Recurrent Neural Networks for Online Remaining Useful Life Estimation in Ion Mill Etching System

Vishnu TV[1], Priyanka Gupta[2], Pankaj Malhotra[3], Lovekesh Vig[4], and Gautam Shroff[5]

[1,2,3,4,5]*TCS Research, Noida, Uttar Pradesh, 201309, India*

vishnu.tv@tcs.com priyanka.g35@tcs.com malhotra.pankaj@tcs.com lovekesh.vig@tcs.com gautam.shroff@tcs.com

ABSTRACT

We describe the approach – submitted as part of the 2018 PHM Data Challenge – for estimating time-to-failure or Remaining Useful Life (RUL) of Ion Mill Etching Systems in an online fashion using data from multiple sensors. RUL estimation from multi-sensor data can be considered as learning a regression function that maps a multivariate time series to a real-valued number, i.e. the RUL. We use a deep Recurrent Neural Network (RNN) to learn the metric regression function from multivariate time series. We highlight practical aspects of the RUL estimation problem in this data challenge such as i) multiple operating conditions, ii) lack of knowledge of exact onset of failure or degradation, iii) different operational behavior across tools in terms of range of values of parameters, etc. We describe our solution in the context of these challenges. Importantly, multiple modes of failure are possible in an ion mill etching system; therefore, it is desirable to estimate the RUL with respect to each of the failure modes. The data challenge considers three such modes of failures and requires estimating RULs with respect to each one, implying learning three metric regression functions - one corresponding to each failure mode. We propose a simple yet effective extension to existing methods of RUL estimation using RNN based regression to learn a single deep RNN model that can simultaneously estimate RULs corresponding to all three failure modes. Our best model is an ensemble of two such RNN models and achieves a score of 1.91×10^7 on the final validation set.

1. INTRODUCTION

With the advent of Industrial Internet of Things (IIOT) (Xu et al., 2014), large amounts of temporal sensor data is available in (near) real-time leading to an increasing interest in remote monitoring of equipment. Typically, a large number of sensors are installed across various components and subcomponents of a complex system. This leads manual monitoring of the system extremely challenging. Data-driven approaches can aid operators to monitor the sensor data and generate suitable alerts along with potential diagnostics in case of malfunctioning system. Building data-driven or machine learning based models for fault detection and prognostics (remaining useful life estimation) from sensor data can help in real-time monitoring of equipment, avoid catastrophic failures, enable condition-based maintenances, as well as help to take key engineering decisions, e.g. to improve future manufacturing processes.

Recently, deep Recurrent Neural Networks (RNNs) based on gated units such as Long Short Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) have been successfully used for modeling sequential data. It has been shown that RNNs can model the temporal (sequential) aspect of the sensor data as well as capture the inter-sensor dependencies Malhotra et al. (2015). RNNs have been used to model behavior of machines based on multi-sensor time series with applications to anomaly and fault detection (Malhotra et al., 2015; Malhotra, Ramakrishnan, et al., 2016; Yadav et al., 2016; Filonov et al., 2016), Remaining Useful Life (RUL) estimation (Malhotra, TV, et al., 2016; Gugulothu et al., 2017; TV et al., 2018), and diagnostics (TV et al., 2017; Gugulothu et al., 2018).

Several approaches for RUL estimation using RNNs have been proposed in the past for various type of equipment, e.g. turbofan engines (Heimes, 2008; Malhotra, TV, et al., 2016; Gugulothu et al., 2017), milling machines (Malhotra, TV, et al., 2016), etc. These approaches can be categorised into two types: supervised and semi-supervised. Supervised approaches model RUL estimation as a metric regression problem where RUL is considered to be a real-valued number and a metric regression function – modeled via a (deep) RNN – is learned to map the time series of sensor data to RUL. Examples of this approach include (Heimes, 2008; Zheng et al., 2017; TV et al., 2018). Semi-supervised approaches first learn a deep RNN based model of normal behavior, which is then used to obtain a health index trend of any instance of a machine. The health index trend of a test instance is com-

pared to that of the historical failed train instances to obtain an estimate of RUL via curve matching Wang et al. (2008); Malhotra, TV, et al. (2016); Gugulothu et al. (2017). In this work, we adapt and modify the supervised metric regression approaches using deep RNNs. Specifically, we build a metric regression model using deep LSTM networks (LSTM-MR) in a multi-task setting for RUL estimation.

The rest of the paper is organized as follows: In Section 2 we present some related work and provide details of the data challenge in Section 3. We describe briefly the deep LSTM Networks in Section 4 and give details of our approach and experiments in Section 5 and Section 6 respectively, and finally conclude in Section 7.

2. RELATED WORK

An important class of approaches for RUL estimation is based on trajectory similarity, e.g. Wang et al. (2008); Khelif et al. (2014); Lam et al. (2014); Malhotra, TV, et al. (2016); Gugulothu et al. (2017). These approaches compare the health index trajectory or trend of a test instance with the trajectories of failed train instances to estimate RUL using a distance metric such as Euclidean distance. Such approaches work well when trajectories are smooth and monotonic in nature but are likely to fail in scenarios when there is noise or intermittent disturbances (e.g. spikes, operating mode change, etc.) as the distance metric may not be robust to such scenarios Gugulothu et al. (2017).

