RULENet: End-to-end Learning with the Dual-estimator for Remaining Useful Life Estimation

*

Masanao Natsumeda
NEC Laboratories America, Inc.
Princeton, NJ
mnatsumeda@nec-labs.com

Haifeng Chen
NEC Laboratories America, Inc.
Princeton, NJ
haifeng@nec-labs.com

***Abstract*—Remaining Useful Life (RUL) estimation is a key element in Predictive maintenance. System agnostic approaches which just utilize sensor and operational time series have gained popularity due to its ease of implementation. Due to the nature of measurement or degradation mechanisms, its accurate estimation is not always feasible. Existing methods suppose the range of RUL with feasible estimation is given from results at upstream tasks or prior knowledge. In this work, we propose the novel framework of end-to-end learning for RUL estimation, which is called RULENet. RULENet simultaneously optimizes its Dual-estimator for RUL estimation and its feasible range estimation. Experimental results on NASA C-MAPSS benchmark data show the superiority of the end-to-end framework.**

***Index Terms*—remaining useful life estimation, deep learning**

I. Introduction

Remaining Useful Life (RUL) estimation is central to Prognostics and Health Management [1], which aims to enhance the effective reliability and availability of a physical system in-service by detecting current and approaching failures and by providing mitigation of the system risks. Its application areas span a wide range from sub-systems to components, e.g., airplane engines [2], wind turbine [3], lithium-ion batteries [4], milling machine cutting tools [5], ball screws [6] and equipment bearings [7].

RUL of a sub-system or a component is defined as the length from the current time to the end of its useful life and it can be used to characterize its current health status [8]. The end of the useful life is not limited by happening of failure. It can be the time when significant degradation which requires maintenance actions is observed through Condition-based maintenance management. The drawback of Condition-based maintenance management is that maintenances cannot be planned in advance [9]. However it forms basis to move forward to Predictive maintenance.

RUL is vital information for Predictive maintenance, which requires such information to plan maintenances in advance [9], [10], given system or component conditions. Predictive maintenance effectively increases system's availability, efficiency, reliability, productivity and safety since maintenance actions are taken only when necessary. Because of its reasonability, Predictive maintenance and its related technologies have attracted not only industries but also academias [1], [7].

The typical pipeline for RUL estimation consists of four steps, namely data acquisition, Health Index (HI) construction, health stage division and RUL estimation [7]. At first, sensors are deployed to monitor the health condition of a target system. Next, HI is constructed based upon the sensor readings and other attributes such as operational settings, in order to represent the health condition. Then, the whole lifetime is divided into two or more different health stages if applicable. Finally, the RUL estimation model is constructed based upon the HI especially for the health stage with consistent degradation trends, which may not be noticeable from raw representation of sensor readings.

Although there are various approaches to form the pipeline, it can be categorized into two large groups [1], [7], [8], i.e., first principal dependent approaches and system agnostic approaches. Their combination falls into first principal dependent approaches. First principal dependent approaches rely on prior knowledge about the failure or degradation process, which is driven by first principals such as physics and chemistry. It is also called another name such as model-based approaches or domain knowledge based approaches. System agnostic approaches rely on available data and problem setting and extract feature representation for RUL estimation from the available data based upon the problem setting using machine learning technologies. It is also called another name such as data-driven approaches.

First principal dependent approaches are preferable if applicable since they are well supported by mechanisms of failures, thus tend to be accurate. However it is impossible or infeasible to apply the approaches to complex systems.

With the advancement of sensing technologies and Information and Communications Technology (ICT), we become to be able to observe situations or status of systems more deeply in feasible ways. Meanwhile, the advancement of machine

978-1-7281-6286-7/20/$31.00 ©2020 IEEE

learning technologies has made us to achieve our goal in various tasks without significant work for the supervision when sufficient amount of data is available. The areas include image, video, speech and natural language processing as well as time series data processing [11], [12], [13], [14], [15]. Besides, it is easy to implement. As a result, system agnostic approaches have been gaining popularity.

Recently, deep neural networks have attracted a lot of research and industrial interests since they outperform other methods in various tasks [13], [14], [16], [17], [18]. In RUL estimation, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks, which is a variant of recurrent neural network have been especially used, in order to take temporal information into account [2], [19], [20], [21].

However they requires prior knowledge about a change point from the unpredictable health stage to the predictable health stage for each sequence in training data. This drawback makes implementing this series of solutions difficult. Engineers need another technology or prior knowledge to identify the change points. In general, HI is not available without modeling and it is not obvious from raw sensor readings. Hence the gap between identification of the change points and RUL estimation modeling makes the implementation costly and inaccurate in practice.

To address the above issue, we propose the novel framework of end-to-end learning for RUL estimation, which is called RULENet. RULENet simultaneously optimizes its Dual-estimator for RUL estimation and the change point estimate from the unpredictable health stage to the predictable health stage.

The rest of this paper is organized as follows. Section II introduces preliminaries and problem setting. Section III describes RULENet in detail. Section IV shows experimental results on the NASA C-MAPSS benchmark data. Section V discusses possible extension. Section VI concludes the paper.

II. Preliminaries and Problem Definition

In this section, we introduce RUL estimation with a deep neural network followed by piece-wise RUL, which plays an important role in system agnostic approaches for RUL estimation, and then describe the problem setting.

