applied sciences

MDPI

Article

Analysis of Efficient and Fast Prediction Method for the Kinematics Solution of the Steel Bar Grinding Robot

Wei Shi [1,2,3], Jinzhu Zhang [1,2,3,*], Lina Li [1], Ziliang Li [1,2,3], Yanjie Zhang [1,2,3], Xiaoyan Xiong [1], Tao Wang [1,2,3] and Qingxue Huang [1,2,3]

1 College of Mechanical and Vehicle Engineering, Taiyuan University of Technology, Taiyuan 030024, China
2 Engineering Research Center of Advanced Metal Composites Forming Technology and Equipment, Ministry of Education, Taiyuan University of Technology, Taiyuan 030024, China
3 Institute of Advanced Forming and Intelligent Equipment, Taiyuan University of Technology, Taiyuan 030024, China
* Correspondence: zhangjinzhu@tyut.edu.cn

Abstract: Aiming at the robotization of the grinding process in the steel bar finishing process, the steel bar grinding robot can achieve the goal of fast, efficient, and accurate online grinding operation, a multi-layer forward propagating deep neural network (DNN) method is proposed to efficiently predict the kinematic solution of grinding robot. The process and kinematics model of the grinding robot are introduced. Based on the proposed method, simulations of the end position and orientation, and joint angle of the grinding robot are given. Three different methods, including SGD + tanh, Nadam + tanh, Nadam + ELU, are used to test the DNN calculation process results show that the method combining Nadam with ELU function has the fastest solution speed and higher accuracy can be obtained with the increase in iteration times. Finally, the Nadam optimizer is used to optimize the calculation results of the example. The optimization results show that this method accelerates the convergence rate of trajectory prediction error and improves the accuracy of trajectory prediction. Thus, the proposed method in this paper is an effective method to predict the kinematic solution when the grinding robot works online.

Keywords: grinding robot; kinematics; deep neural network; efficient prediction; precision

check for updates

Citation: Shi, W.; Zhang, J.; Li, L.; Li, Z.; Zhang, Y.; Xiong, X.; Wang, T.; Huang, Q. Analysis of Efficient and Fast Prediction Method for the Kinematics Solution of the Steel Bar Grinding Robot. *Appl. Sci.* **2023**, *13*, 1212. https://doi.org/10.3390/app13021212

Academic Editor: Dimitris Mourtzis

Received: 16 December 2022
Revised: 13 January 2023
Accepted: 13 January 2023
Published: 16 January 2023

1. Introduction

Modern society is featured with the rapid advance of industrialization, the replacement of manual labor by robots is accelerated due to the existence of heavy manual labor and potential safety risks. More and more manipulators have assumed the responsibility of assisting or even replacing human beings to complete tasks in industrial automation, equipment manufacturing, medical assistance, aerospace, etc. [1,2].

As we all know, the steel industry is one of the process industries with a high degree of automation. However, there are some problems in the finishing area of the special steel bar, such as frequent operation, harsh environment, heavy labor, and high safety risk, which have been a great threat and hidden danger to the life safety of workers for a long time. In view of this, the traditional steel industry urgently needs to be promoted to robotic operation, in order to meet the requirements of unmanned positions and finishing processes of high quality and efficiency. The bar finishing process consists of straightening, shot blasting, chamfering, polishing, and other processes (the finishing process is shown in Figure 1), grinding as one of the most important processes is given more and more attention by producers. In order to achieve the goal of fast, efficient, and accurate online grinding operation of the grinding robot, it is very necessary to efficiently predict the kinematic solution of the grinding robot before working [3–5].

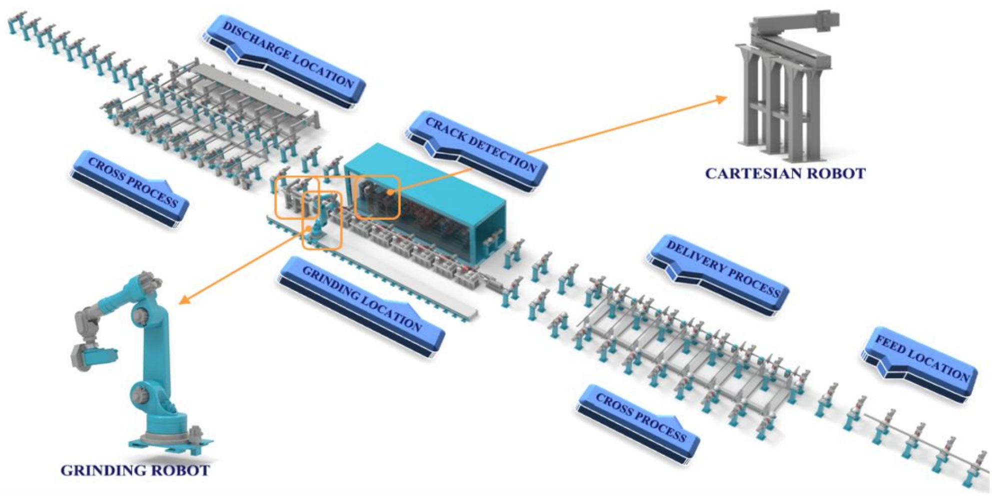

Figure 1. The steel bar finishing process.

Kinematics analysis of robots is an important field of robot studies. Kinematics analysis of grinding robots (6-DOF articulated industrial robots) is actually the study of the mapping relationship of position vector and orientation vector between the joint space of the manipulator and Cartesian coordinate space [6–8]. Kinematics analysis can provide some reference value for the motion control of the robot. In reference [9], the forward kinematics of the 6-DOF parallel robot was solved based on BP neural network, and the nonlinear vector mapping from joint space to workspace was realized by the L-M method. In reference [10], linear interpolation and circular interpolation were used to optimize the spatial interpolation points, and the trajectory optimization was solved. In reference [11], the BP and RBF neural networks were applied to solve the inverse kinematics of the 6-DOF manipulator, and the proposed algorithm was faster than the traditional method, with less computation time. In reference [12], a sliding mode trajectory tracking control method based on a convolution neural network was proposed, the convergence rate of trajectory tracking error was accelerated, and the trajectory tracking accuracy was effectively improved. In reference [13], a scale optimization GEMM optimization method was proposed to further optimize convolution. In reference [14], a special acceleration structure for graph neural network application is proposed. In reference [15], a robust zeroing neural network (RZNN) model for real-time kinematic analysis of manipulators was proposed. In reference [16], a new motion controller based on a recursive neural network (RNN) was proposed, which had good model adaptive ability. In reference [17], a neural network method for residual positioning and positioning error compensation was proposed. In reference [18], a visual analysis tool for solving forward and inverse kinematics of 7-DOF robots based on artificial neural networks was developed. In reference [19], an artificial neural network (ANN) approach to the reliable calculation of direct kinematics problem (DKP) was presented. In reference [20], BNN is proposed to establish a highly nonlinear kinematic and dynamic model of a tendon-driven surgical robot to ensure the accuracy and safety of robotic surgery. In reference [21], two novel loss functions are proposed to train the feed-forward artificial neural network (ANN), and the differential relationship between position and velocity mapping is incorporated into the forward kinematics model of the robot structure, the study shows that the introduction of velocity mapping can improve the adaptability of the learning model to the control task. In reference [22], a novel feed-forward artificial neural network (ANN) structure is proposed to learn the complete pose of the robot in SE (3), and the difference relation is introduced into the learning process. The research shows that the proposed method can correctly model the pose of the robot. In reference [23], the authors do not consider reinforcement learning or trial and error methods. Reinforcement learning is to apply these mapped actions, get a series of feedback reward values, and then select the action with the largest reward value. When the degree of freedom of the robot increases, the convergence performance of the controller will be significantly worse. As you get more and more degrees of freedom, it takes longer and longer to converge. In reference [24], the transfer learning method was applied to train neural networks, which greatly reduced the number of training images required. In reference [25], a new robust return to zero neural networks (RZNN) model was proposed to study the timing trajectory tracking control of wheeled mobile robots in a noise pollution environment. Neural networks have many other applications as well. In reference [26], The finite-time stabilization problem of complex-valued neural networks with proportional delays and inertia terms is studied. In reference [27], an improved convolutional neural network (CNN) was developed for fault feature extraction and classification of hydraulic piston pumps. The DNN method proposed in this paper is a deep learning method included in ANN. In this paper, literature [19] is introduced to better understand the feasibility of the DNN method proposed. DNN is more capable of modeling or abstracting things and can simulate more complex models. The problem of inverse kinematics of grinding robots is characterized by high computational complexity and time consumption. Hence, it is necessary to introduce DNN into the inverse kinematics of grinding robots.

