ABSTRACT

Title of Dissertation: QUANTILE-BASED LSTM REMAINING USEFUL LIFE PREDICTOR

Yonatan Saadon, Doctor of Philosophy, 2021

Dissertation directed by: Professor Patrick McCluskey, Department of Mechanical Engineering

Accurate prediction of the remaining useful life (RUL) of a degrading component is crucial to prognostics and health management for electronic systems, to monitor conditions and avoid reaching failure while minimizing downtime. However, the shortage of sufficiently large run-to-failure datasets is a serious bottleneck impeding the performance of data-driven approaches, and in particular, those involving neural network architectures. Here, this work shows a new data-driven prognostic method to predict the RUL using an ensemble of quantile-based Long Short-Term Memory

(LSTM) neural networks, which represents the RUL prediction task to a set of simpler, binary classification problems that are amenable for prediction with LSTMs, even with limited data. This methodology was tested on two run-to-failure datasets, power MOSFETs and filtration system, and showed promising results on both datasets it demonstrates that this approach obtains improved RUL estimation accuracy for both the power MOSFETs and the filtration system, especially with a small training dataset that is characterized by a wide range of the RUL.

QUANTILE-BASED LSTM REMAINING USEFUL LIFE PREDICTOR

by

Yonatan Saadon

Dissertation submitted to the Faculty of the Graduate School of the

University of Maryland, College Park, in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

2021

Advisory Committee:

Professor Patrick McCluskey, Chair

Professor Hugh Bruck

Professor Mark Fuge

Professor Peter Sandborn

Professor Mohamad Al-Sheikhly

Dedication

I would like to dedicate this work to my wife Noam, for her guidance and help for the past five years, to my dog and best friend Gucci, our cat Roger, and to my loving family, my father Yair, my stepmom Irit and to my amazing brother and sisters, Inbal, Liel, and Ohad.

Acknowledgments

I would like to thank my advisor Professor Patrick McCluskey for his support and mentorship throughout my graduate studies. I would also like to thank my committee members:
Professor Hugh Bruck, for welcoming me into UMD as the director of graduate studies, and for his mentoring and support at my first year, Professor Mark Fuge, for inspiring me to investigate machine learning, after taking his class in my first semester, Professor Peter Sandborn, for replying to my email long before I joined UMD and giving me a tour at CALCE, and Professor Mohamad Al-Sheikhly, for teaching me everything I know about polymer physics, and for being there every time I needed friendly advice. I would also like to thank my lab colleagues: Maxim Serebreni, Subramani Manoharan, Jennifa Li, Zhaoxi Yao, He Yun, Sriram Jayanthi, and Kunal Ahuja.

Table of Contents

List of Figures

List of Tables

Introduction

Prediction of the Remaining Useful Life (RUL) is crucial for mitigating system shutdown and failure. It can also decrease the costs of maintenance by indicating the live status of the system. Existing strategies to address the challenge of RUL prediction may be categorized into (1) experience-based approaches, which aim to infer a simple global function describing failure, (2) analytical model-based approaches, which aim to derive a set of mathematical equations to represent the state of a system, and (3) data-driven approaches, which employ machine learning and pattern recognition techniques to evaluate the RUL based on data measurements from the system. Although numerous studies have been conducted to develop such techniques, these are mostly limited, hindered by the lack of a simple global function to predict the RUL with an experience-based approach, and the lack of sufficient data to construct a reliable model or train a robust machine learning classifier via a model-based or data-driven approach, respectively. Furthermore, it is desirable that an approach designed for RUL prediction would be capable of analyzing 'online' measurements and adjust the RUL by the observed changes in the system, by considering temporal dynamics and analyzing the long-term dependencies within the data.

Currently, Recurrent Neural Networks (RNNs), and particularly, gated RNNs such as Long Short Term Memory machines (LSTMs), demonstrate the state of the art performance in time series forecasting by holding a long-term memory of a time series data and identifying long term patterns that contribute to the classification task. The main issue impeding the utilization of LSTMs to construct a robust predictor of the RUL is the requirement for a large dataset to train the network, which would allow it to learn the complex dependencies

between the measurements and the full range of the RUL at each given time point, whereas sufficiently large training datasets are currently unavailable.

In this work, a framework was developed that overcame these limitations and facilitates LSTM-based RUL prediction by simplifying the classification task in a way that would eliminate the need for large training datasets and allow training on the smaller datasets that are currently available. The general classification problem of RUL prediction was converted, to a set of simpler classification tasks, of predicting whether the RUL is larger than a given threshold, for a set of considered thresholds. This demonstrated that each of these simplified tasks can be successfully addressed by training LSTMs, even with the current, relatively small, available datasets; a high accuracy for the LSTMs trained to predict whether the RUL is greater than a given threshold, at every time point. Building on these results, I developed an ensemble technique that incorporates different threshold-based classifiers, for which I observe such high prediction accuracy, and utilize these to accurately infer the precise RUL at every time point. This method can be applied to any given run to failure dataset when the failure mechanism and system/component are constant across the dataset.

In this work, I will start with an explanation of the different categories of prognostics methods and a literature review of existing strategies for RUL prediction. Next, I will describe the approach that I developed and introduce deep learning and particularly LSTMs, to provide the intuition and motivation for utilizing LSTMs in this work. This work contained two projects, first, a dataset with temporal measurements and RUL of power MOSFETs, second, constant measurements and RUL of filtration systems. Before each project, I will provide a brief introduction, for the first project I will provide an introduction

to MOSFETs and power MOSFETs, and the first dataset origin, then for the second dataset, I will provide the data origin and the experimental rig breakdown.

Maintenance

To prevent the wear of the device, the manufacturer provides a maintenance schedule, based on the statistics of the reliability tests done by the manufacturer. These guides are sufficient in most cases, but, since the schedule is constructed on statistics, it will not be adequate when using these components for mission-critical systems. There are two main issues with following the manufacturer schedule, we can either replace parts too soon, which will cost more, or too late, due to an unforeseen incident as mentioned before, which can result in a system failure. To address these issues, we need to consider using monitoring based prognostic health management methods

Prognostics

Prognosis is an emerging field in mechanical engineering, which is applied to many areas, aiming to predict the remaining useful life (RUL) of the system to minimize its maintenance and downtime. Prognostics is a science aiming to accurately detect early signs of degradation, as well as to analyze failure modes and fault conditions. The monitoring of the component condition is done *in-situ*, the data is collected from each component and subsequently analyzed using one or more of the following three categories of methods.

Experience based

Experience based – inferred via simple reliability functions, such as using Weibull law to analyze the time to failure [1-3] or utilizing Paris law to capture the rate of defect propagation [4-6].

Paris law

The general equation for Paris law is as follow:

$$\frac{da}{dN} = C(\Delta K)^m$$

Where:

a – crack length

N – load cycles

C and m – material coefficient

ΔK – stress range

This equation represents the relation between the crack growths over stress cycles to the stress range and two coefficients C and m. This function is widely used to correlate crack growth in many fields.

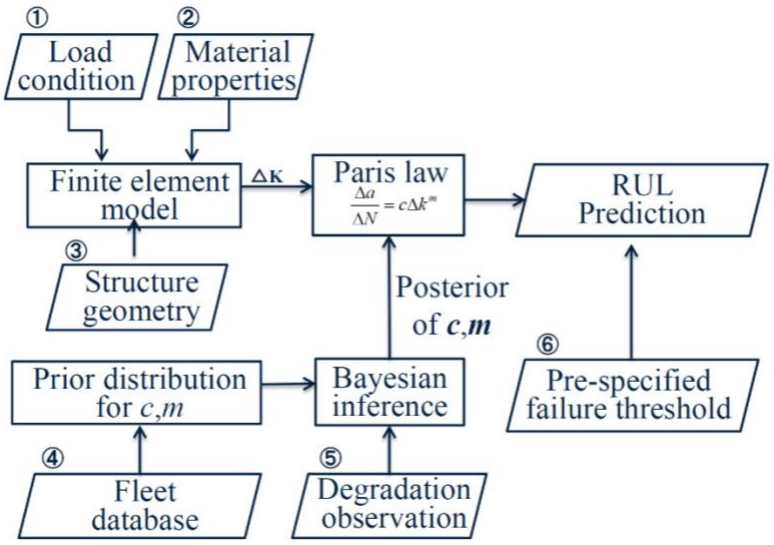

Figure 1 Wang et al process

For example, in their work Wang et al [7], used Paris law to determine the RUL of an aircraft structure. They used Finite Element Analysis (FEA) to determine the stress range for each structural element, and used previous fleet data and degradation observation to determine the coefficients (C and m). The entire process is shown in Fig.1.

Analytical model based

Analytical Model based - a set of differential or algebraic equations derived to represent the behavior and degradation of a system. This approach is based on the physics models that represent the behavior of the components and the system. This approach is been tested in different fields, for example, Yu et al[8] developed a new stress-based model of fatigue for bearings. Matej et al[9] used stochastic dynamical models to perform prognostics on gear health, and Celaya et al [10] developed a model-based methodology for predicting the remaining useful life of electrolytic capacitors. Analytical model based prognostics is broadly used in many fields. This method is especially preferred when dealing with a complex system. However, not enough data has been collected to date to reliably enable the application of a data driven approach.

Extended Kalman Filter (EKF)

The general form of EKF is as follow: $x_k = f(x_{k-1}, u_k) + w_k$ and $y_k = h(x_k) + v_k$ where both h and f are nonlinear equations. w_k and v_k are the process and observation noises, they both assumed to have zero mean, and u_k is the control vector. x_k is the predicted state and y_k is the predicted measurement.

Particle Filter (PF)

The main idea behind PF is the construction of the Probability Density Function (PDF) that will represent the state based on the available information. We create the PDF using a set of points (particles) that represent a sample of values from an unknown state with a related set of weights that represent the discrete probability masses. The set of points are recursively updated from the nonlinear process model, this process can be summarized as a technique of recursive Bayesian filter implemented with Monte Carlo (MC) simulation.

Data driven

Data driven based approaches convert sensors measurements (such as thermal, vibration acoustic emission, etc.) to quantitative information and subsequently utilize learning tools to estimate the RUL (such as Wavelet Packet Decomposition, WPD[11-13]). These learning tools include, but are not limited to, artificial neural networks[14-18] Hidden Markov models[19-21], and Support Vectors Machines[22-25], as well as some ensembled approaches that integrate multiple learning models for combined prediction[26]. The state of the art in the artificial intelligence field is deep learning, a family of machine learning approaches that utilize the artificial neural network to predict the RUL of a system.

Hidden Markov Models (HMMs)

Are stochastic models that enable modeling a Markovian process given hidden states. HMMs are based on augmenting a Markov chain, which is a model that learns and provides information on a sequence of random variables, or states (of the model) that have a finite set of values that could

be assigned to them. The assumption, on which a Markov chain is established, is that only the current state is important and useful for the prediction of the next state, whereas the previous states before it do not affect the prediction of the next state, only by affecting the calculation of the current state. Hence, HMM can be thought of as having 'short memory', where the past is not used for predicting the future, only the present. More formally, in a sequence of state variables X_1….X_(n+1), the Markov chain and HMMs only consider time point (n) when evaluating time point n+1.

$$\Pr\ (X\_(n+1)=j \mid X\_0=i\_0, X\_1=i\_1, \ldots, X\_n=i\_n) = \Pr\ (X\_(n+1)=j \mid X\_n=i\_n)$$

Also, HMMs require the pre-evaluation of the transition probabilities. Overall, HMM-based time series prediction relies on a short-term memory assumption, which may not be valid for RUL prediction, and require a separate step of comprehensive training.

Support Vector Machines (SVMs)

Is a supervised, discriminative machine-learning algorithm, that may be employed for either classification or regression tasks. SVMs are defined by finding a separating hyperplane, which categorizes the training examples into the defined two classes (labels). The basic training algorithm of the SVMs is meant to identify the separating hyperplane that maximizes the margin, which is the distance between the hyperplane and data points located on each side of the hyperplane (which would normally belong to different classes). The training points that are used to define the hyperplane by this function (and are hence located on the margin) are termed the support vectors (SVs). After training the classifier, new examples are classified based on their location relative to the defined hyperplane. If the two classes of training examples are not

linearly separable, it is possible to train SVMs with the following: (1) SVMs with soft margin, which uses a hinge-loss function, to allow some level of points that may be located on the "wrong" side of the margin, but minimizes such number of points. However, using SVMs with soft margins would normally still require an "almost linear" separation of the training examples. Alternatively (2) using a Kernel function can fit a maximum margin hyperplane to a transformed feature space, where the transformation applied to the feature space may be non-linear and high dimensional. However, this requires knowing the transformation that would make the data separable (or the dimension/function that would separate the data) which is highly challenging.

Hence, SVMs are not ideally suitable for time series prediction, especially when there may be non-linear relations between variables that are relevant for prediction.

Literature review

After reviewing the different approaches for prognostic and RUL estimation, I decided to focus on a fused data and physics-driven method that showed promise in improving the estimates RUL in complex systems. Recently artificial neural network has shown tremendous success in solving complex problems such as; computer vision[27, 28], speech recognition[29-31], and medical diagnosis[32-34]. After reviewing the different capabilities of ANN we looked for an ANN architecture that will fit our problem, since we have a time series prediction we looked into different methods that solve this type of problem. The most used architecture of ANN for this type of problem is the recurrent neural network[35-38]. However, this method is not optimal due to the explosion or vanishing gradient (which will be discussed in the RNN chapter). To overcome this issue we chose the long short term memory (LSTM) architecture, this structure

allows us to maintain long term memory and showed great potential in similar problems as ours in time series prediction in different fields. Pankaj et al.[39] showed the capability of RNN LSTM to detect anomalies in data. They applied the network on four datasets: (viz. ECG, space shuttle, power consumption, and engine sensors data), where they showed high precision (over 93%) in all of them. The results were also compared to standard RNN and showed that LSTM is more accurate overall, for example for the space shuttle dataset LSTM scored 0.93 compared to 0.89 with RNN. Xiaolei et al.[40] compared different methods to predict traffic speed. In their work they applied the following methods; Auto Regressive Integrated Moving Average (ARIMA), Kalman filter, Support Vector Machine (SVM), Elman NN, Time Delay Neural Network (TDNN), nonlinear autoregressive with exogenous inputs (NARX), and LSTM. They showed by comparison that LSTM is the best fit for that type of problem (time series prediction). Thomas et al.[41] deployed the LSTM network to predict stock directional movements in price, by doing so they showed that LSTM can perform on large datasets, they used S&P 500 from 1992 to 2015, also they showed that LSTM outperforms random forest, deep net, and simple logistics regression. After reviewing this literature I could carefully consider the advantages and disadvantages of the approaches for time-series prediction and I could make an informed decision that LSTM would be the most fitting choice for predicting RUL. All of these different approaches facilitate the prediction of the RUL and were widely utilized in several different fields. Because our first dataset is based on the accelerated aging run to failure MOSFET data [42] that is analyzed and learned with a neural network, this section will be focusing on RUL estimation with the following literature:

- Applying a recurrent neural network to predict the RUL of bearings [43].

- Recent (2019) work done on similar MOSFET run to failure data using different neural network methods [44].
- A similar neural network method was applied to Lithium-Ion batteries [45].
- The paper that published the dataset we used and compared the performance of three algorithms, two model based methods; extended Kalman filter and particle filter, and one data driven method based on the Gaussian process regression framework [42].
- A dual-LSTM framework combining change point detection and remaining useful life prediction [46].
- Remaining Useful Life Prediction for Rolling Bearings Using EMD-RISI-LSTM [47].

Liang et al. bearings RUL prediction using LSTM RNN

Liang et al. used Long Short Term Memory (LSTM) networks, a type of Recurrent Neural Network (RNN), to predict the RUL of bearings. They started by acquiring run to failure data of eight bearings, then they extracted eight relevant features from that data. Next, they added six related similarity features to overcome the large variance of classical statistical features, and they used feature selection and fusion to find the most relevant features, which yielded eight final features to be used. These features were normalized by using the distribution of these features when measured during initial inspections. They used the normalized features from the data obtained from eight bearings to train an LSTM RNN network. After training the network, it was applied to predict the failure time and the RUL of a test set that was left out and unseen during training. They compared their results with a self-organizing map (SOM) and the results are shown in the following table:

Testing dataset	Current time (s)	True RUL (s)	Predicted RUL (s)	RNN error (%)	SOM error (%)
Bearing1_3	18,010	5,730	3,520	43.28	-31.76
Bearing1_4	11.380	2,900	1,100	67.55	62.76
Bearing1_5	23,010	1,610	1,980	-22.98	-136.03
Bearing1_6	23,010	1,460	1,150	21.23	-32.88
Bearing1_7	15,010	7,570	6,220	17.83	-11.09
Bearing2_3	12,010	7,530	4,680	37.84	44.22
Bearing2_4	6,110	1,390	1,660	-19.42	-55.40
Bearing2_5	20,010	3,090	1,410	54.37	68.61
Bearing2_6	5,710	1,290	1,470	-13.95	-51.94
Bearing2_7	1,710	580	900	-55.17	-68.97
Bearing3_3	3,510	820	790	3.66	-21.96
Mean of error				32.48	53.24

Table 1 RNN and SOM RUL predictions

We can see from the table that LSTM RNN predictions were better than SOM. After analyzing the test dataset Liang et al. Used the trained LSTM RNN and SOM on a new industrial available dataset of generator bearings of wind turbines. The results of that test are shown in the following table:

Testing dataset	Current time (h)	True RUL (h)	Predicted RUL (h)	RNN error (%)	SOM error (%)
FJ1	300	183	108	40.9	-210.35
FJ2	120	58	18	69	28.96
FJ3	135	40	38	5	-117.5
FJ4	130	48	46	4.1	-8.33
FJ5	320	166	124	0.25	-14.16
FJ6	330	109	131	-20.2	22.94
Mean of error				23.24	67.09

Table 2 RNN and SOM RUL prediction on a new dataset

Again, we see that LSTM RNN predicted more accurately the RUL compared to SOM. In conclusion, we can learn that LSTM RNN can be used to predict the RUL. Since this paper did not share the actual procedure used to obtain these results, it is hard to understand why their results are not as accurate as we believe can be achieved using this method. The only changes we can offer to improve their results will be to first minimize the output range and normalize it, as LSTM RNN can achieve much higher accuracy when the output range is smaller, and secondly, we will suggest increasing the dataset size by randomly sample it.

