Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-021-01750-x

Remaining useful life estimation via transformer encoder enhanced by a gated convolutional unit

Yu Mo[1,2] · Qianhui Wu[1] · Xiu Li[2] · Biqing Huang[1]

Received: 30 May 2020 / Accepted: 18 February 2021

Abstract
Remaining Useful Life (RUL) estimation is a fundamental task in the prognostic and health management (PHM) of industrial equipment and systems. To this end, we propose a novel approach for RUL estimation in this paper, based on deep neural architecture due to its great success in sequence learning. Specifically, we take the Transformer encoder as the backbone of our model to capture short- and long-term dependencies in a time sequence. Compared with convolutional neural network based methods, there is no limitation from the kernel size for a complete receptive field of all time steps. While compared with recurrent neural network based methods, we develop our model based on dot-product self-attention, enabling it to fully exploit parallel computation. Moreover, we further propose a gated convolutional unit to facilitate the model's ability of incorporating local contexts at each time step, for the attention mechanism used in the Transformer encoder makes the output high-level features insensitive to local contexts. We conduct experiments on the C-MAPSS datasets and show that, the performance of our model is superior or comparable to those of other existing methods. We also carry out ablation studies and demonstrate the necessity and effectiveness of each component used in the proposed model.

Keywords Remaining useful life estimation · Transformer encoder · Gated convolutional unit · Turbofan engine

Introduction

Prognostic and health management (PHM) plays an important role in promoting the transition of industrial systems from failure-based corrective maintenance and time-based preventive maintenance to condition-based predictive maintenance (Peng et al. 2010), which can help avoid unscheduled downtime, reduce maintenance costs and complexity, so as to increase productivity and improve profitability. Remaining useful life (RUL) estimation is one of the major tasks in PHM. It aims to estimate the time interval between the current moment to the end of a system's service life, and has attracted considerable attention from the research community in recent years.

Traditional physics-based methods for RUL estimation mostly employ mathematical descriptions such as algebraic equations and differential equations, to model the degradation process of a system, for example, the Paris-Erdogan model (Li et al. 1999), the Forman crack growth law (Oppenheimer and Loparo 2002), and the three-dimensional fracture mechanics lifting model (Kacprzynski et al. 2004). However, such methods heavily rely on prior knowledge of system degradation mechanics and few systems can be modeled accurately by solvable mathematical models due to the complexity of the real world. With the advance in sensor techniques, the scale of system condition monitoring data is gradually increasing, and thus a variety of data-driven models have been studied. These methods intend to explore the underlying relation between the sensor monitoring data and the system degradation state by training designed models, such as support vector machine

Yu Mo and Qianhui Wu are contributed equally to this work.

✉ Xiu Li
li.xiu@sz.tsinghua.edu.cn

✉ Biqing Huang
hbq@tsinghua.edu.cn

Yu Mo
mo-y20@mails.tsinghua.edu.cn

Qianhui Wu
wuqianhui@tsinghua.org.cn

[1] Beijing National Research Center for Information Science and Technology, Department of Automation, Tsinghua University, Beijing 100084, China

[2] Division of Information Science, Tsinghua Shenzhen International Graduate School, Shenzhen 518055, China

Published online: 15 March 2021

(SVM) (Nieto et al. 2015; Wang and Chen 2014), support vector regression (SVR) (Benkedjouh et al., 2015), hidden Markov model (HMM) (Tobon-Mejia et al. 2011, 2012), Bayesian belief networks (Zhang et al. 2014; Mosallam et al. 2016; Zhang et al. 2016), artificial neural networks (Huang et al. 2007; Tian et al. 2010; Tian 2012), *etc*. Compared with physics-based methods, data-driven models require less prior knowledge and show better generalization ability, and thus are widely used in industrial applications.

With the increase of data volume and computing power, deep learning has become a research hotspot, and has brought great success to the field of computer vision (Voulodimos et al. 2018) and natural language processing (Fahad and Yahya 2018). Meanwhile, the research interests of data-driven methods for RUL estimation are also shifting from conventional machine learning models to deep neural architectures. For instance, some early methods employ traditional recurrent neural network (RNN) to model time series in an autoregressive fashion (Malhi et al. 2011). However, RNNs are difficult to train due to the issue of gradient vanishing and exploding, which makes the model lack the ability of learning long-term dependencies. To tackle that, the long short-term memory (LSTM) network (Hochreiter and Schmidhuber 1997) and the gated recurrent unit (GRU) (Cho et al. 2014) introduce several gates to control the information flow, and several prior works for RUL estimation were carried out by using these gated recurrent networks (Yuan et al. 2016; Zheng et al. 2017; Guo et al. 2017; Wu et al. 2018). Instead of learning in a recurrent manner, some other approaches apply convolutional neural network (CNN) based model for RUL estimation and achieve quite competitive performance (Babu et al., 2016; Li et al. 2018; San Kim andSohn 2020).

However, there are limitations in both RNN based methods and CNN based methods. For the former, RNNs process historical sensor data in a sequential manner, which inherently precludes them from fully exploiting parallel computation, and thus can lead to higher time costs to train and inference. As for the later, the receptive field at a specific time step depends on the convolutional kernel size and the number of layers, i.e., the ability to capture long-ranged information is limited.

