*sustainability*

MDPI

*Article*

# An Integrated Deep-Learning-Based Approach for Energy Consumption Prediction of Machining Systems

**Meihang Zhang [1,2], Hua Zhang [1,3], Wei Yan [4,5,*], Zhigang Jiang [1,2] and Shuo Zhu [3,4]**

[1] Key Laboratory of Metallurgical Equipment and Control Technology, Ministry of Education, Wuhan University of Science and Technology, Wuhan 430081, China; zhangmeihang@wust.edu.cn (M.Z.); zhanghua@wust.edu.cn (H.Z.); jiangzhigang@wust.edu.cn (Z.J.)
[2] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan 430081, China
[3] Precision Manufacturing Research Institute of Wuhan University of Science and Technology, Wuhan 430081, China; zhushuo@wust.edu.cn
[4] Academy of Green Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan 430081, China
[5] School of Automotive and Traffic Engineering, Wuhan University of Science and Technology, Wuhan 430065, China
* Correspondence: yanwei81@wust.edu.cn

**Abstract:** Large and extensive manufacturing systems consume a large proportion of manufacturing energy. A key component of energy efficiency management is the accurate prediction of energy efficiency. However, the nonlinear and vibration characteristics of machining systems' energy consumption (EC) pose a challenge to the accurate prediction of system EC. To address this challenge, an energy consumption prediction method for machining systems is presented, which is based on an improved particle swarm optimization (IPSO) algorithm to optimize long short-term memory (LSTM) neural networks. The proposed method optimizes the LSTM hyperparameters by improving the particle swarm algorithm with dynamic inertia weights (DIWPSO-LSTM), which enhances the prediction accuracy and efficiency of the model. In the experimental results, we compared several improved optimization algorithms, and the proposed method has a performance improvement of more than 30% in mean absolute error (*MAE*)and mean error(*ME*).

**Keywords:** energy consumption prediction; particle swarm optimization algorithm of dynamic inertia weights (DIWPSO); long short-term memory network (LSTM); machining systems; deep learning



**Citation:** Zhang , M.; Zhang, H.; Yan, W.; Jiang, Z.; Zhu, S. An Integrated Deep-Learning-Based Approach for Energy Consumption Prediction of Machining Systems. *Sustainability* **2023**, *15*, 5781. https://doi.org/10.3390/su15075781

Academic Editor: Jun Qin

Received: 9 February 2023
Revised: 20 March 2023
Accepted: 22 March 2023
Published: 27 March 2023



## 1. Introduction

Energy and environmental issues have received extensive attention in recent years [1–3]. Due to the wide use of machine tools, machining systems are considered to be the largest energy consumer in the manufacturing industry [4,5]. Research shows that the EC of machining systems accounts for 74.7% of the total EC of the manufacturing industry [6]. The growing energy demand has become an important factor in current environmental and economic challenges [7]. If the EC of the processing process can be predicted in advance and optimized in a targeted manner, it will have a positive effect on energy problems and environmental improvement.

Experts and scholars have conducted numerous studies on the energy consumption prediction of machining systems. The more mainstream EC forecasting methods can be divided into traditional forecasting methods and data-driven forecasting methods. Traditional EC forecasting models include time series models, regression models, and gray models. Predictions are made primarily through historical data, empirical formula derivation, and limited experimental data. Liu et al. [8] used a linear empirical model to describe the relationship between the tool tip cutting power and the total power consumption of the

machine tool. Shi et al. [9] developed an improved cutting-power-based EC model which consists of an idle part as a function of spindle speed and an additional part proportional to the cutting power. Wang et al. [10] took the machining feature as the entry point; divided the prism machining feature into plane, step, groove, hole, etc.; and then summed the cutting EC of all features. Based on the model of the material removal mechanism, He et al. [11] proposed a method for predicting the machining EC of turning workpieces that integrates design parameters and manufacturing parameters. Lv et al. [12] established a spindle acceleration EC model for computer numerical control (CNC) lathes and derived and counted spindle acceleration EC by theoretical equations. In other EC forecasts, time series models are used more widely. A hybrid-autoregressive-score-integrated moving average and least square support vector machine model to predict short-term wind power generation is proposed by Yuan et al. [13].

On the other hand, with the development of artificial intelligence, data-driven forecasting is booming. Hu et al. [14] considered that the EC of machine tools includes cutting and non-cutting EC and used two optimization methods, depth-first search (DFS) and genetic algorithm (GA), to optimize the machining sequence to reduce EC. Brillinger et al. [15] and Li et al. [16] used the use of RF and GPR to construct EC prediction models for CNC machining processes, respectively. Qin et al. [17] used linear regression (LR), k-nearest neighbor (k-NN), and decision tree (DT) to predict EC in additive manufacturing processes. Liu et al. [18] proposed a hybrid prediction method combining data-driven machine learning and process mechanics to predict the specific cutting energy. Kim et al. [19] created a transfer learning approach for processing power prediction by transferring knowledge gained from previous processing to a target processing environment lacking processing power data. He et al. [20] and Kahraman et al. [21] used a convolutional neural network (CNN) and deep neural networks as the prediction model.

Machine learning methods (such as BP, SVM, and RF) are shallow networks, and the prediction accuracy still needs to be improved. Deep-learning-based prediction methods are mostly used for the prediction of time series data, such as the prediction of building energy consumption using LSTM [22–24], predicting residential electricity consumption using CNN and LSTM [25–27], and predicting building energy consumption using GAN networks [28]. Alternatively, a comparison between traditional machine learning and deep learning methods finds that deep learning prediction methods have better results. For example, Xiao et al. [29] used three traditional machine learning algorithms and three deep learning methods for a comparative study of workshop energy consumption, showing that traditional machine learning algorithms are more suitable for small sample situations, while deep learning algorithms have better feature learning performance on images and time series. A relatively small number of studies have been conducted on the use of deep learning models for energy consumption prediction of machining systems at present. This is mainly because the machining system is a typical nonlinear system with dynamic disturbance factors. The processing energy consumption is closely related to the processing task, processing technology, processing characteristics, and processing environment. Therefore, it is challenging to establish a complete energy consumption prediction model. Additionally, it is difficult to collect data on machining system energy consumption characteristics. Therefore, we wanted to construct a basic energy consumption dataset, integrating influencing factors such as process parameters, processing time, tool blade number, etc. At the same time, the optimization algorithm is integrated into the deep learning model to improve the efficiency of super parameter selection. This is to improve the prediction efficiency and accuracy of the energy consumption prediction model.

Therefore, this paper constructs an energy consumption prediction model for machining systems based on deep LSTM techniques, automatically optimizing the hyperparameters of LSTM by an improved particle swarm algorithm, reducing the time of manual parameter tuning, and improving the accuracy and prediction efficiency of short-time machining system EC prediction. The contributions of this paper are as follows:

- A framework for predicting the EC of machining systems based on deep learning is proposed. A particle swarm algorithm is improved using dynamic inertia weights, and then a double-layer LSTM network is introduced to predict the EC of machining systems.
- A novel method for constructing EC datasets is proposed. This method is intended to improve the generalization ability of the prediction model by mixing EC data generated in different ways to produce a novel EC dataset.
- The EC patterns of the machining systems are classified according to the processing characteristics. The processing energy consumption data of different processing characteristics are collected, so as to accurately predict the processing EC.

This paper is organized as follows. In the second section, the related work is reviewed. The third section proposes a data-driven method for predicting EC in milling operations. In the fourth section, we build an experimental platform and conduct case analyses based on it. The conclusion is presented in the last section.

## 2. Theoretical Background

This section details the theoretical background of the EC prediction of machining systems, particle swarm optimization algorithms, and long short-term memory neural networks.