Another class of approaches is based on metric regression. Unlike trajectory similarity based methods which rely on comparison of trends, metric regression methods attempt to learn a function to directly map sensor data to RUL, e.g. Heimes (2008); Benkedjouh et al. (2013); Dong et al. (2014); Babu et al. (2016); Gugulothu et al. (2017); Zheng et al. (2017). Such methods can better deal with non-monotonic and noisy scenarios by learning to focus on the relevant underlying trends irrespective of noise. Within metric regression methods, few methods consider non-temporal models such as Support Vector Regression for learning the mapping from values of sensors at a given time instance to RUL, e.g. Benkedjouh et al. (2013); Dong et al. (2014). Deep temporal models such as those based on RNNs Heimes (2008); Malhotra, TV, et al. (2016); Gugulothu et al. (2017); Zheng et al. (2017) or Convolutional Neural Networks (CNNs) Babu et al. (2016) can capture the degradation trends better compared to non-temporal models, and are proven to perform better. Moreover, these models can be trained in an end-to-end learning manner without requiring feature engineering.

3. DATA CHALLENGE DESCRIPTION

The data challenge focuses on predicting time-to-failure (for each of three types of fault) at specific times of an ion mill etching tool.

3.1. Ion Mill Etching System

An ion mill etching tool is shown in Figure 1(a). The process of ion mill etching typically consists of the following steps:

- Inserting a wafer into the mill.
- Configuring wafer settings (rotation speed, angles, beam current / voltages, etc).
- Processing the wafer for a set amount of time.
- Repeating the 2^{nd} or 3^{rd} step for different steps of recipe.
- Removing wafer from mill.

An ion source generates ions that are accelerated through an electric field using a series of grids set at specific voltages. This creates an ion beam that travels and eventually strikes the wafer surface. Material is removed from the wafer when ions hit the wafer surface. The wafer is placed on a rotating fixture that can be tilted at different angles facing the incoming ion beam. The wafer can be shielded from the ion beam until ready for milling operation to commence using a shutter mechanism as shown in Figure 1(b). A Particle Beam Neutralizer (PBN) control system influences the ion beam shape / ion distribution as it travels to the wafer surface. The wafer is cooled by a helium / water system called flowcool. The cooling system passes helium gas behind the wafer at a specified flow rate. The helium gas is indirectly cooled by a water system. The wafer and fixture o-ring separates the flowcool gas from the ion mill vacuum chamber.

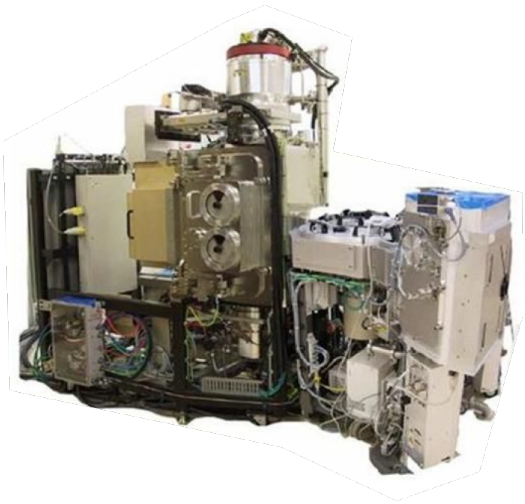
3.2. Objectives

The objective is to build a model from time-series sensor data collected from various ion mill etching tools operating under various conditions and settings. The goal is to examine the fault behavior of an ion mill etching tool used in a wafer manufacturing process[1]. Many different failure mechanisms such as leaks between flowcool and ion mill chambers, electric grid wear, ion chamber wear, etc. can be present in this system. Predicting the time of these failures can help in condition-based maintenance and schedule downtimes of the ion mills for maintenance operations. The problem consists of diagnosing failures (i.e. detect and identify) and determining time remaining until next failure (i.e. predict remaining useful life).

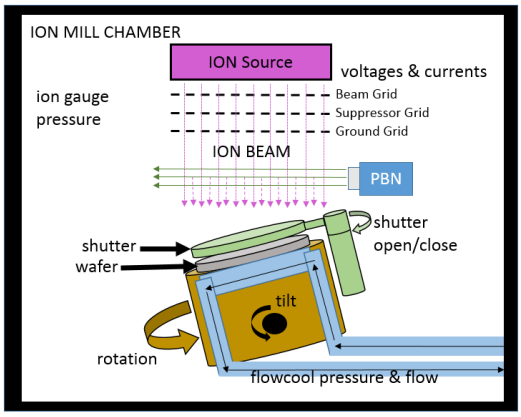
Time-to-failure for the following three different failure modes is of interest:

- F_1: Flowcool Pressure Dropped Below Limit (FCP Low. Figure 2(a))
- F_2: Flowcool Pressure Too High Check Flowcool Pump (FCP High. Figure 2(b))
- F_3: Flowcool leak (FC Leak)

[1] http://www.ionbeammilling.com/about_the_ion_milling_process, https://www.azom.com/article.aspx?ArticleID=7533



(a) An Ion Mill Etching System



(b) Wafer and Ion Mill Etching Process

Figure 1. Ion Mill Etching System and Process Overview

3.3. Dataset details

The training data corresponds to 20 tools. The testing data corresponds to a subset containing 5 tools out of the 20 tools. The data consists of 24 columns and 3 fault types. The time period for testing data is after the training data such that there is no overlapping time period. The columns S1 - S24 contain sensor data and other process information for tool ids arranged with timestamp. The various parameters are listed in Table 1. There are 5 categorical variables, 14 numeric operating condition related variables and 5 numeric parameters obtained through sensors installed on the system. The timestamp and type of failure are available for the 20 tools in the training dataset. The data has been anonymized so the units of measurement for various parameters are not provided.