In order to more effectively control the trajectory of the grinding robot and correctly solve the forward and inverse kinematics parameters of the robot, and to avoid the complicated process of solving kinematics equations and the un-uniqueness of inverse kinematics solutions, to achieve fast and efficient online grinding operation of bar grinding robot, in this paper a method based on the multi-layer forward propagation deep neural network is proposed. The proposed method is used to solve the forward and inverse kinematics and realize the kinematics calculation and is compared with the traditional analytical calculation method. It is demonstrated that not only the complicated mathematical theoretical derivation calculation process can be avoided, but results with higher accuracies can also be achieved, it can achieve the goal of fast, efficient, and accurate online grinding operation of bar grinding robot.

This paper takes KUKA KR series industrial robot (grinding robot) as the research object, the main structure of the article is arranged as follows: in Section 2, the basic concepts and related theories of the grinding robot process and kinematics are introduced. A strategy for solving grinding robot kinematics equations based on a multi-layer forward propagation deep neural network is proposed in Section 3. In Section 4, the experimental scheme is stated, and the main simulation experiments and optimization are described in detail. In Section 5 some conclusions are drawn.

2. Grinding Robot Process Introduction and Kinematics Model

2.1. Grinding Robot Process Introduction

Finishing is the last process of quality control of special steel products, and it is animportant means to ensure the quality of steel products, improve the grade of products and create fine products. It is through the special steel products shot blasting, straightening, chamfering, flaw detection, grinding, labeling, baling, weighing, heat treatment, surface peeling and other processing, to eliminate surface and internal defects of the special steel bar, so that special steel products can meet the factory standards and user requirements, but also can greatly improve the added value of products. The finishing process includes online finishing and offline finishing. The online finishing process is shown in Figure 2.

Online finishing generally refers to the product in the rolling line completing the relevant finishing process. The offline finishing is independent of the rolling production line, which is a process route for the advanced finishing treatment of steel. There are different offline processing processes according to different steel products. As an important part of the finishing process, grinding is paid more and more attention by scientific and technical workers. Grinding robots are mainly used for polishing sanitary porcelain, tables, chairs, and body and deburring castings, these are floating flexible polish. Special steel bar grinding belongs to rigid grinding, grinding material is hard, compared with manual grinding existing problems of uncontrollable depth, and surface quality is not high, it is necessary to realize the special steel bar grinding robotization. Therefore, it is necessary to efficiently predict the kinematic solution of the grinding robot before the online grinding of the special steel bar.

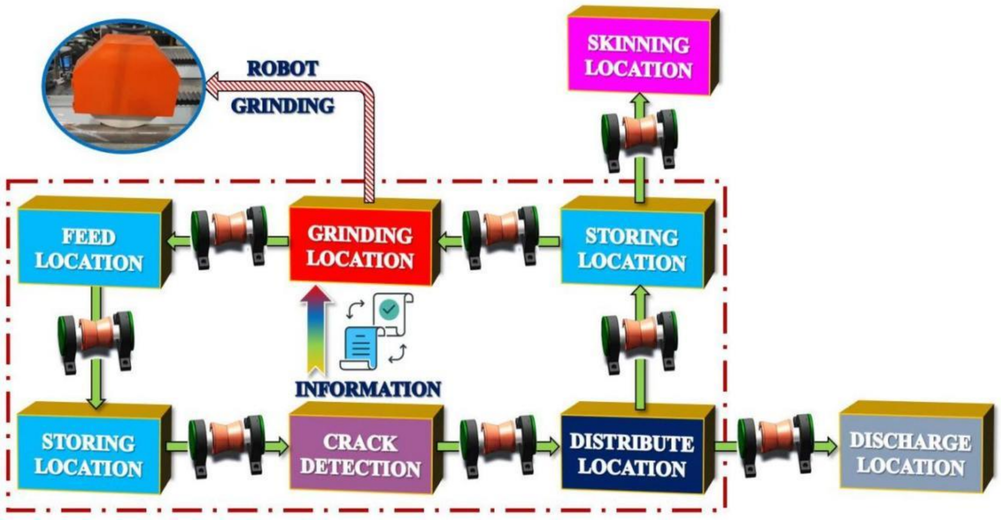

Figure 2. Online finishing process of the steel bar.

Before the grinding robot is used to grind the defects of the special steel bar, the robot grinding system needs to recognize the bar defect data provided by the crack detection device, then the program in the robot database is called to command the grinding robot to realize the grinding of bar defects. According to the actual working conditions provided parameters, the parameters of bar defect are depth greater than 0.2 mm and length greater than 0.5 mm. The defect location and sample diagram of the bar are shown in Figure 3. According to the requirements of the grinding compliance rate of the production line, the compliance rate of the bar after the robot grinding should be greater than or equal to 95%. The actual transfer speed of the special steel bar production line is 1~2 m per s.

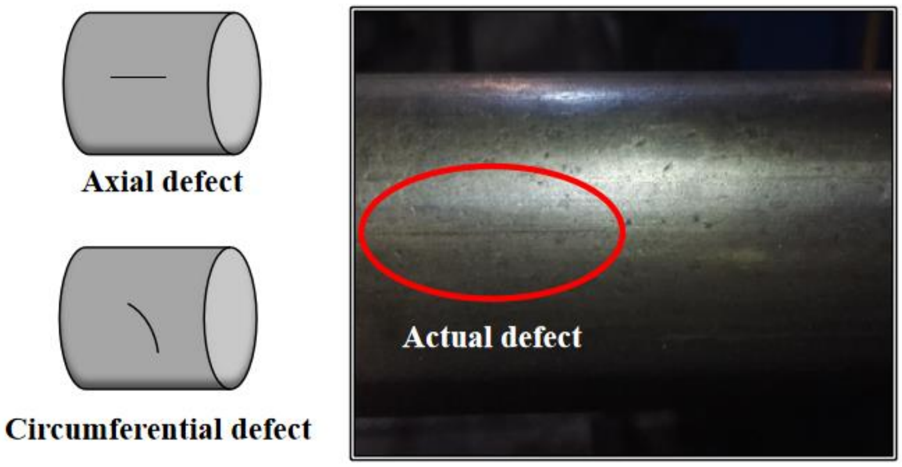

Figure 3. The defect location and sample diagram of the bar.