### *2.1. EC Characteristics of Machining Systems*

Consider multi-source EC prediction for machining systems with $n$ different variables, that is, deploying multiple sensors in the machining systems, such as photoelectric velocity sensors on the spindle and high-temperature infrared sensors near the spindle. For each time stamp, the energy consumed by the machining systems consists of energy-consuming components, such as spindles, feed axes, lighting units, hydraulic lubrication units, and workpiece characteristics, and process parameters. Different energy-consuming component units contain a variety of processing parameters, mainly including spindle speed, feed rate, cutting depth, number of cutting edges, tool diameter, etc., which can be expressed by Equation (1).

$$X = \begin{bmatrix} x_1^1, x_1^2, x_1^3, \cdots x_1^j, \cdots x_1^n \\ x_2^1, x_2^2, x_2^3, \cdots x_2^j, \cdots x_2^n \\ \vdots \qquad\qquad \vdots \\ x_t^1, x_t^2, x_t^3, \cdots x_t^j, \cdots x_t^n \end{bmatrix} \tag{1}$$

where $x_t^j$ represents the value of the $j$-th parameter at moment $t$.

Under the action of various parameters, the instantaneous EC of the machining system at time $t$ is $E_t$. The $E_t$ can be expressed by Equation (2).

$$E_t = f(V_r, V_f, a_p, a_e, N_e), \tag{2}$$

where $V_r$, $V_f$, $a_p$, $a_e$, $N_e$ indicate the spindle speed, feed rate, depth of cut, cutting width, and number of cutting edges, etc.

### *2.2. Long Short-Term Memory*

The LSTM was originally proposed by Hochreiter and Schmidhuber [30]. Due to the rise of deep learning, LSTM has undergone several development phases, resulting in a relatively systematic and complete LSTM framework, which has been widely used in many fields. The LSTM neural networks are a significant improvement to RNN networks that use self-connected memory units and gate units in the hidden layer to solve the vanishing gradient problem or gradient explosion problem in RNN [31]. The LSTM has a strong generalization capability and high learning ability for both larger and smaller datasets,

which is advantageous in dealing with nonlinear problems [32]. The structure of the basic LSTM unit is shown in Figure 1.

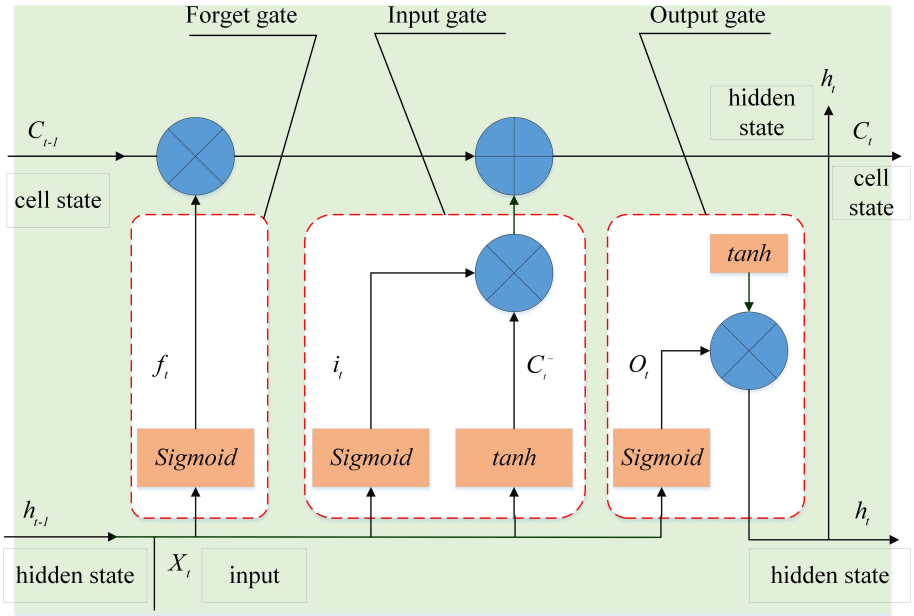


**Figure 1.** LSTM basic unit structure.

Unlike traditional RNN, LSTM consists of memory blocks with one or more memory cells acting as neurons with multiplicative gates (input gate $i_t$, forget gate $f_t$, and output gate $O_t$). The $f_t$ is a vector, which can be expressed by Equation (3), usually using sigmoid as the activation function; the output of sigmoid is a value between [0,1].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \tag{3}$$

where $f_t$ is called the forget gate; $\sigma$ is the sigmoid function; $x_t$ is the input at time $t$; $W_f$, $b_f$ are the weight matrix and bias vector; and $h_{t-1}$ is the hidden layer state at time $t-1$.

The input gate $i_t$ is used to update the cell state. The output value of sigmoid is multiplied by the output value of tanh to decide which information is important and needs to be retained. The input gate $i_t$ is calculated by Equation (4).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i). \tag{4}$$

where $i_t$ is the input gate, and $W_i$, $b_i$ are the weight matrix and bias vector.

The current information received by the memory cell is represented by Equation (5), and the cell state is calculated by Equation (6).

$$C_t^{\sim} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c). \tag{5}$$

$$C_t = f_t * C_{t-1} + i_t * C_t^{\sim}. \tag{6}$$

where $C_t^{\sim}$ is the input node state at time $t$, and $C_{t-1}$ and $C_t$ are the cell states at time $t-1$ and $t$, respectively. Tanh is a hyperbolic tangent function. $W_c$,$b_c$ are the corresponding weight matrix and bias vector.

The output gate $O_t$ is used to determine the value of the next hidden state, which contains the information of the previous input. The previous hidden state and the current input are passed to the sigmoid function, and then the newly obtained cell state is passed to the tanh function. Subsequently, the output of tanh is multiplied with the output of

sigmoid to determine what information the hidden state should carry. The mathematical model of the output gate is represented as follows:

$$O_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{7}$$

and,

$$h_t = O_t * \tanh(C_t). \tag{8}$$

where $\sigma$ is the sigmoid function, and $h_{t-1}$ and $h_t$ are the outputs at time $t-1$ and $t$, respectively; $W_o$, $b_o$ are the corresponding weight matrix and bias vector.

In practical applications, the initial values of the weights and bias terms of the LSTM are randomly generated during the training process. As the selection of the key parameters of LSTM greatly influences the prediction effect of processing EC, these hyperparameters need to be selected thoughtfully.

### *2.3. Particle Swarm Optimization Algorithm*

Particle swarm optimization (PSO) is an evolutionary computation technique whose basic idea is to find the optimal solution through collaboration and information sharing among individuals in a population [33]. The particles have two attributes: velocity and position. Each particle individually searches for the optimal solution in the search space, which is recorded as the most recent individual extreme value and shared with the other particles in the whole particle swarm. All particles in the swarm adjust their velocities and positions according to the current individual extreme values they find. In addition, they share a global optimal solution shared by the whole swarm. The updated equation for the velocity and position of the $i$-th particle at time $t$ is

$$V_{i,t} = \omega V_{i,t-1} + C_1 R_1 (P_{i,t-1} - X_{i,t-1}) + C_2 R_2 (G_{i,t-1} - X_{i,t-1}), \tag{9}$$

and

$$X_{i,t} = X_{i,\text{t-1}} + V_{i,t}, \tag{10}$$

where $\omega$ is the inertia factor; $V_{i,t-1}$ is the velocity of the $i$-th particle at moment $t-1$; $C_1$ and $C_2$ are the acceleration constants, and $C_1, C_2 \in [0, 4]$; $R_1$ and $R_2$ are random numbers between [0, 1]; $P_{i,t-1}$, $X_{i,t-1}$, and $G_{i,t-1}$ are the individual optimal value, position, and global optimal solution of the $i$-th example at moment $t-1$, respectively.

## 3. Methodology

In this section, an energy consumption prediction method relying on particle swarm algorithm optimized LSTM with dynamic inertia weights (DIWPSO-LSTM) is introduced. Firstly, the framework of energy consumption prediction is built, and then the process of the energy consumption prediction method based on deep learning is introduced. This prediction method is a data-driven deep learning method that can accurately predict the short-term energy consumption of machining systems between a cutting and non-cutting energy consumption mode state change (time change).