Karkulali et al. MOSFET prognostics using feed-forward neural network

In their work, Karkulali et al. used a feed-forward neural network with three inputs one output, and one hidden layer that contain from one to forty hidden neurons. Their data contained only two modules with a limited number of data points, they separated the data into training and test datasets and were able to achieve almost 85% accuracy on these modules. The accuracy was measured using the following equation: $RA = 1 - \frac{|True\ RUL - Predicted\ RUL|}{True\ RUL}$ when RA is the relative accuracy. They summarized their sensitivity test on a different number of hidden neurons (N_h), their computational time and their relative accuracy, using the equation mentioned, in the following table:

N_h	Computational time (s)	Relative accuracy
1	104.89	0.8413
5	107.32	0.8493
10	109.88	0.8049
20	113.73	0.7865
30	135.24	0.7909
40	150.72	0.4939

Table 3 computational time vs accuracy

When applied their network on a new dataset (same module), that had different behavior the network could not estimate the RUL correctly. Their main issue was the lack of data, a small dataset as used in this work is not adequate to allow the network to train properly. Another issue

is the use of a feed-forward network, in which the data is fed to the network as steps and there is no memory in the network, this can cause the training of the system to create dependencies between steps and creating overfitting. Given the drawbacks mentioned above the results of Karkulali et al. are less relevant to this work but were mentioned because they addressed MOSFET RUL estimation using a neural network.

Estimating RUL of Lithium-Ion batteries using RNN LSTM

Yongzhi et al. used long short term memory recurrent neural networks to predict the remaining useful life of lithium-Ion batteries. They used two layers of LSTM, the first layer contained fifty hidden units, and the second layer contained one hundred hidden units. This is a large network and its large size can cause an overfitting issue, as presented in their work, to overcome this issue Yongzhi et al. used the dropout method. This method, which was developed by Google, prevents complex co-adaptions when training the network and helps to reduce the risk of overfitting. By comparing their results to SVM and simple RNN they showed the advantage of using LSTM RNN for predicting RUL. Their network attempt to estimate the RUL on four cells in two scenarios the first was after training the network on fifty percent of each cell data and the second was done after training on seventy percent of each cell data both without offline training data, which means that each network was trained and based on separate cell. Also, they estimate the RUL of cells one and three when using offline data that was obtained from an additional two cells, which underwent the same conditions as cells one and three, for training the network. Their results are as follow:

The results of cells 1-4 without offline data using RNN LSTM compared to SVM and simple RNN are shown in the table below.

	Cell 1		Cell 2		Cell 3		Cell 4	
Method	Stating cycle	Error	Stating cycle	Error	Stating cycle	Error	Stating cycle	Error
LSTM RNN	253	-3	285	48	289	35	278	58
	354	15	399	26	404	19	389	14
SVM	253	-21	285	-34	289	40	278	-47
	354	30	399	42	404	23	389	15
RNN	253	135	285	195	289	181	278	-
	354	78	399	95	404	72	389	88

Table 4 cycles error with different methods

It shows that in most cases LSTM RNN is superior to the other methods and can predict more precisely the RUL of the cells. As mention before Yongzhi et al. also used offline data to create a network and then used a smaller portion of the data (120 cycles) to predict the RUL of cells one and three, the results were compared to particle filter (PF) method and simple RNN that used the same portion of the data. The results of that comparison are shown in the following table.

	Cell 1 Error	Cell 3 Error
LSTM RNN	-19	40
PF	52	58
RNN	110	93

Table 5 cycles error with different methods

Again, it shows the advantage of using LSTM RNN over other methods.

In summary, this work showed the potential of LSTM RNN in predicting RUL, although they may have used the too large network to analyze the data, as the data does not seem large enough, and had to overcome this issue by applying the dropout method they managed to show the advantage of LSTM RNN in predicting time series data.

Power MOSFET RUL prediction comparison between two model based and one data driven methods

Celaya et al. were the same group from NASA that collected the data that we are using in our work. In their work they made the following assumptions:

- They used $\Delta R_{DS(on)}$ as their single health indicator feature.
- The die-attach failure mechanism is the only degradation that happens during accelerated testing.
- $\Delta R_{DS(on)}$ is responsible for the degradation from nominal through failure condition.
- 0.05 increase in $\Delta R_{DS(on)}$ is set as a failure threshold.
- Each prognostic tool, out of the three that were used, will predict the RUL from a set time point, hence the future load is estimated to remain similar.

In their work, they used the same accelerated aging data of MOSFETs and it was analyzed using the following three methods:

Gaussian Process Regression (GPR): this data driven approach is used to estimate degradation state using the training measurement data. To predict the state of degradation we will first

assume a prior distribution using prior knowledge [48], then we will modify the distribution to fit our measurements with a probabilistic function for regression over the training data [49]. The result of this process will be a mean function that describes the behavior of the data and additional functions that describe the uncertainty. EKF and PF were explained in the previous chapter.

After applying these three methods, they summarized the results in the following table:

t_p	RUL	GPR	EKF	PF
140	88	N/A	64.95 (23.02)	77.65 (10.35)
150	78	N/A	80.22 (-2.22)	65.85 (12.15)
160	68	N/A	56.64 (11.36)	58.33 (9.67)
170	58	N/A	50.15 (7.85)	49.47 (8.53)
180	48	73.2 (-25.2)	42.75 (5.25)	38.68 (9.32)
190	38	33.4 (4.6)	30.35 (7.65)	27.14 (10.86)
195	33	17.6 (15.4)	18.57 (14.43)	24.76 (8.24)
200	28	14.6 (13.4)	17.24 (10.76)	21.09 (6.91)
205	23	13.8 (9.2)	18.28 (4.72)	16.66 (6.34)
210	18	11.8 (6.2)	13.46 (4.54)	14.68 (3.32)

Table 6 RUL prediction with error in parentheses

This work presents the results that were acquired using the same data as used in our work, all of the techniques were underestimating the RUL and can result in an earlier replacement of parts to

avoid failure. The main setback of this work is the assumption that the load will continue to be the same. This assumption could not be applicable when estimating a real-time working module.

A dual-LSTM framework combining change point detection and remaining useful life prediction

Shi et al. used two-stage prediction, first they normalize the data, smooth it, and selected the sensors, and then they added the RUL labels. They trained two classifiers, the first classifier was trained to recognize the start of degradation, when the first classifier indicates it the second classifier is used to determine the RUL. The full flowchart of the process can be seen in the following figure:

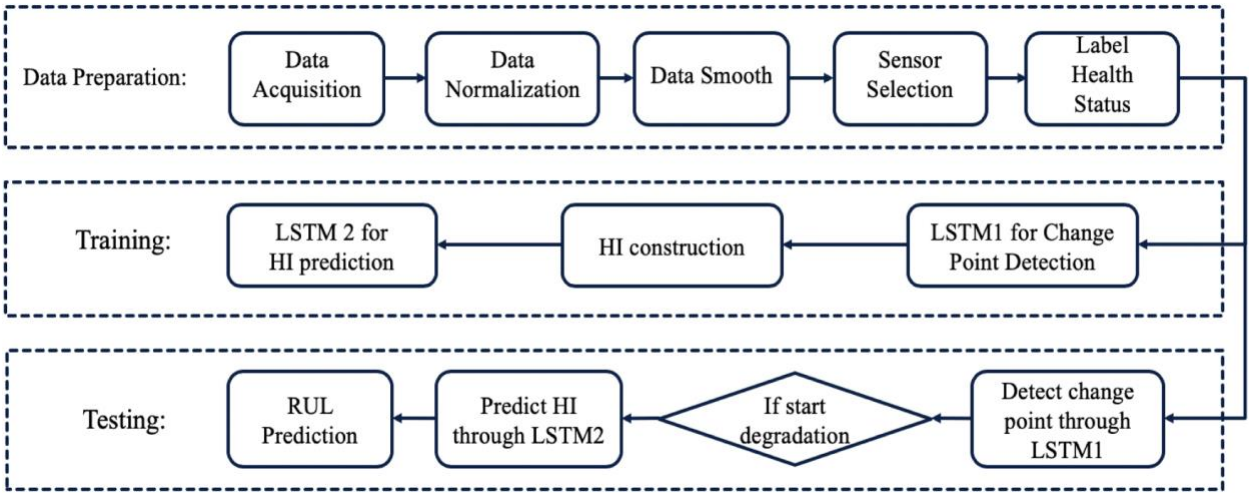

Figure 2 Flowchart of the proposed Dual-LSTM framework [46].

This method was used on two publicly available turbofan engine degradation datasets. They compare the results of their method with Vanilla LSTM, RNN with fixed change point, and

LSTM with fixed change point. The comparison between the methods can be seen in the following figure:

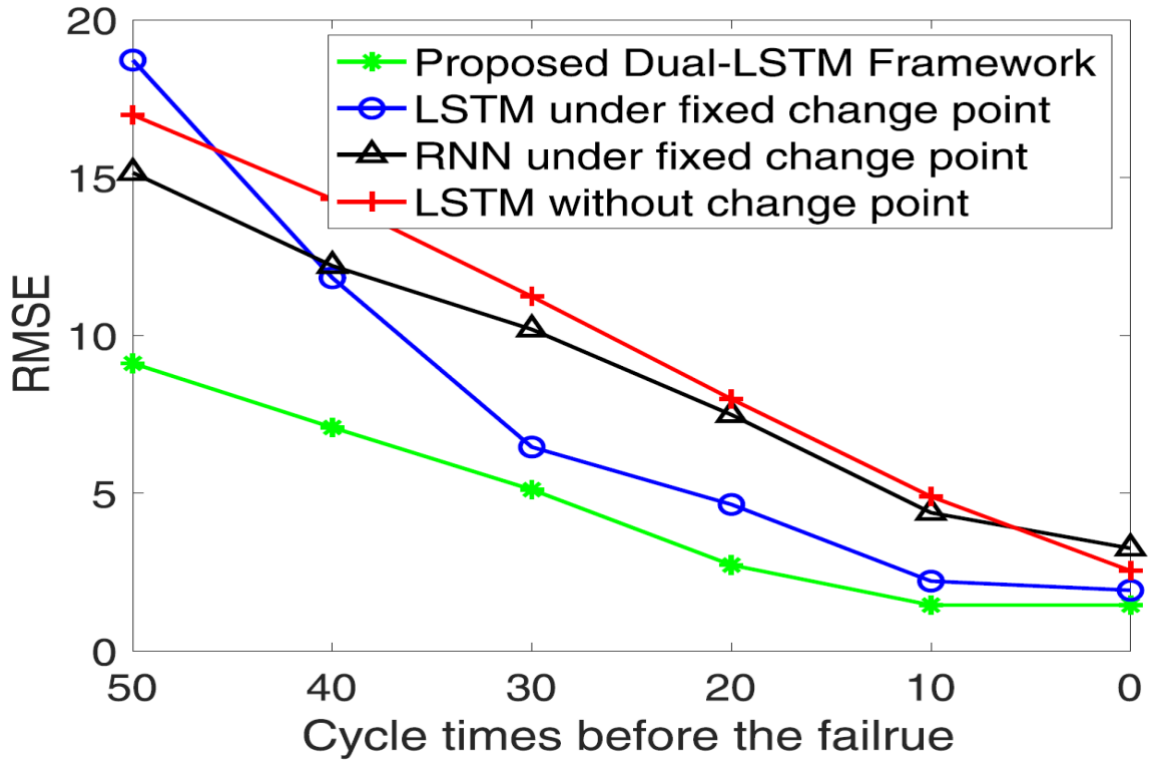

Figure 3 RMSE comparison [46]

The results show that in most cases LSTM is more accurate than RNN for predicting RUL and that adding stages for predicting the RUL is increasing the prediction accuracy.

Remaining Useful Life Prediction for Rolling Bearings Using EMD-RISI-LSTM

In their work, Guo et al. applied empirical mode decomposition (EMD) and long short-term memory (LSTM) network, to improve accuracy and robustness under different working conditions. The architecture integrates three parts. First, the failure vibration signal decomposed

into several intrinsic mode functions (IMFs), a residual through EMD decomposition, in parallel, to select the IMFs with more degradation features, a new method named RISI (representative IMF selection index), which is based on cosine similarity, and Euclidean distance. In the next step, the LSTM model is trained for each IMF and residuals. The final step will be using the selected IMF prediction and utilize it to determine the RUL prediction. The entire process can be seen in the following figure:

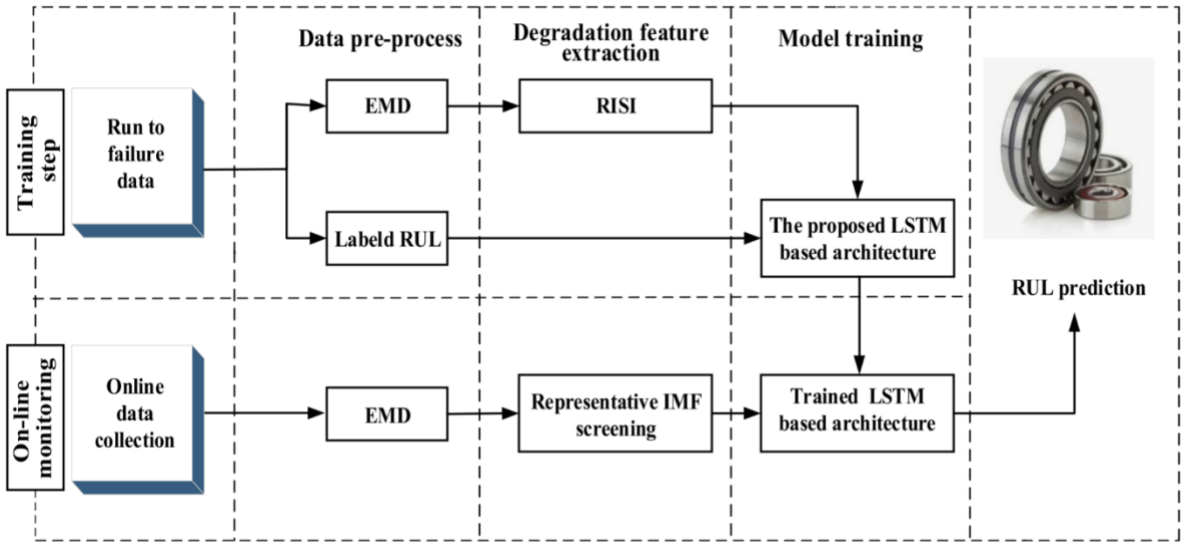

Figure 4 Guo et al. proposed framework [47]

To evaluate their proposed framework they compared their method with EMD-LSTM, LSTM, and BPNN (back propagation neural network). The results can be seen in the following table:

	Current time (s)	True RUL (s)	Predicted RUL (s)	Guo et al. (%)	EMD-LSTM (%)	LSTM (%)	BPNN (%)
Bearing 1-3	18010	5730	4740	17.28	24.35	43.28	-31.76
Bearing 1-4	11380	2900	1730	40.34	60.43	67.55	62.76

Bearing 1-5	23010	1610	2050	-27.33	-24.54	-29.76	-136.03
Bearing 1-6	23010	1460	1960	-34.25	-24.65	35.76	-32.88
Bearing 1-7	15020	7570	7180	5.15	13.98	17.28	-11.09
Bearing 2-3	12000	7530	8410	-11.69	-16.87	-19.43	44.22
Bearing 2-4	6100	1390	1830	-31.65	-22.74	54.23	-55.40
Bearing 2-5	20010	3090	3370	-9.06	20.34	27.28	68.61
Bearing 2-6	5720	1290	1470	-13.95	-15.98	-16.97	-59.49
Bearing 2-7	1710	580	290	50	-52.57	-55.17	-68.98
Mean value				22.10	29.87	32.48	40.65
Score				0.31	0.29	0.26	0.06

Table 7 Evaluation and analysis of prediction results of four models

The results show that for RUL predictions, LSTM is more accurate than BPNN and that by improving the input of the LSTM we can further improve its accuracy.