Special steel bar grinding belongs to rigid grinding, grinding material is hard, compared with the traditional manual grinding, there are problems of uncontrollable depth, and surface quality is not high. The kinematics control of the robot must be considered to realize the robotic operation of the grinding process. The efficiency and accuracy of the robot grinding steel bar depend on whether the robot can accurately reach the target grinding point. The motion trajectory and drive control of the robot have a great influence on the grinding, so it is necessary to effectively predict the robot's kinematic solution in advance. In the process of robot grinding, there will be kinematic problems such as rotation of each joint, trajectory planning, position, and attitude of the end-effector. Considering the timeliness of robot motion control by the robot grinding strategy program, it is necessary and effective to predict the robot kinematic solution to improve the grinding efficiency and precision. In order to accurately control the robot to reach the grinding target point, it must be able to accurately predict the actual state displacement of the robot in the moving space.

In this paper, the deep neural network proposed instead of the traditional D-H kinematics calculation method is aimed at the specific practical application field of robot grinding special steel bar defects. When the robot is used to grind the steel bar, the grinding wheel installed at the end of the robot will wear off its own abrasive particles with friction contact with the steel bar, resulting in the reduction of the diameter of the grinding wheel. Therefore, the diameter change in the grinding wheel becomes an uncertain parameter in the actual robot grinding process. It is well known that the traditional D-H kinematics calculation method can accurately obtain the parameter information of robot kinematics. However, in view of the fact that the uncertainty caused by the change in grinding wheel diameter in the actual grinding process will affect the efficiency and accuracy of robot grinding. In this paper, the deep neural network is proposed to predict the robot kinematics in advance, which can effectively avoid the above problems. The robot kinematics solution of was predicted to eliminate the uncertainty of grinding caused by the change of grinding wheel diameter in the actual grinding process. Some compensation and correction are given to the robot before grinding so that the robot can adjust in real time when grinding the steel bar, so as to complete the grinding task more efficiently and accurately.

As mentioned above, when the grinding wheel is grinding the steel bar, the abrasive particles will be worn off and the diameter of the grinding wheel will become smaller. Therefore, the uncertainty caused by the change of grinding wheel diameter becomes a factor affecting the efficiency and precision of robot grinding. The reduction of grinding precision and the change of surface morphology of steel bar defects, which leads to the secondary inspection of the steel bar. As shown in Figure 2, bar finishing process flow chart. It can be seen that if secondary inspection of steel bar defects is carried out, the "detecting-distributing-grinding" cycle process will be re-implemented. As a result, steel bar accumulation occurred in the storage area, and the efficiency of the whole finishing operation will be significantly reduced, thus greatly reducing the overall grinding efficiency. Therefore, it is necessary to predict the robot kinematics of the steel bar before the robot grinding, which can fully ensure the efficiency and accuracy of the robot grinding.

To sum up, it is necessary to effectively predict the kinematic solution of the grinding robot before the grinding robot is used to repair the defects of the special steel bar, in order to achieve the fast, efficient and accurate online grinding operation goal of the grinding robot.

2.2. Grinding Robot Kinematics Model

This paper takes the KUKA KR series robot (grinding robot) as the research object, which is composed of a base frame, rotating column, link arm, in-line wrist, arm, and other parts. The main purpose is to install a grinding wheel at the end of the robot manipulator to grind the special steel bar with surface defects, the existing robot manipulator, and the steel bar grinding system (the work of this paper is to provide motion support for the realization of robot grinding steel bar) in the laboratory are shown in Figure 4.

The essence of the kinematic analysis of the grinding robot is the process of matrix transformation between two adjacent coordinate systems, and then the transformed matrices are multiplied successively, and finally, the kinematic equation of the robot is obtained. The solution of the kinematics equation consists of two parts, forward kinematics solution, and inverse kinematics solution. The forward kinematics solution is based on the known angle of each joint variable to solve the position and orientation of the end-effector. The inverse kinematics is based on the known position and orientation of the end-effector to solve the angle of each joint variable [28]. The specific solution process is described in detail in the literature [8], which will not be described here. In order to obtain higher calculation accuracy and simplify the calculation process, a method based on a multi-layer forward propagation deep neural network was proposed to solve the kinematics equation of the robot.

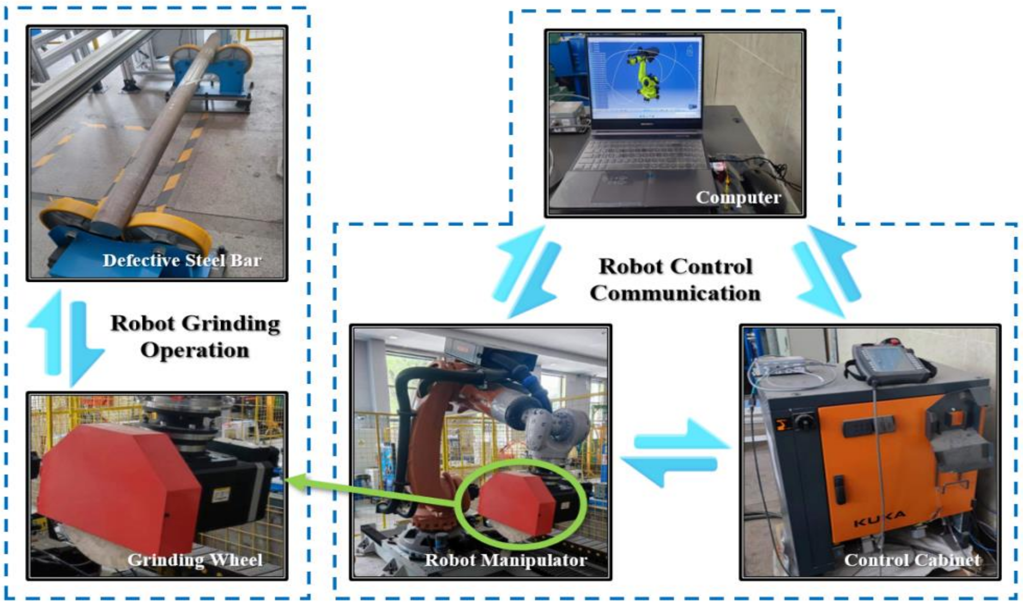

Figure 4. Robot manipulator and grinding system.

According to D-H model parameter method, the coordinate system of each link of grinding robot is established, as shown in Figure 5. The grinding robot is revolute joint, the end is connected with an actuator. No matter how the revolute joint 0 connecting the base frame and the rotating column rotates, its length has no influence on the movement of grinding robot, so the reference coordinate system 0 and coordinate system 1 are coaxial. According to the right-hand rule, the coordinate axes direction of remaining joints are determined [28,29].

