering*, Vol. 101, pp. 528–543, Nov. 2016.

22. Lee, I., "Big data: Dimensions, evolution, impacts, and challenges", *Business Horizons*, Vol. 60, No. 3, pp. 293–303, May 2017.

23. Lee, J., H.-A. Kao and S. Yang, "Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment", *Procedia CIRP*, Vol. 16, pp. 3–8, Jan. 2014.

24. Lee, J., H. D. Ardakani, S. Yang and B. Bagheri, "Industrial Big Data Analytics and Cyber-physical Systems for Future Maintenance & Service Innovation", *Procedia CIRP*, Vol. 38, pp. 3–7, Jan. 2015.

25. O'Donovan, P., K. Leahy, K. Bruton and D. O' Sullivan, "An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities", *Journal of Big Data*, Vol. 2, p. 25, Nov. 2015.

26. Lee, J., H. Davari, J. Singh and V. Pandhare, "Industrial Artificial Intelligence for industry 4.0-based manufacturing systems", *Manufacturing Letters*, Vol. 18,

pp. 20–23, Oct. 2018.

27. Wang, J., W. Zhang, Y. Shi, S. Duan and J. Liu, "Industrial Big Data Analytics: Challenges, Methodologies, and Applications", *arXiv:1807.01016 [cs]*, Dec. 2018, `http://arxiv.org/abs/1807.01016`.

28. Salonen, A. and M. Deleryd, "Cost of poor maintenance: A concept for maintenance performance improvement", *Journal of Quality in Maintenance Engineering*, Vol. 17, No. 1, pp. 63–73, Jan. 2011.

29. Lee, J., B. Bagheri and H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems", *Manufacturing Letters*, Vol. 3, pp. 18–23, Jan. 2015.

30. Guillén, A. J., A. Crespo, M. Macchi and J. Gómez, "On the role of Prognostics and Health Management in advanced maintenance systems", *Production Planning & Control*, Vol. 27, No. 12, pp. 991–1004, Sep. 2016.

31. Sutharssan, T., S. Stoyanov, C. Bailey and C. Yin, "Prognostic and health management for engineering systems: a review of the data-driven approach and algorithms", *The Journal of Engineering*, Vol. 2015, No. 7, pp. 215–222, 2015.

32. Vogl, G. W., B. A. Weiss and M. Helu, "A review of diagnostic and prognostic capabilities and best practices for manufacturing", *Journal of Intelligent Manufacturing*, Vol. 30, No. 1, pp. 79–95, Jan. 2019.

33. Atamuradov, V., K. Medjaher, P. Dersin, B. Lamoureux and N. Zerhouni, "Prognostics and health management for maintenance practitioners - Review, implementation and tools evaluation", *International Journal of Prognostics and Health Management*, Vol. 8, No. 060, pp. 1–31, Dec. 2017.

34. Tsui, K.-L., N. Chen, Q. Zhou, Y. Hai and W. Wang, "Prognostics and Health Management: A Review on Data Driven Approaches", *Mathematical Problems in*

Engineering, Vol. 2015, pp. 1–17, May 2015.

35. Zio, E., "Prognostics and Health Management of Industrial Equipment", *Diagnostics and Prognostics of Engineering Systems: Methods and Techniques*, pp. 333–356, IGI Global, Hershey, PA, USA, 2012.

36. Elattar, H. M., H. K. Elminir and A. M. Riad, "Prognostics: a literature review", *Complex & Intelligent Systems*, Vol. 2, No. 2, pp. 125–154, Jun. 2016.

37. Lee, J., F. Wu, W. Zhao, M. Ghaffari, L. Liao and D. Siegel, "Prognostics and health management design for rotary machinery systems—Reviews, methodology and applications", *Mechanical Systems and Signal Processing*, Vol. 42, No. 1, pp. 314–334, Jan. 2014.

38. Zhao, R., R. Yan, Z. Chen, K. Mao, P. Wang and R. X. Gao, "Deep learning and its applications to machine health monitoring", *Mechanical Systems and Signal Processing*, Vol. 115, pp. 213–237, Jan. 2019.

39. Saxena, A., K. Goebel, D. Simon and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation", *2008 International Conference on Prognostics and Health Management*, pp. 1–9, Denver, CO, USA, Oct. 2008.

40. Gebraeel, N., A. Elwany and J. Pan, "Residual Life Predictions in the Absence of Prior Degradation Knowledge", *IEEE Transactions on Reliability*, Vol. 58, No. 1, pp. 106–117, Mar. 2009.

41. Nadakatti, M., "A knowledge based approach [maintenance engineering]", *Manufacturing Engineer*, Vol. 85, No. 2, pp. 24–27, Apr. 2006.

42. Peng, Y., M. Dong and M. J. Zuo, "Current status of machine prognostics in condition-based maintenance: a review", *The International Journal of Advanced Manufacturing Technology*, Vol. 50, No. 1-4, pp. 297–313, Sep. 2010.

43. Roberts, S. J., "Novelty detection using extreme value statistics", *IEE Proceedings - Vision, Image, and Signal Processing*, Vol. 146, No. 3, pp. 124–129, 1999.

44. Markou, M. and S. Singh, "Novelty detection: a review—part 1: statistical approaches", *Signal Processing*, Vol. 83, No. 12, pp. 2481–2497, Dec. 2003.

45. Clifton, D. A., L. A. Clifton, P. R. Bannister and L. Tarassenko, "Automated Novelty Detection in Industrial Systems", Y. Liu, A. Sun, H. T. Loh, W. F. Lu and E.-P. Lim (Editors), *Advances of Computational Intelligence in Industrial Systems*, Studies in Computational Intelligence, pp. 269–296, Springer, Berlin, Heidelberg, 2008.

46. Yeung, D.-Y. and C. Chow, "Parzen-window network intrusion detectors", *Object recognition supported by user interaction for service robots*, Vol. 4, pp. 385–388, Quebec City, Quebec, Canada, Aug. 2002.

47. Heyns, T., P. S. Heyns and J. P. de Villiers, "Combining synchronous averaging with a Gaussian mixture model novelty detection scheme for vibration-based condition monitoring of a gearbox", *Mechanical Systems and Signal Processing*, Vol. 32, pp. 200–215, Oct. 2012.

48. Gebraeel, N. Z. and M. A. Lawley, "A Neural Network Degradation Model for Computing and Updating Residual Life Distributions", *IEEE Transactions on Automation Science and Engineering*, Vol. 5, No. 1, pp. 154–163, Jan. 2008.

49. Baccarini, L. M. R., V. V. Rocha e Silva, B. R. de Menezes and W. M. Caminhas, "SVM practical industrial application for mechanical faults diagnostic", *Expert Systems with Applications*, Vol. 38, No. 6, pp. 6980–6984, Jun. 2011.

50. Widodo, A. and B.-S. Yang, "Machine health prognostics using survival probability and support vector machine", *Expert Systems with Applications*, Vol. 38, No. 7, pp. 8430–8437, Jul. 2011.

51. Fang, X., K. Paynabar and N. Gebraeel, "Image-Based Prognostics Using Penalized Tensor Regression", *Technometrics*, Vol. 61, No. 3, pp. 369–384, Mar. 2019.

52. Liu, D., J. Pang, J. Zhou, Y. Peng and M. Pecht, "Prognostics for state of health estimation of lithium-ion batteries based on combination Gaussian process functional regression", *Microelectronics Reliability*, Vol. 53, No. 6, pp. 832–839, Jun. 2013.

53. Medjaher, K., J.-Y. Moya and N. Zerhouni, "Failure prognostic by using dynamic Bayesian Networks.", M. P. Fanti and M. Dotoli (Editors), *2nd IFAC Workshop on Dependable Control of Discrete Systems, DCDS'09.*, Vol. 1, pp. 291–296, IFAC, Bari, Italy, Jun. 2009.

54. Dong, M. and Z.-b. Yang, "Dynamic Bayesian network based prognosis in machining processes", *Journal of Shanghai Jiaotong University (Science)*, Vol. 13, No. 3, pp. 318–322, Jun. 2008.

55. Ocak, H. and K. A. Loparo, "HMM-Based Fault Detection and Diagnosis Scheme for Rolling Element Bearings", *Journal of Vibration and Acoustics*, Vol. 127, No. 4, pp. 299–306, Aug. 2005.

56. Sloukia, F., M. E. Aroussi, H. Medromi and M. Wahbi, "Bearings prognostic using Mixture of Gaussians Hidden Markov Model and Support Vector Machine", *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–4, Ifrane, Morocco, May 2013.

57. Dong, S. and T. Luo, "Bearing degradation process prediction based on the PCA and optimized LS-SVM model", *Measurement*, Vol. 46, No. 9, pp. 3143–3152, Nov. 2013.

58. Malhi, A. and R. Gao, "PCA-based feature selection scheme for machine defect

classification", *IEEE Transactions on Instrumentation and Measurement*, Vol. 53, No. 6, pp. 1517–1525, Dec. 2004.

59. Ondel, O., E. Boutleux, E. Blanco and G. Clerc, "Coupling Pattern Recognition With State Estimation Using Kalman Filter for Fault Diagnosis", *IEEE Transactions on Industrial Electronics*, Vol. 59, No. 11, pp. 4293–4300, Nov. 2012.

60. Lim, C. K. R. and D. Mba, "Switching Kalman filter for failure prognostic", *Mechanical Systems and Signal Processing*, Vol. 52-53, pp. 426–435, Feb. 2015.

61. Xian, W., B. Long, M. Li and H. Wang, "Prognostics of Lithium-Ion Batteries Based on the Verhulst Model, Particle Swarm Optimization and Particle Filter", *IEEE Transactions on Instrumentation and Measurement*, Vol. 63, No. 1, pp. 2–17, Jan. 2014.

62. de Freitas, N., "Rao-Blackwellised particle filtering for fault diagnosis", *Proceedings, IEEE Aerospace Conference*, Vol. 4, pp. 4/1767–4/1772, Big Sky, MT, USA, Mar. 2002.

63. Lei, Y., N. Li, L. Guo, N. Li, T. Yan and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction", *Mechanical Systems and Signal Processing*, Vol. 104, pp. 799–834, May 2018.

64. Patil, S., A. Patil, V. Handikherkar, S. Desai, V. M. Phalle and F. S. Kazi, "Remaining Useful Life (RUL) Prediction of Rolling Element Bearing Using Random Forest and Gradient Boosting Technique", *ASME 2018 International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers Digital Collection, Pittsburgh, PA, USA, Nov. 2018.

65. Benkedjouh, T., K. Medjaher, N. Zerhouni and S. Rechak, "Remaining useful life estimation based on nonlinear feature reduction and support vector regression", *Engineering Applications of Artificial Intelligence*, Vol. 26, No. 7, pp. 1751–1760,

Aug. 2013.

