Multi-scale Convolutional Gated Recurrent Unit Networks for Tool Wear Prediction in Smart Manufacturing

Weixin Xu, Huihui Miao, Zhibin Zhao, Jinxin Liu, Chuang Sun, Ruqiang Yan*

School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China.

Abstract: As an integrated application of modern information technologies and artificial intelligent, Prognostic and Health Management (PHM) is important for machine health monitoring. Prediction of tool wear is one of the symbolic applications of PHM technology in modern manufacturing systems and industry. In this paper, a multi-scale Convolutional Gated Recurrent Unit network (MCGRU) is proposed to address raw sensory data for tool wear prediction. At the bottom of MCGRU, six parallel and independent branches with different kernel sizes are designed to form a multi-scale convolutional neural network, which augments the adaptability to features of different time scales. These features of different scales extracted from raw data are then fed into a Deep Gated Recurrent Unit network to capture long-term dependencies and learn significant representations. At the top of the MCGRU, a fully connected layer and a regression layer are built for cutting tool wear prediction. Two case studies are performed to verify the capability and effectiveness of the proposed MCGRU network and results show that MCGRU outperforms several state-of-the-art baseline models.

Keywords: tool wear prediction, multi-scale, convolutional neural networks, gated recurrent unit.

1. Introduction

The development of Prognostic and Health Management (PHM) has motivated the research in the field of machine health monitoring to detect faults and predict machine's future conditions [1-4]. In modern manufacturing system, the worn tool is harmful for the metal cutting process and often causes additional costs [5]. The cutting tools will gradually become blunt during the manufacturing process, as shown in Figure 1, because of a lot of factors like abrasion, deformation and attrition. As a result, the quality of the products will be degraded. It is therefore crucial to monitor and predict the cutting tool wear online so as to prevent the quality from degradation [6].

Aiming at monitoring the working conditions of cutting tools and predicting tool wear, many methods, direct or indirect, online or offline, have been researched. Traditionally, by performing cutting tests under different working conditions, data about the cutting tools are acquired and then analyzed with the help of optimization techniques including the response surface methodology (RSM) and the design of experiments (DOE). This approach is time-consuming and inefficient because the number of the tests required is large [7]. The finite element method

* Corresponding author: yanruqiang@xjtu.edu.cn.

(FEM) [8-10] has also been used in different cutting tasks [11, 12] and to predict cutting tool wear. Over the last

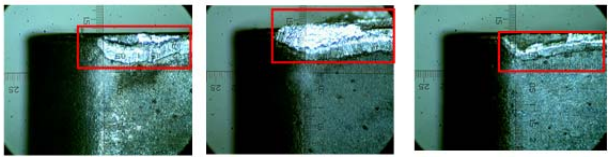


Figure 1. Degradation of cutting tools.

two decades, methods based on deep learning and neural networks have started to be used for the estimation and prediction of cutting tool wear. Ko et al. [13] designed an autoregressive model followed by a highly parallel neural network to monitor the cutting state. Özel et al. [14] used neural networks for the prediction of cutting tool wear and its surface roughness. Ghosh et al. [6] designed a sensor fusion model with the help of neural network to extract and fuse features from various signals for the estimation of cutting tool's average flank wear. By using Adaptive Neuro Fuzzy Inference System, Sharma et al. [15] developed a method for tool wear estimation. Venkata et al. [16] fed the cutting speed, the radius of nose and the volume of removed material to a multilayer perceptron model for the prediction of amplitude vibration, surface roughness, and tool wear. Zhao et al. [17] designed a convolutional bi-directional LSTM networks to monitor machine health, as well as to predict the tool wear depth. In general, researchers divide these methods into two major categories: physics-based methods and data-driven methods. In tasks of tool wear prediction, physics-based methods based on grey models and particle filters [18] have proven to be effective. However, these methods usually require accurate and high-quality domain knowledge, which is often unavailable under complex and noisy working conditions. Moreover, most of them are unable to be upgraded with online data. Data-driven methods are now more attractive because they are able to address these issues. Deep learning theories and large amounts of data collected by advanced sensors have promoted the development of data-driven online methods. Two phases are usually included in data-driven models [19], where the first phase is to train models with collected data and then the other phase is to apply the trained models to online data to monitor the conditions or make predictions. The key of these two phases is that deep learning theories enables the model to better extract features and derive representations of machine conditions hidden in the data, and therefore enables it to make better predictions based on online data. In this paper, we research data-driven methods with the help of deep learning theories to predict cutting tool wear.

Data-driven methods take single or multiple sensory data as input, feeding them into training models to extract features and learn representations. Online data will then be fed into the well trained models to make predictions. Data and models are two core parts of data-driven methods. Figure 2 shows the basic framework of data-driven methods. The raw time series data collected by sensors are in sequential forms, whose sequential characteristics are difficult to be discovered by previous work focusing on developing models to extract multi-domain features. These models, trying to extract statistical, frequency and time-frequency features, require intensive expert knowledge and feature engineering. Some models, such as the Markov models and Kalman filters [20-22], are capable of

addressing sequential data, but they are not good at capturing long-range dependencies. It is important to capture connections and information in time scale because in real working conditions, the features are often submerged by heavy background noise, which will cause failures in these models. The development in the field of neural networks and deep learning has offered solutions to address these issues, and one of these solutions is Recurrent Neural Network (RNN). Traditionally, neural networks deal with inputs and outputs independently, which is not so reasonable in some sequential tasks. RNN is proposed to make use of information in arbitrary long sequences, and to capture the calculated information. However, the problem of gradient exploding and vanishing in traditional RNN weakens its power. Some improved variants of RNN have been designed to solve this problem, and one of them is LSTM, namely Long Short-Term Memory Network. LSTM is good at solving problems that in need of information about previous events [23], which means that it is better at addressing sequential data of various length and capturing long-term dependencies. LSTM needs sufficient data to train while in real working conditions, while there may be no sufficient labeled data. GRU network performs better under such situations. Proposed in 2014, Gated Recurrent Unit (GRU) [24] is a more efficient variant of LSTM that shares many similar properties. With comparable performance to LSTM on sequence modeling, GRU has fewer parameters and is easier to train. Here, we introduce GRU networks to be one part of our network architecture. As one type of neural networks, GRU is able to extract features and learn representations without expert or domain knowledge, but it may be not robust because of the existence of noise in the raw sensory data. Compared with GRU, Convolutional Neural Network (CNN) is more robust when the data has noise interference. The convolutional operations in the CNN are able to extract abstract features by applying learnable filters to the convolutional layers to convolve with sequential data. For this reason, in [17], Zhao et al. adopted a one-layer CNN as a local feature extractor. However, as the information hidden in the sequential data is complicated and diverse, this local feature extractor with only one kernel size cannot extract all of the useful information. To address this issue, filters of different sizes are adopted to form a multi-scale convolutional layer to extract different significant features. Here, we use this multi-scale convolutional network to extract hidden but important features and then these features will be concatenated into a single feature map.

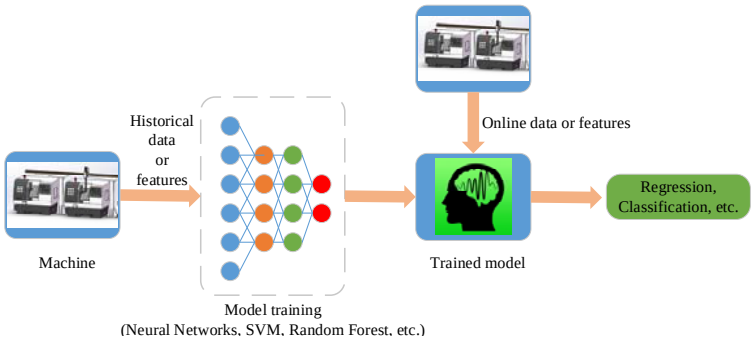


Figure 2. Basic framework of data-driven methods.

In this paper, we propose a model combining multi-scale CNN and GRU named Multi-scale Convolutional

Gated Recurrent Unit Network (MCGRU) to predict cutting tool wear. In this model, the multi-scale CNN consists of six parallel branches, and they are independent of each other. These branches are able to extract local features, as well as abstract ones from high level. Then these feature maps will be merged into a single feature map. Temporal information is encoded and representations are learnt in a two-layer GRU network, built on the top of the merge layer. We experiment on an open source dataset from cutters of high-speed Computer Numerical Control (CNC) milling machine, containing acoustic emission data, accelerometer data and dynamometer data [25]. Additionally, another experiment of CNC tool wear is carried out, through which current and vibration data and tool wear depth are sampled. Based on these sequential data and their corresponding tool wear depth, we compare the predicting ability of our model with that of several state-of-the-art models.