It is to be noted that the time of failure is actually the time when the operator shuts down the machine for maintenance rather than the time when the actual fault is observed. The actual start of the failure may occur much earlier than the provide failure time. The train folder contains the training data to be used for modeling purposes. The test folder contains the test data that is to be used to generate submissions. The time where faults occur is found in the train/train_faults folder. Number of data points and faults for each tool id are listed in Table 2. Example for time-to-failure examples are provided in the train/train_tff folder. There are 'null' (NaN) values where faults do not occur in within a specified time horizon. The 20 .csv files under the train folder represent the sensor data that are used as predictors. Each of these files represent a separate ion milling tool. The sensor-wise statistics are provided in Table 3. From Table, we can see that mean and standard deviation of all sensors are very close to 0 and 1, which indicates that provided data is Z-normalized.

3.4. Scoring functions

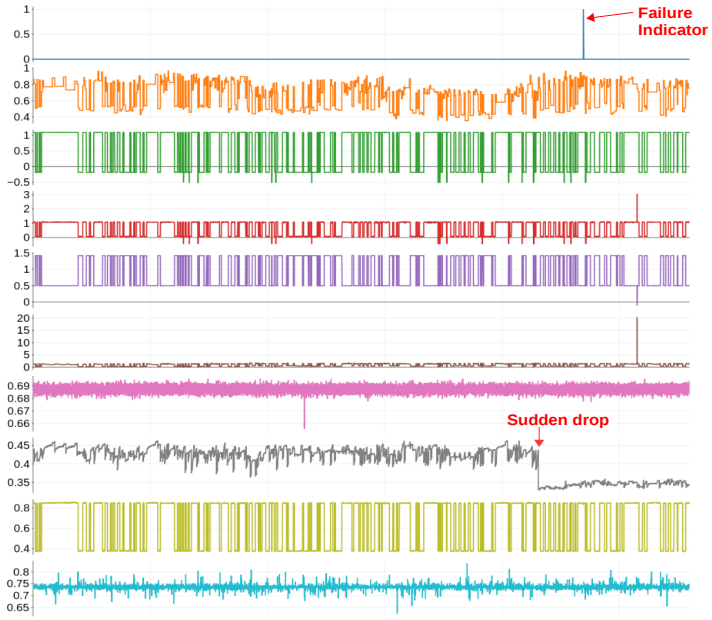
The functions for computing Original Score (S_1) and Secondary Score (S_2) used during the testing phase (Phase-1) and validation (Phase-2) phases respectively, are provided in Table 4. In this data challenge, *lower scores indicate better performance*. A secondary score is used in the validation portion of the contest. The secondary score is similar in nature to the original score. However, the penalty is more severe for false positives and false negatives. The final score (S) is the average of the original and secondary scores, and is computed as follows:

$$S = \frac{S_1 + S_2}{2}$$

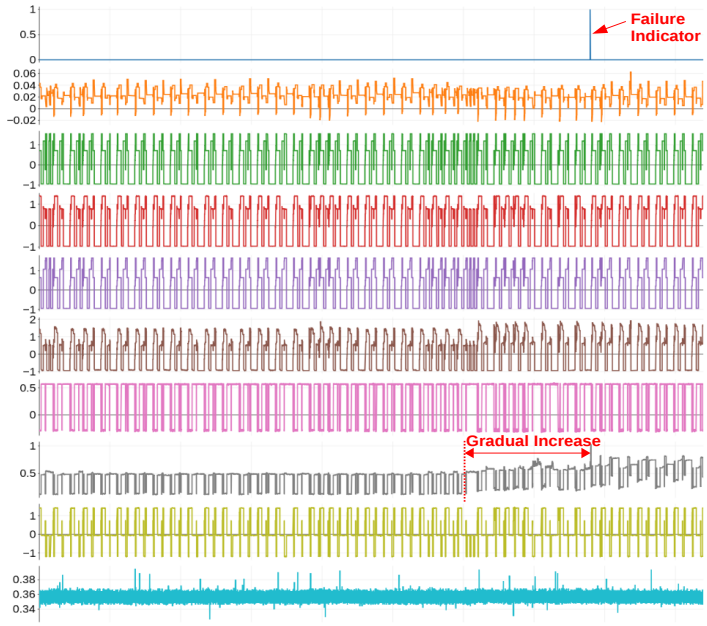
3.5. Challenges

We first highlight few aspects and challenges while formulating an approach for the 2018 PHM data challenge for RUL estimation of ion mill etching System, and then describe our approach that can potentially deal with these challenges:

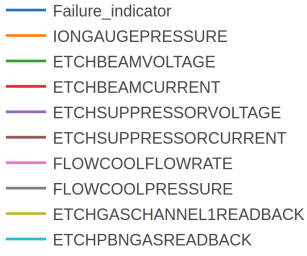
1. *Dealing with multiple fault types leading to failures with missing data prior to reported time of failure*: The failures considered corresponded to three types of faults. Data prior to failures which is critical to estimate RULs (data points close to failure) is sparse. Average, minimum and maximum missing points before failure shutdowns are provided in Table 5. This issue is further magnified when we consider failure types independently, leading to very few failure instances with sensor data available close to failure time.
2. The *nature of evolution of faults over time* is not known, i.e. the nature of machine health degradation trends is not known. It may be possible that one or more faults are instantaneous in nature. The time at which the machine is shutdown due to a particular type of fault is given. However, the time taken to respond after the observation of symptoms can vary and not known. For example, it can be seen from Figure 2(a) that there is a sudden drop



(a) F_1 : Flowcool Pressure Dropped Below Limit (FCP Low)



(b) F_2 : Flowcool Pressure Too High Check Flowcool Pump (FCP High)



(c) Legend for subfigures (a)-(b)

Figure 2. Sample time series plots with fault signatures. Image best viewed in color.

