Long Short-Term Memory Network for Remaining Useful Life Estimation

Shuai Zheng
Department of CSE,
University of Texas at Arlington, TX
Email: zhengs123@gmail.com

Kosta Ristovski and Ahmed Farahat and Chetan Gupta
Big Data Laboratory, Hitachi America Ltd., Santa Clara, CA
Email: firstname.lastname@hal.hitachi.com

Abstract—**Remaining Useful Life (RUL) of a component or a system is defined as the length from the current time to the end of the useful life. Accurate RUL estimation plays a critical role in Prognostics and Health Management(PHM). Data driven approaches for RUL estimation use sensor data and operational data to estimate RUL. Traditional regression based approaches and recent Convolutional Neural Network (CNN) approach use features created from sliding windows to build models. However, sequence information is not fully considered in these approaches. Sequence learning models such as Hidden Markov Models (HMMs) and Recurrent Neural Networks (RNNs) have flaws when modeling sequence information. HMMs are limited to discrete hidden states and are known to have issues when modeling long-term dependencies in the data. RNNs also have issues with long-term dependencies. In this work, we propose a Long Short-Term Memory (LSTM) approach for RUL estimation, which can make full use of the sensor sequence information and expose hidden patterns within sensor data with multiple operating conditions, fault and degradation models. Extensive experiments using three widely adopted Prognostics and Health Management data sets show that LSTM for RUL estimation significantly outperforms traditional approaches for RUL estimation as well as Convolutional Neural Network (CNN).**

I. Introduction

Remaining Useful Life (RUL) of a component or a system is defined as the length from the current time to the end of the useful life [1]. The criteria to define whether the component or system is still usable is already known to the domain experts of the component or system. In industry operational research, there is an interest in modeling methods for RUL estimation given component or system condition and health monitoring information [1]. Modeling methods of RUL estimation can be categorized into model-based approaches and data-driven approaches. Model-based approaches build physical failure models for degradation, such as crack, wear, corrosion, etc. Physical models are very useful to solve RUL problem in use-cases where there is no enough failure data available. In such cases, we can induce failures within physical models, augment the actual data with physical model failure data and learn models for RUL estimation. However, such physical failure models are very complex and difficult to build, and physical models for many components do not exist. On the other hand for equipment with enough failures, data-driven approaches which use sensor data and operational condition data to estimate RUL are preferable and cost effective. In many industrial applications, this data is relatively easy to collect which makes data-driven approaches feasible to provide accurate RUL estimation.

Accurate RUL estimation plays a critical role in Prognostics and Health Management(PHM). For example, for turbofan engine, if we know the accurate Remaining Useful Life (RUL), we can schedule next maintenance in advance and ship new parts early to ensure smooth replacement and maintenance; we can also reduce cost by canceling unnecessary maintenance; we can also adjust operating conditions, such as speed, load, to prolong the life of the engine. RUL estimation is very important in many industry areas, including aircraft industries, medical equipment, power plants, mining, etc. Unfortunately, in some applications, inaccurate RUL estimation may cause catastrophic damage to the production.

Sensor data are time series in nature. To estimate RUL accurately, we need to build models to capture time sequence information in the data. Sliding window approaches and sequence learning Hidden Markov Model (HMM) and Recurrent Neural Network (RNN) were applied in this area.

In [2], [3], series of time sequence sensor data are segmented into sliding windows. Each sliding window is assigned a target RUL value. RUL estimation model is then built based on sliding windows. Time sequence information is contained in each sliding window and larger is the time window, the more time sequence information is contained. On the other hand, time window can not be too large because of model complexity and overfitting. Also, sequence information is not fully exploited. The constraint between any two sliding windows is not enforced; since RUL decreases along with time, for two consecutive sliding windows, a window that comes first in time should have larger RUL than the window that comes later. There are time dependency between sliding windows, but sliding windows considered are independent in these traditional approaches [2], [3].

From the perspective of sequence learning, Hidden Markov Model (HMM) approaches for prognostics are proposed in [4]. However, due to the limitations of HMM [5], HMM hidden states must be drawn from a modestly sized discrete state space. HMM computational complexity and storage makes HMM unfeasible when the set of possible hidden stages grows large. Further, each hidden state can only depend on the immediately previous state. Markov models have been shown

978-1-5090-5710-8/17/$31.00 ©2017 IEEE

to be computationally impractical for modeling long-range dependencies [6].

Recurrent Neural Network (RNN) [7] can model time sequence data. Some work [8] applied RNN for RUL estimation. However, RNN is known to have long-term time dependency problems [9], where the gradients propagated over many stages tend to either vanish or explode.

Recently, deep learning, such as Convolutional Neural Network (CNN), Long Short-Term Memory Network (LSTM), has been shown to be efficient in applications such as computer vision, image, video, speech and natural language processing [10], [11], [12]. A CNN based RUL estimation approach is proposed in [13], where the sensor data is split into sliding windows and then CNN is applied on each sliding window. In this case, CNN also considers each sliding window independently.

Long Short-Term Memory Network (LSTM) [11] is a type of RNN network for sequence learning tasks and has achieved great success on speech recognition and machine translation. LSTM does not have long-term time dependency problems by controlling information flow using input gate, forget gate and output gate. Remembering information for long periods of time is practically their default behavior. Due to inherent sequential nature of sensor data, LSTMs are well-suited for RUL estimation using sensor data. In this work, we propose a LSTM based approach for RUL estimation, which uses multiple layers of LSTM cells in combination with standard feed forward layers to discover hidden patterns from sensor and operational data with multiple operating conditions, fault and degradation models.

