machines

MDPI

Article

Predicting Tool Wear with ParaCRN-AMResNet: A Hybrid Deep Learning Approach

Lian Guo and Yongguo Wang *

School of Mechatronics Engineering and Automation, Shanghai University, Shanghai 200444, China; godferyguo@shu.edu.cn
* Correspondence: ygwang@shu.edu.cn

Abstract: In the manufacturing sector, tool wear substantially affects product quality and production efficiency. While traditional sequential deep learning models can handle time-series tasks, their neglect of complex temporal relationships in time-series data often leads to errors accumulating in continuous predictions, which reduces their forecasting accuracy for tool wear. For addressing these limitations, the parallel convolutional and recurrent neural networks with attention-modulated residual learning (ParaCRN-AMResNet) model is introduced. Compared with conventional deep learning models, ParaCRN-AMResNet markedly enhances the efficiency and precision of feature extraction from time-series data through its innovative parallel architecture. The model adeptly combines dilated convolution neural network and bidirectional gated recurrent units, effectively addressing distance dependencies and enriching the quantity and dimensions of extracted features. The strength of ParaCRN-AMResNet lies in its refined ability to capture the complex dynamics of time-series data, significantly boosting the model's accuracy and generalization capability. The model's efficacy was validated through comprehensive milling experiments and vibration signal analyses, showcasing ParaCRN-AMResNet's superior performance. In evaluation metrics, the model achieved a MAE of 2.6015, MSE of 15.1921, R^2 of 0.9897, and MAPE of 2.7997%, conclusively proving its efficiency and accuracy in the precise prediction of tool wear.

Keywords: tool wear prediction; dilated convolution neural network; bidirectional gated recurrent unit; attention mechanism

Citation: Guo, L.; Wang, Y. Predicting Tool Wear with ParaCRN-AMResNet: A Hybrid Deep Learning Approach. *Machines* **2024**, *12*, 341. https://doi.org/10.3390/machines12050341

Academic Editor: Kai Cheng

Received: 4 April 2024
Revised: 4 May 2024
Accepted: 12 May 2024
Published: 15 May 2024

1. Introduction

With the rapid development of the manufacturing industry, an increasing demand for advanced machining technologies has been observed. In the machining process, wear on cutting tools lead to a reduction in product quality and a decrease in production efficiency [1,2]. Therefore, tool-condition monitoring (TCM) plays a crucial role in providing valuable guidance for the reasonable use of tools. For companies, a reliable TCM has a significant production value due to its potential in preventing unplanned downtime and avoiding corresponding economic losses [3,4]. Considering the dynamic and nonlinear nature of the tool wear process, influenced by complex working environments, the accurate prediction of tool wear continues to face significant challenges [5].

Facing predictive challenges, this study employed deep learning techniques. Unlike traditional machine learning models that require pre-defined feature extraction, deep learning autonomously learns from complex data, crucial for understanding nonlinear and multivariate tool wear processes [6]. Its robust adaptability and generalization also effectively handle real-world data inconsistencies and noise. This study aimed to develop a deep learning-based TCM model to enhance the accuracy and efficiency of cutting tool wear prediction, with convolutional neural networks (CNNs) [7] and recurrent neural networks (RNNs) [8] being the most utilized in recent studies.

CNNs have demonstrated superior feature extraction capabilities in large-scale image recognition tasks due to their unique convolutional and pooling layers [9]. For example,

Kumar et al. [10] employed a deep CNN architecture using images of surfaces machined without cutting fluid as inputs. By selecting the right training parameters, they classified cutting tool wear, reaching a model recognition and classification accuracy of 99.9%. Additionally, Lim et al. [11] conducted a comparison between DNN and CNN networks in the field of tool wear recognition. They found that CNNs are more reliable in using cropped images of machined surface contours to predict the amount of flank wear on tools during turning processes, achieving an accuracy rate of 98.9% and an average test RMSE of 2.0969. Meanwhile, García-Pérez et al. [12] employed multi-view camera technology, supplemented by data augmentation and class weighting, to manage the number of worn tools assessed and the costs associated with image collection. They considered and tested two CNN architectures, reaching an experimental accuracy as high as 97.8% (with a Matthews correlation coefficient of 0.955), and they were able to detect defects in various blade types. Zhang et al. [13] obtained the initial dataset through wavelet transformation, followed by the use of a conditional variational autoencoder with CNN to augment the dataset, addressing the issue of data imbalance. This augmented dataset served as the input for a CNN. Subsequently, they described tool wear using a multistage nonlinear Wiener process model. Brili et al. [14] implemented an infrared camera for process monitoring, capturing the visual and thermal states during the cutting process, and created a dataset with more than 9000 images. Using a CNN, they developed a predictive model for tool wear and tool damage. The model automatically assesses the condition of cutting tools (ranging from no wear to high wear) using thermal imaging data, with a classification accuracy of 99.55%. While CNNs have made advancements in the prediction of tool wear, existing models mainly focus on the spatial correlations of machining signals, often overlooking the inherent temporal associations and dynamic features within the signals. This bias leads to current CNN models' difficulty in effectively addressing long-term dependency issues, which are crucial for analyzing physical quantities in tool operations as these quantities are information-rich along the temporal dimension.

The core characteristic of RNNs lies in their hidden layers, which allow the model to consider the temporal order and dependencies between current input and historical information when processing data [15]. However, the main challenge that RNNs are prone to is gradient explosion, especially when there are more hidden layers. To address this issue, researchers have proposed variants such as long short-term memory networks (LSTMs) and gated recurrent units (GRUs). Shah et al. [16] used sensors to capture acoustic emission and vibration signals, creating scaleograms with Morlet wavelets. They utilized the relative wavelet energy criterion to choose appropriate wavelet functions and employed SinGAN to produce extra scaleograms. Subsequently, they extracted several image quality parameters to build feature vectors, which were fed into a stacked LSTM model, achieving outstanding performance indicators. Li et al. [17] used radar charts to integrate multi-source signal features and combined them with AdaBoost and Stacked BiLSTM for accurate tool wear prediction. Mahmood et al. [18] employed the singular spectrum analysis algorithm to denoise and extract features from original force signal data. Utilizing principal component analysis techniques to reduce data dimensionality and one-hot encoding to transform the model's target variables from text to binary numerical format, they inputted these data into a BLSTM model, which successfully recognized the state of the tools. Marani et al. [19] proposed a predictive model based on LSTM for predicting tool flank wear in the machining process of steel alloys. They tested the LSTM model using the spindle motor current signals gathered during experiments performed on a lathe. Bilgili et al. [20] developed a neural network based on LSTM architecture to predict tool flank wear using measured spindle motor current and dynamometer signals. Although RNNs excel in processing time-series data, particularly in capturing temporal correlations and long-term dependencies, they still encounter limitations in the field of tool wear prediction. Existing RNN models typically process sequence data at fixed time steps, a structure that restricts RNNs from naturally adapting to and capturing the multi-scale features present in sequence data. This limitation

inevitably leads to a significant loss of valuable information and results in an incomplete representation of features.

Although all of these methods employ variants of CNN or RNN and combine them with other unique processing to obtain good results, they may still not be able to fully learn all the relevant information as they are mainly based on a single network structure, which may limit the performance of the models. Furthermore, the stability of these models in harsh environments remains to be further verified. Therefore, deeper exploration into network structures is crucial.

In recent years, the combination of CNNs and RNNs has been widely explored in various fields, as this combination is able to take advantage of the unique strengths of both networks in feature extraction. Marei et al. [21] developed a hybrid CNN-LSTM model that incorporates a transfer learning mechanism, using multimodal data from cutting tools. They employed a pre-trained ResNet-18 CNN model to extract features from visual inspection images of the cutting tools. They implemented transfer learning based on maximum mean discrepancy to adapt the trained model specifically for cutting tools. Zhou et al. [22] used a GRU to capture the temporal dependence in the tool cutting signal and then used CNN to extract multidimensional features, which were mapped to tool wear values by linear regression. Si et al. [23] proposed BiLPReS, a novel predictive model that utilizes a hybrid architecture integrating LSTM, an encoder actuator, and residual skip connections. Compared with CNNs and RNNs, this model achieves global perception of long-range dependencies and parallel computation. An et al. [24] first extracted local features using CNN and reduced the dimensionality, then stacked BiLSTM with LSTM for denoising and coding, followed by multiple fully connected layers and regression layers to predict the remaining useful life of the tool. Bazi et al. [25] decomposed signals into a sub-time-series known as intrinsic mode functions through variational mode decomposition. Using these intrinsic mode functions as inputs, they successfully achieved relatively accurate tool wear predictions by employing a combination of CNN and BiLSTM. While the fusion of CNNs and RNNs in the field of tool cutting wear prediction has shown significant effectiveness, the current mainstream architecture of serial models exhibits clear limitations. Specifically, the architecture where the output of one model serves sequentially as the input for the subsequent model leads to a key issue: errors at each stage may be cumulatively amplified in subsequent stages, thereby affecting the accuracy of the final output. Moreover, this serial dependency nature restricts the model's parallel processing capabilities, further reducing computational efficiency.

To address the above issues, this paper introduces a novel approach named parallel convolutional and recurrent neural networks with attention-modulated residual learning (ParaCRN-AMResNet). The framework adopts a parallel structure that integrates multi-scale dilated CNN modules with BiGRU modules. In addition, residual blocks with an attention mechanism are introduced to compensate for uncaptured critical information, using standard residual blocks to accelerate convergence and stabilize the computation. Global average pooling (GAP) is employed to identify and retain the most representative local spatial features while also reducing the spatial dimensions of the feature mappings. The selected prominent features are fused through a fully connected layer, outputting the predicted tool wear amount.

The main contributions are as follows:

1. A new tool wear prediction method has been proposed that completes wear prediction through an end-to-end mechanism, significantly surpassing traditional sequential deep learning models, especially in terms of processing speed improvement.
2. A parallel architecture is adopted, enabling independent feature capture among CNN modules, residual blocks, and RNN modules, which significantly enhances the model's computational efficiency and accuracy and reduces error accumulation.
3. Different sizes of dilated convolution structures and BiGRU structures have been designed to capture feature information across various time dimensions, effectively solving the time-dependency issues found in traditional models.

4. Effective attention units have been integrated, with SimAM emphasizing and highlighting key features, while ResNeSt compensates for potentially uncaptured critical information, further enhancing prediction accuracy and the model's noise resistance capability.

