Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network

Mei Yuan, Yuting Wu and Li Lin

***Abstract*—Aero engine is a kind of sophisticated and expensive industrial product. Accurate fault location and Remaining Useful Life (RUL) estimation for aero engine can lead to appropriate maintenance actions to avoid catastrophic failures and minimize economic losses. The aim of this paper is to propose utilizing Long Short-Term Memory (LSTM) neural network to get good diagnosis and prediction performance in the cases of complicated operations, hybrid faults and strong noises. The whole proposition is demonstrated and discussed by carrying out tests on a health monitoring dataset of aircraft turbofan engines provided by NASA. Performances of LSTM and some of its modifications were tested and contrasted. Experiment results show the standard LSTM outperformed others.**

I. Introduction

Due to high complexity of aero engine, it is a challenging task to locate faults and predict remaining useful life accurately and rapidly depending on traditional maintenance concepts, modes and methods. With the high speed developments in sensing, signal processing and artificial intelligence technology. Prognostics and Health Management (PHM) technologies based on data (experience) have gradually become the mainstream solution either in fault diagnosis or remaining useful life estimation, instead of physics-based methods who can be expensive and tedious to develop [1-3].

Among data-driven approaches, neural network is one of the most state-of-the-art models for a variety of sequence classification and prediction problem in recent years. Peel et al. tried to use Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF) networks to evaluate the RUL of aero-engines [4] while Heimes et al. using classical RNN [5]. Hermes's network achieve performance improvement thanks to the RNN`s hidden neurons who implicitly contain information about the history of all the past elements in the sequence.

Unfortunately, if we try to unfold recurrent connections of standard RNN, it will be found that RNN is a very deep feedforward network in which all the layers share the same weights. This is called "long-term dependencies", causing that it is difficult to learn to store information for very long [6]. LSTM is a significant branch of Recurrent Neural Networks (RNN), capable of learning long-term dependencies [7, 8]. It has been shown to be effective in many sequence problem, such as music [9], speech [10], text [11] and motion capture data [12].

In the research field of PHM, on one hand, LSTM has not gotten wide attention. On the other hand, many researchers usually only studied the RUL estimation performance of their algorithms under a certain fault [1, 4, 5, 13-15]. In these cases, we have to train multiple estimator for various failure modes. When real system works, algorithm must judge the type of fault first, and then, select a suitable life estimator according to the fault type. The shortcomings of this approach are apparent. It is necessary to train a number of models, which makes the prediction system complicated, and is difficult to adapt to hyper-fault scenarios, which is very common in real systems.

Based on the above challenges, we propose to utilize LSTM to build a common model for multiple faults and hybrid faults. Besides, we can get RUL estimate value and probability of each fault at the same time. On the basis of ensuring the high accuracy, it can greatly simplify the complexity of the whole PHM system and provide more comprehensive information for maintenance plan.

II. Concept of LSTM

The initial version of LSTM was presented by Hochreiter and Schmidhuber in 1997 [7]. The most popular version was a modification refined by many people [16, 17], which is named vanilla LSTM (hereinafter referred to as LSTM). The architecture of LSTM can be described as Fig 1.

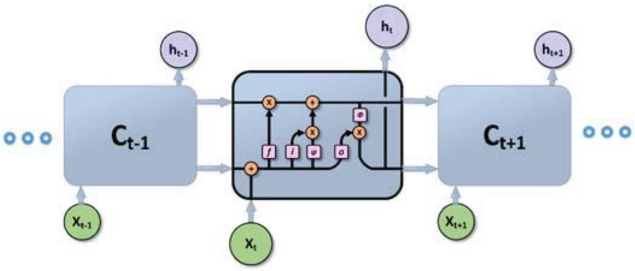

Figure.1. Vanilla LSTM

Its formulas are as follows

$$f_t = \sigma(W_f \cdot X_t + R_f \cdot h_{t-1} + b_f) \quad (1)$$

$$i_t = \sigma(W_i \cdot X_t + R_i \cdot h_{t-1} + b_i) \quad (2)$$

$$o_t = \ \sigma(\ W_o \cdot X_t + R_o \cdot h_{t-1} + b_o) \quad (3)$$

$$\widetilde{C_t} = \varphi(W_C \cdot X_t + R_C \cdot h_{t-1} + b_C) \quad (4)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (5)$$

$$h_t = o_t * \phi(C_t) \quad (6)$$

(1) ~ (3) are forget gate, input gate and output gate respectively. The Forget gate decides which historical

Mei Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (corresponding author phone: +86 010-82338757; E-mail: yuanm@buaa.edu.cn).