The rest of the paper is organized as follows. Section II introduces LSTM model, data preparation process and evaluation criteria; Section III shows experiment results on three real PHM data sets; Section IV discusses extension and future work of applying LSTM for RUL estimation; Section V concludes the paper.

II. LSTM Network for RUL Estimation

In this section, we present the LSTM network for RUL estimation, including the structure of the network, the information flow within a LSTM cells, and how the sensor data is fed into the LSTM network. We will also discuss the cost function to optimize LSTM network.

A. LSTM Model

Assume that we have N components of the same type with run to failure data or run to any other event that describes component end of life. Each component is part of the equipment that provides a set of multivariate time-series of sensor data. Assume that there are p sensors of the same type on each equipment. Then data collected from each equipment can be represented in a matrix form $X_n = [\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^t, ..., \mathbf{x}^{T_n}] \in \mathbb{R}^{p\times T_n} (n = 1, ..., N)$ where T_n is time of the failure and $\mathbf{x}^t = [x_1^t, ..., x_p^t] \in \mathbb{R}^{p\times 1}, t = 1, 2, ..., T_n$ is p-dimensional vector of sensor measurements at time t $(t = 1, 2, ..., T_n)$. Also we denote $\mathbf{x}_i = [x_i^1, x_i^2, ..., x_i^{T_n}] \in \mathbb{R}^{1\times T} (i = 1, 2, ..., p)$ to be a time sequence of i-th sensor measurements. LSTM model takes each sequence of sensor measurements X_n and learns how model the whole sequence with respect to target RUL. This sequence learning setup is very similar to HMMs. In practice, observed sequence of sensor measurements is supplied to the network and RUL is obtained. Note that sequences can be of different length. Proposed unfolded LSTM structure for RUL estimation is shown in Figure 1.

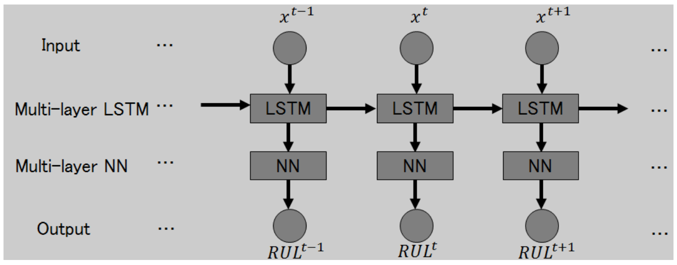


Fig. 1: Deep LSTM model for RUL estimation.

At time t, LSTM model takes the p-dimensional sensor data $\mathbf{x}^t$ as an input data and outputs estimated RUL_{est}^t. Model is composed of multiple layers of LSTMs, which are then followed by multiple layers of feed forward neural networks (NNs). Each layer of LSTM, which contains chain of repeating cells as indicated in Figure 1, is defined by cell structure and number of nodes within the cell. Repeating cells within LSTM layer have the same structure and parameter values. Proposed structure is adequate for RUL estimation due to the fact it utilizes the complementarity modeling capability of LSTM and NN. More precisely, LSTM is good at temporal modeling, while fully connected NN is good at mapping LSTM feature to regression outcomes. Since the machine usually has multiple operating conditions, failure modes, and degradation modes, this structure uses multiple layers and multiple nodes in each layer to discover complex relationship within the sensor data.

LSTM cell structure at time t is shown in Figure 2 [14]. We differentiate output of LSTM cell and cell state denoted as $\mathbf{h}_t$ and $\mathbf{c}_t$, respectively. Vector size of the output and cell state is the same and it is defined by number of nodes in the cell. Let denote m as number of nodes, then output of this LSTM cell is $\mathbf{h}^t \in \mathbb{R}^{m\times 1}$ and cell state is $\mathbf{c}^t \in \mathbb{R}^{m\times 1}$. The output $\mathbf{h}^{t-1}$ and hidden state $\mathbf{c}^{t-1}$ of LSTM cell at time $t-1$ will serve as input of LSTM at time t which is indicated in Figure 2. This LSTM cell also takes sensor data $\mathbf{x}^t$ as an input. There are three gates that control the information flow within cell: (1) input gate $\mathbf{i}^t \in \mathbb{R}^{m\times 1}$ controls what information based on output $\mathbf{h}^{t-1}$ and sensor measurements $\mathbf{x}^t$ will be passed to memory cell, (2) output gate $\mathbf{o}^t \in \mathbb{R}^{m\times 1}$ controls what information will be carried to the next time step, and (3) forget gate $\mathbf{f}^t \in \mathbb{R}^{m\times 1}$ controls how memory cell will be updated as shown in Figure 2. In this work, all LSTM cells that are used in the models are implemented as follows:

$$\mathbf{i}^t = \sigma(W_i\mathbf{x}^t + U_i\mathbf{h}^{t-1} + \mathbf{b}_i), \quad (1)$$
$$\mathbf{o}^t = \sigma(W_o\mathbf{x}^t + U_o\mathbf{h}^{t-1} + \mathbf{b}_o), \quad (2)$$
$$\mathbf{f}^t = \sigma(W_f\mathbf{x}^t + U_f\mathbf{h}^{t-1} + \mathbf{b}_f), \quad (3)$$
$$\mathbf{a}^t = tanh(W_c\mathbf{x}^t + U_c\mathbf{h}^{t-1} + \mathbf{b}_c), \quad (4)$$
$$\mathbf{c}^t = \mathbf{f}^t \circ \mathbf{c}^{t-1} + \mathbf{i}^t \circ \mathbf{a}^t, \quad (5)$$
$$\mathbf{h}^t = \mathbf{o}^t \circ tanh(\mathbf{c}^t), \quad (6)$$

