Tool Wear State Recognition Based on 1D-CNN

Wenji Wang
Zibo Vocational Institute, Zibo, Shandong, China

Abstract: Machine learning classification models have the problems of complex feature engineering and unsatisfactory state recognition. In this paper, a deep learning network, One-dimensional convolutional neural networks (1D-CNN), is proposed to recognize the state of tool wear. After the original data is cleaned and pre-processed, it is directly put into the 1D-CNN model for feature self-extraction and state recognition, which improves the automation, accuracy and efficiency of the whole recognition process.

Keywords: Data preprocessing; Deep Learning; 1D-CNN; State Recognition.

1. Data Preprocessing

Cleaning and preprocessing for dirty and bad original data: head and tail knife invalid data removal, data downsampling, outlier filtering. The data preprocessing of c1 is illustrated by the first, 11th, 51, 101, 151, 201, 251, 301, 314 times in total. Figure 1 is the original 7-channel signal data of the 51st cutting of c1. It can be seen from the figure that the signal data at the head of the signal data is gradually larger caused by the milling feed process, and the corresponding figure has a section of data at the end of the data caused by the milling retracting tool. These two sections of data are abnormal milling signal data, so they are invalid data that need to be manually eliminated. Figure 3 shows the original data of 7 signal channels of c1's first tool walk. It can be seen from the figure that there are obvious sharp spikes and sharp drops in the middle of the Y-axis milling force signal. This signal data is an outlier point and needs filtering processing.

1.1 Invalid Data Truncation

For the head and tail of invalid data, the truncation method is taken here to directly eliminate, the detailed steps are as follows:

(1)Find the critical value of the invalid data of the head and tail of each cutting process: the upper quartile value Q of the original data of each cutting process.

(2)The head of the original data is to find the first data greater than or equal to Q from the front to the back as the end position of the feed, and eliminate all the signal data less than Q in the front; In the same way, the tail from the back to find the first data greater than or equal to Q as the starting position of the knife, eliminate all the data less than Q behind.

As can be seen from Figure 1, the data in the two ellipses at the head and tail areinvalid data formed by feed and retracting, which is directly eliminated by truncation method. Matlab uses the quantile() function to find the upper quartile value of the matrix by column:

Q=quantile(X,0.25,1)

Where X is of size (m,n), 1 represents the upper quartile value of each column by column, and Q is the calculated upper quartile value of size (1,n).

Figure 1 and Figure 2 below are the comparison of 7 channel signal data before and after preprocessing of the 51st milling walk of cutter 1. As can be seen from the comparison of the two figures, two segments of invalid data in the process of feed and back cutting are eliminated, the amount of data is reduced from 200,000 to 20,000, and the trend of signal change is well retained, achieving the purpose of data preprocessing.

1.2 Data Downsampling

At the original sampling frequency, there are about 200,000 data collected from each tool walk, which is too large, and the direct use of the data is easy to cause the subsequent calculation is too complicated. Therefore, this paper uses the method of downsampling to process the data to speed up the calculation of later feature extraction. At the same time, the method of downsampling ensures the data change trend of the original signal each time. Here 1:10 downsampling is used, that is, the

frequency is 5KHz, and the data of each milling cutter walk is about 10,000 to 25,000.

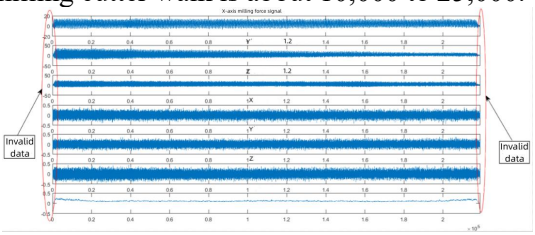

Figure 1. Invalid Data at the Beginning and End of the 51st Milling Process of C1

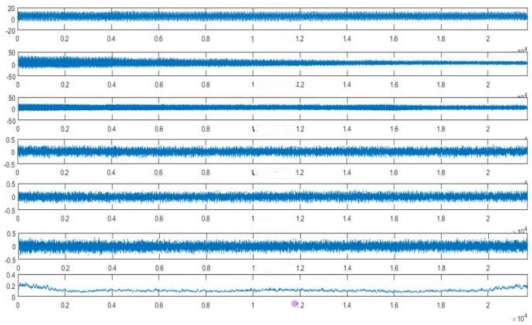

Figure 2. Data after Pretreatment of the 51st Milling Process of C1

1.3 Outlier Removal

In this paper, hampel filtering is used to remove[1] the outliers such as sudden spike and sudden drop in the signal data. Principle of hampel filtering algorithm:

(1)First filter all the outlier points in the signal data. Generate a sliding window of fixed length for each data point in the signal, calculate the signal median of the window segment, and further calculate the median deviationof each data point from this median value, where the outlier that differs from the median by more than two standard deviations is the outlier point to be found.