### *3.1. A Framework for Predicting EC in Machining Systems*

The proposed processing EC prediction model framework consists of four parts: (1) data acquisition and storage layer; (2) data preprocessing layer; (3) data analysis layer; (4) application layer. Each layer is composed of several modules to realize the prediction of processing EC, as shown in Figure 2.

**(1) Data acquisition and storage**

The EC data acquisition of the machining systems includes two methods: (1) collect power, current, and voltage data through a high-precision power tester; (2) arrange photoelectric sensors on the spindle and feed axis, and use LabVIEW programming to collect spindle speed, power, EC, and other data. The data generated in both ways are saved

in csv format for further processing and use. Details about the data generation and the construction of the dataset are given in Section 4.2.

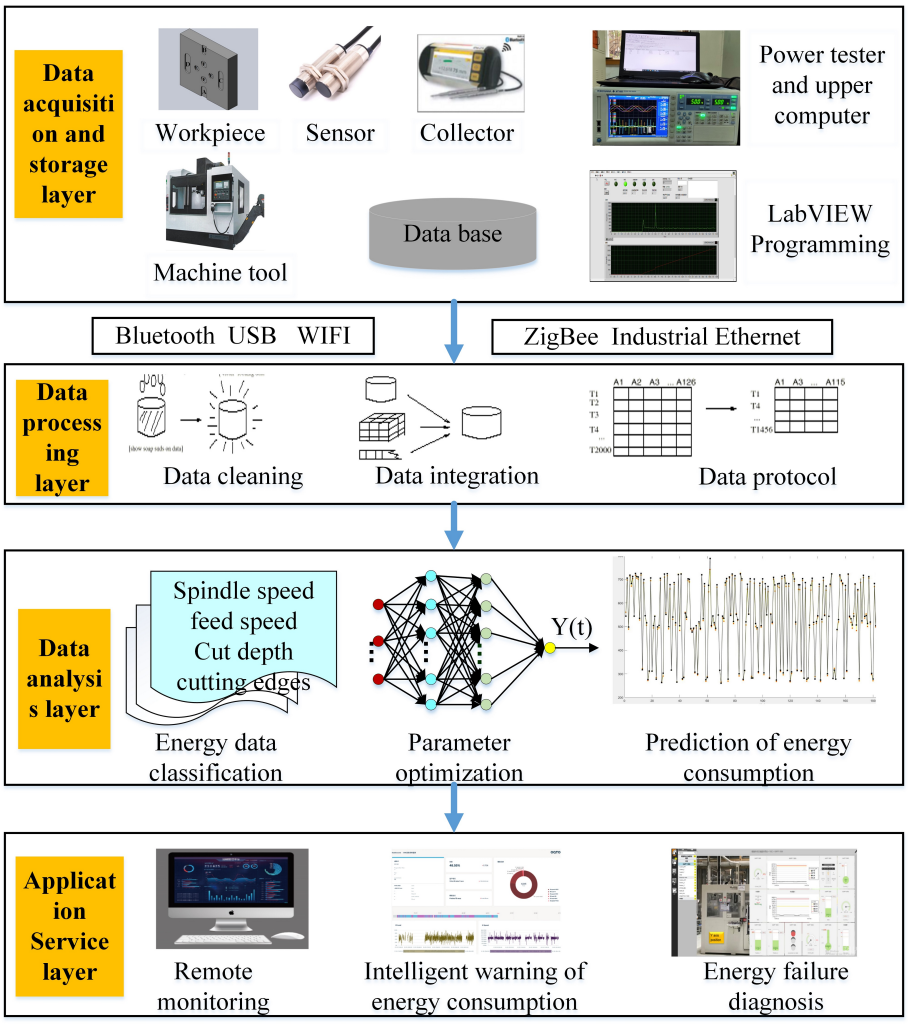


**Figure 2.** A framework for EC prediction of machining system.

**(2) Data preprocessing**

Raw data collected and stored by a power tester or LabVIEW platform often contain noise or missing data due to malfunctions, etc. Generally, missing data are cleaned by sliding windows, moving average filters, and interpolation techniques [34]. The data were also subjected to max-min normalization before model training to facilitate stable convergence of the weights and biases of the learning model. The preprocessed EC datasets were randomly mixed, and a total of 80% of the energy consumption dataset was used for training, 20% of the data was used for prediction, and 200 groups of verification data were randomly selected from the training data.

**(3) Data analysis**

The data analysis layer is used to analyze the prediction based on the processed data and optimize the hyperparameters of the LSTM network (such as learning rate, number of hidden layers, and gradient threshold, etc.), utilizing an IPSO algorithm to improve prediction accuracy and efficiency. The double-layer LSTM network is then used to predict the EC of the milling machining process. The training is completed when the root mean square error and loss function of DIWPSO-LSTM are minimized.

**(4) Application**

In the production workshop or machining center, we can remotely monitor the state of machining energy consumption of mechanical machining systems to reduce the production

cost. In addition, we use DIWPSO-LSTM to predict the energy consumption of processing. If instantaneous processing energy consumption deviates from the set threshold range, an emergency or fault may occur during the processing of the workpiece. Energy consumption data can be used by operators to provide energy efficiency alerts and adjust processing parameters or shut down equipment for inspection to reduce energy consumption.

### *3.2. DIWPSO-LSTM: Energy Consumption Prediction Method*

The main goal of DIWPSO-LSTM is to minimize the prediction error and iterative training time of the LSTM by identifying the optimal combination of hyperparameters. The double-layer LSTM for the milling machining EC prediction model consists of an input layer, hidden layers, dropout layers, fully connected layer, and output layer. The structure of the double-layer LSTM prediction model is shown in Figure 3. The pseudocode of the process energy prediction model is shown in Algorithm 1.

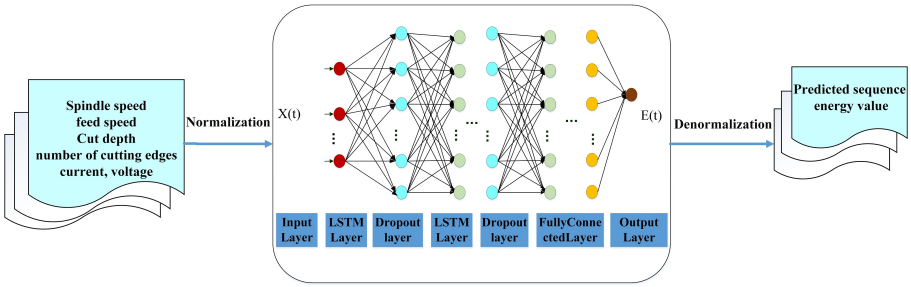


**Figure 3.** Double-layer LSTM prediction model network structure.

The overall work of DIWPSO-LSTM can be divided into six stages: (1) data preprocessing; (2) encoding; (3) training a double-layer LSTM; (4) evaluating DIWPSO-LSTM; (5) termination condition; and (6) position update. The workflow is shown in Figure 4. The steps of DIWPSO-LSTM are as follows.