where variable weights and bias to be computed during training process are $W_i, W_o, W_f, W_c \in \mathbb{R}^{m\times p}$, $U_i, U_o, U_f, U_c \in \mathbb{R}^{m\times m}$, $\mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_f, \mathbf{b}_c \in \mathbb{R}^{m\times 1}$. $\circ$ is element-wise multiplication of two vectors (Hadamard product). σ is element-wise logistic sigmoid activation function. $tanh$ is element-wise hyperbolic tangent activation function. There are many variants of LSTMs [14] and experimentation on this is part of our future work. For example, the nonlinear sigmoid and hyperbolic tangent activation function can be replaced using other activation functions. One can also choose to use the cell state $\mathbf{c}^{t-1}$ as an extra input into the three gates. Connection between different layers of LSTMs in Figure 2 is achieved such that the output of one layer is as an input to the next layer. Sensor measurements are input only to the first LSTM layer.

B. *Learning LSTM*

The objective of the learning is to find optimal parameters (weights and biases) such that objective function is minimized. Let RUL_{est}^t be the estimated RUL at time t. Let RUL_{calc}^t be the target RUL at time t, which is usually calculated as difference between current time and end-of-life. The proposed LSTM model minimizes the following objective function:

$$J = \sum_t \|(RUL_{est}^t - RUL_{calc}^t\|^2, \quad (7)$$

with respect to W_i, W_o, W_f, W_c, U_i, U_o, U_f, U_c, $\mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_f, \mathbf{b}_c$. Algorithms to optimize Eq.(7) can be found in [14]. We use RMSprop optimizer to train models. We also use dropout and L2 regularization to control model overfitting. We are early stopping in the training process when there is no improvement on the validation data set. In addition, to avoid explosion of gradients, which causes bad fit, and to incorporate different workloads and operating conditions, data supplied to the network should be prepared appropriately. The amount of sensor data can be very huge. Though there exist some work using dimensionality reduction [15], [16], [17], [18] and cloud computing [19], [20], [21] for large data, deep learning can take the benefit of big data and performs better in big data era using GPU for parallel computing.

C. *Data Preparation*

In real applications, many raw sensor data, operating parameters, loads, and failure times will be given. Because the value scale of different sensors may vary, sensor data need to be normalized with respect to each sensor before training and testing. Also, for many systems where health of a system does not linearly degrade from the beginning of operations, we need to limit the calculated RUL_{calc}^t target using piece-wise functions. Some systems have different workloads, operating conditions, deterioration modes, and if available, we should integrate these information in the RUL estimation model.

1) Data Normalization: There are several different ways to normalize sensor data. Let μ_i be the mean i-th sensor data from all equipments, σ_i be the corresponding standard deviation. Z-score normalization is given in Eq.(8), where $\mathbf{x}'_i$ is the normalized sensor data. Min-Max normalization in Eq.(9) scales the raw sensor data within the range of $[0, 1]$.

z-score normalization

$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \mu_i}{\sigma_i} \quad (8)$$

min-max normalization

$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \min \mathbf{x}_i}{\max \mathbf{x}_i - \min \mathbf{x}_i} \quad (9)$$

2) Workloads, Operating Conditions, Deterioration Modes: When the system has different workloads, operating conditions, deterioration modes, we should integrate these information into RUL estimation model. For example, if workloads are measured by continuous numbers, we can take workloads as an extra sensor data. When there are different operating conditions, we can use one-hot encoding to convert operating conditions.

D. *Evaluation*

In order to evaluate the performance of a RUL estimation model on the test data, a *scoring* function is given to measure the quality of the models [8]. Eq.(10) shows the definition of scoring function.

$$S = \begin{cases} \sum_{i=1}^n (e^{-\frac{h_i}{13}} - 1), \text{when } h_i < 0 \\ \sum_{i=1}^n (e^{\frac{h_i}{10}} - 1), \text{when } h_i \geq 0, \end{cases} \quad (10)$$

where n is total number of samples in the test set, $h_i = RUL_{est,i} - RUL_i$, RUL_i is true RUL for the test sample i. Eq.(10) gives different penalty when the model underestimates RUL and when the model overestimates RUL. If estimated