Table 1. Sensors used in model building

Sensor Name	Sensor Type	Used in model building
stage	Categorical	N
Lot	Categorical	N
runnum	Numeric	N
recipe	Categorical	N
recipe_step	Categorical	N
IONGAUGEPRESSURE	Numeric (Sensor)	Y
ETCHBEAMVOLTAGE	Numeric	Y
ETCHBEAMCURRENT	Numeric	Y
ETCHSUPPRESSORVOLTAGE	Numeric	Y
ETCHSUPPRESSORCURRENT	Numeric (Sensor)	Y
FLOWCOOLFLOWRATE	Numeric	Y
FLOWCOOLPRESSURE	Numeric (Sensor)	Y
ETCHGASCHANNEL1READBACK	Numeric	Y
ETCHPBNGASREADBACK	Numeric	Y
FIXTURETILTANGLE	Numeric	N
ROTATIONSPEED	Numeric	N
ACTUALROTATIONANGLE	Numeric (Sensor)	N
FIXTURESHUTTERPOSITION	Numeric	Y
ETCHSOURCEUSAGE	Numeric	N
ETCHAUXSOURCETIMER	Numeric	N
ETCHAUX2SOURCETIMER	Numeric	N
ACTUALSTEPDURATION	Numeric (Sensor)	N

Table 2. Number of data points and faults for each tool id

Tool-ID	Total Points $(/10^6)$	F_1	F_2	F_3
01_M02	5.11	53	40	16
02_M02	4.14	5	28	0
03_M01	3.43	42	51	0
04_M01	5.10	7	1	1
06_M01	3.81	10	3	3

in Flowcool Pressure but the failure is marked at a later point in time.

3. *Lack of knowledge of exact time of onset of failures in the training set:* Despite having access to large number of failure instances, the onset of failure is very challenging to identify as there is a large variance in the time between onset of a failure and the shutdown of a machine. This can be seen, for example, in Figure 2(b) where there is a gradual increase in Flowcool Pressure as we move towards failure.

4. *Extremely large range of possible RUL values*: It is well-known that it is difficult to estimate the remaining useful life unless there is at least one symptom of an approaching failure or an onset of failure. In this dataset, the average time between two failures is 60.82×10^4 with the maximum time between two failures being as large as 22.26×10^6. It is therefore challenging to model such a large variance in the RUL values. Possible range of RUL values for each tool id are listed in Table 6.

5. *Sequence of faults in a given tool*: Several shutdowns caused by different types of faults are reported for each tool. Therefore, it is difficult to model the normal operational behavior of a tool as a tool may be normal with respect to one fault type but may be depicting abnormal behavior or symptoms with respect to another fault type.

6. *Multiple operating conditions*: There are various parameters such as fixture shutter position, stage, recipe, recipe steps, that determine the operating condition. Each parameter can have a large number of values (one at a time). All possible combinations of all parameter's values (all operating conditions) are large in number. Percentage of data points with respect to shutter position are listed in Table 7.

4. BACKGROUND: DEEP LSTM NETWORKS

We use a variant of LSTMs as described in Zaremba et al. (2014) in the hidden layers of the neural network. Intuitively, an LSTM unit maintains a cell state using an input gate, a forget gate, and an output gate: at a given time step, the input gate decides what should be added to the cell state, forget gate decides what should be removed from the cell state, and the output gate decides what part of cell state should be given as an output from the LSTM unit. Hereafter, we denote column vectors by bold small letters and matrices by bold capital letters. For a hidden layer with h LSTM units, the values for the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, output gate $\mathbf{o}_t$, hidden state $\mathbf{z}_t$, and cell state $\mathbf{c}_t$ at time t are computed using the current input $\mathbf{x}_t$, the previous hidden state $\mathbf{z}_{t-1}$, and the cell state $\mathbf{c}_{t-1}$, where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{z}_t$, and $\mathbf{c}_t$ are real-valued h-dimensional vectors such that $\mathbf{z}_t = f(\mathbf{x}_t, \mathbf{z}_{t-1}, \mathbf{c}_{t-1})$ as given by Equations 1.

Consider $W_{n_1,n_2} : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$ to be an affine transform of the form $\mathbf{z} \mapsto \mathbf{W}\mathbf{z} + \mathbf{b}$ for matrix $\mathbf{W}$ and vector $\mathbf{b}$ of appropriate dimensions. In the case of a multi-layered LSTM network with L layers and h units in each layer, the hidden state $\mathbf{z}_t^l$ at time t for the l-th hidden layer is obtained from the hidden state at $t-1$ for that layer $\mathbf{z}_{t-1}^l$ and the hidden state at t for the previous $(l-1)$-th hidden layer $\mathbf{z}_t^{l-1}$. The time