This paper is organized as follows. In Section 2, we review some related work about CNN and GRU. Based on classic CNN and GRU, the CCGRU is designed and its details are presented in Section 3. Two case studies on the prediction of tool wear are conducted in Section 4. More details about the model and future steps are discussed in section 5. Finally, conclusions are presented in Section 6.

2. Review of related work

2.1. Convolutional Neural Network

Convolutional Neural Network has proven very powerful in many recognition and classification tasks [26-28]. It has also shown the power to address sequential data in task of natural language processing [29-31]. In the convolutional layers, filters slide over sequential data to extract features and filters in the pooling layers will focus on the most salient ones. Additionally, the training process can be sped up and the model's performance can be improved by adding batch normalization layers [32]. The capability of CNN can be further improved by stacking the above layers to build a "deep" CNN. Besides, the width of CNN can also influence its performance. In the inception module [33], parallel branches consisting of convolutional and pooling layers with different kernels are designed. This architecture allows the model to recover both local features via kernels of smaller sizes and high abstract features via that of larger sizes.

In our MCGRU network, an architecture of six parallel branches of CNN is designed to process the input sequential data before it is fed to GRU units. Kernels with different sizes are adopted in different branches to extract local and abstract features at the same time. The model itself is going to determine which features are significant to be chosen.

2.2. RNN, LSTM and GRU

Recurrent Neural Network (RNN) is mainly proposed to handle long term dependencies while processing sequential data in task of natural language processing. The hidden states in RNN use the outputs of the previous

states as the inputs of the next states, which means that the sequential information is preserved. As weights are shared across time, RNN is able to process sequential input of any length. However, the problem of gradient exploding and vanishing emerged as the major obstacle to traditional RNN's performance. To avoid this problem, Long Short-Term Memory Network was proposed by Sepp and Jürgen [34] in 1997 and was improved by Felix Gers' team [35] in 2000. It is able to prevent backpropagated error from vanishing [36] and memorize a state for different time periods with the help of the input gate, forget gate and output gate, which manage the flow of information in the network. As another variant of RNN, Gated Recurrent Unit (GRU) was introduced to solve the vanishing problem. With only a reset gate and an update gate, GRU has comparable performance to LSTM. However, there are fewer parameters in GRU because it lacks an output gate and has less complex structure, which means that it is more efficient and can be used under situations where there is no sufficient data. Considering the effectiveness of GRU, it has been more and more widely used to learn significant representations in time series data.

In the proposed MCGRU network, two-layer GRUs are adopted to process the output of the multi-scale convolutional layers. Effective representations will be learnt here and to be used in the prediction of tool wear.

2.3. Neural networks and tool wear prediction

Neural networks have been successfully used in tasks of machine condition monitoring like tool wear prediction because of their excellent features extraction and representations learning capabilities [37-42]. Artificial Neural Networks (ANN) were firstly adopted and were proved to have good performance in the machine condition monitoring tasks. However, with more and more interference and as the working conditions become more and more complex, ANN is no longer good at solving these problems. As a result, Convolutional Neural Networks (CNN) was introduced in this field. The depth of the networks and their learning ability enable themselves to learn what they need in these tasks. However, most of these models make use of the features manually extracted and designed from raw data, while ignoring the representations and the relations of different time steps hidden in the sequential data. For tool wear prediction, the information about the condition of cutting tools remains to be discovered. Our proposed MCGRU combines multi-scale CNN with GRU to learn features and representations without the intervention of human designed features, which the information behind the raw sensory data can be explored as much as possible and the prediction accuracy will be improved.

3. Model

Before presenting the MCGRU network, some notations used in this paper are clarified here. The task is to design a model for tool wear prediction based on the multiple in-process sensory data. A labeled time series dataset is given as $D = \left\{\left(\mathbf{x}_i, y_i\right)\right\}_{i=1}^{N}$, which contains N tool conditions, and their corresponding labels y_i, i.e., each tool

condition corresponding to a tool wear that is measured and recorded as y_i. Assuming that in each tool condition $\mathbf{x}_i$, q channels of sensory data are sampled and the length of each channel of sensory data is L. For each channel, the whole sequence is divided into l sections, i.e., l time steps. The i^{th} cutting tool condition is

$$\mathbf{x}_i = \left[\mathbf{x}_i^1 \quad \mathbf{x}_i^2 \quad \cdots \quad \mathbf{x}_i^l \right]^{\mathrm{T}} \tag{1}$$

where vector $\mathbf{x}_i^t \in R^d$ is the multiple channels of sensory data sampled at time step t, i.e., the t^{th} section, and $d = q * (L/l)$ is the dimensionality of $\mathbf{x}_i^t$, and $(\cdot)^{\mathrm{T}}$ represents the transpose. The goal is to predict the tool wear $\hat{y}_i$ through $\mathbf{x}_i$. In our proposed Multi-scale Convolutional Gated Recurrent Unit, the Multi-scale Convolutional Network functions as a feature extractor and the Gated Recurrent Unit functions as a temporal encoder. Six parallel and independent branches of Convolutional Neural Network consisting of different kernels are designed to process the raw sensory data. Local and abstract features extracted are fed into a merge layer, on the top of which is a two-layer GRU designed to learn significant representations. Finally, the prediction is performed by a fully connected layer and a regression layer. The MCGRU network is shown in Figure 3..

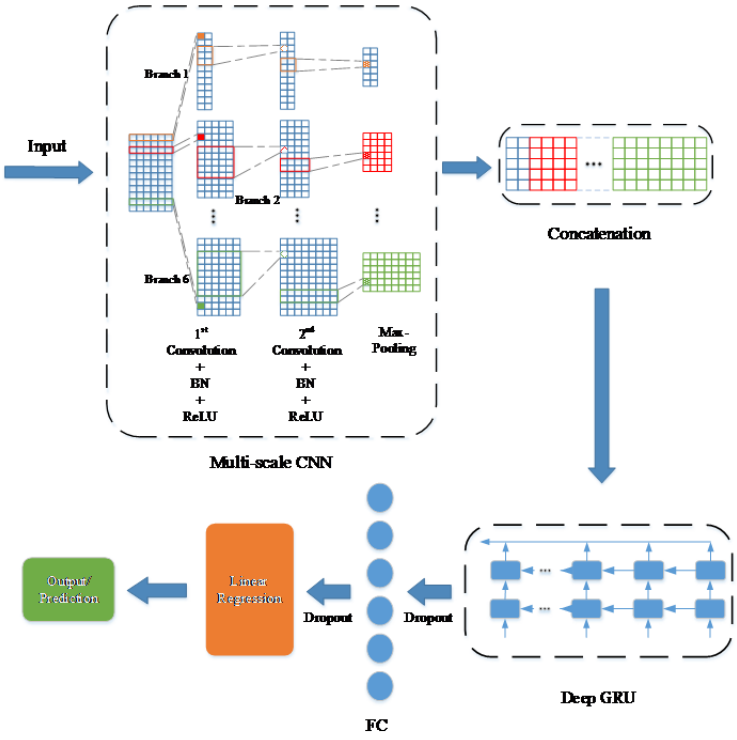


Figure 3. Structure of the proposed MCGRU

3.1. Multi-scale CNN

In each branch of the multi-scale CNN, a five-layer CNN is adopted, which consists of two convolutional layers, one max-pooling layer and two batch normalization layers. In the first convolutional layer of each branch, the kernel size equals to 1. The adoption of this convolutional layer is not only able to help extract more significant features, but also able to reduce the parameters of the model. For example, in one-dimensional convolution, a CNN containing one convolutional layer with kernel size equaling to 7 has more parameters than that containing two

convolutional layers with kernel sizes equaling to 1 and 7, respectively. A batch normalization layer is adopted at the top of the first convolutional layer. Batch normalization layers in hidden layers help to accelerate the training and augment the predicting accuracy. Then, the output of the batch normalization layer is fed into the second convolutional layer. The second convolutional layers with different kernels in different branches extract multiple time scale features hidden in the sequential data. Small kernels are able to extract local features, while large kernels are able to extract abstract features. Based on these multi-time-scale features, the model itself learns to determine which ones should be concerned about. The last two layers are another batch normalization layer and a max-pooling layer. The max-pooling layer compresses the previous feature maps to further learn more significant features. Then in a merge layer, all of the feature maps from different branches are concatenated into a single feature map. All of the features extracted from branches are reserved. The organization of these two kinds of structure is shown in Figure 4. Details are presented in the following contents respectively. Here we take the operations in one branch as example, and operations in other branches are the same.