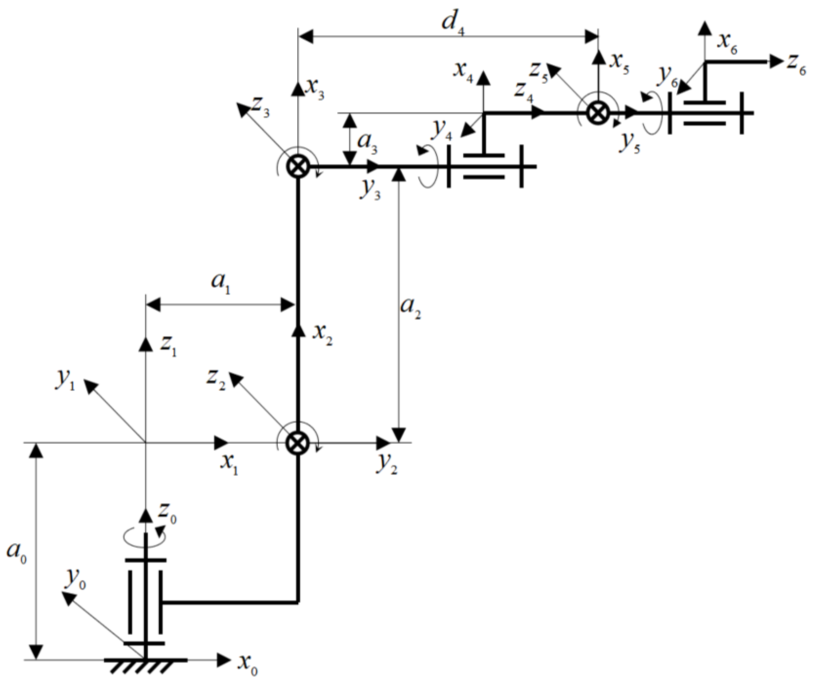

Figure 5. Schematic diagram of manipulator coordinate system.

When the coordinate system of grinding robot manipulator is established, the corresponding kinematics analysis and calculation can be carried out according to the given parameters of each link of grinding robot. The variation range of D-H model parameters and related variables of each link is shown in Table 1.

Table 1. Basic parameters of grinding robot.

$\mathbf{i}$	$\mathbf{a_{i-1}}$	$\boldsymbol{\alpha_{i-1}}$	$\mathbf{d_i}$	$\boldsymbol{\theta_i}$	$\boldsymbol{\theta_i}$ **Scope**
1	0	0	0	θ_1	+/−185°
2	a_1	−90°	0	θ_2	−5~−140°
3	a_2	0	0	θ_3	+155~−120°
4	a_3	−90°	d_4	θ_4	+/−350°
5	0	90°	0	θ_5	+/−122.5°
6	0	−90°	0	θ_6	+/−350°

Where a_{i-1} is the length of the link and represents the common perpendicular of two adjacent joints axes connecting link $i-1$, α_{i-1} is the angle of the link and represents the angle between two adjacent joints, d_i is the offset of the link and represents the distance from the point where a_{i-1} intersects axis i to the point where a_i intersects axis i, θ_i is the angular displacement between adjacent joints and represents the angle between the extension of a_{i-1} and the extension of a_i.

The transformation matrix ${}^{i-1}_{i}T$ of coordinate system $\{i\}$ with respect to coordinate system $\{i-1\}$ is as follows.

$$ {}^{i-1}_{i}T = Rot(x,\alpha_{i-1})Trans(x,a_{i-1})Rot(z,\theta_i)Trans(z,d_i) \tag{1}$$

$$ {}^{i-1}_{i}T = Screw(x,a_{i-1},\alpha_{i-1})Screw(z,d_i,\theta_i) \tag{2}$$

where $Screw(L,r,\varphi)$ is translating r along axis L and rotating φ around axis L.

The general expression of transformation matrix ${}^{i-1}_{i}T$ between links is as follows.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The position vector P represents the position of the terminal link, the rotation matrix $R = \begin{bmatrix} n & o & a \end{bmatrix}$ represents the orientation of the terminal link. The kinematics equation of manipulator is as follows.

$$ \begin{bmatrix} {}^{0}_{n}R & {}^{0}_{n}p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{0}_{1}T(\theta_1){}^{1}_{2}T(\theta_2)\ldots{}^{\mathrm{n}-1}_{n}T(\theta_n) \tag{4}$$

where the equation represents the relationship between the position and orientation of the terminal link (n,o,a,p) and the joint variables $\theta_1,\theta_2,\cdots,\theta_n$.

The robot kinematics model is the core technology of robot application, especially the inverse kinematics model of series robots and the forward kinematics model of parallel robots. The solution process is complicated, and most of them even do not have analytical solutions. All these problems bring difficulties for the application of robots. The solution of the robot kinematics model based on a neural network is the process of generating data fitting the forward kinematics model according to the inverse kinematics model, or generating data fitting the inverse kinematics model according to the forward kinematics model. When solving the kinematics of the 6-DOF robot, selecting solutions and missing solutions to problems will arise. Deep neural networks can deal with many complex underfitting problems. It is possible to solve some complex problems by increasing the depth of the network and improving learning ability. In view of the deep learning algorithm has the advantages of automatic learning, automatic optimization, and incremental learning of advanced functions from data. Therefore, the deep neural net is introduced in this paper to solve and predict the kinematics model of the robot.

The fast prediction method proposed in this paper is different from the traditional robot kinematics analysis method. The deep neural network is applied to solve the robot kinematics, and a large amount of data drive and mapping relationship between parameters are used to solve and analyze the robot kinematics, so as to replace the traditional analytical method to realize the application of the robot.

3. DNN Based Grinding Robot Kinematics Solution

3.1. DNN Overview

Deep neural network (DNN) is a technique in the field of machine learning, it refers to a neural network containing multiple hidden layers. According to the position of different levels, the neural network layer inside DNN can be divided into three categories, including the input layer, hidden layer, and output layer, and the layers are fully connected [30,31]. By using DNN to make a prediction, the accuracy will get better as the number of training sessions increases. The application of a deep learning model to grinding robots for trajectory tracking and predictive control is the current research direction of the industry [32,33].

3.2. DNN Model

In this paper, a deep neural network (DNN) method is proposed to solve the kinematic problems of grinding robot, the problem of solving robot kinematics is transformed into a weight coefficient matrix, bias vector, and input vector of training a neural network, and a series of linear operations and activation operations are carried out.

This paper takes the KUKA KR300 R2500 6-DOF series robot (grinding robot) as an example. The essence of a forward kinematics solution is to solve the position and orientation (X,Y,Z,A,B,C) of the end-effector of the manipulator under the condition of known joint variables (θ_1,θ_2,θ_3,θ_4,θ_5,θ_6). Similarly, the inverse kinematic solution is to solve the angle of each joint in the case of known terminal position and orientation. Therefore, the six joint angles of the manipulator are taken as the input of the whole neural network, the input vector is $J = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$, the position and orientation of the end-effector are taken as the output of the whole neural network, and the output is $W = [X, Y, Z, A, B, C]$. On the contrary, the inverse kinematics is solved by taking the terminal position and orientation as input and the joint angle as output.