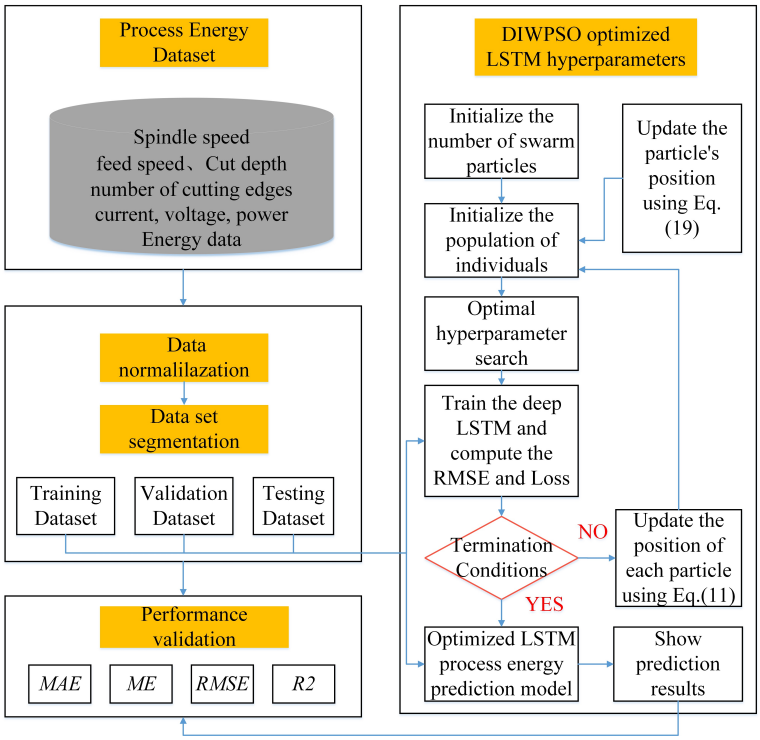


**Figure 4.** DIWPSO-LSTM workflow.

**Algorithm 1** Process energy prediction model based on DIWPSO-LSTM.

**Input:** $D_{EC} \leftarrow x(t), s.t.\ t \in N^*$ and $t \leq n$ // Energy consumption dataset
**Output:** $Hun^* \leftarrow$ Optimal number of Hidden units;
$Lr^* \leftarrow$ Optimal learning rate;
$Lrdf^* \leftarrow$ Optimal learning rate descent factor;
$Lrdp^* \leftarrow$ Optimal learning rate descent period.

**function** DIWPSO-LSTM()
  Initialize the values of number of population ($N_{Ppv}$), maximum number of generation ($t_{\max}$), and:
    $t \leftarrow 0$, $fit \leftarrow 0$, $best_{fit} \leftarrow 0$, $gbest_{fit} \leftarrow 0$
  Data set segmentation:
    $D_{train} \leftarrow x(t); \{t = 1, 2, \cdots, n * 0.8\}$
    $D_{val} \leftarrow x(t); \{t = (n * 0.8 - 200), \cdots n * 0.8\}$
    $D_{test} \leftarrow x(t); \{t = (n * 0.8 + 1), \cdots n\}$
  **for** each $i \in [1,\ N_{Ppv}]$ **do**
    **for** each dimension $t$ **do**
      Initailize the positon and velocity $Ppv_i$ and $v_{i,t}$, and compute the hyperparameters using Equations (11) and (12).
    **end for**
  **end for**
  Train the double-layer LSTM using computed hyperparameters and $D_{train}$.
  Calculate the $fit_i$ for $Ppv_i$ with $D_{val}$.
  **if** $fit_i(RMSE) \leq fit(pbest_i)$ **then**
    $pbest_{fit} \leftarrow best_{fit}$
  **end if**
  $gbest_{Ppv} \leftarrow Ppv(bestindex)$
  Update the position with $gbest_{Ppv}$ using Equation (19).
  **for** each $i \leftarrow 1$ *to* $N_{Ppv}$ **do**
    **while** $t \leq t_{\max}$ **do**
      Compute the $Ppv_i$ using Equation (12).
      Train the LSTM using computed hyperparameters and $D_{train}$.
      Calculate the $fit_i$ for $Ppv_i$ with $D_{val}$.
      **if** $fit_i(RMSE) \leq fit(pbest_i)$ **then**
        $pbest_{fit} \leftarrow best_{fit}$
        $gbest_{Ppv} \leftarrow Ppv(bestindex)$
      **end if**
      $t \leftarrow t + 1$
    **end while**
  **end for**
  Construction of double-layer LSTM models using optimal hyperparameters.
  Predict the results using the $D_{test}$ and calculate *MAE*, *ME*, *RMSE*, *R2*.
**end function**

**Step 1: Data preprocessing**. Interpolation and smoothing techniques are used to process the collected experimental data to obtain the processing energy dataset, and then the mapminmax is applied to normalize the processing energy dataset X, so that the data in the dataset are all in the range of $[-1, 1]$. The training and test samples are drawn using a no-replacement random sampling technique with a ratio of 8:2. Then, 200 samples are randomly selected for verification.

**Step 2: Encoding strategy**. The DIWPSO-LSTM uses a vector encoding strategy to generate the initial population because it has to adjust several parameters (such as the number of hidden units, learning rate, learning rate descent factor, maximum number of iterations, etc.), each with a defined range. In the encoding strategy, the position of each population is represented as a vector whose length corresponds to the number of parameters to be optimized. We optimize four hyperparameters in our paper, namely number of hidden units(Hun), learning rate(Lr), learning rate descent factor(Lrdf), and

learning rate descent period(Lrdp) in DIWPSO-LSTM. The particle population vector is ($Ppv$) expressed in Equation (11).

$$Ppv_i = [Hun_i, Lr_i, Lrdf_i, Lrdp_i]; i = (1, 2 \cdots, N_{Ppv}), \tag{11}$$

where $N_{Ppv}$ is the total number of the population.

We generate each population vector in a random manner in the range [0, 1] and convert it to the corresponding specific parameter range using Equation (12):

$$Op_v = Ppv * (Ppv_{Max} - Ppv_{Min}) + Ppv_{Min}, \tag{12}$$

where $Op_v$ is the overall population vector; $Ppv_{Max}$ and $Ppv_{Min}$ are the the maximum values and minimum values of the hyperparameter; and $Ppv$ is the randomly generated population.

**Step 3: Training the double-layer LSTM**. During the training process, the hyperparameters obtained from each population (Step 2) and the training dataset are used to train the double-layer LSTM. A double-layer LSTM for superposition is found to have a better prediction effect based on experiments with different layers of LSTM. Finally, a double-layer LSTM is chosen as the model used in this experiment, and the prediction effect is significantly higher than that of the low-level model. Because the more layers that are superimposed, the higher the computational consumption and the greater the chance of overfitting, the double-layer LSTM model is the most suitable choice.

**Step 4: Evaluating DIWPSO-LSTM**. The hyperparametric training LSTM performance is evaluated for each population using the validation dataset in terms of loss function and root mean square error ($RMSE$) as shown in Equations (13) and (14). The performance evaluation of the double-layer LSTM model prediction results is evaluated in terms of mean absolute error ($MAE$), mean error ($ME$), root mean square error ($RMSE$), and coefficient of determination ($R2$). The specific formulas are given in Equations (14)–(17).

$$LOSS = \frac{1}{N}\sum_{t=1}^{N}(|\hat{y}_t - y_t|)^2, \tag{13}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}|\hat{y}_t - y_t|}, \tag{14}$$

$$MAE = \frac{1}{N}\sum_{t=1}^{N}|\hat{y}_t - y_t|, \tag{15}$$

$$ME = \max(|\hat{y}_t - y_t|), \tag{16}$$

$$R2 = 1 - \frac{\sum_{t=1}^{N}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{N}(y_t - \bar{y}_t)^2}, \tag{17}$$

where $\hat{y}_t$ is the predicted processing EC at time stamp $t$, $y_t$ is the actual processing EC at time stamp $t$, $\bar{y}_t$ is the average of the actual processing EC up to moment $t$, and $N$ is the total number of data points in the dataset.

**Step 5: Termination condition**. The evaluation of LSTM is to identify the population with the smallest RMSE (fitness value) as the potential solution. Therefore, $MAE$, $ME$, $RMSE$, and $R2$ were used to evaluate the performance of the double-layer LSTM.