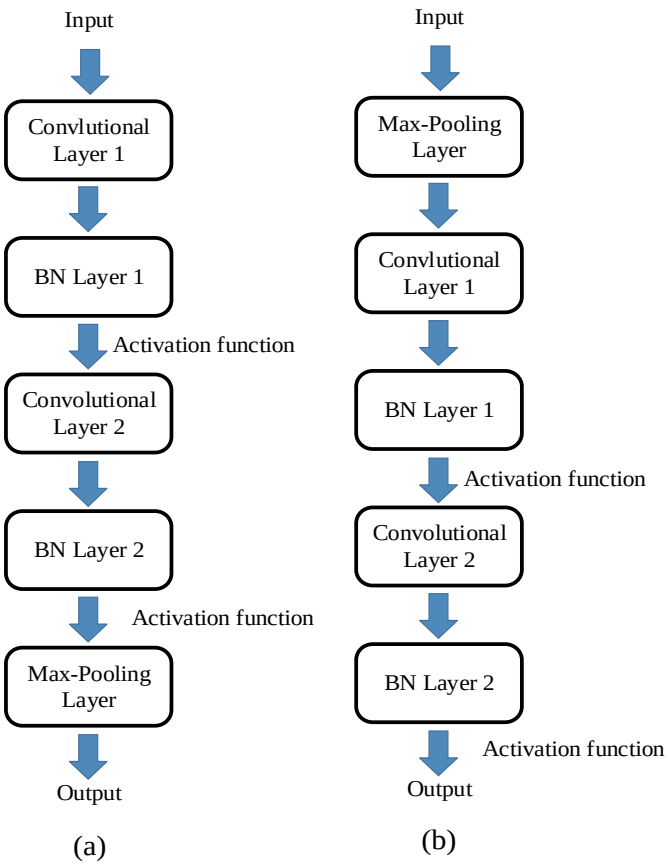


Figure 4: (a) Structure of five of the six CNN branches. (b) Structure of the sixth CNN branch

3.1.1. Convolution

In the convolutional layer of each branch in the Multi-scale CNN, the 1-dimensional convolution operation is achieved by using a filter (kernel) $\mathbf{v} \in R^{h \times d}$ to slide over $\mathbf{x}_i \in R^{l \times d}$ to convolve with the subsection

$\mathbf{x}_i^{t:t+h-1} \in R^{h \times d}$ from time step t to time step $t+h-1$. The $\mathbf{x}_i^{t:t+h-1}$ is given as follows:

$$\mathbf{x}_i^{t:t+h-1} = \left[\mathbf{x}_i^t \quad \mathbf{x}_i^{t+1} \quad \cdots \quad \mathbf{x}_i^{t+h-1} \right]^{\mathrm{T}} \tag{2}$$

where h is the kernel size. Additionally, a bias term b is added to get the complete convolution operation, which can be given as:

$$c_j^t = \mathbf{v}_j \circ \mathbf{x}_i^{t:t+h-1} + b \tag{3}$$

where $j \in R$ represents the j^{th} filter $\mathbf{v}$ and $\circ$ represents the Hadamard product.

As the filter slides over $\mathbf{x}_i$ and the convolution operation is done, we get a vector $\mathbf{c}_j$, which is given by:

$$\mathbf{c}_j = \left[c_j^1 \quad c_j^2 \quad \cdots \quad c_j^{(l-h+2p)/s+1} \right]^{\mathrm{T}} \tag{4}$$

where p is the amount of zero padding, s is the sliding stride of the kernel, and $(l-h+2p)/s+1$ is the length of the output after convolution operation. When s and p are set, the length of the output depends on the kernel size h. Different kernels size results in different output sizes. It is an important point because in each branch of our proposed Multi-scale CNN, different kernel sizes are chosen.

Specially, to concatenate different outputs from different branches, it is more meaningful to get outputs with same size. Therefore, the trick of zero padding is adopted in the convolution operation. In different branches, different amounts of zero padding are adopted, which helps the output to have the same size as the input, no matter which kernel size is chosen. As a result, the feature map can be given by:

$$\mathbf{c}_j = \left[c_j^1 \quad c_j^2 \quad \cdots \quad c_j^l \right]^{\mathrm{T}} \tag{5}$$

3.1.2. Batch normalization

Instead of just normalizing the input of the CNN, we adopt batch normalization [32] layers to normalize the inputs within the network by using the variance and the mean of the values in the current mini-batch. In the batch normalization layer, the operation can be represented as follows:

$$\mathbf{bn}_j = \mathrm{BN}\left(\mathbf{c}_j\right) \tag{6}$$

As batch normalization layer does not change the feature map's size, we therefore get:

$$\mathbf{bn}_j = \left[\mathrm{BN}\left(c_j^1\right) \quad \mathrm{BN}\left(c_j^2\right) \quad \cdots \quad \mathrm{BN}\left(c_j^l\right) \right]^{\mathrm{T}} \tag{7}$$

3.1.3. Activation function

After the convolution and batch normalization operations, an activation function is added to bring in non-linear properties and therefore to learn non-linear complex arbitrary functional relationships between inputs and outputs. As a result, the convolution, batch normalization and activation operations can be together given by:

$$a_j^t = f\left(\mathrm{BN}\left(\mathbf{v}_j^{\mathrm{T}} \mathbf{x}_i^{t:t+s-1} + b\right)\right) \tag{8}$$

where $f(\cdot)$ is an activation function. Here, we choose Rectified Linear Units (ReLU) [43] as the activation function in our proposed model.

The above three operations result in a feature map, which can be given by:

$$\mathbf{a}_j = \begin{bmatrix} a_j^1 & a_j^2 & \cdots & a_j^l \end{bmatrix}^{\mathrm{T}} \tag{9}$$

3.1.4. Max-pooling

By introducing pooling layers in the network, the previous feature maps' size can be further reduced and more significant and abstract features can be extracted. Here, we adopt max-pooling operation. In one-dimensional pooling, with the pooling length k, the max-pooling operation uses a kernel to slide over the feature map to get the max value over the k consecutive values. Here we let the sliding stride equal to k, and as a result, the output of max-pooling operation can be given by:

$$\mathbf{m}_j = \begin{bmatrix} m_j^1 & m_j^2 & \cdots & m_j^{(l-k+2p)/s+1} \end{bmatrix}^{\mathrm{T}} \tag{10}$$

where $m_j^i = \max\left(a_j^{(i-1)s}, a_j^{(i-1)s+1}, \cdots, a_j^{(i-1)s+k-1}\right)$.

3.1.5. Concatenation

In the concatenating layer, the feature maps from different branches will be concatenated into a single feature map to merge all the local and abstract features. Assuming that in the i^{th} branch, the j^{th} output of this branch is given by:

$$\mathbf{m}_{ij} = \begin{bmatrix} m_{ij}^1 & m_{ij}^2 & \cdots & m_{ij}^{(l-k+2p)/s+1} \end{bmatrix}^{\mathrm{T}} \tag{11}$$

Then, the output of the concatenation layer isgiven as:

$$\mathbf{Concatenation} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{M}_2 & \cdots & \mathbf{M}_N \end{bmatrix} \tag{12}$$

where N is the serial number of branch, and $\mathbf{M}_i$ can be represented as

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{m}_{i1} & \mathbf{m}_{i2} & \cdots & \mathbf{m}_{iK_i} \end{bmatrix} \tag{13}$$

where K_i is the number of output of the i^{th} branch.

To summarize, in the Multi-scale CNN, the shape of the input sequence is $n \times l \times d$. Here, n represents the total number of working conditions. As descripted above, before the concatenating layer, the output shape of the i^{th} branch is $n \times \left(\left(l-k+2p\right)/s+1\right) \times K_i$. In different branches, kernel sizes from small to large help to extract local and abstract features. Compared to the original raw sequence, these multi-time-scale features can better represent the properties of the working conditions. As these features are merged in the concatenating layer, the following GRU is added to learn significant representations of the working conditions. To be more specific, the framework of the Multi-scale CNN is illustrated in figure 5.

3.2. Deep GRU

Under real industrial conditions, clean sample data is difficult to obtain. Compared to LSTM, GRU is better at dealing with such situations where there is no sufficient data. Here, on the top of the Multi-scale CNN, a two-layer

GRU network is designed to excavate vital representations from the multi-time-scale features. The deep GRU is presented as follows.