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. (E-mail: zvc_xoyo@buaa.edu.cn).

Li Lin is a Master candidate student in the School of Energy and Power Engineering, Beihang University. (E-mail: 15656516636@163.com).

978-1-5090-1087-5/16/$31.00©2016 IEEE

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.

information will be discarded from the cell state, the input gate decides which states will be updated and the output gate decides which part of the cell states will be outputted. In this way, LSTM has the ability to remove or add information to the cell state, instead of the mechanism of completely overriding cell states taken by classical RNN. (W_*, R_*, b_*) are each gate`s input weights, recurrent weights and bias respectively (* can be *f*, *i*, or *o*), σ is nonlinear function (σ here usually is sigmoid function) and "·" means matrix multiplication. Then, (4) is used to calculate the candidate values that will be added to the new state (time *t*) described by (5), together with the values of old state (time *t*-1) which are regulated by forget gate. Eventually, (6) represents the final outputs of LSTM unit. It is worth noting that φ and ϕ are often hyperbolic tangent function but can be other rational nonlinear activation functions. In addition, the point-wise multiplication of two vectors is denoted with"*".

III. SAMPLE LABELING AND DATA PREPROCESSING

The dataset for experiment consists of hundreds of cases of multivariate data that track from normal operation through fault onset to system failure. Data were provided by the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) which 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 [18].

In this experiment, four subsets from different issues are used (709 samples in total), which include 21 of 58 output variables (partially displayed in Fig 2(a), the data in figure have been normalized and more implementation details can be found in Part B of this section) and three operating condition indicators (shown in Fig 2(b)). Issue 1 (No.1-100) describes a situation that a fleet of engines suffered a failure of high pressure compressor with a single operation condition; Issue 2 (No.101-360) describes a situation that engines suffered a failure of high pressure compressor with six operation conditions; Issue 3 (No.361-460) is the situation that a fleet of engines suffered degradations of high pressure compressor and fan with a single operation condition; Issue 4 (No.461-709) is the situation that a batch of engines suffered degradations of high pressure compressor together fan with six operation conditions.

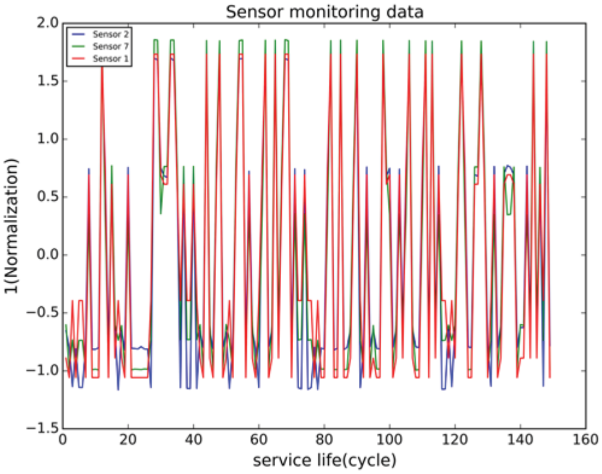

Figure.2(a). Part of sensor monitoring data

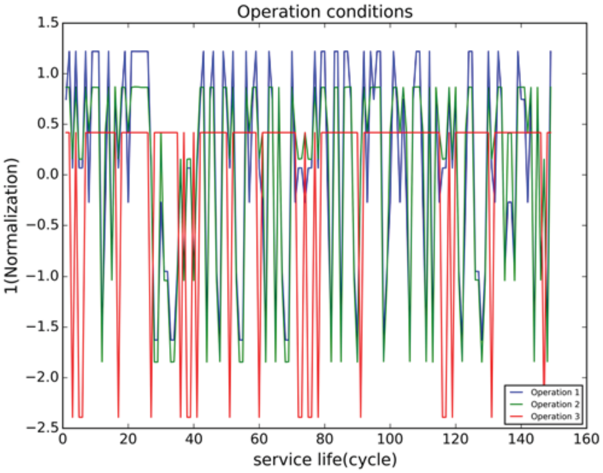

Figure. 2(b) Operation conditions

Each simulated engine was given some initial level of wear which would be considered within normal limits, and faults were initiated at some random time during the simulation. Engine health was determined as the minimum health margin of the rotating equipment, where the health margin was a function of efficiency and flow for that particular component. When this health indicator reached zero, the simulated engine was considered failed.