**Step 6: Position update**. The traditional PSO algorithm uses a fixed inertia weight, which can easily fall into local extreme values, and affects the globality of the search and the speed of convergence [35]. We introduce an exponential function based on the bata

distribution to dynamically adjust the inertia weights for position updating in PSO. The calculation formula is shown in Equation (18). The 'Betarnd' is a random number that can be randomly generated to match the bata distribution, which can increase the ability of the algorithm to search globally and reduce the possibility of the algorithm falling into localization in the late PSO. The position of each population is updated using Equation (19). When $R \leq V_P$ is satisfied, random points (hyperparameters) are selected from the overall vector and the corresponding hyperparameter values are calculated using Equation (19) and Equation (20). During this procedure, the variance probability of the random number $R$ is $V_P$, expressed by Equation (21):

$$\omega = \omega_{\min}+(\omega_{\max}-\omega_{\min})*\exp(-t/t_{\max})+\sigma*\text{betarnd}(p,q), \quad (18)$$

$$Ppv_{i,t} = \omega Ppv_{i,t-1} + C_1R(p_{i,t-1} - X_{i,t-1}) + C_2R(g_{i,t-1} - X_{i,t-1}), \quad (19)$$

$$X_{i,t} = X_{i,t\text{-}1} + Ppv_{i,t}, \quad (20)$$

$$V_P = 1 - {}^{t}/_{t_{\max}}, \quad (21)$$

where $t$ is the current iteration number, $t_{\max}$ is the maximum iteration number, $\sigma$ is the inertia adjustment factor and is taken as 0.1; $\omega_{\max}$ and $\omega_{\min}$ are the initial inertia weight and the inertia weight at the maximum iteration number, respectively; $\omega_{\max}$ and $\omega_{\min}$ are generally taken as 1.2 and 0.8; $p$ is generally taken as 1, and $q$ is generally taken as 3; $R$ is a random number between [0, 1]; and C1 and C2 are acceleration constants, which are taken as 2 in this paper.

## 4. Case Study

### *4.1. Construction of Experimental Platform*

A real-time EC acquisition experimental platform was established, as shown in Figure 5. The experimental platform is implemented based on LabVIEW programming and Swallow simulation software. The platform consists of a photoelectric sensor, XK713 CNC milling machine, high-precision power tester, computer host, simulator, and communication terminal. The milling experiments were performed by selecting different combinations of parameters to machine different features and collecting energy consumption data, some of which are shown in Figure 6. The sampling frequency is set to 100 ms, and the real-time data are smoothed twice.

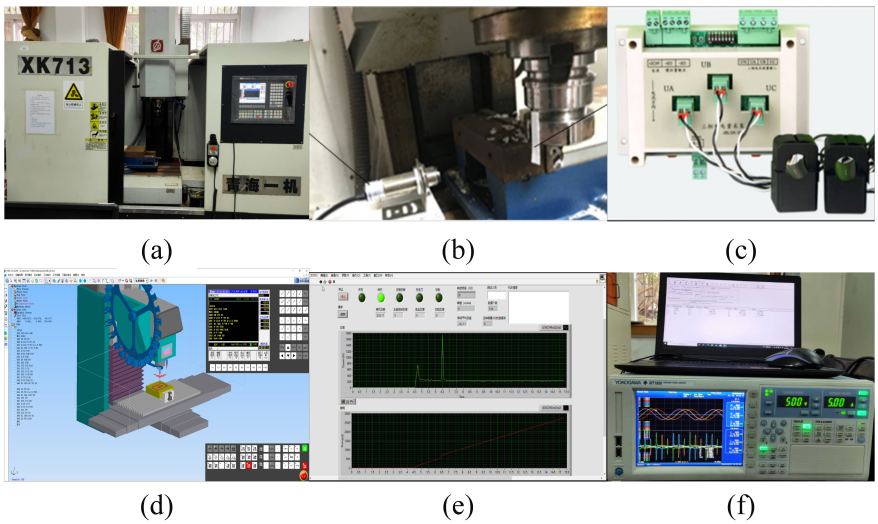


**Figure 5.** Milling machining energy efficiency prediction platform. (**a**) CNC milling machine; (**b**) photoelectric sensor; (**c**) Hall sensor; (**d**) CNC machining simulator; (**e**) real-time EC acquisition; (**f**) power tester and host computer.

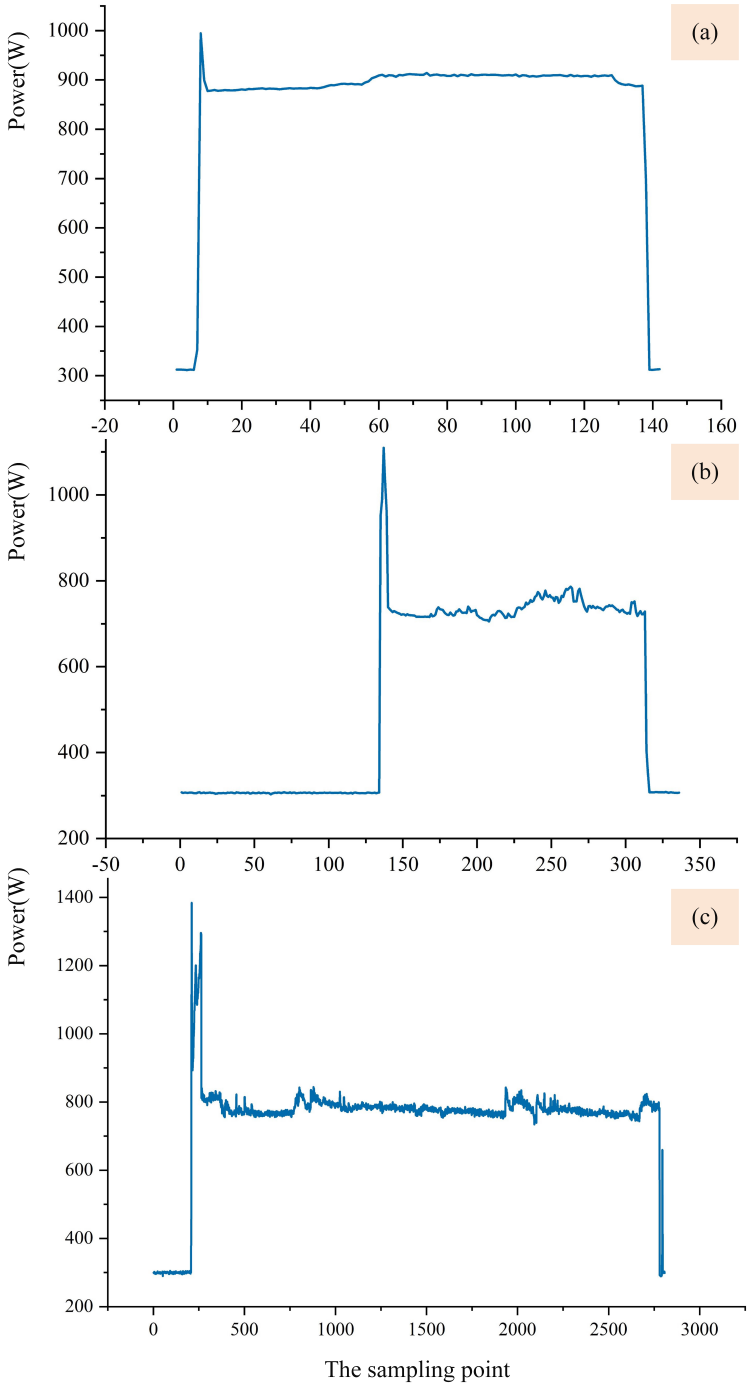


**Figure 6.** Acquisition of real-time power data for processing different features. (**a**) Real-time power of milling planes. (**b**) Real-time power for drilling. (**c**) Real-time power for milling slots.

### *4.2. Process EC Dataset Construction*

The construction of the dataset is crucial for the effect of EC prediction. Machine learning methods can be used to perform small sample sampling and prediction, but their accuracy cannot be guaranteed. Therefore, when the deep learning method is adopted, the amount of data required is relatively large. When a complete dataset is constructed, the prediction model of deep learning can ensure a better prediction effect. The method of constructing the processing energy dataset in this paper is as follows. Firstly, the WT1800 high-precision power tester was used to collect real-time EC data of the machining process; secondly, LabVIEW was used to program and deploy sensors to collect real-time machining EC data. The data collected by both methods are mixed 7:3 to build the updated EC dataset.