For the analysis of grinding robot kinematics, according to the topological structure type of neural network [34,35], a forward propagation neural network structure with six dimensions of input, six dimensions of output, and multiple hidden layers is established (as shown in Figure 6) to realize the desired mapping vector relationship from input space to output space.

According to the classification of DNN's internal neural network layers, the layers are fully connected, any neuron in layer i must be connected to any neuron in layer $i+1$, it is a linear relationship $z = \sum c_i x_i + b$ plus an activation function $\sigma(z)$.

If the activation function is $\sigma(z)$ and the output value of the hidden layer and output layer is w, then the output of the second layer is available. The expression is as follows:

$$\begin{gathered} w_1^2 = \sigma(z_1^2) = \sigma(c_{11}^2 x_1 + c_{12}^2 x_2 + \cdots + c_{1k}^2 x_k + b_1^2) \\ w_2^2 = \sigma(z_2^2) = \sigma(c_{21}^2 x_1 + c_{22}^2 x_2 + \cdots + c_{2k}^2 x_k + b_2^2) \\ \vdots \;\; \vdots \;\; \vdots \\ w_k^2 = \sigma(z_k^2) = \sigma(c_{k1}^2 x_1 + c_{k2}^2 x_2 + \cdots + c_{kk}^2 x_k + b_k^2) \end{gathered} \tag{5}$$

The output of the third layer is w_1^3, the expression is as follows:

$$w_1^3 = \sigma(z_1^3) = \sigma(c_{11}^3 w_1^2 + c_{12}^3 w_2^2 + \cdots + c_{1k}^3 w_k^2 + b_1^3) \tag{6}$$

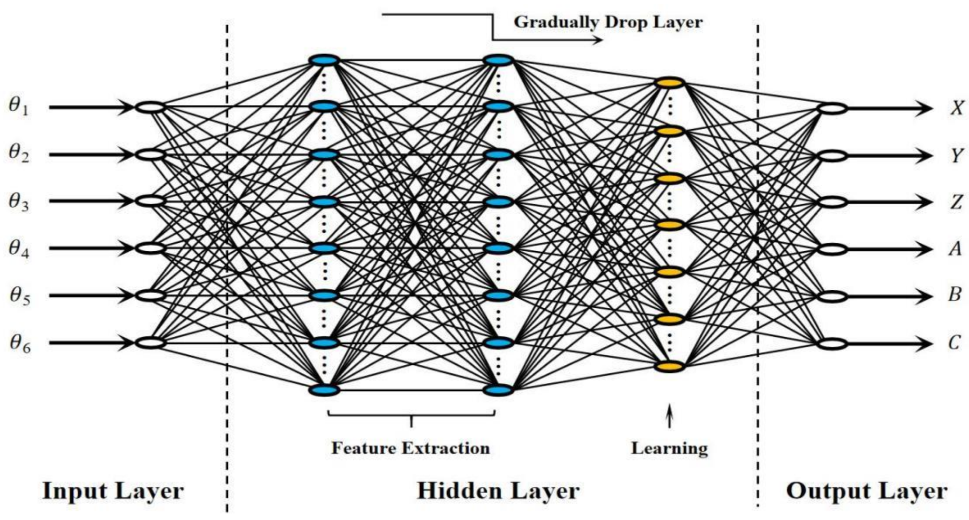

Figure 6. Multi-layer forward propagating neural network structure diagram.

Assuming that layer $i-1$ has a total of m neurons, then the output of the j neuron of layer i is w_j^i, the algebraic expression is as follows:

$$w_j^i = \sigma(z_j^i) = \sigma(\sum_{k=1}^{m} c_{jk}^i w_k^{i-1} + b_j^i) \tag{7}$$

Assuming that the layer $i-1$ has a total of m neurons, the layer i has a total of n neurons, then the layer i has coefficient matrix C^i of the order $n \times m$, the bias vector for layer i has vector b^i of the order $n \times 1$, the output of layer $i-1$ has vector w^{i-1} of the order $m \times 1$, the output of the inactive parameter at layer i has vector z^i of the order $n \times 1$, the output of layer i has vector w^i of the order $n \times 1$, the matrix expression is as follows:

$$w^i = \sigma(z^i) = \sigma(C^i w^{i-1} + b^i) \tag{8}$$

where C refers to the matrix corresponding to the hidden layer and the output layer, b is the bias vector, z refers to the parameters contained in the activation function, w refers to the matrix corresponding to the output layer.

3.3. Complexity

In order to better prove the effectiveness of the method proposed in this paper, the space complexity and time complexity of DNN method are explained, which meet the conditions of increasing relationship between execution efficiency and data volume. Time complexity refers to the number of operations of the model, which measures how fast or slow the model runs. Time complexity refers to the number of operations of the model, which is an indicator to measure hardware performance and determines the training and prediction time of the model. If the complexity is too high, the model training and prediction will consume a lot of time, which can neither verify the idea and improve the model quickly nor make fast prediction. Space complexity refers to the number of model parameters and determines the number of model parameters. Due to the limitation of curse of dimensionality, the more parameters of the model, the greater the amount of data required to train the model, while the data set in real life is usually not too large, which will lead to the training of the model is easier to overfit.

The time complexity is calculated as follows: suppose the input sequence is of dimension n, each element has dimension d, and the number of neurons is x The big O is used to simplify time complexity. The number of repetitions of the basic operation of the algorithm is a function $f(n)$ of module n. Therefore, the time complexity of the algorithm is denoted as $T(n) = O(f(n))$. Less complexity means better code. Space complexity is a measure of the amount of storage space temporarily occupied by an algorithm during its operation. The calculation rule of space complexity is basically similar to that of time complexity, which can be denoted as $S(n) = S(f(n))$.

(1) Forward kinematics solution

Table 2 shows the calculation process of time complexity and space complexity of forward kinematics.

Table 2. Forward kinematics complexity calculation.

i	Layer	Input Dimension *n*	Element Dimension d	Neuron x	Time Complexity	Space Complexity
1	Fully Connected	6	1	64	O (*n* * d * x)	S (*n* * d * x)
2	Activation	64	1	N/A	O (*n* * d)	S (*n*)
3	Fully Connected	64	1	64	O (*n* * d * x)	S (*n* * d * x)
4	Activation	64	1	N/A	O (*n* * d)	S (*n*)
5	Fully Connected	64	1	64	O (*n* * d * x)	S (*n* * d * x)
6	Activation	64	1	N/A	O (*n* * d)	S (*n*)
7	Fully Connected	64	1	32	O (*n* * d * x)	S (*n* * d * x)
8	Activation	32	1	N/A	O (*n* * d)	S (*n*)
9	Fully Connected	32	1	16	O (*n* * d * x)	S (*n* * d * x)
10	Activation	16	1	N/A	O (*n* * d)	S (*n*)
11	Fully Connected	16	1	6	O (*n* * d * x)	S (*n* * d * x)

The number of times a statement is executed in an algorithm is called the statement frequency or time–frequency, which can be denoted as $T(n)$. In the time–frequency $T(n)$, n is called the scale of the problem. When n keeps changing, the time–frequency $T(n)$ will also keep changing. Space complexity is a measure of the amount of storage space temporarily occupied by an algorithm during its run, also reflecting a trend. The amount of storage space required by an algorithm is expressed as $f(n)$. In Table 2, $T(n) = O(f(n))$ indicates that with the increase of problem size n, the growth rate of algorithm execution time is the same as that of $f(n)$. $S(n) = S(f(n))$ indicates that when the space complexity of an algorithm is linearly proportional to n, it can be expressed as $S(n)$.