66. Wu, Q., K. Ding and B. Huang, "Approach for fault prognosis using recurrent neural network", *Journal of Intelligent Manufacturing*, Jun. 2018.

67. Sateesh Babu, G., P. Zhao and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life", S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang and H. Xiong (Editors), *Database Systems for Advanced Applications*, Lecture Notes in Computer Science, pp. 214–228, Springer International Publishing, 2016.

68. Heimes, F. O., "Recurrent neural networks for remaining useful life estimation", *2008 International Conference on Prognostics and Health Management*, pp. 1–6, Denver, CO, USA, Oct. 2008.

69. Ramasso, E., "Investigating computational geometry for failure prognostics", *International Journal of Prognostics and Health Management*, Vol. 5, No. 1, pp. 1–18, Jul. 2014.

70. Zheng, S., K. Ristovski, A. Farahat and C. Gupta, "Long Short-Term Memory Network for Remaining Useful Life estimation", *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95, Jun. 2017.

71. Li, X., Q. Ding and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks", *Reliability Engineering & System Safety*, Vol. 172, pp. 1–11, Apr. 2018.

72. Al-Dulaimi, A., S. Zabihi, A. Asif and A. Mohammadi, "A multimodal and hybrid deep neural network model for Remaining Useful Life estimation", *Computers in Industry*, Vol. 108, pp. 186–196, Jun. 2019.

73. Nouri, M., B. K. Fussell, B. L. Ziniti and E. Linder, "Real-time tool wear monitoring in milling using a cutting condition independent method", *International*

Journal of Machine Tools and Manufacture, Vol. 89, pp. 1–13, Feb. 2015.

74. Yuan, M., Y. Wu and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network", *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, pp. 135–140, Beijing, China, Oct. 2016.

75. Mahamad, A. K., S. Saon and T. Hiyama, "Predicting remaining useful life of rotating machinery based artificial neural network", *Computers & Mathematics with Applications*, Vol. 60, No. 4, pp. 1078–1087, Aug. 2010.

76. Zhang, C., P. Lim, A. K. Qin and K. C. Tan, "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 10, pp. 2306–2318, Oct. 2017.

77. Liao, Y., L. Zhang and C. Liu, "Uncertainty Prediction of Remaining Useful Life Using Long Short-Term Memory Network Based on Bootstrap Method", *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–8, Seattle, Washington, USA, Jun. 2018.

78. Liu, H., Z. Liu, W. Jia and X. Lin, "Remaining Useful Life Prediction Using A Novel Feature-attention Based End-to-end Approach", *IEEE Transactions on Industrial Informatics*, pp. 1–1, Mar. 2020.

79. Huang, C.-G., H.-Z. Huang and Y.-F. Li, "A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions", *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 11, pp. 8792–8802, Nov. 2019.

80. Al-Dahidi, S., F. Di Maio, P. Baraldi and E. Zio, "Remaining useful life estimation in heterogeneous fleets working under variable operating conditions", *Reliability Engineering & System Safety*, Vol. 156, pp. 109–124, Dec. 2016.

81. Li, N., Y. Lei, T. Yan, N. Li and T. Han, "A Wiener-Process-Model-Based Method

for Remaining Useful Life Prediction Considering Unit-to-Unit Variability", *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 3, pp. 2092–2101, Mar. 2019.

82. Khan, S. and T. Yairi, "A review on the application of deep learning in system health management", *Mechanical Systems and Signal Processing*, Vol. 107, pp. 241–265, Jul. 2018.

83. Neogi, N., D. K. Mohanta and P. K. Dutta, "Review of vision-based steel surface inspection systems", *EURASIP Journal on Image and Video Processing*, Vol. 2014, No. 1, pp. 1–19, Nov. 2014.

84. Mcdonald, S. A., M. Preuss, E. Maire, J.-Y. Buffiere, P. M. Mummery and P. J. Withers, "X-ray tomographic imaging of Ti/SiC composites", *Journal of Microscopy*, Vol. 209, No. 2, pp. 102–112, 2003.

85. Hiasa, S., R. Birgul and F. N. Catbas, "Infrared thermography for civil structural assessment: demonstrations with laboratory and field studies", *Journal of Civil Structural Health Monitoring*, Vol. 6, No. 3, pp. 619–636, Jul. 2016.

86. Berthel, B., B. Wattrisse, A. Chrysochoos and A. Galtier, "Thermographic Analysis of Fatigue Dissipation Properties of Steel Sheets", *Strain*, Vol. 43, No. 3, pp. 273–279, 2007.

87. Dutta, S., S. K. Pal, S. Mukhopadhyay and R. Sen, "Application of digital image processing in tool condition monitoring: A review", *CIRP Journal of Manufacturing Science and Technology*, Vol. 6, No. 3, pp. 212–232, Jan. 2013.

88. Seo, J. J., H. Yoon, H. Ha, D. P. Hong and W. Kim, "Infrared Thermographic Diagnosis Mechnism for Fault Detection of Ball Bearing under Dynamic Loading Conditions", *Advanced Materials Research*, Vol. 295–297, pp. 1544–1547, 2011.

89. Yan, H., K. Paynabar and J. Shi, "Real-Time Monitoring of High-Dimensional Functional Data Streams via Spatio-Temporal Smooth Sparse Decomposition",

Technometrics, Vol. 60, No. 2, pp. 181–197, Apr. 2018.

90. Liu, X., K. Yeo and J. Kalagnanam, "Statistical Modeling for Spatio-Temporal Degradation Data", *Journal of Quality Technology*, Vol. 50, No. 2, pp. 166–182, Sep. 2018.

91. Zhou, F., Y. Gao and C. Wen, "A Novel Multimode Fault Classification Method Based on Deep Learning", *Journal of Control Science and Engineering*, Vol. 2017, pp. 1–14, Oct. 2017.

92. Lu, C., Z.-Y. Wang, W.-L. Qin and J. Ma, "Fault diagnosis of rotary machinery components using a stacked denoising autoencoder-based health state identification", *Signal Processing*, Vol. 130, pp. 377–388, Jan. 2017.

93. Ince, T., S. Kiranyaz, L. Eren, M. Askar and M. Gabbouj, "Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks", *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 11, pp. 7067–7075, Nov. 2016.

94. Wang, P., Ananya, R. Yan and R. X. Gao, "Virtualization and deep recognition for system fault classification", *Journal of Manufacturing Systems*, Vol. 44, pp. 310–316, Jul. 2017.

95. Liao, L., W. Jin and R. Pavel, "Enhanced Restricted Boltzmann Machine With Prognosability Regularization for Prognostics and Health Assessment", *IEEE Transactions on Industrial Electronics*, Vol. 63, No. 11, pp. 7076–7083, Nov. 2016.

96. Zhao, R., R. Yan, J. Wang and K. Mao, "Learning to Monitor Machine Health with Convolutional Bi-Directional LSTM Networks", *Sensors*, Vol. 17, No. 2, pp. 273–290, Feb. 2017.

97. Malhotra, P., V. Tv, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal and G. Shroff, "Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder", *1st ACM SIGKDD Workshop on Machine Learning*

for Prognostics and Health Management, pp. 1–10, San Francisco, CA, USA, 08 2016.

98. Aydemir, G. and B. Acar, "Anomaly Triggered Remaining Useful Life Estimation", Submitted to Journal of Manufacturing Systems.

99. Randall, R. B., *Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications*, Wiley, Chichester, West Sussex, U.K., 1st edn., Feb. 2011.

100. Page, E. S., "Cumulative Sum Charts", *Technometrics*, Vol. 3, No. 1, pp. 1–9, Feb. 1961.

101. Gebraeel, N., "Sensory-Updated Residual Life Distributions for Components With Exponential Degradation Patterns", *IEEE Transactions on Automation Science and Engineering*, Vol. 3, No. 4, pp. 382–393, Oct. 2006.

102. Li, N., Y. Lei, J. Lin and S. X. Ding, "An Improved Exponential Model for Predicting Remaining Useful Life of Rolling Element Bearings", *IEEE Transactions on Industrial Electronics*, Vol. 62, No. 12, pp. 7762–7773, Dec. 2015.

103. Jr, J. J. P. and G. C. Runger, "Comparisons of Multivariate CUSUM Charts", *Journal of Quality Technology*, Vol. 22, No. 3, pp. 173–186, Jul. 1990.

104. Ryan, A. G., L. J. Wells and W. H. Woodall, "Methods for Monitoring Multiple Proportions When Inspecting Continuously", *Journal of Quality Technology*, Vol. 43, No. 3, Jul. 2011.

105. Montgomery, D. C., *Statistical quality control: a modern introduction*, Wiley, Hoboken, NJ, 7th edn., 2013.

106. Breiman, L. (Editor), *Classification and regression trees*, Chapman & Hall [u.a.], Boca Raton, repr edn., 1998.

107. Buitinck, L., G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project", *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, Prague, Czech Republic, 2013.

108. Bromley, J., I. Guyon, Y. LeCun, E. Sackinger and R. Shah, "Signature verification using a "Siamese" time delay neural network", *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pp. 737–744, Morgan Kaufmann Publishers Inc., Denver, Colorado, Nov. 1993.

109. Chopra, S., R. Hadsell and Y. LeCun, "Learning a Similarity Metric Discriminatively, with Application to Face Verification", *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1, pp. 539–546, IEEE, San Diego, CA, USA, 2005.

110. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, Nov. 2016, `http://www.deeplearningbook.org`.

111. Duchi, J., E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", *Journal of Machine Learning Research*, Vol. 12, No. Jul, pp. 2121–2159, 2011.

112. Chollet, F. *et al.*, "Keras", `https://keras.io`, 2015.

113. Kingma, D. and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations*, pp. 1–1, 12 2014.

114. Aydemir, G. and K. Paynabar, "Image-based Prognostics Using Deep Learning Approach", *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019, `https://ieeexplore.ieee.org/document/8915753/`, © 2019 IEEE.

115. Krizhevsky, A., I. Sutskever and G. E. Hinton, "ImageNet Classification with

Deep Convolutional Neural Networks", *NIPS*, pp. 1097–1105, Lake Tahoe, NV, USA, Dec. 2012.

116. Maas, A. L., A. Y. Hannun and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models", *Proceedings of the 30th International Conference on Machine Learning*, Vol. 30, pp. 1097–1105, Atlanta, GA, USA, Dec. 2013.

117. Hecht-nielsen, R., "Theory of the Backpropagation Neural Network", H. Wechsler (Editor), *Neural Networks for Perception*, pp. 65–93, Academic Press, Jan. 1992.

118. Cun, Y. L., L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard and W. Hubbard, "Handwritten digit recognition: applications of neural network chips and automatic learning", *IEEE Communications Magazine*, Vol. 27, No. 11, pp. 41–46, Nov. 1989.