Table 3. Sensor-wise statistics

Sensor Name	Min.	Max.	Mean	St.Dev.
IONGAUGEPRESSURE	-1.9079	165.6602	0.0095	1.0737
ETCHBEAMVOLTAGE	-1.5069	3.6962	0.0032	0.9975
ETCHBEAMCURRENT	-1.5120	3.6552	0.0035	0.9968
ETCHSUPPRESSORVOLTAGE	-1.5752	2.0325	0.0093	0.9983
ETCHSUPPRESSORCURRENT	-1.5069	24.6522	0.0098	1.0046
FLOWCOOLFLOWRATE	-2.8429	2.9403	0.0011	1.0007
FLOWCOOLPRESSURE	-2.3330	15.1639	0.0281	1.0619
ETCHGASCHANNEL1READBACK	-1.9010	6.2425	0.0045	0.9968
ETCHPBNGASREADBACK	-2.8016	2.5711	0.0034	0.9985
FIXTURETILTANGLE	-1.6493	22.2856	0.0008	1.0139
ROTATIONSPEED	-91.6578	25.6777	-0.0179	0.7124
ACTUALROTATIONANGLE	-17.2244	27.7990	-0.0166	0.9578

Table 4. Scoring functions used in test and validation phases. Here, R: Ground Truth TTF, $\hat{R}$: Submission TTF.

R	$\hat{R}$	**Original Score** (S_1)	**Secondary Score** (S_2)
Num	Num	exp(-0.001*R)*abs(R-$\hat{R}$)	0.1*(R-$\hat{R}$)^2
NaN	Num	exp(-0.001*$\hat{R}$)*abs($\hat{R}$)	5.0/(abs($\hat{R}$)+3)
Num	NaN	exp(-0.001*R)*abs(R)	20*exp(-1.0/(abs(R)+0.1))
NaN	NaN	0	0

Table 5. Average, minimum and maximum missing points before failure shutdowns

Tool-ID	Min	Max ($/10^3$)	Average ($/10^3$)
01_M02	0	120.5	14.01
02_M02	0	81.7	16.27
03_M01	0	83.2	12.57
04_M01	0	5.6	1.17
06_M01	0	9.7	2.51

series goes through the following transformations iteratively at l-th hidden layer for $t = 1$ through T, where T is length of the time series:

$$\begin{pmatrix} \mathbf{i}_t^l \\ \mathbf{f}_t^l \\ \mathbf{o}_t^l \\ \mathbf{g}_t^l \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W_{2h,4h} \begin{pmatrix} \mathbf{D}(\mathbf{z}_t^{l-1}) \\ \mathbf{z}_{t-1}^l \end{pmatrix} \tag{1}$$

where the cell state $\mathbf{c}_t^l$ is given by $\mathbf{c}_t^l = \mathbf{f}_t^l \mathbf{c}_{t-1}^l + \mathbf{i}_t^l \mathbf{g}_t^l$, and the hidden state $\mathbf{z}_t^l$ is given by $\mathbf{z}_t^l = \mathbf{o}_t^l tanh(\mathbf{c}_t^l)$. We use dropout for regularization Pham et al. (2014), which is applied only to the non-recurrent connections, ensuring information flow across time-steps for any LSTM unit. The dropout operator $\mathbf{D}(\cdot)$ randomly sets the dimensions of its argument to zero with probability equal to a dropout rate, $\mathbf{z}_t^0$ is the input $\mathbf{x}_t$ at time t. The sigmoid (σ) and $tanh$ activation functions are applied element-wise.

In a nutshell, this series of transformations for $t = 1 \dots T$, converts the input time series $\mathbf{x}^{1 \dots T}$ of length T to a fixed-dimensional vector $\mathbf{z}_T^L \in \mathbb{R}^h$. We, therefore, represent the LSTM network by a function f_{LSTM} such that $\mathbf{z}_T^L = f_{LSTM}(\mathbf{x}; \mathbf{W})$, where $\mathbf{W}$ represents all the parameters of the LSTM network.

5. REMAINING USEFUL LIFE ESTIMATION USING MULTI-TASK LSTM-MR

In this section we describe how we formulate the RUL estimation problem. We intend to learn a single mapping function to estimate RUL values for all the fault types simultaneously and so we model it as a multi tasking problem of metric regression tasks. Each metric regression task performs RUL estimation for each of the fault types and the LSTM-MR network is trained to perform all the tasks at once.

5.1. RUL Estimation Problem Formulation

Consider a training set $\mathcal{D} = \{\mathbf{x}_i^{1 \dots T}, \mathbf{r}_i\}_{i=1}^N$ containing multivariate real-valued time series $\mathbf{x}_i^{1 \dots T} = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \dots, \mathbf{x}_i^T\}$ where $\mathbf{x}_i^t \in \mathbb{R}^m$ and RUL vector corresponding to the last timestep of the time series $\mathbf{r}_i = \{r_i^1, r_i^2, ..., r_i^C\}$. Here m is the number of sensors (input parameters), T is the length of the time series, $r_i^j \in \mathbb{R}$ with $j = 1, \dots C$, where C is the number of faults, such that there is one RUL value corresponding to each fault type. In general, the N instances are obtained from data across tools by suitable windowing and normalization as described in detail in Section 6. The goal is to learn a mapping function f_{MR} that maps any given multivariate time series $\mathbf{x}_i^{1 \dots T}$ to a vector of RUL estimates $\hat{\mathbf{r}}_i$ corresponding to the ground truth $\mathbf{r}_i$. In the following subsection, we describe simple pre-processing done on the time series before inputting it to the model.

