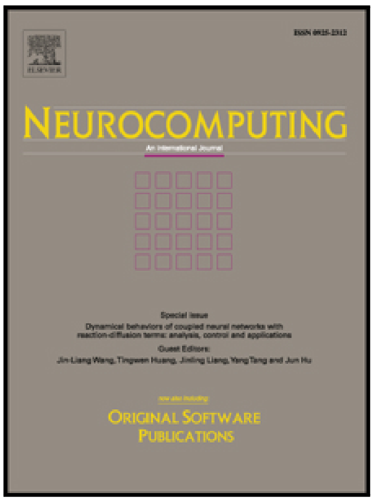


Remaining Useful Life Estimation of Engineered Systems using vanilla LSTM Neural Networks

Yuting Wu , Mei Yuan , Shaopeng Dong , Li Lin , Yingqi Liu

PII: S0925-2312(17)30950-5
DOI: 10.1016/j.neucom.2017.05.063
Reference: NEUCOM 18481

To appear in: *Neurocomputing*

Received date: 27 May 2016
Revised date: 22 December 2016
Accepted date: 26 May 2017

Highlights

- *Extract extra dynamic differential features* – Inter-frame dynamic changes contains a great amount of model degradation information. Therefore, a dynamic difference technology was used to extract new features from original datasets.
- *Higher prediction accuracy* - The vanilla LSTM can get higher prediction than the standard RNN and GRU. As the research objects become more and more complex, the prediction accuracies did not obviously decrease.
- *Advanced Regularization mechanism and optimization algorithm* - Dropout mechanism is used to improve the generalization ability of vanilla LSTM while Adam algorithm is used to reduce the effects of learning rate on the final optimization results.

*Corresponding author. Tel: +86-10-82338757 E-mail address: yuanm@buaa.edu.cn

Remaining Useful Life Estimation of Engineered Systems using vanilla LSTM Neural Networks

Yuting Wu[a, c], Mei Yuan[b, c,*], Shaopeng Dong [b], Li Lin[a, c], Yingqi Liu [b]

[a] School of Energy and Power Engineering, Beihang University, Beijing 100191, China

[b] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China

[c] Collaborative Innovation Center for Advanced Aero-Engine, Beihang University, Beijing 100191, China

ABSTRACT

Long Short-Term Memory (LSTM) networks are a significant branch of Recurrent Neural Networks (RNN), capable of learning long-term dependencies. In recent years, vanilla LSTM (a variation of original LSTM above) has become the state-of-the-art model for a variety of machine learning problems, especially Natural Language Processing (NLP). However, in industry, this powerful Deep Neural Network (DNN) has not aroused wide concern. In research focusing on Prognostics and Health Management (PHM) technology for complex engineered systems, Remaining Useful Life (RUL) estimation is one of the most challenging problems, which can lead to appropriate maintenance actions to be scheduled proactively to avoid catastrophic failures and minimize economic losses of the systems. Following that, this paper aims to propose utilizing vanilla LSTM neural networks to get good RUL prediction accuracy which makes the most of long short-term memory ability, in the cases of complicated operations, working conditions, model degradations and strong noises. In addition, to promote cognition ability about model degradation processes, a dynamic differential technology was proposed to extract inter-frame information. The whole proposition is illustrated and discussed by performing tests on a case of the health monitoring of aircraft turbofan engines which have four different issues. Performances of vanilla LSTM are benchmarked with standard RNN and Gated Recurrent Unit (GRU) LSTM. Results show the significance of performance improvement achieved by vanilla LSTM.

Keywords

Long Short-Term Memory Neural Network, Remaining Useful Life Estimation, Prognostics and Health Management, Dynamic Differential Feature

1. Introduction

In industry, the remaining useful life estimation of a system or component is usually dependent on operating conditions and sensor readings. Obviously, the more historical data available, the more accurate predictions will be. At the data competition of 1st international conference on Prognostics and Health Management (PHM08), Peel et al. used Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) networks [1] to estimate the RUL of aero-engines while Heimes et al. used classical RNN. Hermes's algorithm got better performance than Peel`s owing to the RNN`s hidden units which implicitly contain information about the history of all past elements in the sequence [2]. By virtue of weight sharing idea and feedback structure, recurrent neural network is able to use all of the historical conditions and sensing data to predict with low model complexity. Nowadays, recurrent neural network has become one of the important subfields of deep learning [3]. It has been widely used to generate sequences in domains as diverse as music [4], speech [5], text [6] and motion capture data [7].

Unfortunately, Trying to unfold recurrent connections of RNN lead us to find that RNN can be a very deep feedforward network. This is called "long-term dependencies", making it hard to learn to store information for very long [8]. In the past two decades, researches made every effort to solve this issue and proposed some variants, the most famous two are Echo State Networks (ESN) [9] and LSTM [10] respectively. On one side, since learning the recurrent and input weights is difficult, Jaeger and Haas [9] presented setting those weights such that the recurrent hidden units can capture the history of past inputs well, and only learn the output weights. This is the core of the echo state network. Echo state networks have been shown to be an effective RNN variant and achieved some success in RUL prediction problems, such as satellite lithium battery RUL estimation developed by Wang Hong et al. [11]. On the other side, the central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell. However, the original LSTM (no forget gate), proposed by Hochreiter & Schmidhuber in 1997 [10], did not perform well. The most commonly used LSTM architectures nowadays were originally introduced by Graves & Schmidhuber (2005) [12]. People refer to it as vanilla LSTM. Vanilla LSTMs have forget gate allowing learning of continual tasks, and they use full gradients training instead of setting parts of weights just like ESN. Although it was also then modeled on vanilla LSTM to create a number of typical variants, such as GRU (Cho et al., 2014) [13], Greff et al. (2015) did a nice comparison of popular variants, finding that “vanilla LSTM performs reasonably well on various datasets and using any of eight possible modifications does not significantly improve the LSTM performance.”[14]. These modifications included GRU.

Above all, the aim of this paper is to utilize vanilla LSTM, which usually deals with supervised learning on language modeling, and related state-of-the-art technologies of feature extraction to improve accuracy in RUL prediction problems for complicated industrial objects. The main contributions of this paper are as follows:

(1) *Add dynamic differential features* – Raw features in RUL estimation problem are often stationary. But inter-frame dynamic changes (observed by the sensors under different operation conditions) contain a great amount of model degradation information. Therefore, a dynamic difference technology was used to extract new features from original health monitoring datasets.

(2) *Higher prediction accuracy* - The vanilla LSTM can get higher prediction than the standard RNN and GRU under the same number of hidden neurons in a single layer. As research objects become increasingly complex, the prediction accuracies can be obtained by the model do not obviously decrease.

(3) *Advanced regularization mechanism and optimization algorithm* - Dropout mechanism is used to improve the generalization ability of vanilla LSTM while advanced optimization algorithm (Adam) is used to reduce the effects of learning rate on final optimization results.

This paper is organized as shown below. Application backgrounds of neural networks in RUL estimation of engineered systems are given in Section 2. This part illustrates advantages and drawbacks of classical neural networks when dealing with RUL estimation problems. On this basis, Section 3 proposes using vanilla LSTM to make effectively full of historical data to assess the RUL. This section also briefly introduces the main schemes of vanilla LSTM. Performances of vanilla LSTM are benchmarked by performing tests on aircraft turbofan engines datasets from NASA in Section 4. Four different issues are considered: a single fault and single operating mode problem, a single fault and multiple operating modes problem, a hybrid fault and single operating mode problem and a hybrid fault and multiple operating modes problem. Through comparisons with standard RNN and GRU LSTM, vanilla LSTM`s excellent performance in RUL estimation field is demonstrated. Finally, Section 5 concludes this work and proposes some future aspects.

2. Neural networks in RUL estimation

As one of the most important members in the field of machine learning, neural network is considered as a mature prognostic algorithm. Multilayer perceptron, radial basis function networks and other neural networks have been widely used in anomaly detection, damage clustering and fault diagnosis. Excitedly, they achieved remarkable success [1,15-17].

