Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Gated Recurrent Unit Networks for Remaining Useful Life Prediction ⋆

Li Li [*,**], **Zhen Zhao** [*], **Xiaoxiao Zhao** [*], **Kuo-Yi Lin** [*,**]

[*] *College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China*
[**] *Institute of Intelligent Science and Technology, Tongji University, Shanghai 201203, China*
(e-mail: lili@tongji.edu.cn, 1730718@tongji.edu.cn, zh_xiaoxiao@tongji.edu.cn, 19603@tongji.edu.cn)

Abstract: Remaining useful life prediction is a key procedure for prognostics and health management. However, traditional data-driven methods rely on handcrafted feature selection from the whole range of time series data, which may not obtain the temporal information for complex systems. This study proposes a gated recurrent unit networks based approach to predict remaining useful life. First, time window approach is applied on sample preparation for multiple sensor data. In particular, unsupervised stacked sparse autoencoder is utilized to automatically extract nonlinear features, then the selected features are fed into gated recurrent unit based recurrent neural networks to predict remaining useful life. The effectiveness of the proposed method is demonstrated on the commercial modular aero-propulsion system simulation data from NASA. Experimental results validate that the proposed approach achieves better prediction performance than other methods.

Keywords: prognostics, remaining useful life, autoencoder, gated recurrent unit.

1. INTRODUCTION

Prognostics and health management (PHM) play an important role in predictive maintenance and have been receiving increasing attention in the past decades (Tao et al. (2018)). PHM can maximize the operation availability, reduce the maintenance costs and improve the system safety and reliability. Specially, remaining useful life (RUL) prediction is one of the most significant tasks in PHM.

There are different ways to categorize the RUL prediction methods. Generally, the existing approaches for RUL prediction can be classified into three categories: model-based approaches, data-driven approaches and hybrid approaches. Model-based approaches use physical mechanisms or mathematical functions to model the degradation and failure process of a system or component. Common model-based approaches include state space algorithms, such as extended Kalman filter (EKF) and particle filter (Baraldi et al. (2012)), and classical deterioration methods such as Weibull distribution model (Liu et al. (2020)). However, they require extensive expert knowledge, and it is very challenging to obtain prior knowledge in practice due to system complexity and stochastic degradation behavior of components. Data-driven approaches estimate RUL using information from condition monitoring (CM) data collected by various sensors, and they can be easily applied and generalized in practical applications. Different kinds of data-driven algorithms have been proposed and obtained efficient results. Artificial intelligent methods include neuro-fuzzy (NF) systems, artificial neural networks (ANN), support vector machines (SVM) and Gaussian process regression (GPR) (Lei et al. (2018)). Most of these methods consist of two stages: an offline learning with the feature extractor and the degradation state learner; an online stage for RUL prediction via the learned model. But traditional data-driven approaches rely heavily on manual feature extraction. A hybrid approach attempts to integrate advantages of different approaches. Acuña and Orchard (2017) used different methods to construct a degradation model and combined it with particle filter for RUL prediction. Du et al. (2012) predicted RULs using several prediction methods and made a final decision through fusion strategies. It is still difficult to find an effective hybrid approach for the combination of model-based and data-driven approaches.

Recently, deep learning is emerging a highly effective networks applied for many applications, such as image recognition, object detection, natural language processing and speech recognition fields (LeCun et al. (2015)). Since the raw data obtained from machinery health monitoring share similar high dimensionality with those in image processing, deep learning has great potential in RUL prediction (Khan and Yairi (2018), Zhao et al. (2019)). Moreover, layer-by-layer feature learning in deep networks can learn essential and representative features hidden in condition monitoring data. Zhang et al. (2016) proposed a multi-objective deep belief networks (DBN) ensemble

⋆ This research has been supported by the Key Research and Development Project of National Ministry of Science and Technology under grant no. 2018YFB1305304, the National Natural Science Foundation of China under grant no. 61873191 and no. 51475334

method, and the evolutionary algorithm is integrated with traditional DBN training to evolve multiple DBNs simultaneously subject to accuracy and diversity. Zhu et al. (2018) presented a new deep feature learning method for RUL estimation through time frequency representation (TFR) and multiscale convolutional neural network (MSCNN). Li et al. (2018) proposed a deep convolutional neural network for RUL estimation using raw collected data. However, its difficult to extract heterogeneous features indicating the variation until the failure of a machine.