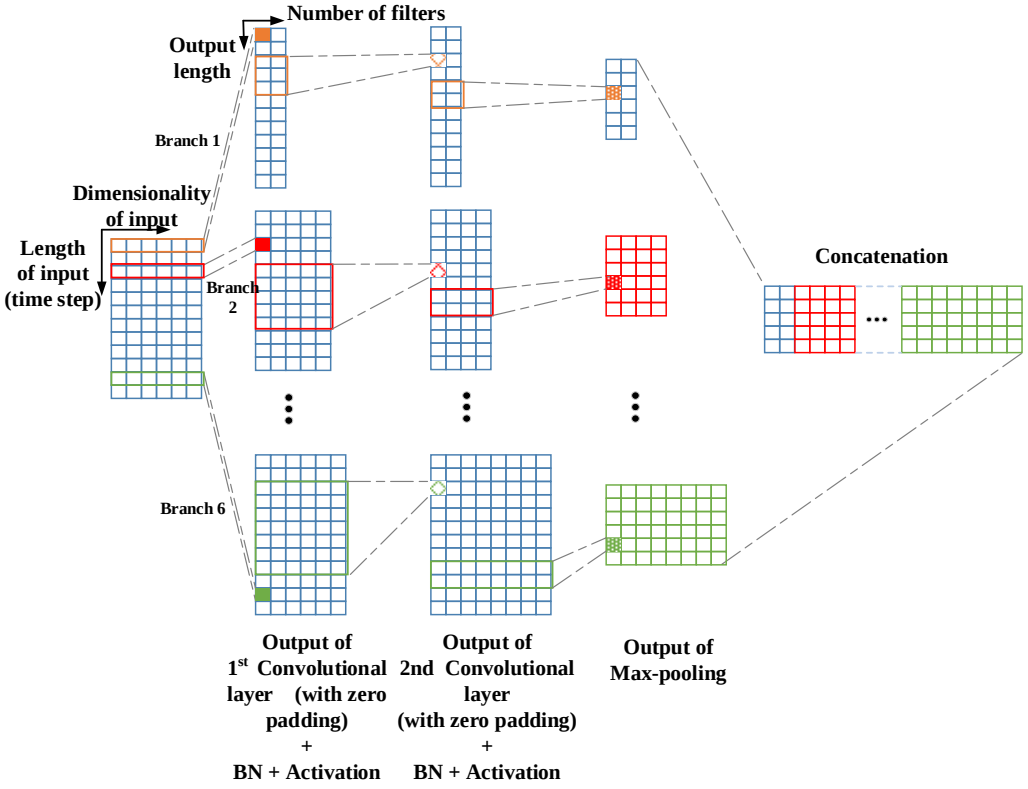


Figure 5. framework of Multi-scale CNN

3.2.1. Gated Recurrent Unit

In GRU, the inputs are the hidden state $\mathbf{h}_{t-1}$ at previous time step $t-1$ and the data $\mathbf{x}_t$ at the current time step t, and the output is the hidden state $\mathbf{h}_t$. The output $\mathbf{h}_t$ depends on the previous hidden state $\mathbf{h}_{t-1}$, the update gate $\mathbf{z}_t$, the reset gate $\mathbf{r}_t$ and the candidate hidden state $\tilde{\mathbf{h}}_t$. The reset gate $\mathbf{r}_t$ enables the unit to drop any information in the hidden state that is less meaningful or irrelevant, so as to focus on the information that is more important. The update gate $\mathbf{z}_t$ determines the information from the previous and the candidate hidden state that can be passed to the current hidden state [23]. The relating equations can be given by:

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z\right) \tag{14}$$

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r\right) \tag{15}$$

$$\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\left(\mathbf{r}_t \circ \mathbf{h}_{t-1}\right) + \mathbf{b}_h\right) \tag{16}$$

$$\mathbf{h}_t = \left(1-\mathbf{z}_t\right) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \tilde{\mathbf{h}}_t \tag{17}$$

where σ is the sigmoid activation function, $\mathbf{W}_z$, $\mathbf{U}_z$, $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{W}_h$ and $\mathbf{U}_h$ are shared weight matrices which are learned during training, $\mathbf{b}_z$, $\mathbf{b}_r$, $\mathbf{b}_h$ are learnable biases. The basic structure of a one-layer GRU is shown in figure 6.

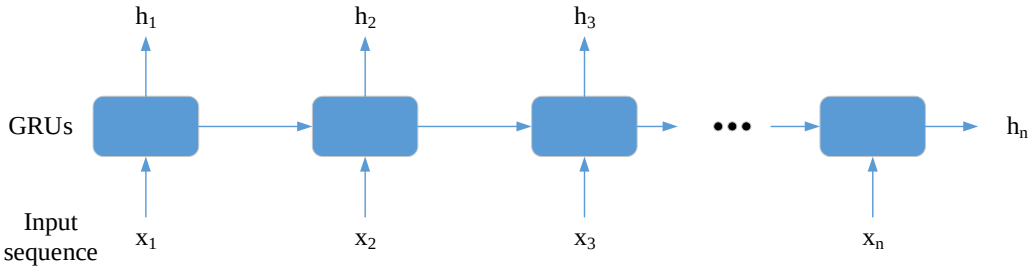


Figure 6. one-layer GRU

3.2.2. Deep GRU

As mentioned above, the capability of a neural network can be improved by "going deeper" [44, 45]. In a deep neural network, there exists more non-linear operations and more abstract features and representations can be learned. Inspired by this idea, we stack two GRU layers to get a deep architecture, in which each GRU layer contains different number of units. In the deep GRU, as shown in figure 7, while the output of each hidden state in one layer propagating through time, it is also the input of the hidden state in the next layer. Features at low level are therefore learned and passed to the next layer to learn higher-level representations. By stacking GRU layers, the network is able to learn essential representations at different time scales more effectively.

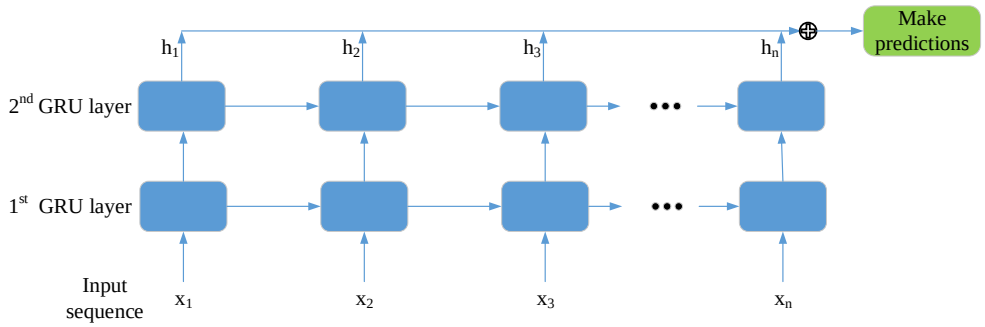


Figure 7. Deep GRU

3.3. Fully Connected and Linear Regression Layer

The output representation of GRU network is flattened as $\mathbf{h}$ and then fed into a fully connected layer to be prepared for the linear regression layer. The operation of the fully connected layer can be given as follow:

$$\mathbf{o} = f\left(\mathbf{W}\mathbf{h} + \mathbf{b}\right) \tag{18}$$

where $\mathbf{o}$ is the output of the fully connected layer, $\mathbf{W}$ is the transformation matrix, $\mathbf{b}$ is the bias, and $f(\cdot)$ is the activation function. We use ReLU here as the activation function. Finally, the fully connected layer's output $\mathbf{o}$ is fed into a regression layer and the tool wear of the i^{th} working condition is therefore predicted, which can be given by

$$\hat{y}_i = \mathbf{W}\mathbf{o}_i \tag{19}$$

3.4. Training and regularization of MCGRU

The Mean Absolut Error (MAE) is adopted as the loss in the training process,, which is given by:

$$loss = MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{20}$$

where n represents the total number the samples.

The optimizer we adopt here is Root Mean Square Propagation (RMSProp) [46]. It is a very robust optimizer with pseudo curvature information. RMSProp is useful for mini batch learning because the gradients are normalized by the magnitude of the recent gradients, enabling it to handle stochastic objectives properly. RMSProp is a nice optimizer for recurrent neural networks like LSTM and GRU.

As mentioned above, GRU rather than LSTM is chosen in our proposed model because in real working conditions, there is usually no sufficient labeled data. When going deep and when there is no sufficient data, the network may be too complex to train and the problem of overfitting may appear. In order to solve this problem, regularization methods should be added within the network. Here, we adopt a Dropout [47] layer after the GRU network, as well as after the fully connected layer. Dropout layer enables the network to ignore those neurons that are randomly selected during the process of forward propagation. Therefore, the network will not rely too much on some local features. In our proposed model, we only use dropout during training process, but not in testing process, and the dropout ratio is set to be 0.3.

4. Experiments

4.1. Case 1: High speed CNC machine tool wear dataset

4.1.1. Descriptions of datasets

The first experiment is a high speed CNC machine running under dry milling operations [48]. This dataset is presented on the “prognostic data challenge 2010” database [25]. The experimental platform and the details are shown in Figure 8. In this experiment, six cutters are used to cut over an identical workpiece while each cutter made 315 cuts. When training and testing our model, six channels of data including forces and vibrations are used. A LEICA MZ12 microscope was utilized to measurethe flank wear of each flute when the experiment was finished. The values of the wear were then taken as the target value.