A. RUL estimation with a deep neural network

In RUL estimation, deep neural networks directly map given observation to RUL. In this framework, RUL is used as HI in the typical pipeline for RUL estimation. A deep neural network usually consists of two components [2], [19]. One is a converter from a time series set to a vector, named Tss2Vec, and the other is a converter from a vector to HI, named Vec2HI. In [19], CNN is used as Tss2Vec, and a single layer perceptron is used as Vec2HI. In [2], LSTM is used as Tss2Vec, and a Multi-Layer Perceptron (MLP) is used as Vec2HI.

Let $\boldsymbol{x} \in \mathbb{R}^{L \times A}$ be the time series for RUL estimation, where L denotes the length of the time series, A denotes the number of attributes, e.g., sensor readings or operational settings. Let

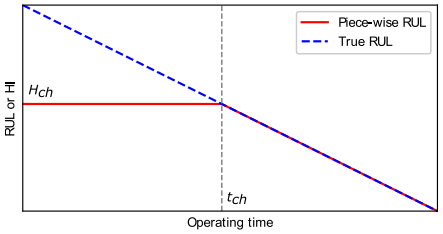


Fig. 1: Piece-wise RUL.

$H \in \mathbb{R}$ be HI at the end of the time series data $\boldsymbol{x}$. Let $RUL \in \mathbb{R}$ be RUL of H. Let $\mathcal{F}$ be Tss2Vec. Let $\mathcal{G}$ be Vec2HI.

Feature representation $\boldsymbol{v}$ from the time series data $\boldsymbol{x}$ is given as:

$$\boldsymbol{v} = \mathcal{F}(\boldsymbol{x}). \tag{1}$$

Then RUL is given as:

$$RUL = H = \mathcal{G}(\boldsymbol{v}). \tag{2}$$

B. Piece-wise RUL

RUL linearly decreases over time, but HI does not always [7] due to nonlinearity in degradation process. It is especially severe when the degradation process has multi-stages. There is a typical scenario in practice, i.e., healthy stage followed by degradation stage. At the beginning of use, degradation in a system can be negligible and HI doesn't change. Approaching to the end of life, the degradation becomes severe and HI starts to change. In addition, there are limitations to get true HI due to several reasons, e.g., modeling error and observational error.

In order to deal with this scenario, a piece-wise RUL function was proposed in [22] and used in other researches, [2], [19], [23], which is shown in Fig. 1, which limits the maximum RUL to a constant value. Let $H_{ch} \in \mathbb{R}$ be the constant value indicating change point in the slope of HI. Then RUL is given as:

$$RUL = \min(H, H_{ch}). \tag{3}$$

This definition makes it different from true RUL. It means:

$$RUL = \begin{cases} H, & H < H_{ch} \quad (4a) \\ \text{uncertain}, & \text{otherwise} \quad (4b) \end{cases}$$

and the feasible range of RUL estimation is defined by H_{ch}.

C. Problem Setting

Our main goal is to learn feature representation from multivariate time series data for RUL estimation without taking H_{ch} as an input and accurately estimate RUL based upon the representation.

Let $\boldsymbol{X}^{(k)} \in \mathbb{R}^{l_k \times A}$ be the kth example from K run-to-failure data X, where $X^{(k,j)}$ represents observation at the failure. Here j denotes the time index in run-to-failure data whose smaller value indicates older records, l_k denotes the length of the time series. Note that subsequence of $\boldsymbol{X}^{(k)}$ can be $\boldsymbol{x}$.

The problem we aim to study is to get a mapping function g from a sequence until the end of useful life $\boldsymbol{X}$ to H_{ch} based upon $\boldsymbol{X}^{(k)}$ and a mapping function f from a subsequence $\boldsymbol{x}$ to H:

$$H = f(\boldsymbol{x}), \tag{5}$$

$$H_{ch} = g(\boldsymbol{X}), \tag{6}$$

and then estimate RUL with (4a) and (4b).

III. RULENet

In this section, we elaborate our proposal, RULENet in detail. At first, we introduce its entire architecture, named the Dual-estimator followed by each component, and then we introduce its training procedure including its loss function.

A. The Dual-estimator

The Dual-estimator consists of five components: Tss2Vec, Tss2Mat, Vec2HI, Mat2HIch and HIch2HI as shown in Fig. 2. The Dual-estimator uses all components at training and uses Tss2Vec and Vec2HI at inference.

Let $\boldsymbol{x}^{(k,j)}$ be a subsequence of $\boldsymbol{X}^{(k)}$ ending at jth time index, $\boldsymbol{v}^{(k,j)} \in \mathbb{R}^{N\times 1}$ be its feature representation, and $\boldsymbol{V}^{(k)} = [\boldsymbol{v}^{(k,1)}, \boldsymbol{v}^{(k,2)}, \ldots, \boldsymbol{v}^{(k,l_k)}] \in \mathbb{R}^{N\times l_k}$ be feature representation for the entire $\boldsymbol{X}^{(k)}$. Note that the beginning of $\boldsymbol{x}^{(k,j)}$ can be at the beginning of $\boldsymbol{X}^{(k)}$ but not necessary.

Given $\boldsymbol{x}^{(k,j)}$, Tss2Vec outputs its feature representation $\boldsymbol{v}^{(k,j)}$, and Vec2HI gives RUL estimation at j as $H_1^{(k,j)}$ based upon $\boldsymbol{v}^{(k,j)}$.