As for time series data, such as the samples in RUL prediction problem, researchers have been searching for more reasonable models and algorithms to tap into their potential. Traditional static neural networks (like MLP) directly consider each time point independently and create a prediction at each time step, which have to discard much information in historical data. A slightly better plan is to consider using a phase space embedding representation, in which a sequence of instances is generated using a fixed length sliding window. However, phase space embedding has the significant drawback of increasing the number of dimensions proportionally with each time step represented, giving rise to the problems associated with the "curse of dimensionality"[1]. This structure is Time Delay Neural Network (TDNN).

Recurrent neural networks address the issue above. They are networks with loops, allowing information to persist arbitrarily long period of time theoretically with a core idea in deep learning — weight sharing. Experimental results showed that more reasonable use of historical data made it a better RUL estimator with obvious performance improvement [2].

Unfortunately, training standard RNNs, just like the network in [2], has proved to be problematic because it is difficult for the gradients to backpropagate far over many time steps consistently with a reasonable range. This problem is summarized as “vanishing gradients” and “exploding gradients” intuitively [8]. The exploding gradient problem is relatively easy to address by enforcing a hard constraint over the norm of the gradients [18]. On the other hand, the vanishing gradients were preliminarily addressed by original LSTM [10].

3. Vanilla LSTM: concept and application in RUL estimation

3.1 Concept of vanilla LSTM

After refinement and popularization, the variant of LSTM, vanilla, is most commonly used in literature. The schematic of the vanilla LSTM block can be seen in Figure 1.

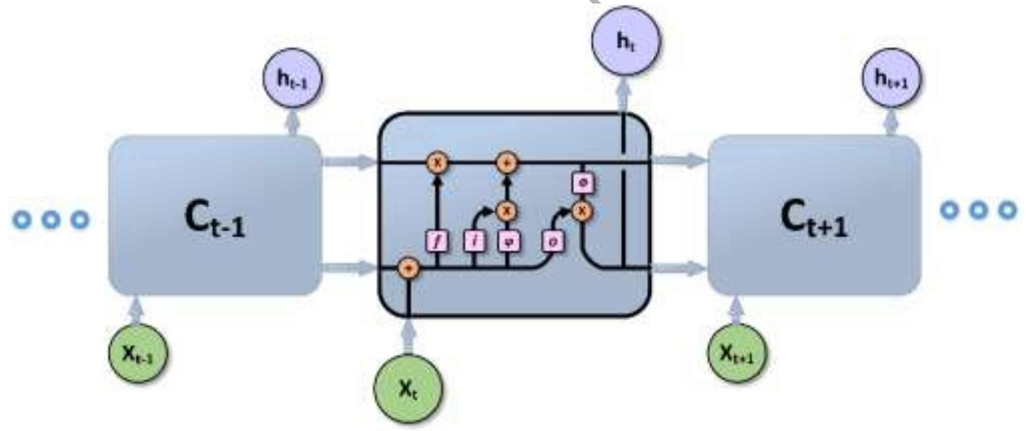


Fig.1. Vanilla LSTM

As shown in Figure 1, the core idea of LSTM lies in the information flows represented by the two black horizontal lines. The bottom one indicates the combination of input of the current time (X_t) and output of the previous time (h_{t-1}). In classical RNN, this integrated information is used for overwriting cell state directly. As for LSTM, this is used for calculating forget control signal, input control signal, candidate update value of cell state and output control signal (f_t , i_t, φ and o_t, from left to right) by a series of nonlinear functions respectively. In addition, the top one indicates the cell state of LSTM which has two branches. One of them is similar to the cell state in classical RNN except for some nonlinear transformation. And the other one is the old state at previous time.

Then, the two branches, in the control of the respective gate signal (i_t and f_t), form a new cell state. Eventually, output of LSTM at current time (h_t) is calculated by cell state through another nonlinear transformation (ϕ) and output gate (o_t).

In brief, LSTM has the ability to remove or add information to the cell state. It can be carefully regulated by structures called input, output and forget gates. In comparison, classical RNN completely overrides cell states. Besides, thanks to extra information flow than classical RNN, LSTM is able to ease the "vanishing gradients" problem greatly in training process.

Details about how LSTM work are illustrated in equations below.

First of all, the decision which information will be thrown away from the cell state is made by forget gate. It is mathematically expressed as

$$\overline{f_t} = W_f \cdot X_t + R_f \cdot h_{t-1} + b_f \tag{1}$$

$$f_t = \sigma(\overline{f_t}) \tag{2}$$

where f_t and $\bar{f_t}$ are the forget gate output and input respectively at time t.(W_f, R_f, b_f) are forget gate`s input weights, recurrent weights and bias. σ is nonlinear function (usually is sigmoid function), and "·" means matrix multiplication.

Next, input gate decides which states will be updated (Eq. (3, 4)). Meanwhile, a nonlinear layer φ (usually hyperbolic tangent, denoted by "tanh") creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state (Eq. (5, 6)). After these, combine these two parts to create an update to the states.

$$\overline{\iota_t} = W_i \cdot X_t + R_i \cdot h_{t-1} + b_i \tag{3}$$

$$i_t = \sigma(\overline{\iota_t}) \tag{4}$$

$$\bar{\tilde{C}}_t = W_C \cdot X_t + R_C \cdot h_{t-1} + b_C \tag{5}$$

$$\widetilde{C}_t = \varphi(\bar{\tilde{C}}_t) \tag{6}$$

where (W_i, R_i , b_i) and (W_C, R_C, b_C) are input weights, recurrent weights and bias of input gate and cell state layer. ($\overline{\iota_t}$, i_t) and ($\bar{\tilde{C}}_t$, $\widetilde{C}_t$) have similar meanings to (f_t,$\bar{f_t}$). "*" is element-wise multiplication.

Then, update the old cell state C_{t-1} into the new cell state C_t (see Eq. (7)).

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{7}$$

Lastly, the output gate decides which part of the cell states will be outputted (Eq. (8, 9)). Ultimately, the output h_t of LSTM unit at time t can be obtained by applying hyperbolic tangent function to cell states through and multiplying it by output gate values (Eq. (10)) and we can get the output h_t of LSTM unit at time t.

$$\overline{o_t} = W_o \cdot X_t + R_o \cdot h_{t-1} + b_o \tag{8}$$

$$o_t = \sigma(\overline{o_t}) \tag{9}$$

$$h_t = o_t * \phi(C_t) \tag{10}$$

where $(W_o,\ b_o)$ are the weights and bias of output gate, ϕ is activation function (usually tanh), "*" represents scalar multiplication and $(\overline{o_t}, o_t)$ have similar meaning to $(f_t,\bar{f_t})$.

According to the vanilla LSTM structure above, we can calculate the gradients of weights and biases everywhere. After that, put them into optimization algorithms, such as Stochastic Gradient Descent (SGD), RMSprop, or Adam, we will get the optimal value of cost function as well as the optimal solution. To avoid arriving at a local optimal solution, we repeated the training process three times for each trial.

3.2 Dynamic differential features

In health monitoring datasets, raw data are usually composed of operation history and sensor readings. These features are stationary. When a system or component works under a simple condition, these stationary features may be enough for vanilla LSTM to assess the health state of the study object (such as the issue in Section 4.4). But when the object works in complex operating modes and surfs hybrid faults (such as experiments in Section 4.5), vanilla LSTM fails to utilize only stationary features to master the underlying physical processes.

A direct way is to increase the network complexity, such as adding more hidden layers. Indeed, deeper architecture helps neural network extract and understand complicated features. Unfortunately, a larger model usually requires more training data correspondingly. In reality, very few expensive or safety critical systems are allowed to run to failure, which causing limit access to training data.

Another choice is to extract more features before training. Since inter-frame dynamic information contains a great amount of degradation information, we proposed a dynamic difference method to get new features from raw datasets. In details, three parts of features are designed.