The remainder of this paper is structured as follows. Section 2 thoroughly discusses the detailed structure of the functional modules and overall framework of the proposed deep learning model. Section 3 details the construction of the experimental rig and delves into an in-depth analysis of the experimental results on tool wear prediction, thereby confirming the efficacy and noise resistance capability of the proposed ParaCRN-AMResNet model. Section 4 presents some important conclusions of the paper.

2. Proposed Methodological Framework

This section primarily introduces the methodological principles employed by the ParaCRN-AMResNet model. It combines the advantages of dilated CNN and BiGRU models in a parallel structure. The model conducts in-depth spatio-temporal feature extraction through the SimAM module and a series of dilated CNN layers, utilizing ResNeSt for feature refinement and focus. Meanwhile, a Seq2Seq-structured BiGRU processes sequential data, capturing temporal features across different scales. The two streams are then merged, with final predictions being performed through a fully connected layer.

2.1. Dilated CNN

Dilated CNN has been recognized as a significant technological advancement in the field of deep learning. Its advantage lies in systematically enlarging the receptive field of the convolutional kernel without adding any extra parameters [26,27]. This adjustment allows for a deeper and broader exploration of the input features, thereby improving the model's capacity for processing time-series data. Focusing on the architecture of the dilated CNN, it is characterized by its ability to modify the layout of the convolutional kernel to enhance functionality. This design not only enhances the model's ability to capture long-term dependencies but also maintains computational efficiency.

In terms of technical details, the key difference between dilated CNN and traditional CNN is the specific interval arrangement of elements within the convolutional kernel, with the scale of this interval being defined by the dilation rate. For the dilated CNN of 1D data, the specific layout can be exhaustively described by the following equation:

$$y[i] = \sum_{j=1}^{M} x[i - j \cdot d] \cdot k[j] \tag{1}$$

where $y[i]$ is the output feature at position i; x is the input feature; k is the weights in the convolution kernel; j is traversing all positions of the convolution kernel; i is the current data point position; d is the dilation rate, defining the interval between weights in the convolution kernel; and M is the length of the convolution kernel. To further elucidate, Figure 1 provides an intuitive illustration. In this figure, Figure 1a depicts a standard convolution kernel covering a 5×5 feature range; Figure 1b shows a convolution kernel with a dilation rate of 2, where each pair of adjacent elements has a clear gap, thus expanding its receptive field to 9×9; Figure 1c presents a convolution kernel with a dilation rate of 4, where there are 3 gaps between each pair of adjacent elements, leading to a further increase in the receptive field to 17×17.

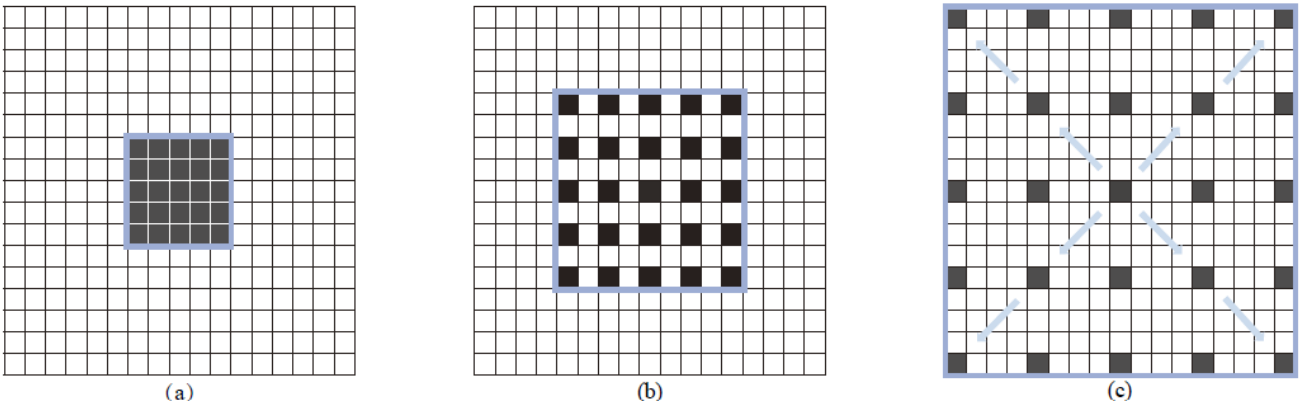

Figure 1. Dilated convolution visualization: (**a**) dilation rate = 1, (**b**) dilation rate = 2, (**c**) dilation rate = 4.

2.2. Global Average Pooling

GAP is a variant of conventional pooling, and its general structure can be seen in Figure 2 [28]. It is positioned at the end of the CNN, following the last convolutional layer. Unlike the traditional flatten layer, the introduction of the GAP layer aims to effectively reduce the number of parameters in the model, mitigate overfitting, and enhance the model's global understanding of spatial features in the input data. For time-series data, GAP1D is commonly used, where the average is calculated as follows:

$$GAP(F) = \frac{1}{L}\sum_{i=1}^{L} f_i \tag{2}$$

where F is the feature vector; L is the length of the feature vector; and f_i is the i-th feature value in the vector. Specifically, the GAP layer performs a global average pooling operation on the feature maps output by the last convolutional layer, transforming each feature map into a single numerical value. This not only simplifies the subsequent processing steps but also retains essential spatial information within the feature maps. Hence, the GAP layer serves as a key transition point within the model's structure and acts as a bridge between feature extraction and classification decision making.

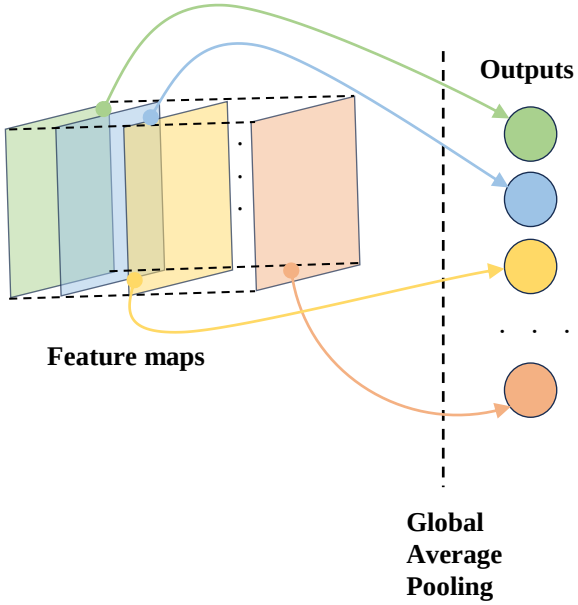

Figure 2. The structure of GAP.

2.3. Bidirectional Gated Recurrent Unit

GRUs [29] are the same as the LSTM network, which was proposed to address the issues of long-term memory and gradient problems in traditional RNN networks during backpropagation. These units are designed to process sequential data, particularly in contexts requiring the capture of long-term dependencies. GRU controls the flow of information by introducing a gating mechanism, with its structure being shown in Figure 3. For a sequence $x = (x_1, x_2, x_3, \ldots, x_t)$, where x_t is the input at time step t, the update gate z_t determines the extent to which the hidden state from the previous time step h_{t-1} is retained in the current time step:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{3}$$

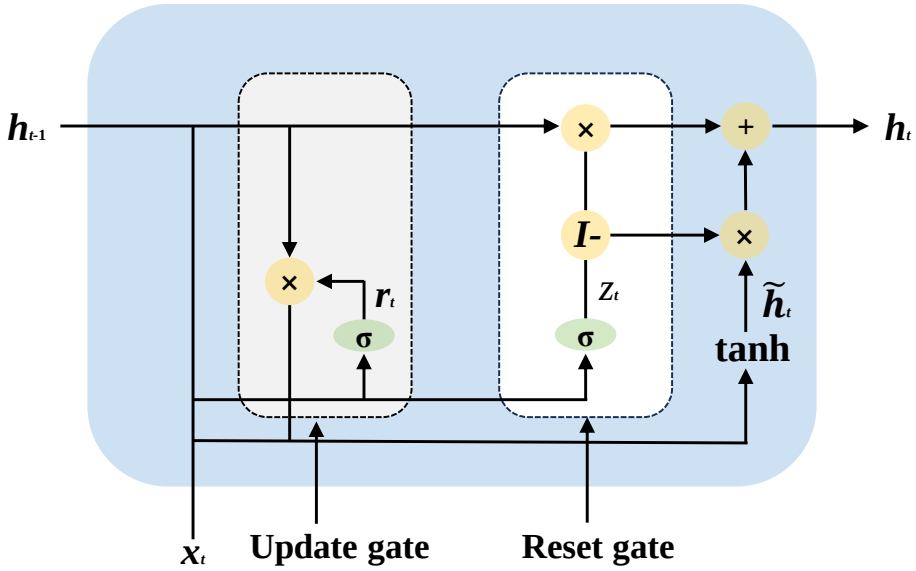

Figure 3. The structure of a GRU.

The reset gate r_t controls the influence of the previous time step's hidden state h_{t-1} on calculating the candidate hidden state $\widetilde{h}_{t-1}$ at the current time step:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{4}$$

The candidate hidden state $\widetilde{h}_t$ is calculated based on the reset previous hidden state and the current input, providing a candidate value for the new hidden state:

$$\widetilde{h}_t = tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b) \tag{5}$$

The final hidden state h_t is determined through interaction with the update gate, which dictates the proportion of the previous hidden state to be retained and the extent of the new candidate hidden state to be incorporated:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \tag{6}$$

where σ is the sigmoid activation function, used to control the flow of information; *tanh* is the hyperbolic tangent activation function; $\odot$ is the Hadamard product; W_z, W_r, and W are the weight matrices for the respective gates; and b_z, b_r, and b are the bias vectors.

BiGRU deploys two independent GRUs at each time point, one processing the forward flow of the sequence and the other handling the backward flow, enabling it to encode both forward and backward information of a sequence simultaneously. This bidirectional structure allows the model to understand the data from two directions, providing a more comprehensive analysis of the sequence. Its structure is shown in Figure 4. The hidden

states of both forward and backward directions can be calculated by the above Equations (3)–(6). The overall hidden state at time point t is given by:

$$h_t^{BiGRU} = \left[h_t^{forward}, h_t^{backward}\right] \tag{7}$$

where $h_t^{forward}$ is the forward hidden state at time t; $h_t^{backward}$ is the hidden state at moment t in the reverse direction.