In this dataset, six cutting tools are used to do the experiment, which means six collections of data ($\mathbf{C}_1$, $\mathbf{C}_2$, ⋯, $\mathbf{C}_6$) can be used. To compare with the results in [17], we adopt three cutting tools, i.e. three data collections $\mathbf{C}_1$, $\mathbf{C}_4$ and $\mathbf{C}_6$ as our training and testing sets here. Each data collection contains 315 samples, corresponding to 315 tool wear. To make good use of this dataset, a three-fold strategy is adopted. Among these three data collections, two of them are taken as the training set and the other is the testing one. As a result, we get three cases. For example, when $\mathbf{C}_1$ is testing set and $\mathbf{C}_4$, $\mathbf{C}_6$ are training test, this case is denoted as c_1. The other two cases c_4, c_6 can be deduced from the above example. The details of these three cases are shown in table 1.

Table 1 Details of the three dataset cases

Set Name	Training Set	Testing Set	Number of samples in Training Set	Number of samples in Testing Set
c_1	$\mathbf{C}_4 + \mathbf{C}_6$	$\mathbf{C}_1$		
c_4	$\mathbf{C}_1 + \mathbf{C}_6$	$\mathbf{C}_4$	630	315
c_6	$\mathbf{C}_1 + \mathbf{C}_4$	$\mathbf{C}_6$		

As the sampling frequency is too high, for each channel, the sampled sequence is divided by 512 to get several sections, and the first forty sections are used. As a result, each original sequence is transformed into a datum with a length of 40, and therefore at each time step, the dimensionality is 3072 (6 channels). As descripted above, in the training process, the input shape of the network is 630×40×3072 and in the testing process, that of the network is 315×40×3072.

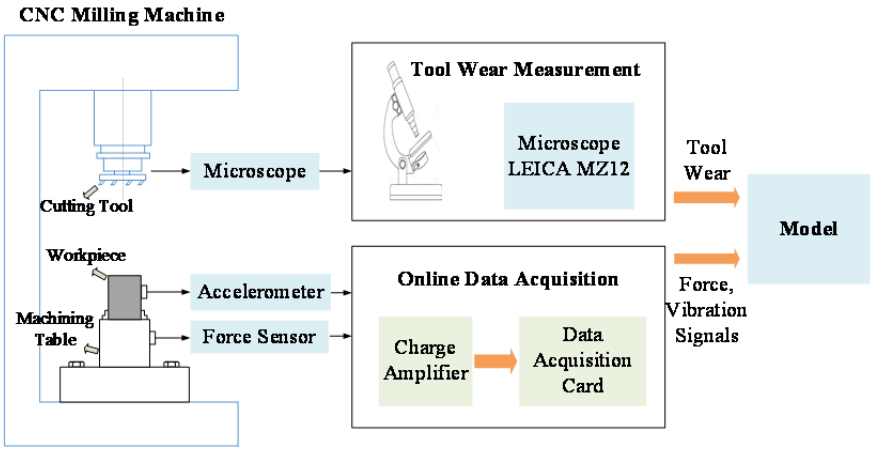


Figure 8. Details of the CNC machine and the data collected system

4.1.2. Experiment setup

The following models shown in table 2 will be compared with our proposed MCGRU model. Regression models including LR, SVR and MLP, cannot process sequential data directly, and hence we firstly extract the related features. Here, ten features, containing statistical features, frequency features and time-frequency features are extracted from raw data. Details are shown in table 3. As there are six channels of signals, the dimensionality of the input is 60. In LR, there is no hyper parameter. In SVR, the regularization parameter is set as 0.1 and the kernel is Radial Basis Function (RBF). As for the MLP, the parameters of three hidden layers are set as (140, 280, 900) and we choose ReLU as the activation function.

The other compared models are able to address sequential data directly. The input shape is therefore 40×3072. The five-layer CNN has the same structure as one branch of our proposed MCGRU. The kernel sizes in the two

convolutional layers are 1 and 7, quantities of kernels are 32 and 64, and the pooling size is set as 2. The setting of the MCNN is the same as that of our MCGRU. As for the basic recurrent models, including RNN, LSTM and GRU, the quantity of units is set as 192. And for the deep recurrent models, including Deep RNN, Deep LSTM, and Deep GRU, the quantity of the units in two layers is set as (180, 240). The CBLSTM, that is Convolutional Bi-Directional LSTMs, is proposed by Zhao et al. [17]. Here, the same settings in [17] are adopted for this model. The CGRU has a five-layer CNN with the same settings as the previous CNN model and a two-layer GRU with units (180, 240).

Table 2 Details of the compared models

Models	**Details**	**Input of the model**
LR	Linear Regression	Features extracted from raw signals
SVR	Support Vector Regression	
MLP	Multi-layer Perceptron	
CNN	Five-layer Convolutional Neural Network	Raw signals
MCNN	Multi-scale Convolutional Network	
RNN	Basic RNN	
Deep RNN	A two-layer RNN	
LSTM	A one-layer LSTM	
Deep LSTM	A two-layer LSTM	
GRU	A one-layer GRU	
Deep GRU	A two-layer GRU	
CBLSTM	Convolutional Bi-Directional LSTMs [18]	
CGRU	Convolutional GRU	

In our proposed MCGRU, from branch 1 to branch 6, the kernel size of the convolutional layers is set as (1, 1), (1, 3), (1, 5), (1, 7), (1, 9), (1, 11) , and the quantity of kernels is set as (32, 64). The kernel size of the pooling layer in all the branches is set as 2. Here, as the input shape is 40×3072, and the zero padding is adopted, the output shapes of each branch are the same, that is 40×32. Then, these six outputs are concatenated to get an output shape of 40×192. The quantity of units in the next two GRU layers is set as (180, 240) and the output units of the fully connected layer is set as 120. All of the activation functions in our model are Rectified Linear Unit (ReLU).

To evaluate the capability of the previous models, the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) are adopted. The MAE focuses on the average magnitude of the errors, without considering their direction. As a quadratic scoring rule, the RMSE measures the average magnitude of the error. The MAE is given in Eq. (20), and the RMSE is given by:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}_i - y_i\right)^2} \tag{21}$$

where $\hat{y}_i$ represents the predicted tool wear and y_i is the actual tool wear.

These models are trained and tested using a Linux Server with two NVIDIA 1080Ti GPUs and a 4.2 GHz INTEL i7-7700K CPU.

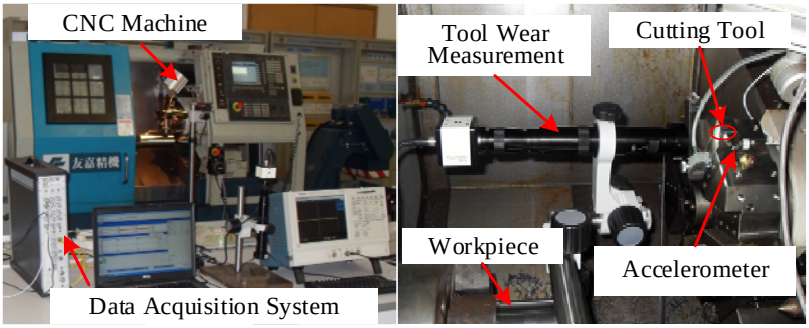
Table 3 Details of the extracted features

Domains	Features	Expressions
Statistical	Root Mean Square	$\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}$
	Variance	$\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{x}\right)^2$
	Maximum	$\max\left(x\right)$
	Skewness	$E\left[\left(\frac{x-\mu}{\sigma}\right)^3\right]$
	Kurtosis	$E\left[\left(\frac{x-\mu}{\sigma}\right)^4\right]$
	Peak-to-Peak	$\max\left(x\right) - \min\left(x\right)$
Frequency	Spectral Skewness	$\sum_{i=1} k\left(\frac{f_i - \overline{f}}{\sigma}\right)^3 S\left(f_i\right)$
	Spectral Kurtosis	$\sum_{i=1} k\left(\frac{f_i - \overline{f}}{\sigma}\right)^4 S\left(f_i\right)$
	Spectral Power	$\sum_{i=1}^{k}\left(f_i\right)^3 S\left(f_i\right)$
Time-Frequency	Wavelet Energy	$\sum_{i=1}^{N} \omega t_\phi^2\left(i\right) / N$

4.2. Case 2: Experiment of the reliability of CNC machine tool

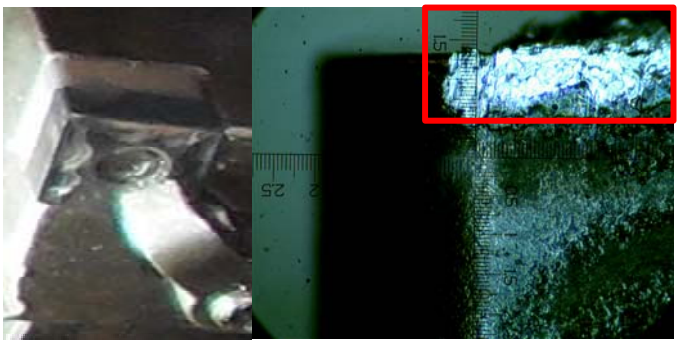
4.2.1. Descriptions of datasets

The second experiment is of the reliability test of CNC machine tool. It is carried out on a CNC machine, as showed in figure 9. The cutting tool, as showed in figure 10, is utilized to process a 45# steel bar, and the relating parameters are shown in table 4.