1. The clustering result for operation values obtained by the K-means algorithm shows that there are six operation conditions in the datasets. The first part of new features are the cumulative number of times for different operating modes.
2. The current operation mode label (given by the trained K-means model above), which is a vector of six dimensions. Specifically, it is a "one-hot" encoding (a vector of zeroes with "1" at a single position) indicating the current operation mode.
3. Forward difference between current sensor monitoring values and previous values under the same operation mode in the history record. (When current operation mode firstly appears in the record, we initialize these features as "0").

The whole procedure is described in Algorithm 1 while the schematic diagram of new feature vector is shown in Figure 2.

Algorithm 1: Adding forward difference features

```
Assume:
- $\mathbf{u}$ operating values, $\mathbf{m}$ operation modes, $\mathbf{n}$ sensor values
-a single sequence sample with length $\mathbf{T}$ with ($\mathbf{u}+\mathbf{n}+\mathbf{m}*\mathbf{2}$) features
- $\mathbf{t}$ indicates the current time step ($\mathbf{t} \in [\mathbf{0}, \mathbf{T}-\mathbf{1}], \mathbf{t} \in \mathbf{N}$)
Require:
-the one-hot vector $\boldsymbol{\alpha}_t$ indicating current operation mode ($\boldsymbol{\alpha}_t \in \mathfrak{R}^m$)
-the vector $\boldsymbol{\beta}$ recording the previous position in the whole sequence where the operation, which is the same as the operation mode at time step t, occurred ($\boldsymbol{\beta}$ are initialized by zeroes, $\boldsymbol{\beta} \in \mathfrak{R}^m$) .

t:=0
```

```
while t<T:
    Load the number of current operation from $\boldsymbol{\alpha_t}$, denoted by $\boldsymbol{i}$.
    if $\boldsymbol{\beta_i}$=0:
        Extend the feature vector with n zeroes.
    else if $\boldsymbol{\beta_i}$!=0:
        Extend the feature vector with the forward differences of sensor values
        between the time step t and the time step $\boldsymbol{\beta_i}$.
    end if
    Update $\boldsymbol{\beta_i}$ with t.
end while
```

Specific performance analysis of these new features will be shown in Section 4.5.

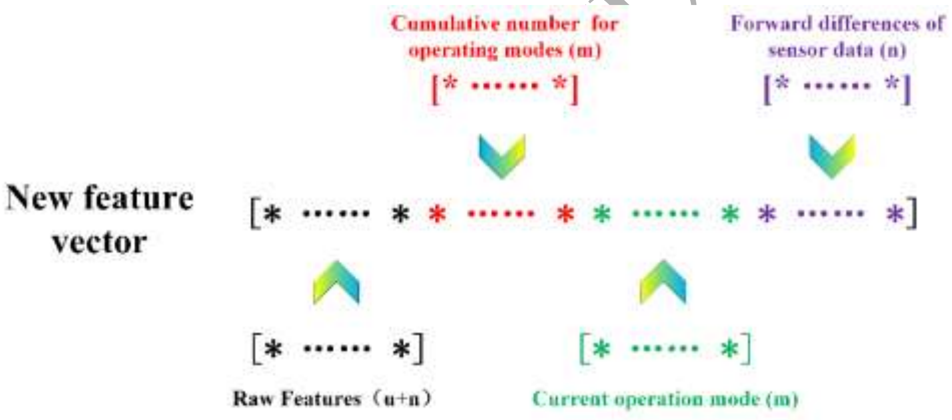

Cumulative number for operating modes (m)
Forward differences of sensor data (n)
New feature vector
Raw Features (u+n)
Current operation mode (m)

Fig.2. New feature vector

4. Experiments and discussion

The aim of this part is to demonstrate fast modeling and enhanced performances of using vanilla LSTM in the challenge of RUL estimation, in comparison to standard RNN and GRU LSTM. Experiments are carried out on four aircraft turbofan engine simulation datasets which are injected faults unknown by data analyzers and work in different complex conditions with noises. In model training phase, Man Square Error (MSE) on cross validation set is used to evaluate performance of the trained neural networks. To ensure good performance in face of new samples, final tests on the reserved test set are conducted; relative errors at certain detection points as well as a standard asymmetric scoring function are considered.

4.1. Benchmark dataset overview: damage propagation data about aircraft gas turbine engines

There are four issues for testing. These datasets consist of hundreds of cases of multivariate data that track from normal operation through fault onset to system failure. Data provided by the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) could model the damage propagation of aircraft gas turbine engines. This engine simulator, developed by NASA, allows faults to be injected in any of the five rotating components and gives output responses for 58 sensed engine variables. In this experiment, four datasets of different issues include three operating condition indicators and 21 of 58 output variables. The operating condition indicators are altitude, Mach number and throttle resolver angle respectively, which determine different flight conditions of aero-engine (namely the operation conditions discussed below). Description of sensed engine variables can be found in Table 1. Each simulated engine was given some initial level of wear within normal limits, and faults were initiated at some random time during simulation. Engine health was defined as the minimum health margin of the rotating equipment, where health margin is a function of efficiency and flow for that particular component. When this health indicator reached zero, the simulated engine was considered failed.

Table.1. Description of sensed engine variables

Sensor data number	Description	Units
1	Total temperature at fan inlet	° R
2	Total temperature at low pressure compressor outlet	° R
3	Total temperature at high pressure compressor outlet	° R
4	Total temperature at low pressure turbine outlet	° R
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure at high pressure compressor outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio	/
11	Static pressure at high pressure compressor outlet(Ps30)	psia
12	Ratio of fuel flow to Ps30	pps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass Ratio	/

16	Burner fuel-air ratio	/
17	Bleed Enthalpy	/
18	Demanded fan speed	rpm
19	Demanded corrected fan speed	rpm
20	High pressure turbine coolant bleed	lbm/s
21	Low pressure turbine coolant bleed	lbm/s

Table.2. Basic information of C-MAPSS dataset

Issue	**Fault type**	**Scale**	**Maximum life span (Cycles)**	**Average life span (Cycles)**	**Minimum life span (Cycles)**
1	High pressure compressor	100	362	206	**128**
2	High pressure compressor	260	378	206	**128**
3	High pressure compressor & Fan	100	525	247	145
4	High pressure compressor & Fan	249	**543**	245	**128**
All	/		543	226	128

Basic information of datasets is given in Table 2. Specifically, Issue 1 describes a situation that a fleet of engines suffered a failure of high pressure compressor with a single operation condition; Issue 2 describes a situation that engines suffered a failure of high pressure compressor with six operation conditions; Issue 3 is the situation that a fleet of engines suffered degradations of high pressure compressor and fan with a single operation condition; Issue 4 is the situation that a batch of engines suffered degradations of high pressure compressor and fan with six operation conditions. In addition, some other statistics, such as scales and maximum life span, are listed in Table 2. (See [19] for more details).

4.2. Software and hardware environment description

Experiments are run on a personal computer with Intel Core i5-M430 (2.27GHz) CPU, 4GB memory and Microsoft Windows 7 ultimate operation system.

Programming language is Python 2.7 with scientific computing library “Theano” (0.7.0) [20,21] and deep learning library “Keras” (0.3.2) [22]

4.3. Data preprocessing

First of all, we used mean and variance to normalize the data. It is worth noting that, in some cases, the number of operation conditions or sensor readings are likely to be constant, which causing related variances are zeroes. In these cases, the normalization operation of dividing by variance is abandoned.

Secondly, we padded zeroes to the header of sequences in order to make the length of all sequences the same as that of the longest one. After that, we masked input sequences to identify padding, which is achieved by an architecture named "masking layer". This layer copies the input to the output layer with identified padding replaced with zeroes and creates an output mask in the process. In this case, at each time step, if the values all equal the padding value (0 here), then the corresponding mask value for the time step is 0 (skipped); otherwise it is 1. These mask values can tell the neural networks above the masking layer whether a time step contains information for not.