Our approach

After reviewing all the relevant literature, I reasoned that deep learning approaches are preferred for estimation RUL, and thus chose a deep learning approach fused with a physics-based approach that was applied to the data using my knowledge of different failure mechanisms. I searched for a dataset that is large enough to allow me to properly train the network. After reviewing the different neural network structures and their applications, I decided that LSTM RNN would be the optimal fit to address the challenge of RUL prediction. I found two datasets the first was from the NASA website of accelerated aging of MOSFET that includes 42 modules the second was filtration system run to failure given by

PHME2020 fifth European conference of the prognostics and health management society 2020, as part of the PHME2020 data challenge. At first, we tried applying RNN LSTM on the first dataset but we were unable to convert the training and achieved very poor results, after reviewing the lifespans of the modules we realized we need to find a better way to train the network, as we do not have enough data to train the network. We realized that with the amount of data we have, we could not train a network that can answer such a complex question. Therefore, we decide to simplify the question and ask the classifier to answer a simpler question. Instead of asking what the RUL is in each time point we asked did we passed a certain threshold at that point. By changing the question to a simpler question, we were able to use a much smaller dataset and achieve a more accurate result. After changing the question we so dramatic increase in the accuracy, but, we still had the issue of the wide lifespan range. We saw that issue since we had high accuracy in modules with similar lifespan when selecting the thresholds that were in their lifespan. In order to overcome that issue, we changed the thresholds to quantiles, that change allowed us to accurately predict RUL regardless of the lifespan range. To prevent overfitting of the training data, we used a relatively small network with a single LSTM cell and five hidden units, which has a limited number of parameters and hence is less prone to memorization and overfitting. This small architecture further allowed a robust prediction without any filtering techniques, such as dropout, which were necessary for previous studies. To further enlarge the dataset that we had, we randomly segmented pieces of the data, which facilitated a fivefold increase in the sample size. After the training of the network, we were able to achieve high accuracy on the test set that was constructed out of never-seen modules, which were entirely left out during training. To describe the approach that enabled this level of accuracy, the next chapter will

provide the background and basics of neural networks and backpropagation, with a focus on recurrent neural networks and long short-term memory algorithm. This review would provide the background and reasoning to the approach designed here, and why it is highly fitting to resolve the RUL prediction problem.

Deep learning

Deep learning enables a non-linear separation using multiple levels of abstract data via multiple processing layers of computational models. Part of the learning process is the representation of the data, which should be optimized, to boost the ability of the network to process and learn the data. The representation of the data is fed into different layers of network architectures, each, in turn, applies non-linear functions, and learns how to weight different parts of the dataset, to optimize the prediction objective. The commonly used objective in neural networks is the minimization of a loss function. These methods allow the community of artificial intelligence an advance in solving different complex problems that were unsolvable for many years and not amenable to standard techniques. Deep learning tools have a unique capability in uncovering and learning complex structures in high-dimensional data. Naturally, with the recent increase of high computational power, they became broadly used in various fields, including business, government, and science. These approaches are currently considered the state of the art for speech recognition [50-52], image recognition [53-56], and outperform many machine-learning algorithms in predicting reliability [43, 45], clinical applications [57], etc. Deep learning is a wider family of machine learning methods that are based on neural network architecture.

Artificial neural network (ANN)

This architecture was inspired by biological systems, although there is no real resemblance between artificial neural networks (ANN) and the neurons observed in biological systems. The purpose of a neural network is to alter input data into the desired output. The structure of the network is based on weights and non-linear functions, the data is inserted into the network through the input layer, and each node is multiplied by the weight of that node and applied a

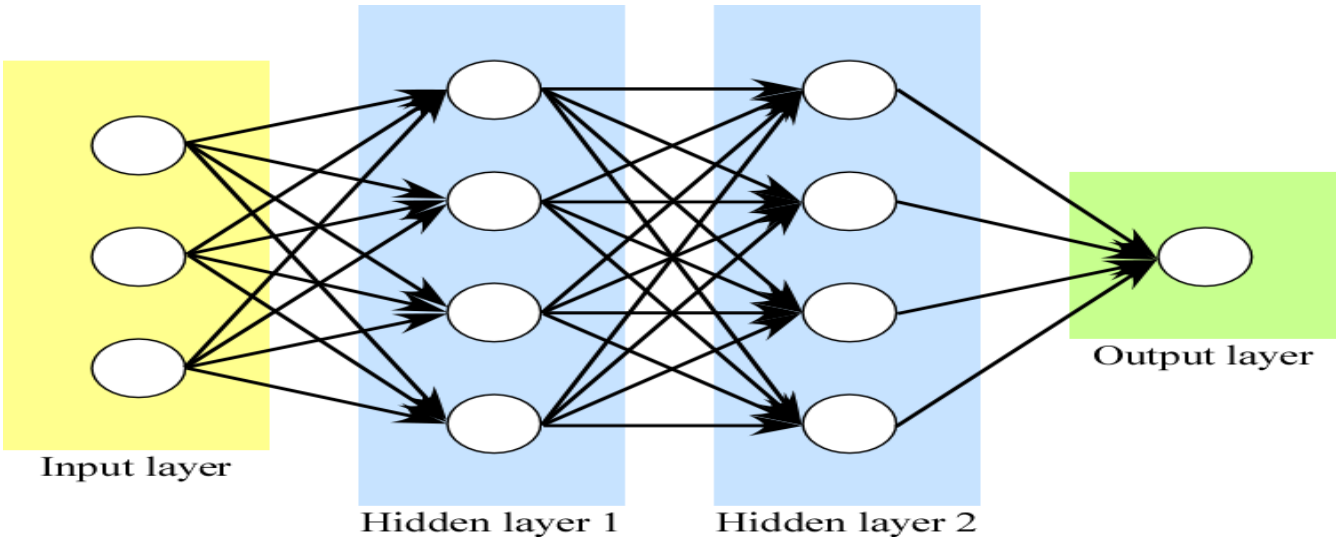

Figure 5 fully connected network

non-linear function the results is then fed to the next layer along with the results of all the other nodes. This procedure continues in each layer, input and hidden layers until reaching the output layer as shown in Fig.5.

This type of network is called the vanilla network and each step can be represented by the following function:

$$h_i = \mathrm{F}(W_i * x_i)$$

Where:

x_i is the input to the i node.

W_i is the weight of the i node.

h_i is the output of the i node.

And F can be either one of the following non-linear functions: tanh, sigmoid, and RELU (Fig.6).

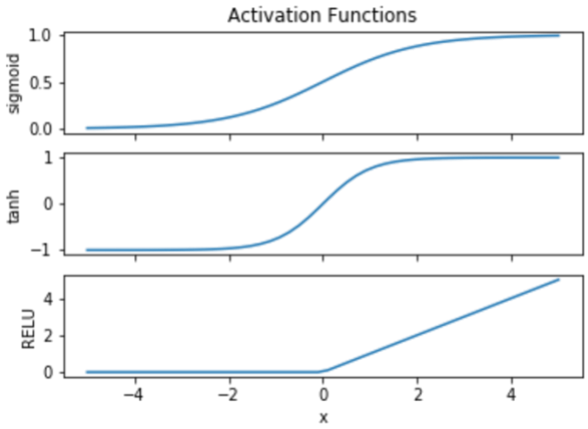

Figure 6 Functions for ANN

Each of these functions is usually used for a different purpose, for example; for recurrent neural networks, we will use tanh, for image recognition RELU, etc.

To use the network we must first train it. The training is done by adding labels to the data that define the required output from the system. For example, if we want the network to distinguish between a dog and a cat, we will have to label the dog images as dogs and the cat's images as cats. Then we will feed the images to the network, after each iteration the network will know if it was able to distinguish between the images and if the network was wrong it would modify the weights to adjust the output. After we feed the network with enough images and let it run the required number of iterations, the network should be able to decide when seeing new images (not the ones used in training) to distinguish between a cat and a dog image. Although this procedure seems simple, we need to understand that weights change is done to each node and each node will be usually fully connected to the next step (fully connected network) as seen in Fig.5. In each step, we will apply the nonlinear function and after each iteration, we will modify the weights from the first layer, this process requires heavy computational abilities and is the main reason why it was neglected for a long

time. To overcome that issue a more simple calculation approach was needed, it was only in the mid-1980s where the backpropagation was introduced.

Backpropagation

Ever since pattern recognition started, researchers aimed to replace their hand calculations with a trained multi-layer network. It was only until the mid-1980s that the solution was developed. Several different groups during the 1970s and 1980s [58-60] discovered the idea that this could be done, and that it worked, independently. The idea was to train multilayer architectures by simple stochastic gradient descent. The procedure of backpropagation that is used to compute the gradient concerning the weights can be seen as the chain rule for derivatives. This approach reduces the computing resources needed to train the network dramatically, as it computes the gradient change backward with respect to the output, and once we compute the gradients, the computations of the weights of each module will be straightforward. Even though this method reduces the computational load, it was only until 2010 when the computational cost was reduced to a point were ANN was considered again.

Recurrent neural network (RNN)

The architecture that gained the most out of backpropagation was RNN. This method is optimal for speech and language as they involve sequential inputs. RNN processes the input elements in order and integrates the output of each element to the input of the next one, which will require even more gradient and weights computation, hence the advantage of backpropagation. The

distinctive structure of RNN, and its training way, allows it to fit perfectly when needed to predict the next word in a sentence [61], the next character in a word [62], or even more complex tasks such as; predicting the remaining useful life of an ion battery [45], bearings [43] and more. RNN can be seen as a deep feedforward network, once unfolded, where the layers share the same weights. Its structure can be seen in the following equation:

$$h_t = F\left(W_i \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Where:

x_t is the input of that time step.

W_i is the weight of the i node.

h_t is the output of that time step.

h_{t-1} is the output of the previous time step.

F is the same as mentioned in ANN

Although their purposes are to learn long-term dependencies, it is not the case. One of the reasons for the lack of long-term memory can be found in the backpropagation method. Even though backpropagation is beneficial to RNN, it has a major flaw, due to a large number of time steps, the gradient can grow or shrink drastically and cause it to explode or vanish. This issue led to the development of the Long Short Term Memory (LSTM) method.

Long Short-Term Memory (LSTM) machines

Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Networks that use a time series as input for various prediction tasks, and is composed of the following components:

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_c(W_f x_t + U_f h_{t-1} + b_f)$$

$$h_t = f_t \odot \sigma_c(c_t)$$

Where the initial values are $c_0 = 0$ and $h_0 = 0$. $\odot$ denotes the Hadamard product. X_t are the input vectors to the LSTM unit (sensors measurements). f_t, i_t and o_t are the activation vectors for the forget gate, input gate and output gate, respectively. h_t is the output vector of the LSTM unit, and c_t is a cell state vector. W and U are the weight matrices and b are the bias matrices that are learned during training. σ are the non-linear functions, where σ_g is a sigmoid function and σ_c is the $tanh$ function [63].

Although LSTM requires additional cells (extra gates) it enables the solution for the vanishing and exploding gradient by allowing the cell to forget and to change the inner weights of different cells. Also, the structure allows it to overcome long time lags, discard noise, and much more. Unlike Markov models, LSTM can handle unlimited state numbers. LSTM is also able to distinguish between two or more widely separated occurrences.

LSTM networks have shown remarkable results in many fields compare to traditional RNNs when having several layers for each time step. It is widely used in encoders and decoders, such as Google translate, and many more machine translations.

The structure of the LSTM cell is as follow:

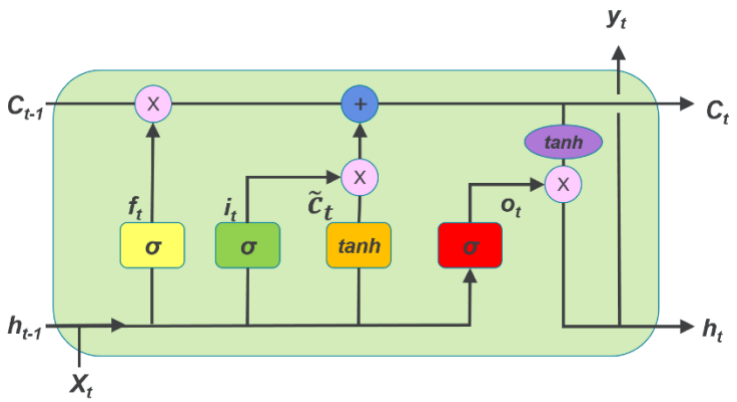

Figure 7 LSTM cell inner structure

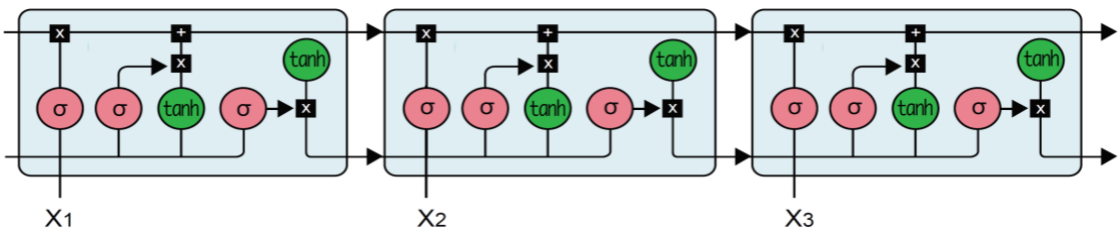

Figure 8 LSTM network

The function of LSTM is shown above and can be summarized:

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

Each letter represents a gate where i is the input gate its value indicates how much we write to the cell. f is the forget gate which indicates how much we delete from the cell. o is the output gate which indicates how much to reveal the cell and finally g is called the gate gate, and it indicates how much to write to the cell.

Project one: Quantile-based LSTM Remaining Useful Life prediction of MOSFETs.

Background

Power modules

The heart of a power electronic system is the power semiconductor switching module. Power modules can be based on many different semiconductor switching technologies:

- Silicon Metal Oxide Semiconductor Field Effect Transistor (MOSFETs) (1980s).
- Silicon Insulated Gate Bipolar Transistor (IGBTs) and MOS Controlled Thyristors (MCTs) (1990-2000s).
- Silicon Carbide MOSFETs and GaN High Electron Mobility Transistors (HEMTs) for higher frequency applications.

Our work will focus on power MOSFET modules.

Power MOSFET

General characteristics

Standard n-channel MOSFETs have four terminals, source (S), gate (G), drain (D), and body (B) terminals as shown in fig.9. The terminals are usually reduced to three, since the body (B) terminal is typically connected to the gate (G) terminal, to prevent it from a free float, which can limit transistor control. The gate voltage controls the depth of the channel, through a capacitor formed by the gate substrate and a thin layer of (yellow line in fig.9) SiO2 (grown or deposited) or rare earth metal oxide (e.g. HfO2). The potential of the gate is relative to the source. The electrons are responsible for the main power current flow through the source terminal and exit through the drain terminal.

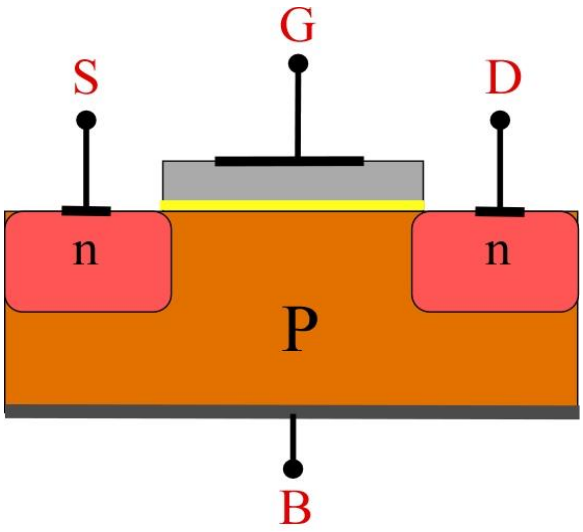

Figure 9 MOSFET structure

MOSFET lifespan under standard conditions is too long for lab testing since the MOSFET fabrication process improved over the years, hence an accelerated testing approach is typically used. When using accelerated testing, as the data we used for this work, it is important to make sure that the failure mechanism in the accelerated test remains the same as in the standard conditions.

Power MOSFET failure mechanisms

The power MOSFET structure is different from a normal MOSFET structure since it handles high power levels. Power MOSFETs are structured, in most cases, vertically as seen in fig.10, compared to the normal MOSFET planar structure as seen in fig.9. This unique structure of the power MOSFET causes several failure modes for example; Gate oxide breakdown: to allow an increase in switching speeds, manufacturers had to decrease the gate dielectric thickness. This reduction caused the gate oxide region to be vulnerable to damage from high gate voltage. Hence, when operating power MOSFET with gate voltage values that are above device limitation the power MOSFET will be subjected to a reduced lifetime and it may even cause immediate failure. Maximum drain to source voltage: power MOSFETs specifications indicate the maximum drain to source voltage, exceeding that voltage can cause breakdown and may damage other circuits in the device due to higher power dissipation. These limitations may be breached by many causes that are not design-related or even usage-related such as, electrostatic discharge which may cause exceeding the gate voltage limit and can cause instant failure [64].

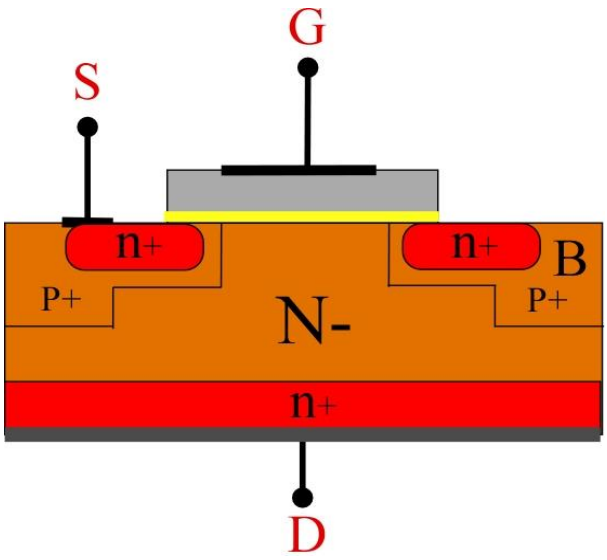

Figure 10 Power MOSFET structure

In this work, the MOSFETs dataset was used where the major failure mechanism was die-attached degradation, which is typical for discrete devices with lead-free solder die-attachment. The data was given by the work of Celaya et al. [42] where they determined in their experiment that die-attach degradation was the failure mechanism resulting in an increase in ON-state

resistance due to its dependence on junction temperature. Increasing resistance, thus, can be used as a precursor of failure for the die-attach failure mechanism under thermal stress, therefore the Rdson was measured as the resistance increased as the die-attach degrades under high thermal stresses.