Recurrent neural networks (RNN) are widely used in RUL prediction because of its ability in dealing with explicit time series data. Guo et al. (2017) proposed a recurrent neural network based health indicator (RNN-HI) for RUL prediction of bearings, which can obtain high monotonicity and correlation values and can be beneficial to bear RUL prediction. However, RNN cannot link two similar data if they are separated too far away. Long short term memory (LSTM) is proposed to overcome the weakness of RNN, which incorporates input gate, output gate and cell state into RNN. To deal with uncertainty due to operational and environmental disturbance, Zhang et al. (2018) proposed a bi-directional LSTM for characterizing system degradation behaviour and predicting RUL, in which information flow through the LSTM cells forwards for prediction and backwards for ruling out the disturbance and smoothing the tracking. Huang et al. (2019) developed a novel prognostics method based on bi-directional LSTM, which integrated multiple sensors data with operational condition data for RUL prediction. Wu et al. (2018) proposed utilizing vanilla LSTM neural networks to predict RUL for complicated systems under dynamic operational modes, and a dynamic differential technology was used to extract inter-frame information to find the real physical degradation mechanism behind sensor readings.

However, each memory blocks in LSTM requires an input gate and an output gate, and these gates make the training more difficult and increase the training time. A simplified variant of LSTM named gated recurrent unit (GRU) is proposed to reduce training time and improve network performance (Tang et al. (2015)). The GRU combines the forget gate and the input gate into a single update gate and mixes the cell state and the hidden state into one state as well.

Inspired by these prior researches, this paper presents a GRU based approach for RUL prediction. First, unsupervised stacked sparse autoencoder is applied to automatically extract nonlinear features from consecutive time windows of multiple sensor data. Then, the codings of stacked sparse autoencoders are treated as the input of GRU to predict RUL. A case study based on aircraft engines is used to verify the performance of the proposed approach.

The rest of this paper is organized as follows. Section 2 describes the proposed SAE-GRU based regression model for RUL prediction. In Section 3, the proposed approach is applied on the C-MAPSS dataset to demonstrate the effectiveness. Conclusion and future works are drawn in Section 4.

NOMENCLATURE

Acronyms

RUL	remaining useful life
PHM	prognostics and health management
SAE	sparse autoencoder
RNN	recurrent neural network
LSTM	long short term memory
GRU	gated recurrent unit
FNN	feedforward neural network
MAE	mean absolute error

Notation

n	number of data samples
$\boldsymbol{x}$	input data(sensor data)
$\boldsymbol{y}$	system states
$\boldsymbol{y'}$	predicted states
$y_{threshold}$	a predefined failure threshold
RUL_{actual}	actual RUL value
RUL_{pred}	predicted RUL value
$\boldsymbol{h}$	hidden state
$\tilde{\boldsymbol{x}}$	reconstructed input
$\boldsymbol{W}, \boldsymbol{V}$	weights of network
$\boldsymbol{b}$	the bias
g, f	activation functions
J	cost function
s_l	number of neurons in l-th layer
λ	weight decay parameter
ρ	sparsity parameter
r	reset gate
z	update gate
N_{tw}	size of time window
R^2	coefficient of determination

2. METHODOLOGY

2.1 Problem formulation

Giving sensor data collected from n time steps $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$ and corresponding system states $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]$, the RUL prediction is to find $\boldsymbol{y}$ over time through exploring the variation of sensor data, it can be defined as,

$$\begin{aligned} \boldsymbol{x}_n &= F\left[\boldsymbol{x}_{n-1}, \boldsymbol{x}_{n-2}, \ldots, \boldsymbol{x}_1\right] \\ \Rightarrow y_n &= G\left[y_{n-1}, y_{n-2}, \ldots, y_1\right] \end{aligned} \tag{1}$$

which means current state performance is associated with its previous state performance. Through machine learning techniques, the predicted states are labelled as follows,

$$\boldsymbol{y'} = \left[y'_{n+1}, y'_{n+2}, \ldots\right] \tag{2}$$

where $y'_{n+1} = G\left[y_n, y_{n-1}, \ldots, y_1\right]$.

Given a predefined failure threshold, the predicted RUL can be defined as,

$$RUL_{pred} = \inf\left\{k : y'_{n+k} \leq y_{threshold}\right\} \tag{3}$$

The main challenge of this study is to capture non-linearity associated with the system variation and data uncertainty (e.g. sensor failure or environmental changes). To handle with the non-linearity, stacked sparse autoencoder is presented in this study as well as GRU, which has better performance in dealing with time series data.