Thirdly, since the degradation in a system will generally remain negligible until after some period of operation time and when the initial failure had developed, it is probably unreasonable to estimate RUL until the system begins to degrade. To solve this problem, Hermes [2] considered the RUL when the system is new as a constant value. But how to judge whether a system is new or not? Hermes observed the samples and judged the faults usually occur at the time step around 120 to 130. After some experiments, the value 130 was finally used because of its best performance on test dataset. In this paper, we try to use support vector machine (SVM) as anomaly detector whose penalty parameter is 2 and kernel function is radial basis kernel with the reach of 0.8409. These two hyperparameters were found by grid search technique on the whole four datasets. Besides, the algorithm will make an abnormal judgment only if three times consecutive anomaly warnings are issued, through which we can get rid of disturbance to some degree.

Lastly, when there are more than one operation condition in the degradation processes, like Issue 2 and 4, we find that the monitoring data are almost completely changed with the change of operation. It means the changes caused by faults may be regarded as noises by LSTM because of their smaller variances. Therefore, the dynamic difference technology mentioned in Section 3.2 was used to extract new features from original healthy monitoring datasets.

4.4 First issue: single degradation with single operation mode and working condition

4.4.1. Data visualization

Figure 3(a) shows all sensed engine variables of a certain sample (No.1 sample of Issue 1). Through observation, we roughly sort the trends into three categories and list the details in Table 3. Further, figure 3(b) shows three sensed variables serving as typical representatives of three trends. In addition, features in Figure 3 are all normalized.

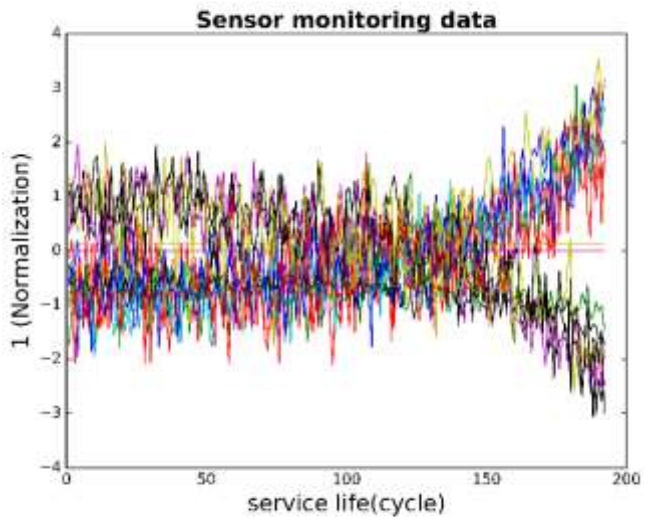


Fig.3 (a). All of sensor monitoring data in Issue 1.

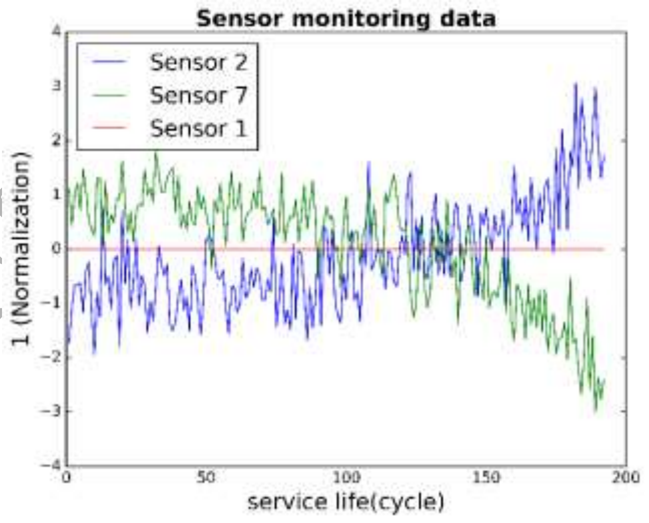


Fig.3 (b). Part of sensor monitoring data in Issue 1

Table.3. Sensor data trends in Issue 1

Trend	Sensor data number
Ascending	2, 3, 4, 8, 9, 11, 13, 15, 17
Descending	7, 12, 20, 21
Irregular/Unchanged	1, 5, 6, 10, 14, 16, 18, 19

In Figure 3 (b), sensor 2 is a representative of features that indicates a gradually upward trend over time, while sensor 7 behaves the opposite. In addition, features, like sensor 1, remain relatively stable over time. In summary, we find that this kind of fault affects sensors readings in different ways.

4.4.2. Networks configuration and training

As for model, we use hyperbolic tangent as the activation of hidden neurons as well as hard sigmoid [22] for the gates. Initialization approach set the bias of the forget gate as recommended by Jozefowicz et al [23]. Probability of dropping out neurons at output layer of LSTM is simply set to 0.5 as recommended by Hinton et al [24]. Due to the powerful ability of Dropout for preventing neural networks from overfitting, the size of model need to be large enough as shown in Table 5. In addition, an earlystopping mechanism to monitor the performance improvement on cross validation set are also used to prevent model from overfitting (the number of epochs with no improvement is set to 5).

As for data, on one hand, all features are used as inputs for training, because neural network can handle the useless features well automatically. On the other hand, Due to no target output (real life) supplied by raw datasets, before training learning models, we have to label the RUL at every time step for each sample. Some labeling methods are tested and their interpretations can be found as follows.

20% samples of train set data are randomly designated as cross validation set. Besides, all experiments below have been repeated for three times, and only the best items evaluated by the MSE on cross valid set are listed.

4. Some labeling methods for RUL of training set were tested, with results listed in Table 4;
5. Different optimizers (RMSprop, Adadelta, Adam, Adamax) were compared in Table 5;
6. Different quantities of recurrent layer neurons were tried and results were shown in Table 6;
7. GRU and standard RNN with the similar configuration of vanilla LSTM were tested, and the results can be found in Table 7;

Table.4. Labeling methods

Labeling method	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
1-linearly decrease	1619.7873	1015.4997
2-anomaly detection, linearly decrease	876.328	946.5884
3-anomaly detection, decrease by power function of 1.1	722.5342	**779.6424**
4-anomaly detection, decrease by power function of 1.2	880.8783	1200.7709
5-anomaly detection, decrease by power function of 1.5	1102.825	1038.0184
6-anomaly detection, decrease by quadratic function	1042.2564	910.1109

Where method 1 means "RUL linearly decrease", method 2-6 means "Use SVM to detect the position of abnormal point. Points before that are labeled by the real RUL at this critical point while the RULs after that are considered decreasing as power functions". The powers in method 2-6 are 1, 1.1, 1.2, 1.5, and 2 respectively. It should be noted that experiments above have been run on settings of 128 neurons in the hidden layer and the RMSprop optimization algorithm.

Table.5. Optimizers

Optimizer	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
RMSprop	722.5342	779.6424
Adadelta[25]	1163.4858	731.5966
Adamax	1035.9937	650.8824
Adam[26]	1222.1583	**557.7891**

The results above based on the mentioned configuration of labeling method 3 and 128 nodes in recurrent layer.

Table.6. Quantity of Hidden layer neurons

Hidden layer neuron	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
32	1254.665	687.6032
64	898.0582	461.1558
128	1222.1583	557.7891
192	794.9226	928.1835
256	1194.6137	**390.3185**

The items in this table was based on the configuration of labeling method 3 and Adam algorithm.

Table.7. Types of RNN (256 nodes)

Architecture	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
Standard	871.5052	1259.0516
GRU LSTM	745.7424	620.8585
vanilla LSTM	1194.6137	**390.3185**

Similarly, these results are based on the best setting in previous experiments: labeling method 3, 256 neurons and Adam.

Last but not least, according to the study of Klaus Greff [14], the hyperparameter interactions in LSTM is quite small. This implies that we can tune the optimizer, network size and other hyperparameters independently, thus saving a lot of experimentation time.