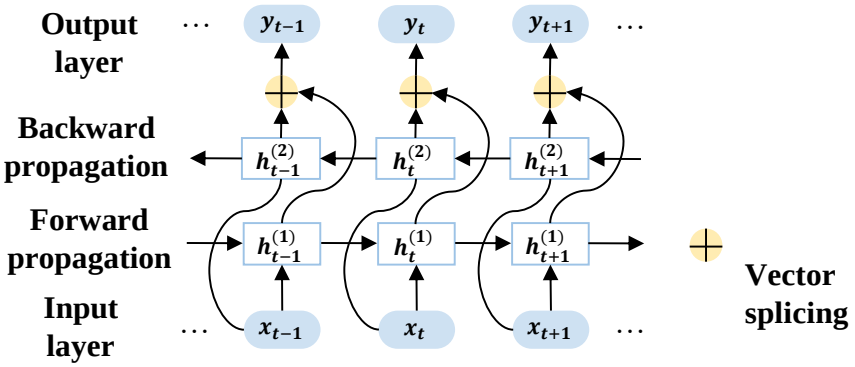

Figure 4. The structure of a BiGRU.

Compared with the network structure of LSTM, GRU only contains reset and update gates, while LSTM has a forget gate, an input gate, and an output gate. This means that GRU has fewer model parameters and a more streamlined network structure under the premise of the same number of hidden units. Based on these advantages, this paper selects BiGRU and designs it as a Seq2Seq structure, which can learn directly from the source sequence to the target sequence without the need for manually designing complex features. Additionally, the model can remember and utilize long-distance dependency information in the input sequence.

2.4. Residual Network

The cornerstone of ResNet [30] is the residual block, which brings a structural contribution by introducing a direct channel for information flow in deep networks. This design allows the network's original input to be directly transmitted to subsequent layers through shortcut connections, thereby effectively facilitating the backpropagation of gradients. In deep CNNs, the training process is prone to gradient vanishing or exploding as the network depth increases. The introduction of residual blocks ensures that gradients can be transmitted without obstacles, even in very deep networks, significantly enhancing the model's training efficiency and stability. The structural design of the residual block allows each block to directly utilize information from previous layers, not solely relying on the outcomes of the current layer's processing. Specifically, a residual block can be described as follows:

$$H(x) = F(x) + x \tag{8}$$

where $H(x)$ is the final output mapping; $F(x)$ is the residual mapping; and x is the identity mapping.

In this paper, ResNeSt as an improved version of the residual block is also used, with its structure being shown in Figure 5a [31]. It is used in parallel with a dilated convolutional neural network, a structural arrangement that enables the ResNeSt block to more effectively supplement feature information that the main network might miss and helps to avoid the problem of gradient explosion. Within it, the input features are first divided into multiple cardinals, and each cardinal is further divided into radix groups; the convolution and split-attention operations are performed separately, and then all outputs are concatenated with another layer of convolutional outputs. The core of this process is called split-attention, whose internal structure is shown in Figure 5b. Given a set of input tensors [X_1, X_2, …,

X_r] with the dimensions [$h \times w \times c$], a global average pooling operation is applied to each tensor X_i, which is:

$$P_i = GP(X_i) \tag{9}$$

where $i = 1, 2, \ldots, r$; subsequently, each P_i undergoes a fully connected operation followed by a batch normalization (BN) operation, and is then activated:

$$D_i = ReLU(BN(Dense(P_i))) \tag{10}$$

then, for each output D_i, three different fully connected operations are applied to generate a set of attention weights:

$$A_{ij} = Dense(D_i) \tag{11}$$

where $j = 1, 2, 3$; then, an r-Softmax operation is used to normalize the weights to obtain S_{ij}. S_{ij} is used to weigh the input X_i, and all results are summed to obtain the final output, as shown in (12) and (13).

$$S_{ij} = rSoftmax(A_{ij}) \tag{12}$$

$$Y = \sum_{i=1}^{r}(S_{i1} \times X_i + S_{i2} \times X_i + S_{i3} \times X_i) \tag{13}$$

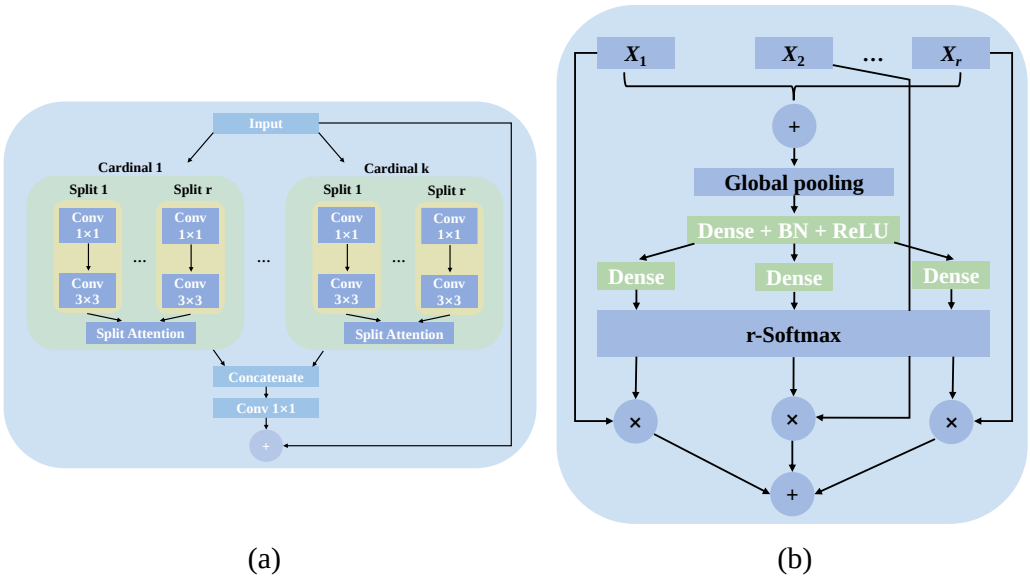

Figure 5. Schematic diagram of ResNeSt and its internal attention structure: (**a**) Structure of ResNeSt; (**b**) structure of split-attention.

The effectiveness of residual networks in various computer vision tasks has been widely demonstrated.

2.5. Attention Mechanisms for Convolutional Part

Attention mechanisms as an emerging layer have been proven to enable models to focus more on the relevant parts of a task, thus achieving significant results in improving model performance. To avoid introducing excessive computational overhead while ensuring performance enhancement, this paper adopts the SimAM attention mechanism proposed by Yang et al. [32]. Its purpose is to emphasize important features closely related to the task and suppress redundant features within the convolutional module. In contrast to the existing attention modules, which are mainly based on channels and spatial dimensions, the SimAM mechanism focuses on adjusting the weights of the feature maps to enhance feature discriminability and does not add extra parameters to the original network. Its specific structure is shown in Figure 6.

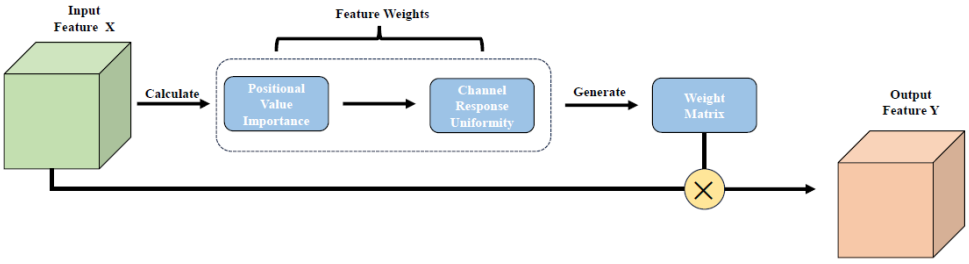

Figure 6. The structure of SimAM attention.

The SimAM attention mechanism is based on neuroscience theory, and its core is the 'energy function', which is expressed as:

$$e_t(w_t; b_t; y; x_i) = \frac{1}{M-1}\sum_{i=1}^{M-1}(-1-(w_t x_i + b_t))^2 + (1-(w_t + b_t))^2 + \lambda w_t^2 \tag{14}$$

$$w_t = \frac{-2(t-\mu_t)}{(t-\mu_t)^2 + 2\sigma_t^2 + 2\lambda} \tag{15}$$

$$b_t = -\frac{1}{2}(t+\mu_t)w_t \tag{16}$$

where e_t is the energy; w_t and b_t are the weighting and bias transformations; t and x_i represent the target neuron and other neurons in a single channel of the input features, respectively; i is the index in the spatial dimension; M is the number of neurons in that channel; λ is a normalization parameter; and μ_t and σ^2 are the mean and variance computed for all neurons in the channel excluding t. This function quickly calculates the energy of each neuron, thereby determining its importance. Because SimAM is designed with reference to the attention mechanism of mammals, it employs a scaling operator to represent the brain's gain effect $\widetilde{X}$ on neuronal responses, expressed as:

$$\widetilde{X} = sigmoid\left(\frac{1}{E}\right) \odot X \tag{17}$$

where E groups all minimum energy differences across channels and spatial dimensions, and then a sigmoid function is used to limit excessively large values in E. $\odot$ is the Hadamard product. It is used to emphasize task-relevant important features and suppress redundant features in the convolutional module.

2.6. Parallel Modelling Structure

As previously mentioned, CNNs and RNNs each have their unique advantages, but different types of neural networks also have their limitations. A single module may struggle to capture all information, limiting its effectiveness in complex applications. Therefore, hybrid models that combine multiple network architectures are particularly crucial. These models often integrate the advantages of various structures and can even, to some extent, compensate for certain deficiencies.

The structure widely used by current researchers is the sequential stacking of networks; two sequential stacking structures are shown in Figure 7. Different types of networks are sequentially organized according to their feature extraction capabilities. Although this structure has been proven to be effective to a certain extent, there are still challenges with this sequential structure in series. Due to the inherent sequential dependency, the performance of subsequent networks is largely limited by the feature extraction effectiveness of preceding networks. This effect, known as 'error accumulation', can amplify minor errors from one module to the next, significantly reducing overall performance. Furthermore, the computational efficiency of the sequential structure model is inherently constrained by the order of computation.