Dataset one

The dataset used throughout this project is of accelerated aging of MOSFET IRF520Npbf (TO-220 package), where the failure mechanism is die-attach degradation due to thermal and power cycling stress conditions, which is typical for lead-free solder die attachment [23]. This failure is associated with a mismatch between the coefficients of thermal expansion (CTE) in the component structure, which induces thermo-mechanical stresses to the component caused by the thermal cycling overstress. The dataset, which describes thermal overstress applied to the devices to achieve accelerated aging, was download from the NASA website [24]. The failure conditions were thermal run-away, Latch-up, and failure to turn ON due to loss of gate control. The thermal cycles were induced by not using an external heat sink, thus substantially reducing the heat dissipation capabilities. The thermal cycling was controlled using a thermal sensor on the device case. The power cycling had a gate voltage which was a square wave signal with an amplitude of 15V, a frequency of 1KHz, and a 40% duty cycle. The drain-source was biased at 4V dc and a resistive load of 0.2 Ω and was used on the collector side output of the device [22]. Temperature control was employed within the low and high ranges. The aging process used is described in detail in [25], and the accelerated aging methodology is presented in detail in [26]. The complete accelerated aging system diagram is shown in Fig.11

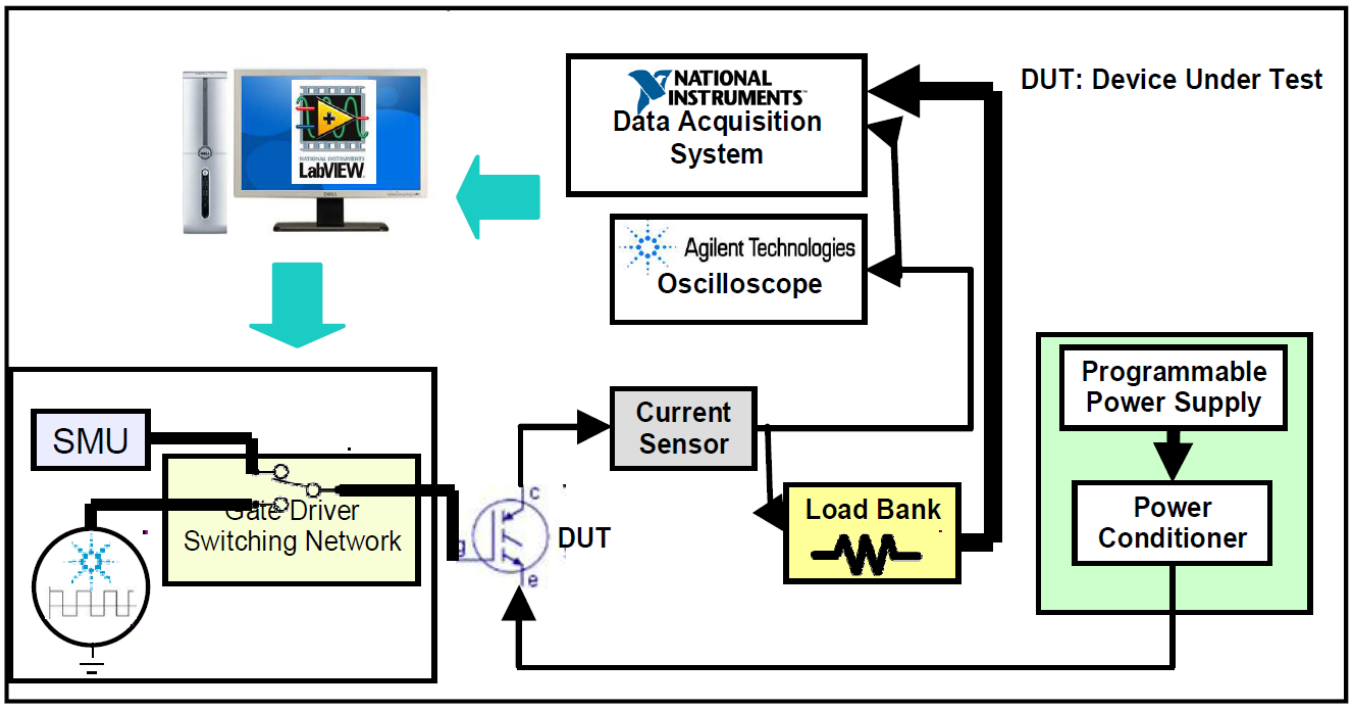

Figure 11 complete accelerated aging system diagram [42]

Procedure

Data representation

To enable the utilization of LSTMs to predict the RUL given run-to-failure data, we first represent the measurements that will be provided as input to the LSTM. The data utilized includes seven features, summarized in Table 8. The transient features were consistently collected during the entire experiment, whereas the steady-state data was not, and thus containing time gaps. Therefore, the five steady-state features and the two transient features were synchronized by the steady-state data point. Each transient data point was assigned with a steady-state value by finding the steady-state sample with the closest time to the transient data point. Next, the drain voltage was divided by the drain current (square signal), to obtain Rds(on). We used the 0.25 and 0.75 quantiles to produce the high and low values of each pulse and remove noisy outliers, which were utilized as the transient input features.

Feature	Type
Supply voltage	Steady state
Drain-source voltage	Steady state
Drain current	Steady state
Package temperatures	Steady state
Flange temperatures	Steady state
Drain current	Transient
Drain voltage	Transient

Table 8 The features included for RUL prediction.

The seven features for each model M_i yielded a feature table of size 7Xn_i for each module, where n_i is the number of time points measured for that module. The RUL for each time point in n_i was converted into constant intervals of timeline data. The gaps (stoppage of the experiment at the end of each day) were removed to maintain an "active time" measure (i.e., the actual time that the module has been operating). Therefore, each time point t in a M_i table was assigned with a distinct RUL, denoting how much more "active time" module M_i has left until failure.

To allow RUL prediction at different time points in the life of a module, the converted run to failure data described above was segmented, to produce multiple fractions of measurements for each module. The segmented data was used for training and testing, where each data point describes a certain time frame of measurements. Therefore, each feature table of module M_i was segmented into 7X(n_i/k) tables, where $n_i$is the number of time points measured, and K was set

50 (through parameter optimization on the training dataset), resulting with ~n_i/50 segments for each module.

Due to large and inconsistent measurement intervals and measuring errors, for instance, fault readings from temperature sensors (over 1000 degrees), 17 modules were discarded and 25 were used for training and evaluation. The training dataset that was used to train the network included 16 modules. The performance was evaluated on the validation dataset, including 4 modules, for parameter tuning and algorithm design. Eventually, and only after the final approach has been established, it was tested on the test dataset, of five modules that were unseen during the entire training process.

Module number	Set assignment	Lifetime minutes
1	training	186.0678
2	training	146.6166
3	training	859.1858
4	training	704.5452
5	training	135.5832
6	training	284.4624
7	training	215.8803
8	training	231.8026
9	training	614.3334

10	training	308.783
11	training	88.28323
12	training	320.5675
13	training	437.0867
14	training	477.2924
15	training	299.7784
16	training	388.3094
17	validation	950.0717
18	validation	897.989
19	validation	821.034
20	validation	868.3347
21	test	1404.347
22	test	311.559
23	test	358.1089
24	test	395.9074
25	test	290.7152

Table 9 The MOSFET modules used throughout this study for training, validation, and testing, and the respective lifespan of these modules

Training process

Because binary classification tasks can achieve higher performance rates when trained with smaller datasets [65], compared with complex classification tasks such as directly predicting the RUL, we re-define the classification problem of the RUL prediction as a set of binary classification problems. This is crucial given the relatively small dataset that does not facilitate training a predictor to directly predict the RUL (Fig.12). We first train a sequence-to-sequence LSTM to predict, from the input data features, what is the precise remaining useful life (RUL) at each segment of measurements (each segment having 100 data points). We used the training data to train the LSTM, and the validation to evaluate the performance, via (1) RMSE (Root Mean Square Error) and (2) correlation between the predicted RUL and the actual RUL. We trained LSTMs with different hyper parameters: setting the number of hidden units to (5,50,100,500) and the number of epochs to (10,20,50).

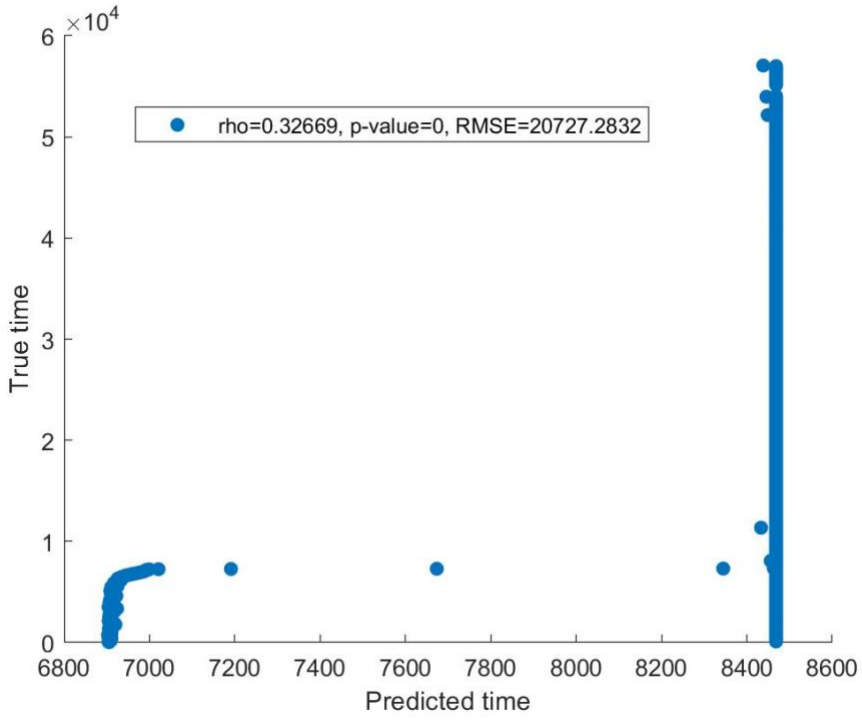

Figure 12 Naive LSTM

Therefore, we define binary classification tasks that are amenable given the shortage of data, and use them in combination, to predict the precise RUL. Four classifiers were trained, to predict when a module has reached four stages of its lifetime; half-life, final third, final quarter, and final fifth. Each classifier predicts when a module has reached a specific quantile of its life, based on the segmented measurements input data. Quantiles were used as thresholds because of the large variability in the lifespan of different modules, which ranges from 90 minutes to 1400 minutes (Table 9). All classifiers were trained using the Adam optimizer [66], fifty epochs, mini-batch size of twenty-seven, and one LSTM cell with five hidden units. These parameters were set via hyper parameter optimization, applied to the training data.

Therefore, the training step resulted in four trained LSTMs, predicting whether a module has reached the last half, third, quarter, and fifth of its life. Application of each LSTM to the segmented validation dataset produced classification scores between zero and one for each segment, classifying whether a given segment is derived from a time point in the last half, third, quarter, and fifth of the module's lifetime. Since these are binary classification tasks, we examined the receiver operating characteristic (ROC) curves, and the precision-recall curves (Fig.13) for the four validation modules, for each of the four LSTMs trained. We found that in all cases, the quantile-based performances were robust, providing a solid foundation for using these scores to infer the precise RUL at different time points.

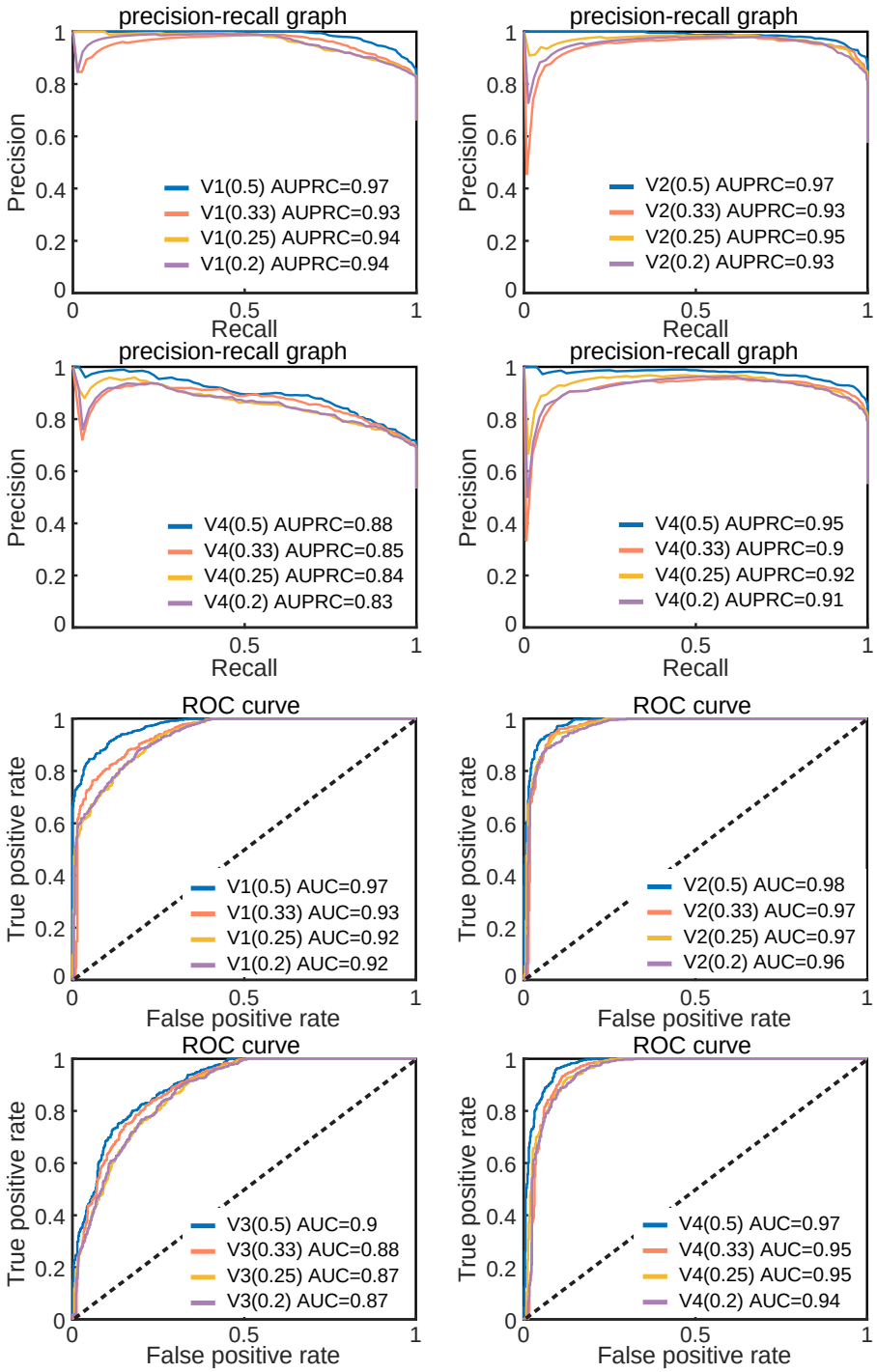

Figure 13 Classifier performance evaluation. Receiver operating characteristic (ROC) curves and the precision-recall curves showing the performances of the four classifiers.

LSTMs trained to predict the last half (0.5), third (0.3), a quarter (0.25), and fifth (0.2), when applied to the four validation modules (V1-V4 validation modules 1 through 4).

RUL prediction

The training process yielded four classifiers, where each classifier assigns a score to every segment of data when it is applied to the segmented input measurements. The score ranges from zero to one, where a higher score denotes a higher likelihood that a segment of data has passed a threshold (last half, third, quarter, and fifth). To use these scores and infer the precise RUL at any given time point, we designed an ensemble technique that combines the scores from all four classifiers. This works through four steps; First, the LSTM that was trained to predict whether a module has reached the last half of its life is applied to each segment of data from a module (ordered from first to last). When the scores of this LSTM for a segment have passed the threshold, we start predicting the RUL and assign that point with the time that has already passed (because the LSTM predicted that half of the time has passed). Second, after the module is predicted to be in the last half of its life, we use the scores produced by the LSTM that was trained to predict when a module has reached the last third of its life. In the time point when these scores pass the threshold, we assign that time point half of the time that has already passed. Similarly, after the first two LSTM thresholds were reached, we use the scores produced by the third LSTM, denoting whether a module has reached the last quarter of its life, and when the scores pass this threshold, we assign that time point one-third of the time that has already passed. Finally, after all, three LSTM thresholds were reached, when the scores produced by the fourth LSTM, predicting when a module has reached the last fifth of its life, pass the threshold, we assign that time point one-fourth of the time that has already passed.