Given kth example of run-to-failure data $\boldsymbol{X}^{(k)}$, Tss2Mat outputs $\boldsymbol{V}^{(k)}$ and then Mat2HIch converts $\boldsymbol{V}^{(k)}$ to the change point of HI $H_{ch}^{(k)}$. Finally, HIch2HI gives RUL estimation at j as $H_2^{(k,j)}$ based upon $H_{ch}^{(k)}$.

Tss2Vec and Tss2Mat are identical except for their input and output. Tss2Vec takes a subsequence $\boldsymbol{x}^{(k,j)}$ and gives a vector $\boldsymbol{v}^{(k,j)}$. Tss2Mat just iterates the process for all j and concatenates all the vectors to be a matrix. The intuition behind this design is better feature representation works not only for RUL estimation but also for the feasible range estimation. It also helps to reduce search space of parameters.

At training, the Dual-estimator takes an example of run-to-failure data $\boldsymbol{X}^{(k)}$ and its subsequence $\boldsymbol{x}^{(k,j)}$ as its input at a time, and then it gives two values of RUL estimation: $H_1^{(k,j)}$ and $H_2^{(k,j)}$.

At inference, it takes time series $\boldsymbol{x}$ as the input, then it outputs H, which is HI at the end of the time series data $\boldsymbol{x}$.

B. Mat2HIch

In principal, Mat2HIch can be any function mapping a matrix to a single value, but it might be inappropriate for RUL estimation since the number of examples for training is limited due to its nature. Restricting search space is a promising candidate. We propose to use a weighted average of true RUL, where the weights are determined based upon $\boldsymbol{v}^{(k,j)}$ by a neural network, so called attention mechanism, and must sum to one, independent of the length of sequence. It enforces the solution bounds to the maximum true RUL in the training examples to zero and to be represented by the weight average. Let $R^{(k,j)}$ be true RUL at jth time index in kth run-to-failure and $a^{(k,j)}$ be the weight, then:

$$H_{ch}^{k} = \sum_{j=1}^{l_k} a^{(k,j)} R^{(k,j)}. \tag{7}$$

In this study, the gated attention mechanism [24] is employed to capture complex relationships among feature representations. The gated attention mechanism gives the weights as:

$$a^{(k,j)} = \frac{\exp\left\{\mathbf{w}\left(\tanh\left(\mathbf{P}\mathbf{v}^{(k,j)}\right) \odot \mathrm{sigm}\left(\mathbf{Q}\mathbf{v}^{(k,j)}\right)\right)\right\}}{\sum_{i=1}^{l_k} \exp\left\{\mathbf{w}\left(\tanh\left(\mathbf{P}\mathbf{v}^{(k,i)}\right) \odot \mathrm{sigm}\left(\mathbf{Q}\mathbf{v}^{(k,i)}\right)\right)\right\}}, \tag{8}$$

where $\boldsymbol{w} \in \mathbb{R}^{1\times M}$, $\boldsymbol{P} \in \mathbb{R}^{M\times N}$ and $\boldsymbol{Q} \in \mathbb{R}^{M\times N}$ are parameters, $\odot$ is element-wise multiplication, $\mathrm{sigm}(\cdot)$ is sigmoid function and M is the number of elements in w.

C. HIch2HI

The most simple HIch2HI is the piece-wise RUL function. However it might have a potential problem since the gradient is zero for the region with H_{ch}. In order to deal with the potential problem, we propose to give negative slope for the region. We name the function leaky truncated RUL function $\mathcal{H}(R)$ which is similar to leaky ReLU [25] that yields:

$$\mathcal{H}(x) = \min(y_1(R), y_2(R)), \tag{9}$$

where:

$$y_1(R) = -a_1(-R + H_{ch}) + H_{ch}, \tag{10}$$

$$y_2(R) = -a_2(-R + H_{ch}) + H_{ch}, \tag{11}$$

where a_1 and a_2 are parameters and $a_2 \ll a_1$. When $a_2 = 0$, it is identical with the piece-wise RUL function. In this study we set $a_1 = 1.0$ and $a_2 = 0.01$.

D. Training

The objective of the training is to find optimal values for parameters including hyper-parameters so that they give the best performance at inference.

The Dual-estimator gives two value of RUL estimation at the time index j: $H_1^{(k,j)}$ and $H_2^{(k,j)}$. Although they are from different approaches, they must be similar each other. That's why minimization of difference between $H_1^{(k,j)}$ and $H_2^{(k,j)}$ is one of the objective and the RMSE loss L_b is given as:

$$L_b = \sum_{\langle j,k\rangle} \left(H_1^{(k,j)} - H_2^{(k,j)}\right)^2, \tag{12}$$

where $\langle j,k\rangle$ represents a pair of the indices in the training data.