119. Hochreiter, S. and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, Nov. 1997.

120. Werbos, P., "Backpropagation through time: what it does and how to do it", *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1550–1560, Oct. 1990.

121. Gal, Y. and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks", D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon and R. Garnett (Editors), *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 1027–1035, Barcelona, SPAIN, Dec. 2016.

122. Abadi, M., A. Agarwal, P. Barham and et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", `http://tensorflow.org`, 2015.

APPENDIX A: EXAMPLES OF THE RAW SENSOR MEASUREMENTS AND OPERATING CONDITION INVARIANT FEATURES

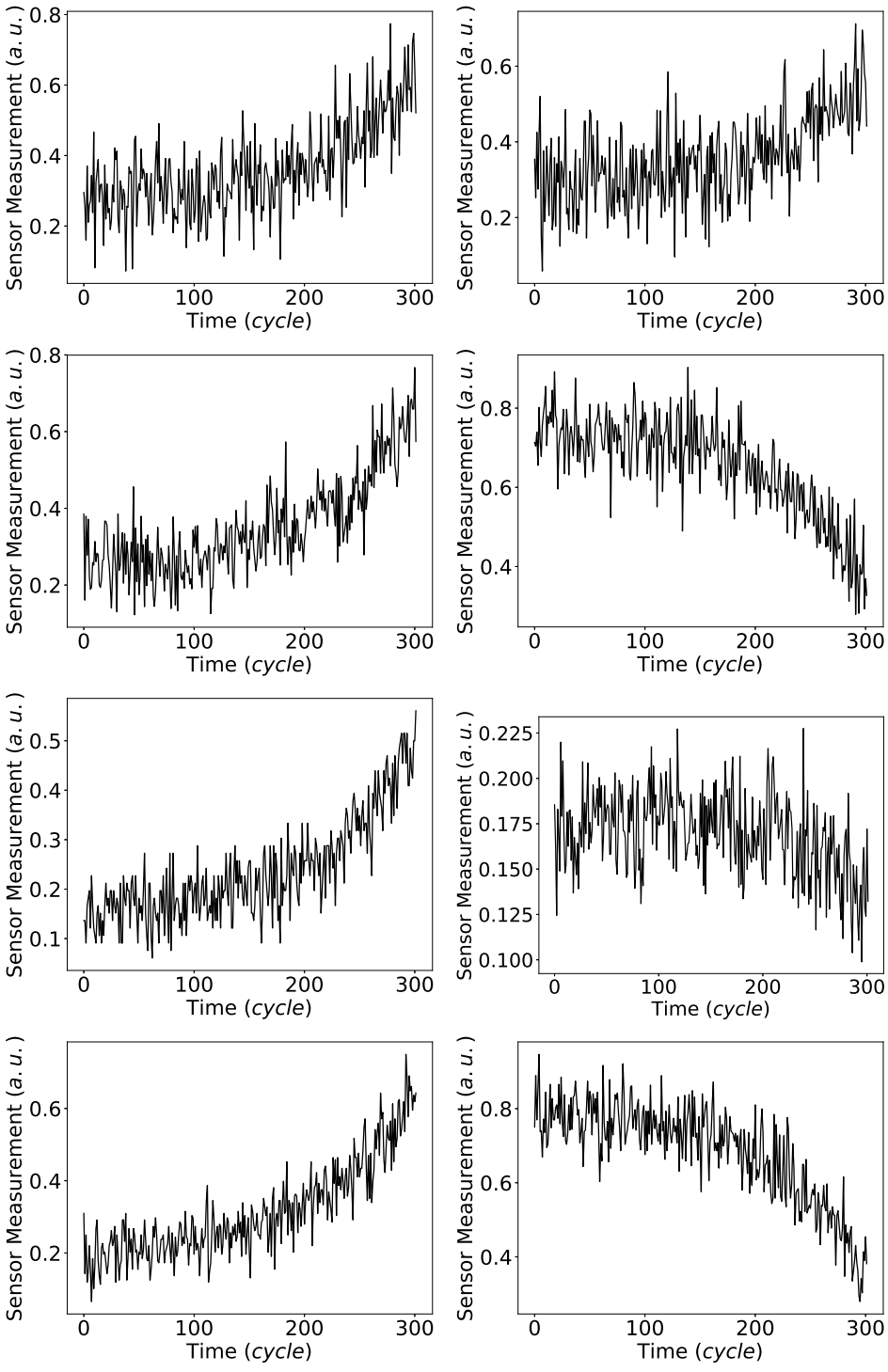


Figure A.1. The raw sensor measurements for a sample engine with single operating condition (Engine 48 in the test set of FD001).

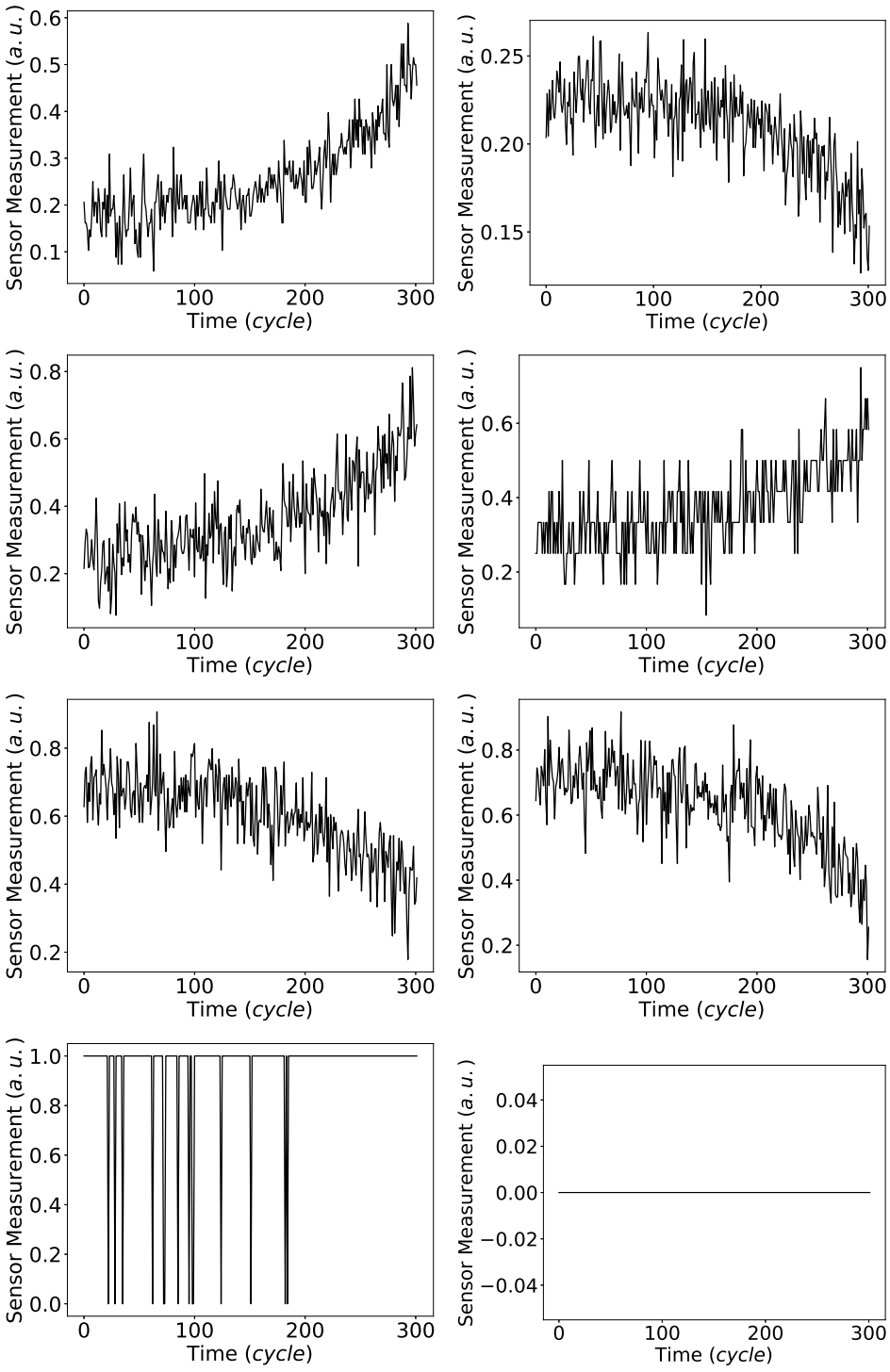


Figure A.2. The raw sensor measurements for a sample engine with single operating condition (Engine 48 in the test set of FD001).