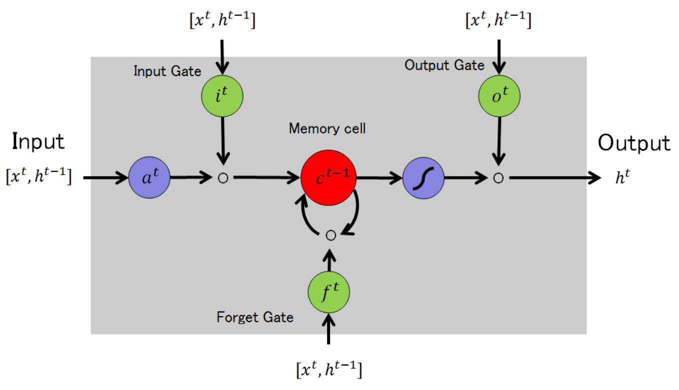


Fig. 2: LSTM Cell

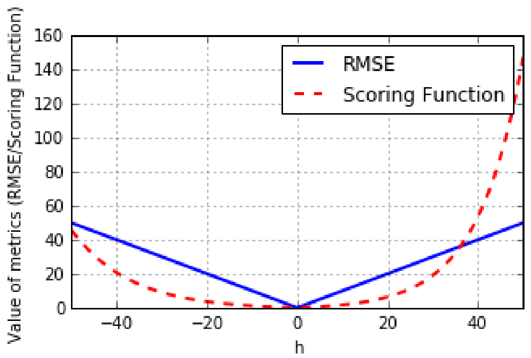


Fig. 3: Comparison of Scoring Function and RMSE ($h = RUL' - RUL$, RUL' is estimated RUL, RUL is true RUL)

RUL is less than the true RUL, the penalty is smaller, because there is still time to conduct maintenance and it will not cause significant system failure. If estimated RUL is larger than true RUL, the penalty is larger, because under such estimation, the maintenance will be scheduled later than the required time and it may cause system failure.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} h_i^2} \tag{11}$$

Eq.(11) is *Root Mean Square Error* (RMSE), which is also widely used as an evaluation metric for RUL estimation. RMSE gives equal penalty weights to the model when the estimated RUL is smaller than true RUL and when the estimated RUL is larger than true RUL.

Figure 3 shows the difference of Scoring function and RMSE. $h = 0$ means estimated RUL is the same as true RUL. For both Scoring function and RMSE, the smaller value, the better the result is.

III. Experiments

In experiments, we apply LSTM model for RUL estimation on three data sets: C-MAPSS Data Set, PHM08 Challenge Data Set and Milling Data Set. We compare LSTM model with other approaches including Multi-Layer Perceptron (MLP), Support Vector Regression (SVR), Relevance Vector Regression (RVR) and Convolutional Neural Network (CNN) [13]. We also visualize LSTM features and use an experiment to show the importance of sequence information.

A. *C-MAPSS Data Set*

NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) data set (Turbofan Engine Degradation Simulation Data Set) is a widely used benchmark data[22]. C-MAPSS data includes sensor data with different number of operating conditions and fault conditions.

TABLE I: C-MAPSS Data Set

Data Set	FD001	FD002	FD003	FD004
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

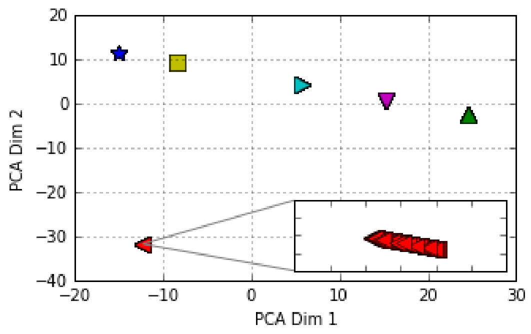


Fig. 4: Operating conditions of FD002 are clustered into 6 clusters.

1) Data: This data has 4 sub-data sets with different number of operating conditions and fault conditions and each sub-data set is further divided into training and test subsets, as shown in Table I. Each row in the data is a snapshot of data taken during a single operating time cycle, which includes 26 columns: 1st column represents engine ID, 2nd column represents the current operational cycle number, 3-5 columns are the three operational settings that have a substantial effect on engine performance, 6-26 columns represent the 21 sensor values. More information about the 21 sensors can be found in [22]. The engine is operating normally at the start of each time series, and develops a fault at some point in time which is unknown. In the training set, the fault grows in magnitude until a system failure. In the test set, data is provided up to some time prior to system failure. The goal is to estimate the number of remaining operational cycles before failure on the test data. In order to compare results with other published work and to be consistent [13], we apply z-score normalization (Eq.(8)).

2) Operating Conditions: It has been noted that the operating setting values can be clustered into distinct clusters[13]. We use K-means to cluster data FD002 and FD004 into 6 clusters, since it is given that there are 6 operating conditions. To visualize the clustering result, we use PCA to project 53759 3-dimensional FD002 operating setting values into 2D PCA space and plot different clusters using different markers in Figure 4. As we can see, there are clearly 6 clusters and samples of the same operating conditions are overlapped.

3) RUL Target Function: In traditional way of assigning target values for RUL estimation, RUL decreases linearly along with time. This definition implies that the health of a system degrades linearly along with time. In practical applications, degradation of a component is negligible at the begin-

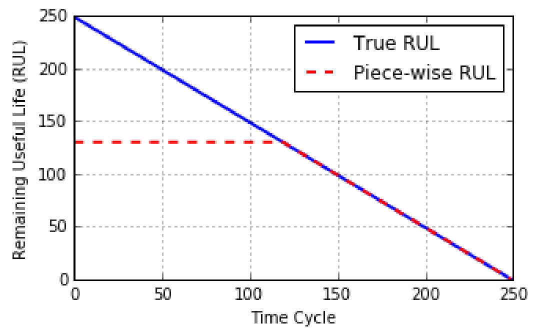


Fig. 5: Piece-wise RUL of C-MAPSS Data Set (Piece-wise maximum RUL is 130 time cycles)

TABLE II: Score and RMSE of different network architectures 10-fold cross validation (best values are in bold).