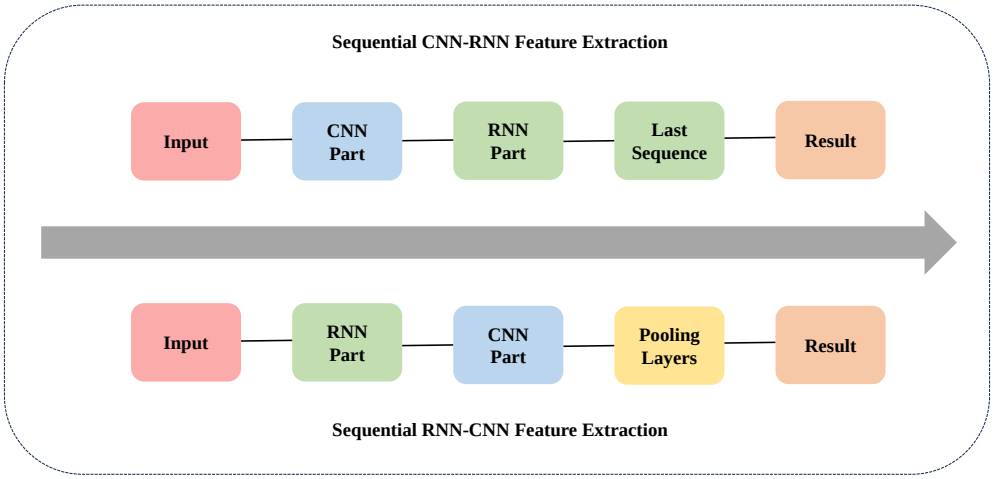

Figure 7. Common stacking order.

In response to these challenges, this study proposes a novel parallel network structure as shown in Figure 8. The structure adopts a multi-input strategy, where the original features are input into the CNN part and the RNN part for independent feature learning, respectively, after undergoing a simple pre-processing. Throughout the supervised model training process, each input stream remains independent, avoiding mutual interference. After a series of operations, all extracted features are eventually concatenated into a 1D vector, preparing the ground for subsequent tool wear analysis.

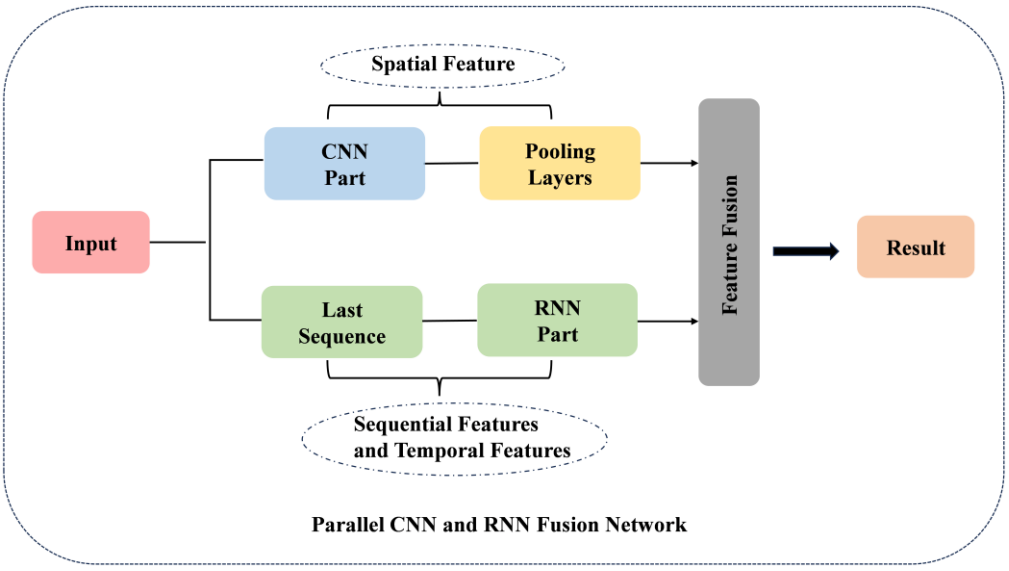

Figure 8. Parallel network structure.

2.7. Proposed Model Structure

This paper proposes a novel model, ParaCRN-AMResNet, depicted in Figure 9. The model consists of three main parts: the CNN part, the RNN part, and a fully connected layer. The specific implementation process is as follows:

(1) The model employs wavelet transformation and concatenation to preprocess the signal, ensuring a comprehensive multi-scale feature input.
(2) It uses dilated CNN and ResNeSt blocks in a parallel layout to extract diverse scale features without cross-interference. The integration of the SimAM attention mechanism selectively focuses on crucial features, streamlining the feature set.

(3) A Seq2Seq BiGRU module is in parallel, aligned with the CNN layers. This configuration efficiently captures temporal features, with a varying number of units in hidden layers to address different time scales.

(4) The outputs from the CNN and RNN segments are combined, which is followed by a fully connected layer, to accurately predict tool wear.

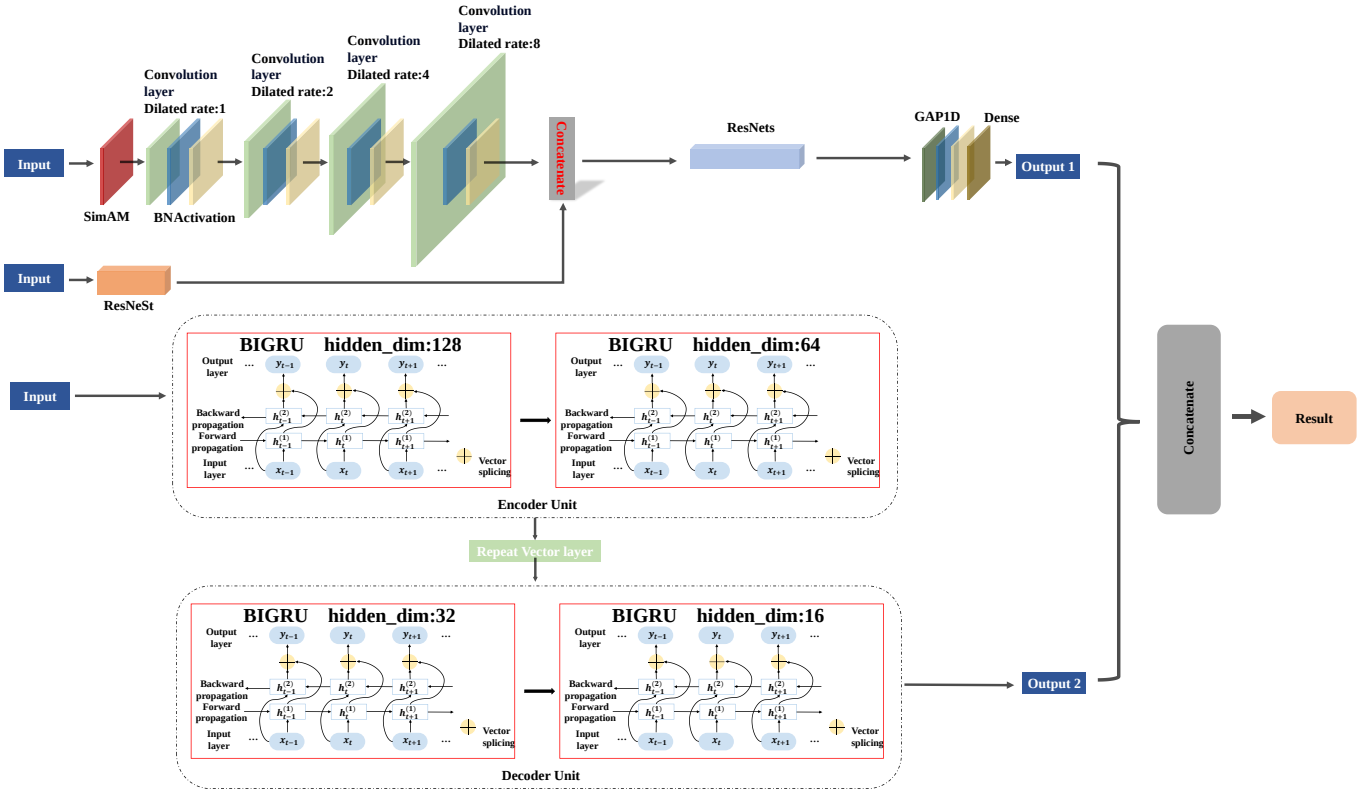

Figure 9. The detailed schematic diagram of ParaCRN-AMResNet.

The model's parallel computing architecture allows for independent and effective feature extraction, merging the benefits of dilated CNN and BiGRU. The SimAM attention mechanism enhances focus on task-relevant features, improving the model's sensitivity and precision. The incorporation of ResNeSt blocks supplements the model, ensuring no critical feature is overlooked. An increasing dilation rate in the dilated CNN captures detailed features, ranging from localized to broader contextual information. Concurrently, the BiGRU module, with its hidden layers gradually decreasing in size, adeptly captures temporal information across varying scales, enhancing the model's ability to process complex time-series data.

In summary, the ParaCRN-AMResNet model encapsulates a blend of innovative techniques and structures, enhancing its performance in tool wear prediction.

3. Experiment

In this section, the primary focus encompasses the experimental conditions related to tool wear data, the selection of parameters for the ParaCRN-AMResNet model, the assessment of model noise resistance capability, and the ultimate prediction results.

3.1. Experiment Setup

Experiments are conducted using the IEEE PHM Challenge 2010 dataset [33] to validate the performance of the proposed ParaCRN-AMResNet model. The specific layout of the experimental setup is shown in Figure 10. The experiment utilized a Röders Tech RFM 760 CNC machine tool, and the selected tool was a three-flute carbide ball-end mill with a cutting length of 108 mm. The workpiece material was stainless steel, and the relevant

cutting parameters are listed in Table 1. In order to measure the cutting forces, a three-directional piezoelectric dynamometer from Kistler was installed between the machine tool and the workpiece. At the same time, three Kistler accelerometers were mounted on the workpiece to measure vibration signals, and an acoustic emission (AE) sensor was used to capture elastic waves generated by stress changes. All of these sensor signals were amplified and collected through a Kistler 5019A multi-channel charge amplifier and DAQ Ni PCI1200 data acquisition card, with a sampling rate set at 50 kHz. After each cutting operation, the wear of the mill's flank face was measured using a Leica MZ12 microscope, and this measurement served as the target value for each sample. The IEEE PHM Challenge 2010 dataset consists of six subsets (C1 to C6), and each subset contains data from 7 different sensor signals. Among these, subsets C1, C4, and C6 additionally include corresponding tool wear measurements, while subsets C2, C3, and C5 do not contain such data. Based on these considerations, this study selected subsets C1 and C6, which contain tool wear data, as the training set and used the C4 subset as the test set for subsequent model validation.