5.2. Pre-processing

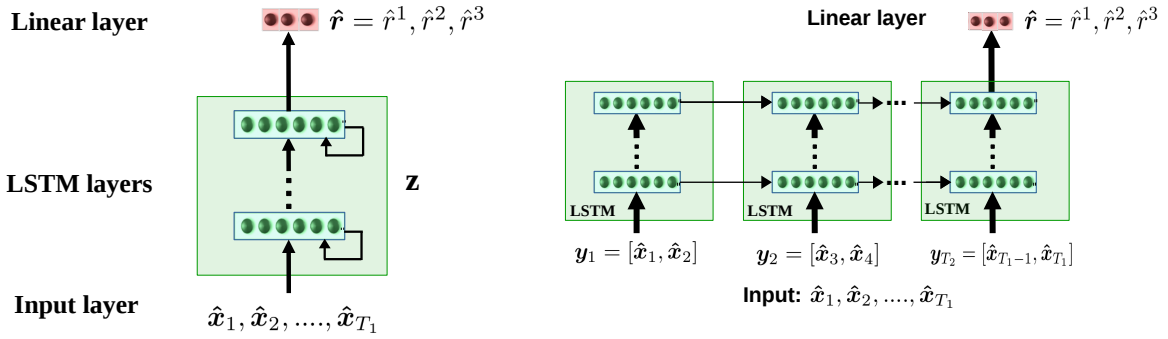
In pre-processing, first, since we have to deal with time series of large lengths, we perform the standard downsampling by a factor of d_1. The downsampled time series is given by,

$$\hat{\mathbf{x}}_i^k = \mathbf{x}_i^{k.d_1} \in \mathbb{R}^m$$

where $k = 1, 2, 3, \dots, T_1$ and $T_1 = T/d_1$. Here $\hat{\mathbf{x}}_i^k$ is an m dimensional vector representing the value at k^{th} timestep of the downsampled time series of length, T_1.

Table 6. Possible range of RUL values

Tool-ID	$\mathbf{F}_1$		$\mathbf{F}_2$		$\mathbf{F}_3$	
	$R_{min.}$	$R_{max.}(/10^6)$	$R_{min.}$	$R_{max.}(/10^6)$	$R_{min.}$	$R_{max.}(/10^6)$
01_M02	0	9.92	0	9.18	0	10.31
02_M02	0	5.87	0	3.12	NaN	NaN
03_M01	0	15.52	0	6.75	NaN	NaN
04_M01	0	22.26	0	.61	0	4.99
06_M01	0	26.54	0	25.57	0	14.37



(a) LSTM-MR (b) LSTM-MR unrolled over time. $d_2 = 2$ for illustration of T2V operation.

Figure 3. LSTM based Metric Regression Neural Network Architecture

Table 7. Percentage of points with respect to shutter position

Shutter Position	Percentage of Points
0	33.599
1	52.589
2	1.985
3	0.784
255	0.057
missing	10.986

In the second step, we further reduce the time series length by a *Time series to Vector* (T2V) operation. In T2V, we decrease the sequence length by a factor of d_2 but also increase the dimensionality of the resultant time series by the same factor d_2 unlike in the usual downsampling. This is done to ensure having a smaller sequence length by retaining all the information in the time-series. Also, we assume that for a small d_2, we do not lose any significant temporal information.

Formally, T2V can be described as follows.

$$\mathbf{y}_i^j = [\hat{\mathbf{x}}_i^{(j-1).d_2+1}, \hat{\mathbf{x}}_i^{(j-1).d_2+2}, \ldots, \hat{\mathbf{x}}_i^{j.d_2}] \in \mathbb{R}^{m.d_2}$$

where $j = 1, 2, 3, \ldots, T_2$ and $T_2 = T_1/d_2$. Here $\mathbf{y}_i^j$ is an $m.d_2$ dimensional vector representing the value at j^{th} timestep of the resultant time series of length, T_2. Effectively, through the two pre-processing steps, a time series window of length T is converted to a time series of length $\frac{T}{d_1.d_2}$, making it computationally more efficient to be processed by the LSTM-MR Network.

We impose an upper bound r_u on any RUL value r_i^j as, in practice, it is not possible to provide meaningful estimates of RUL too early in the life of a tool, e.g. if the tool is in perfect health and there are no symptoms of degradation. So if any $r_i^j \geq r_u$, it is clipped to r_u. We describe our training and inference procedures using f_{MR} in the next subsection.

5.3. Model training

We train an LSTM-MR network to estimate the RUL vector $\hat{\mathbf{r}}_i$, given an input time series $\mathbf{y}_i^{1\ldots T_2}$. At the output of the network we have C linear units estimating the remaining useful life values for C fault types. The network is trained using the standard MSE loss function as given below.

$$\begin{aligned} \mathbf{z}_{T_2}^L &= f_{LSTM}(\mathbf{y}_i^{1\ldots T_2}; \mathbf{W}) \\ \hat{\mathbf{r}}_i &= \mathbf{W}_O\, \mathbf{z}_{T_2}^L + \mathbf{b}_O \\ \mathcal{L}(\mathbf{r_i}, \hat{\mathbf{r}}_i) &= \frac{1}{C}\sum_{p=1}^{C}(r_i^p - \hat{r}_i^p)^2 \end{aligned} \tag{2}$$

where $\mathbf{W}$ represents all the parameters of the LSTM network and $\mathbf{W}_O$ and $\mathbf{b}_O$ represent the weights and biases of the output linear layer which map the LSTM network's output to RUL estimations. The total loss function can be given by $\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\mathbf{r}_i, \hat{\mathbf{r}}_i)$. We minimize this loss using stochastic gradient descent method and the standard backpropagation through time for RNNs.