To handle outliers and noise in the prediction, all scores were flattened over a window of five consecutive segments before being used for prediction. In every time point that does not correspond to a particular time point when a threshold was reached, the predicted time was defined as the time predicted in the last time point minus the time that has passed from the last time point to the current time point. All four steps are defined with the same threshold of 0.5, to avoid overfitting the training data with the ensemble technique. By definition, the RUL represents a monotonically decreasing time series. Therefore, we do not allow the RUL of any time point t to be greater than that of the previous time point $t-1$. To that end, the predicted time is always set to a minimum of the predictions from the four LSTMs. The complete framework is illustrated in Fig.14.

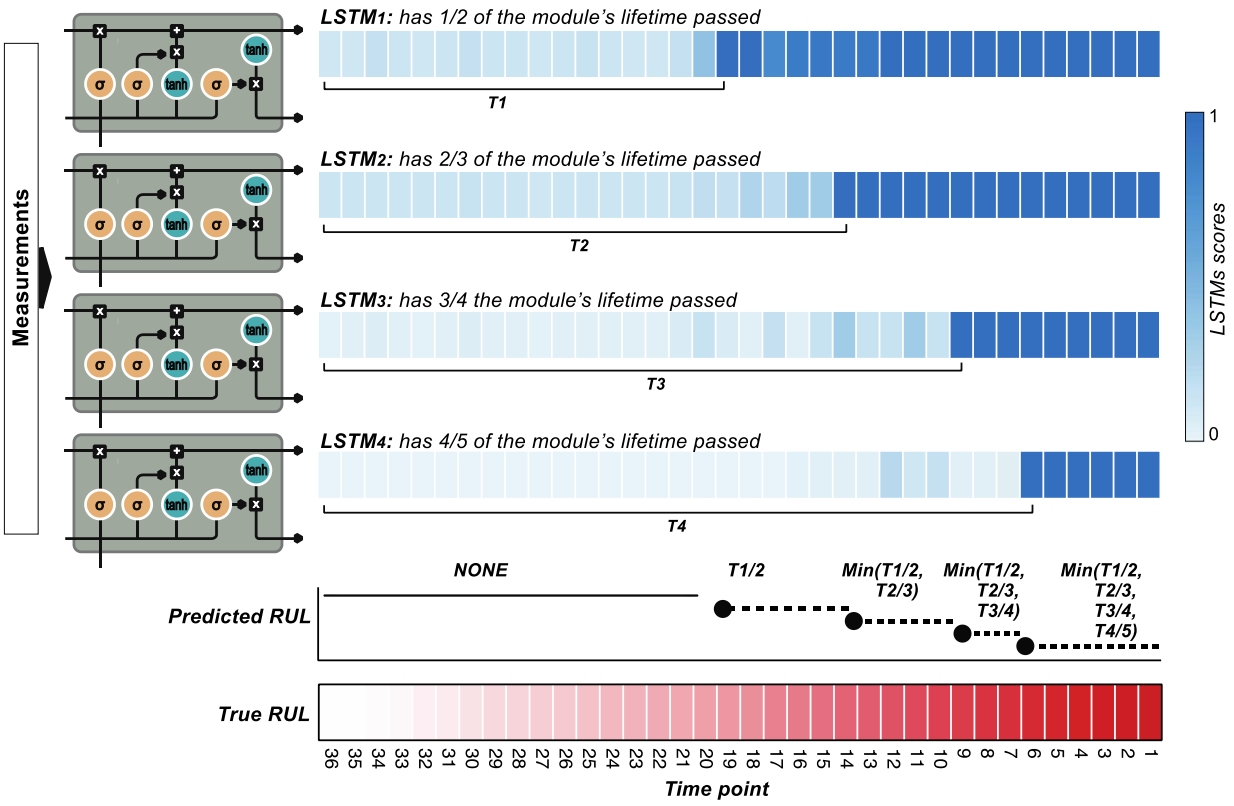

Figure 14 Pipeline illustration. The module's measurements are given as input to the four LSTMs, each is trained to predict when a module has reached a certain quantile in its life (last half, third, quarter, and fifths).

When applied to time points of data from the validation and test, each of the four LSTMs yields a score between zero and one, denoting the likelihood that this module has reached a certain quantile in its lifetime. These scores are used to predict the precise RUL by casting the time passed until each quantile has reached (black circles). In intermediate time points (dashed lines) the RUL is set by subtracting the time passed from the previous time point, from the predicted RUL of the previous time point.

Prediction performances

After the training of the four quantile-based LSTMs using the training dataset, we applied the complete prediction pipeline to the four modules in the validation dataset. Encouragingly, we found that the error rates were consistently low, and almost monotonically decreasing when predictions were made later in the life of each module (Fig.15A). In all but one of the modules in the validation dataset, the prediction of the half-life time point had less than a 20% error rate. In all modules in the validation dataset, the prediction in the last tenth of the module's life had less than 10% error. In particular, the predicted RUL in modules 2 and 4 in the validation were consistently accurate throughout their lifetime, as demonstrated in Figure 15B, C.

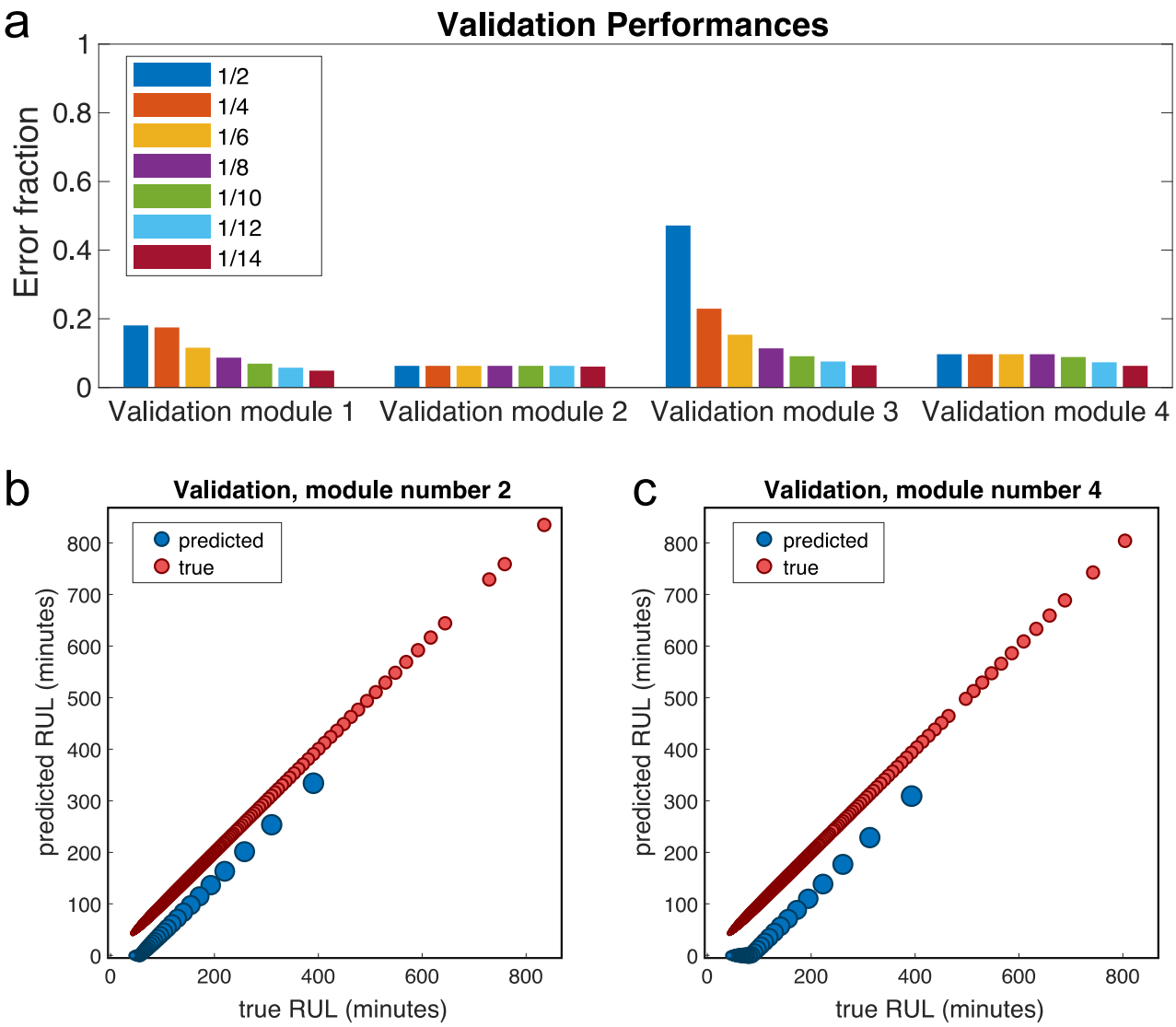

Figure 15 Validation performance. (a) Bar plots showing the error fraction of the four validation modules, when evaluating in different time points in the life of the module (half-life, last 1/4, 1/6, 1/8, 1/10, 1/12, and 1/14).

(b) and (c) demonstrate the true RUL (x-axes, red) vs the predicted RUL (y-axes, blue) for two modules in the validation set (validation modules 2 and 4, respectively). The prediction begins around the half-life of each module.

After confirming that the performance of the integrated quantile-based LSTM models on the validation set was sufficiently high and robust (Fig.15), we turned to test the performance of the approach when applied to the test set, which was left unseen through the entitled training process. We find that the prediction performances on the test set were comparably high when evaluated on time points from the last quarter of the module's life. For all five modules in the test set, the error rate was less than 25% in the last quarter of the modules life, and for four out of five modules it was less than 20% in any subsequent evaluation points. Four of the five modules in the test set also show a consistent and substantial decrease in the error rate when nearing the end of the life of these modules eventually having less than a 10% error rate (Fig.16). These results are particularly encouraging given the small size of the dataset in availability, and the difficulty of the classification problem, where previous work (with different datasets) failed when testing on a new and unseen dataset, even when achieving high performance in the training and validation [44]. The consistently low error rates when applied to the validation and the left-out test set demonstrate the robustness of this method even when trained to predict RUL of modules with high variation in lifespan (90-1400 minutes). It is possible that by converting the classification task to a globally defined problem that does not depend on the distribution of samples (recognizing quantiles in the life of modules), this method allows prediction from scratch, without the need to retrain the models and fit to the new distribution of samples.

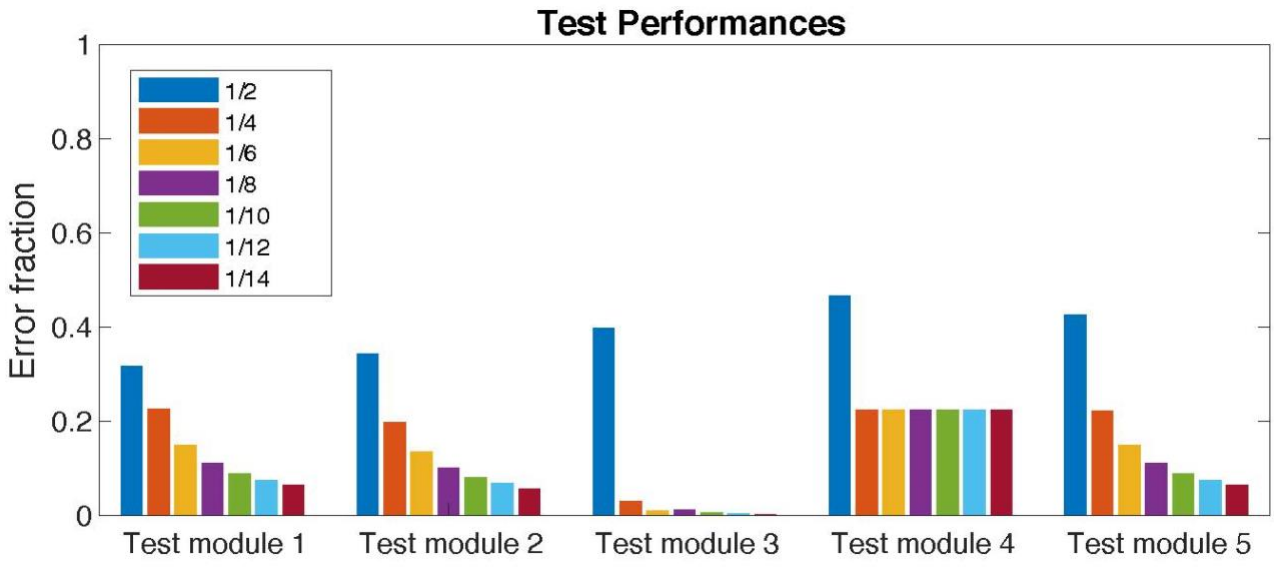

Figure 16 test performance. (a) Bar plots showing the error fraction of the five test modules, when evaluating in different time points in the life of the module (half-life, last 1/4, 1/6, 1/8, 1/10, 1/12, and 1/14).

Comparison to previous work

The dataset that we used throughout this work was obtained and originally used by Celaya et al. [42]. In this study, Celaya et al. utilized only a subset of the modules (5 modules overall), which are characterized by a lifespan range (as it is defined by them) between 150 and 240 minutes. In their work, they left one module out of training as a test (the module numbered 23). Therefore, to allow comparison with the study of Celaya et al, we ensured that module number 23 was left out to the test set in our study as well (Table 9). In their study, Celaya et al. defined the end of the life of each module not by the failure time and the last point measured, but by the time when the module had reached a certain delta in the Rds(ON) measurement (particularly, 0.05 ohm). In contrast, in this study, we aim to predict the precise end of life of each module. To compare our results with the results of Celaya et al. we examined the same interval relative to the end of the life of module 23 (the last 88 minutes). In their study, Celaya et al. compared three methods for RUL prediction including two model based, Extended Kalman Filter (EKF) and Particle Filter

(PF), and one data driven method, Gaussian Process Regression (GPR). PF showed the best performance on the test set module out of the three examined. We compared the prediction from the quantile-based LSTM ensemble on the same test module to the performance of the three methods as described by Celaya et al. Strikingly, we find that the quantile-based LSTM predictor achieved better performances compared to all three techniques, and specifically when reaching the end of the module's life (Fig.17, Table 10). This is especially notable because, in contrast to the methods applied by Celaya et al., the quantile-based LSTM predictor was trained on a set of modules with a wide range of lifespan (Table 9), thus addressing a more general classification task. These results further support the power of this method in accurately predicting the RUL even when trained and evaluated on small datasets of modules with a wide range of time to failure.

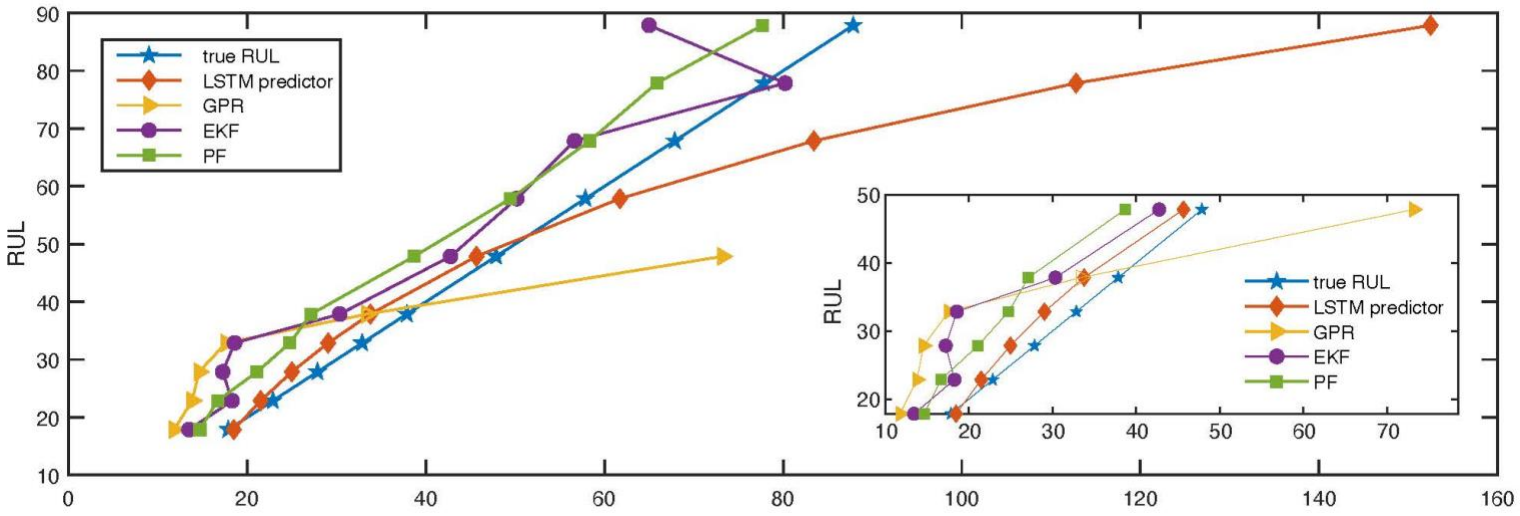

Figure 17 comparison to previous work. RUL prediction performance assessment for module number 26, for GPR, EKF and PF as described for Celaya et al.[22], and for the quantile LSTM predictor.