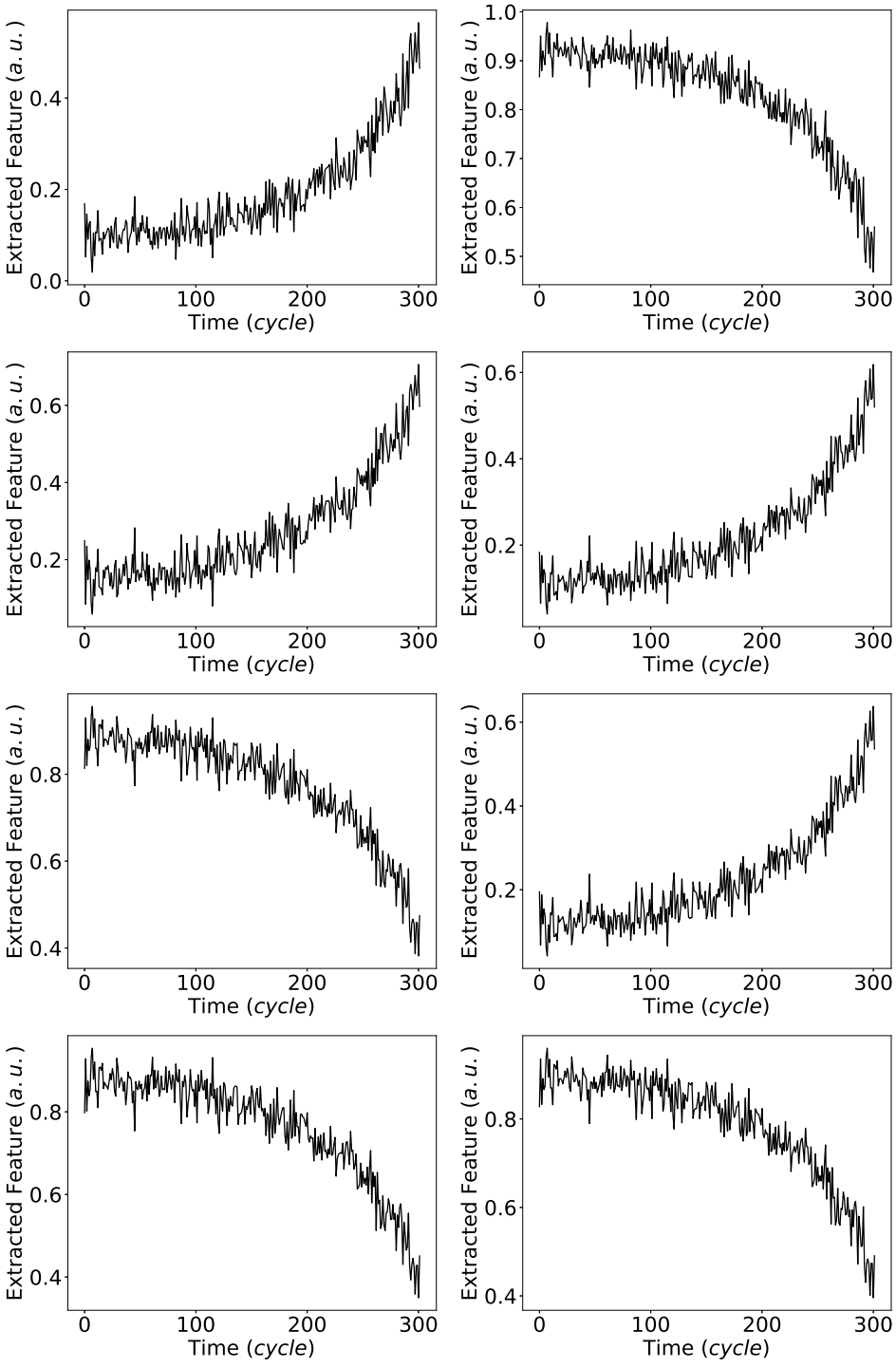


Figure A.3. The operating condition invariant features for a sample engine with single operating condition (Engine 48 in the test set of FD001).

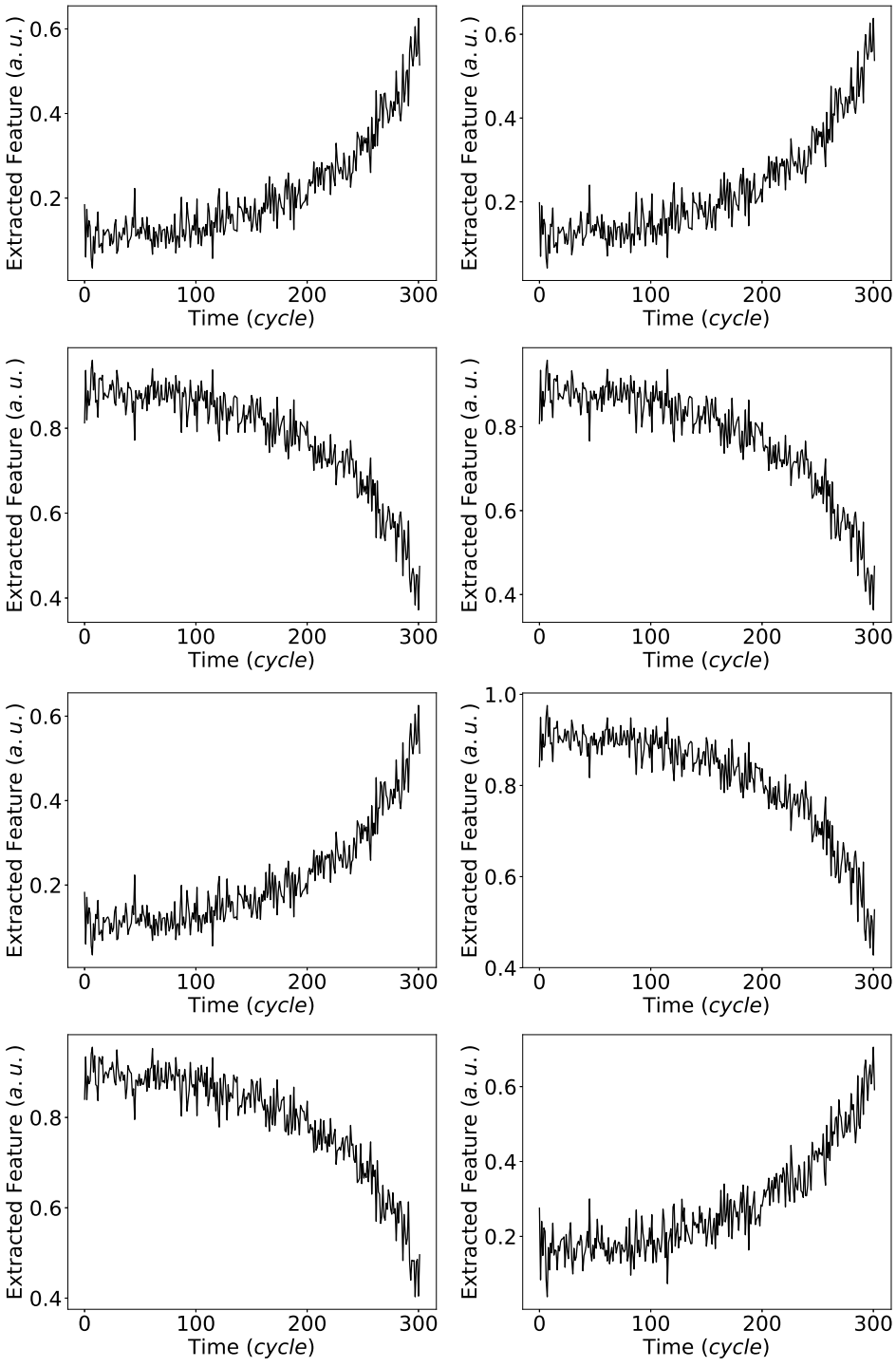


Figure A.4. The operating condition invariant features for a sample engine with single operating condition (Engine 48 in the test set of FD001).

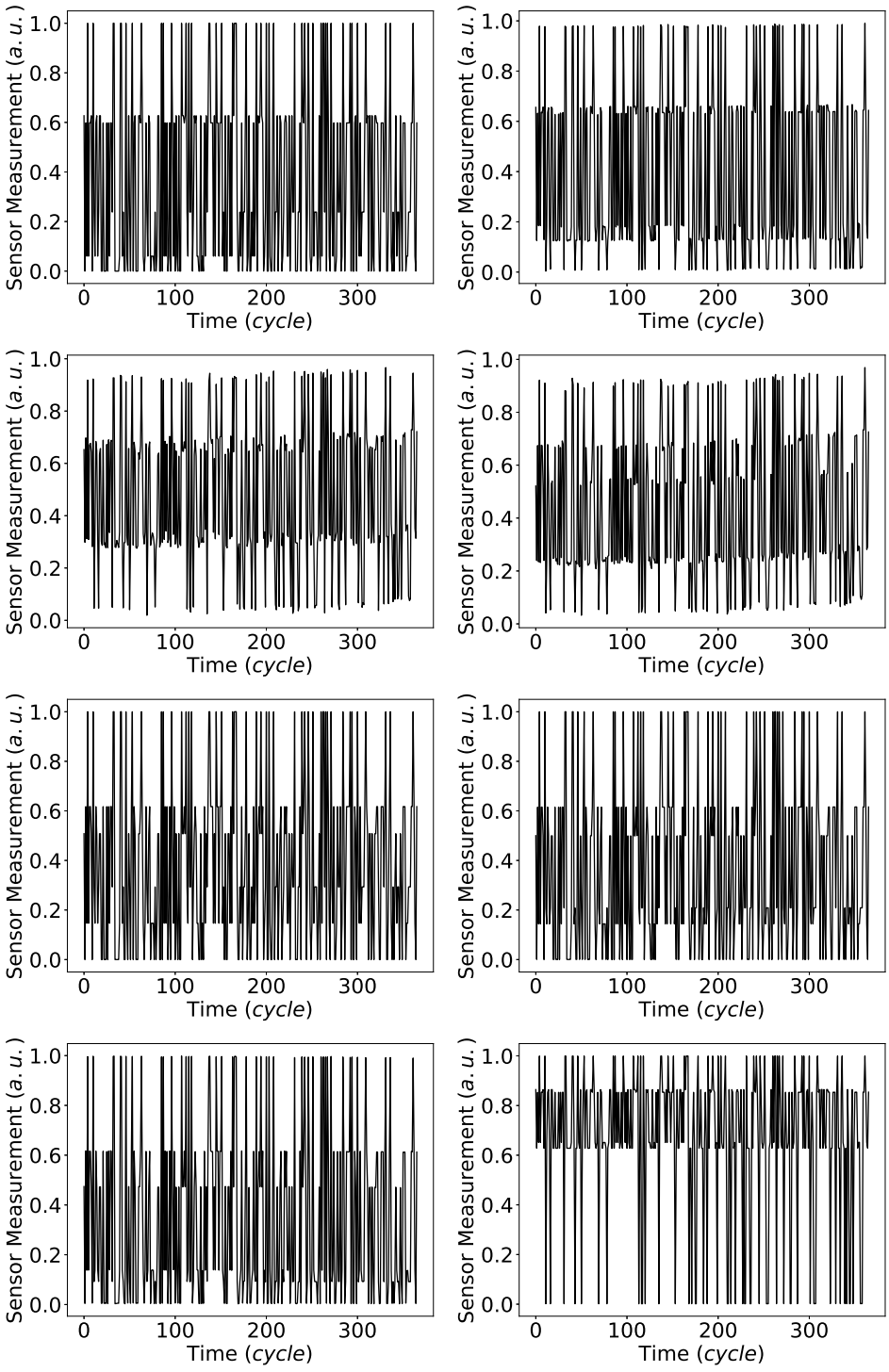


Figure A.5. The raw sensor measurements for a sample engine with multiple operating conditions (Engine 64 in the test set of FD002).