The specific data composition is shown in Table 1, and the sample dataset is shown in Table 2.

**Table 1.** Process energy dataset.

| | Drilling Data | Milling Plane Data | Milling Slot Data | Data Ratio |
|---|---|---|---|---|
| WT1800 PowerTester | 12,298 | 1936 | 10,587 | 70% |
| LabVIEW Programming | 1168 | 1072 | 1525 | 30% |
| Total data | 8959 | 1677 | 7868 | 100% |

**Table 2.** Sample dataset.

| Time | Millisecond | Current (A) | Voltage (V) | Power (W) | Spindle Speed (r/min) | Feed Speed (r/min) | Depth of Cut | Number of Blades | Energy Consumption (W.h) |
|---|---|---|---|---|---|---|---|---|---|
| 15:48:40 | 289 | 0.587 | 406.22 | 298.4 | 1200 | 2000 | 2.5 | 4 | 261.307 |
| 15:48:40 | 388 | 0.585 | 406.22 | 298.2 | 1200 | 2000 | 2.5 | 4 | 261.317 |
| 15:48:41 | 199 | 0.585 | 405.85 | 297.7 | 1200 | 2000 | 2.5 | 4 | 261.399 |
| 15:48:51 | 181 | 3.539 | 405.28 | 1213.7 | 1200 | 2000 | 2.5 | 4 | 264.948 |
| 16:07:15 | 81 | 2.002 | 404.86 | 760.9 | 1800 | 2500 | 2.5 | 4 | 456.939 |
| 16:07:16 | 76 | 1.978 | 404.94 | 752.4 | 1800 | 2500 | 2.5 | 4 | 457.16 |
| 16:12:14 | 93 | 2.935 | 404.52 | 1043.9 | 2000 | 2000 | 2.5 | 4 | 514.282 |
| 16:12:27 | 779 | 2.728 | 404.37 | 1015.7 | 2000 | 2000 | 2.5 | 4 | 518.659 |
| 16:12:28 | 84 | 2.714 | 404.25 | 1013.4 | 2000 | 2000 | 2.5 | 4 | 518.723 |
| 16:12:41 | 61 | 2.737 | 404.64 | 1032.9 | 2000 | 2000 | 2.5 | 4 | 522.913 |
| 16:13:03 | 79 | 2.814 | 403.53 | 1014 | 1000 | 1500 | 2.5 | 4 | 529.826 |
| 16:13:13 | 877 | 2.675 | 403.92 | 786 | 1000 | 1500 | 2.5 | 4 | 533.22 |
| 16:31:27 | 490 | 2.163 | 405.02 | 790 | 1000 | 1500 | 2.5 | 4 | 705.344 |
| 16:32:53 | 155 | 1.969 | 404.51 | 761.3 | 1000 | 1500 | 2.5 | 4 | 726.112 |
| 16:32:53 | 471 | 1.964 | 404.55 | 753.2 | 1000 | 1500 | 2.5 | 4 | 726.16 |
| 15:52:11 | 605 | 2.605 | 405.21 | 954.2 | 1200 | 2000 | 2.5 | 4 | 327.364 |
| 15:52:15 | 289 | 2.514 | 405.5 | 955.7 | 1200 | 2000 | 2.5 | 4 | 328.471 |
| 16:28:09 | 74 | 2.138 | 405.03 | 298.4 | 1200 | 2000 | 2.5 | 4 | 655.898 |
| 16:30:59 | 970 | 2.083 | 404.64 | 771.6 | 1000 | 1500 | 2.5 | 4 | 698.641 |
| 16:41:01 | 670 | 1.934 | 405.75 | 755.7 | 800 | 1200 | 2.5 | 4 | 825.252 |
| 16:42:06 | 180 | 0.602 | 405.06 | 299.9 | 800 | 1200 | 2.5 | 4 | 840.396 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

### *4.3. Model Parameter Settings*

The DIWPSO-LSTM model consists of an input layer, two LSTM layers, two dropout layers, a fully connected layer, and an output layer. The number of hidden layer neurons (Hun), learning rate (Lr), learning rate decline factor (Lrdf), and learning rate decline period (Lrdp) are set as the hyperparameters to be optimized, and the initial setting range of hyperparameters is shown in Table 3. Other parameters are set as follows: Maxepochs = 300, GradientThreshold = 1, InitialLearnRate = 0.005, and inertia weight = [0.8, 1.2]. The training process based on DIWPSO-LSTM is performed on the server with the following parameters: the GPU is NVIDIA 3090, CPU is i7-12700K, RAM 32G, Matlab 2020a.

**Table 3.** Hyperparameter setting and optimization results.

| Hyperparameters | Initial Range Setting | Optimization Results |
|---|---|---|
| hiddenUnit_num (Hun) | [90, 200] | 146 |
| LearningRate (Lr) | [0.001, 0.15] | 0.01211 |
| LearnRateDropFactor (Lrdf) | [0.01, 0.5] | 0.2 |
| LearnRateDropPeriod (Lrdp) | [80, 200] | 125 |

### *4.4. Results and Discussion*

The proposed DIWPSO-LSTM method is compared with the conventional PSO-LSTM, LSTM, and other improved PSO methods, such as IPSO-LSTM and ACMPSO-LSTM; the predicted experimental comparison results are shown in Figure 7, and Table 4 shows the numerical results. The training, validation, and prediction results of EC data for milling slot machining using DIWPSO-LSTM are shown in Figures 8–10.

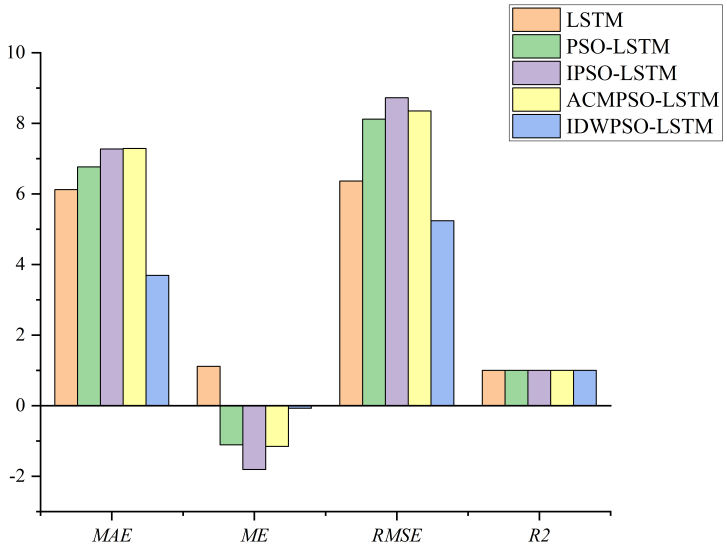


**Figure 7.** Comparison of the results of the proposed method with other methods.

**Table 4.** Values of predicted results by different methods.

| | *MAE* | *ME* | *RMSE* | *R2* |
|---|---|---|---|---|
| LSTM | 6.12214 | 1.11289 | 6.36465 | 0.99882 |
| PSO-LSTM | 6.76289 | −1.11015 | 8.11697 | 0.99885 |
| IPSO-LSTM | 7.27200 | −1.81000 | 8.72457 | 0.99830 |
| ACMPSO-LSTM | 7.28726 | −1.15409 | 8.35037 | 0.99863 |
| **DIWPSO-LSTM** | **3.68966** | **−0.07248** | **5.23745** | **0.99891** |