A. Sample labeling

- RUL labeling: since the degradation in a system will generally not be obvious until after the system or component has been operated for some period of time and the initial failure had developed, it is probably unreasonable to assess RUL until the system begins to degrade. For the time series samples in training set, the first 30% of the sample points are considered to be healthy samples, and the last 30% of the samples are considered to be abnormal samples. Basing that, we trained a support vector machine (SVM) as anomaly detector to find the time faults occur. In addition, we design a mechanism, which will make an abnormal judgment only if consecutive anomaly warnings of three times are issued, through which we can get rid of disturbance to some degree. For each sequence sample, after finding the initial abnormal point, we label the RULs before this by the real RUL value at the critical point while the RULs after that are considered decreasing as a power function way. Some power values (1, 1.1, 1.2 and 2) were tested (not listed), the 1.1 finally won.

- Fault labeling: similarly, we can use the SVM anomaly detector to label faults` categories for every time step in each sequence. Specifically, for each sequence sample, its fault types causing the degradation process have been given by data provider. Assuming that multiple faults occur simultaneously in the operation process (Of course, if data provider can supply more details about when faults occur, we can complete labeling work in a more natural way without this simultaneity hypothesis), firstly we create a binary vector for every time point, whose dimension is equal to the number of known fault categories (two in this

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.

paper) and each element represents a class of fault; Secondly, as for the time points before the critical point mentioned above, we set all elements with "0" (means healthy) in the vectors, while as to the time points after that, we set "1" at the corresponding position representing current actual fault category.

B. Data preprocessing

- Input normalization: We used mean and variance to normalize the input data. It is worth noting in some cases that the number of operation conditions or sensor readings are likely to be constant, which can cause the related variances are zeroes. In these cases, just give up the operation of dividing by variance.

- Output normalization: the output of RUL estimate values are generally in the range of 1-1000 approximately, while the fault diagnosis results described by probability values are in the range of 0-1. In order to ensure the balance of the cost function, we try to normalize it in the way of dividing the RULs by maximum life span, average life span and minimum life span respectively. Test shows the method of minimum life span achieves the best performance (not listed).

IV. Experiments

A. Software and hardware environment

- Experiments are run on a personal computer with Intel Core i7-4500U (1.80GHz) CPU, 4GB memory and Ubuntu 14.04 (Linux) operation system.

- All codes is written by Python 2.7 with scientific computing library "Theano" (0.8.2) [19, 20] and deep learning library "Keras" (1.0.2) [21].

B. Neural network configuration

The complete framework of neural network is depicted in Fig 3.

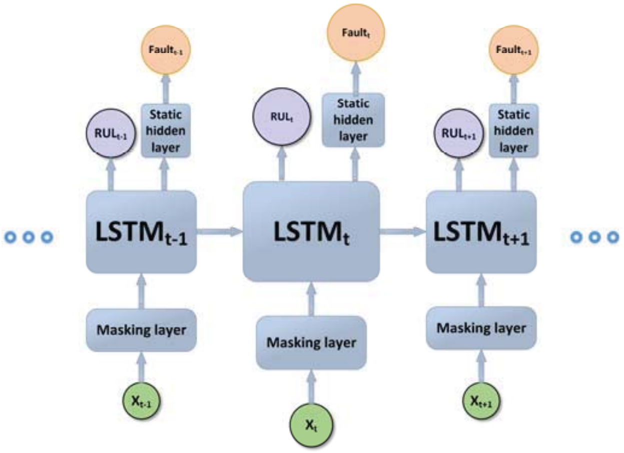

Figure.3. Complete neural network architecture

Input X is organized as a three dimensional array, whose first dimension represents the sample number, second dimension represents the time step and the third represents the features (sensor values and operation records). The mask layer is added above the input layer in order to make all sequence samples have the same length of the second dimension by padding 0s to the beginning of samples whose lives are shorter than the maximum life in the whole dataset. Then data flow through LSTM units just like Fig 1 and arrive at output layers eventually. There are two output layers. One is for RUL estimate, and other one is for fault diagnosis before which there is extra static hidden layer to improve classification ability.