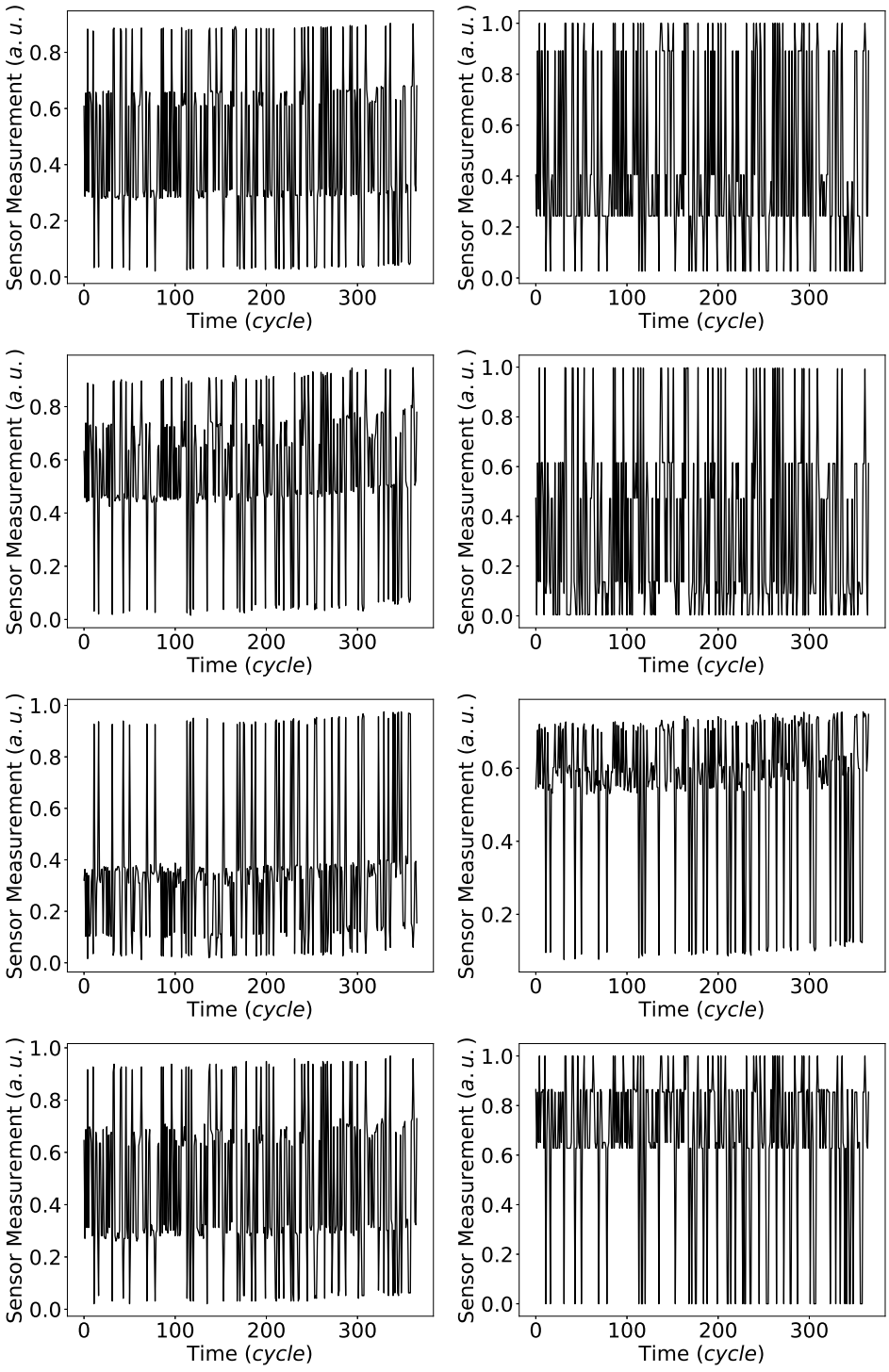


Figure A.6. The raw sensor measurements for a sample engine with multiple operating conditions (Engine 64 in the test set of FD002).

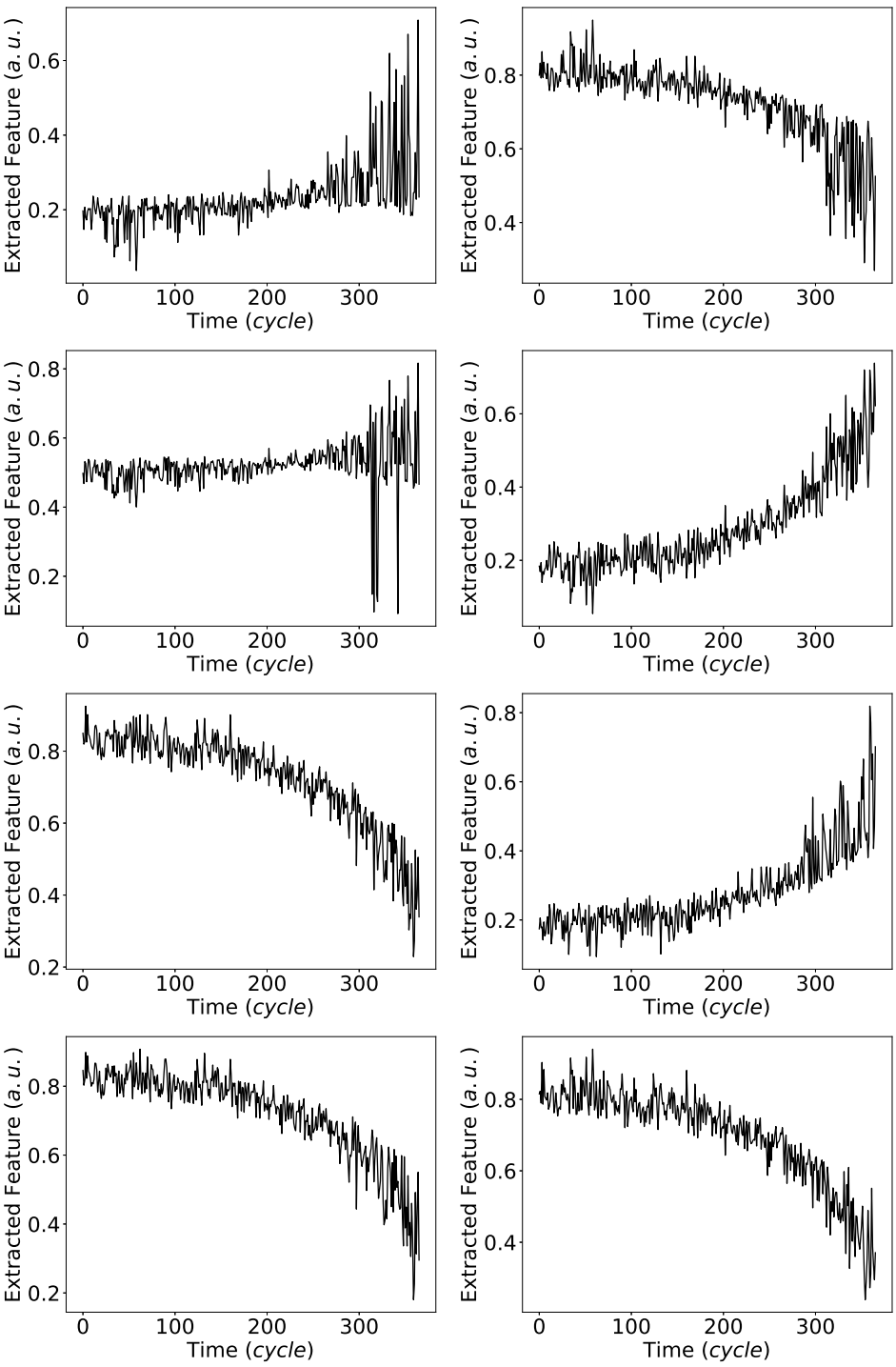


Figure A.7. The operating condition invariant features for a sample engine with multiple operating conditions (Engine 64 in the test set of FD002).

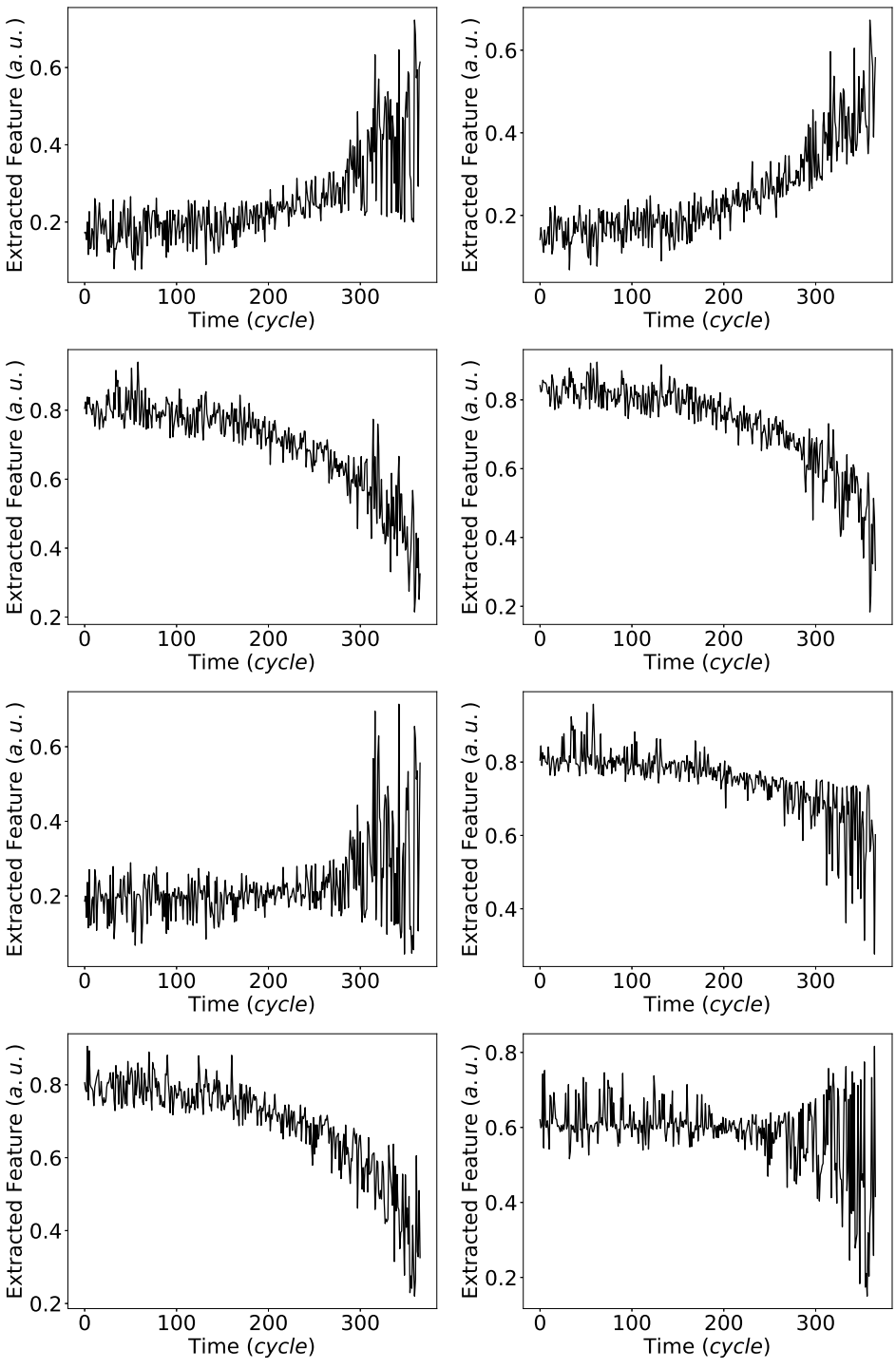


Figure A.8. The operating condition invariant features for a sample engine with multiple operating conditions (Engine 64 in the test set of FD002).