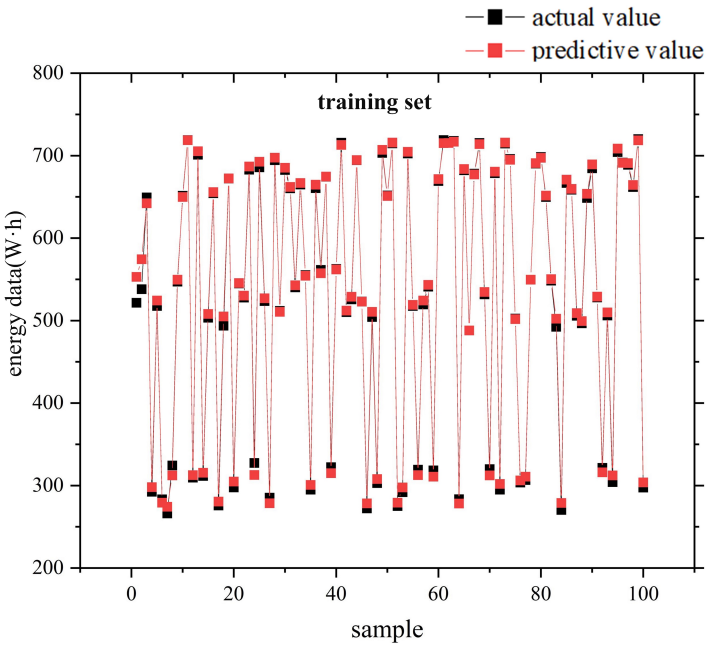


**Figure 8.** Real-time machining energy training for milling slots using DIWPSO-LSTM.

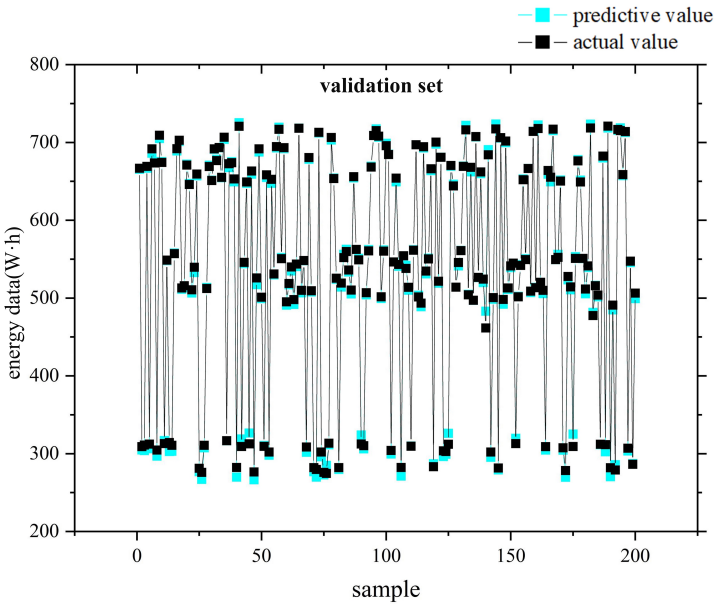


**Figure 9.** Real-time machining energy verification of milling slots using DIWPSO-LSTM.

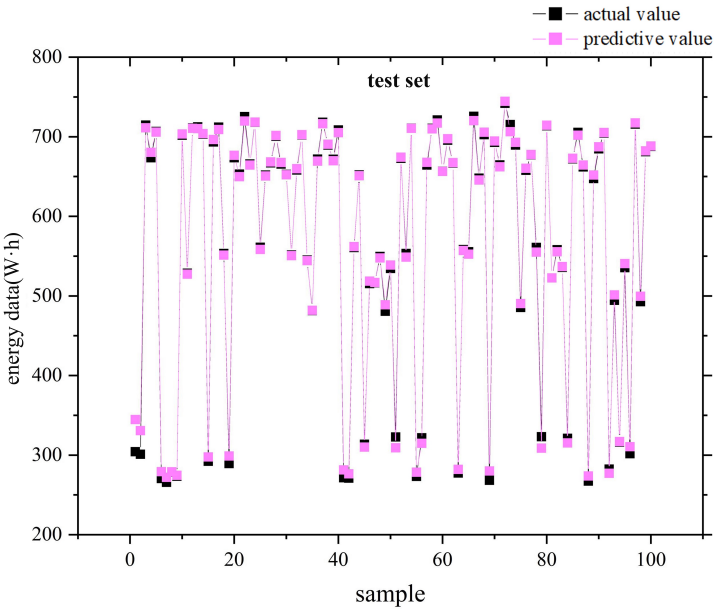


**Figure 10.** Real-time machining energy prediction for milling slots using DIWPSO-LSTM.

The mean errors of the five methods were 1.113, −1.110, −1.810, −1.154, and −0.0725. It is shown that the DIWPSO-LSTM method proposed in this paper is superior to other methods in terms of *ME* value. The *MAE* of the DIWPSO-LSTM method also improved by 39.73%, 45.44%, 49.26%, and 49.36% compared to other methods. The *RMSE* of the DIWPSO-LSTM method improved by 17.70%, 35.47%, 36.96%, and 37.27% compared to other methods. Through experimental comparison, it is found that the double-layer LSTM prediction model works much better than other improved methods.

Compared with PSO-LSTM, IPSO-LSTM, and ACMPSO-LSTM methods, the iterative search time of the recommended method is shorter by 12.05%, 12.96%, and 18.2%, respectively. When using a single-layer LSTM, the training time is not much different for either method. When using the double-layer LSTM model for prediction, the training time tends to be longer, and the comparison of the specific training time is shown in Table 5. The training time for the double-layer LSTM model is twice as long as the other methods, which also indicates that prediction accuracy is improved by increasing the training time to

achieve the prediction effect, which is acceptable. If the number of layers of LSTM increases, the training time is prolonged, and overfitting will also occur. Therefore, for this paper, choosing a double-layer LSTM is the most appropriate choice.

**Table 5.** Comparison of iterative optimization time and training time results of different methods.

| | Optimization Time (Second) | Inertia Weight | Training Time (Second) | Network Layers Numbers |
|---|---|---|---|---|
| PSO-LSTM | 371.0 | 0.80000 | 30.00000 | 4 |
| IPSO-LSTM | 374.0 | 0.80000 | 32.40000 | 4 |
| ACMPSO-LSTM | 391.3 | 0.77000 | 33.90000 | 4 |
| **DIWPSO-LSTM** | **331.1** | **0.97187** | **63.30000** | **7** |

## 5. Conclusions

The IPSO algorithm using dynamic inertial weights (DIWPSO) is proposed to optimize the hyperparameters of LSTM networks to improve the prediction accuracy of LSTM neural networks. The comparison with LSTM, PSO-LSTM, IPSO-LSTM, and ACMPSO-LSTM models shows that the proposed method (DIWPSO-LSTM) has higher prediction accuracy, reduces overfitting, and shortens iterative optimization time. Furthermore, the proposed method can also be applied to the EC prediction of machining systems in a wide range of applications.

A method of constructing a machining EC prediction dataset is proposed. This method segments and fuses the machining EC dataset by different acquisition methods. The generalization performance of the model can be improved by using the fused dataset as a basis for EC prediction.

The evaluation index system of the machining EC prediction results is supplemented with the evaluation of the coefficient of determination (goodness of fit). As a result, the results of the EC prediction for machining systems are more accurate and reliable. However, there are some shortcomings in this paper, mainly that the machined blank parts are simple parts. Next, we will explore the use of a double-layer LSTM prediction model to handle the problem of machining EC prediction for complex parts. Is it possible to use other deep learning methods, such as the transformer or informer model, with higher prediction accuracy and shorter training time? In addition, for the construction of processing EC datasets, the next step is to consider utilizing simulation software to simulate the processing process. This will enable the simulation to generate a large amount of data, which can reduce the time of experimental sampling and improve efficiency.