4.4.3. Model testing and brief summary

In fact, before we started to train models, a test set with a size of 20 were randomly picked out for final accuracy performance test (the scale of test sets in Section 4.5 and 4.6 are the same). Part of prediction results for the test set are randomly chosen to be visualized as is showed in Figure 4.

The final performance of model is measured by the relative errors (more specifically, the percentages of samples whose relative errors are less than or equal to 5%, 10% and 20% respectively in the test set.) and a standard asymmetric scoring function proposed in [19], at some monitoring points (0, 5, 10, 50 cycles away from completely failure).

The scoring function mentioned above is as follows

$$d = RUL_{estimated} - RUL_{actual} \tag{11}$$

$$score = \begin{cases} e^{-d/13} - 1, d < 0 \\ e^{-d/10} - 1, d \geq 0 \end{cases} \tag{12}$$

Since less scores a model indicates its achieving a better performance, we can conclude that an early prediction is better than a late one if they have the same absolute error. In other words, a late prediction in reality may be catastrophic.

In short, an ideal model achieves small relative errors and low value of the scoring function on the test set.

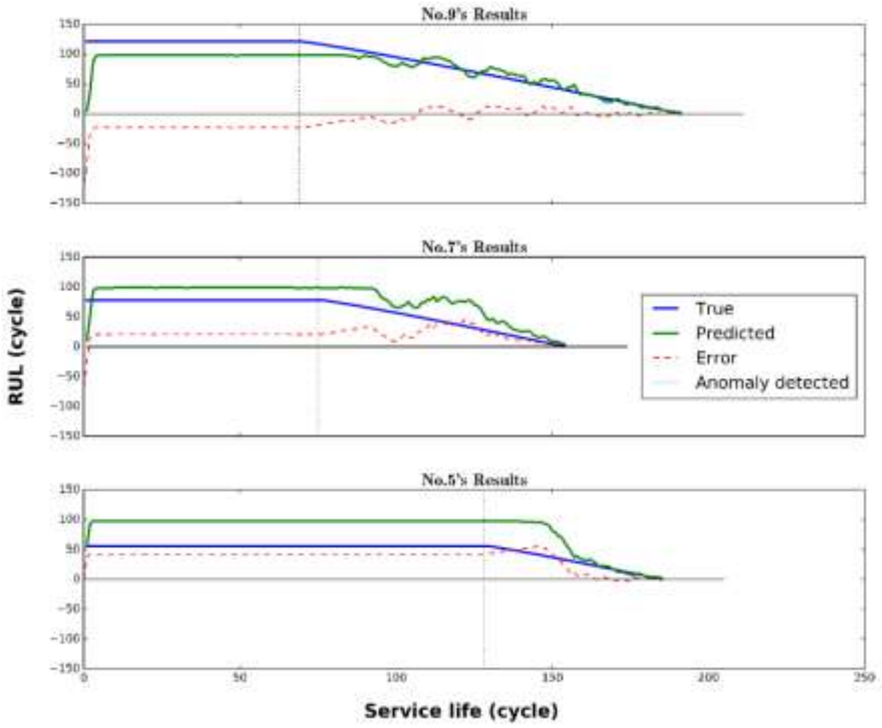


Fig.4. Parts of prediction results on test set

The results of comparison among three types of RNNs are listed in Table 8(in Appendix).For convenience, the percentages of samples whose relative errors are less than or equal to some certain values are denoted by P_*, where the subscript represents the relative error limit (0.05 ,0.1 and 0.2 in this paper). Besides, the scoring function mentioned above is denoted by SF.

At last, experiments about Issue 1 are summarized as follows

1. According to the first half of the remaining life curves (before abnormality occurs), vanilla LSTM has successfully learnt the method to automatically evaluate the lives of this kind of engines under healthy state as showed in Figure 4. In fact, the large errors in this period are of no great importance;
2. During degradation process (the last half of the remaining life curves), the RULs of each engine are continuously reducing in a slightly accelerated manner, demonstrated in Table 4. The prediction errors are also decreasing with time generally, as shown in Figure 4;
3. Table 5 illustrates that four advanced optimization algorithms yield considerable performances while Adam is a stroke above eventually;
4. Table 6 shows that the quantity of neurons in recurrent layer does not affect performance obviously in a large range, which may be owed to the

dropout mechanism;

5. LSTMs, including vanilla and GRU, outperform either in long term prediction or short term prediction than standard RNN. The performance gap between vanilla LSTM and GRU on test set is not significant (Table 8). In the long-term prediction (50 and 20 cycles left), the GRU achieved better performance than the vanilla LSTM did, whereas in the short-term prediction (5 cycles left), the vanilla LSTM won.

4.5. Second issue: single degradation with multiple operation modes and working conditions

4.5.1. Data visualization

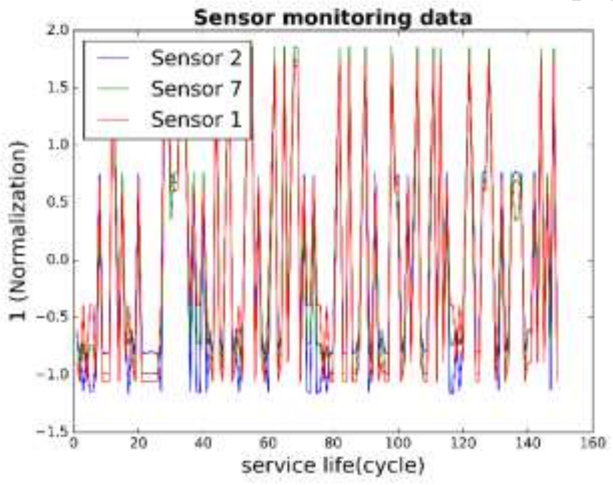


Fig.5 (a). Part of sensor monitoring data in Issue 2

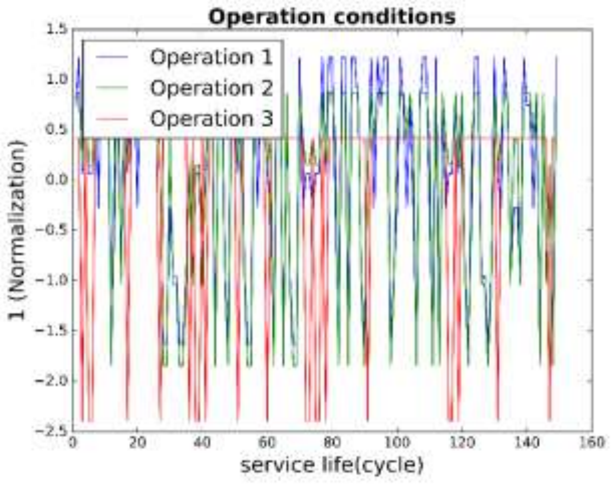


Fig.5 (b). Operation conditions data in Issue 2

As we can see in Figure 5, the samples in Issue 2 are significantly different from those in Issue 1. It seems that sensor readings (The same sensors with Issue 1 are displayed) no longer have obvious trends either upward or downward. Through the observation on Figure 5 (b), we find that these sensor data may depend on the operations to a large degree, which makes the changes caused by degradation unapparent.