The layers of the deep neural network structure are connected in series, fully connected, and gradually descend during deep learning. Space complexity is a measure of the amount of storage space temporarily occupied by an algorithm during running, not the number of bytes occupied by a program, so space complexity is the number of variables.

(2) Inverse kinematics solution

Table 3 shows the calculation process of time complexity and space complexity of inverse kinematics.

The number of times a statement is executed in an algorithm is called the statement frequency or time–frequency, which can be denoted as $T(n)$. In the time–frequency $T(n)$, n is called the scale of the problem. When n keeps changing, the time–frequency $T(n)$ will also keep changing. Space complexity is a measure of the amount of storage space temporarily occupied by an algorithm during its run, also reflecting a trend. The amount of storage space required by an algorithm is expressed as $f(n)$. In Table 2, $T(n) = O(f(n))$ indicates that with the increase of problem size n, the growth rate of algorithm execution time is the same as that of $f(n)$. $S(n) = S(f(n))$ indicates that when the space complexity of an algorithm is linearly proportional to n, it can be expressed as $S(n)$.

Table 3. Inverse kinematics complexity calculation.

i	Layer	Input Dimension *n*	Element Dimension d	Neuron x	Time Complexity	Space Complexity
1	Fully Connected	6	1	128	O (*n* * d * x)	S (*n* * d * x)
2	Activation	128	1	N/A	O (*n* * d)	S (*n*)
3	Fully Connected	128	1	128	O (*n* * d * x)	S (*n* * d * x)
4	Activation	128	1	N/A	O (*n* * d)	S (*n*)
5	Fully Connected	128	1	128	O (*n* * d * x)	S (*n* * d * x)
6	Activation	128	1	N/A	O (*n* * d)	S (n)
7	Fully Connected	128	1	128	O (*n* * d * x)	S (*n* * d * x)
8	Activation	128	1	N/A	O (*n* * d)	S (*n*)
9	Fully Connected	128	1	128	O (*n* * d * x)	S (*n* * d * x)
10	Activation	128	1	N/A	O (*n* * d)	S (*n*)
11	Fully Connected	128	1	128	O (*n* * d * x)	S (*n* * d * x)
12	Activation	128	1	N/A	O (*n* * d)	S (*n*)
13	Fully Connected	128	1	64	O (*n* * d * x)	S (*n* * d * x)
14	Activation	64	1	N/A	O (*n* * d)	S (*n*)
15	Fully Connected	64	1	32	O (*n* * d * x)	S (*n* * d * x)
16	Activation	32	1	N/A	O (*n* * d)	S (*n*)
17	Fully Connected	32	1	16	O (*n* * d * x)	S (*n* * d * x)
18	Activation	16	1	N/A	O (*n* * d)	S (n)
19	Fully Connected	16	1	6	O (*n* * d * x)	S (*n* * d * x)

The layers of the deep neural network structure are connected in series, fully connected, and gradually descend during deep learning. Space complexity is a measure of the amount of storage space temporarily occupied by an algorithm during running, not the number of bytes occupied by a program, so space complexity is the number of variables.

4. Simulation Experiments and Optimization

4.1. Experimental Scheme

This paper takes the KUKA KR300 R2500 6-DOF series robot (grinding robot) as an example, the parameters are given as follows: a_1 = 350 mm, a_2 = 1150 mm, a_3 = 41 mm, d_4 = 1000 mm.

Based on the above theories, 550 training samples are selected in this paper, the data used to train the deep neural network in this paper are extracted from the existing Leica-AT960-SR laser tracker in the laboratory, and the data acquisition platform is shown in Figure 7. The KUKA robot model was selected in RoboDyn software, and parameters such as the D-H parameter table and motion range of each joint were imported into the software, and the operating workspace boundary was set. Multiple sets of data were automatically generated by the software and then a simulation was carried out. The corresponding inverse kinematic joint angle is derived by software, excluding the data generated by the irregular running trajectory when coupling occurs. The simulation data were input into the robot demonstrator, the software and the laser tracker were connected to the same subnet, and the T-Mac sensor was installed at the end of the robot. After confirming the accuracy of the calibration, the final data was obtained through the laser tracker. Joint angle sample data are shown in Table 4.

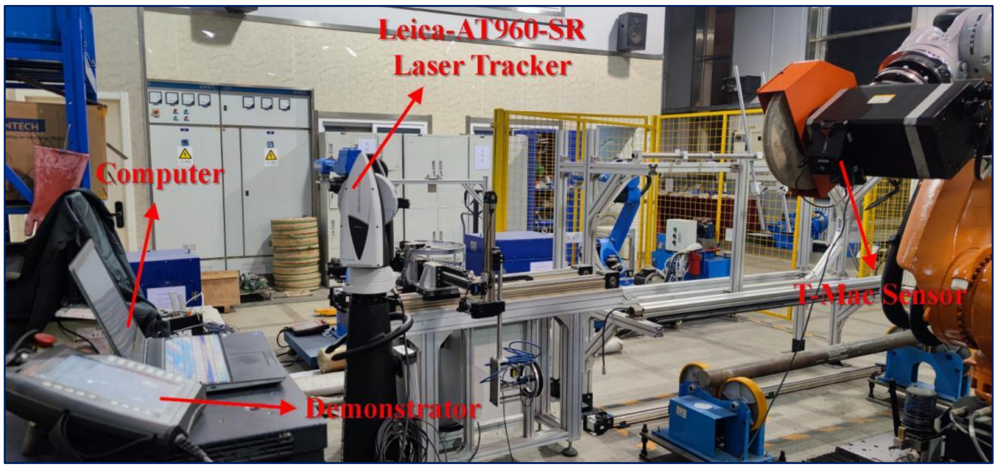

Figure 7. The data acquisition platform.

Table 4. Joint angles sample data. The row represents joint angles, the column represents the sample number.

i	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
1	169.152	−74.474	100.077	−250.680	−19.168	291.015
2	108.117	−10.469	60.329	−325.002	85.537	303.795
3	66.132	−37.705	84.361	−75.441	38.092	−230.169
4	76.237	−135.703	−43.846	−317.680	−98.703	226.420
5	−140.971	−72.721	143.930	−111.730	20.891	−193.332
6	−92.969	−105.562	19.138	139.354	95.771	321.504
7	17.470	−121.286	−78.944	−169.744	83.476	−172.002
8	116.285	−107.124	135.547	−105.011	−74.334	−174.241
9	42.937	−76.106	−23.294	231.580	20.890	34.807
10	154.362	−101.412	88.230	177.610	−29.291	47.475
⋮	⋮	⋮	⋮	⋮	⋮	⋮

The training method of DNN is adopted to set the network structure as 6-128-128- . . . -128-128-64-32-16-6, let every 100 iterations be an epoch, batch size for each iteration is set to 550, so one epoch is equivalent to 55,000 training samples, represented by the number of sessions per hour on the server as the number of training epochs. Use Python to execute code programs.