The $H_1^{(k,j)}$ is arbitrary but $H_2^{(k,j)}$ is not. The $H_2^{(k,j)}$ is constraint by the leaky truncated RUL function, which means there is only one parameter to change its value over kth example of run-to-failure data. In addition, $H_2^{(k,j)}$ must be more accurate since Mat2HIch and HIch2HI can observe entire

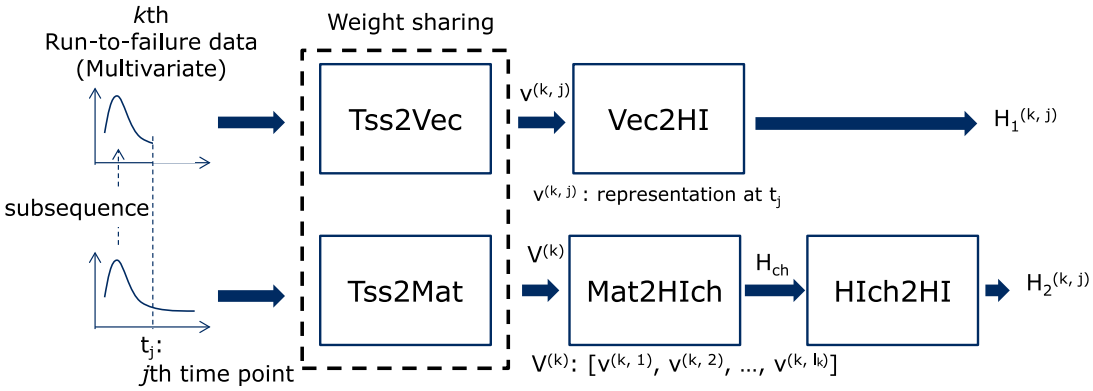


Fig. 2: Architecture of the Dual-estimator.

run-to-failure data and it includes future information from the perspective of the pipeline with Tss2Vec and Vec2HI. Due to these mechanisms, $H_2^{(k,j)}$ plays as a teacher for Tss2Vec and Vec2HI. Eventually, the pipeline with Tss2Vec and Vec2HI gives accurate RUL estimation without future information.

However it is inappropriate just to minimize the difference due to the trivial solution, which is $H_{ch}^{(k)} = 0$. When $H_{ch}^{(k)} = 0$, $H_1^{(k,j)}$ and $H_2^{(k,j)}$ become zero and it is easily achievable just ignoring values in input data $\boldsymbol{X}^{(k)}$. In order to avoid the trivial solution, we add a loss term, named value loss, which is given as:

$$L_v = \sum_{\langle j,k \rangle} \left(\frac{1}{H_{ch}^{(k)}} \right)^2. \tag{13}$$

Note that $H_{ch}^{(k)}$ is given for every j as well since a pair of $\boldsymbol{X}^{(k)}$ and its subsequence $\boldsymbol{x}^{(k,j)}$ is the input and a training sample. This loss enforces to have larger values of $H_{ch}^{(k)}$, and affects to the representation learning at Tss2Vec due to weight sharing. This means better feature representation should give not only better RUL estimation but also better change point of HI. Finally the loss function L is given as:

$$L = L_b + \lambda L_v, \tag{14}$$

where λ is a hyper-parameter.

Available data for training is divided into training data and validation data. The model is trained with the mini-batch gradient descent and its parameters are optimized with every training sample for the number of iteration, so called epochs. For each epoch, a performance metric is computed for the validation data. If the performance metric is better than that from previous epoch, the newer parameters are kept as the best parameters at the moment. In order to determine the best hyper-parameters, a cross-validation method is employed.

The performance metric should be designed for each use case based upon its requirements. In typical cases, there is a range of interest for RUL estimation since no action is needed at sufficient RUL and no action can be taken at quite short RUL.

Let $h_{est}^{(i)}$ be an estimated RUL, and $h_{true}^{(i)}$ be the true RUL. Given the range as r_{min} to r_{max} and estimation error $d^{(i)}$, then the performance metric P is given as:

$$P = \sum_{i \in D} \mathcal{P}(d^{(i)}), \tag{15}$$

where:

$$d^{(i)} = h_{est}^{(i)} - h_{true}^{(i)}, \tag{16}$$

$$D = \{i \in \mathbb{N} | r_{min} \leq r_i \leq r_{max}\}, \tag{17}$$

and $\mathcal{P}(\cdot)$ is a function to compute performance metric for each estimation.

IV. Experiments

In the experiments, we aim to evaluate the proposed framework, RULENet. We evaluate RULENet on NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) benchmark data [26], which is a benchmark dataset widely used in works related to RUL estimation. We compare the performance of RULENet to a variety of state-of-the-art with deep learning approaches: the CNN based ('CNN') in [19], the LSTM based ('Deep LSTM') in [2], and the bootstrapping based LSTM ('LSTMBS') in [21], as well as other approaches: Support Vector Regression ('SVR') in [19] and Relevance Vector Regression ('RVR') in [19]. As shown by the experimental results, the proposed approach, RULENet significantly outperforms all these alternatives.

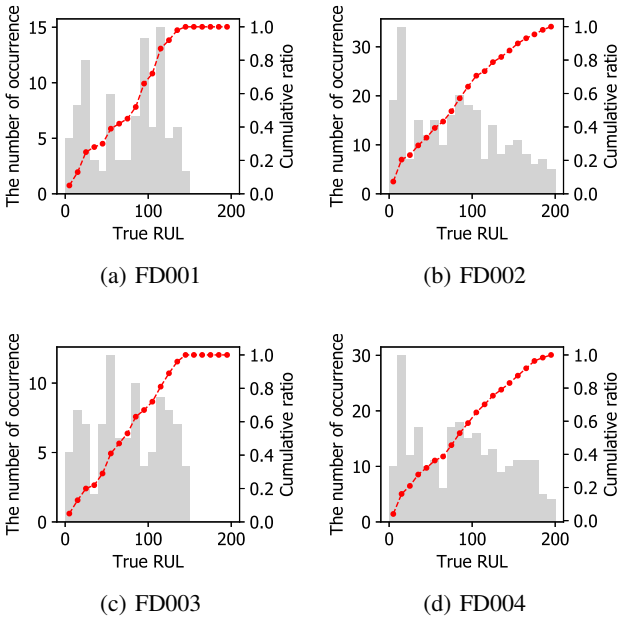
A. C-MAPSS dataset

C-MAPSS dataset is about run-to-failure data from a fleet of turbofan engines of the same type. The run-to-failure data has 24 attributes for each engine, namely three operational settings and 21 sensors. It contains four subsets of run-to-failure data. They are simulated data at various operational conditions and fault modes, and each dataset consists of training data and testing data, as shown in Table I. The training data is run-to-failure data. The testing data is pairs of time series data and RUL at the end of the time series data.