4.5.2. Networks configuration and training

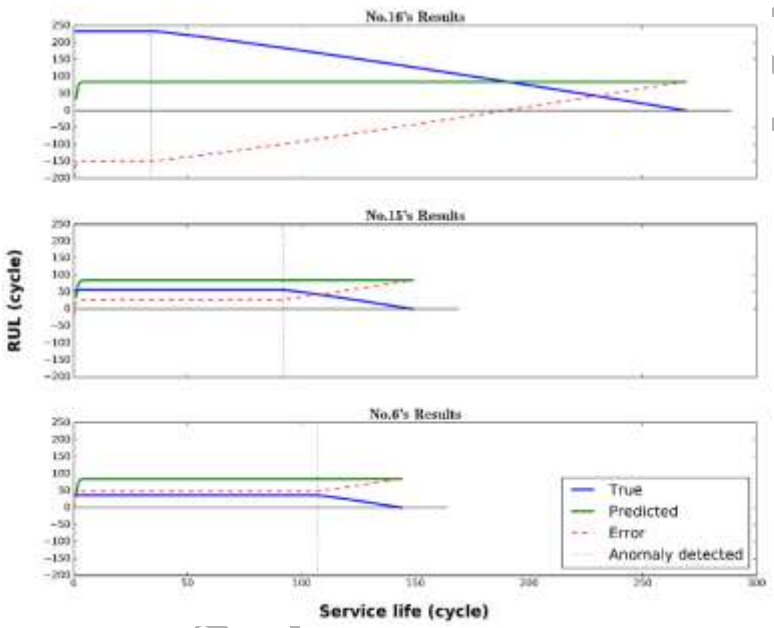


Fig.6. Prediction results without dynamic differential features

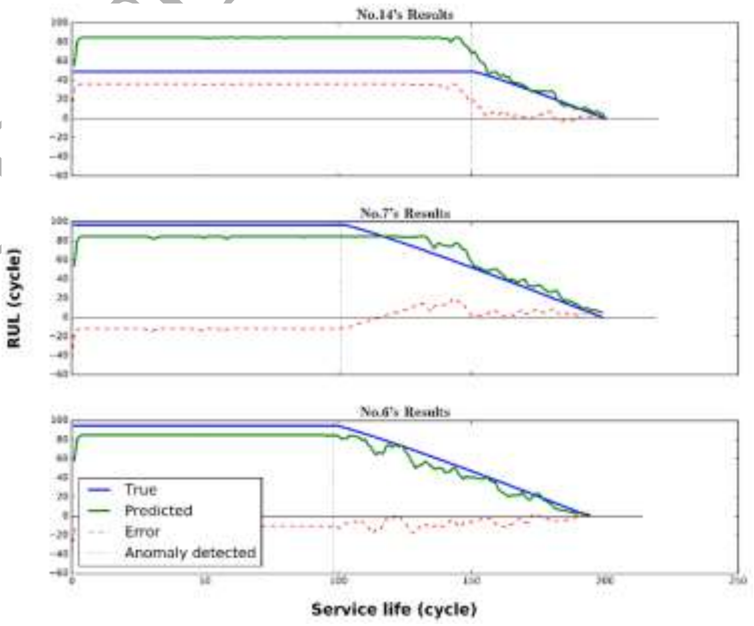


Fig.7. Parts of prediction results on test set

Firstly, the best configuration acquired from Issue 1 was used to solve Issue 2. Unfortunately, as we can see in Table 9 and Figure 6, it failed to find the law behind these complex features. Then we proposed adding derived features described in Section 3.2 to solve these problems. The results in Table 9 shows that these extra features help LSTM achieve excellent improvement. Lastly, further experiments were carried out to determine the optimal number of neurons (Table 9) and to compare the performance among three types of RNNs with the same 32 hidden units (Table 10).

Table.9. Quantity of Hidden layer neurons

Hidden layer neuron	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
256(Without derived features)	1942.0045	2604.6789
256	1221.3657	933.5156
192	1293.3593	829.3265
128	1128.5305	1319.3696
64	1257.1059	1155.1759
32	1472.5239	**743.3566**

Table.10. Types of RNN (32 nodes)

Architecture	**Train set performance (MSE)**	**Cross validation set performance (MSE)**
standard RNN	1791.0168	1061.5426
GRU LSTM	1513.361	**739.8178**
vanilla LSTM	1472.5239	743.3566

4.5.3. Model testing and brief summary

Some prediction results (of vanilla LSTM with 32 hidden neurons) for random sample sequences in test set are depicted in Figure 7 intuitively. Meanwhile, quantitative indicators are listed in Table 11(in Appendix).

Experiments about Issue 2 are summarized as follows

1. The method proposed to add new dynamic differential features solved the challenge of multiple operating modes well;
2. Vanilla LSTM still achieved the best prediction accuracy at most of the monitoring points;
3. Judging from Table 9 and 11, the model with best cross validation performance was best did not obtain the best performance on test set, which can be concluded from the experiment between LSTM and GRU

with the same complexity and that carried on LSTMs with different complexity. This may be largely caused by a lack of training samples.

4.6. Issue 3 and 4: Multiple faults modes

4.6.1. Networks configuration and training

Issue 3 is a problem about two faults happened in different components of aero-engine under a single operating mode while Issue 4 is under six operating modes. As Issue 3 and Issue 4 is similar to Issue 1 and Issue 2, we will simplify the description of these two data sets.

We listed the training results with different numbers of hidden neurons in Table 12 and Table 13 (Issue 3 and Issue 4 respectively) to look for the optimal RNN model for cross validation set. The different performances of simple RNN, GRU LSTM and vanilla LSTM are showed in Table 14 and 15.

Table.12. Quantity of Hidden layer neurons (Issue 3)

Hidden layer neuron	Train set performance (MSE)	Cross validation set performance (MSE)
256	3485.0663	**578.1255**
192	2705.3609	1028.0435
128	3256.9508	1128.8954
64	2279.0373	1738.476
32	2993.8324	1318.4552

Table.13. Quantity of Hidden layer neurons (Issue 4)

Hidden layer neuron	Train set performance (MSE)	Cross validation set performance (MSE)
256	2507.6985	1486.3111
192	2130.7925	1261.6147
128	2169.9917	1445.5568
64	2057.2721	1515.675
32	2710.7073	**1205.4032**

Table.14. Types of RNN (Issue 3, 256 nodes)

Architecture	Train set performance (MSE)	Cross validation set performance (MSE)
standard RNN	3002.1542	766.9153
GRU LSTM	2068.7082	907.21
vanilla LSTM	3485.0663	**578.1255**

Table.15. Types of RNN (Issue 4, 32 nodes)

Architecture	Train set performance (MSE)	Cross validation set performance (MSE)
standard RNN	3069.8547	1333.1466
GRU LSTM	2458.4576	1299.4646
vanilla LSTM	2710.7073	**1205.4032**

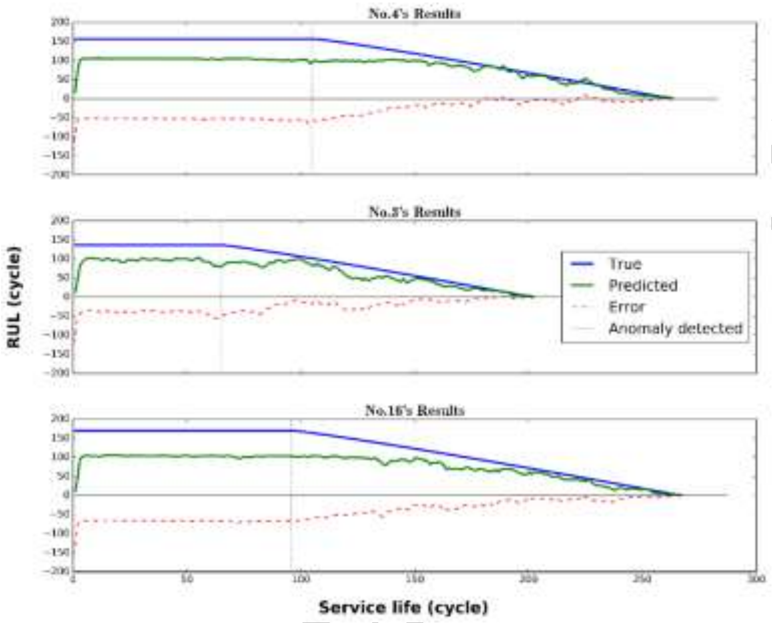


Fig.8. Parts of prediction results on test set (Issue 3)

4.6.2. Model testing and brief summary

In this test, estimates of part of samples are demonstrated in Figure 8 and Figure 9, while quantitative appraising criteria are listed in Table 16 and Table 17 respectively (in Appendix).