The well-known self-attention based Transformer network (Vaswani et al. 2017) has recently been proposed for sequence modeling. It enables to effectively and highly parallelly capture long-range dependencies over time with single layers and can be easily adapted for different input sequence length. It has achieved enormous success in the filed of machine translation (Dai et al. 2019), music generation (Huang et al. 2018), traffic flow prediction (Xu et al. 2020), *etc*. In this paper, we propose a Transformer encoder based model for RUL estimation. Unlike RNN based models and CNN based models, Transformer architecture can access any part of the historical data regardless of distance by leveraging attention mechanism to process a sequence of data at once, which makes it potentially more powerful to capture long-term dependencies. Nevertheless, canonical dot-product self-attention makes the extracted high-level feature at each time step insensitive to its local contexts (Li et al. 2019), on which usually should be put more effort to estimate the corresponding RUL. Therefore, we further introduce a gated convolutional unit (GCU) by exploiting multiple control schemes on CNN network to emphasize the contribution from local contexts.

The main contributions of this work are:

- We propose a Transformer encoder based model for RUL estimation. To the best of our knowledge, this is the first successful attempt in adapting Transformer architecture to RUL estimation.
- We investigate a gated convolutional unit to better incorporate the local contexts into the attention mechanism *w.r.t*. each time step.
- We conduct experiments on four simulated turbofan engine degradation datasets, and show that the performance of our model is superior or comparable to those of other existing methods.

The overall organization of the paper is as follows. Section 2 introduces the related work on RUL estimation and the backbone of the proposed method. Section 3 describes the proposed model structure. Section 4 considers the experimental details, results and analysis. Finally, Sect. 5 concludes the work.

Related work

Deep learning based RUL estimation

Given a historical sensor data sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{t}$ with i denoting time steps, $x_i \in \mathbb{R}^m$, and m denoting feature dimension, an RUL estimation model aims to output the left time y_t *w.r.t*. x_t, before the time point beyond which a system cannot perform as reliable as desired.

One of the typical deep learning based approaches is to model RUL estimation as a regression problem, *i.e.*, directly mapping the input data to RULs. Malhi et al. (2011) first separated the training data into several degradation states via competitive learning. Then, an Elman RNN network was trained with the historical data before the last state. And finally, the performance of the latest state was predicted through this trained RNN network. Yuan et al. (2016) and Wu et al. (2018) applied a LSTM network and trained it in a supervised manner to directly estimate the remaining useful life. Based on that, Zheng et al. (2017) added a feed-forward neural network (FNN) above the LSTM layer to

improve the performance. Ellefsen et al. (2019) further attach a restricted Boltzmann machine (RBM) before the LSTM layer as an unsupervised pre-training stage in order to learn abstract features from raw unlabeled input data automatically. Besides, the genetic algorithm (GA) was applied to tune the diverse amount of hyper-parameters. In addition to the aforementioned RNN based methods, some prior works also utilized CNN for RUL estimation. For example, Babu et al., (2016) and Li et al. (2018) proposed to adopt the convolution and pooling filters along the temporal dimension over the multi-channel sensor data to incorporate automated feature learning, on which FNNs were also utilized for RUL regression. Xu et al. (2020) further replaced the CNN in Babu et al., (2016), Li et al. (2018) with a dilated CNN (DCNN) structure to accelerated model training and inference.

Some other methods explored effective health index (HI) that could reflect the machine degradation progress as much as possible. For instance, Guo et al. (2017) used LSTMs to fuse the selected features as the health indicator RNN-HI, and then, took an exponential model with a pre-defined failure threshold of RNN-HI to estimate the RUL. Yu et al. (2019) first constructed HIs by training a bidirectional LSTM based auto-encoder in an unsupervised way and calculated the RUL by comparing the HI curve of an on-line instance to the degradation patterns built in the offline phase with a similarity-based curve matching technique.

Transformer architecture

Transformer was first proposed for text encoding (Vaswani et al. 2017). It is a sequence-to-sequence architecture that consists of an encoder and a decoder. The encoder takes the input sequence and maps it into a higher dimensional vector, which is then fed into the decoder to generate an output sequence. Unlike the sequential nature of RNNs, Transformer builds long-term dependencies via a dot-product base attention-mechanism. It is proved that Transformer can improve the results of many natural language processing tasks such as machine translation (Vaswani et al. 2017), named entity recognition, general language understanding, and question answering (Devlin et al. 2018).

In this paper, we propose a Transformer encoder based architecture for RUL estimation, to cope with the limitations of RNN based and CNN based methods as described in Sect. 1. Moreover, we enhance the transformer encoder by introducing a CNN based gating mechanism, to improve the sensitivity of the model to local information *w.r.t.* each time step.