2.2 Sparse autoencoder

Autoencoder is an artificial neural network for unsupervised learning of feature extraction and dimensionality reduction (Bengio et al. (2013)). An autoencoder is generally composed of encoder and decoder. The network architecture of an autoencoder with 5 layers is shown in Fig 1.

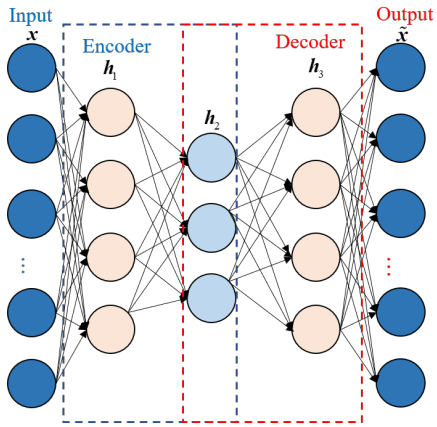


Fig. 1. Network architecture of autoencoder with 5 layers

Assume that $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]$ is the original input data consisting of n data samples, f and g are the activation functions for encoding and decoding respectively. The hidden representation $\boldsymbol{h}$ after encoding is shown as follows,

$$\boldsymbol{h_1} = f(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}), \boldsymbol{h_2} = f(\boldsymbol{W}^{(2)}\boldsymbol{h_1} + \boldsymbol{b}^{(2)}) \tag{4}$$

and the reconstructed output $\tilde{\boldsymbol{x}}$ after decoding is calculated as follows,

$$\tilde{\boldsymbol{x}} = g(\boldsymbol{W}^{(4)}\boldsymbol{h_3} + \boldsymbol{b}^{(4)}), \tag{5}$$

where $\boldsymbol{h_3} = g(\boldsymbol{W}^{(3)}\boldsymbol{h_2} + \boldsymbol{b}^{(3)})$, $\boldsymbol{W}$, $\boldsymbol{b}$ are the weights and bias of networks.

The aim of autoencoder is to minimize reconstruction error between the original input $\boldsymbol{x_i}$ and the reconstructed output $\tilde{\boldsymbol{x}}_i$ from the hidden representation $\boldsymbol{h}$. The cost function is defined as follows,

$$\begin{aligned} J(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{2}\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|^2\right) \\ +\frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\boldsymbol{W}_{ij}^{(l)}\right)^2 \end{aligned} \tag{6}$$

where $J(\boldsymbol{W}, \boldsymbol{b})$ is the cost function, n_l is the number of layers in the network, and s_l is the number of neurons in l-th layer. The first term is the reconstruction error, and the second term is the regularization term to prevent overfitting. The weight decay parameter λ controls the relative importance of the two terms.

The autoencoder just simply copies the input. Although the learned feature representation may perfectly reconstruct the original input data, the features are redundant and are not representative enough. Thus, the cost function of autoencoder is added with a sparsity penalty term. Sparse autoencoder (SAE), which has great potential to learn more abstract and representative compressed features than autoencoder. In a hidden layer, averaged activation of j-th neuron over the dataset is expressed as,

$$\rho_j = \frac{1}{n}\sum_{i=1}^{n} h_j(\boldsymbol{x}_i), j = 1, \ldots, s_l \tag{7}$$

A sparse parameter ρ is given to limit activation of hidden-layer, and overall constraint for all neurons in l-th hidden layer is expressed as follows,

$$\sum_{j=1}^{s_l}\mathrm{KL}(\rho\,\|\rho_j) = \sum_{j=1}^{s_l}\rho\log\frac{\rho}{\rho_j} + (1-\rho)\log\frac{(1-\rho)}{(1-\rho_j)} \tag{8}$$

The $\mathrm{KL}(\rho\,\|\rho_j)$ is Kullback-Leibler (KL) divergence between a Bernoulli random variable with mean ρ and a Bernoulli random variable with mean ρ_j, and activation of neurons ρ_j is made to be close to the given sparse parameter ρ with this constraint. Considering sparse constraint, cost function of SAE is defined as follows,

$$J_{sparse}(\boldsymbol{W}, \boldsymbol{b}) = J(\boldsymbol{W}, \boldsymbol{b}) + \beta\sum_{j=1}^{s_l}\mathrm{KL}(\rho\,\|\rho_j) \tag{9}$$

where β controls the weight of the sparsity penalty term. Because SAE can learn more representative features, it is utilized to extract features from sensor data automatically in the proposed method.