Experiments about Issue 3 and Issue 4 are summarized as below:

1. LSTMs also achieved far better accuracies than standard RNN for the multiple fault problems.
2. Vanilla LSTM was still a stroke above GRU LSTM except at the monitoring point “50 cycles left” in Issue 3 (Table 16).
3. Without obvious model complexity improvement, the accuracies showed no apparent decline on datasets whose degradation processes become more complicated. It proves that vanilla LSTM has powerful abilities to recognize and learn hybrid faults. With enough samples, this algorithm could learn more complex objects of degradation without many changes.

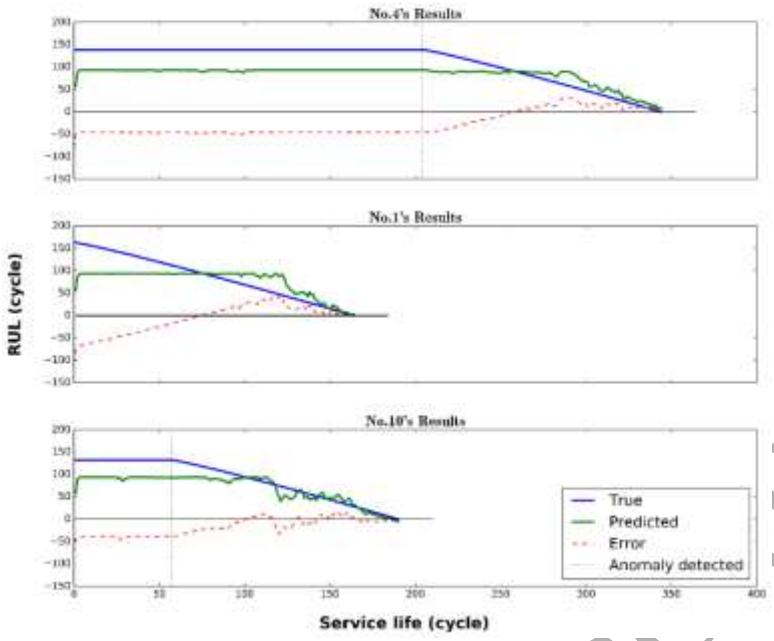


Fig.9. Parts of prediction results on test set (Issue 4)

4.7. Unified LSTM model for all different situation

From Section 4.4 to Section 4.6, we tested the ability of LSTM network to predict the remaining useful life of the Issue 1 to 4 successively. During this process, we proposed a dynamic difference method to get new features from raw dataset. However, there are some flaws in the above process:

1. The derived features are not added into experiments until Issue 2. In other words, the experiment about Issue 1 does not use this method;
2. Experiments in Section 4.4 to 4.6 verify that LSTM can make accurate predictions for a variety of situations. However, in the actual system, it is often difficult to sort data into all kinds of situations and then train different algorithm models purposefully.

Based on those two points, in order to take care of the integrity of test about derived features and to further verify the adaptability of the LSTM algorithm to real industrial environment, we try to utilize the whole model degradation data (Issue 1-4, shuffled) to train a unified LSTM model.

Because of a larger sample size, we are able to use a larger test set of 50 samples for the performance test. The configuration of LSTM model is similar to Section 4.4 to 4.6. In this experiment, different numbers of LSTM state cells (32-96) are tested and their quantitative appraising criteria are listed in Table 18 (in Appendix).

Compared with the data in Table 8, 11, 16 and 17, there is no obvious loss in prediction performance compared to experiments using separate data for training. Therefore, we conclude that in face of the dataset of mixed cases, LSTM can be adjusted to the adaptive prediction scheme spontaneously and show sufficient fitting ability. Meanwhile, the derived features designed in Section 3.2 above is also compatible for Issue 1.

5. Conclusion

In this paper, vanilla LSTM neural networks, which usually work effectively in the field of natural language processing, are utilized to solve the bottleneck problem of high-precision RUL estimation for complicated engineered systems. Besides, a dynamic difference technology is proposed to extract new features from raw health monitoring data, with which RNNs can make full use of inter-frame information to find the real physical degradation mechanism behind sensor readings under complex and changeable operating modes. Both of these works tremendously contribute to condition-based maintenance of complex engineered system for lower maintenance cost and catastrophic loss.

The performances of the proposed methods mentioned above are benchmarked with standard RNN (the model used by the Champion group of PHM08) and GRU LSTM (a simplified version of LSTM proposed recently) for four types of prediction problems of aircraft turbofan engines. In all cases, vanilla LSTM shows several times (even up to hundreds of times) performance improvements at different monitoring points, based on the official scoring function. From another point of view, at the monitoring points after 10 cycles (included), the samples whose relative errors are less than 5% occupy above 90% of all the test samples.

In this paper, the verification object is RUL of aero engine. However, the research process generalize as much as possible. Therefore, the proposed methods can be easily transferred to any other datasets like "operation condition - sensor variable".

Last but not least, we assume that random partitioning for dataset in each experiment yields the instability during the training phase and the performance inconsistency, which sometimes existing between cross validation set. This phenomenon may be relieved by more available samples for training.

Acknowledgements

This work was supported by National Natural Science Foundation of China under Grant No. 51375030. First of all, I would like to thank NASA Ames Research Center for providing turbofan engine degradation simulation data set. Secondly, I would like to express my sincere thanks to my supervisor Mei Yuan and Shaopeng Dong, who have given me so much useful advices on my writing and have tried their best to improve my paper. Last but not the least, I'd like to thank my junior apprentice Lin Li and Yingqi Liu for their hard work in searching for best hyperparameters of neural network models.

References:

[1]L. Peel, Data driven prognostics using a Kalman filter ensemble of neural network models, (IEEE, 2008), pp.1-6.

[2]F.O. Heimes, Recurrent Neural Networks for Remaining Useful Life Estimation, 2008), pp.59-64.

[3]Y. LeCun, Y. Bengio, G. Hinton, Deep learning, NATURE, 521(2015)436-444.

[4]N. Boulanger-Lewandowski, Y. Bengio, P. Vincent, Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription, (2012).

[5]A. Graves, A. Mohamed, G. Hinton, Speech Recognition with Deep Recurrent Neural Networks, (2013).

[6]I. Sutskever, J. Martens, G. Hinton, Generating Text with Recurrent Neural Networks, ICML, 2011.

[7]I. Sutskever, G.E. Hinton, G.W. Taylor, The Recurrent Temporal Restricted Boltzmann Machine, (2008)1601-1608.

[8]Y. Bengio, P. Simard, P. Frasconi, Learning Long-Short Term dependency is difficult, IEEE Transactions ON Neural Networks, 5(1994)157-166.

[9]H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, SCIENCE, 304(2004)78-80.

[10]S. Hochreiter, J.U.R. Schmidhuber, Long short-term memory, NEURAL COMPUT, 9(1997)1735-1780.

[11]W. Hong, RESEARCH ON REMAINING USEFUL LIFE PREDICTION FOR SATELLITE LITHIUM-ION BATTERY(In Chinese)
, Master, Harbin Institute of Technology, 2013.

[12]A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, NEURAL NETWORKS, 18(2005)602-610.

[13]K. Cho, B. van Merrienboer, D. Bahdanau, Y. Bengio, On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, (2014).

[14]K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM: A Search Space Odyssey, (2015)10, 8.

[15]K. Javed, R. Gouriveau, N. Zerhouni, SW-ELM: A summation wavelet extreme learning machine algorithm with a priori parameter initialization, NEUROCOMPUTING, 123(2014)299-307.

[16]W. Yu, F. Zhuang, Q. He, Z. Shi, Learning deep representations via extreme learning machines, NEUROCOMPUTING, 149(2015)308-315.

[17]P. Tamilselvan, P. Wang, Failure diagnosis using deep belief learning based health state classification, RELIAB ENG SYST SAFE, 115(2013)124-135.