Network	FD001		FD002	
	Score	RMSE	Score	RMSE
L(32.32)N(8.8)	7.59	5.608	5.31×10^4	57.77
L(64.64)N(8.8)	**5.78**	**4.597**	7.75×10^4	72.24
L(96.96)N(8.8)	7.06	5.573	2.51×10^4	55.81
L(64.64.64)N(8.8)	9.22	6.421	6.01×10^4	70.29
L(32.64)N(16.16)	8.63	6.044	$\mathbf{1.62 \times 10^4}$	**28.77**
L(32.64)N(8.8)	7.04	5.335	1.89×10^4	31.89

ning of use, and increases when component approaches end-of-Life. To better model the Remaining Useful Life changes along with time, in [8] [13], a piece-wise linear RUL target function was proposed, as in Figure 5, which limits the maximum RUL to a constant value and then start linear degradation after a certain degree of usage. For C-MAPSS Datasets, we set the maximum limit as 130 time cycles [8] [13].

4) Network Architecture for C-MAPSS: To choose the best network architecture for the C-MAPSS data, we performed 10-fold cross validation on training data to estimate performance of models that use different network architectures. We change number of nodes, layers and report results on FD001 and FD002 in Table II. We report the mean score and RMSE of 10-fold cross validation for each of selected structures. Notation L(32.32)N(8.8) refers to network that has 4 hidden layers with 32 nodes in the first LSTM layer, 32 nodes in the second LSTM layer, 8 and 8 nodes in the third and the forth layer where standard feed forward neural network was used. Finally there is a 1-dimensional output layer. For different data sets, the best network architecture would be different. Interestingly, Network L(32.64)N(8.8) is the second best in both data sets, so we decided to use it in all the following C-MAPSS data experiments.

Our applied LSTM model was able to discover hidden features (patterns) from the sensor data. Figure 6 shows sensor measurements and discovered features by the second LSTM layer on FD004 dataset. FD004 dataset has 6 operating conditions and 2 fault conditions, which makes the sensor data complex and difficult to find patterns directly in the sensor data. Figure 6a shows two sample sensor data from

TABLE III: RMSE comparison on C-MAPSS data (Improvement of Deep LSTM over CNN, $IMP = 1 - DeepLSTM/CNN$).

Data Set	FD001	FD002	FD003	FD004
MLP[13]	37.56	80.03	37.39	77.37
SVR[13]	20.96	42.00	21.05	45.35
RVR[13]	23.80	31.30	22.37	34.34
CNN[13]	18.45	30.29	19.82	29.16
Deep LSTM	**16.14**	**24.49**	**16.18**	**28.17**
IMP	12.52%	19.15%	18.37%	3.40%

TABLE IV: Score comparison on C-MAPSS data (Improvement of Deep LSTM over CNN, $IMP = 1 - DeepLSTM/CNN$).

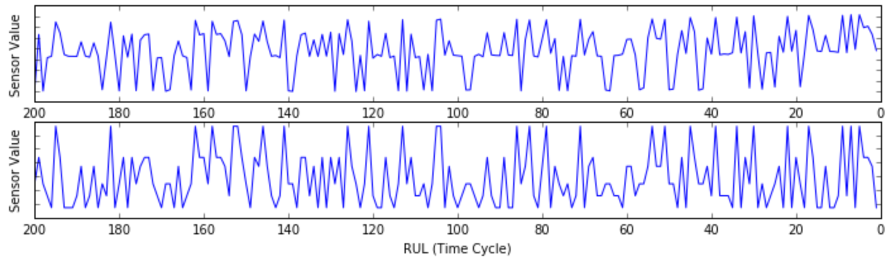
Data Set	FD001	FD002	FD003	FD004
MLP[13]	1.80×10^4	7.80×10^6	1.74×10^4	5.62×10^6
SVR[13]	1.38×10^3	5.90×10^5	1.60×10^3	3.71×10^5
RVR[13]	1.50×10^3	1.74×10^4	1.43×10^3	2.65×10^4
CNN[13]	1.29×10^3	1.36×10^4	1.60×10^3	7.89×10^3
Deep LSTM	$\mathbf{3.38 \times 10^2}$	$\mathbf{4.45 \times 10^3}$	$\mathbf{8.52 \times 10^2}$	$\mathbf{5.55 \times 10^3}$
IMP	73.80%	67.28%	46.75%	29.66%

$RUL_{calc} = 200$ to $RUL_{calc} = 0$. There are no clear trends in the sensor data because the operating conditions are changing and fault conditions vary from time to time and from trajectory to trajectory. The output of a LSTM layer is the multiplication of a $tanh$ function and a sigmoid function, so the output values range in $[-1, 1]$. That is why each column of Figure 6b has 64 values ranging in $[-1, 1]$. Figure 6b has 200 columns in total representing the time cycles from $RUL_{calc} = 200$ to $RUL_{calc} = 0$. We can observe trending patterns with respect to RUL_{calc} in Figure 6b which can be some indicator of health. Figure 6 tells us that using two layer LSTM our model was able to discover the hidden patterns from sensor data with multiple operating conditions and fault conditions on this dataset.