RUL	Quantile-LSTM	GPR	EKF	PF
88	152.5069228	NAN	64.98	77.65
78	112.8094278	NAN	80.22	65.85
68	83.44517583	NAN	56.64	58.33
48	61.72442772	NAN	50.15	49.47
48	45.65758223	73.2	42.75	38.68
38	33.77293062	33.4	30.35	27.14
33	29.04669114	17.6	18.57	24.76
28	24.98184939	14.6	17.24	21.09
23	21.48584828	13.8	18.28	16.66
18	18.47908332	11.8	13.46	14.68

Table 10 RUL prediction results for GPR, EKF, PF, and the quantile-LSTM for the last 88 minutes in the lifetime of module number 23. RUL prediction error is between parentheses.

Project one code overview

First function

RUN_TRAIN_VALIDATION.m: this script will take the raw data and will give out a structure that contains the validation and training modules: index, scores, labels, and RUL.

Input: raw data

Output: RES (struct)

This function has five steps:

First step

Step1_process_raw_data:

Input: raw data

Output: Data (struct)

This function goes through each module and all of its parts and uses **process_raw_module**, which will synchronize the steady-state and in situ measurements. The output will be a table for each module that contain the following features:

1. Rds(ON) upper limit
2. Rds(ON) lower limit
3. In situ measurement time
4. Steady state time
5. Supply voltage
6. Package temperature
7. Drain to source voltage
8. Drain current
9. Flange temperature

Second step

step2_add_rul

Input: data (struct)

Output: data26

This step takes the 42 module data structure from the previous step filter the modules that have issues and adds the RUL as the 10th feature.

Third step

step3_convert_data_for_lstm

Input: Data (struct from the previous step), t1 and t2 represent the range of modules we want to use for training (and validation), splt is what quantile to mark as 1 (percentile of RUL).

Output: Samp is the training (and validation) dataset (each point is 100 timepoints in a module) Samp.S2 is the training, and Samp.l is the labels. moduleN: the module index corresponding to each training point in Samp (from t1 to t2) and RUL: the corresponding RUL.

This function takes the modules we selected and breaks each module to 100 timepoints with 50 timepoints overlap. It will remove the steady-state and in situ times and put the RUL in a separate section and the label in the label struct.

In the main script we can select the quantiles we wish to train for in line 12 under quants[].

Forth step

TrainLSTM1

Input: Samp from the previous step, number of features, training modules, and 0 (not save, saves later in script).

Output: TNET, the neural network.

This step will use the training data to train a network and will save later in the script each network (for every quantile).

The function currently uses 5 hidden units and 50 Epochs.

Fifth step

testLSTM1

Input: the TNET from the previous step, the training data, and its labels.

Output: YPred, scores, L1 and L2

The output is the scores and labels of the data.

Each step fills more data to the RES structure that contains the quantiles, module index, training and validation module numbers, networks, labels, scores, and RUL.

Project 1 pseudocode

Begin function step1_process_raw_data

For m in modules

    For the length of each module steady state time points

        Convert and insert time point from epoch to date and time

        Insert supply voltage

        Insert package temp

        Insert drain source voltage

        Insert drain current

        Insert flange temp

    End

For the length of each module transient time points

Convert and insert time point from epoch to date and time

Calculate Rdson at each transient time point

Insert the 0.25 and 0.75 quantiles of Rdson

End

For the length of each module transient time points

Match each transient time point with the closest steady state time point

Insert supply voltage

Insert package temp

Insert drain source voltage

Insert drain current

Insert flange temp

End

End

End function

```
Begin function step2_add_rul
For  m in modules
        Remove time gaps
        Calculate and insert RUL
End
End


Begin function step3_convert_data_for_lstm
For  m in modules
        Remove times
        Add labels
End


For tm in training and validation modules
        For k from 1 until module length minus 100 with 50 points steps
                Insert labels to label structure
Remove the label from the data
                Insert data to structure
```

End

End

End

Begin function trainLSTM1

Insert L2 with converted categorical cell to a matrix of labels

Set max epoch to 50

Set the mini-batch size to 27

Set input size to size of training points

Set the number of hidden units to 5

Set the numclass to the length of L2

Set layers as

Sequence input layer (input size)

lstm layer (number of hidden units, outputmode, last)

Fully connected layer (numclass)

Soft max layer

Classification layer

Set options as

Training options (Adam, execution environment, cpu, max epoch, mini-batch size, gradient threshold, 1, verbose, 0, plots, training progress)

Tnet is a train network with data L2 layers and options

End

Begin function testLSTM1

Set L1 as the conversion of the labels

Calculate the prediction and scores using the trained network and the data

End

Project two: Quantile-based LSTM Remaining Useful Life prediction of filtration system.

Dataset 2

This dataset was published on the PHME2020 fifth European conference of the prognostics and health management society 2020, as part of the PHME2020 data challenge. The dataset was collected using the following experiment:

An experimental rig was constructed to demonstrate a clogging failure in a filter, it was designed using the following components: Pump, liquid tanks, tank stirrer, pressure and flow rate sensors, pulsation dampener, filter, and data acquisition system.

The experiment rig contains a circuit with a pump flowing a liquid through a filter from one tank to another. The circuit is monitoring the flow rate and the liquid pressure before and after the filter, using sensors. The fluid injected in the system is a suspension composed of Polyetheretherketone (PEEK) particles and water with different concentrations. The circuit includes a dampener to eliminate possible pulsations in the flow. Figure 18 depicts the employed experimental rig [67].

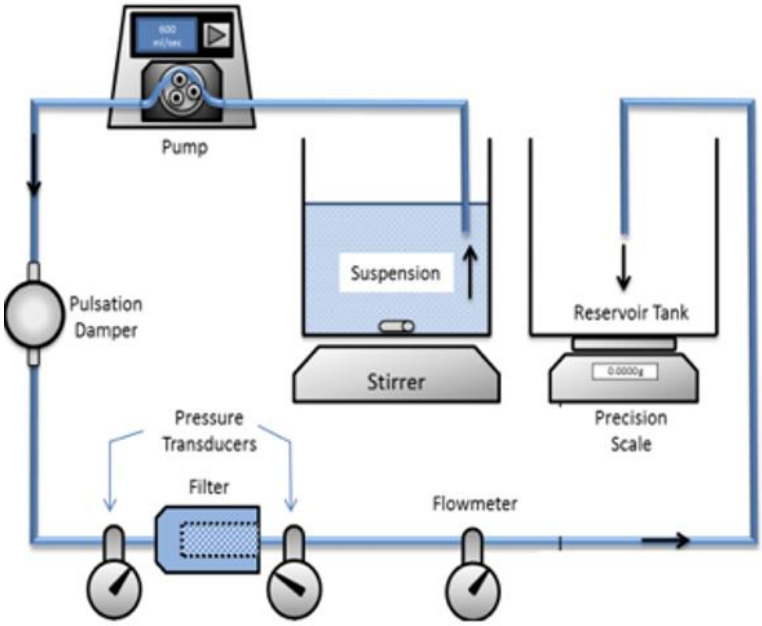

Figure 18 System of the experimental rig [67].

The main components of this experimental rig are the following:

Pump: A peristaltic pump has been used since the system will involve contaminants in the fluid, as its mechanism is more tolerant to particles in the liquid. The model of the peristaltic pump is

Masterflex® SN-77921-70 (Drive: 07523-80, Two Heads: 77200-62, Tubing: L/S© 24) it was installed in the system to maintain the flow of the prepared suspension. The pump is providing a flow rate ranging from 0.28 to 1700 ml/min (i.e. from 0.1 to 600 RPM).

Dampener: To prevent the system from unwanted tube expansion due to pressure build-up, which affects the actual pressure build-up generated from filter clogging they used rigid tubing. They installed a Masterflex® pulse dampener, and to eliminate any pulsation in flow, it was installed on the downstream side of the pump. The pump side is covered with a flexible Tygon® LFL pump tubing and the majority of the system is furnished with rigid polypropylene tubing.

Particles: The suspension is composed of water and Polyetheretherketone (PEEK) particles. PEEK particles have a significantly low water absorption level (0.1% / 24 hours, ASTM D570) and a density of (1.3g/cm3) which is close to that of room temperature water. The low water absorption level will prevent particles expansion when they mix with water. Also, the closer density with water allows particles to suspend longer in water.

Flow Rate Sensor: To keep track of the flow rate in the system A GMAG100 series electromagnetic flowmeter is installed which has a measurement range from 3 to 25,000 milliliters per minute.

Pressure Sensors: To capture the pressure drop (ΔP) across the filter, which is considered as the main indicator of clogging, upstream and downstream Ashcroft® G2 pressure transducers were installed which measurement range from 0 to 100 PSI.

Filter: the filter has a pore mesh size of 125µm as shown in Figure 19.

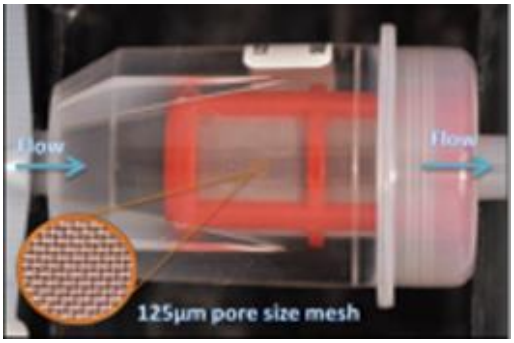

Figure 19 Filter under study

Several experiments have been run with this experimental rig, with suspensions of different concentrations and particle sizes. For each particle size, we perform four experiments with the concentration reported in Table 11.

	Profile Number			
	1	2	3	4
Water (g) in the suspension tank	7968	7497	7079	6704
Particle (g)	32	32	32	32
Solid ratio	0.004 (0.40%)	0.00425 (0.43%)	0.0045 (0.45%)	0.00475 (0.48%)

Table 11 Suspension profile details

We add particles of three possible sizes to create the suspensions, we added small, medium, and large. Details about the particle size are reported in Table 12.

	Particle Size (µm)
Small	45-53
Medium	53-63
Large	63-75

Table 12 Particle size data

The collected dataset of the experiment contains the liquid flow rate, and the pressure before and after the filter (upstream and downstream), it was collected at a 10Hz rate. Any pressure drop higher than 20 psi was identify as failure due to filter clog, the Pressure Drop was computed by deducting the upstream Pressure from the downstream pressure.

The dataset was divided into three sets training, validation, and test.

The training dataset contained 24 runs to failure experiments of the filter, the dataset contains experiments with small or large particles sizes and different particle concentrations. Four experiments were done for each particle size and concentration combination. The following table describes the sets.

Profile Number	Particle Size (µm)	Solid Ratio (%)	Sample Size
1	45-53	0.4	4
2	Small	0.425	4
3		0.45	4

1 2 3	63-75 Large	0.4 0.425 0.45	4 4 4

Table 13 Training set – 24 samples

The validation dataset contained 8 runs to failure experiments of the filter, the dataset contains experiments with small or large particles sizes and constant particle concentrations. Four experiments were done for each particle size. The following table describes the sets.

Profile Number	Particle Size (µm)	Solid Ratio (%)	Sample Size
1	45-53 Small	0.475	4
2	63-75 Large	0.475	4

Table 14 Validation set – 8 samples

The test dataset contained 16 runs to failure experiments of the filter, the dataset contains experiments with only medium particles sizes and different particle concentrations. Four experiments were done for each particle concentration. The following table describes the sets.

Profile Number	Particle Size (µm)	Solid Ratio (%)	Sample Size
1	53-63 Medium	0.4	4
2	53-63 Medium	0.425	4
3	53-63 Medium	0.45	4
4	53-63 Medium	0.475	4

Table 15 Test set – 16 samples

Procedure

Data representation

The dataset was downloaded in .cvs format and was imported into MATLAB, after it was imported it was converted from a table format to an array, this array included the following features:

- Time in seconds with 0.1-second constant intervals.
- The flow rate is in ml/m.
- Upstream pressure in psi.
- Downstream pressure in psi.
- Particle size in microns and solid ratio percentage were given in separate .xlsx files and were manually added.

- Pressure drop was also calculated by deducting the upstream Pressure from the downstream pressure.

The dataset provider defined a clogged filter as a filter with 20 (psi) pressure drop; therefore, we only used the data up to that point.

The lifespan of the filter varies in a wide range from 277.5-172.6 seconds. The list of the filters life span can be seen in the following table:

Filter number	Set assignment	Lifetime seconds
1	Training	273.4
2	Training	277.5
3	Training	273.5
4	Training	256.1
5	Training	265.5
6	Training	266.2
7	Training	234.8
8	Training	238.1
9	Training	235.2
10	Training	212.2
11	Training	212.9
12	Training	210.8
13	Training	213.4
14	Training	206
15	Training	203.5

16	Training	192.8
17	Training	194.6
18	Training	193.1
19	Training	178
20	Training	175.6
21	Training	175.2
22	Training	172.6
23	Training	176
24	Training	173.2
25	Validation	273.1
26	Validation	261.1
27	Validation	234.2
28	Validation	210.9
29	Validation	204.7
30	Validation	198.8
31	Validation	182.1
32	Validation	173.2
33	Test	221.5
34	Test	223.7
35	Test	224.7
36	Test	226.1
37	Test	213
38	Test	215.5

39	Test	215.3
40	Test	210.8
41	Test	205.7
42	Test	206.1
43	Test	206.8
44	Test	208.9
45	Test	196.8
46	Test	198.9
47	Test	200.4
48	Test	198.3

Table 16 filters set assignments and lifespans

After creating the feature table, we add the labels. Then the data was converted to data points that each one contains five-time steps (half a second), and the label of each data point was the roundup of the average of the labels. The validation data was then put aside and the training dataset was used to train the LSTM network, in this work five classifiers were created; the last 50%, 40%, 30%, 20%, and last 10%, after training the classifiers it was used to evaluate the training and validation datasets and eventually the test dataset. This procedure was done five times for each of the classifiers. After gathering all the scores given by the classifiers an RUL estimator was designed to convert the scores to an RUL estimation.

RUL prediction

After the training process, we have five classifiers, each of these classifiers produces a score between zero and one for each data point when applying the network on the data, where a higher score represents a higher probability of not passing the threshold, of that classifier (10%-50%).

After combining all the scores we achieve a matrix in the size of Mi*5 where Mi is the length of each filter data point, each data point is equivalent to 0.5 seconds. To convert the matrix to an actual RUL we developed a similar ensemble technique to the one we used in the first dataset. In this ensemble method we have five steps; First, we use the first classifier we trained, to predict the point we reach 50%, we scan the data points scores, until the score of the first classifier reaches below our set threshold of 0.8 when we reach that point we set our initial RUL prediction, as the same time that has passed, as we predict we reached half of the filter life. Only after reaching the first threshold, we start looking at the second classifier scores, when that score reaches below 0.8, we multiply the time that has passed by 2/3, since we believe 60% of the time has passed and we predict that 40% has left ($60\% * \frac{2}{3} = 40\%$). After reaching the second classifier we move to the next classifier, we use the same threshold of 0.8, and when reaching that point we multiply the time that passed by 3/7, as we predict that 70% of the time has passed and 30% left. The fourth and fifth step uses the same threshold of 0.8 and multiplies the times by 1/4 and 1/9 to complete all the prediction of the RUL.

Prediction performances

In the following section, the prediction results of each filter RUL will be presented.

In the small graphs, the red line represents the prediction, the first change in the red line shows the start of the prediction, while each change in the line afterword represents the next classifier prediction, and the blue line represents the true RUL. The X-axis will show the time, each time point is worth 0.5 seconds of data, and the Y-axis is the RUL in seconds. The final graph at each section will show the full-length RUL prediction given by each classifier, every classifier will be shown in a different color and a dark blue line will show the true RUL, in that graph the X-axis represents the filter number, and the Y-axis shows the RUL in seconds.

Training and validation datasets

The following figures show the prediction results on the training and validation datasets.

The first change in the red line shows the start of the prediction, and we can see that in most cases we underpredict the true RUL, which can prevent an unforeseen failure of the system, by under predicting the RUL.