4.2. Simulation Experiments

(1) Conventional stochastic gradient optimization is solved by using an activation function combining stochastic gradient descent (SGD) and hyperbolic tangent (tanh). The curves of the Sigmoid function and tanh function are shown in Figure 8. The advantage is that it is smooth and easy to derivate and can map a real number to the interval [0, 1]. The disadvantage is that the exponential operation has a large amount of calculation, slow descent speed, and there is division in the derivation when the backpropagation is solving the error gradient. In the process of backpropagation, saturated neurons will lead to the disappearance of the gradient, so that the training of the deep network cannot be completed. The fitting situation of SGD plus tanh solution is shown in Figures 9 and 10.

(2) The activation function combining Nesterov adaptive moment estimation (Nadam) and hyperbolic tangent (tanh) is used to complete the experiment. Nadam has a stronger constraint on the learning rate and a more direct influence on the gradient update. In this case, the gradient disappearance still exists. The main advantage of Nadam over Adam is better performance in the case of disappearing gradients. In general, where you want to use RMSprop or Adaptive Moment Estimation (Adam),

most can use Nadam for better results. The fitting situation of the Nadam plus tanh solution is shown in Figures 11 and 12.

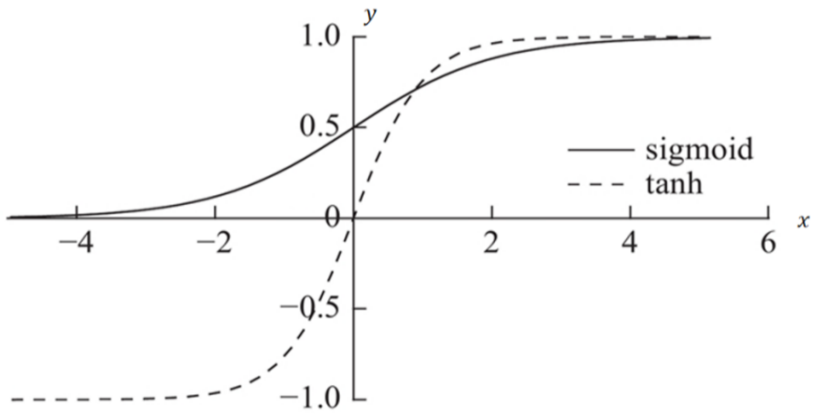

Figure 8. The curves of Sigmoid function and tanh.

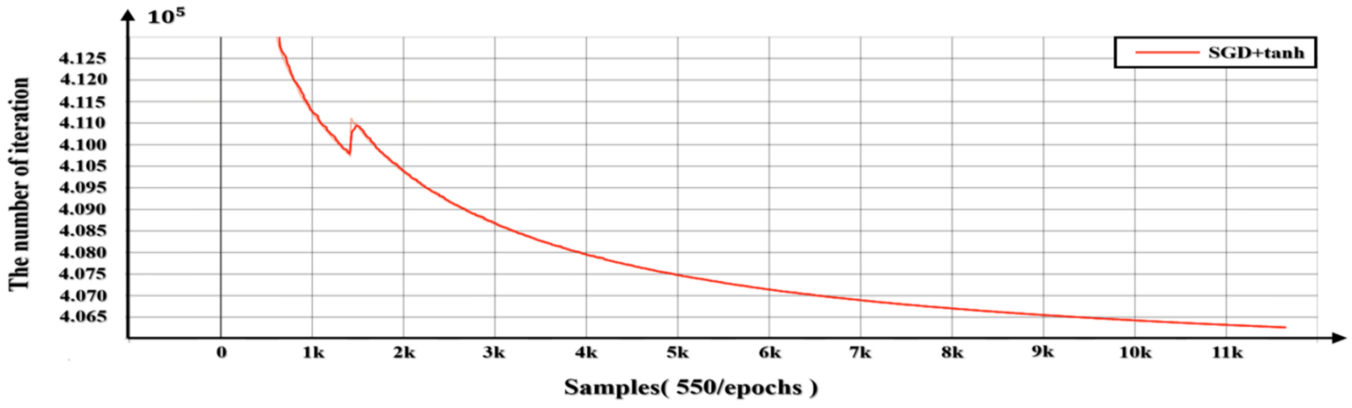

Figure 9. The loss function curve of SGD + tanh solution method.

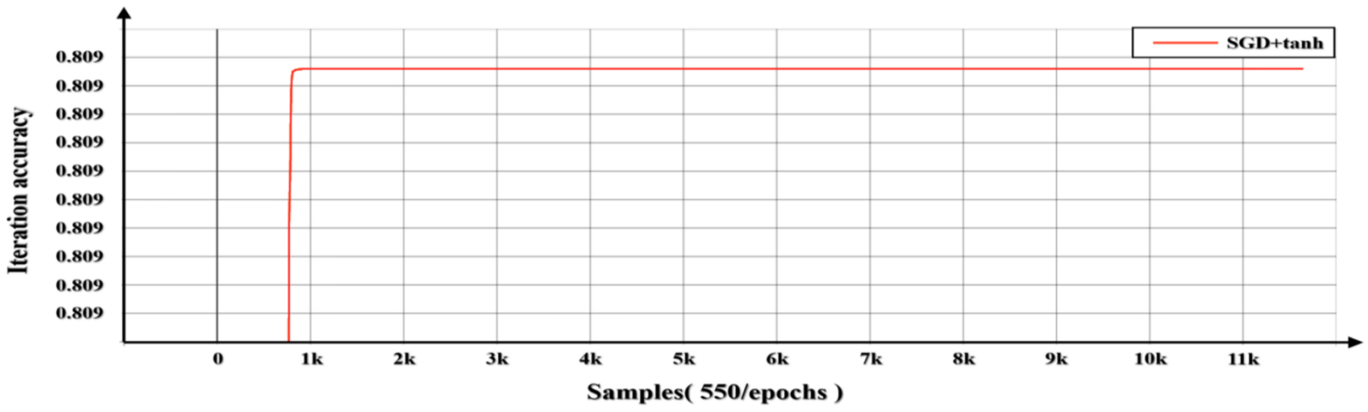

Figure 10. The precision curve of SGD + tanh solution method.

(3) The activation function combining Nesterov adaptive moment estimation (Nadam) and improved ReLU function (ELU) is used to complete the experiment. The curve of the ELU function is shown in Figure 13. Compared with the ReLU function, there is a certain number of outputs in the case of negative input, and this part of the output has a certain anti-interference ability, which can eliminate the problem of ReLU dying. However, there are still problems with gradient saturation and exponential operation, and the calculation intensity is high. The fitting situation of the Nadam plus ELU solution is shown in Figures 14 and 15.