(a) (b)

Figure 9. CNC machine system for tool wear testing. (a) CNC machine and data acquisition system. (b) Tool wear monitoring system.



(a) (b)

Figure 10. Tool wear. (a) Tool used for processing. (b) Wear in tool.

Table 4 Details of the second experimental case.

Parameters	**Values**
CNC Machine	FEELER FTC-20 CNC machine
Cutting Tool	Korea KORLOY, CNMG120408-HM
Cutting Speed	200 m/min
Feed Rate	0.15 mm/r
Cutting Depth	2 mm
Workpiece	Material: GB 45# steel Diameter: 100 mm Length: 300 mm
Vibration Signal	Sampling frequency: 32768 Hz Sampling time: 2 s
Acoustic Emission Signal (AE-RMS)	Sampling frequency: 25000 Hz Sampling time: 2 s
Current Signal of Spindle Motor	Sampling frequency: 32768 Hz Sampling time: 2 s
Surface Roughness Measuring Instrument	Mitutoyo SJ-201P
Tool Wear Measuring Instrument	MTO MZDH0670 microscope

As shown in table 4, three channels of data are sampled, including the vibration signal, the AE-RMS signal and the current signal of the spindle motor. The tool wear corresponding to each working condition is also measured

and recorded to be the label. Here, we did the experiment with 9 cutting tools and got 840 samples in total. All of the cutting tools have the same initial tool wear. Each sample corresponds to a working condition. To be different from the first experimental case, in this case, we randomly chose samples from all of the cutting tools as the testing set and the training set. Both the training set and the testing set contain samples from the nine cutting tools. As a result, the training set include 735 of the 840 samples, and the rest are used as the testing set. Similarly, as the sampling frequency is too high, for each signal, the sequence is divided by 256 to get several sections, and the first 20 sections are selected. Hence, each data sample is transformed into a sequential datum with a length of 20, and at each time step, the dimensionality is 768 (3 channels). As descripted above, in the training process, the input shape of the network is 735×20×768 and in the testing process, that of the network is 105×20×768.

4.2.2. Experimental setup

The setup of this second experiment is almost the same as the first one. Same compared models and same settings for these models are adopted. The two indexes used to evaluate the performance of the models are the MAE and the RMSE. The models are trained and tested using a Linux Server with two NVIDIA 1080Ti GPUs and a 4.2 GHz INTEL i7-7700K CPU.

4.3. Results

In this section, the comparison based on the MAE and RMSE of the above models are shown. Table 5 shows the MAE of each model in the first experimental case, while table 6 shows the RMSE. The MAE and RMSE of each model in the second experiment are shown in table 7.

As shown in the tables, the regression models LR, SVR and MLP have shown their capability to make prediction of tool wear based on the features extracted from raw data, while they are not as good as the convolutional models and recurrent models. Linear Regression performs worst in this task because it is a linear model in nature, which cannot make full use of the extracted features to make predictions. The SVR and the MLP perform better because the nonlinearity is introduced into the models so that the relationships among different features can be better explored. In SVR, by adopting different kernels, the samples can be mapped to a high dimensional space. The RBF kernel we chose here shows its power to address regression tasks. The MLP is able to search an efficient mapping mode actively, which is effective and different from SVR.

However, compared to the above regression models, the convolutional models and recurrent models can address raw data to learn significant features and representations, which enables them to have better performance. By choosing different kernels, CNN is able to extract local or abstract features. In this task, MCNN performs better than CNN because it contains kernels of different sizes to extract local and abstract features in the same time. The depth and width we introduced here in these two models also help in predicting tool wear. Long-term dependencies

in sequential data cannot be discovered by convolution operation, but it can be captured by recurrent models. As shown in the tables, in this task of addressing time series data, recurrent models perform slightly better than convolutional models. Here, LSTM and GRU are better than basic RNN because their gates enable them to be more powerful to capture long-term dependencies. What's more, in this task, the amount of data is not large, which shows the GRU's advantages under the situation where there is no sufficient data. Here, the GRU performs better than LSTM. And as expected, the three deep recurrent models perform better than the three normal ones. In [17], the proposed Convolutional Bi-Directional LSTMs combines a local feature extractor CNN with a temporal encoder deep Bi-Directional LSTMs, which is able to excavate useful features hidden in the raw sensory data in both forward and backward ways. The CBLSTMs performs better than most of the above convolutional and recurrent models. As for the CGRU, it performs well, as the GRU is also able to learn significant representations on basis of the features extracted by the CNN. Specially, in the second experimental case, the deep GRU and CGRU perform even better than the CBLSTMs, which shows the power of the GRU in dealing with small amount of data. Our proposed model, the MCGRU, performs best among these compared models.

The result reveals that there is much information hidden behind raw sensory data that cannot be discovered by human designed features, while Multi-scale CNN is able to filter the noise from real working environment and explore the information as much as possible. The deep GRU is able to excavate the temporal information to find a more accurate relationship between the input and output, namely the raw sensory data and the predicted tool wear. As the network goes wider, more meaningful features of different time scales can be discovered, and as it goes deeper, the abstract and significant representations can be learnt. The combination of the multi-scale features extractor Multi-scale CNN and the temporal encoder deep GRU is therefore proven to perform well in the task of tool wear prediction.

Table 5 The MAE of all the models in the first experimental case. Bold face indicates the best performance.

Models	**Datasets**		
	c_1	c_4	c_6
LR	20.02	39.00	47.19
SVR	21.41	30.84	28.32
MLP	16.50	23.95	16.82
CNN	15.29	17.15	11.72
MCNN	14.34	16.51	11.07
RNN	14.37	14.85	15.78
Deep RNN	12.93	13.86	13.43
LSTM	13.86	14.52	13.64

Deep LSTM	12.59	13.30	11.46
GRU	13.36	14.29	12.12
Deep GRU	11.41	13.09	10.55
CBLSTM [18]	7.50	6.10	8.10
CGRU	9.65	10.98	10.06
MCGRU	**6.87**	**6.04**	**7.42**

Table 6 The RMSE of all the models in the first experimental case. Bold face indicates the best performance.

Models	**Datasets**		
	C_1	C_4	C_6
LR	27.89	40.98	48.86
SVR	25.89	35.95	33.72
MLP	21.71	26.04	20.56
CNN	20.15	18.82	13.35
MCNN	21.09	18.26	12.69
RNN	17.48	18.38	19.56
Deep RNN	15.65	18.19	16.63
LSTM	16.76	17.64	16.26
Deep LSTM	16.32	17.27	13.79
GRU	16.57	17.48	13.93
Deep GRU	14.41	16.88	12.59
CBLSTM [18]	10.80	7.10	9.80
CGRU	14.38	13.95	12.27
MCGRU	**8.27**	**6.81**	**9.08**

Table 7 The MAE and RMSE of all the models in the second experimental case. Bold face indicates the best performance.

Models	**MAE**	**RMSE**
LR	8.28	9.32
SVR	8.27	8.78
MLP	7.76	8.54
CNN	6.53	8.27
MCNN	6.42	8.13
RNN	6.41	8.01

Deep RNN	6.37	7.93
LSTM	6.26	7.84
Deep LSTM	5.75	7.61
GRU	5.87	7.42
Deep GRU	5.02	6.37
CBLSTM[18]	5.17	7.02
CGRU	4.79	6.10
MCGRU	**3.71**	**4.96**

To be more specific, the prediction of the tool wear, the corresponding actual tool wear, and the error between these two values are illustrated in figure 11, 12, 13 and 14. It is shown that in the first three figures, i.e., in the results of the first experimental case, the trend of the degradation of the cutting tool is robustly captured and error is acceptable. Specially, in figure 14, nine ascending curves can be found in the curve of actual tool wear, that's because in the second experimental case, we have sampled data from all of the nine cutting tools to be the testing set and each ascending curve represents the data from a cutting tool. In this case, the results are also satisfying. Moreover, for each epoch, it consumes about 1s to train. When testing, it consumes only 0.8s to predict the tool wear of about 300 samples, which means that our proposed model is efficient enough to be used in real-time prediction.