C. Other configurations

- Data set partitioning: According to the ratio of 7:2:1, the data set is divided into training set, cross validation set and test set.

- Cost function: We take Mean Square Error (MSE) as the cost function in training process either for RUL output or fault diagnosis output.

- Regularization: To prevent model from overfitting, we take dropout and training process earlystopping mechanism. Probability of dropping out neurons at output layer of LSTM are set to 0.5. And the monitor object of earlystopping is the performance of cross validation set.

- Optimizer: Adam [22] are used to carry out all experiments in this paper.

D. Test results and discussion

To alleviate the effect of local optimal solution, we repeated the training process three times for each test.

1) RUL estimation results

In order to illustrate the performance of RUL estimator, two indices are considered. One of them is relative error (Specifically, the percentages of samples whose relative errors are less than or equal to 5%, 10% and 20% respectively in the test set). The other one is a standard asymmetric scoring function proposed in [18]. It can be written as follows

$$d = RUL_{estimated} - RUL_{actual} \tag{7}$$

$$score = \begin{cases} e^{-d/13} - 1, d < 0 \\ e^{-d/10} - 1, d \geq 0 \end{cases} \tag{8}$$

In this case, an ideal RUL estimator is one who achieves small relative errors and low values of the scoring function on the test set.

The results of comparison among three types of RNNs (simple RNN, GRU LSTM and standard LSTM) and an ensemble method of LSTM with AdaBoost.R2 (denoted by AdaBoost-LSTM) are listed in Table 1. All the terms are the best results for models mentioned above respectively gained by adjusting hyperparameters (the number of hidden neuron, for example) and repeated training.

For convenience, in the Table 1, 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.

From Fig 4-7 we can see more intuitive prediction results of standard LSTM about issue 1-4 respectively. The blue solid curves represent the learning objective of neural network, namely the real RULs. The green solid curves are the estimates of neural network. The red solid curves mean the

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.

TABLE 1 PERFORMANCE ON TEST SET

Performance Architecture	50 cycles left				10 cycles left				5 cycles left				Completely failure			
	$P_{0.05}$ (%)	$P_{0.1}$ (%)	$P_{0.2}$ (%)	*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	45.71	64.28	87.14	16.93	62.85	90.00	**100.00**	2.96	61.42	90.00	**100.00**	2.94	51.42	77.14	**100.00**	3.97
GRU	**50.00**	64.28	88.57	15.24	61.42	87.14	**100.00**	2.40	58.57	91.42	**100.00**	2.53	55.71	90.00	**100.00**	2.78
AdaBoost-LSTM[a]	**50.00**	**74.00**	90.00	12.55	76.00	90.00	96.00	5.84	76.00	90.00	98.00	4.11	76.00	88.00	96.00	4.09
Standard LSTM	47.14	70.0	**91.42**	**10.63**	**80.00**	**98.57**	98.57	**1.40**	**82.85**	**97.14**	100.00	**1.42**	**80.0**	**95.71**	**100.00**	**1.47**

a. This term is tested on the test set with sample size of 50 instead other terms whose sizes are 70(10% of 709)

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.