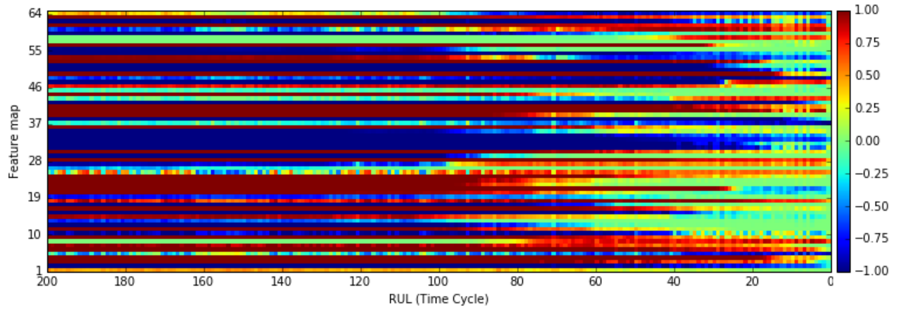
Since learning of LSTM is a non-convex optimization problem, we run LSTM 5 times and record structure with parameters corresponding to the best result on validation set. We then applied the learned structure to the test data. Table III and Table IV show the result of RMSE and Score. We compare LSTM results with other reported results [13]. LSTM outperforms all other approaches, and using the scoring function, LSTM performs much better than CNN.

5) Importance of Sequence Length: In this part, we use an experiment to show that when the testing sequence is shorter, the estimated RUL will have larger RMSE and Score values, which indicates that longer sequences contain more information and sequence length affects accuracy for RUL estimation.

Figure 7 takes one testing trajectory as example from each of the 4 C-MAPSS sub data set and shows the estimated RUL using LSTM at any time step of test sequence X. For example, we use sequence $X^k = [\mathbf{x}^1, ..., \mathbf{x}^t, ..., \mathbf{x}^{T-k}]$ to estimate RUL at time $T - k$ and compare to calculated piece-wise labels using true RUL given in the test set at time T and lag value



(a) Sample sensor data (200 time cycles, from $RUL = 200$ to $RUL = 0$)



(b) 2nd layer LSTM feature (64 rows, 200 time cycles(columns), from $RUL = 200$ to $RUL = 0$)

Fig. 6: LSTM feature visualization of data FD004.

k. We can find that, in Figure 7, when it is close to the end of the sequence, the estimated RUL is more accurate and close to the piece-wise RUL, because LSTM model uses longer testing sequence at the end of the sequence as well as it is easier to predict in near future than over long horizon.

To quantify observations from Figure 7 we have created Figure 8 which shows the RMSE and Mean Score of all testing trajectories when we shorten the testing sequence length by $k = 1, 2, ..., 50$. It is clear that as k increases, both RMSE and Mean Score increase. This indicates that sequence information is important in RUL estimation, and using shorter sequence for RUL estimation will result in higher RMSE and Score. Also it indicates that reliability engineers should trust accuracy of LSTM model when there is a significant degradation from healthy condition of the component.

B. PHM08 Challenge Data Set

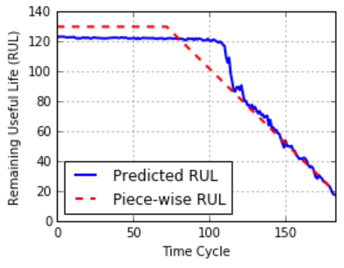
PHM08 Challenge Data Set [23] is similar to C-MAPSS data except the true RUL values are not revealed. PHM08 Challenge Data Set has 218 training trajectories and 218 testing trajectories, with 6 operating conditions and 2 fault conditions. Results need to be uploaded to NASA Data Repository website and a single score (Eq.(10)) was then calculated by the website as the final output. Table V shows the score results. The data preparation process and Deep LSTM model structure is the same as C-MAPSS data experiment. As we can see, Deep LSTM gives the best results compared to other approaches.

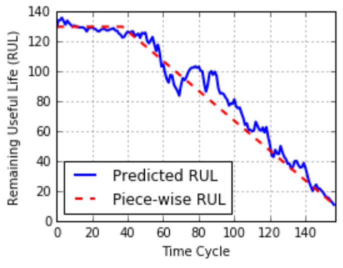
C. Milling Data Set

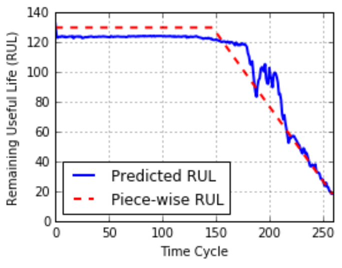
Milling data set [24] contains experiment data from runs on a milling machine under various operating conditions. Flank wear is measured and used as an indicator of the health status of the milling machine. This data has 3 operating parameters (Depth of Cut, Feed and material type), and 6 sensors (acoustic emission sensor at 2 different positions, table and spindle; vibration sensor at 2 different positions, table and spindle; AC spindle motor current sensor and DC spindle motor current sensor). There are 16 cases with varying number of runs for each case. The number of runs was dependent on the degree of flank wear that was measured between runs. In our experiment, when the flank wear is more than 0.45 for the first time, we set the Remaining Useful Life as 0, which is consistent with other publications [25] [26]. There are 167 runs across 16 cases, with 109 runs for material type 1 and 58 runs for material type 2. Case number 6 of material 2 has only one run, and thus is not considered for experiments [25]. In our experiment, we use the last 2 cases of material type 1 (case 11 and case 12) and last 2 cases of material type 2 (case 15 and case 16) as testing samples. The remaining 11 cases are used as training samples. Each run has a fixed length of 3000 time series sensor data. Each time point is represented using a 9-dimensional vector (3 operating parameters and 6 sensors). In each run, there

TABLE V: Score comparison on PHM08 Challenge Data Set (Improvement of Deep LSTM over CNN, $IMP = 1 - DeepLSTM/CNN$).