Distribution of RUL at the end of test trajectories is shown in Fig. 3. They almost equally distribute over ranges from the minimum to the maximum value. RUL with higher than H_{ch} is unpredictable. Specifically, when H_{ch} is fixed to 130, the percentages of samples whose RUL at the end of test

TABLE I: Summary on C-MAPSS dataset.

Property \ Dataest	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	248
Test trajectories	100	259	100	249
Operational settings	1	6	1	6
Fault modes	1	1	2	2
Min. RUL at test	7	6	6	6
Max. RUL at test	145	194	145	195



(a) FD001 (b) FD002

(c) FD003 (d) FD004

Fig. 3: Distribution of RUL at the end of test trajectories. Each grey box shows the number of occurrence within its range. The red lines show cumulative ratio from the minimum to the maximum value.

trajectories are larger than H_{ch} are 8%, 21%, 11% and 24% for FD001, FD002, FD003 and FD004, respectively.

B. Evaluation metrics

We evaluate the performance of RULENet with the same evaluation metrics used in [2], [19], [21], i.e., RMSE and the Score.

Let I be the number of examples for testing, then RMSE is defined with estimation error $d^{(i)}$ by:

$$RMSE = \sqrt{\frac{1}{I}\sum_{i=1}^{I}(d^{(i)})^2}, \tag{18}$$

and the Score S is defined by:

$$S = \sum_{i=1}^{I}\mathcal{F}_s(d^{(i)}), \tag{19}$$

where:

$$\mathcal{F}_s(d^{(i)}) = \max\left\{\exp\left(-\frac{d^{(i)}}{13}\right), \exp\left(\frac{d^{(i)}}{10}\right)\right\} - 1. \tag{20}$$

C. Preprocessing

In order to compare results with other published works and to be consistent with [2], the same preprocessing is employed. As for unspecified preprocessing, the same preprocessing in [19] is employed.

In order to take the operating conditions into account, one-hot encoding representation of operating conditions is added to input time series data. K-means clustering algorithm is employed to time series of the three operational settings in order to cluster the data FD002 and FD004 into six clusters. They are well clustered for each operating condition [2].

In order to fairly treat all sensors across all operating conditions, the time series of 21 sensors are normalized with z-score normalization for each cluster. Let $x^{(i,j)}$ be a value of ith sensor at jth time index, $\mu^{(c_j,i)}$ and $\sigma^{(c_j,i)}$ respectively be the mean and standard deviation in c_jth operating condition, then normalized value $x_n^{(i,j)}$ is given as:

$$x_n^{(i,j)} = \frac{x^{(i,j)} - \mu^{(c_j,i)}}{\sigma^{(c_j,i)}}. \tag{21}$$

D. Model

In this study, a stacked LSTM is used as Tss2Vec and Tss2Mat, and a MLP is used as Vec2HI, since it is evaluated for various benchmark datasets and significantly outperforms traditional approaches for RUL estimation [2]. The same values in [2] are set to corresponding hyper-parameters, i.e., the number of LSTM layers is two, the number of nodes in the 1st LSTM Cell is 32, the number of nodes in the 2nd LSTM Cell is 64, the number of MLP layers is two, the number of nodes in both the 1st layer and the 2nd layer is eight. The rest hyper-parameters are optimized through cross-validation. As for the training, RMSprop is used, which is also same. At the training, 80% of available data for training is used as training data and the rest is used as validation data.

Since the Score is more sensitive to a large estimation error than RMSE, the Score is used as the performance metric and $\mathcal{F}_s(d^{(i)})$ is used as $\mathcal{P}(d^{(i)})$. Here, the smaller value of this metric, the better.

In this study, we set zero to r_{min} for all datasets and 130 to r_{max} of FD001 and FD003 since it is traditionally used for H_{ch} and only 10% of test trajectories have greater value than H_{ch} at the end of trajectories. However, the percentages are larger than 20% for FD002 and FD004. In order to cover majority of them, we set 190 to r_{max} of FD002 and FD004.

E. Results

The RMSE and Score of our RULENet together with those from past methods are summarized in Table II and Table III. Here the best performance is indicated with the bold characters. Specifically, the performance is compared to that with Deep LSTM since they have identical structure at inference. It indicates improvement by the feature representation learning strategy but not the network structure for RUL estimation. The improvement over Deep LSTM IMP are calculated by:

$$IMP = 1 - \frac{p_{rule}}{p_{lstm}}, \tag{22}$$

TABLE II: RMSE comparison on C-MAPSS dataset and improvement IMP of RULENet over Deep LSTM.