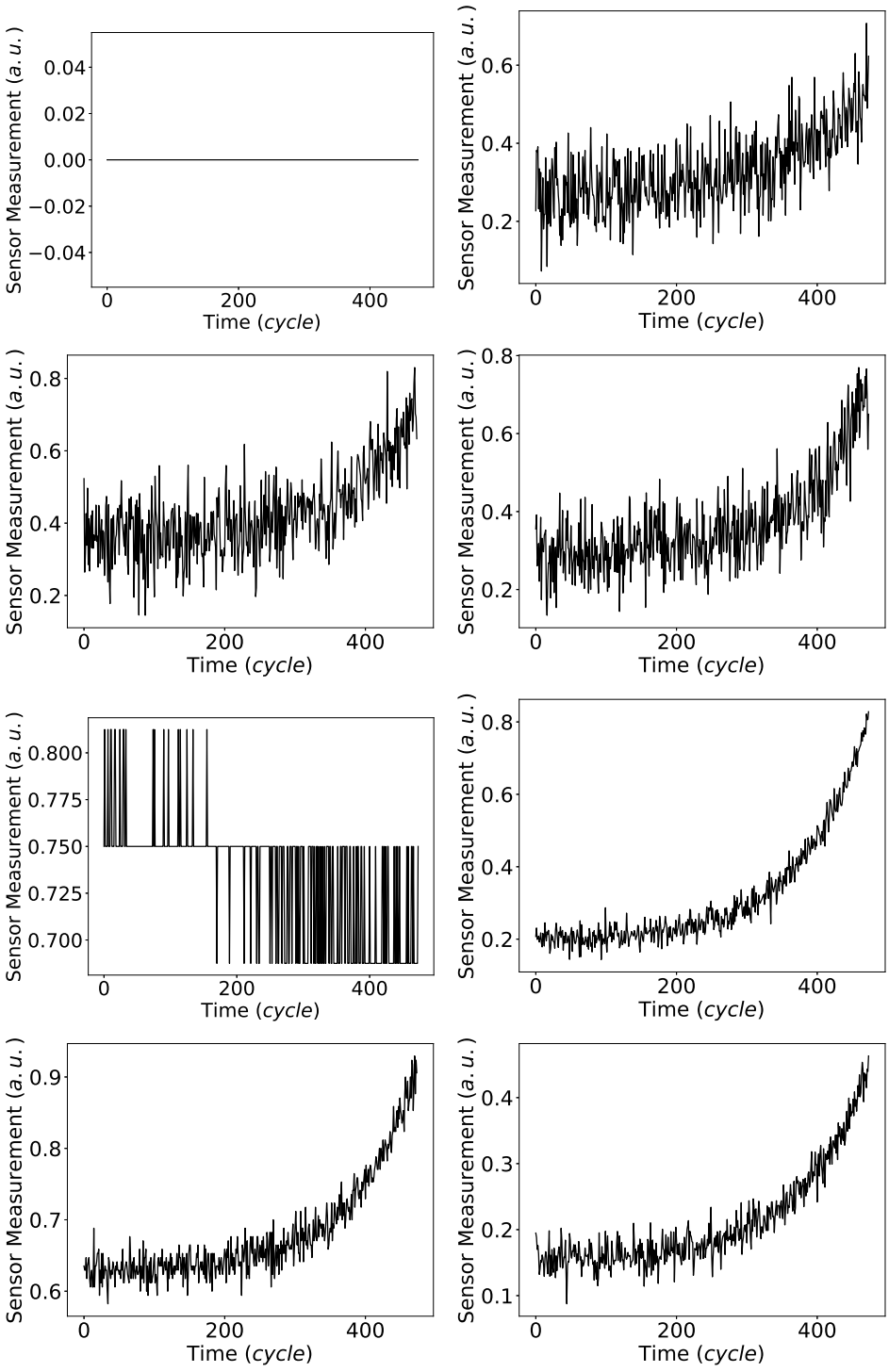


Figure A.9. The raw sensor measurements for a sample engine with single operating condition (Engine 23 in the test set of FD003).

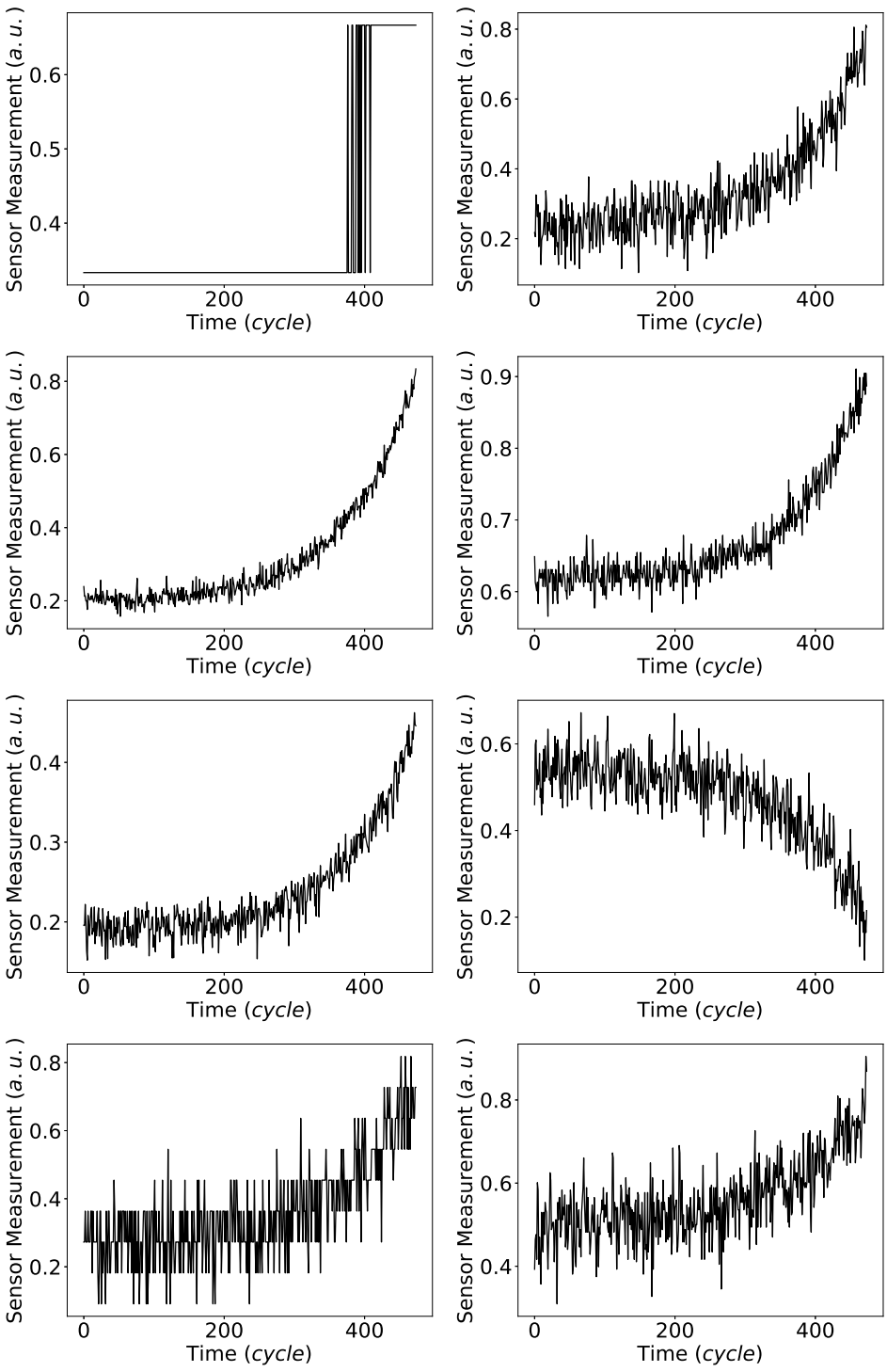


Figure A.10. The raw sensor measurements for a sample engine with single operating condition (Engine 23 in the test set of FD003).

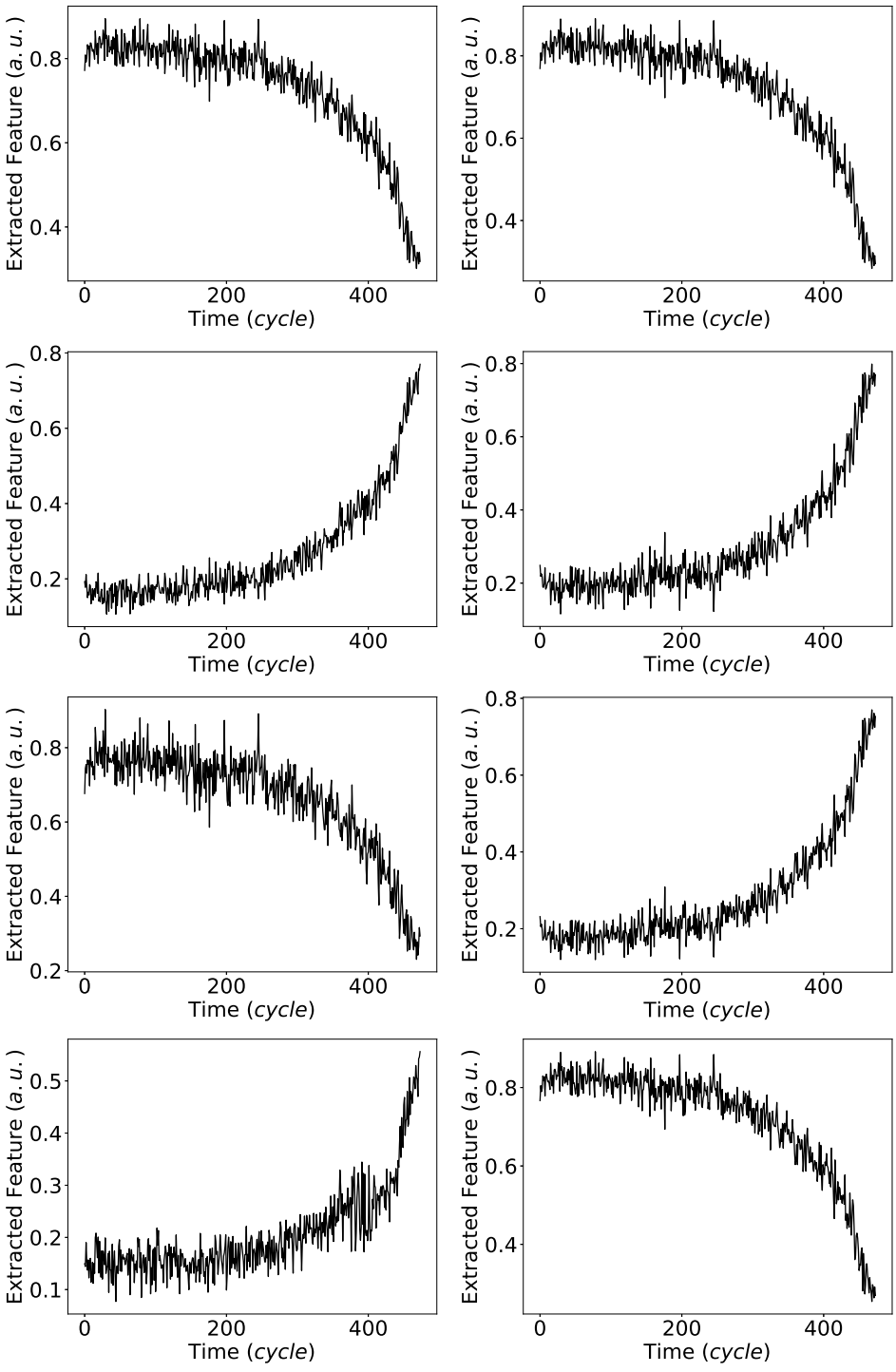


Figure A.11. The operating condition invariant features for a sample engine with single operating condition (Engine 23 in the test set of FD003).