[18]R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training Recurrent Neural Networks, (2012).

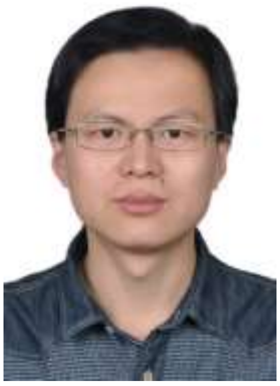
[19]A. Saxena, K. Goebel, D. Simon, N. Eklund, Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation, International Conference on Prognostics and Health Management, (IEEE, 2008), pp.1-9.

[20]F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, Y. Bengio, Theano: new features and speed improvements, (2012).

[21]J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: A CPU and GPU Math Compiler in Python, Proceedings of the Python for Scientific Computing Conference (SciPy), Austin, TX, 2010.
[22]F. Chollet, Keras, GitHub repository（https://github.com/fchollet/keras）, 2015.
[23]R. Jozefowicz, W. Zaremba, I. Sutskever, An Empirical Exploration of Recurrent Network Architectures, Proceedings of the 32 nd International Conference on Machine Learning, Lille, France, 2015.
[24]N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, J MACH LEARN RES, 15(2014)1929-1958.
[25]M.D. Zeiler, ADADELTA: an adaptive learning rate method, arXiv preprint arXiv:1212.5701, (2012).
[26]D. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, (eprint arXiv:1412.6980, 2014.

Yuting Wu is a master candidate student jointly educated by School of Automation Science and Electrical Engineering and School of Energy and Power Engineering, Beihang University. He received the bachelor's degree from School of Information Science and Engineering, Central South University, Changsha, China in 2014. His research interests include machine learning, data mining and deep learning.

Mei Yuan is an Associate Professor, School of Automation Science and Electrical Engineering, Collaborative Innovation Center for Advanced Aero-Engine, Beihang University, Beijing, China. Her current research interests include prognostic and health management, advanced signal processing, embedded systems, structural health monitoring of complex system. She is currently the member and secretary of Chinese Society of Aeronautics and Astronautics GNC branch, director and member of Chinese Instrument and Control Society SHM branch, senior member of Chinese Metrology Society.



Shaopeng Dong is currently a lecturer and working towards the Ph.D. degree at School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. He received the bachelor's degree in Automation from China Agriculture University, Beijing, China in 2004. He received the master's degree in Detection Technology and Automatic Equipment from Beihang University, Beijing, China in 2007. His main research interests include prognostic and health management, embedded system, signal processing, structural health monitoring of complex system.

Li Lin is a Master candidate student in the School of Energy and Power Engineering, Beihang University. She received the bachelor`s degree from the School of Electric Engineering and Automation, Hefei University of Technology, Anhui, China, in 2015. Her research interests include machine learning, automatic test system and sensor technology.

Yingqi Liu is currently a Master candidate in the school of Automation Science and Electrical Enginnering at Beihang University. He is graduated in Ecole Centrale de Pekin from Beihang University, Beijing, China, in 2015. His main research interests include prognostic and health management (PHM), deep learning, machine learning and data mining.

Appendix

Table.8. Performance on test set (Issue 1)

Monitoring point	50 cycles left				10 cycles left				5 cycles left				Completely failure			
Performance / Architecture	P_5 (%)	P_{10} (%)	P_{20} (%)	SF	P_5 (%)	P_{10} (%)	P_{20} (%)	SF	P_5 (%)	P_{10} (%)	P_{20} (%)	SF	P_5 (%)	P_{10} (%)	P_{20} (%)	SF
Standard RNN	10.0	35.0	60.0	5581.4146	75.0	90.0	95.0	256.3407	85.0	95.0	95.0	174.2749	95.0	95.0	95.0	118.5207
GRU LSTM	**45.0**	**85.0**	**85.0**	**17.9591**	**100.0**	**100.0**	**100.0**	**0.3536**	**100.0**	**100.0**	**100.0**	**0.2987**	**100.0**	**100.0**	**100.0**	0.5267
vanilla LSTM	30.0	55.0	70.0	37.5090	**100.0**	**100.0**	**100.0**	0.4281	**100.0**	**100.0**	**100.0**	**0.2561**	**100.0**	**100.0**	**100.0**	**0.2979**

Table.11. Performance on test set (Issue 2)

Monitoring point	**50 cycles left**				**10 cycles left**				**5 cycles left**				**Completely failure**			
Performance / **Architecture**	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF
standard RNN	5.0	15.0	**100.0**	13.0393	45.0	60.0	70.0	162.6788	20.0	75.0	80.0	227.8299	0.0	55.0	80.0	254.5734
GRU LSTM	**35.0**	55.0	90.0	9.2904	65.0	90.0	**100.0**	1.4557	70.0	90.0	**100.0**	1.0743	85.0	**100.0**	**100.0**	0.7755
vanilla LSTM(32 nodes)	30.0	75.0	90.0	6.5820	90.0	**100.0**	**100.0**	0.6476	95.0	**100.0**	**100.0**	**0.3253**	95.0	**100.0**	**100.0**	0.3906
vanilla LSTM(128 nodes)	30.0	**85.0**	95.0	**3.8973**	**95.0**	**100.0**	**100.0**	**0.4753**	**100.0**	**100.0**	**100.0**	0.6328	**100.0**	**100.0**	**100.0**	**0.3369**

Table.16. Performance on test set (Issue 3)

Monitoring point	**50 cycles left**				**10 cycles left**				**5 cycles left**				**Completely failure**			
Performance / **Architecture**	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF
standard RNN	35.0	60.0	85.0	48.9607	**100.0**	**100.0**	**100.0**	0.3612	95.0	**100.0**	**100.0**	0.4303	90.0	**100.0**	**100.0**	0.5583
GRU LSTM	40.0	65.0	**95.0**	**14.2948**	90.0	**100.0**	**100.0**	1.4595	85.0	**100.0**	**100.0**	1.7369	60.0	**100.0**	**100.0**	2.2087
vanilla LSTM	**50.0**	**70.0**	90.0	30.5857	**100.0**	**100.0**	**100.0**	**0.2302**	**100.0**	**100.0**	**100.0**	**0.2090**	**100.0**	**100.0**	**100.0**	**0.3782**

Table.17. Performance on test set (Issue 4)

Monitoring point	**50 cycles left**				**10 cycles left**				**5 cycles left**				**Completely failure**			
Performance / **Architecture**	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF
standard RNN	20.0	40.0	80.0	22.6458	0.0	50.0	85.0	18.0165	5.0	20.0	85.0	19.9516	0.0	20.0	75.0	36.5018
GRU LSTM	35.0	60.0	85.0	30.4508	85.0	**100.0**	**100.0**	**1.0037**	80.0	**100.0**	**100.0**	1.2014	70.0	95.0	**100.0**	1.6341
vanilla LSTM	**45.0**	**65.0**	**90.0**	**10.4662**	**90.0**	**100.0**	**100.0**	1.0753	**95.0**	95.0	**100.0**	**0.8152**	**95.0**	**100.0**	**100.0**	**0.7303**

Table.18. Unified model Performance on test set

Monitoring point	**50 cycles left**				**10 cycles left**				**5 cycles left**				**Completely failure**			
Performa-nce / **Hidden neuron**	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	SF
32	46.0	86.0	90.0	9.3185	80	100.0	100.0	1.2116	80.0	96.0	100.0	3.1440	**85.0**	95.0	100.0	1.6172
64	**60.0**	**94.0**	**98.0**	**4.0756**	**90.0**	**98.0**	**100.0**	**0.7922**	**84.0**	**100.0**	**100.0**	**0.7891**	76.0	**98.0**	**100.0**	**1.1202**
96	38.0	62.0	78.0	36.5882	56.0	64.0	88.0	46.5021	64.0	64.0	84.0	29.2324	62.0	64.0	82.0	32.4454