(2)Replace the outlier in this segment with the median of the sliding segment. The specific operations of hampel filtering in Matlab are as follows:

[outX,outPos,xMedian,xSigma]=hampel(X,k,nSigma)

Where, X: the signal that needs outlier filtering processing; k: half the lengthof the sliding window; nSigma: the upper and lower bounds of the data that are classified as outliers; outX: output signal after hampel filtering; outpost: the position where the outlier is found and processed; xMedian: the median of a sliding window used to replace outliers; xSigma: the standard deviation of the absolute deviation of each data point from the median of the observation window.

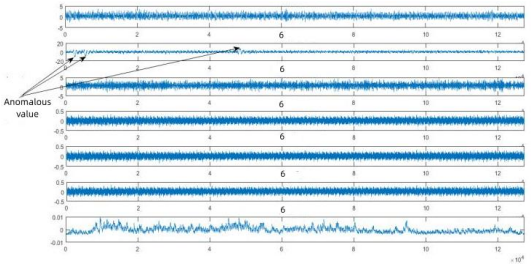

Figure 3. C1 First Milling Process Anomaly Data

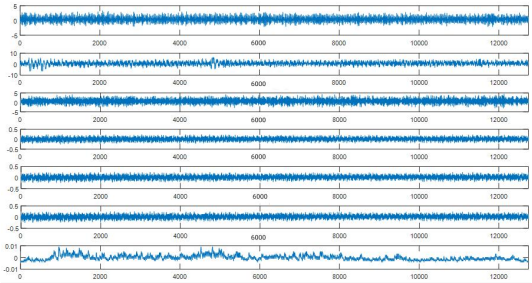

Figure 4. Data of C1 after Pretreatment of the First Milling Process

The comparison of signal data before and after the pretreatment of 7 channels for the first milling of tool 1 was used to further verify the effect of hampel filtering on the pretreatment of outliers. Figure 3 is before pretreatment and Figure 4 is after pretreatment. As can be seen from the comparison of the two figures, the peaks and valleys existing in the middle of the signal have been effectively processed, and the data after pretreatment is smoother, with less obvious outliers and more availability.

After a series of pre-processing including invalid data truncation at the head and tail, outlier filtering and downsampling, all signal data is turned into a clean signal with high usability, which is convenient for the subsequent input into the 1D-CNN network for status recognition.

2. 1D Convolutional Neural Network

The layer types and connection modes adopted by each layer of the 1D-CNN network built in this paper, the changes of input and output vectors and other parameter settings are shown in detail[2] in Figure 5 below. The question mark in the figure represents the number of unknown training batch samples. The building process for each layer is explained in detail below.

Input data: After a series of preprocessing described in Chapter 2, the raw data of the tool is intercepted into several time slices of fixed length in order to input it into the 1D-CNN

network. Each piece of data entered into the 1D-CNN network is recorded with a fixed time step of 80, and each piece of data stores the signal data of 7 channels, resulting in an 80×7 matrix. In order to make the constructed 1D-CNN neural network model have a wider platform adaptability, the data transmitted to the neural network must be converted into a plane vector with a growth degree of 560. Therefore, when building the network using keras in python, the first layer of the network must be reshaped to its original shape of 80×7.

The first layer of one-dimensional convolution layer: Defining 100 convolution cores of size 10 makes the network train 100 different features in the first layer. This calculation results in a matrix with an output of 71×100, where the weight of one convolution kernel occupies one column of the output matrix. The size of the selected kernel and the step size of the input data together determine that the number of weights for each convolution kernel is 71,and the activation function of the convolution layer is set to the ReLU function.

The second layer of one-dimensional convolution layer: 100 convolution kernel of size 10 are also defined and trained on this layer, and the output matrix of 62×100 can be obtained by the same calculation method. The third layer of maximum pooling layer: After convolution,a pooling layer of size 3 is chosen to reduce network complexity and prevent network training from overfitting. Therefore, the output of this layer becomes 20×100.

Layer 4 and Layer 5 One-dimensional CNN layer: Set these two layers to be exactly the same one-dimensional convolution layer as another convolution sequence for learning higher level features. 160 convolution kernels of size 10 are defined respectively. The output matrix sizes for these two layers are 11×160 and 2×160, respectively. The sixth average pooling layer: To avoid overfitting of the new convolution sequence, choose to build a pooling layer that takes the average of the two weights. The output feature weight size is reduced to half of the input size: 1×160.

Seventh layer Dropout layer[3]: In order to solve the low network generalization ability caused by the network's sensitivity to small data changes and improve the model's classification accuracy on unpredictable data, the Dropout layer of 0.5 is used here. 50% of the neurons are randomly selected and their weights are set to 0 so that they no longer function. Layer 8 Fully connected layer: Because eventually 3 different wear states have to be identified. So reducing this layer reduces the output to 3. To achieve the goal that the outputs of the network sum to 1, the Softmax activation function is used as the classifier,and the output value represents the probability that the input data records belong to 3 classes respectively.