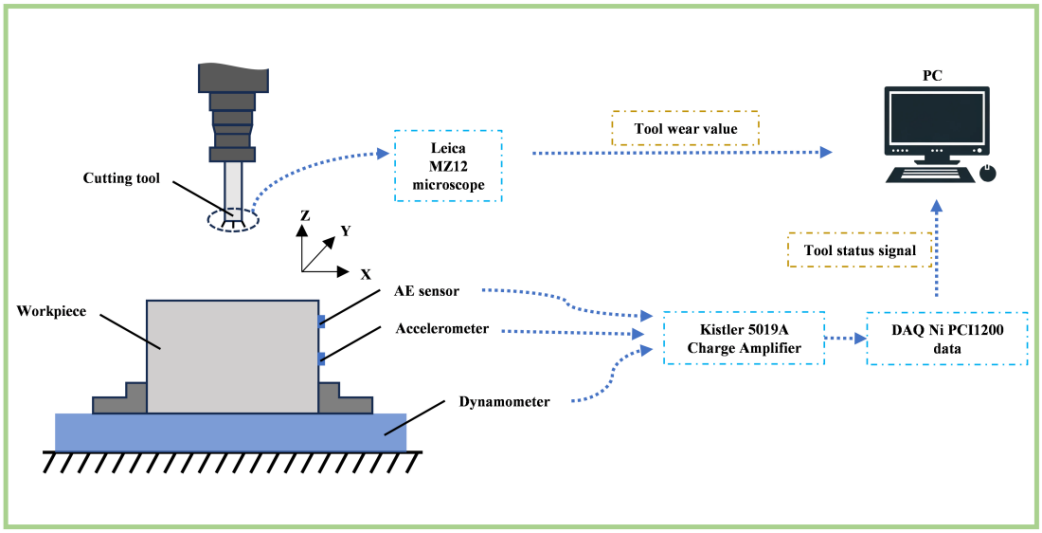

Figure 10. Experimental setup.

Table 1. Cutting parameters.

Property	Values
Y-depth of cut	0.125 mm
Z-depth of cut	0.2 mm
Feed rate	1555 mm/min

3.2. ParaCRN-AMResNet Traning and Testing Procedure

To build and assess the performance of the training and testing models, the data used are an already labeled dataset. During the training period for supervised model hyperparameter optimization, considering that the task is tool wear prediction, the loss function was set as the mean squared error (MSE), which is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - y_{pred_i}\right)^2 \tag{18}$$

where n is the number of samples; y_i is the actual value of the i sample; and y_{pred_i} is the predicted value for the i sample.

Adaptive moment estimation (Adam) was chosen as the optimizer, known for accelerating gradient descent, thereby enabling efficient and robust training acceleration [34]. The activation function selected was Swish [35], which is expressed as follows:

$$Swish(x) = x \cdot \sigma(\beta x) \tag{19}$$

where x is the input, σ is the sigmoid function, and β is an adjustable parameter, which for the sake of simplifying calculations was set to 1 in this paper. Compared with other activation functions, the Swish function is smoother and can effectively assist the optimizer in updating weights. The initial learning rate was set to 0.0005, the batch size was chosen as 16, the number of training epochs was fixed at 100, and dropout was set at 0.4. The model construction, training, and testing were all implemented using Python 3.10.12 and Keras 2.11.0, with the Keras backend being Tensorflow-gpu 2.11.0, on an Intel(R) Xeon(R) Gold 5318Y CPU @ 2.10 GHz processor and NVIDIA A100 PCIe 40 GB graphics card. The server's operating system was Ubuntu 20.04.

3.3. Data Preprocessing

In the C1 and C6 datasets, 80% of the data were used for the training set, while the remaining 20% served as the validation set. Due to the sampling frequency set at 50 kHz, a large amount of data were generated in each cutting process, significantly increasing the time of model training. To alleviate this issue, a downsampling method was employed to extract 5000 equidistant data points from each signal for sequential concatenation. Additionally, signal processing utilized the Daubechies wavelet (db4) and a 2-level decomposition wavelet transform, enabling the model to capture both the general trends and the detailed information within the signal. Furthermore, the flank face wear values of the milling tool, measured after actual cutting operations, were used as sample labels to construct the tool wear dataset.

3.4. Evaluation Criteria

Four regression metrics were selected for the quantitative evaluation of the model's predictive performance: mean absolute error (MAE), MSE, mean absolute percentage error (MAPE), and coefficient of determination r-squared (R^2). The standard formulas for MSE, MAPE, and R^2 are as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_{pred_i}\right| \tag{20}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - y_{pred_i}}{y_i}\right| \tag{21}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(y_i - y_{pred_i}\right)^2}{\sum_{i=1}^{n}\left(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\right)^2} \tag{22}$$

The following experiments were repeated three times, and the final metrics are the average values of these three trials.

3.5. Hyperparameter Optimization

For the proposed ParaCRN-AMResNet network, its performance is primarily influenced by five key hyperparameters: (1) the dilation rate D_r in dilated convolution; (2) cardinality in ResNeSt, represented by N_c, which specifies the number of feature groups, with the radix set to 1 considering the volume of training data and computational efficiency; (3) the number of ResNeSt blocks N_s; (4) the number of ResNet blocks N_r; and (5) the number of units in the BiGRU hidden layer, N_h. The dilation rate directly affects the ability of the convolution module to perceive and capture temporal information, while the middle three parameters are directly linked to the model's computational efficiency and representational and noise resistance capability; the number of units in the BiGRU layer influences the model's information storage capacity. To thoroughly assess the impact of these five hyperparameters, experiments were conducted.

3.5.1. Selection of Dilation Rate

In dilated CNN, the dilation rate is used to increase the receptive field of the convolution operation, which means it directly impacts the convolution layer's ability to capture information in the data, thus affecting the model's perception of the entire dataset. How to balance the richness and scale of the features captured by the convolutional layer was the focus of this experiment. The dilation rates for the four convolution layers were limited to three options: [1, 1, 1, 1], [1, 2, 4, 8], and [1, 3, 6, 9]. Additionally, the cardinality N_c in ResNeSt was tentatively set to 3 and the number of stacking layers N_s to 10; the number of ResNet stacking layers N_r was tentatively set to 4, and the number of units in the BiGRU hidden layer N_h was set to 128. Other parameter settings can be found in Table 2.

Table 2. The provided ParaCRN-AMResNet model parameters.

Item	Value/List
Epoch	100
Batch size	16
Learning rate	0.0005
Optimization function	Adam
Activation function	Swish
Length of the time-series	5000
Number of the time-series	945
Filter size	5
Filter number	128
Dilation rate	[1, 1, 1, 1], [1, 2, 4, 8], [1, 3, 6, 9]
Neuron number in BiGRU unit	[64, 128, 256, 512]
Stacking number of ResNets unit	[2, 4, 6, 8]
Stacking number of ResNeSt unit	[8, 10, 12, 14]
The number of cardinalities	[2, 3, 7]
The number of neurons in the dense layer	128
Dropout rate in dilated CNN	0.4
Method of feature fusion	Concatenate
Method of padding	Same

The experimental results are shown in Table 3. It showcases the model's response to different dilation rate configurations, with the [1, 2, 4, 8] setting demonstrating a notable enhancement in predictive performance. This configuration yields the most favorable outcomes in terms of MAE, MSE, MAPE, and R^2 metrics.

Table 3. Experimental results of model performance under different D_r.

D_r	MAE	MSE	R^2	MAPE
[1, 1, 1, 1]	3.8684	29.18	0.9801	4.0195%
[1, 2, 4, 8]	2.6015	15.1921	0.9897	2.7997%
[1, 3, 6, 9]	4.4606	41.5637	0.9717	4.7167%

These metrics indicate the smallest discrepancy between the predicted values and the actual values, suggesting that the model performs best under this dilation rate configuration. This also highlights the importance of appropriately selecting dilation rates for optimizing the performance of CNNs. Specifically, the configuration of dilation rates [1, 2, 4, 8] most effectively balances the scope of feature capture and the preservation of detail, ensuring that the model can comprehensively and accurately process the critical information within the data. Consequently, [1, 2, 4, 8] will be chosen as the dilation rates for the convolutional layers in subsequent experiments.

3.5.2. Selection of Cardinality

In the ResNeSt architecture, cardinality is used to specify how many different groups to split the feature channel into. Increasing the number of cardinalities allows each group to

specialize in learning specific features or attributes in the input data, enabling the model to capture the diversity and complexity of the data more finely. However, a higher cardinality value also leads to a linear increase in computational cost and may even result in decreased performance due to increased network complexity. Therefore, it is necessary to find a balance between computational cost and model performance. Given that the data have seven feature channels, the range for cardinality N_c was set between [2, 3, 7]. Additionally, the number of stacking layers N_s in ResNeSt was set to 10, the number of stacking layers N_r in ResNets to 4, and the number of units in the BiGRU hidden layer N_h to 128. Other parameter settings can be found in Table 2. The final experimental results are shown in Table 4.

Table 4. Experimental results of model performance under different N_c.

N_c	MAE	MSE	R^2	MAPE
2	3.8600	29.3686	0.9794	4.0383%
3	2.6015	15.1921	0.9897	2.7997%
7	4.6292	41.9538	0.9715	4.6604%

As shown in Table 4, when N_c is set to 3, the model achieves the lowest values in MAE, MSE, and MAPE and the highest in R^2. When N_c = 7, the model's performance is slightly inferior to the setting of N_c = 2. It is noteworthy that the computation time per epoch for N_c = 7 is 108 s, which represents a significant increase in computational cost compared with 55 s for N_c = 3 and 50 s for N_c = 2.

The results indicate that while increasing cardinality can enhance the model's ability to capture the diversity and complexity of input data, a higher cardinality value also linearly increases computational costs and may even lead to performance degradation due to increased network complexity. Therefore, a balance must be struck between computational cost and model performance. Considering both model performance and computational efficiency, setting the cardinality N_c to 3 is identified as the most appropriate choice.

3.5.3. Selection of ResNeSt Stacking Number

The number of stacking layers N_s in ResNeSt not only affects the model's ability to supplement and extract temporal features but also directly relates to computational costs and the risk of overfitting. To determine the optimal value for N_s, a series of experiments are conducted. Based on preliminary results, the range of values for this parameter is limited to [8, 10, 12, 14]. Temporarily, the number of stacking layers N_r in ResNets is set to 4, and the number of units in the BiGRU hidden layer N_h is set to 128. Other parameter settings can be found in Table 2. The final experimental results are shown in Table 5.

Table 5. Experimental results of model performance under different N_s.