Method \ Dataset	FD001	FD002	FD003	FD004
MLP	37.56	80.03	37.39	77.37
SVR	20.96	42.00	21.05	45.35
RVR	23.80	31.30	22.37	34.34
CNN	18.45	30.29	19.82	29.16
LSTMBS	14.89	26.86	15.11	27.11
Deep LSTM	16.14	24.49	16.18	28.17
RULENet	**13.96**	**22.19**	**14.79**	**25.41**
Oracle	2.40	16.31	3.19	17.35
IMP	13.53%	9.40%	8.62%	9.81%

TABLE III: Score comparison on C-MAPSS dataset and improvement IMP of RULENet over Deep LSTM.

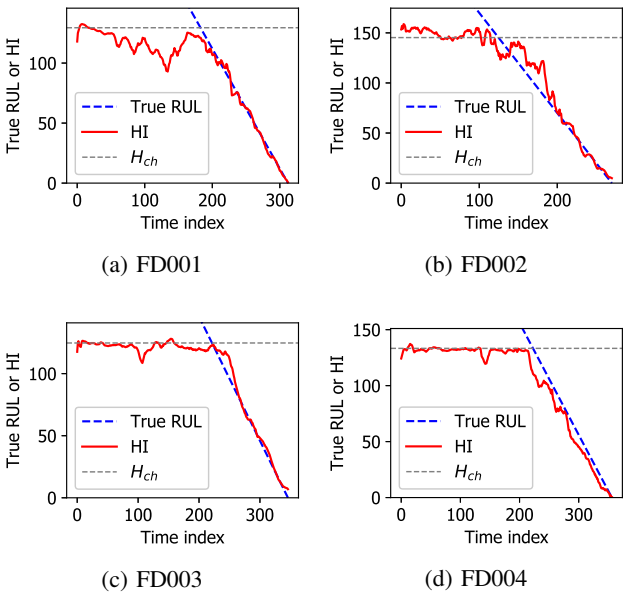
Method	Dataset FD001	FD002	FD003	FD004
MLP	1.8×10^4	7.8×10^6	1.7×10^4	5.6×10^6
SVR	1.4×10^3	5.9×10^6	1.6×10^3	3.7×10^5
RVR	1.5×10^3	1.7×10^4	1.4×10^3	2.7×10^4
CNN	1.3×10^3	1.4×10^4	1.6×10^3	7.9×10^3
LSTMBS	4.8×10^2	8.0×10^3	4.9×10^2	5.2×10^3
Deep LSTM	3.4×10^2	4.5×10^3	8.5×10^2	5.6×10^3
RULENet	$\mathbf{3.1 \times 10^2}$	$\mathbf{3.9 \times 10^3}$	$\mathbf{3.1 \times 10^2}$	$\mathbf{3.8 \times 10^3}$
Oracle	7.0×10^0	1.4×10^3	1.1×10^1	1.4×10^3
IMP	9.40%	14.23%	63.63%	32.96%

where p_{rule} denotes a metric value of RULENet, and p_{lstm} denotes a metric value of Deep LSTM. In past researches, H_{ch} is set to 130. Meanwhile, maximum RUL in testing data is greater than that. That's why perfect RUL estimation based upon piece-wise RUL function with the parameter value, 130, can not be zero. For reference, the evaluation metric values from the perfect RUL estimation are shown together as Oracle.

As shown in Table II and III, RULENet outperforms the past methods in both metrics for all the four datasets. The average improvements over four datasets are respectively 10% in RMSE and 30% in the Score.

Trajectories of RUL estimation are visualized together with their true RUL in order to visually inspect whether they have two slopes in their trend and the estimation fits true RUL around the end of useful life as designed. An example of the trajectory estimated with validation data is selected for each dataset and is shown in Fig. 4. They show the trajectories have two slopes in their trend as designed. Meanwhile, tendency to have sudden change in HI is observed at the beginning of the sequence.

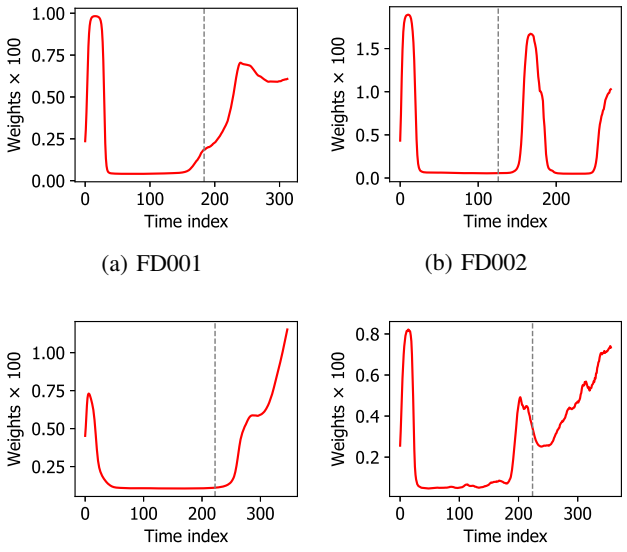
Given the current design of Mat2HIch, there are two scenarios to find the best H_{ch}. One is to have significantly high values in the attention weights around H_{ch}. The other is to have hight weights in the unpredictable stage and the predictable stage. The first scenario indicates the feature vectors around H_{ch} capture information to identify the boundary. This is the same scenario with change point detection tasks. The second scenario indicates both feature representation about the unpredictable stage and the predictable stage are important



(a) FD001 (b) FD002

(c) FD003 (d) FD004

Fig. 4: Example trajectories of RUL estimation in validation data. HI is computed with all data until the corresponding time point. H_{ch} is computed with all data in each example.