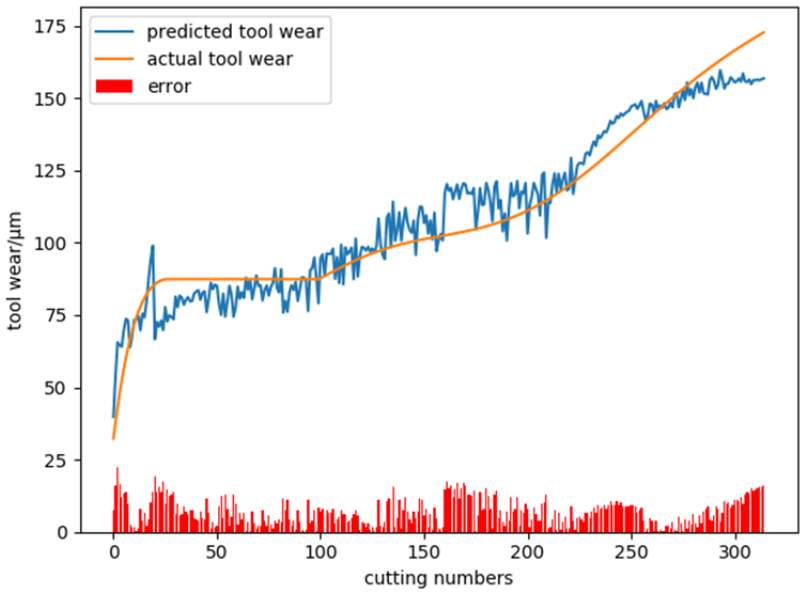


Figure 11 Results of the first experimental case, when c_1 is the testing set: the prediction of the tool wear, the corresponding actual tool wear, and the error between these two values.

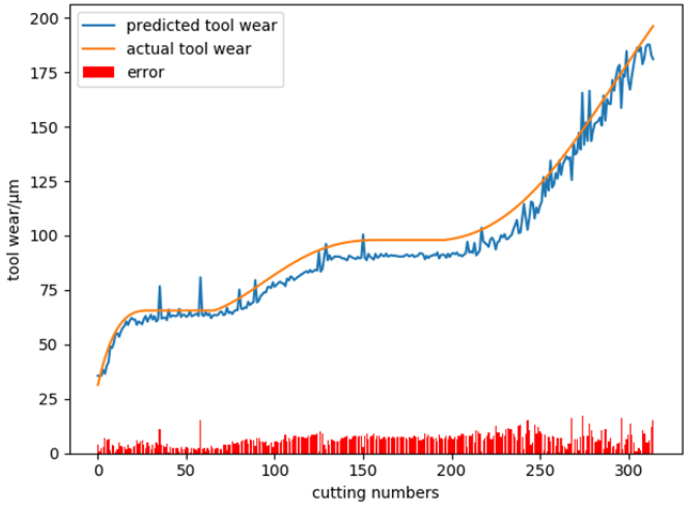


Figure 12. Results of the first experimental case, when c_4 is the testing set: the prediction of the tool wear, the corresponding actual tool wear, and the error between these two values.

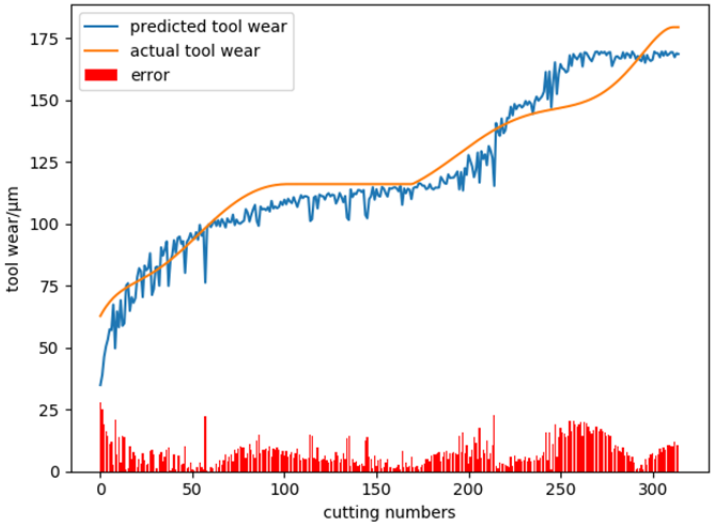


Figure 13. Results of the first experimental case, when c_6 is the testing set: the prediction of the tool wear, the corresponding actual tool wear, and the error between these two values.

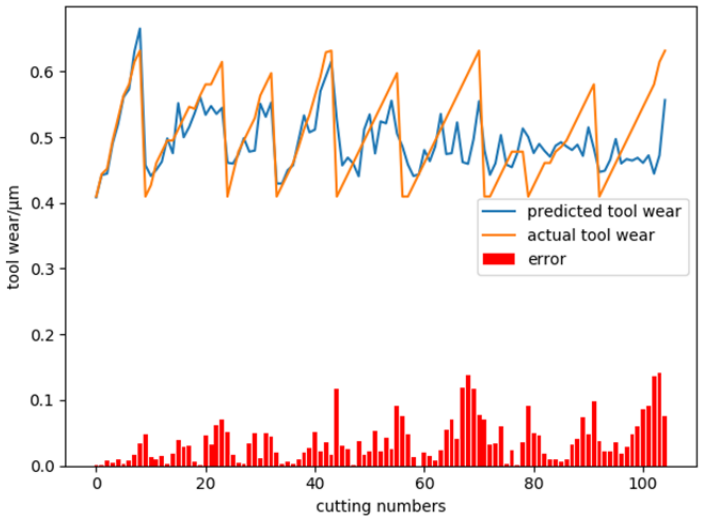


Figure 14. Results of the second experimental case: the predicted tool wear, the actual tool wear and the error.

5. Discussion

In this section, we discuss the impact of the number of the branches in the Multi-scale CNN and the influence of the depth of the GRU. Some insights and motivation for the future steps are also discussed.

1) As we go wider by using multi branches to extract more features, it is important to point out that this operation increases the model's parameters, which results in the difficulty in training and the risk of over fitting. Here, based on dataset c_1 we compare six numbers of branches (2, 4, 6, 8, 10, 20) in a MCGRU and the MAE and RMSE results are illustrated in table 8. It shows that as the number of branches increases, the performance of the model gets better and then remains almost the same, and when there are 10 or 20 branches, the performance gets worse, which means that blindly increasing the number of branches does harm to the model and cannot improve its performance. Here we finally adopt 6 branches of CNN in our MCGRU.

Table 8 Comparison of different numbers of branches in the MCGRU

Dataset	C_1	
Number of branches	**MAE**	**RMSE**
2	9.25	14.50
4	7.61	11.97
6	**6.87**	**8.27**
8	7.04	8.95
10	9.38	12.74
20	9.71	13.49

2) The depth of the model also affects the performance of the model. We change the layers of GRU in the MCGRU to explore the impact of the depth. The number of layers of GRU is set as (2, 4, 6, 8) and the results are shown in table 9. It is clear that the performance of these four models is almost the same. A reasonable explanation is that in our two experiments, there is no sufficient labeled data and therefore a shallow depth of GRU is powerful enough to discover the information behind the data. When there is a large amount of data, a GRU of more layers can be tried to further improve the capability of the model.

Table 9 Comparison of different numbers of GRU layers in the MCGRU model

Dataset	C_1	
Number of GRU layers	**MAE**	**RMSE**
2	**6.87**	**8.27**
4	7.02	8.92
6	7.13	9.45
8	6.97	8.67

3) The robustness of a model is important to evaluate the performance of a model. In real working environment,

the quality of the samples signals may be influenced by the noise. It is important and interesting to build a model that is robust when there is a large amount of noise. And in our settings, different signals are combined directly, it is meaningful to design a better way of fusing the data from different sensors.

6. Conclusion

In this paper, we proposed a Multi-scale Convolutional Gated Recurrent Unit Network (MCGRU) to address tool wear prediction task. We interpret the structure of this model by introducing the feature extractor: Multi-scale CNN and the encoder: Deep GRU. The Multi-scale CNN is able to extract both local and abstract features by kernels of different sizes, and the Deep GRU is capable of capturing long-term dependencies and learning significant representations based on the features extracted in Multi-scale CNN. Moreover, the GRU performs better when there is no sufficient labeled data in real working conditions. Profiting from these advantages, the MCGRU is able to make accurate and effective tool wear prediction based on raw sensory data, without expert knowledge and feature engineering. Its satisfactory performance is further verified by two experimental cases and the comparisons with other models.

Competing interests

There are no competing interests in this paper.

Author's contributions

Weixin Xu: Writing, review & editing; Huihui Miao: review & discussion; Zhibin Zhao: review; Jinxin Liu: review; Chuang Sun: Revision, editing & supervision; Ruqiang Yan: Review & supervision.

Funding

This work was supported in part by Natural Science Foundation of China (No. 51835009, No. 51705398), Shaanxi province 2020 natural science basic research plan (No. 2020JQ-042), and Aeronautical Science Foundation(No. 2019ZB070001).