N_s	MAE	MSE	R^2	MAPE
8	3.3891	23.5464	0.9840	3.7081%
10	2.6015	15.1921	0.9897	2.7997%
12	3.5141	23.6523	0.9839	3.7366%
14	4.8587	43.2362	0.9706	4.8708%

According to the data in Table 5. When the number of stacked layers N_s in the ResNeSt architecture is set to 10, the model exhibits its best predictive performance. The specific performance metrics are as follows: MAE: 2.6015, MSE: 15.1921, R^2:0.9897, MAPE: 2.7997%.

The experimental results underscore the critical importance of optimizing N_s in the design process of this model. Specifically, the model needs to strike a balance between enhancing the capability to extract temporal features and controlling computational resource consumption to avoid overfitting. Among all tested configurations, setting N_s to 10 not only significantly optimized the model's prediction accuracy but also demonstrated an

effective compromise between increasing the depth of the model structure and maintaining computational efficiency. Therefore, 10 is determined as the best choice for N_s to be used in subsequent experiments.

3.5.4. Selection of ResNets Stacking Number

The number of stacking layers N_r in ResNets and N_s in ResNeSt similarly impact the model, primarily with respect to the learning of model feature hierarchies and the efficiency of model convergence. Based on the data from preliminary experiments, the range for N_r is set to [2, 4, 6, 8]. Similarly, the number of units in the BiGRU hidden layer N_h is set to 128. The settings for the other parameters can be found in Table 2.

As shown in Table 6, increasing the number of N_r does not lead to an improvement in model performance; instead, a decrease in performance is observed. The model exhibits its best performance when N_r is set to 4.

Table 6. Experimental results of model performance under different N_r.

N_r	MAE	MSE	R^2	MAPE
2	3.6867	35.3543	0.9745	3.5255%
4	2.6015	15.1921	0.9897	2.7997%
6	3.6133	31.6816	0.9785	3.7820%
8	5.0007	47.1499	0.9679	4.7343%

The phenomenon shown in the Table 6 can likely be attributed to the increased complexity of the model, which becomes a burden in the presence of an insufficient number of training samples, leading to overfitting and a reduction in generalization ability. Therefore, finding a balance between model complexity and performance is particularly crucial. When N_r = 4, all performance indicators are at their best, making this setting the optimal choice for subsequent experiments.

3.5.5. Selection of BiGRU Hidden Layers Number

The model with a higher number of hidden layers N_h in BiGRU shows better performance in three aspects: temporal and sequential feature extraction, information storage, and model capacity. However, a higher number of N_h also affects the training speed and practicality of deploying the model, making it more difficult to interpret. Therefore, an appropriate value for N_h must be chosen based on experimental results. Based on experience, the range for N_h is set to [64, 128, 256, 512]. Other parameters of the model are in accordance with Table 2. The specific experimental results are presented in Table 7.

Table 7. Experimental results of model performance under different N_h.

N_h	MAE	MSE	R^2	MAPE
64	4.7822	39.4534	0.9732	5.1045%
128	2.6015	15.1921	0.9897	2.7997%
256	5.3561	47.4445	0.9677	5.9055%
512	5.2891	58.2212	0.9604	5.4853%

From Table 7, it can be observed that when N_h = 128, the model demonstrates strong temporal and information storage capabilities, reaching its optimal capacity at this setting. When N_h exceeds this value, MAE, MSE, and MAPE all increase and R^2 decreases.

When N_h = 128, it balances the model's performance with the practicality of its training and deployment, avoiding overfitting issues that could arise from having too many hidden layers. Additionally, this setting ensures that the model has sufficient capacity to effectively process time-series data without sacrificing operational efficiency. Further considering the runtime of the model for each epoch, N_h = 128 is ultimately selected as the optimal number of hidden layers in BiGRU for use in subsequent experiments.

3.6. Ablation Study

To verify the necessity and impact of each component in the proposed ParaCRN-AMResNet model, a series of ablation experiments were conducted.

Initially, to validate the necessity of integrating the CNN and RNN modules, these two modules were individually removed from ParaCRN-AMResNet and tested separately. Then, ResNeSt was removed from the CNN module to evaluate its contribution to supplementing feature information. Subsequently, ResNet was removed from the model to assess its contribution to computational optimization. To further validate the importance of the residual structure in the model, both ResNeSt and ResNet were removed simultaneously, creating the ParaCRN-AM model. Additionally, a model without the attention mechanism was constructed by removing all attention mechanisms from the CNN module to assess their contribution during the convolution process. To ensure the effectiveness of the parallel structure, the CNN and RNN modules were concatenated sequentially, forming two sequential stacking models: CNN-RNN and RNN-CNN. Furthermore, to evaluate the differences in temporal information capture between BiGRU and traditional GRU, BiGRU in the original model was replaced with GRU, named GRU-AMResNet for experimentation. The number of hidden layer units in the RNN module was fixed to create the FixRNN model to assess the effectiveness of the structure designed for capturing temporal features of different scales. All of these experimental models were ensured to have parameters consistent with ParaCRN-AMResNet.

As shown in Table 8, all deep learning network structures demonstrate relatively accurate predictive performance in the task of tool wear prediction. Notably, the proposed ParaCRN-AMResNet model achieves a standout performance with an MAE of 2.6015, MSE of 15.1921, R^2 value of 0.9897, and MAPE of 2.7997%.

Table 8. Summary of the performance of different models.

Model	MAE	MSE	R^2	MAPE
Without RNN	7.8019	113.6746	0.9227	7.9843%
Without CNN	7.3734	88.3356	0.9399	7.3206%
Without ResNeSt	5.3170	53.2287	0.9638	5.1565%
Without ResNet	5.2976	51.2169	0.9652	4.9864%
ParaCRN-AM	7.3900	83.1821	0.9435	7.6285%
Without attention	8.093	117.9771	0.9198	7.7312%
CNN-RNN	9.3849	144.3835	0.9019	8.9123%
RNN-CNN	11.4813	213.6568	0.8548	10.9744%
GRU-AMResNet	5.1335	51.4938	0.9650	4.8982%
FixRNN	7.5067	97.0103	0.9340	7.8128%
ParaCRN-AMResNet	2.6015	15.1921	0.9897	2.7997%

Through the comparative analysis of metrics across distinct models and the model proposed herein, it is evidenced that the incorporation of residual blocks designed to apprehend features that might be omitted by dilated convolution neural network significantly augments the performance of the model. Furthermore, the introduction of the overall residual block structure into the network notably enhances model performance. The adoption of an attention convolution block structure is demonstrably pivotal to the model's success, as evidenced by the inferior performance metrics of the without attention model. The performance of the FixRNN model underscores the importance of capturing temporal features at different scales, aiding the model in understanding multi-scale information from various periods. The overall structure of the model significantly impacts its performance, as seen in the CNN-RNN and RNN-CNN models, which show a marked decrease in performance, even more so than the without attention model, further validating the superiority of the proposed parallel structure.

Additionally, considering safety, Figure 11 shows the comparison of predicted and actual wear values on the cutting blades with the greatest wear for the ParaCRN-AMResNet,

without RNN, without CNN, without attention, and ParaCRN-AM models. Aligned with the metrics presented in Table 8, variations in the accuracy of predictions across these models are evident, with the proposed ParaCRN-AMResNet model achieving the highest proximity to the actual values. These results underscore the model's superior capability in capturing temporal features at varying scales and its efficiently designed parallel structure, positioning the ParaCRN-AMResNet model ahead of other reference models in terms of overall performance.

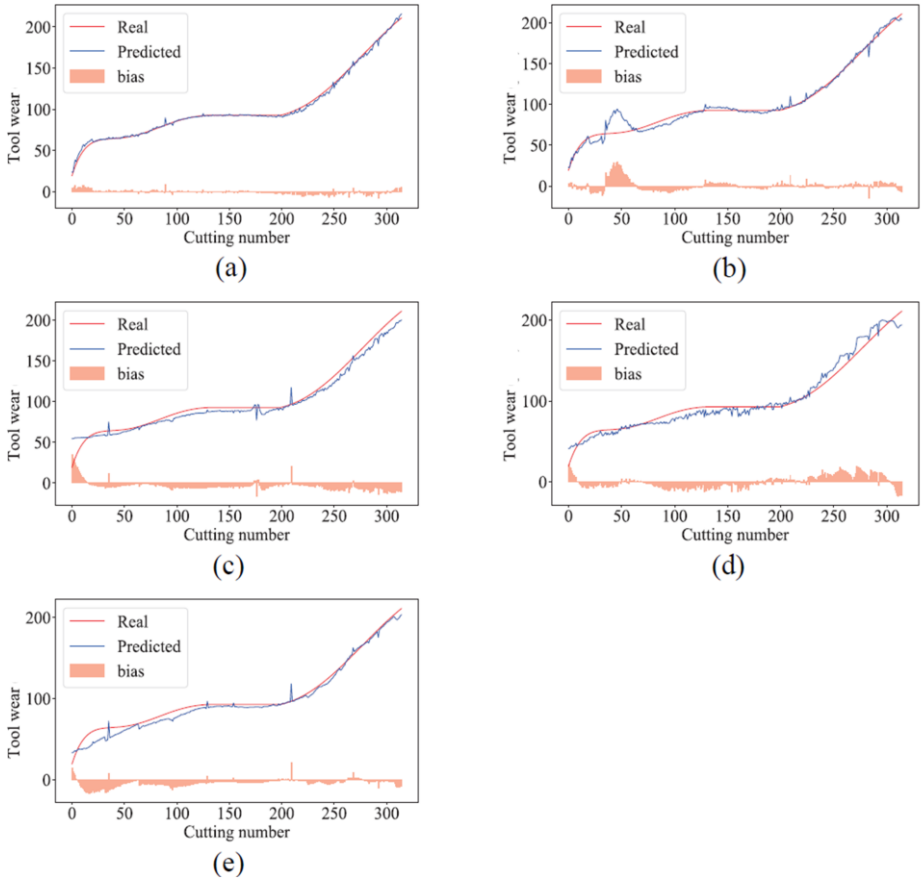

Figure 11. The comparative results of tool wear values: actual measurements versus predictions by various deep learning models: (**a**) the ParaCRN-AMResNet model, (**b**) the without RNN model, (**c**) the without CNN model, (**d**) the without attention model, (**e**) the ParaCRN-AM model.