**Author Contributions:** Conceptualization, M.Z. and W.Y.; methodology, M.Z., H.Z. and W.Y.; software, M.Z. and H.Z.; validation, M.Z.; formal analysis, H.Z. and W.Y.; resources, H.Z., Z.J. and S.Z.; data curation, M.Z.; writing—original draft preparation, M.Z.; writing—review and editing, M.Z. and W.Y.; supervision, W.Y. and H.Z.; project administration, H.Z. and Z.J.; financial support, W.Y. and H.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China grant number 51975432, 51905392, 52075396.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Qi, R.; Shi, C.; Wang, M.Y. Carbon emission rush in response to the carbon reduction policy in China. *China Inf.* **2022**, *37*, 0920203X221093188. [CrossRef]
2. Tracker, C.A. To Show Climate Leadership, US 2030 Target Should Be at Least 57%—63%, 2021. Available online: https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwjixt7ly_P9AhWTbN4KHXVaCioQFnoECA0QAw&url=https%3A%2F%2Fclimateactiontracker.org%2Fdocuments%2F846%2F2021_03_CAT_1.5C-consistent_US_NDC.pdf&usg=AOvVaw1I1hcAtVP-mj_lZUkqOp4P (accessed on 8 February 2023).
3. Tsiropoulos, I.; Nijs, W.; Tarvydas, D.; Ruiz, P. Towards Net-Zero Emissions in the EU Energy System by 2050. Insights from Scenarios in Line with the 2020. Available online: https://publications.jrc.ec.europa.eu/repository/handle/JRC118592 (accessed on 8 February 2023).
4. González-Torres, M.; Pérez-Lombard, L.; Coronel, J.F.; Maestre, I.R.; Yan, D. A review on buildings energy information: Trends, end-uses, fuels and drivers. *Energy Rep.* **2022**, *8*, 626–637. [CrossRef]
5. Gadaleta, M.; Pellicciari, M.; Berselli, G. Optimization of the energy consumption of industrial robots for automatic code generation. *Robot. Comput.-Integr. Manuf.* **2019**, *57*, 452–464. [CrossRef]
6. Giampieri, A.; Ling-Chin, J.; Ma, Z.; Smallbone, A.; Roskilly, A. A review of the current automotive manufacturing practice from an energy perspective. *Appl. Energy* **2020**, *261*, 114074. [CrossRef]
7. Moradnazhad, M.; Unver, H.O. Energy efficiency of machining operations: A review. *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **2017**, *231*, 1871–1889. [CrossRef]
8. Liu, N.; Zhang, Y.; Lu, W. A hybrid approach to energy consumption modelling based on cutting power: A milling case. *J. Clean. Prod.* **2015**, *104*, 264–272. [CrossRef]
9. Shi, K.; Ren, J.; Wang, S.; Liu, N.; Liu, Z.; Zhang, D.; Lu, W. An improved cutting power-based model for evaluating total energy consumption in general end milling process. *J. Clean. Prod.* **2019**, *231*, 1330–1341. [CrossRef]
10. Wang, L.; Meng, Y.; Ji, W.; Liu, X. Cutting energy consumption modelling for prismatic machining features. *Int. J. Adv. Manuf. Technol.* **2019**, *103*, 1657–1667. [CrossRef]
11. He, Y.; Wang, L.; Wang, Y.; Li, Y.; Wang, S.; Wang, Y.; Liu, C.; Hao, C. An analytical model for predicting specific cutting energy in whirling milling process. *J. Clean. Prod.* **2019**, *240*, 118181. [CrossRef]
12. Lv, J.; Tang, R.; Tang, W.; Liu, Y.; Zhang, Y.; Jia, S. An investigation into reducing the spindle acceleration energy consumption of machine tools. *J. Clean. Prod.* **2017**, *143*, 794–803. [CrossRef]
13. Yuan, X.; Tan, Q.; Lei, X.; Yuan, Y.; Wu, X. Wind power prediction using hybrid autoregressive fractionally integrated moving average and least square support vector machine. *Energy* **2017**, *129*, 122–137. [CrossRef]
14. Hu, L.; Peng, C.; Evans, S.; Peng, T.; Liu, Y.; Tang, R.; Tiwari, A. Minimising the machining energy consumption of a machine tool by sequencing the features of a part. *Energy* **2017**, *121*, 292–305. [CrossRef]
15. Brillinger, M.; Wuwer, M.; Hadi, M.A.; Haas, F. Energy prediction for CNC machining with machine learning. *CIRP J. Manuf. Sci. Technol.* **2021**, *35*, 715–723. [CrossRef]
16. LI, C.; Yin, Y.; Xiao, Q.; Long, Y.; Zhao, X. Data-driven Energy Consumption Prediction Method of CNC Turning Based on Meta-action. *China Mech. Eng.* **2020**, *31*, 2601.
17. Qin, J.; Liu, Y.; Grosvenor, R. Multi-source data analytics for AM energy consumption prediction. *Adv. Eng. Inform.* **2018**, *38*, 840–850. [CrossRef]
18. Liu, Z.; Guo, Y. A hybrid approach to integrate machine learning and process mechanics for the prediction of specific cutting energy. *CIRP Ann.* **2018**, *67*, 57–60. [CrossRef]
19. Kim, Y.M.; Shin, S.J.; Cho, H.W. Predictive modeling for machining power based on multi-source transfer learning in metal cutting. *Int. J. Precis. Eng. Manuf. Green Technol.* **2022**, *9*, 107–125. [CrossRef]
20. He, Y.; Wu, P.; Li, Y.; Wang, Y.; Tao, F.; Wang, Y. A generic energy prediction model of machine tools using deep learning algorithms. *Appl. Energy* **2020**, *275*, 115402. [CrossRef]
21. Kahraman, A.; Kantardzic, M.; Kahraman, M.M.; Kotan, M. A data-driven multi-regime approach for predicting energy consumption. *Energies* **2021**, *14*, 6763. [CrossRef]
22. Karijadi, I.; Chou, S.Y. A hybrid RF-LSTM based on CEEMDAN for improving the accuracy of building energy consumption prediction. *Energy Build.* **2022**, *259*, 111908. [CrossRef]
23. Somu, N.; MR, G.R.; Ramamritham, K. A hybrid model for building energy consumption forecasting using long short term memory networks. *Appl. Energy* **2020**, *261*, 114131. [CrossRef]
24. Luo, X.; Oyedele, L.O. Forecasting building energy consumption: Adaptive long-short term memory neural networks driven by genetic algorithm. *Adv. Eng. Inform.* **2021**, *50*, 101357. [CrossRef]
25. Zhou, X.; Lin, W.; Kumar, R.; Cui, P.; Ma, Z. A data-driven strategy using long short term memory models and reinforcement learning to predict building electricity consumption. *Appl. Energy* **2022**, *306*, 118078. [CrossRef]
26. Ullah, F.U.M.; Ullah, A.; Haq, I.U.; Rho, S.; Baik, S.W. Short-term prediction of residential power energy consumption via CNN and multi-layer bi-directional LSTM networks. *IEEE Access* **2019**, *8*, 123369–123380. [CrossRef]
27. Salam, A.; El Hibaoui, A. Energy consumption prediction model with deep inception residual network inspiration and LSTM. *Math. Comput. Simul.* **2021**, *190*, 97–109. [CrossRef]

28. Fan, S. Research on deep learning energy consumption prediction based on generating confrontation network. *IEEE Access* **2019**, *7*, 165143–165154. [CrossRef]
29. Xiao, Q.; Li, C.; Tang, Y.; Chen, X. Energy efficiency modeling for configuration-dependent machining via machine learning: A comparative study. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 717–730. [CrossRef]
30. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
31. Moreno, S.R.; da Silva, R.G.; Mariani, V.C.; dos Santos Coelho, L. Multi-step wind speed forecasting based on hybrid multi-stage decomposition model and long short-term memory neural network. *Energy Convers. Manag.* **2020**, *213*, 112869. [CrossRef]
32. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep learning for smart manufacturing: Methods and applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
33. Marini, F.; Walczak, B. Particle swarm optimization (PSO). A tutorial. *Chemom. Intell. Lab. Syst.* **2015**, *149*, 153–165. [CrossRef]
34. Brand, M. Incremental singular value decomposition of uncertain data with missing values. In Proceedings of the European Conference on Computer Vision, Copenhagen, Denmark, 28–31 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 707–720.
35. Bai, Q. Analysis of particle swarm optimization algorithm. *Comput. Inf. Sci.* **2010**, *3*, 180. [CrossRef]