From equation 2 and the pre-processing described in 5.2, we

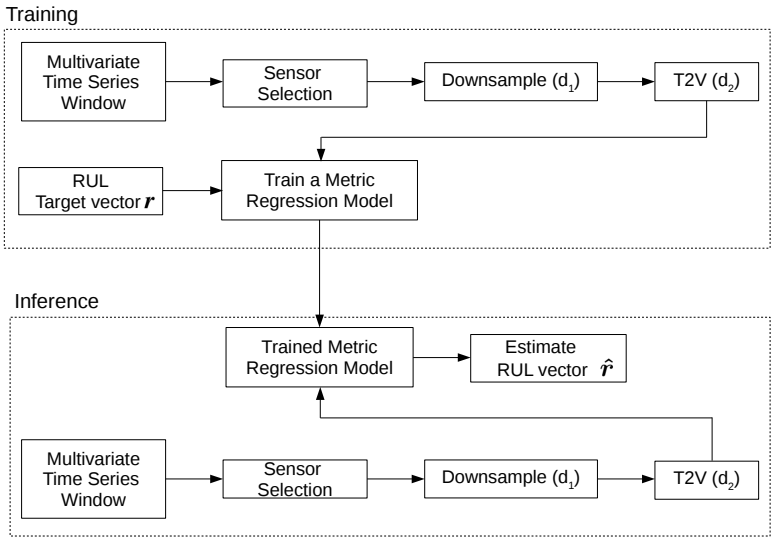


Figure 4. Approach Overview with Training and Inference Phases. Here, T2V: Time series to Vector operation.

can concisely represent the entire process as,

$$\hat{\mathbf{r}}_i = f_{MR}(\mathbf{x}_i^{1...T}; d_1, d_2, \mathbf{W}, \mathbf{W}_O, \mathbf{b}_O)$$

Once trained, it can be noted that f_{MR} can be used in an online fashion. For a current time instance t, the latest T points (corresponding to time instances $t-T+1, \ldots, t$), can be input to f_{MR} to estimate the RUL values for the C fault types. The process of training and inference using f_{MR} is shown in Figure 4.

6. EXPERIMENTAL EVALUATION

In this section, we present our experimental setup which includes the details of pre-processing and choosing values for various parameters. We later present the results of our LSTM-MR approach.

6.1. Experimental Setup

In the training set, we are provided with data for 20 tools. According to the functional description of the apparatus, we infer that the etching process is carried out only when the parameter *fixture shutter position* takes a value of 1. So, for all the tools, we consider only those points for which *fixture shutter position* is equal to 1 and also use only the 9 sensors as marked in Table 1 for model building. We then split the data of each tool into overlapping windows of length 2000 ($T = 2000$) points with an overlap of 500 points. Using the time to failure values given for the tools in the train set, we get the values for the target RUL vector $\mathbf{r}_i$ for each of the windows. We use a value of $r_u = 500$. We had around 0.1M windows out of which only around 150 were having at least one of $r_i^1, r_i^2, ..., r_i^C$ less than r_u. To handle such an imbalance in the target RUL values in training, we retain all the windows for which at least one of $r_i^1, r_i^2, ..., r_i^C$ is less than r_u and randomly sample from the remaining set of windows to form our complete training set. For training, we divide the clipped target RUL values by r_u to normalize them, such that the target RUL values lie in the $[0, 1]$ range. After estimating $\hat{\mathbf{r}}_i$, RUL values on the original scale are obtained by multiplying with r_u. According to the dataset we have three fault types and hence $C = 3$. For the two steps described in section 5.2, we use $d_1 = 2$ and $d_2 = 10$ to pre-process the time series before inputting them to the model.

For the LSTM network, we choose the number of units h from the set $\{50, 100, 150, 200\}$ and use $L = 2$ layers. We use early stopping with a maximum of 2500 iterations of training with a batch size of 32. Also we use dropout (Zaremba et al., 2014) with a value of 0.4 over the feedforward connections for regularization, and use Adam optimizer (Kingma & Ba, 2014) for optimizing the weights of the networks with an initial learning rate of 0.005 for all our experiments. We chose the best architecture (by varying the number of hidden units (h)) as the one with minimum RMSE on a labeled hold out set (taken from the training set of 20 tools). We will present results of our approach in the next subsection.

6.2. Results

We have evaluated the performance of our LSTM-MR approach using f_{MR} on the data sets provided in the test (phase-1) and validation (phase-2) phases provided during the challenge. We restrict our RUL estimations between 100 and 150 and any RUL estimate falling below 100 or above 150

Table 8. Performance of LSTM-MR and LSTM-MR-Ensemble in terms of S_1 and S_2 on test and validation phase data sets. P_{test} and $P_{validation}$ denote the number of non-NaN estimations made on the test and validation phase data sets respectively. S_1, S_2 and S are scores as described in Section 3.4. Lower scores indicate better performance.