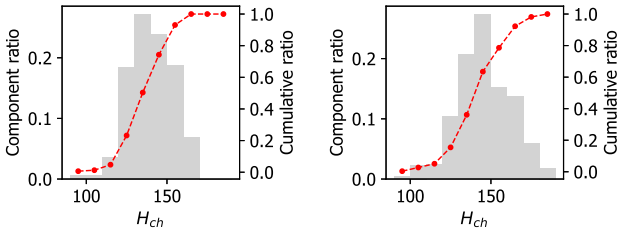
(a) FD001 (b) FD002

(c) FD003 (d) FD004

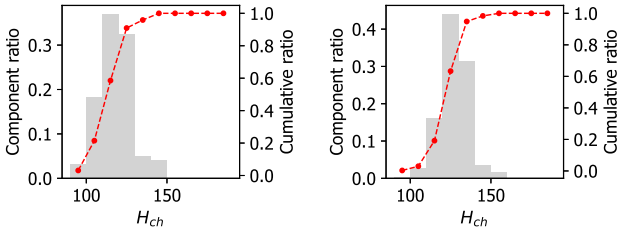
Fig. 5: Corresponding attention weights on the example trajectories in Fig. 4. Each grey dashed line indicates time point of H_{ch}.

to identify H_{ch}. This is the same scenario with classification tasks. Fig. 5 shows corresponding attention weights on the trajectories in Fig. 4. In all examples, high weight values distribute at the beginning of the run-to-failure data and after H_{ch}. They indicate RULENet learns H_{ch} following the second scenario.

Distribution of H_{ch} in each dataset is shown in Fig. 6. Their medians are 140.7, 145.8, 118.3 and 127.0, for FD001, FD002, FD003 and FD004, respectively. They are similar to H_{ch} in

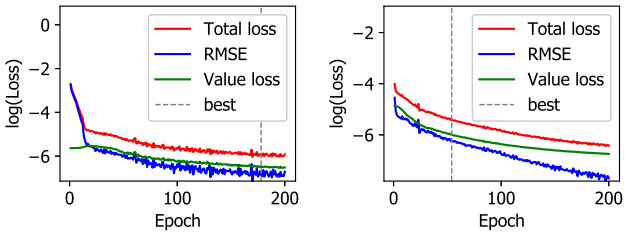


(a) FD001 (b) FD002

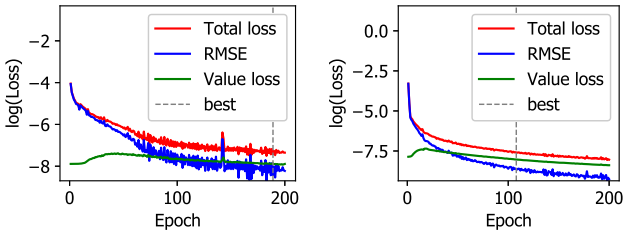


(c) FD003 (d) FD004

Fig. 6: Distribution of H_{ch}. Each grey box shows the component ratio of occurrence within its range. The red lines show cumulative ratio from the minimum to the maximum value.

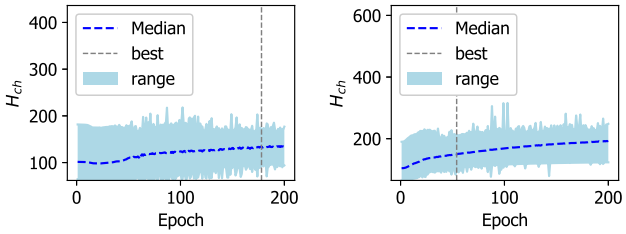


(a) FD001 (b) FD002

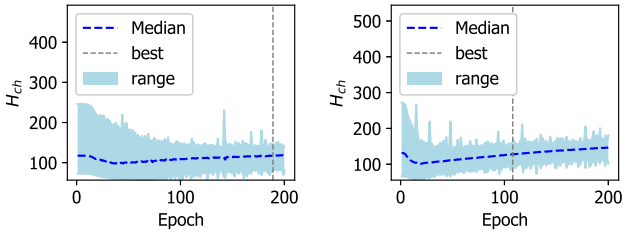


(c) FD003 (d) FD004

Fig. 7: Developments of losses over training. Each grey dashed line indicates each epoch with the best performance at validation data.



(a) FD001 (b) FD002



(c) FD003 (d) FD004

Fig. 8: Developments of distribution of H_{ch} over training. Each grey dashed line indicates each epoch with the best performance at validation data.

past researches, which is 130.

The H_{ch} in FD002 and FD004 have wider range of distribution and smaller H_{ch} than those of FD001 and FD003. FD002 and FD004 contain two fault modes, but FD001 and FD003 contain one fault mode. It implies multiple fault modes make RUL estimation at early degradation difficult. Meanwhile, they all are unimodal distribution. It implies the two fault modes give similar degradation in terms of feasibility of RUL estimation. On the other hand, the distribution is similar between FD001 and FD002, and between FD003 and FD004. It means the number of operational settings makes weak impacts to feasibility of RUL estimation.

In order to understand learning process of RULENet, learning dynamics are investigated through this experiment, especially for checking if overfitting is avoided via validation, understanding the relationship between RMSE loss and value loss in the loss function, and checking how distribution of H_{ch} changes during training.