MLP[13]	3212
SVR[13]	15886
RVR[13]	8242
CNN[13]	2056
Deep LSTM	**1862**
IMP	9.44%


(a) FD001 example


(b) FD002 example


(c) FD003 example

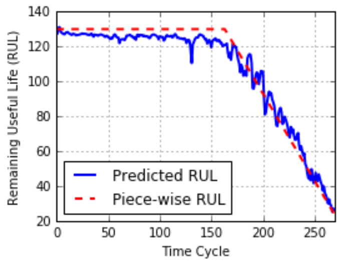
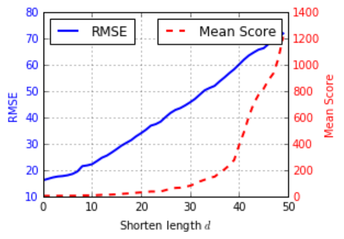
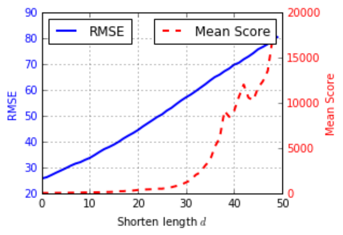
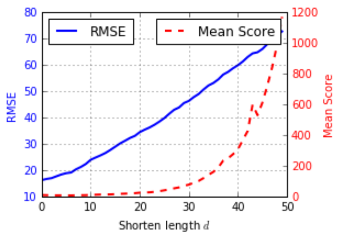

(d) FD004 example

Fig. 7: RUL estimation at each time stamp using Deep LSTM (use sequence $X = [\mathbf{x}^1, ..., \mathbf{x}^t, ..., \mathbf{x}^{T-1}]$ to estimate RUL at time $T-1$, with the true RUL as $RUL+1$; use sequence $X = [\mathbf{x}^1, ..., \mathbf{x}^t, ..., \mathbf{x}^{T-2}]$ to estimate RUL at time $T-2$, with the true RUL as $RUL+2$, etc..)


(a) FD001


(b) FD002


(c) FD003

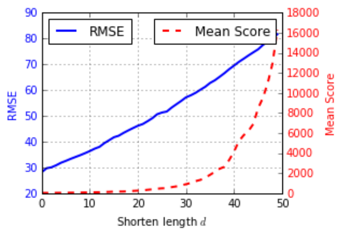

(d) FD004

Fig. 8: RMSE and Mean Score with respect to shorten length d using Deep LSTM (when $d = 1$, we use use sequence $X = [\mathbf{x}^1, ..., \mathbf{x}^t, ..., \mathbf{x}^{T-1}]$ to estimate RUL at time $T-1$, with the true RUL as $RUL+1$; when $d = 2$, we use use sequence $X = [\mathbf{x}^1, ..., \mathbf{x}^t, ..., \mathbf{x}^{T-2}]$ to estimate RUL at time $T-2$, with the true RUL as $RUL+2$, etc..)

are three stages: machine starting stage, operating stage and terminating stage. We undersample the operating stage data at rate $1/3$. The maximum length of processed Milling Data Set is 226 (which has similar length as C-MAPSS data). This data is then normalized using Eq.(8).

For evaluation, we use Root Mean Square Error (RMSE) of Eq.(11). For MLP, SVR, RVR, we use sliding window of length 10. MLP is a neural network with number of nodes $90-4-4-1$. CNN uses sliding window of length 15, with one convolution layer of 16 nodes with kernel filter size 4×1, one average pooling layer with pooling filter size 2×1, and then a $4-1$ neural network. The LSTM model has two LSTM layers with 32 nodes and 64 nodes and then a $8-8-1$ neural network. We tried several different network structures for MLP, CNN and LSTM, and found these were the best structures. Table VI shows the result of RMSE on Milling Data Set using MLP, SVR, RVR, CNN and Deep LSTM. Deep LSTM gives the best and minimum RMSE compared to other approaches.

IV. Discussion and Future Work

In this work we have presented new deep learning technology for remaining useful life estimation. We have seen clear benefits of using it, but we have also identified space for further improvements in following areas: automatic piece-wise function detection, different LSTM structures, different optimization functions, high frequency sensor data, online learning of LSTM networks.

TABLE VI: RMSE comparison on Milling Data Set (Improvement of Deep LSTM over CNN, $IMP = 1 - DeepLSTM/CNN$).

MLP	6.26
SVR	7.35
RVR	17.22
CNN	6.15
Deep LSTM	**2.80**
IMP	54.47%

A. Automatic piece-wise function detection

As we have seen in the experiments with C-MAPSS, using piece-wise functions for creating the labels provide better accuracy. Current approach is to limit the maximum value of RUL, but this approach is very generic and does not take into account degradation of individual components. In our future work we will implement detection degradation point for each of the sequences and assign labels accordingly.