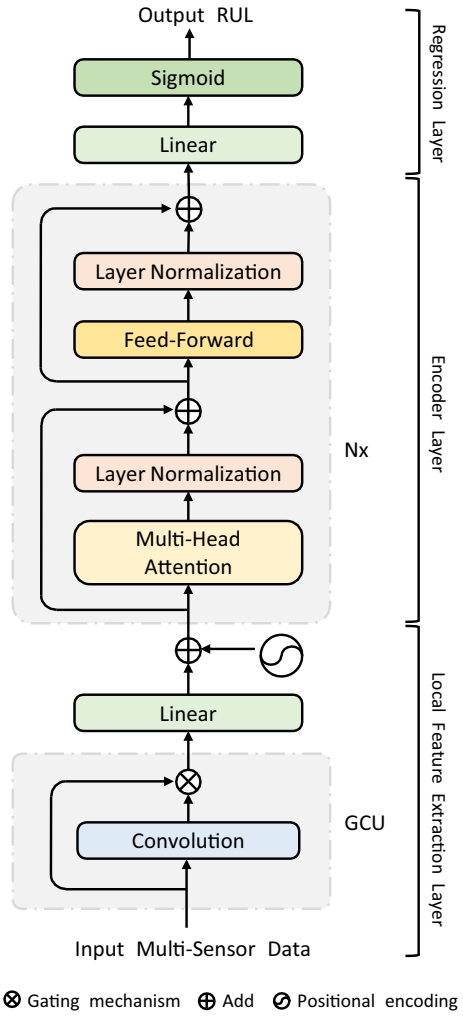


Fig. 1 Overall architecture of the proposed model

Methodology

In this section, we elaborate on the overall architecture of our proposed model in detail. As illustrated in Fig. 1, we organize our model with three layers, *i.e.*, the local feature extraction layer, the encoder layer, and the regression layer. To ease the explanation, we will first clarify the encoder layer in the following subsections and then introduce the other layers respectively.

Encoder layer

Here we exploit the Transformer encoder to capture both long- and short-term dependencies for RUL estimation. The Transformer encoder was first proposed in Vaswani et al. (2017) for natural language processing. It is composed of N identical sublayers and each sublayer has two modules: the first is a multi-head self-attention module, and the second is a feed-forward module. Following (Rush 2018), We employ

layer normalization in both modules and adopt a residual connection around each of them. We briefly introduce each module here and please refer to Vaswani et al. (2017) for more details.

Multi-head attention

Multi-Head Attention is actually composed of multiple heads applied the self-attention function. An attention function can be depicted as mapping a query and a set of key-value pairs to the output, where the query, keys, values are all matrices composed of vectors converted from the previous output. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

In practice, we calculate the attention function on a set of query vectors concurrently, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. The queries actually are the hidden vectors encoded by the previous layer, and the matrices K and V are assigned the same value as Q in self-attention. We denote the input of the multi-head self-attention module as $\boldsymbol{f} = \{f_i\}_{i=1}^{t}$ with f_i *w.r.t.* x_i and $f_i \in \mathbb{R}^d$. We first transform f to d_k-dimensional keys, values and queries:

$$K_j = fW_j^k \tag{1}$$
$$V_j = fW_j^v \tag{2}$$
$$Q_j = fW_j^q \tag{3}$$

where $W_j^k, W_j^v, W_j^q \in \mathbb{R}^{d\times d_k}$ are trainable projection matrices. With the derived Q_j, K_j, and V_j, we apply scaled dot-product attention:

$$\text{Attention}(Q, K, V)_j = \text{softmax}\left(\frac{Q_j K_j^T}{\sqrt{d_k}}\right) V_j \tag{4}$$

To jointly attend to information from different representation subspaces at different positions, we further employ H parallel attention calculation, *i.e.*, the multi-head attention, which can be formulated as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\{\text{head}_j\}_{j=1}^{H})W^A \tag{5}$$

where $\text{head}_j = \text{Attention}(Q, K, V)_j$, $W^A \in \mathbb{R}^{Hd_k\times d}$ and $d_k = d/H$.

Feed forward network

The feed-forward network consists of two linear transformations with a ReLU activation in between. It is applied to each time step separately and identically.

$$\text{FFN}(x) = W_2 \cdot \text{ReLU}(W_1 x + b_1) + b_2 \tag{6}$$

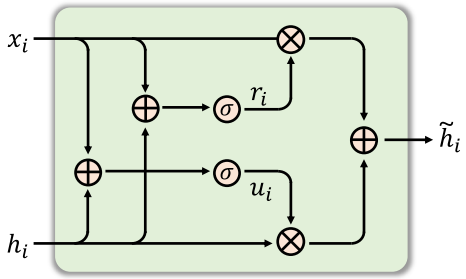


Fig. 2 The proposed gating mechanism

Local feature extraction layer

Here we proposed a the local feature extraction layer to map raw sensor data into distributed semantic representations, and provide informative local features among neighboring time steps to the upper layers at each time step, with the consideration that there could exist stronger dependencies among neighboring time steps in a time sequence. The proposed local feature extraction layer is comprised of a convolution operation, a gating mechanism, a linear mapping, and a position encoding. Note that, we refer to the combination of the first two components as the gated convolutional unit (GCU) mentioned in Sect. 1 and 2.

Convolution

Given an input sequence $\boldsymbol{x} = \{x_i\}_{i=1}^{t}$, $x_i \in \mathbb{R}^m$, the convolution sublayer extracts the local information h_i within a window-size k at each time step i:

$$h_i = \text{conv}([x_{i-k/2}, \ldots, x_i, \ldots, x_{i+k/2}]) \tag{7}$$

Gating mechanism

The proposed gating mechanism functions on the raw sensor data and the corresponding output from the convolution operation. It consists of two control schemes, *i.e.*, a reset gate r_i, and an update gate u_i. Figure 2 shows the calculation steps.