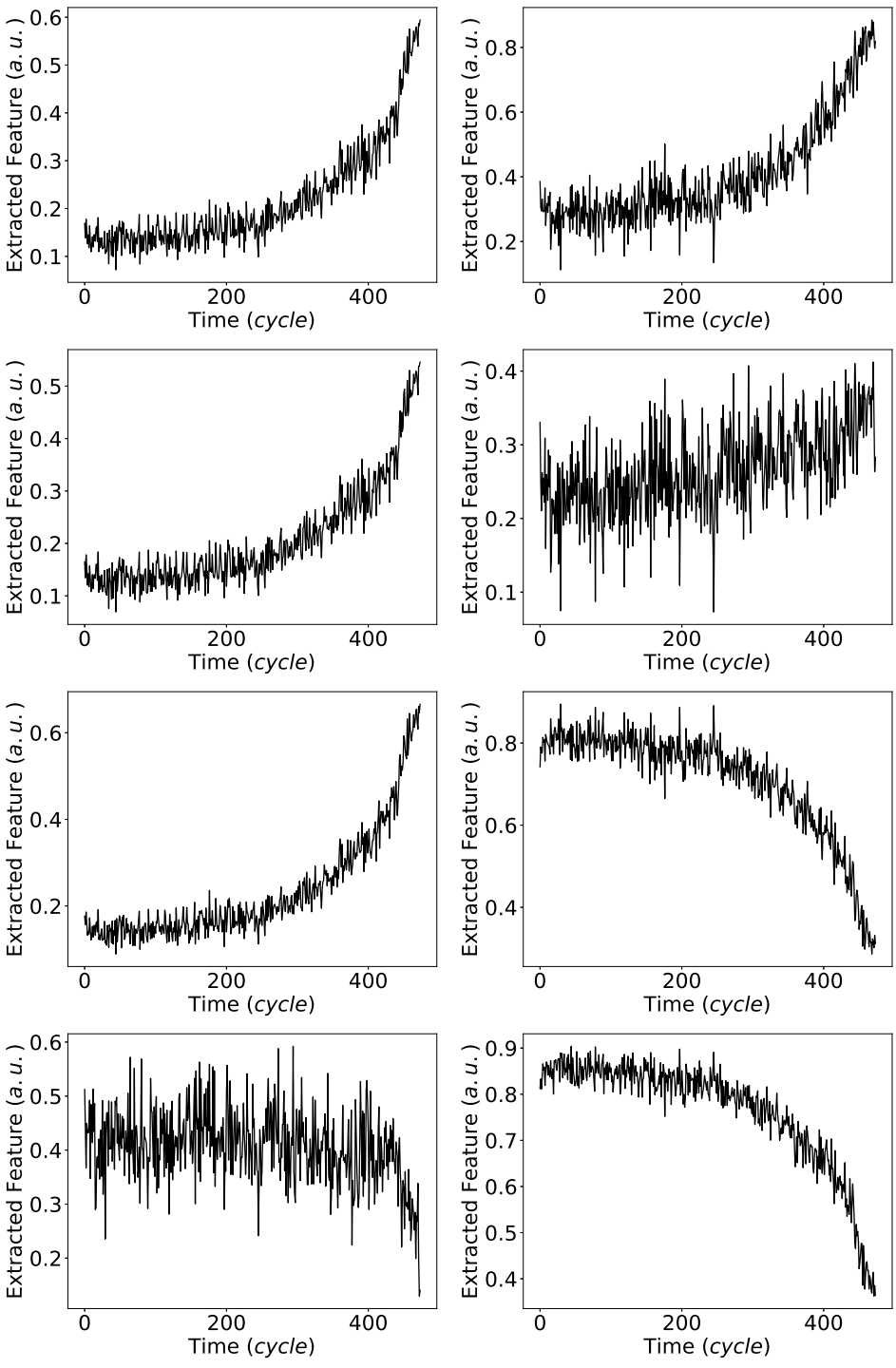


Figure A.12. The operating condition invariant features for a sample engine with single operating condition (Engine 23 in the test set of FD003).

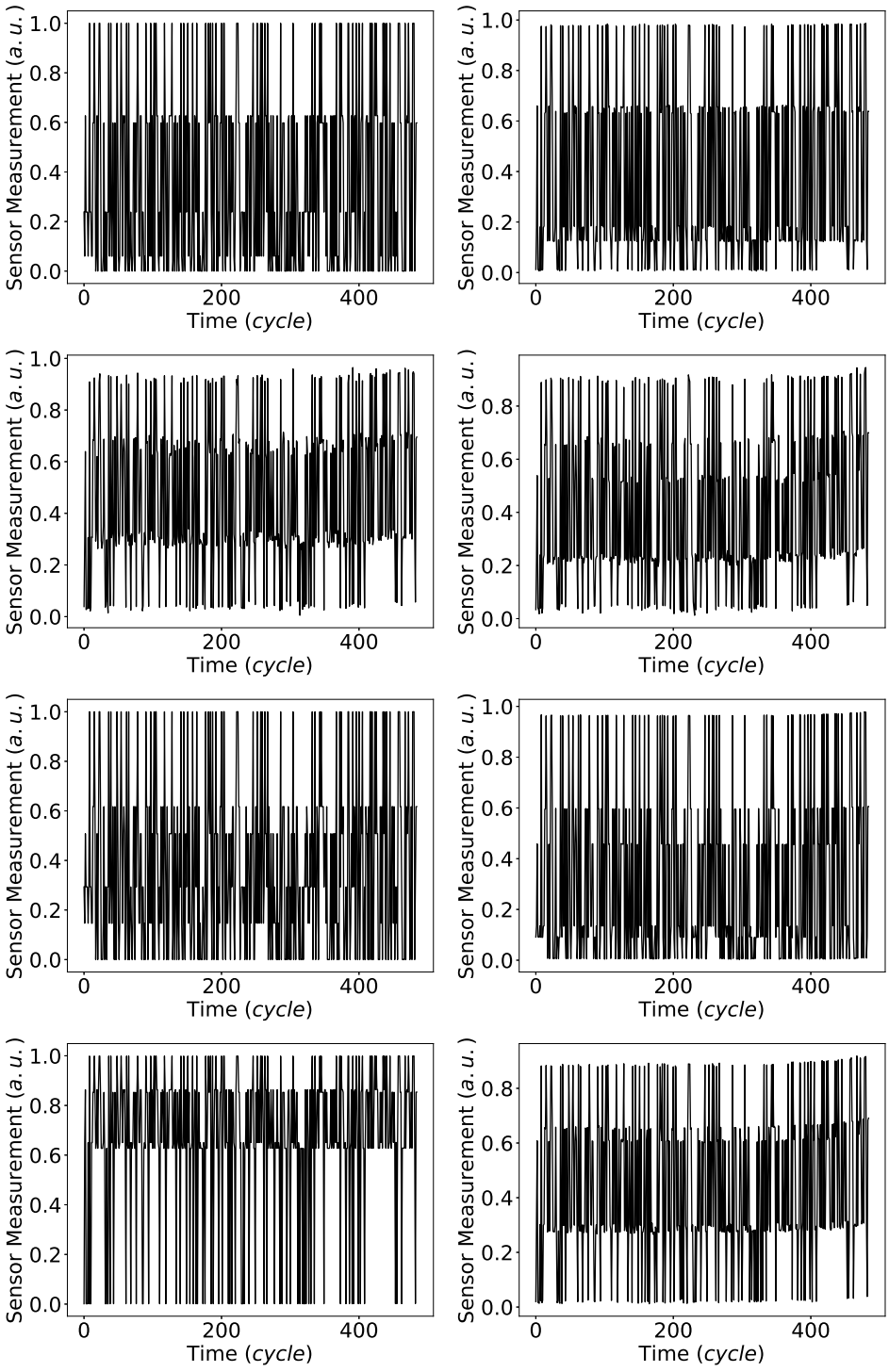


Figure A.13. The raw sensor measurements for a sample engine with multiple operating conditions (Engine 24 in the test set of FD004).