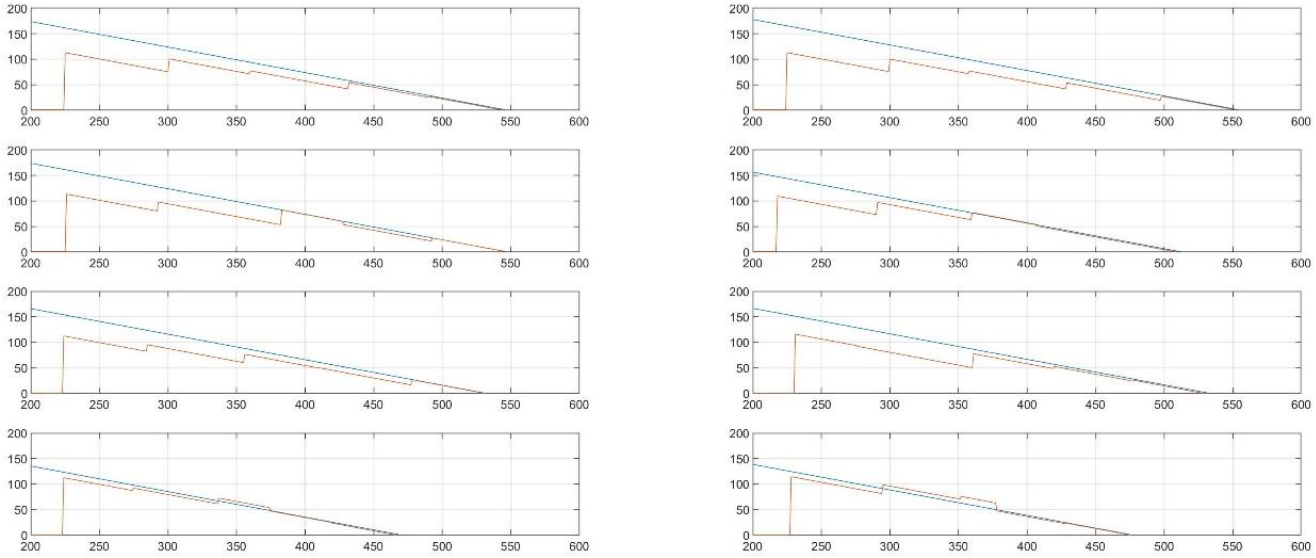

Figure 20 Filters 1-8 from the training dataset

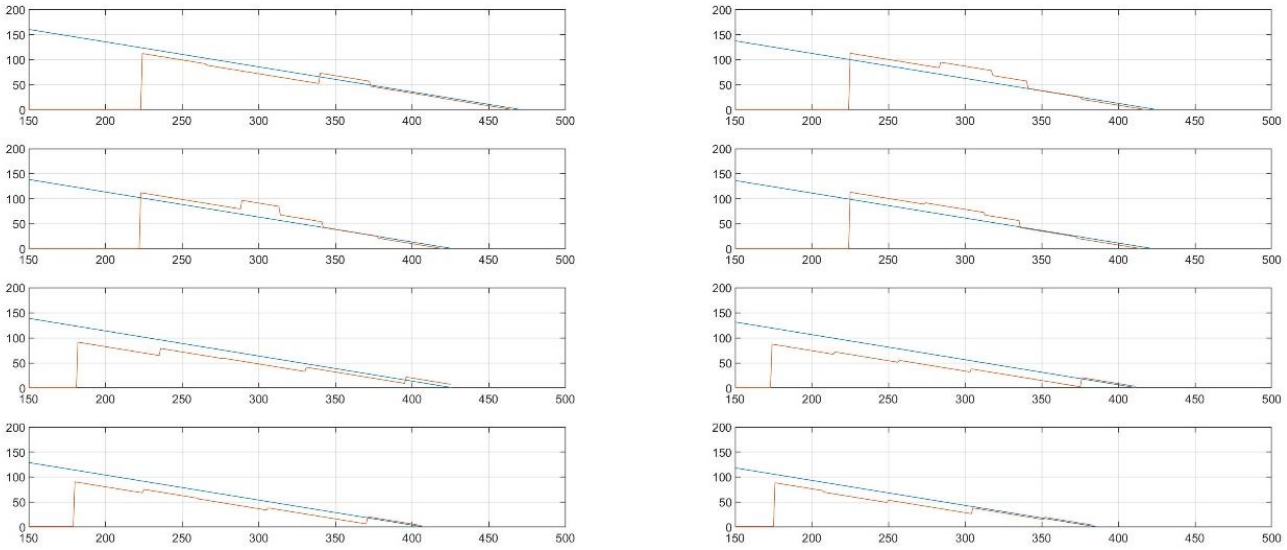

Figure 21 Filters 9-16 from the training dataset

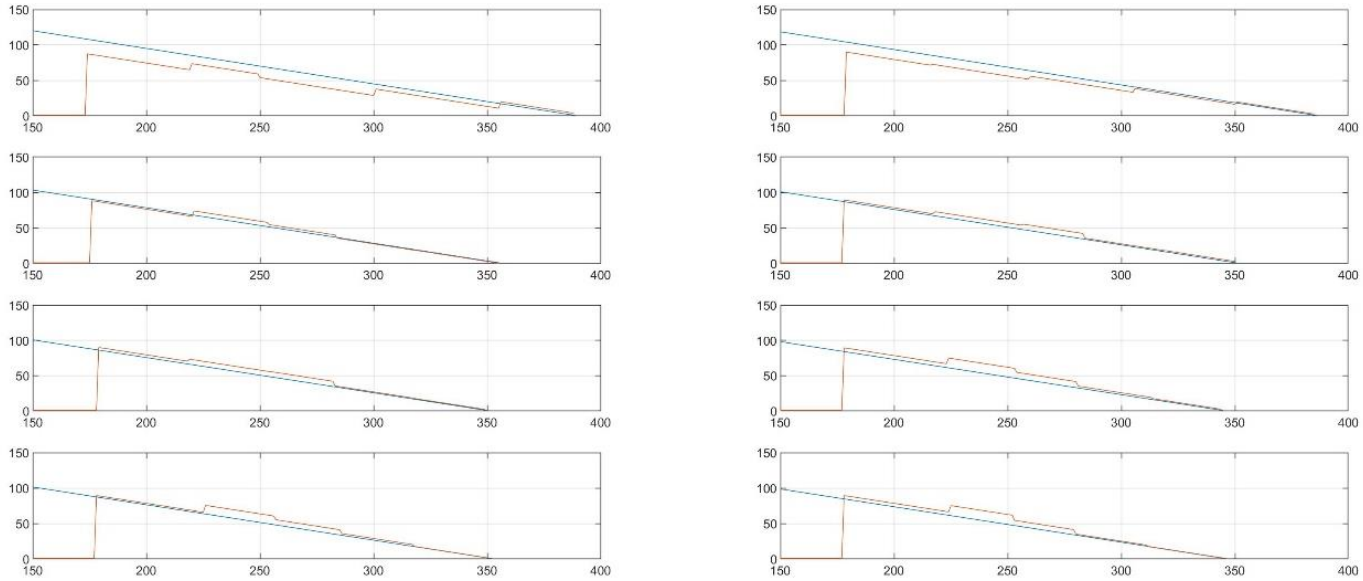

Figure 22 Filters 17-24 from the training dataset

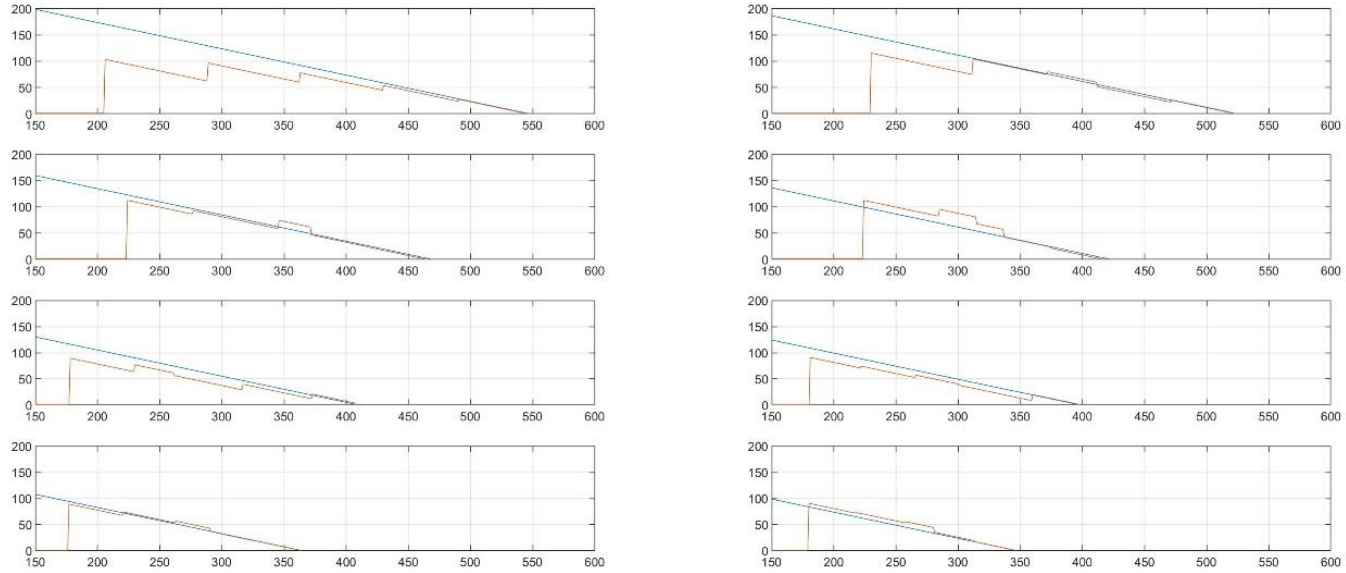

Figure 23 Filters 1-8 from the validation dataset

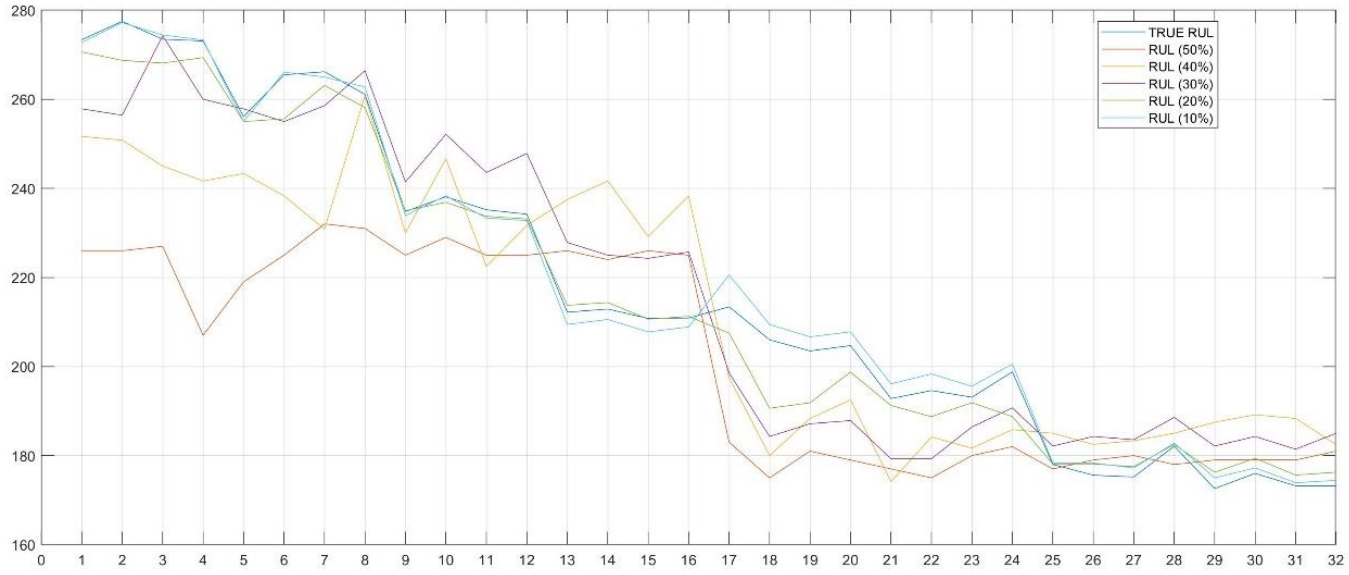

Figure 24 Training + Validation

We can also see that the prediction improves, as we get closer to the end of life of the filter, the improved prediction by the later classifiers ensure higher accuracy of the RUL as we approach the end of life. The graphs also show that in the case of over predicting the RUL, the next classifier will most likely mitigate the error. These performances looked promising and after seeing that the overall accuracy levels remain the same, the mean absolute error (MEA) and the percentage error were calculated for each classifier as can be seen in table 17.

$$MAE = \frac{\sum_{i=1}^{32} |y_i - x_i|}{32}$$

In addition, the percentage error was calculated using the following formula:

$$M = \frac{100}{32} \sum_{i=1}^{32} \left| \frac{x_i - y_i}{x_i} \right|$$

Where:

y_i - RUL prediction.

x_i - RUL true value.

Classifier	MAE (seconds)	M (%)
50%	10.27	8
40%	9.61	2.74
30%	7.73	0.38
20%	3.01	1.22
10%	1.71	0.3

Table 17 MAE and M per classifier for training and validation datasets combined

The results show that the classifiers are accurate in predicting the RUL. It also shows the improvement in predicting the RUL when getting closer to the end of life of the filter. After reviewing the results of the training and validation, we proceed to apply the same classifiers and ensemble procedure on the test dataset.

Test dataset

The following figures show the prediction results on the test dataset.

The first change in the red line shows the start of the prediction, and we can see that similar to the training and validation dataset, in most cases we underpredict the true RUL at first.

Unlike the training and validation datasets, we over predict with the second classifier, in most cases, but the following classifiers adjust the prediction and improve the overall accuracy of the prediction.

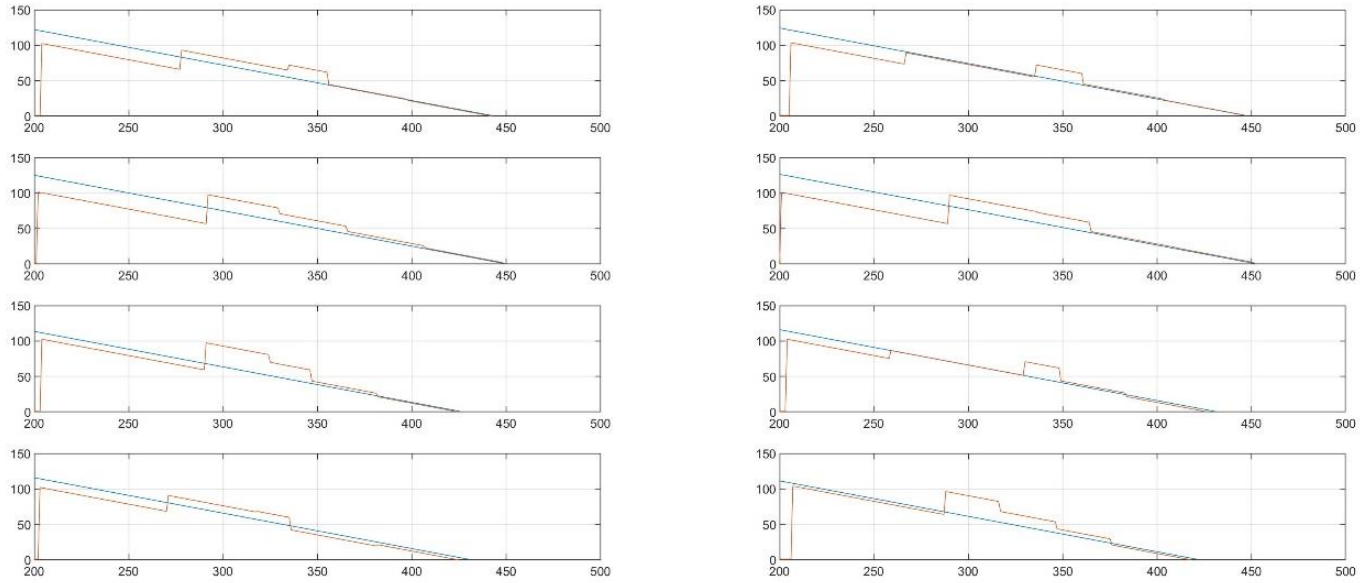

Figure 25 Filters 1-8 from the test dataset

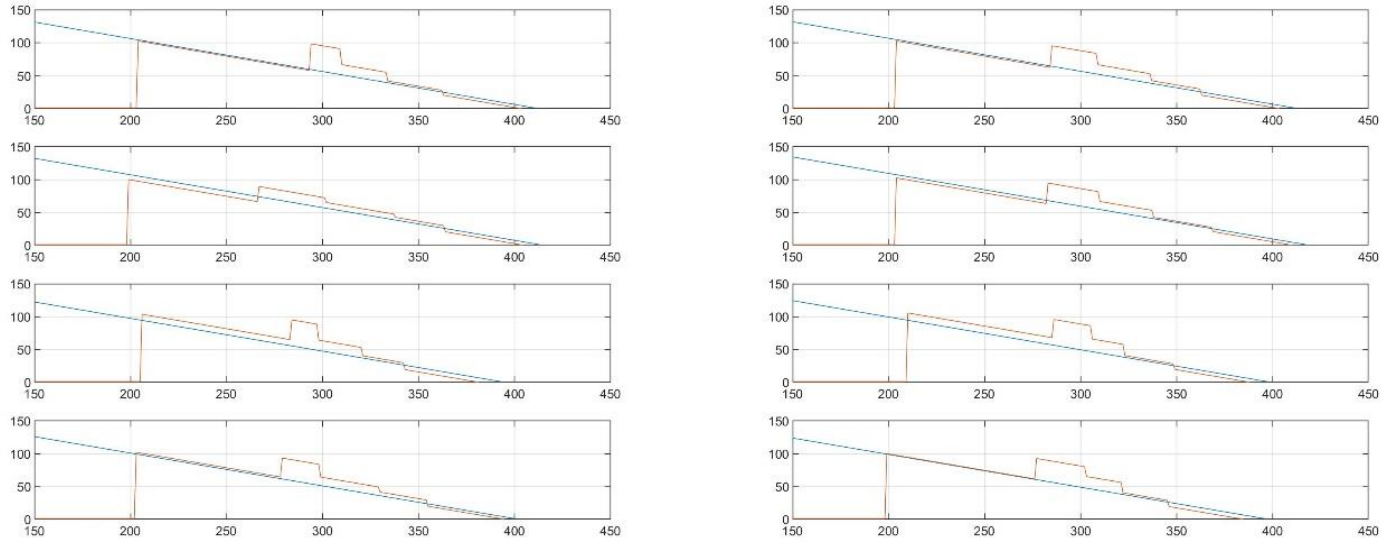

Figure 26 Filters 9-16 from the test dataset

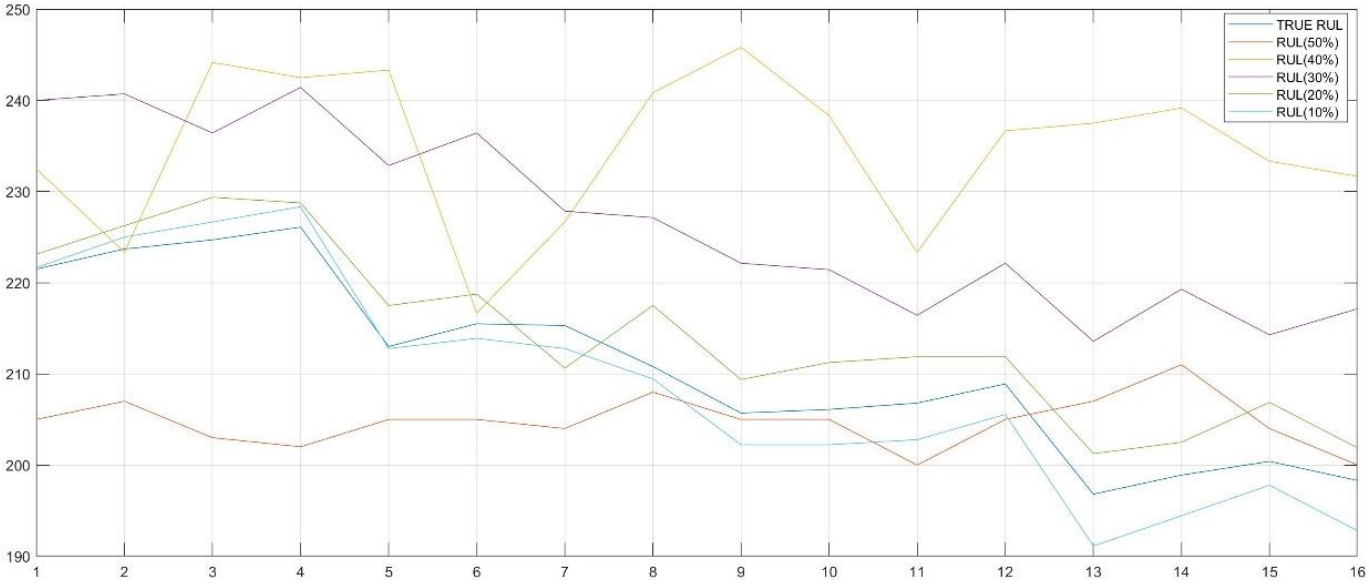

Figure 27 Test Dataset

We can also see that, similar to the training and validation, the prediction improves as we get closer to the end of life of the filter.