– **The reset gate**: As indicated in Chung et al. (2014), the input x_i plays a crucial role in alleviating information decay in deep neural structures. Inspired by that, here we exploit a reset gate to re-read the raw input information after the convolution operation:

$$r_i = \sigma(W_r h_i + V_r x_i + b_r) \tag{8}$$

where $W_u \in \mathbb{R}^{m\times m}$, $V_u \in \mathbb{R}^{m\times m}$, and $b_u \in \mathbb{R}^m$ are trainable parameters. $\sigma(\cdot)$ denotes the sigmoid function.

– **The update gate**: This gate targets at controlling the magnitude of the features with local context information, which is defined as:

$$u_i = \sigma(W_u h_i + V_u x_i + b_u) \tag{9}$$

where $W_u \in \mathbb{R}^{m\times m}$, $V_u \in \mathbb{R}^{m\times m}$, and $b_u \in \mathbb{R}^m$ are trainable parameters.

With the derived r_i and u_i, the output of the gating mechanism $\tilde{h}_i$ *w.r.t.* x_i is formulated as:

$$\tilde{h}_i = h_i \otimes u_i + x_i \otimes r_i \tag{10}$$

where $\otimes$ denotes the element-wise multiplication.

Linear mapping

We further apply a linear mapping to adapt the dimension of the local features M to that of the encoder layer d, which is constrained by the number of attention heads, *i.e.*, d should be divided by H with no remainder. Denote the output of the linear mapping as e_i, it is calculated as:

$$e_i = W_e \tilde{h}_i + b_e \tag{11}$$

where $W_e \in \mathbb{R}^{d\times m}$ and $b_e \in \mathbb{R}^d$ are trainable parameters, and d is the input feature dimension of the upper encoder layer.

Position Encoding

Let p_i denote the position encoding *w.r.t.* time step i, s denote the dimension. Here we use sine and cosine functions of different frequencies for position encoding:

$$\begin{aligned} p_i^{(2s)} &= sin(i/10000^{2s/d}) \\ p_i^{(2s+1)} &= cos(i/10000^{2s/d}) \end{aligned} \tag{12}$$

In this way, for any given l, p_{i+l} has a linear relationship with p_i. As a result, the model is allowed to effortlessly learn to reach by relative position.

In the end, we denote the final output of the local feature extraction layer as f_i, it is calculated as:

$$f_i = e_i + p_i \tag{13}$$

Regression layer

We finally take a linear regression layer with $g_t \in \mathbb{R}^d$ being the output of the encoder layer *w.r.t.* x_t as the input to predict the RUL:

$$y_t = \sigma(w_o g_t + b_o) \tag{14}$$

where y_t is the estimated RUL for the input sequence $\boldsymbol{x}$, $\sigma(\cdot)$ is the sigmoid function, $w_o \in \mathbb{R}^d$, and b_o is a scalar. Note that, we implement min-max normalization on RULs in advance for computational stability.

Table 1 Statistics of the C-MAPSS datasets

	$FD001$	FD002	FD003	FD004
Training sequences	100	260	100	249
Testing sequences	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

Experiments

In this section, we evaluate our proposed approach for RUL estimation and compare our approach to current state-of-the-art methods.

Settings

Benchmark Datasets

We conduct experiments on the C-MAPSS datasets to validate the effectiveness of our proposed method for RUL estimation. The C-MAPSS datasets are several challenge datasets from the first International Conference on PHM (Saxena et al. 2008), which were created by the Commercial Modular Aero-Propulsion System Simulation to simulate the degradation process of turbofan engines. As shown in Table 1, there are four subsets available, *i.e.*, FD001, FD002, FD003, and FD004, considering different operating conditions and fault modes. Each subset was further divided into multiple training sequences and test sequences. Both kinds of sequences come from different turbofan engines and start with different settings of initial wear and manufacturing variation. Note that all training sequences contain complete run-to-failure information while all testing sequences are terminated at sometime before a failure is observed, *i.e.*, the left time is the to-be-estimated RUL.

Data pre-processing

Sensor selection. As illustrated in Table 2, each subset, for instance, FD001, includes 26 columns: engine number, time step, three operation settings, and 21 sensor measurements

Table 2 Description of all 26 columns of the C-MAPSS datasets

Column Number	Description
1	Unit number
2	Time, in cycles
3	Operation setting 1
4	Operation setting 2
5	Operation setting 3
6	Sensor measurement 1
7	Sensor measurement 2
...	...
26	Sensor measurement 21

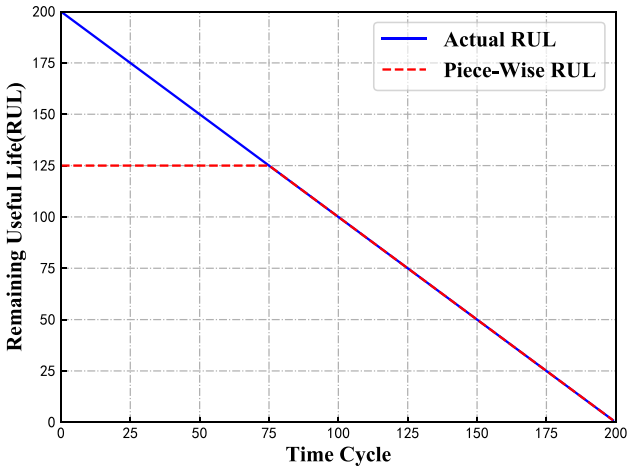


Fig. 3 Piece-Wise RUL of C-MAPSS Dataset, RUL_{max} is 125 time cycles

available. However, not all sensor data have information that could contribute to the RUL estimation. For example, measurements from several sensors are constants throughout the life cycle. In order to reduce the computational complexity, we employ the criteria proposed in Wu et al. (2018) for sensor selection. More specifically, the monotonicity matrix and correlation matrix are used for sensor selection. The monotonicity matrix represents the tendency of features along the time dimension, the feature dimensions with monotonicity contain more abundant degeneration information. The correlation matrix reflects the relationship between feature dimension and operation time. The two metrics can be calculated as follows:

$$\text{Mono}_i = \left| \frac{d\boldsymbol{x}^{(i)} > 0}{T-1} - \frac{d\boldsymbol{x}^{(i)} < 0}{T-1} \right| \tag{15}$$

$$\text{Corr}_i = \frac{\left| \sum_{t=1}^{T} (x_t^{(i)} - \bar{x}_t^{(i)})(t - \bar{t}) \right|}{\sqrt{\left| \sum_{t=1}^{T} (x_t^{(i)} - \bar{x}_t^{(i)})^2 \sum_{t=1}^{T} (t - \bar{t})^2 \right|}} \tag{16}$$

The contribution of features to RUL is positively correlated with both metrics. Consequently a linear combination of $Mono_i$ and $Corr_i$ can be calculated as the feature selection criteria:

$$\text{Cri}_i = \alpha \cdot \text{Corr}_i + (1 - \alpha) \cdot \text{Mono}_i - \gamma \tag{17}$$

where T is the length of the sequence during the whole lifetime, $\boldsymbol{x}^{(i)}$ is the time sequence that corresponds to the i-th column, t denotes the time step, $\bar{\cdot}$ denotes mean value, and $d\boldsymbol{x}^{(i)}$ denotes the differential of $\boldsymbol{x}^{(i)}$. $\alpha \in [0, 1]$ is a trade-off factor and γ is a threshold for sensor selection. We finally take all i-th columns with $\text{Cri}_i > 0$ as the input of our model, where $i \in \{1, 2, \ldots, 21\}$ and Cri_i is calculated by averaging over the whole subset.

Normalization. With the consideration that different sensor measurements lead to different numerical ranges, we apply a min-max normalization to normalize the input sensor data within the range of [0,1]. In this way, the convergence speed and the prediction accuracy of our model are expected to be improved. The min-max normalization formula is as follows:

$$x_t^{(i)} \leftarrow \frac{x_t^{(i)} - \min_{t,\boldsymbol{x}^{(i)}}(\{\boldsymbol{x}^{(i)}\})}{\max_{t,\boldsymbol{x}^{(i)}}(\{\boldsymbol{x}^{(i)}\}) - \min_{t,\boldsymbol{x}^{(i)}}(\{\boldsymbol{x}^{(i)}\})} \tag{18}$$

where $\{\boldsymbol{x}^{(i)}\}$ denotes the whole subset *w.r.t.* the i-th sensor sequences.

Piece-wise RUL. Another issue we consider here is the RUL labeling when given a run-to-fail sequence. In some early works such as Wu et al. (2018), RULs were set to decrease linearly along with time, with the assumption that the degradation state of a system develops linearly along with time. Yet in the actual situation, the system degradation is negligible at the early stage of its life cycle. Therefore, we employ a piece-wise linear regression model as in Zheng et al. (2017) to assign RUL labels for the training sequences. As demonstrated in Fig. 3, we first label RULs at the beginning with a constant RUL_{max}, *i.e.*, the horizon dot line. After running for a period of time, the system begins to enter a phase of linear degradation until it fails.

Implementation details

We implement our approach on PyTorch 1.3.1[1] based on the Annotated Transformer[2]. For data pre-processing, we simply set α in Eq. (17) to 0.5 by regarding the monotonicity and the correlation of equal importance. We empirically set the threshold γ to 0.2, resulting in an input dimension of $m = 14$. For the local Feature Extraction layer, we set the convolutional kernel size $k = 3$, and take padding of 1 to

[1] https://pytorch.org/

[2] https://github.com/harvardnlp/annotated-transformer

Table 3 Performance comparison of the proposed approach and prior state-of-the-art methods for RUL estimation.

Method	FD001	FD002	FD003	FD004	Average
CNN (Badu *et al*., 2016) Babu et al., (2016)	18.45	30.29	19.82	29.16	24.43
DBN (Zhang *et al*., 2016) Zhang et al. (2016)	15.04	25.05	12.51	28.66	20.32
LSTM-FNN (Zheng *et al*., 2017) Zheng et al. (2017)	16.14	24.49	16.18	28.17	21.25
CNN-FNN (Li *et al*., 2018) Li et al. (2018)	12.61	22.36	12.64	22.43	**17.51**
Auto-Encoder (Yu *et al*., 2019) Yu et al. (2019)	14.74	22.07	17.48	23.49	19.45
RBM-LSTM-FNN (Ellefsen *et al*., 2019) Ellefsen et al. (2019)	12.56	22.73	12.10	22.66	**17.51**
DCNN-FNN (Xu *et al*., 2020) Xu et al. (2020)	12.61	28.51	12.62	30.73	21.12
Auto-Encoder (Yu *et al*., 2020) Yu et al. (2020)	13.58	**19.59**	19.16	**22.15**	18.62
GCU-Transformer (this paper)	**11.27**	22.81	**11.42**	24.86	17.59

The **bold** results indicate the best performance while the underlined results imply the second best performance.

keep the sequence length as a constant. As for the encoder layer, we use $N = 2$ Transformer encoder blocks, $d = 128$ hidden units, and $H = 4$ self-attention heads. We heuristically set the dropout rate to 0.1 for experiments on RD001 and RD003, while setting the dropout rate to 0.5 for experiments on RD002 and RD004, with the consideration that the operation conditions and fault modes are more complex in the latter two datasets. As for the piece-wise RUL modeling, we empirically set the RUL limitation RUL_{max} to 125. For the optimizer of our model, we use Adam (Kingma and Ba 2014) with a learning rate of 0.001.