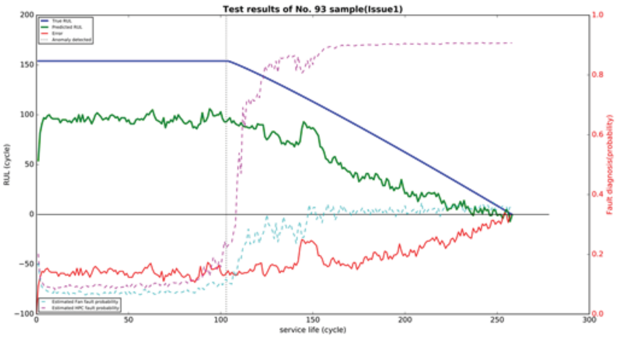

Figure.4. Test results of a random sample in issue 1

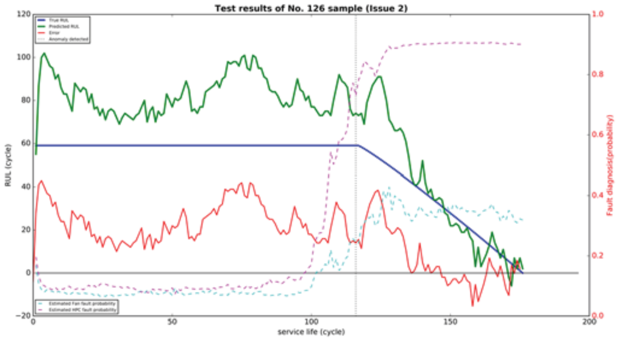

Figure.5. Test results of a random sample in issue 2

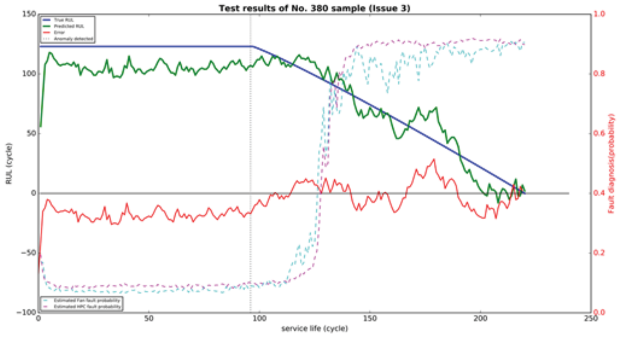

Figure.6. Test results of a random sample in issue 3

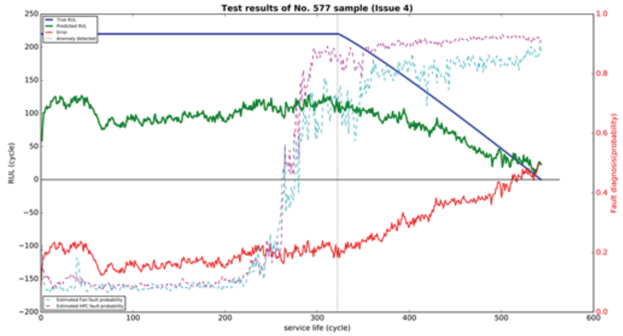

Figure.7. Test results of a random sample in issue 4

errors *d* in (7). The vertical black dashed lines indicate the locations of the anomaly that the SVM found.

2) Fault diagnosis results

As for fault diagnosis, due to lack of real labeled data, we could just display referential evaluation results (gained by the same standard LSTM in Part *1)* of this section).

In Fig 4-7, the results of fault diagnosis are given in the form of probability statement. The probability of fault onset of the high-pressure compressor is given by the magenta dashed lines while the occurrence probability of fan failure is given by the cyan dashed lines.

3) Discussion

- According to the first half of the remaining life curves (before abnormality occurs), LSTM has successfully learnt the method to automatically evaluate the lives of these engines under healthy state as showed in Fig4-7. In fact, the large errors in this period are of no great significance;

- During degradation process (the last half of the remaining life curves), the estimation errors of RUL are continuously reducing generally (except the sample in Fig 4, its estimates are always very accurate). Here, we can believe this is due to the more and more obvious changes of sensor values caused by model degradations as time goes on. It provides more and more significant information for the learning algorithm to make the correct judgment of life. This assumption can be confirmed by the data in Table 1 where relative errors and score functions are gradually reducing with time lapse;

- LSTMs, including standard version and GRU, outperform standard RNN either in long term prediction or short term prediction. It is a pity that the ensemble method of AdaBoost.R2 seems not to make LSTM better. This may be because LSTM itself is already a strong learning machine. Thus, AdaBoost.R2 can reduce the generalization performance of learning model to pay much attention to some large error samples, which overemphasizes noises and other unexpected circumstances.

- For fault diagnosis, on one hand, the abnormal points located by LSTM, which are marked by the rapid rise of the assessment probability for some fault(s), are essentially in agreement with the results gained by SVM (showed in Fig 4-7). On the other hand, the LSTM can also find the correct failure modes either single fault or hybrid faults. In this way, if maintenance staff set reasonable alert thresholds (determined by engineering practice) for every kind of known fault in advance, individual fault tolerance limits (probability values) can be easily designed.