B. Cell structure and optimization

In the current implementation we have used standard setting for cell structure and optimization objective. There are other proposed structures of LSTM cells which were not examined in this work. Also optimization function for model learning (7) penalizes errors for the overestimating and underestimating RUL equally. Also estimations which are far away from end-of-life are treated equally as the estimations which are closer to

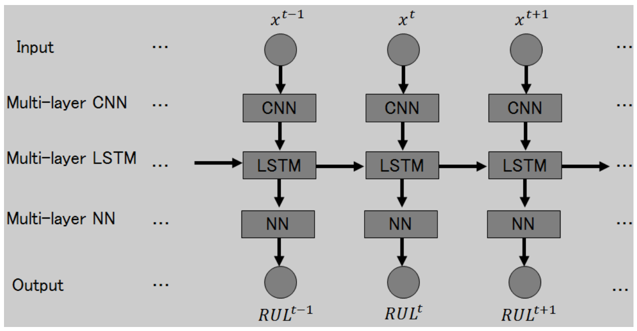


Fig. 9: Deep CNN+LSTM model for RUL estimation.

end-of-life. In our future work we are planning to investigate temporal weighting within cost function.

C. Extension to Fine-Grained High Frequency Sensor Data

When the sensor data is fine-grained, it will cost a lot of computing resource for the model in Figure 1. If there is high frequency sensor measurements or large amount of sensors involved, there would be benefit of adding a Deep CNN layer before the LSTM layer, as in Figure 9. CNN serves the purpose of reducing frequency and noise, LSTM can create temporal features, and fully connected Neural Network layers can learn a good regression model.

D. Online Learning LSTM

Sensor data are time series and the amount of sensor data increases along with time. Because of the nature of LSTM model, it can be re-trained using only newly available sensor data which can benefit many industrial applications.

V. Conclusion

We proposed deep learning approach for Remaining Useful Life (RUL) estimation and we showed its benefits by taking sequence information when estimating RUL. Our approach involved sequence of layers of LSTM followed by feed forward networks. Our experiments on 3 widely used data sets, C-MAPSS Data Set, PHM08 Challenge Data Set and Milling Data Set, showed that LSTM model outperforms other approaches and gives the best performance in RUL estimation. We have also identified area for further improvements of deep learning models for RUL estimation. With expansion of Industrial IoT applications, vast amount of data will be collected and deep learning models will be extremely useful for predictive health management.

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.-j. Wu, N. Gebraeel, M. A. Lawley, and Y. Yih, “A neural network integrated decision support system for condition-based optimal predictive maintenance policy,” *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 37, no. 2, pp. 226–236, 2007.

[3] Z. Tian, “An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring,” *Journal of Intelligent Manufacturing*, vol. 23, no. 2, pp. 227–237, 2012.

[4] P. Baruah and R. B. Chinnam*, “Hmms for diagnostics and prognostics in machining processes,” *International Journal of Production Research*, vol. 43, no. 6, pp. 1275–1293, 2005.

[5] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with lstm recurrent networks,” *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[6] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.

[7] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[8] F. O. Heimes, “Recurrent neural networks for remaining useful life estimation,” in *Prognostics and Health Management, 2008. PHM 2008. International Conference on*. IEEE, 2008, pp. 1–6.

[9] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[10] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[11] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[13] G. S. Babu, P. Zhao, and X.-L. Li, “Deep convolutional neural network based regression approach for estimation of remaining useful life,” in *International Conference on Database Systems for Advanced Applications*. Springer, 2016, pp. 214–228.

[14] I. G. Y. Bengio and A. Courville, “Deep learning,” 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

[15] Y. Zhu, E. Song, J. Zhou, and Z. You, “Optimal dimensionality reduction of sensor data in multisensor estimation fusion,” *IEEE Transactions on Signal Processing*, vol. 53, no. 5, pp. 1631–1639, 2005.

[16] S. Zheng, F. Nie, C. Ding, and H. Huang, “A harmonic mean linear discriminant analysis for robust image classification,” in *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*. IEEE, 2016, pp. 402–409.

[17] S. Zheng and C. Ding, “Kernel alignment inspired linear discriminant analysis,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2014, pp. 401–416.

[18] S. Zheng, X. Cai, C. H. Ding, F. Nie, and H. Huang, “A closed form solution to multi-view low-rank regression.” in *AAAI*, 2015, pp. 1973–1979.

[19] S. Zheng, Z.-Y. Shae, X. Zhang, H. Jamjoom, and L. Fong, “Analysis and modeling of social influence in high performance computing workloads,” in *European Conference on Parallel Processing*. Springer Berlin Heidelberg, 2011, pp. 193–204.

[20] D. Williams, S. Zheng, X. Zhang, and H. Jamjoom, “Tidewatch: Fingerprinting the cyclicality of big data workloads,” in *INFOCOM, 2014 Proceedings IEEE*. IEEE, 2014, pp. 2031–2039.

[21] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, “Virtual machine migration in an over-committed cloud,” in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 196–203.

[22] E. Ramasso and A. Saxena, “Performance benchmarking and analysis of prognostic methods for cmapss datasets.” *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15, 2014.

[23] A. Saxena and K. Goebel, “Phm08 challenge data set,” *NASA Ames Prognostics Data Repository (http://ti. arc. nasa. gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA*, 2008.

[24] A. Agogino and K. Goebel, “Mill data set,” *NASA Ames Prognostics Data Repository (http://ti. arc. nasa. gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA*, 2007.

[25] P. Malhotra, V. TV, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, “Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder,” *arXiv preprint arXiv:1608.06154*, 2016.

[26] J. B. Coble, “Merging data sources to predict remaining useful life–an automated method to identify prognostic parameters,” 2010.