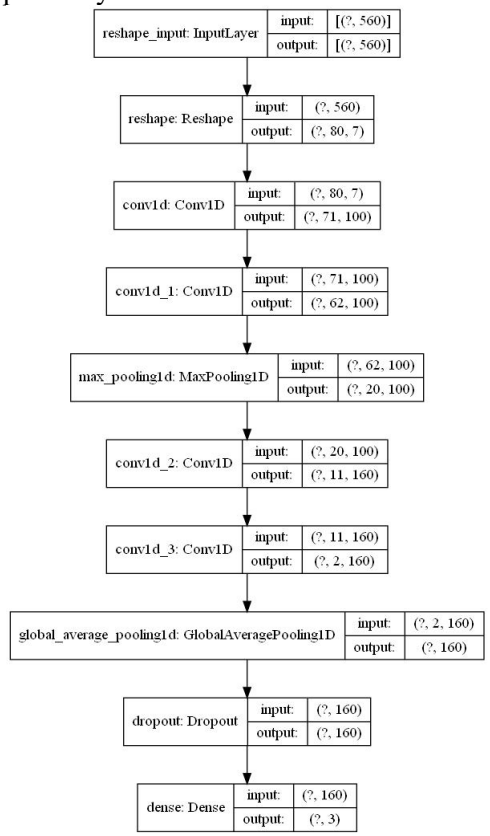

Figure 5. Establishing the Input and Output Relationships of 1D-CNN Layers

3. Recognition of Tool Wear State Based on 1D Convolutional Network

The 1D-CNN network structure built in this paper is shown in Figure 6 below, which shows the name of each layer of the entire network, the connection mode between each layer, and the size and number of convolutional kernels used by each layer.

It is introduced above that each input data of 1D-CNN network must be taken as a matrix of (80×7). In order to expand the training data amount of 1D-CNN network, 70 80×7 data records are sequentially intercepted by milling cutter 1 as the training set for each cutting data.Therefore,a total of 22,050 data records are formed from the 315 tool walks of training

set c1, and one data record is generated for each milling tool walk of the test set for classification and recognition. Table 1 below shows the data record allocation of training set milling cutters 1, test set milling cutters 4 and 6 at each stage.

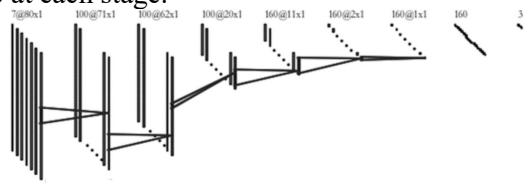

Figure 6. Shows The Structure of The 1D-CNN Network

Firstly, 22,050 80×7 data records generated by training set c1 were input into the constructed 1D-CNN classification model for training, and the trained network weigh and overall structure were obtained. Then, 315 80×7 data records generated by test set milling cutter 4 and 6 were input into the trained network for tool wear state prediction and recognition, respectively, and their respective classification results are shown in Figure 7 and Figure 8. The overall classification accuracy of test sets 4 and 6 is 94.92% and 88.57% respectively.

Table 1. Sample Division and Data Record Composition

Number of samples in each stage	Initial wear	Normal wear and tear	Sharp wear
Training set c1	(80×7)×70×30	(80×7)×70×210	(80×7)×70×75
Test set c4	(80×7)×1×30	(80×7)×1×210	(80×7)×1×75
Test set c6	(80×7)×1×30	(80×7)×1×210	(80×7)×1×75

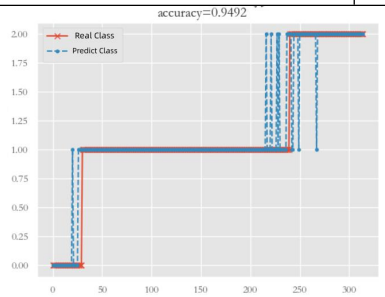

Figure 7. Wear state identification results of milling cutter 4 1D-CNN model

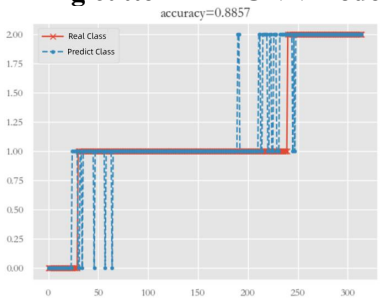

Figure 8. Wear State Recognition Results of Milling Cutter 6 1D-CNN model

REFERENCES

[1] Simon G D, Deivanathan R. Early detection of drilling tool wear by vibration data acquisition and classification[J]. Manufacturing Letters, 2019, 21: 60-65.

[2] Gomes M C, Brito L C, da Silva M B, et al. Tool wear monitoring in micromilling using Support Vector Machine with vibration and sound sensors[J]. Precision Engineering, 2021, 67: 137-151.

[3] Dou J, Xu C, Jiao S, et al. An unsupervised online monitoring method for tool wear using a sparse auto-encoder[J]. The International Journal of Advanced Manufacturing Technology, 2020, 106(5): 2493-2507.