Different from the training and validation classifier, we can see that the first classifier, 50% of life, is very accurate on the test dataset, MAE is 4.74 seconds. We see in most cases the second classifier, last 40% of life is less accurate, but, as in the training and validation, we see that the following classifiers will most likely mitigate the error. We can see in the following table the accuracy of the different classifiers on the test dataset.

Classifier	MAE (seconds)	M (%)
50%	4.74	3.06
40%	14.4	11.13
30%	11.2	7.6
20%	3.28	1.65
10%	2.5	0.93

Table 18 MAE and M per classifier for the test dataset

Conclusions

Recurrent neural networks, and particularly gated recurrent neural networks such as LSTM, present the state of the art performance in learning time-series data and for time-series classification tasks. However, in the field of prognostics, the shortage of reliable data of measurement of operational features and the respective RUL at each time point, presents a considerable challenge, hindering the straightforward application of LSTM to predict the RUL given a set of measurements (Fig.12).

In this work, I develop an approach that can overcome these limitations by representing the classification problem (RUL prediction given a set of measurements) as a set of simpler classification problems, for all of which we can achieve a high level of performance using LSTM recurrent neural networks. By converting the classification problem of "what is the module RUL at time point t" that could not be answered by the LSTM, to a set of questions: "did the module passed half of its life", "did the module passed 2/3 of its life", "did the module passed ¾ of its life" etc., which are successfully answered by the LSTM, we can eventually answer the original question "what is the RUL".

There are 3 novel components to this work: (1) the representation of the data: the features that are used and represented to maximize the performance eventually and increase the number of time points while allowing the classifier to learn. (2) The proof of concept that while the classification task of "what is the RUL" given a set of measurements is difficult cannot be solved with confidence, there is a set of classification tasks that can be solved with confidence given a set of measurement, such as "did the module passed half of its life", and are solved with high levels of accuracy by LSTM recurrent neural networks. (3) Finally, I developed an ensemble

pipeline for the integration of different quantile-based LSTM classifiers, which allows the prediction of the precise RUL at different points in the life of the module.

Overall, this work facilitates utilization of LSTM RNNs, the state of the art in time series prediction, for accurate prediction of the RUL for MOSFET modules and filtration system with a wide lifespan range, when trained on a small dataset, thus allowing a substantial improvement for data-driven RUL prediction. Future work is warranted to further study the application of this technique to different classification problems in reliability that involve time-series data.

Contributions

- Developed a new method that outperform standard RNN LSTM in predicting RUL.
- Established a robust and more accurate method to determine RUL from run to failure dataset, when given a small dataset, using RNN LSTM.
- A new method of data representation, data blocks, increases the number of data points and their significance.
- Using quantiles as thresholds to make the method more robust.
- Ability to use a wide range of lifespan data and utilize it to predict RUL.
- A method that can be applied to any run to failure datasets, not just electronics.

Future work

- Identifying the type of failure in addition to the RUL.
- Adding confident interval and probability function to the RUL prediction.
- Using real-time maintenance data to determine the maintenance contribution and its increase to the RUL.

Citations

[1] A. K. Schömig and O. Rose, "On the suitability of the Weibull distribution for the approximation of machine failures," in IIE Annual Conference. Proceedings, 2003, p. 1: Institute of Industrial and Systems Engineers (IISE).

[2] A. Heng, A. C. Tan, J. Mathew, N. Montgomery, D. Banjevic, and A. K. Jardine, "Intelligent condition-based prediction of machinery reliability," Mechanical Systems and Signal Processing, vol. 23, no. 5, pp. 1600-1614, 2009.

[3] D. Banjevic and A. Jardine, "Calculation of reliability function and remaining useful life for a Markov failure time process," IMA journal of management mathematics, vol. 17, no. 2, pp. 115-130, 2006.

[4] E. Bechhoefer, "A method for generalized prognostics of a component using Paris law," in Annual Forum Proceedings-American Helicopter Society, 2008, vol. 64, no. 2, p. 1460: AMERICAN HELICOPTER SOCIETY, INC.

[5] F. Zhao, Z. Tian, and Y. Zeng, "Uncertainty quantification in gear remaining useful life prediction through an integrated prognostics method," IEEE Transactions on Reliability, vol. 62, no. 1, pp. 146-159, 2013.

[6] M. Behzad, H. A. Arghan, A. R. Bastami, and M. J. Zuo, "Prognostics of rolling element bearings with the combination of paris law and reliability method," in 2017 Prognostics and System Health Management Conference (PHM-Harbin), 2017, pp. 1-6: IEEE.

[7] T. Wang, Z. Liu, M. Liao, and N. Mrad, "Life prediction for aircraft structure based on Bayesian inference: towards a digital twin ecosystem," in Annual Conference of the PHM Society, 2020, vol. 12, no. 1, pp. 8-8.

[8] W. K. Yu and T. A. Harris, "A new stress-based fatigue life model for ball bearings," Tribology transactions, vol. 44, no. 1, pp. 11-18, 2001.

[9] M. Gašperin, Đ. Juričić, P. Boškoski, and J. Vižintin, "Model-based prognostics of gear health using stochastic dynamical models," Mechanical Systems and Signal Processing, vol. 25, no. 2, pp. 537-548, 2011.

[10] J. R. Celaya, C. S. Kulkarni, G. Biswas, and K. Goebel, "Towards a model-based prognostics methodology for electrolytic capacitors: A case study based on electrical overstress accelerated aging," Int. J. Prognostics Health Manage., vol. 3, no. 2, p. 33, 2012.

[11] D. A. Tobon-Mejia, K. Medjaher, N. Zerhouni, and G. Tripot, "A data-driven failure prognostics method based on mixture of Gaussians hidden Markov models," IEEE Transactions on reliability, vol. 61, no. 2, pp. 491-503, 2012.

[12] H. Ocak, K. A. Loparo, and F. M. Discenzo, "Online tracking of bearing wear using wavelet packet decomposition and probabilistic modeling: A method for bearing prognostics," Journal of sound and vibration, vol. 302, no. 4-5, pp. 951-961, 2007.

[13] Y. Ao and G. Qiao, "Prognostics for drilling process with wavelet packet decomposition," The International Journal of Advanced Manufacturing Technology, vol. 50, no. 1-4, pp. 47-52, 2010.

[14] R. Huang, L. Xi, X. Li, C. R. Liu, H. Qiu, and J. Lee, "Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods," Mechanical Systems and Signal Processing, vol. 21, no. 1, pp. 193-207, 2007.

[15] P. Baraldi, M. Compare, S. Sauco, and E. Zio, "Ensemble neural network-based particle filtering for prognostics," Mechanical Systems and Signal Processing, vol. 41, no. 1-2, pp. 288-300, 2013.

[16] X. Li, 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, 2018.

[17] N. Khera and S. A. Khan, "Prognostics of aluminum electrolytic capacitors using artificial neural network approach," Microelectronics Reliability, vol. 81, pp. 328-336, 2018.

[18] H. Taghavifar and A. Mardani, "Applying a supervised ANN (artificial neural network) approach to the prognostication of driven wheel energy efficiency indices," Energy, vol. 68, pp. 651-657, 2014.

[19] S. Rebello, H. Yu, and L. Ma, "An integrated approach for system functional reliability assessment using Dynamic Bayesian Network and Hidden Markov Model," Reliability Engineering & System Safety, vol. 180, pp. 124-135, 2018.

[20] Q. Xiao, Y. Fang, Q. Liu, and S. Zhou, "Online machine health prognostics based on modified duration-dependent hidden semi-Markov model and high-order particle filtering," The International Journal of Advanced Manufacturing Technology, vol. 94, no. 1-4, pp. 1283-1297, 2018.

[21] J. Yu, "Adaptive hidden Markov model-based online learning framework for bearing faulty detection and performance degradation monitoring," Mechanical Systems and Signal Processing, vol. 83, pp. 149-162, 2017.

[22] F. Z. Feng, D. D. Zhu, P. C. Jiang, and H. Jiang, "GA-SVR based bearing condition degradation prediction," in Key engineering materials, 2009, vol. 413, pp. 431-437: Trans Tech Publ.

[23] H.-Z. Huang, H.-K. Wang, Y.-F. Li, L. Zhang, and Z. Liu, "Support vector machine based estimation of remaining useful life: current research status and future trends," Journal of Mechanical Science and Technology, vol. 29, no. 1, pp. 151-163, 2015.

[24] A. Nuhic, T. Terzimehic, T. Soczka-Guth, M. Buchholz, and K. Dietmayer, "Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods," Journal of power sources, vol. 239, pp. 680-688, 2013.

[25] H.-E. Kim, A. C. Tan, J. Mathew, E. Y. Kim, and B.-K. Choi, "Machine prognostics based on health state estimation using SVM," in Asset Condition, Information Systems and Decision Models: Springer, 2012, pp. 169-186.

[26] C. Cempel, H. Natke, and J. Yao, "Symptom reliability and hazard for systems condition monitoring," Mechanical Systems and Signal Processing, vol. 14, no. 3, pp. 495-505, 2000.

[27] C.-J. Du and D.-W. Sun, "Learning techniques used in computer vision for food quality evaluation: a review," Journal of food engineering, vol. 72, no. 1, pp. 39-55, 2006.

[28] A. Rastogi, R. Arora, and S. Sharma, "Leaf disease detection and grading using computer vision technology & fuzzy logic," in 2015 2nd international conference on signal processing and integrated networks (SPIN), 2015, pp. 500-505: IEEE.

[29] A. K. Paul, D. Das, and M. M. Kamal, "Bangla speech recognition system using LPC and ANN," in 2009 Seventh International Conference on Advances in Pattern Recognition, 2009, pp. 171-174: IEEE.

[30] V. V. Krishnan, A. Jayakumar, and A. P. Babu, "Speech recognition of isolated Malayalam words using wavelet features and artificial neural network," in 4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008), 2008, pp. 240-243: IEEE.

[31] W. Gevaert, G. Tsenov, and V. Mladenov, "Neural networks used for speech recognition," Journal of Automatic control, vol. 20, no. 1, pp. 1-7, 2010.

[32] R. Jafari-Marandi, S. Davarzani, M. S. Gharibdousti, and B. K. Smith, "An optimum ANN-based breast cancer diagnosis: Bridging gaps between ANN learning and decision-making goals," Applied Soft Computing, vol. 72, pp. 108-120, 2018.

[33] O. W. Samuel, G. M. Asogbon, A. K. Sangaiah, P. Fang, and G. Li, "An integrated decision support system based on ANN and Fuzzy_AHP for heart failure risk prediction," Expert Systems with Applications, vol. 68, pp. 163-172, 2017.

[34] S. Vijayarani, S. Dhayanand, and M. Phil, "Kidney disease prediction using SVM and ANN algorithms," International Journal of Computing and Business Research (IJCBR), vol. 6, no. 2, 2015.

[35] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," IEEE transactions on neural networks, vol. 5, no. 2, pp. 240-254, 1994.

[36] J. Zhang and K. Man, "Time series prediction using RNN in multi-dimension embedding phase space," in SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), 1998, vol. 2, pp. 1868-1873: IEEE.

[37] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," arXiv preprint arXiv:1704.02971, 2017.

[38] X. Cai, N. Zhang, G. K. Venayagamoorthy, and D. C. Wunsch II, "Time series prediction with recurrent neural networks trained by a hybrid PSO–EA algorithm," Neurocomputing, vol. 70, no. 13-15, pp. 2342-2353, 2007.

[39] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in Proceedings, 2015, vol. 89: Presses universitaires de Louvain.

[40] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," Transportation Research Part C: Emerging Technologies, vol. 54, pp. 187-197, 2015.

[41] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," European Journal of Operational Research, vol. 270, no. 2, pp. 654-669, 2018.

[42] J. Celaya, A. Saxena, S. Saha, and K. F. Goebel, "Prognostics of power MOSFETs under thermal stress accelerated aging using data-driven and model-based methodologies," 2011.

[43] L. Guo, N. Li, F. Jia, Y. Lei, and J. Lin, "A recurrent neural network based health indicator for remaining useful life prediction of bearings," Neurocomputing, vol. 240, pp. 98-109, 2017.

[44] K. Pugalenthi, H. Park, and N. Raghavan, "Prognosis of power MOSFET resistance degradation trend using artificial neural network approach," Microelectronics Reliability, vol. 100, p. 113467, 2019.

[45] Y. Zhang, R. Xiong, H. He, and M. G. Pecht, "Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries," IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp. 5695-5705, 2018.

[46] Z. Shi and A. Chehade, "A dual-LSTM framework combining change point detection and remaining useful life prediction," Reliability Engineering & System Safety, vol. 205, p. 107257, 2021.

[47] R. Guo, Y. Wang, H. Zhang, and G. Zhang, "Remaining useful life prediction for rolling bearings using EMD-RISI-LSTM," IEEE Transactions on Instrumentation and Measurement, vol. 70, pp. 1-12, 2021.

[48] K. Goebel, B. Saha, and A. Saxena, "A comparison of three data-driven techniques for prognostics," in 62nd meeting of the society for machinery failure prevention technology (mfpt), 2008, pp. 119-131.

[49] C. K. Williams and C. E. Rasmussen, Gaussian processes for machine learning (no. 3). MIT press Cambridge, MA, 2006.

[50] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černocký, "Strategies for training large scale neural network language models," in 2011 IEEE Workshop on Automatic Speech Recognition & Understanding, 2011, pp. 196-201: IEEE.

[51] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," IEEE Signal processing magazine, vol. 29, no. 6, pp. 82-97, 2012.

[52] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in 2013 IEEE international conference on acoustics, speech and signal processing, 2013, pp. 6645-6649: IEEE.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.

[54] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1915-1929, 2012.

[55] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in Advances in neural information processing systems, 2014, pp. 1799-1807.

[56] C. Szegedy et al., "Going deeper with convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1-9.

[57] N. Auslander, Y. I. Wolf, and E. V. Koonin, "In silico learning of tumor evolution through mutational time series," Proceedings of the National Academy of Sciences, vol. 116, no. 19, pp. 9501-9510, 2019.

[58] P. Werbos, "New tools for Prediction and Analysis in the Behavioral Sciences," Ph. D. dissertation, Harvard University, 1974.

[59] D. Parker and L. Logic, "Technical Report TR-47," Massachusetts Institute of Technology, 1985.

[60] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," nature, vol. 323, no. 6088, pp. 533-536, 1986.

[61] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111-3119.

[62] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in Proceedings of the 28th international conference on machine learning (ICML-11), 2011, pp. 1017-1024.

[63] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[64] S. Saha, J. R. Celaya, V. Vashchenko, S. Mahiuddin, and K. F. Goebel, "Accelerated aging with electrical overstress and prognostics for power MOSFETs," in IEEE 2011 EnergyTech, 2011, pp. 1-6: IEEE.

[65] Y. Umuroglu et al., "Finn: A framework for fast, scalable binarized neural network inference," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017, pp. 65-74.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[67] F. e. c. o. t. p. a. h. m. s. 2020. (2020). phme data challenge 2020. Available: phmeurope.org/2020/data-challenge-2020