- From Fig4 and Fig 5, which represent samples under high pressure compressor fault, we can find the other assessment probability for the fan fault is generally not close to zero. This may be caused by two reasons: (1) skewness of training samples (there are more training samples suffer high pressure compressor fault than fan fault); (2) Both high pressure compressor and fan are rotating components of aero engine. Therefore, we have reasons to think that there are some certain correlations between these two faults.

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.

V. Conclusion

In this paper, LSTM neural networks, which usually work effectively in the field of natural language processing, are utilized to solve the bottleneck problem of high-precision prognostics and health management challenges about aero engine. This deep neural network is able to provide accurate residual life prediction along with the probability of fault occurrence under complex operation modes and hybrid degradations. This thanks to the excellent ability of LSTM——historical data regulation, especially effectively for long term records. The performances of proposed methods are benchmarked with standard RNN，GRU LSTM (a simplified version of LSTM proposed recently) and AdaBoost-LSTM for four types of problems of aircraft turbofan engines. In all cases, standard LSTM shows enhanced performance. In this paper, we use the fault labels created by anomaly detector (SVM). If we get labeled training data in the future study, we will be able to assess performance with quantitative indicators (the accuracy of fault diagnosis, for example).In addition, the problem of skewness in training samples may be relieved by supplying more samples to failure modes who have less available samples. Furthermore, this can be completed by more artificial tagging efforts of data provider or resampling techniques for scarce samples.

Acknowledgment

I would like to thank NASA Ames Research Center for providing turbofan engine degradation simulation data set. I would like to thank my supervisor Mei Yuan for taking time to review this work and my junior sister apprentice Lin Li for her a great deal of work in the search for best hyperparameters of neural network models.

References

[1] Coble JB. Merging Data Sources to Predict Remaining Useful Life – An Automated Method to Identify Prognostic Parameters., vol. The Doctor of Philosophy Degree: University of Tennessee, Knoxville, 2010.

[2] Sikorska JZ, Hodkiewicz M, Ma L. Prognostic modelling options for remaining useful life estimation by industry. MECH SYST SIGNAL PR 2011;25:1803.

[3] Xun-kai W, Ji-hong Z, Liang-feng C, Yue F, Li Y. The Opportunities and Challenges of Data Mining in Gas Turbine Engine Prognostics and Health Management. COMPUTER ENGINEERING & SCIENCE 2012;34:88.

[4] Peel L. Data driven prognostics using a Kalman filter ensemble of neural network models.: IEEE, 2008. p.1.

[5] Heimes FO. Recurrent Neural Networks for Remaining Useful Life Estimation.., 2008. p.59.

[6] Bengio Y, Simard P, Frasconi P. Learning Long-Short Term dependency is difficult. IEEE Transactions ON Neural Networks 1994;5:157.

[7] Hochreiter S, Schmidhuber JUR. Long short-term memory. NEURAL COMPUT 1997;9:1735.

[8] LeCun Y, Bengio Y, Hinton G. Deep learning. NATURE 2015;521:436.

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

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

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

[12] Sutskever I, Hinton GE, Taylor GW. The Recurrent Temporal Restricted Boltzmann Machine. 2008:1601.

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

[14] Yu W, Zhuang F, He Q, Shi Z. Learning deep representations via extreme learning machines. NEUROCOMPUTING 2015;149:308.

[15] Tamilselvan P, Wang P. Failure diagnosis using deep belief learning based health state classification. RELIAB ENG SYST SAFE 2013;115:124.

[16] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. NEURAL NETWORKS 2005;18:602.

[17] Greff K, Srivastava RK, Koutník J, Steunebrink BR, Schmidhuber J. LSTM: A Search Space Odyssey. 2015:10, 8.

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

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

[20] Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, Turian J, Warde-Farley D, Bengio Y. Theano: A CPU and GPU Math Compiler in Python. Proceedings of the Python for Scientific Computing Conference (SciPy). Austin, TX, 2010.

[21] Chollet F. Keras. GitHub repository（https://github.com/fchollet/keras）, 2015.

[22] Kingma D, Ba J. 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.

Authorized licensed use limited to: Amazon. Downloaded on June 05,2020 at 09:59:26 UTC from IEEE Xplore. Restrictions apply.