2.3 Gated recurrent unit

The basic RNN structure consists of a cell with a cyclic loop whose internal state evolves over time by the current sample input $\boldsymbol{x_t}$ and its previous hidden state $\boldsymbol{h_{t-1}}$ at each time step t , then updates the current hidden state $\boldsymbol{h_t}$ as follows,

$$\boldsymbol{h_t} = H(\boldsymbol{x_t}, \boldsymbol{h_{t-1}}) \tag{10}$$

where H defines a nonlinear and differentiable transformation function. The RNN parameters can be trained using backpropagation. However, it suffers from vanishing gradient problem because the recurrent network grows deep as sequence length. One of the most popular solution to address vanishing gradient is long short term memory (LSTM), which is regarded as one of the most efficient RNN models. LSTM uses a well-designed memory cell, which consists of an input gate, a forget gate and an output gate to protect and update the cell state.

GRU is a simpler variant of LSTM, two gates including a reset gate $\boldsymbol{r}$ that adjusts the incorporation of new input with the previous memory and an update gate $\boldsymbol{z}$ that controls the preservation of the previous memory. The architecture of GRU cell is shown in Fig 2.

GRU contains less variables than LSTM, which makes it more efficient. The transition functions in hidden units of GRU are shown as follows,

$$\begin{aligned} \boldsymbol{z}_t &= \sigma(\boldsymbol{W}^z\boldsymbol{x}_t + \boldsymbol{V}^z\boldsymbol{h}_{t-1} + \boldsymbol{b}^z) \\ \boldsymbol{r}_t &= \sigma(\boldsymbol{W}^r\boldsymbol{x}_t + \boldsymbol{V}^r\boldsymbol{h}_{t-1} + \boldsymbol{b}^r) \end{aligned} \tag{11}$$

the new remember $\tilde{\boldsymbol{h}}_t$ is generated by $\boldsymbol{r}_t$ with a tanh layer,

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}^c\boldsymbol{x}_t + \boldsymbol{V}^c(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})) \tag{12}$$

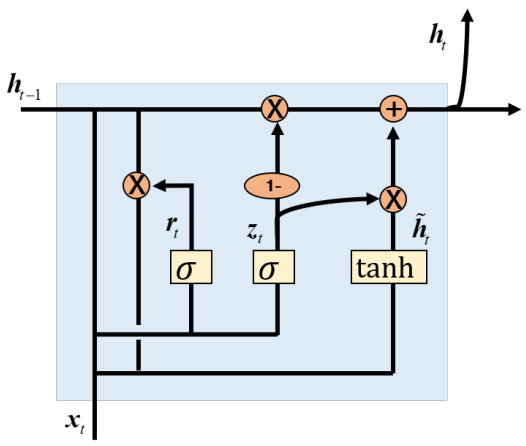


Fig. 2. The architecture of GRU cell

the hidden state value is updated by

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t \tag{13}$$

where model parameters including $\boldsymbol{W}$, $\boldsymbol{V}$ and $\boldsymbol{b}$ are shared by all time steps and learned during training, and $\odot$ represents the element-wise product. Due to its efficiency while achieving similar accuracy, GRU is used to learn latent fault features from time series of sensor data.

2.4 The procedure of proposed method

The flow chart of the proposed RUL prediction approach is presented in Fig 3. As we can see from Fig 3, the proposed approach includes three stages: data preprocessing, feature extraction and prediction model based on SAE-GRU neural networks.

Data preprocessing is applied to improve the data quality. Besides data cleaning, normalization is used to transform these data into a common scale, and a time window strategy is incorporated to convert the multivariate time series data into the desired sequence data. Then the sequence data after using the time window are treated as the input of SAE, and the objective of SAE is dimensionality reduction and nonlinear feature extraction from the normalized data. No prior expert knowledge on prognostics and signal processing is required in the proposed method.

A GRU-FNN model is constructed to handle time series of sensor data, since GRU can learn latent fault features from long-time context and FNN is capable of mapping latent features from GRU into regression outcomes, such as the target RUL of the current time before failure. The codings of SAE are treated as the input of GRU-FNN to obtain the predicted results. In offline stage, the predicted RUL and labels of training data can be used to calculate the loss function to train the proposed SAE-GRU model through back-propagation algorithm. In online stage, after data preprocessing, the testing data samples are fed into the trained models to obtain the predicted RUL values.