3.7. Comparative Experiments

To more comprehensively assess the performance of the model, PR-AUC [36], CGRU-IConvGRU-A [37], ConvLSTM-Att [24], and MDMCNN-BiLSTM [38] were used as benchmarks for comparison. The experimental settings follow those described in the original literature, using MAE and R^2 as performance metrics. All results are presented in Table 9.

Table 9. Summary of the performance of different models.

Model	MAE	R^2
PR-AUC	4.6196	0.9718
CGRU-IConvGRU-A	4.4210	0.9660
ConvLSTM-Att	3.9613	0.9723
MDMCNN-BiLSTM	7.5429	0.9347
ParaCRN-AMResNet	2.6015	0.9897

An analysis of the data from Table 9 clearly indicates that the proposed model significantly outperforms the comparison group on key performance metrics. Compared with traditional sequential structure-based models such as PR-AUC, ConvLSTM-Att, and MDMCNN-BiLSTM, which are prone to accumulating errors during data transmission, thereby affecting prediction accuracy, the introduced ParaCRN-AMResNet model employs a parallel architecture design. This design enables independent parallel processing of various features, effectively preventing the common problem of error accumulation associated with sequential processing. Such parallel processing not only significantly enhances computational efficiency but also reduces the decline in predictive accuracy caused by error propagation. Although the CGRU-IConvGRU-A model also utilizes a parallel structure, the sequential arrangement of its internal CNN and GRU components does not fully eliminate inter-module interference.

Furthermore, compared with the 1D CNN used by CGRU-IConvGRU-A, ConvLSTM-Att, and MDMCNN-BiLSTM, the ParaCRN-AMResNet's implementation of DCNN exhibits superior performance in processing time-series data, benefiting from its wider receptive field and deeper feature abstraction capabilities. While the comparison models attempt to capture multi-scale features using convolutional kernels of various sizes, the inherent limitations of their receptive fields render their performance inferior to that of ParaCRN-AMResNet. Although the PR-AUC model employs DCNN to capture time-series features, ParaCRN-AMResNet combines DCNN with a BiGRU structure. This integration allows the model to more effectively capture features across different temporal scales. The introduction of BiGRU enhances the model's ability to capture long-term temporal dependencies, which is challenging to achieve with DCNN alone. Additionally, the multi-dimensional BiGRU in ParaCRN-AMResNet, based on a Seq2Seq structure, contrasts sharply with the fixed-size RNN architectures in other models, enabling the mentioned model to more effectively capture and utilize long-distance dependencies in time-series data.

Unlike the approach of ConvLSTM-Att and MDMCNN-BiLSTM models, which emphasize features at the end of the model using an attention mechanism, ParaCRN-AMResNet opts to integrate a ResNeSt structure and SimAM attention mechanism within the parallel convolutional component to supplement potentially missed features. The use of the SimAM attention mechanism does not add extra parameters, thereby avoiding additional computational burden and further optimizing the model's performance and efficiency.

3.8. Noise Resistance Experiment

Considering that real manufacturing environments are often accompanied by strong noise, it becomes crucial to assess the stability and prediction accuracy of high-performance models in such contexts. To this end, a series of noise interference experiments were designed to verify the noise resistance capability of the proposed model. Specifically, to simulate extreme working conditions, Gaussian white noise with signal-to-noise ratios (SNRs) of -1 dB, -3 dB, -5 dB, -7 dB, and -9 dB was added to the original signals. Furthermore, models such as ParaCRN-AM, without attention, CNN-RNN, and RNN-CNN were selected as controls to assess their performance durability under different noise conditions. The evaluation criteria were still based on the four metrics: MAPE, MAE, RMSE, and R^2. The detailed experimental results are presented in Figure 12.

The data in Figure 12 reveal a clear trend: as the intensity of the noise increases, the predictive performance of all models shows a declining trend. However, among all models examined, ParaCRN-AMResNet stands out in its performance. Remarkably, even under extreme conditions with an SNR of -9 dB, the model still provides satisfactory prediction results, with corresponding MAE, MSE, R^2, and MAPE values of 13.8012, 282.4792, 0.8080, and 14.2770%, respectively.

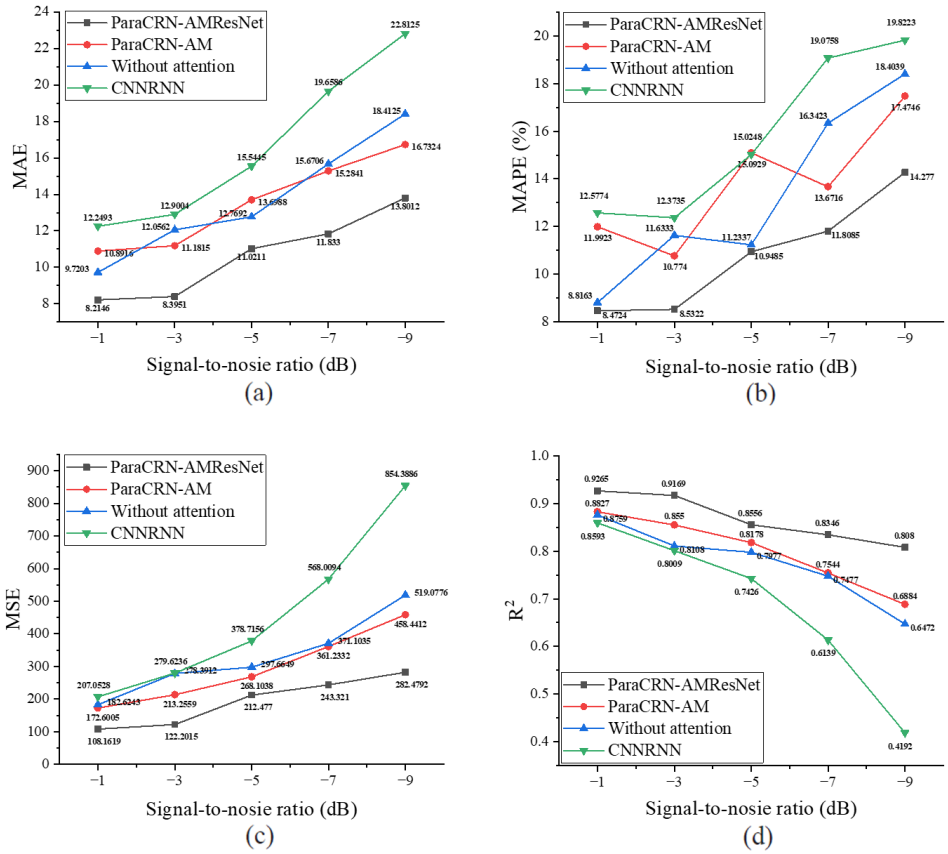

Figure 12. The comparative performance of various models at different SNRs: (**a**) MAE of different models across various SNRs; (**b**) MAPE of different models across various SNRs; (**c**) MSE of different models across various SNRs; (**d**) R^2 of different models across various SNRs.

The ParaCRN-AMResNet model is able to mine and correlate more sensitive features from signals mixed with noise, demonstrating exceptional noise resistance capability. Furthermore, the superiority of the parallel-structured model over the CNN-RNN sequential model further confirms that the proposed parallel structure can successfully avoid mutual interference between modules. In contrast, the performance of the RNN-CNN model in an SNR = −1 dB environment is even worse than a simple mean prediction; hence, its data were not included in Figure 12. The performance comparison between the without attention model and ParaCRN-AMResNet further verifies that the introduced attention module can help the model capture key features in noisy environments. Additionally, the performance of the ParaCRN-AM model compared with ParaCRN-AMResNet demonstrates the ability of the residual block structure to help the model capture additional useful information in harsh environments.

In summary, Figure 13 further compares the predictive results of the ParaCRN-AMResNet model with the actual outcomes, both in the presence and absence of −9 dB noise. Despite the intense background noise, the predictive results still accurately capture the trend of tool wear. This undoubtedly demonstrates the immense industrial application value of the ParaCRN-AMResNet model.

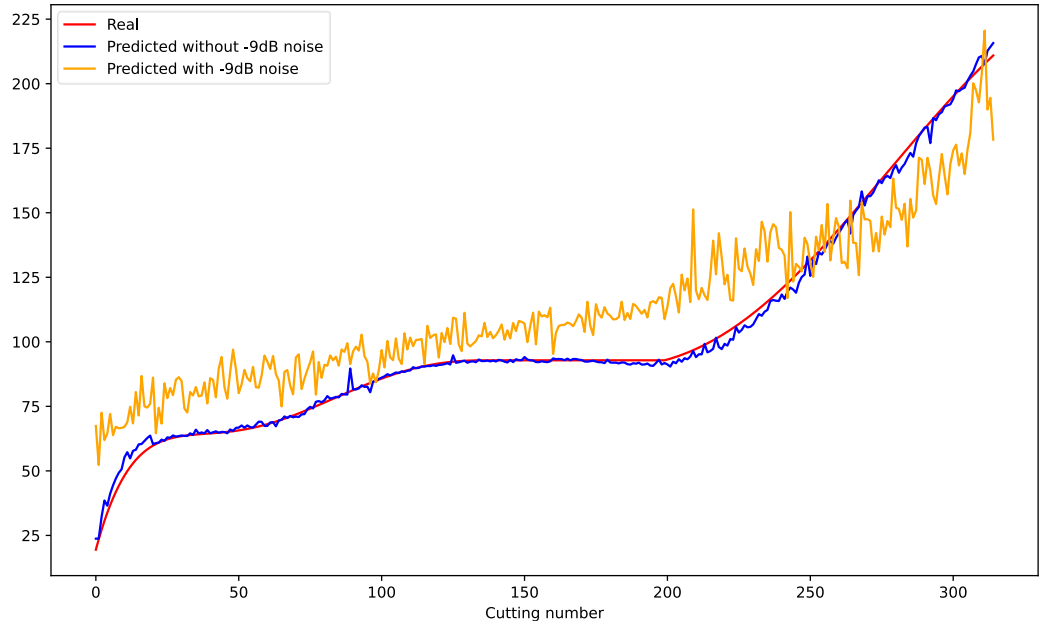

Figure 13. The prediction of the ParaCRN-AMResNet model with and without −9 dB noise.