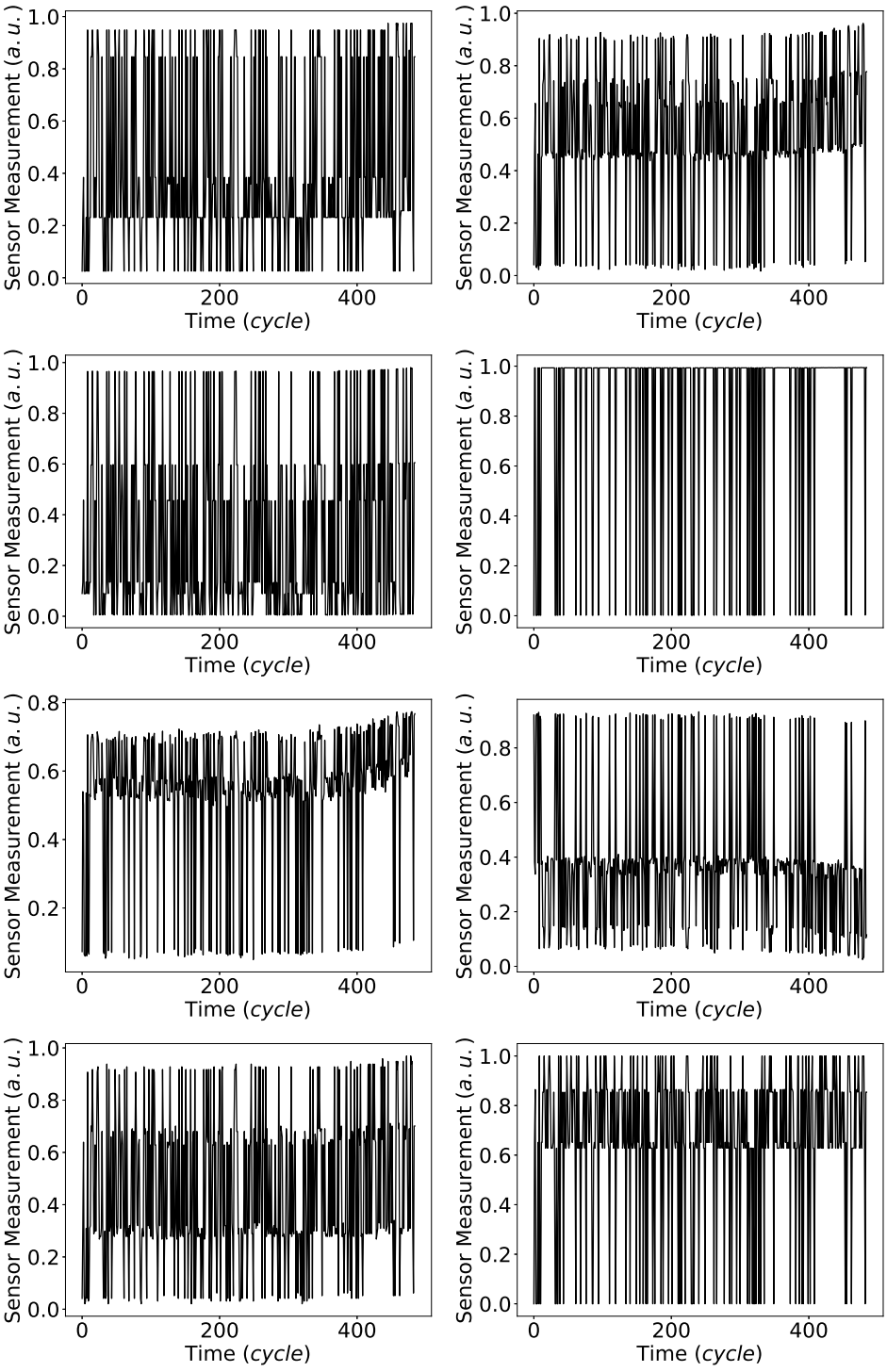


Figure A.14. The raw sensor measurements for a sample engine with multiple operating conditions (Engine 24 in the test set of FD004).

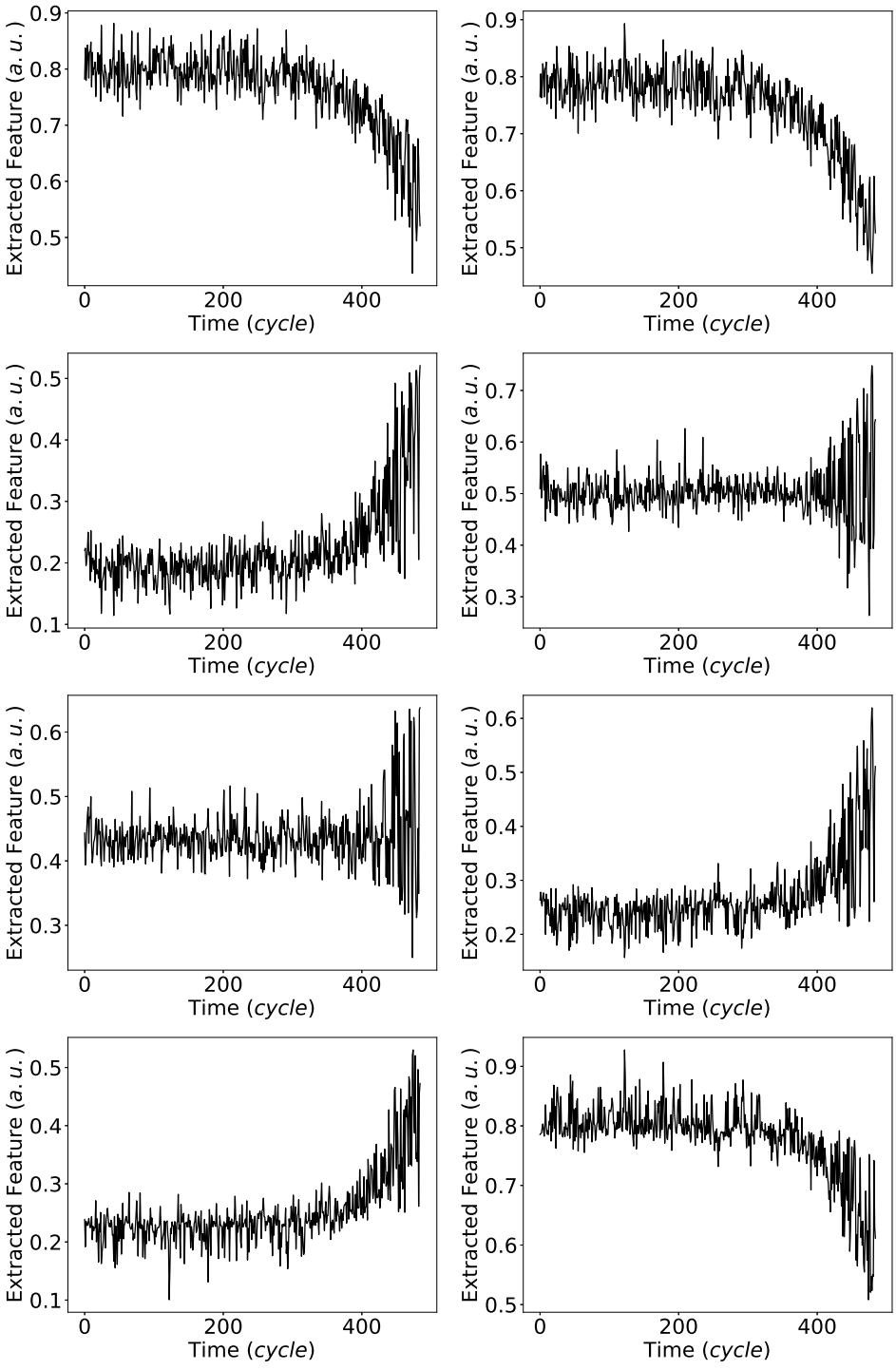


Figure A.15. The operating condition invariant features for a sample engine with multiple operating conditions (Engine 24 in the test set of FD004).

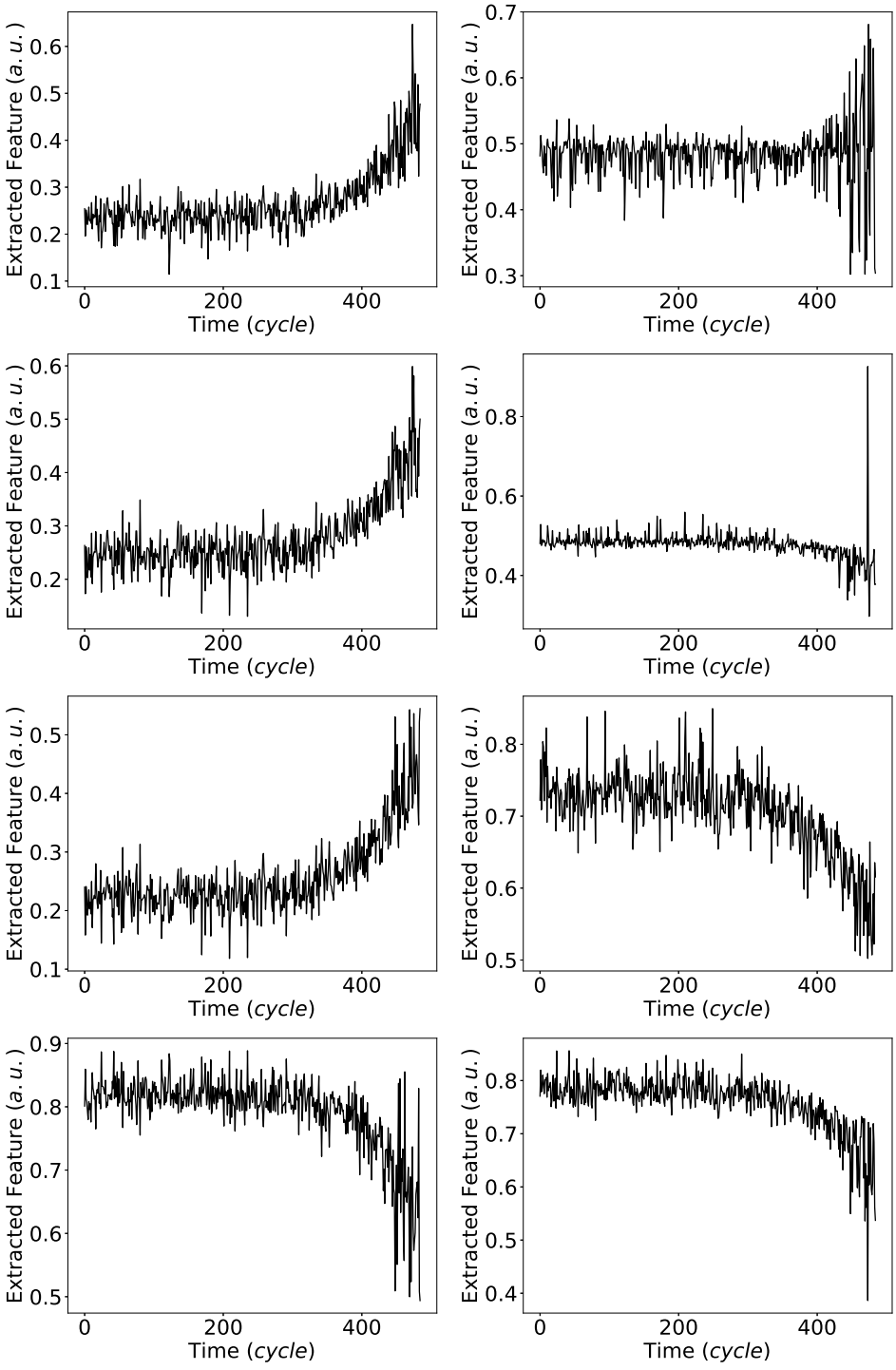


Figure A.16. The operating condition invariant features for a sample engine with multiple operating conditions (Engine 24 in the test set of FD004).