3. EXPERIMENTS

3.1 C-MAPSS dataset

The proposed method is evaluated on a prognostic benchmark dataset: NASA's turbofan engine degradation dataset, which contains simulated data produced by a model-based simulation program Commercial Modular Aero-Propulsion System Simulation (C-MAPSS). The C-MAPSS dataset includes 4 sub-datasets: single operational condition, single fault mode issue, multiple operational conditions and hybrid fault modes issue. Each dataset contains a training set and a testing set, where each row is a snapshot of data from engine running cycles. There are 26 columns: the 1st and 2nd columns represent engine ID and running cycle, the next 3 columns are operational conditions indicating that the engine is running in the current operational condition, and the remaining 21 columns are sensor readings corresponding to degradation information (e.g. temperature, pressure, speed) of engines. The detailed information of the C-MAPSS dataset is shown in Table 1.

Table 1. Information of the C-MAPSS dataset

Dataset	FD001	FD002	FD003	FD004
Engine units	100	260	100	249
Operational conditions	1	6	1	6
Fault conditions	1	1	2	2
Maximum life cycles	362	378	525	543
Minimum life cycles	128	128	145	128
Training samples	17731	48819	21820	57522
Testing samples	100	259	100	248

All the experiments are carried out on a PC with Intel Core i7 CPU, 32GB RAM and GEFORCE GTX 1070 GPU. We find the optimal hyperparameters for the model by random search. The Adam optimizer is used with mini batches for the updates of weights in the networks. For each epoch, the batch size is set as 200, the maximum number of epochs is set as 100. The initial learning rate is 0.001. Sparse parameter ρ for the hidden layer of SAE is 0.01. Parameter β is set as 20 to tune sparsity of SAE.

In particular, dropout technique is applied to solve overfitting problem. Dropout randomly drops neurons (along with their connections) from the neural network during training. Each neuron is retained with a fixed probability p independent of other neurons. We set $p = 0.5$, applying dropout to a neural network is also a simple method for model ensemble, which can improve the feature extraction.

3.2 Data Preprocessing

Data normalization Although there are 21 sensors available in the C-MAPSS dataset, not all of them are informative. Some sensor readings provide constant values in the entire lifetime of engines. Therefore, 14 sensor measurements are used as the raw input as suggested in Li et al. (2018). Different ranges of sensor readings have an influence on the accuracy and convergence speed of the model, so the normalization of data is required prior to training and testing, which ensures that all features under different operational conditions and fault modes have an equal contribution. The collected measure data from each sensor are normalized to be within the range of [-1, 1] using the min-max normalization method shown as,

$$x_{norm}^{i,j} = \frac{2(x^{i,j} - x_{\min}^{j})}{x_{\max}^{j} - x_{\min}^{j}} - 1 \tag{14}$$

where $x^{i,j}$ represents the original i-th data point of the j-th sensor, and $x_{norm}^{i,j}$ is the normalized value of $x^{i,j}$. $x_{\min}^{j}$

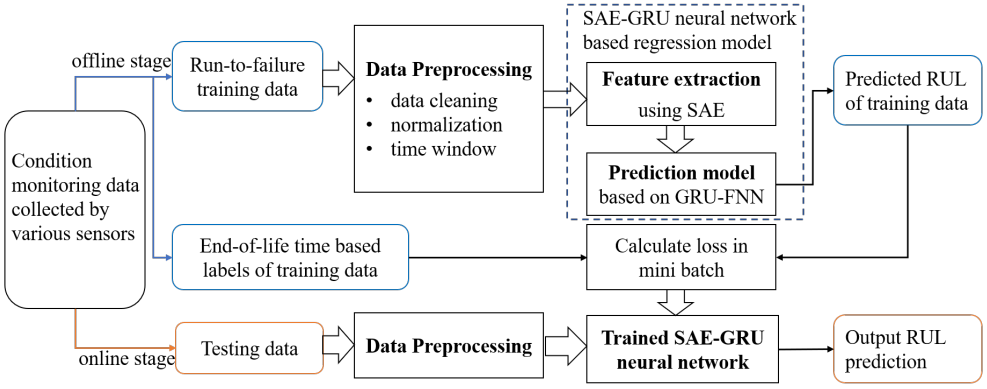
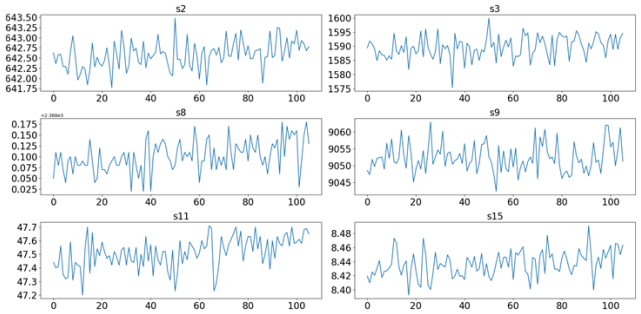
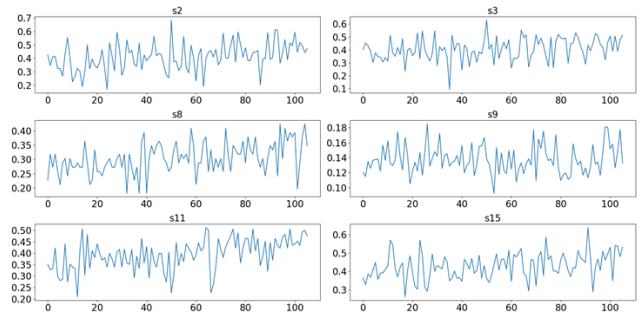