4. Conclusions

In this paper, a novel hybrid deep learning model, ParaCRN-AMResNet, is proposed for the prediction of tool wear. The raw signals are decomposed using wavelet analysis as input data. Subsequently, the model enhances the discernibility of temporally sensitive features through the incorporation of the SimAM attention layer, further employing dilated convolutional neural networks and the ResNeSt structure to capture temporally sensitive features across various scales. BiGRU is incorporated into the model, working in parallel with dilated CNN to capture time-series information. A GAP layer is applied to reduce redundant spatial features and enhance the model's interpretability. These features are fused and used to predict tool wear. The results of conducted ablation experiments, comparative trials, and noise resistance capability tests indicate that:

(1) The parallel structure ensures that each feature extraction pathway operates correctly without interference from others. This approach avoids the impact of the former model on subsequent models.
(2) The use of dilated CNN effectively captures the intrinsic temporal correlations within time-series data, and ResNeSt additionally supplements crucial information for the convolutional component. Meanwhile, BiGRU with different sizes can effectively capture meaningful representations across various temporal dimensions.
(3) Experimental validation demonstrates that ParaCRN-AMResNet outperforms other deep learning models in tool wear prediction, achieving MAE, MSE, R^2, and MAPE values of 2.6015, 15.1921, 0.9897, and 2.7997%, respectively.

While this study was validated on a single dataset, potentially limiting the generalizability of the model's predictive accuracy across different conditions or datasets, and the training duration of the model hinders its immediate deployment in practical settings, future research will focus on how to effectively extend the model's applicability through transfer learning. By transferring knowledge acquired on a specific task to related but distinct tasks, this approach not only aims to enhance the model's adaptability but also significantly reduce the required training time and resources. This research direction seeks to amplify the practicality of the ParaCRN-AMResNet model, offering more flexible and efficient solutions for the advancement of intelligence in manufacturing.

The primary objective of this paper is not to propose an immediately applicable solution but rather to explore a promising approach aimed at enhancing the generalization

performance of tool wear prediction models. By accurately predicting tool wear in real-time, proactive maintenance of CNC machining tools has been facilitated.

Author Contributions: The first author, L.G., was responsible for writing this paper, designing the experimental process, and analyzing the experiment results. The corresponding author, Y.W., was responsible for determining the overall logical structure of the paper and guiding the entire experiment. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare that there are no conflicts of interest.

References

1. Mohanraj, T.; Yerchuru, J.; Krishnan, H.; Nithin Aravind, R.S.; Yameni, R. Development of Tool Condition Monitoring System in End Milling Process Using Wavelet Features and Hoelder's Exponent with Machine Learning Algorithms. *Measurement* **2021**, *173*, 108671. [CrossRef]
2. Zhu, K.; Zhang, Y. A Generic Tool Wear Model and Its Application to Force Modeling and Wear Monitoring in High Speed Milling. *Mech. Syst. Signal Process.* **2019**, *115*, 147–161. [CrossRef]
3. Ren, H.; Guo, W.; Jiang, P.; Wan, X. An Integrated Approach of Active Incremental Fine-Tuning, SegNet, and CRF for Cutting Tool Wearing Areas Segmentation with Small Samples. *Knowl.-Based Syst.* **2021**, *218*, 106838. [CrossRef]
4. Li, X.; Liu, X.; Yue, C.; Liang, S.Y.; Wang, L. Systematic Review on Tool Breakage Monitoring Techniques in Machining Operations. *Int. J. Mach. Tools Manuf.* **2022**, *176*, 103882. [CrossRef]
5. Pimenov, D.Y.; Bustillo, A.; Wojciechowski, S.; Sharma, V.S.; Gupta, M.K.; Kuntoğlu, M. Artificial Intelligence Systems for Tool Condition Monitoring in Machining: Analysis and Critical Review. *J. Intell. Manuf.* **2023**, *34*, 2079–2121. [CrossRef]
6. Mohamed, A.; Hassan, M.; M'Saoubi, R.; Attia, H. Tool Condition Monitoring for High-Performance Machining Systems—A Review. *Sensors* **2022**, *22*, 2206. [CrossRef]
7. Lecun, Y. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
8. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
9. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
10. Kumar, M.P.; Dutta, S.; Murmu, N.C. Tool Wear Classification Based on Machined Surface Images Using Convolution Neural Networks. *Sādhanā* **2021**, *46*, 130. [CrossRef]
11. Lim, M.L.; Derani, M.N.; Ratnam, M.M.; Yusoff, A.R. Tool Wear Prediction in Turning Using Workpiece Surface Profile Images and Deep Learning Neural Networks. *Int. J. Adv. Manuf. Technol.* **2022**, *120*, 8045–8062. [CrossRef]
12. García-Pérez, A.; Ziegenbein, A.; Schmidt, E.; Shamsafar, F.; Fernández-Valdivielso, A.; Llorente-Rodríguez, R.; Weigold, M. CNN-Based in Situ Tool Wear Detection: A Study on Model Training and Data Augmentation in Turning Inserts. *J. Manuf. Syst.* **2023**, *68*, 85–98. [CrossRef]
13. Zhang, X.; Shi, B.; Feng, B.; Liu, L.; Gao, Z. A Hybrid Method for Cutting Tool RUL Prediction Based on CNN and Multistage Wiener Process Using Small Sample Data. *Measurement* **2023**, *213*, 112739. [CrossRef]
14. Brili, N.; Ficko, M.; Klančnik, S. Automatic Identification of Tool Wear Based on Thermography and a Convolutional Neural Network during the Turning Process. *Sensors* **2021**, *21*, 1917. [CrossRef]
15. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
16. Shah, M.; Vakharia, V.; Chaudhari, R.; Vora, J.; Pimenov, D.Y.; Giasin, K. Tool Wear Prediction in Face Milling of Stainless Steel Using Singular Generative Adversarial Network and LSTM Deep Learning Models. *Int. J. Adv. Manuf. Technol.* **2022**, *121*, 723–736. [CrossRef]
17. Li, X.; Liu, X.; Yue, C.; Liu, S.; Zhang, B.; Li, R.; Liang, S.Y.; Wang, L. A Data-Driven Approach for Tool Wear Recognition and Quantitative Prediction Based on Radar Map Feature Fusion. *Measurement* **2021**, *185*, 110072. [CrossRef]
18. Mahmood, J.; Luo, M.; Rehman, M. An Accurate Detection of Tool Wear Type in Drilling Process by Applying PCA and One-Hot Encoding to SSA-BLSTM Model. *Int. J. Adv. Manuf. Technol.* **2022**, *118*, 3897–3916. [CrossRef]
19. Marani, M.; Zeinali, M.; Songmene, V.; Mechefske, C.K. Tool Wear Prediction in High-Speed Turning of a Steel Alloy Using Long Short-Term Memory Modelling. *Measurement* **2021**, *177*, 109329. [CrossRef]
20. Bilgili, D.; Kecibas, G.; Besirova, C.; Chehrehzad, M.R.; Burun, G.; Pehlivan, T.; Uresin, U.; Emekli, E.; Lazoglu, I. Tool Flank Wear Prediction Using High-Frequency Machine Data from Industrial Edge Device. *Procedia CIRP* **2023**, *118*, 483–488. [CrossRef]
21. Marei, M.; Li, W. Cutting Tool Prognostics Enabled by Hybrid CNN-LSTM with Transfer Learning. *Int. J. Adv. Manuf. Technol.* **2022**, *118*, 817–836. [CrossRef]
22. Chaowen, Z.; Jing, J.; Chi, C. Research on Tool Wear Monitoring Based on GRU-CNN. In Proceedings of the 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 9 April 2021; IEEE: Xi'an, China, 2021; pp. 729–733. [CrossRef]

23. Si, Z.; Si, S.; Mu, D. Efficient Tool Wear Prediction in Manufacturing: BiLPReS Hybrid Model with Performer Encoder. *Arab. J. Sci. Eng.* **2024**. [CrossRef]
24. Li, R.; Ye, X.; Yang, F.; Du, K.-L. ConvLSTM-Att: An Attention-Based Composite Deep Neural Network for Tool Wear Prediction. *Machines* **2023**, *11*, 297. [CrossRef]
25. Bazi, R.; Benkedjouh, T.; Habbouche, H.; Rechak, S.; Zerhouni, N. A Hybrid CNN-BiLSTM Approach-Based Variational Mode Decomposition for Tool Wear Monitoring. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 3803–3817. [CrossRef]
26. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. In Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016; ACL: San Juan, Puerto Rico, 2016; p. 10061. [CrossRef]
27. Chang, S.; Zhang, Y.; Han, W.; Yu, M.; Guo, X.; Tan, W.; Cui, X.; Witbrock, M.; Hasegawa-Johnson, M.A.; Huang, T.S. Dilated Recurrent Neural Networks. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17), Red Hook, NY, USA, 6 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 76–86. [CrossRef]
28. Lin, M.; Chen, Q.; Yan, S. Network In Network. In Proceedings of the International Conference on Learning Representations (ICLR), Banff, AB, Canada, 14–16 April 2014; OpenReview.net. ACL: Banff, AB, Canada, 2014. [CrossRef]
29. Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; ACL: Doha, Qatar, 2014; pp. 1724–1734. [CrossRef]
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Las Vegas, NV, USA, 2016; pp. 770–778. [CrossRef]
31. Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Lin, H.; Zhang, Z.; Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; et al. ResNeSt: Split-Attention Networks. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 19–20 June 2022; IEEE: New Orleans, LA, USA, 2022; pp. 2735–2745. [CrossRef]
32. Yang, L.; Zhang, R.-Y.; Li, L.; Xie, X. SimAM: A Simple, Parameter-Free Attention Module for Convolutional Neural Networks. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 11863–11874.
33. 2010 PHM Society Conference Data Challenge 2010. Available online: https://phmsociety.org/phm_competition/2010-phm-society-conference-data-challenge/ (accessed on 10 August 2023).
34. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2015**, arXiv:1412.6980. [CrossRef]
35. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for Activation Functions. *arXiv* **2017**, arXiv:1710.05941. [CrossRef]
36. Hahn, T.V.; Mechefske, C.K. Self-Supervised Learning for Tool Wear Monitoring with a Disentangled-Variational-Autoencoder. *Int. J. Hydromechatron.* **2021**, *4*, 69–98. [CrossRef]
37. Yang, J.; Wu, J.; Li, X.; Qin, X. Tool Wear Prediction Based on Parallel Dual-Channel Adaptive Feature Fusion. *Int. J. Adv. Manuf. Technol.* **2023**, *128*, 145–165. [CrossRef]
38. He, Z.; Liu, Y.; Pang, X.; Zhang, Q. Wear Prediction of Tool Based on Modal Decomposition and MCNN-BiLSTM. *Processes* **2023**, *11*, 2988. [CrossRef]