Approach	P_{test}	$P_{validation}$	S_1 on test	S_1 on validation	S_2 on validation	S on validation
LSTM-MR	4210	3929	63.01	96.48	5.26×10^8	2.63×10^8
LSTM-MR-Ensemble	300	270	**62.93**	**96.40**	$\mathbf{3.82 \times 10^7}$	$\mathbf{1.91 \times 10^7}$

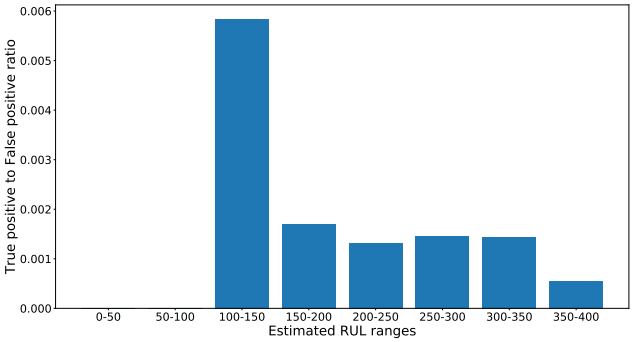


Figure 5. True positives to False positives ratio with respect to estimated RUL ranges

is treated as NaN. These two thresholds were selected by estimating true positives to false positives ratios for various ranges of estimated RUL values, on a sample of points taken from the provided train set. An estimated RUL value is considered a true positive, if $|r_i^p - \hat{r}_i^p| \leq 100$, $p = 1, 2, \ldots, C$, and a false positive, otherwise. As shown in Figure 5, the range 100 to 150 of the estimated RUL values shows the maximum true positives to false positives ratio. Hence the estimates outside the interval $[100, 150]$ were replaced by NaN. Along with LSTM-MR, we also present results for LSTM-MR-Ensemble which is an ensemble of the two best models, say M_1 and M_2, picked on the basis of a hold out set. Even if one of M_1 and M_2 estimates NaN, we consider the final RUL estimate to be NaN and we take the maximum of the estimations from M_1 and M_2 otherwise for LSTM-MR-Ensemble.

From Table 8, we can see that in both the test and validation phases, LSTM-MR-Ensemble performs better than LSTM-MR. It can be noted from Table 4 that, when the ground truth is actually a number, S_2 for estimating the RUL to be a number is mostly higher compared to estimating it to be NaN. This is the reason for the significant difference, in terms of S_2, between LSTM-MR-Ensemble and LSTM-MR on the validation phase data set. Also, LSTM-MR-Ensemble estimates more number of NaN values as compared to LSTM-MR (as denoted by P_{test} and $P_{validation}$ in Table 8) as it is an ensemble and hence results in lesser false positives.

7. CONCLUSION

We have described the approach used for the 2018 PHM Data Challenge. We have highlighted the challenges in the supervised learning based approach such as missing data close to failures, noisy labels for failures due to lack of knowledge of onset of failure, learning to estimate very large RUL values (very early prediction of failure), capturing temporal dependencies from very long multivariate time series, dealing with multiple operating conditions, etc. and described our solution in the context of these challenges. Our approach leverages deep recurrent neural networks to learn a supervised model for remaining useful life estimation for three types of failure modes in ion mill etching system. Our approach leverages data from all tool-ids and failure modes to learn one general-purpose model for all tools and failure modes. Importantly, the proposed approach is able to estimate RUL in an online fashion. We also found that an ensemble of RNN models significantly improves the results. In future, it will be interesting to explore if a semi-supervised approach (e.g. as in Malhotra, Ramakrishnan, et al. (2016); Malhotra, TV, et al. (2016)) can be used to mitigate the issue of lack of knowledge of the time of onset of failure to find the change points, i.e. the time instances when the health index starts decreasing or the first symptoms of failure appear, and the RNN model can then be used to estimate RUL only after the change points.

REFERENCES

Babu, G. S., Zhao, P., & Li, X.-L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International Conference on Database Systems for Advanced Applications* (pp. 214–228).

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2013). Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelligence*, *26*(7), 1751–1760.

Dong, H., Jin, X., Lou, Y., & Wang, C. (2014). Lithium-ion battery state of health monitoring and remaining useful life prediction based on support vector regression-particle filter. *Journal of power sources*, *271*, 114–123.

Filonov, P., Lavrentyev, A., & Vorontsov, A. (2016). Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *NIPS Time Series Workshop 2016, arXiv preprint arXiv:1612.06676*.

Gugulothu, N., TV, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Predicting remaining useful life us-

ing time series embeddings based on recurrent neural networks. *International Journal on Prognostics and Health Management, IJPHM. arXiv preprint arXiv:1709.01073*.

Gugulothu, N., TV, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2018). On practical aspects of using rnns for fault detection in sparsely-labeled multi-sensor time series. In *PHM Conference*.

Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. In *Prognostics and Health Management, 2008. PHM 2008.* (pp. 1–6).

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Khelif, R., Malinowski, S., Chebel-Morello, B., & Zerhouni, N. (2014). Rul prediction based on a new similarity-instance based approach. In *IEEE International Symposium on Industrial Electronics.*

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lam, J., Sankararaman, S., & Stewart, B. (2014). Enhanced trajectory based similarity prediction with uncertainty quantification. *PHM 2014*.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder. *1st ACM SIGKDD Workshop on ML for PHM. arXiv preprint arXiv:1608.06154*.

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 89–94).

Pham, V., Bluche, T., Kermorvant, C., & Louradour, J. (2014). Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR)* (pp. 285–290).

TV, V., Gugulothu, N., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Bayesian networks for interpretable health monitoring of complex systems. In *Workshop on AI for Internet of Things at IJCAI.*

TV, V., Malhotra, P., Vig, L., & Shroff, G. (2018). Deep ordinal regression for remaining useful life estimation from censored data.

Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *IEEE International Conference on Prognostics and Health Management, 2008. PHM 2008.* (pp. 1–6).

Xu, L. D., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, *10*(4), 2233–2243.

Yadav, M., Malhotra, P., Vig, L., Sriram, K., & Shroff, G. (2016). Ode-augmented training improves anomaly detection in sensor data from machines. *arXiv preprint arXiv:1605.01534*.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

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