Fig. 3. Flow chart of the proposed SAE-GRU based RUL prediction

and $x^j_{\max}$ represent the minimum and maximum values of the original data from the j-th sensor respectively. Fig 4 shows the original sensor data and its normalized sensor data of engine ID =4.



(a) The original sensor data of Engine ID = 4



(b) The normalized sensor data of Engine ID = 4

Fig. 4. The sensor data of Engine ID=4 before and after normalization

Target RUL Different from common regression problems, the desired output value of the input data is difficult to determine for RUL prediction problem. In many practical industrial applications, it is difficult to evaluate the accurate health condition and estimate the RUL of the component at each time without a precise physical-based model. However, a sensible solution is to simplify actual time before system failure as the desired output. For the C-MAPSS dataset, a piece-wise linear degradation model has been validated to be suitable and effective in Heimes (2008). As shown in Fig 5, the engine unit works normally in the early age, indicating that the initial degradation is ignored, and then it degrades linearly at a certain point, it is assumed that a constant RUL label in the initial period. We set the constant in the first half as 130 (Heimes (2008)).

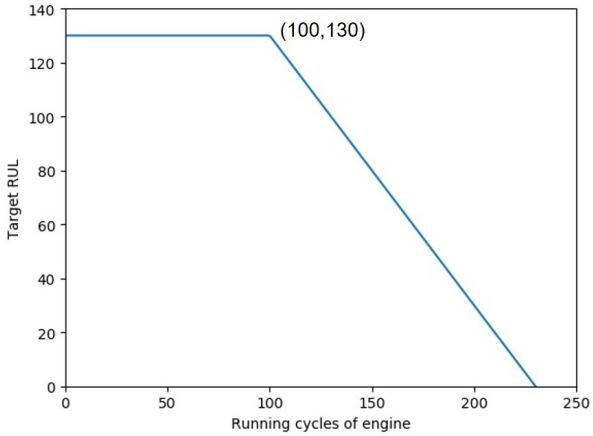


Fig. 5. RUL target function on the C-MAPSS dataset, the initial RUL is set as 130

Time window processing After normalization, a time window strategy is incorporated for data preprocessing. A fixed-length time window is utilized to enclose multivariate data samples at consecutive cycles, then the time window will shift one measurement cycle each time to generate a new time window to the end of life. Finally, each multivariate time window will be taken as the input of networks. Let N_{tw} represent the size of time window, then Fig 6 shows normalized data sample from some selected sensors of a single engine ID =4 within a time window of size 50 in the training set of FD001 subdataset. More information can be obtained from the temporal sequence data in RUL prediction. Therefore time sequence processing can obtain better prediction performance.

3.3 Performance evaluation

Two performance metrics have been used for evaluating the performance of different prognostic methods, the mean absolute error (MAE) and coefficient of determination R^2. The formulation of MAE is shown as follows.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|d_i| \tag{15}$$

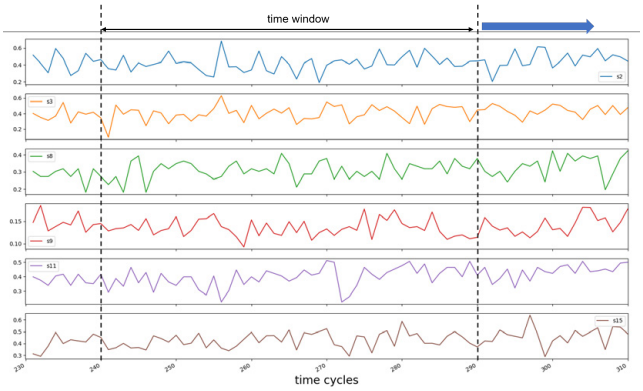


Fig. 6. Some features of Engine ID=4 using a time window of length 50

where n is the number of samples, $d_i = RUL_{pred} - RUL_{actual}$, which is the error between predicted RUL value and actual RUL value for the i-th testing data sample.