Evaluation metric

Here we use the root mean square error(RMSE) as the performance metric. Let $\mathcal{D}_{test} = \{\boldsymbol{x}\}$ denote the test set, y_t and $\hat{y}_t$ denote the ground-true RUL and the estimated RUL *w.r.t.* a certain test case $\boldsymbol{x} = \{x_i\}_{i=1}^{t}$, respectively. Then we formulate the performance metric as:

$$RMSE = \sqrt{\frac{1}{|\mathcal{D}_{test}|} \sum_{\boldsymbol{x} \in \mathcal{D}_{test}} (y_t - \hat{y}_t)^2} \tag{19}$$

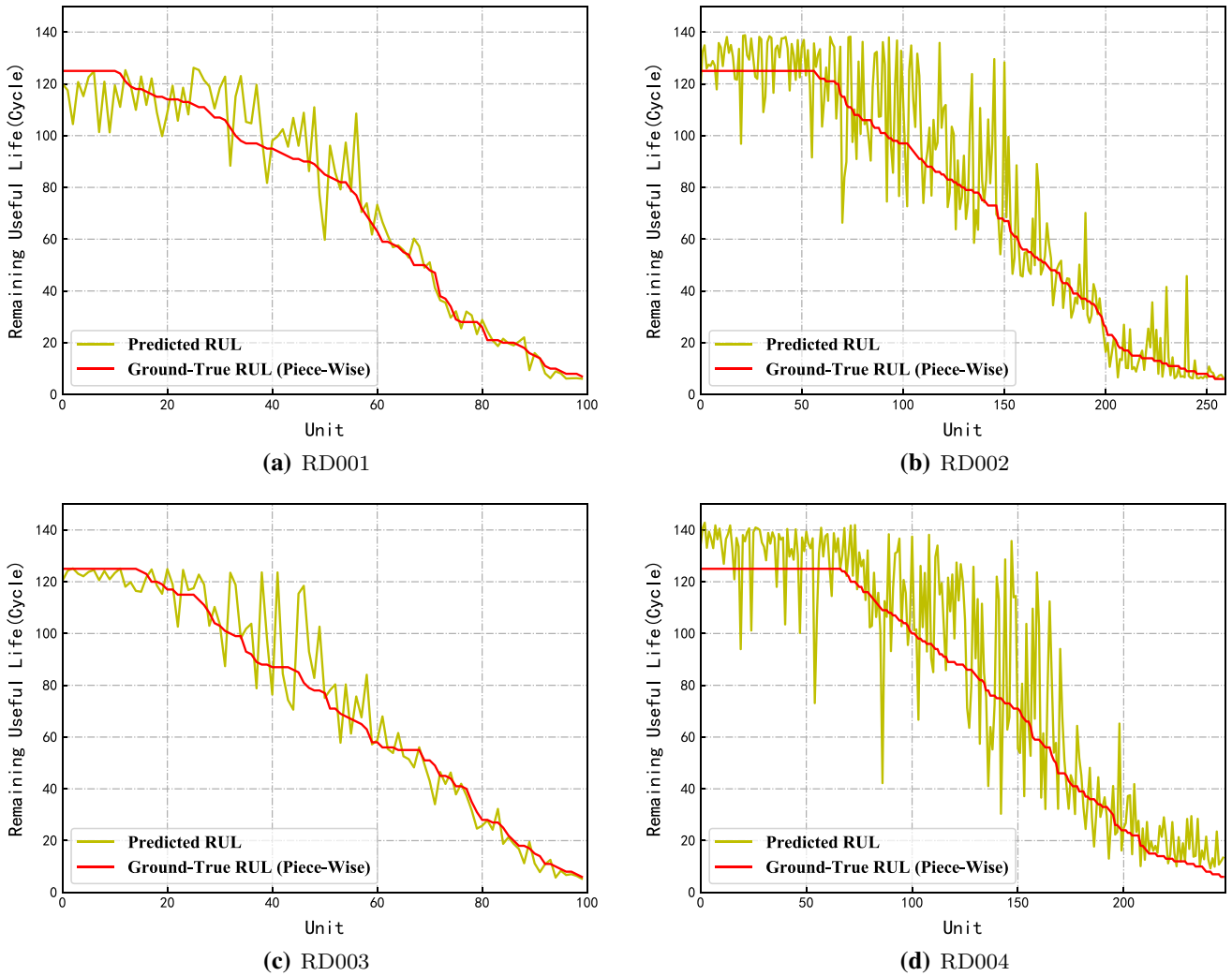
where $|\cdot|$ denotes the set cardinality.