Developments of losses over training are shown in Fig. 7. These figures indicate overfitting is avoided through the training procedure since the best models are not given by the end of training although the total loss become smaller and smaller over training.

Developments of distribution of H_{ch} over training are shown in Fig. 8. They show the ranges of H_{ch} become smaller and smaller over training and their mean values become higher and higher while their RMSE losses are decreasing. The first phenomenon indicates more accurate estimation of H_{ch} gives smaller difference between estimation value with the different ways as designed. The second phenomenon indicates difficulty to ignore changes in input time series since RULENet prefers longer period with larger slope.

V. Discussion

In this paper, we followed the same setting in the past methods. While we have seen the clear superiority of the end-to-end framework through experiments, there are spaces for further improvements.

We generated feature representation for all time points in training data. It means some feature representation is not generated with sufficient information. Especially at the beginning of a sequence, it doesn't contain any temporal information. It

may harm models. In order to guarantee minimum amount of information, we may introduce minimum sequence length to generate feature representation.

We used time series data from the past to current time to generate a feature vector. However this setting may not work or not be efficient for training when irrelevant but significant change exists in the middle of the subsequence. In order to eliminate undesirable effects from the past, we may employ sliding window manner [23] to extract subsequences for Tss2Vec and Tss2Mat.

We used stacked LSTM and MLP for RUL estimation at inference. They have relatively simple structures. We may use more complex models [13] to capture more complex information in data.

We may use different preprocessing especially for treatment of operating conditions so that the explicit mapping from operational settings to operating conditions is not necessary.

VI. Conclusion

In this work, we have presented an end-to-end framework for Remaining Useful Life estimation using deep learning technology and shown its advantage simultaneously optimizing RUL estimation and its feasible range estimation. Our experiments on C-MAPSS dataset have shown it outperforms past approaches in RUL estimation. We have also identified area for further improvements of this framework. We hope the proposed framework makes complex physical system management more efficient and reliable.

References

[1] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou, "Remaining useful life estimation – a review on the statistical data driven approaches," *European Journal of Operational Research*, vol. 213, no. 1, pp. 1 – 14, 2011.

[2] S. Zheng, K. Ristovski, A. Farahat, and C. Gupta, "Long short-term memory network for remaining useful life estimation," in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2017, pp. 88–95.

[3] M. Djeziri, S. Benmoussa, and R. Sanchez, "Hybrid method for remaining useful life prediction in wind turbine systems," *Renewable Energy*, vol. 116, pp. 173 – 187, 2018, real-time monitoring, prognosis and resilient control for wind energy systems.

[4] M. Li, M. Sadoughi, S. Shen, and C. Hu, "Remaining useful life prediction of lithium-ion batteries using multi-model gaussian process," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2019, pp. 1–6.

[5] Y. Liu, Y. Chang, S. Liu, and S. Chen, "Data-driven prognostics of remaining useful life for milling machine cutting tools," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2019, pp. 1–5.

[6] M. Benker, R. Kleinwort, and M. F. Zäh, "Estimating remaining useful life of machine tool ball screws via probabilistic classification," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2019, pp. 1–7.

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

[8] Chen Xiongzi, Yu Jinsong, Tang Diyin, and Wang Yingxun, "Remaining useful life prognostic estimation for aircraft subsystems or components: A review," in *IEEE 2011 10th International Conference on Electronic Measurement Instruments*, vol. 2, Aug 2011, pp. 94–98.

[9] G. A. Susto, A. Beghi, and C. De Luca, "A predictive maintenance system for epitaxy processes based on filtering and prediction techniques," *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 4, pp. 638–649, Nov 2012.

[10] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine learning for predictive maintenance: A multiple classifier approach," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 812–820, June 2015.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, p. 1527–1554, Jul. 2006.

[12] S. Zheng, A. Vishnu, and C. Ding, "Accelerating deep learning with shrinkage and recall," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2016, pp. 963–970.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 5998–6008.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

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

[16] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, ser. IJCAI'17. AAAI Press, 2017, p. 2627–2633.

[17] D. Song, N. Xia, W. Cheng, H. Chen, and D. Tao, "Deep r -th root of rank supervised joint binary embedding for multivariate time series retrieval," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2229–2238.

[18] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[19] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *Database Systems for Advanced Applications*, S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang, and H. Xiong, Eds. Cham: Springer International Publishing, 2016, pp. 214–228.

[20] K. Aggarwal, O. Atan, A. K. Farahat, C. Zhang, K. Ristovski, and C. Gupta, "Two birds with one network: Unifying failure event prediction and time-to-failure modeling," in *2018 IEEE International Conference on Big Data (Big Data)*, Dec 2018, pp. 1308–1317.

[21] Y. Liao, L. Zhang, and C. Liu, "Uncertainty prediction of remaining useful life using long short-term memory network based on bootstrap method," in *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2018, pp. 1–8.

[22] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *2008 International Conference on Prognostics and Health Management*, Oct 2008, pp. 1–6.

[23] Q. Wang, S. Zheng, A. Farahat, S. Serita, and C. Gupta, "Remaining useful life estimation using functional data analysis," in *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, June 2019, pp. 1–8.

[24] M. Ilse, J. Tomczak, and M. Welling, "Attention-based deep multiple instance learning," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 2127–2136.

[25] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[26] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set," NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA, 2008, accessed: 2020-01-22.