3.4 Experimental results

The performance comparison results of different methods on the C-MAPSS datasets are shown in Table 2. It can be observed from Table 2 that the GRU method achieves the best performance in most cases.

Table 2. Performance comparison of different methods on the C-MAPSS dataset

Dataset	Metrics	LSTM	Bi-LSTM	Proposed
FD001	MAE	13.5581	14.3862	**13.4545**
	R^2	0.7682	0.7555	**0.7886**
FD002	MAE	44.2737	**19.9247**	22.0161
	R^2	-0.0484	**0.7228**	0.6896
FD003	MAE	17.0519	20.8028	**16.3749**
	R^2	0.5070	0.2482	**0.5697**
FD004	MAE	47.9062	28.0477	**25.4020**
	R^2	-0.2043	0.5331	**0.5614**

The RUL prediction results of the testing engine units regarding the last recorded data points in FD001 subdataset is shown in Fig 7. As we can see from Fig 7, the predicted RUL values by the proposed method are very close to the actual values for many units. When the RUL value is small, which means the engine unit is working close to the failure point, the predicted accuracy is higher. Because the extracted features can represent more accurate information for better prediction.

Also, the RUL prediction results of the testing engine units in FD002, FD003 and FD004 are shown in Fig 8, Fig 9 and Fig 10 respectively. Just like the results in FD001 subdataset, although there are some errors between the predicted RUL values and actual RUL values, the predicted accuracy is high when the engine units are close to the failure point. So the late period of life time is significant for maintenance and health management.

We further analyse the influence of time window size on prediction performance. The performance metric MAE using different time window size N_{tw} in FD001 subdataset is shown in Fig 11. It should be noted that in the test

Fig. 7. RUL prediction for 100 testing engine units in FD001 subdataset

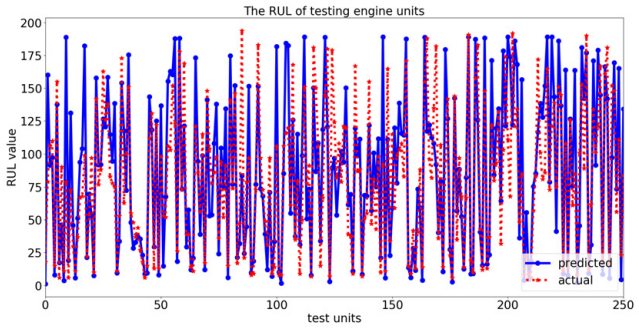


Fig. 8. RUL prediction for 250 testing engine units in FD002 subdataset

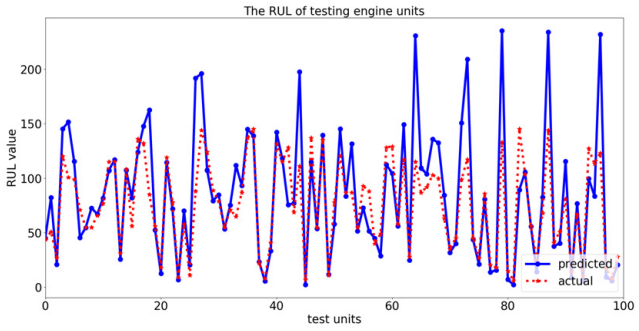


Fig. 9. RUL prediction for 100 testing engine units in FD003 subdataset

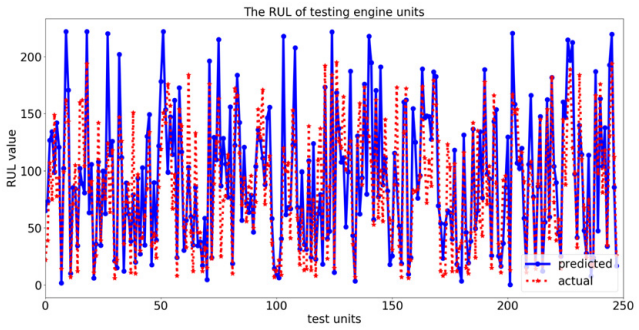


Fig. 10. RUL prediction for 249 testing engine units in FD004 subdataset

dataset, the data cycles of testing units have different lengths. The testing units which have shorter cycles than N_{tw} are removed, so as to provide a more comprehensive analysis of the time window size.