Performance comparison

Table 3 presents the results of our approach for RUL estimation on the four subsets of the C-MAPSS datasets, alongside results from previous works. It is shown that our GCU-Transformer based architecture achieves the best performance on the FD001 dataset and the FD003 dataset, with relative improvements on RMSE compared to the previous state-of-the-art models ranging from 0.68 for FD003 to 1.29 for FD001. Also, our model leads to the second-best performance on the average of all four datasets, with only 0.08 in terms of RMSE inferior to the best results. Compared with the remarkable RBM-LSTM-FNN model proposed in Ellefsen et al. (2019), we do not employ a genetic algorithm for hyper-parameter selection, which could also be integrated into our approach and there is a good chance to further promote the ultimate performance. Compared with the recently proposed Auto-Encoder based method, which firstly takes an auto-encoder to construct HIs and then calculates RULs based on HI curve matching, our method is end-to-end and thus could be implemented with less effort.

We further compare our estimated RULs to the ground-true RULs and visualize the results in Fig. 4. To ease the explanation, we sort all test sequences according to their corresponding ground-true RUL lengths along the horizon axis in ascending order. From Fig. 4, we can see that our model can provide a satisfactory outcome at the beginning state of the life cycle though it only reads few historical information. We contribute this to the piece-wise RUL modeling strategy. Besides, it also shows that the estimation accuracy of our model is generally increasing with the time going, which is particularly important in practical application. The reason is that when it is closer to the breakdown point, an RUL estimation system is expected to provide more authentic results. In this way, the maintenance personnel can accordingly prepare more reasonable maintenance arrangements and further effectively avoid the unplanned system shutdown.

On the other hand, both Table 3 and Fig. 4 indicate that our model achieves the best performance on dataset RD001 while producing the worst results on dataset RD004. We attribute this to the differences in the complexity of the working condition (described by the number of operation conditions and fault modes) and the size of the training dataset. A simpler working condition and a larger training dataset would lead to the better test results. As shown in Table 1, in RD001, 100 training examples are available for the model to learn to predict RULs under a single working condition (the examples are obtained from *one* operation condition and *one* fault mode). While in RD004, only 249 training examples are provided to learn from 12 working conditions (*six* operating conditions



(a) RD001 **(b)** RD002 **(c)** RD003 **(d)** RD004

Fig. 4 Comparison between the actual RULs and the estimated RULs of our model. Each unit actually represents a test instance with a corresponding RUL, all the test units are sorted along the horizon axis according to corresponding ground-true RUL lengths and renumber them from one to the total number of units

and *two* fault modes). Corresponding information for RD002 and RD003 is also listed in Table 1.

Ablation study

In this paper, we propose several components and strategies to adapt the Transformer encoder architecture to the field of RUL estimation, including the gated convolutional unit, and the output layer. In this section, we conduct ablation study experiments to investigate the influence of these factors. Table 4 shows the results.

- *Ours w/o GCU*, which removes the gated convolutional unit (GCU) used in the local feature extraction layer. The performance in terms of RMSE decreases by 2.15 on the FD002 dataset. We conjecture that, without the proposed GCU, features extracted by the encoder layer lack the sensitivity to local context information, which however plays an important role in the RUL estimation at a certain time step.
- *Ours w/o GCU w/o Sigmoid*, which further eliminates the Sigmoid function employed in the output layer from

Table 4 Ablation study of the proposed architecture on the FD002 dataset

Model Structure	RMSE	Δ
Ours	23.43	–
Ours w/o GCU	25.58	-2.15
Ours w/o GCU w/o Sigmoid	26.29	-2.76

– *Ours w/o GCU*. In that case, our model degenerates into a pure Transformer encoder with a linear mapping from hidden features to the output RUL. From Table 4, we can see that this will lead to a performance drop of 2.76 in terms of RMSE, which demonstrates the effectiveness of the Sigmoid rescaling for RUL estimation.

Conclusion

In this paper, we propose a novel, Transformer encoder based deep neural architecture for RUL estimation, which could capture both short- and long-term dependencies in a time sequence. Compared with convolutional neural network based methods, our model is developed based on a dot-product self-attention mechanism across all time steps, and thus the receptive field of context information at each position is not limited by the window size, *e.g.*, the kernel size used in CNNs. Compared with recurrent neural network based methods, there are no sequential dependencies among different time steps in our model. In this way, our model can fully exploit parallel computation, and consequently, it is more computationally efficient. Considering that the attention mechanism used in the Transformer encoder makes the output high-level features insensitive to local contexts, we further propose a gated convolutional unit to promote the model's capacity of incorporating local contexts at each time step. We conduct experiments on the C-MAPSS datasets and show that, the performance of our model is superior or comparable to those of other existing methods. We also carry out ablation studies, which demonstrate the necessity and effectiveness of each component used in our model.

Acknowledgements This work was supported by the National Key R&D Program of China (No. 2018YFF0214705), and a grant from the Institute for Guo Qiang, Tsinghua University (No. 2019GQG0002).

References

Babu, G. S., Zhao, P., & Li, X. L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. *International conference on database systems for advanced applications* (pp. 214–228). Cham: Springer.

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2015). Health assessment and life prediction of cutting tools based on support vector regression. *Journal of Intelligent Manufacturing, 26*(2), 213–223.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q.V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860

Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

Ellefsen, A. L., Bjørlykhaug, E., Æsøy, V., Ushakov, S., & Zhang, H. (2019). Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety, 183,* 240–251.