Acknowledgements

The authors would like to thank Mr. Tianfu Li for his help on this work.

Author details

School of Mechanical Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

References

[1] Liuyang Song, Huaqing Wang, and Peng Chen, Step-by-step Fuzzy Diagnosis Method for Equipment Based on Symptom Extraction and Trivalent Logic Fuzzy Diagnosis Theory, IEEE Transactions on Fuzzy Systems, 2018, 26(6):3467-3478.

[2] Mo Z, Wang J, Zhang H, et al. Weighted Cyclic Harmonic-to-Noise Ratio for Rolling Element Bearing Fault Diagnosis. IEEE Transactions on Instrumentation and Measurement, published online, 2019.

[3] L.L. Cui, X. Wang, Y.G. Xu, H. Jiang, J.P. Zhou, A novel Switching Unscented Kalman Filter method for remaining useful life prediction of rolling bearing, Measurement 135 (2019) 678–684.

[4] Huaqing Wang, Shi Li, Liuyang Song, Lingli Cui, A novel convolutional neural network based fault recognition method via image fusion of multi-vibration-signals, Computers in Industry,2019. 105:182-190.

[5] D.E.D.J.I.J.M.T.M. Snr, Sensor signals for tool-wear monitoring in metal cutting operations—a review of methods, International Journal of Machine Tools and Manufacture, 40 (2000) 1073-1098.

[6] N. Ghosh, Y.B. Ravi, A. Patra, S. Mukhopadhyay, S. Paul, A.R. Mohanty, A.B. Chattopadhyay, Estimation of tool wear during CNC milling using neural network-based sensor fusion, Mechanical Systems & Signal Processing, 21 (2007) 466-479.

[7] Y.-C. Yen, J. Söhner, B. Lilly, T.J.J.o.m.p.t. Altan, Estimation of tool wear in orthogonal cutting using the finite element analysis, Journal of materials processing technology, 146 (2004) 82-91.

[8] J.S. Strenkowski, J.J.J.o.E.f.I. Carroll, A finite element model of orthogonal metal cutting, Journal of Engineering for Industry, 107 (1985) 349-354.

[9] E. Ceretti, P. Fallböhmer, W.T. Wu, T.J.J.o.M.P.T. Altan, Application of 2D FEM to chip formation in orthogonal cutting, Journal of Materials Processing Technology, 59 (1996) 169-180.1

[10] I.S. Jawahir, O.W. Dillonjr., A.K. Balajj, M. Redetzky, N.J.M.S. Fang, Technology, PREDICTIVE MODELING OF MACHINING PERFORMANCE IN TURNING OPERATIONS, Machining Science and Technology, 2 (1998) 253-276.

[11] T. Ozel, M. Lucchi, C.A. Rodríguez, T. Altan, Prediction of chip formation and cutting forces in flat end milling: comparison of process simulations with experiments, Technical Paper-Society of Manufacturing Engineers. AD, (1998) 1-6.

[12] M. Shatla, Y.-C. Yen, T. Altan, Tool-workpiece interface in orthogonal cutting-application of FEM modeling, TRANSACTIONS-NORTH AMERICAN MANUFACTURING RESEARCH INSTITUTION OF SME, (2000) 173-178.

[13] T.J. Ko, W.C. Dong, Cutting state monitoring in milling by a neural network, International Journal of Machine Tools & Manufacture, 34 (1994) 659–676.

[14] ÖZEL Tugrul, K. Yigit, Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks, International Journal of Machine Tools & Manufacture, 45 (2005) 467-479.

[15] V.S. Sharma, S.K. Sharma, A.K. Sharma, Cutting tool wear estimation for turning, Journal of Intelligent Manufacturing, 19 (2008) 99-108.

[16] K.V. Rao, B.S.N. Murthy, N.M. Rao, Prediction of cutting tool wear, surface roughness and vibration of work piece in boring of AISI 316 steel with artificial neural network, Measurement, 51 (2014) 63-70.

[17] R. Zhao, R. Yan, J. Wang, K. Mao, Learning to monitor machine health with convolutional bi-directional LSTM networks, Sensors, 17 (2017) 273.

[18] J. Wang, W. Peng, R.X. Gao, Enhanced particle filter for tool wear prediction, Journal of Manufacturing Systems, 36 (2015) 35-45.

[19] Z. Rui, D. Wang, R. Yan, K. Mao, S. Fei, J. Wang, Machine Health Monitoring Using Local Feature-based Gated Recurrent Unit Networks, IEEE Transactions on Industrial Electronics, PP (2017) 1-1.

[20] T. Juri, S. Emilia, P. Eduardo, R. Stefano, C. Paolo, Validation of Inter-Subject Training for Hidden Markov Models Applied to Gait Phase Detection in Children with Cerebral Palsy, Sensors, 15 (2015) 24514-24529.

[21] K. Wei, W. Lenan, Mobile location with NLOS identification and mitigation based on modified Kalman filtering, Sensors, 11 (2011) 1641-1656.

[22] H.-D. Yang, Sign language recognition with the kinect sensor based on conditional random fields, Sensors, 15 (2015) 135-147.

[23] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw, 61 (2015) 85-117.

[24] K. Cho, B.V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Computer Science, (2014).

[25] P.d.c. PHM Society, https://www.phmsociety.org/competition/phm/10, 2010.1

[26] Y.L. Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten Digit Recognition with a Back-Propagation Network, Advances in Neural Information Processing Systems, 2 (1990) 396--404.

[27] K. Jarrett, K. Kavukcuoglu, Y. LeCun, What is the best multi-stage architecture for object recognition?, 2009 IEEE 12th international conference on computer vision, IEEE, 2009, pp. 2146-2153.

[28] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, International Conference on Neural Information Processing Systems, 2012.

[29] O. Abdel-Hamid, A.R. Mohamed, J. Hui, G. Penn, Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition, IEEE International Conference on Acoustics, 2012.

[30] Y. Kim, Convolutional Neural Networks for Sentence Classification, Eprint Arxiv, (2014).

[31] Z. Rui, K. Mao, Topic-Aware Deep Compositional Models for Sentence Classification, IEEE/ACM Transactions on Audio Speech & Language Processing, 25 (2017) 248-260.

[32] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, (2015).

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going Deeper with Convolutions, (2014).

[34] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation, 9 (1997) 1735-1780.

[35] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: continual prediction with LSTM, International Conference on Artificial Neural Networks, 2002.

[36] S. Hochreiter, Untersuchungen zu dynamischen neuronalen Netzen, Diploma, Technische Universität München, 91 (1991).

[37] R Huang, Y Liao, S Zhang, W. Li. Deep Decoupling Convolutional Neural Network for Intelligent Compound Fault Diagnosis, IEEE Access, 2019, 7: 1848-1858.

[38] C. Sun, M. Ma, Z. Zhao, S. Tian, R. Yan, X. Chen, Deep Transfer Learning Based on Sparse Auto-encoder for Remaining Useful Life Prediction of Tool in Manufacturing, IEEE Transactions on Industrial Informatics.

[39] F. Jia, Y. Lei, J. Lin, X. Zhou, and N. Lu, Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data, Mechanical Systems and Signal Processing, vol. 72, pp. 303-315, 2016.

[40] H. Shao, H. Jiang, and H. Zhang, Electric Locomotive Bearing Fault Diagnosis Using a Novel Convolutional Deep Belief Network, IEEE Transactions on Industrial Electronics, vol. 65, no. 3, pp. 2727-2736, 2018.

[41] C. Sun, M. Ma, Z. Zhao, X. Chen, Sparse Deep Stacking Network for Fault Diagnosis of Motor, IEEE Transactions on Industrial Informatics, 14 (2018) 3261-3270.1

[42] E.O. Ezugwu, S.J. Arthur, E.L. Hines, Tool-wear prediction using artificial neural networks, Journal of Materials Processing Technology, 49 (1995) 255-264.

[43] V. Nair, G.E. Hinton, Rectified Linear Units Improve Restricted Boltzmann Machines, International Conference on International Conference on Machine Learning, 2010.

[44] G.E. Hinton, Learning multiple layers of representation, Trends in Cognitive Sciences, 11 (2007) 428-434.

[45] Y. Bengio, Learning deep architectures for AI, Foundations and trends® in Machine Learning, 2 (2009) 1-127.

[46] T. Tieleman, G. Hinton, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural networks for machine learning, 4 (2012) 26-31.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research, 15 (2014) 1929-1958.

[48] X. Li, B. Lim, J. Zhou, S. Huang, S. Phua, K. Shaw, M. Er, Fuzzy neural network modelling for tool wear estimation in dry milling operation, Annual conference of the prognostics and health management society, 2009, pp. 1-11.