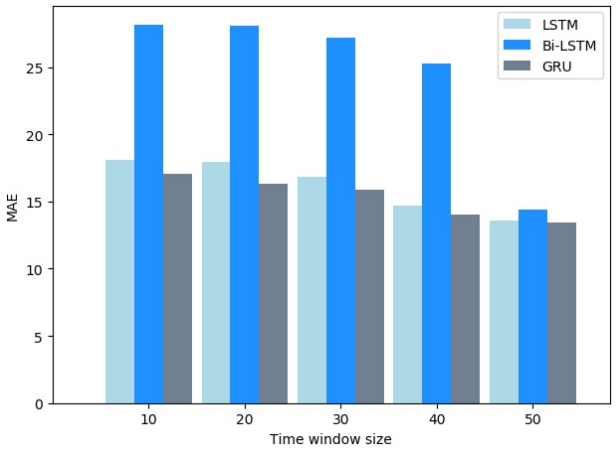


Fig. 11. MAE using different time window sizes for 100 testing engine units in FD001 subdataset

As we can see from Fig 11, larger time window size has better prediction for all RNN-based approaches. Specially, the proposed GRU method shows superiority than others within different sizes of time window. More information can be contained through larger time window, which can be beneficial for extracting informative features. However, the training time of networks increases as the time window size rises. Therefore, we set the $N_{tw} = 50$ in FD001 subdataset to balance the performance and efficiency. The $N_{tw} = 20, 30, 15$ are set for FD002, FD003 and FD004 subdatasets respectively.

4. CONCLUSION

This paper provides a general framework based on GRU for RUL prediction, which includes three stages: data preprocessing, feature extraction and prediction model based on GRU neural networks. After nonlinear feature extraction using the stacked sparse autoencoder, a GRU-FNN network is adopted to learn the representation of sequence of those features and estimate the RUL values. The effectiveness of proposed approach has been verified on C-MAPSS data. We also explore the influence of the time window size to obtain the trade off between accuracy and efficiency. In the future, we will focus on uncertainty propagation and management to improve the accuracy and robustness.

REFERENCES

Acuña, D.E. and Orchard, M.E. (2017). Particle-filtering-based failure prognosis via sigma-points: Application to lithium-ion battery state-of-charge monitoring. *Mechanical Systems and Signal Processing*, 85, 827–848.

Baraldi, P., Mangili, F., and Zio, E. (2012). A kalman filter-based ensemble approach with application to turbine creep prognostics. *IEEE Transactions on Reliability*, 61(4), 966–977.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

Du, S., Lv, J., and Xi, L. (2012). Degradation process prediction for rotational machinery based on hybrid intelligent model. *Robotics and Computer Integrated Manufacturing*, 28(2), 190–207.

Guo, L., Li, N., Jia, F., Lei, Y., and Lin, J. (2017). A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing*, 240, 98–109.

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

Huang, C.G., Huang, H.Z., and Li, Y.F. (2019). A bidirectional lstm prognostics method under multiple operational conditions. *IEEE Transactions on Industrial Electronics*, 66(11), 8792–8802.

Khan, S. and Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, 107, 241–265.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

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

Li, X., Ding, Q., and Sun, J.Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, 172, 1–11.

Liu, B., Do, P., Iung, B., and Xie, M. (2020). Stochastic filtering approach for condition-based maintenance considering sensor degradation. *IEEE Transactions on Automation Science and Engineering*, 17(1), 177–190.

Tang, D., Qin, B., and Liu, T. (2015). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1422–1432.

Tao, F., Qi, Q., Liu, A., and Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48, 157–169.

Wu, Y., Yuan, M., Dong, S., Lin, L., and Liu, Y. (2018). Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275, 167–179.

Zhang, C., Lim, P., Qin, A.K., and Tan, K.C. (2016). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and learning systems*, 28(10), 2306–2318.

Zhang, J., Wang, P., Yan, R., and Gao, R.X. (2018). Long short-term memory for machine remaining life prediction. *Journal of Manufacturing Systems*, 48, 78–86.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., and Gao, R.X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, 115, 213–237.

Zhu, J., Chen, N., and Peng, W. (2018). Estimation of bearing remaining useful life based on multiscale convolutional neural network. *IEEE Transactions on Industrial Electronics*, 66(4), 3208–3216.