Fahad, S.A., & Yahya, A.E. (2018). Inflectional review of deep learning on natural language processing. In: 2018 international conference on smart computing and electronic enterprise (ICSCEE), pp. 1–4. IEEE.

Guo, L., Li, N., Jia, F., Lei, Y., & Lin, J. (2017). A Recurrent Neural Network Based Health Indicator for Remaining Useful Life Prediction of Bearings. *Neurocomputing, 240,* 98–109.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Huang, C.Z.A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., Dai, A.M., Hoffman, M.D., Dinculescu, M., & Eck, D. (2018). Music transformer. arXiv preprint arXiv:1809.04281

Huang, R., Xi, L., Li, X., Liu, C. R., Qiu, H., & Lee, J. (2007). Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mechanical Systems and Signal Processing, 21*(1), 193–207.

Kacprzynski, G., Sarlashkar, A., Roemer, M., Hess, A., & Hardman, B. (2004). Predicting remaining life by fusing the physics of failure modeling with diagnostics. *JOm, 56*(3), 29–35.

Kingma, D.P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.X., & Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. arXiv preprint arXiv:1907.00235

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

Li, Y., Billington, S., Zhang, C., Kurfess, T., Danyluk, S., & Liang, S. (1999). Adaptive prognostics for rolling element bearing condition. *Mechanical Systems and Signal Processing, 13*(1), 103–113.

Malhi, A., Yan, R., & Gao, R. X. (2011). Prognosis of defect propagation based on recurrent neural networks. *IEEE Transactions on Instrumentation and Measurement, 60*(3), 703–711.

Mosallam, A., Medjaher, K., & Zerhouni, N. (2016). Data-driven prognostic method based on bayesian approaches for direct remaining useful life prediction. *Journal of Intelligent Manufacturing, 27*(5), 1037–1048.

Nieto, P. G., Garcia-Gonzalo, E., Lasheras, F. S., & de Cos Juez, F. J. (2015). Hybrid pso-svm-based method for forecasting of the remaining useful life for aircraft engines and evaluation of its reliability. *Reliability Engineering & System Safety, 138,* 219–231.

Oppenheimer, C. H., & Loparo, K. A. (2002). Physically based diagnosis and prognosis of cracked rotor shafts. *Component and systems diagnostics, prognostics, and health management II* (Vol. 4733, pp. 122–132). Washington: International Society for Optics and Photonics.

Peng, Y., Dong, M., & Zuo, M. J. (2010). Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology, 50*(1–4), 297–313.

Rush, A.M. (2018). The annotated transformer. In: Proceedings of workshop for NLP open source software (NLP-OSS), pp. 52–60.

San Kim, T., & Sohn, S.Y. (2020). Multitask learning for health condition identification and remaining useful life prediction: Deep

convolutional neural network approach. *Journal of Intelligent Manufacturing,* 1–11.

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 international conference on prognostics and health management, pp. 1–9. IEEE.

Tian, Z. (2012). An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. *Journal of Intelligent Manufacturing, 23*(2), 227–237.

Tian, Z., Wong, L., & Safaei, N. (2010). A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. *Mechanical Systems and Signal Processing, 24*(5), 1542–1555.

Tobon-Mejia, D.A., Medjaher, K., Zerhouni, N., & Tripot, G. (2011). Hidden markov models for failure diagnostic and prognostic. In: 2011 Prognostics and system health managment confernece, pp. 1–8. IEEE.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2012). A data-driven failure prognostics method based on mixture of gaussians hidden markov models. *IEEE Transactions on Reliability, 61*(2), 491–503.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017) Attention is all you need. arXiv preprint arXiv:1706.03762

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience, 2018*.

Wang, H., & Chen, J. (2014). Performance degradation assessment of rolling bearing based on bispectrum and support vector data description. *Journal of Vibration and Control, 20*(13), 2032–2041.

Wu, Q., Ding, K., & Huang, B. (2018). Approach for fault prognosis using recurrent neural network. *Journal of Intelligent Manufacturing,* 1–13.

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

Xu, M., Dai, W., Liu, C., Gao, X., Lin, W., Qi, G.J., & Xiong, H. (2020). Spatial-temporal transformer networks for traffic flow forecasting. arXiv preprint arXiv:2001.02908

Xu, X., Wu, Q., Li, X., & Huang, B. (2020). Dilated convolution neural network for remaining useful life prediction. *Journal of Computing and Information Science in Engineering, 20*(2).

Yu, W., Kim, I. Y., & Mechefske, C. (2019). Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. *Mechanical Systems and Signal Processing, 129,* 764–780.

Yu, W., Kim, I. Y., & Mechefske, C. (2020). An improved similarity-based prognostic algorithm for rul estimation using an rnn autoencoder scheme. *Reliability Engineering & System Safety, 199,* 106926.

Yuan, M., Wu, Y., & Lin, L. (2016). Fault diagnosis and remaining useful life estimation of aero engine using lstm neural network. In: 2016 IEEE international conference on aircraft utility systems (AUS), pp. 135–140. IEEE.

Zhang, C., Lim, P., Qin, A. K., & 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, X., Kang, J., & Jin, T. (2014). Degradation modeling and maintenance decisions based on bayesian belief networks. *IEEE Transactions on Reliability, 63*(2), 620–633.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In: 2017 IEEE international conference on prognostics and health management (ICPHM), pp. 88–95. IEEE.