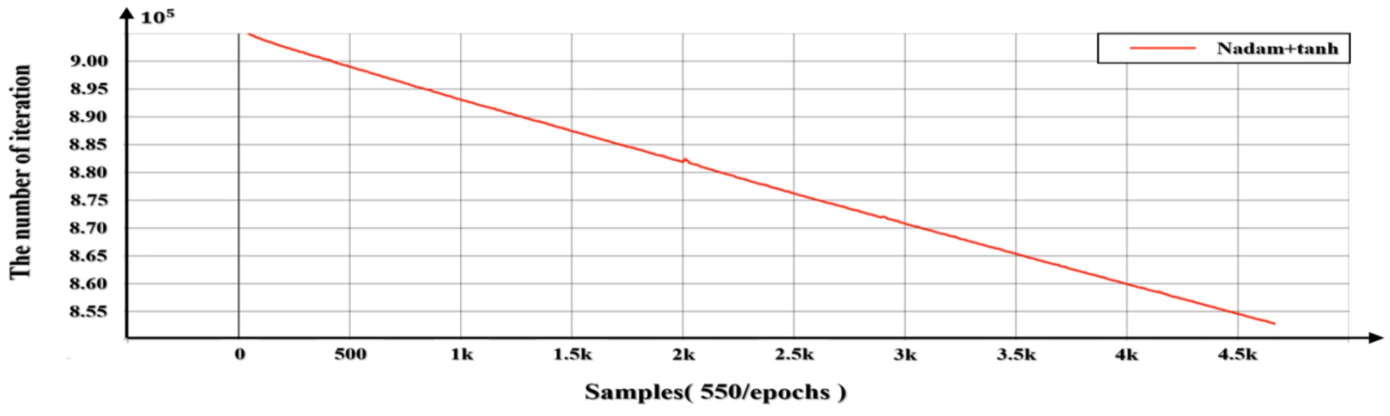

Figure 11. The loss function curve of Nadam + tanh solution method.

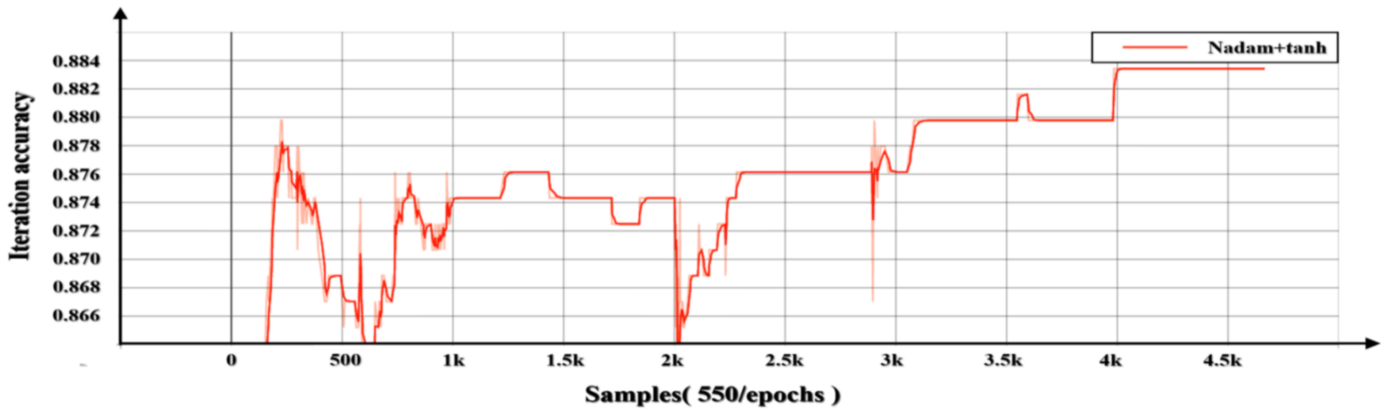

Figure 12. The precision for the solution of Nadam + tanh.

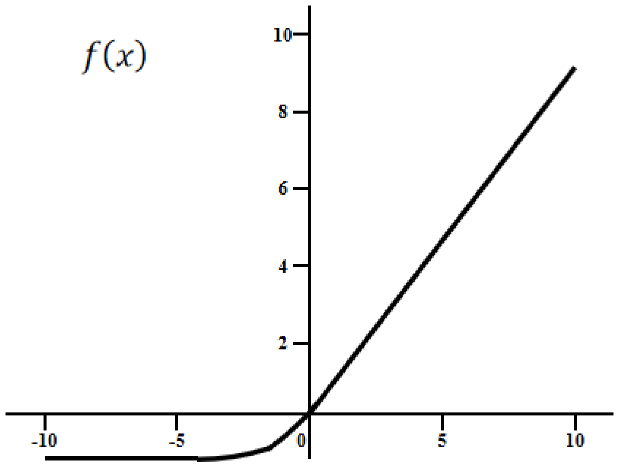

Figure 13. The curves of ELU function.

The loss function used in this paper is mean_squared_error function, which is the mean square error loss function. This function can accurately describe the angle difference when calculating the joint angles, which is equivalent to the accuracy difference of the FK.

The experiment results of the above three solution methods are shown in Figure 16 and Table 5. According to the analysis of data results, the Nadam solver plus ELU function has the fastest solution speed. A higher accuracy is obtained with the increasing number of iterations.

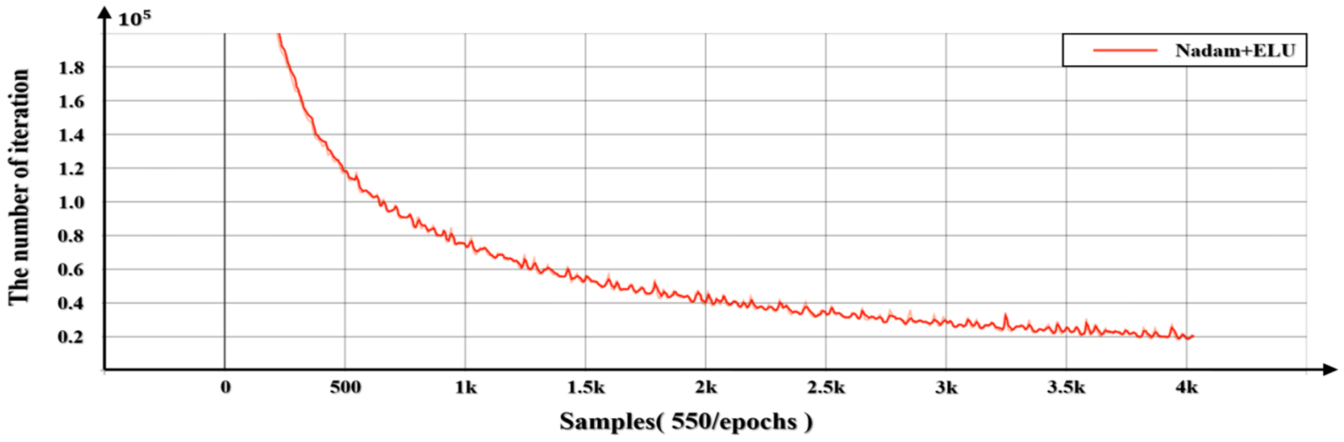

Figure 14. The loss function for the solution of Nadam + ELU.

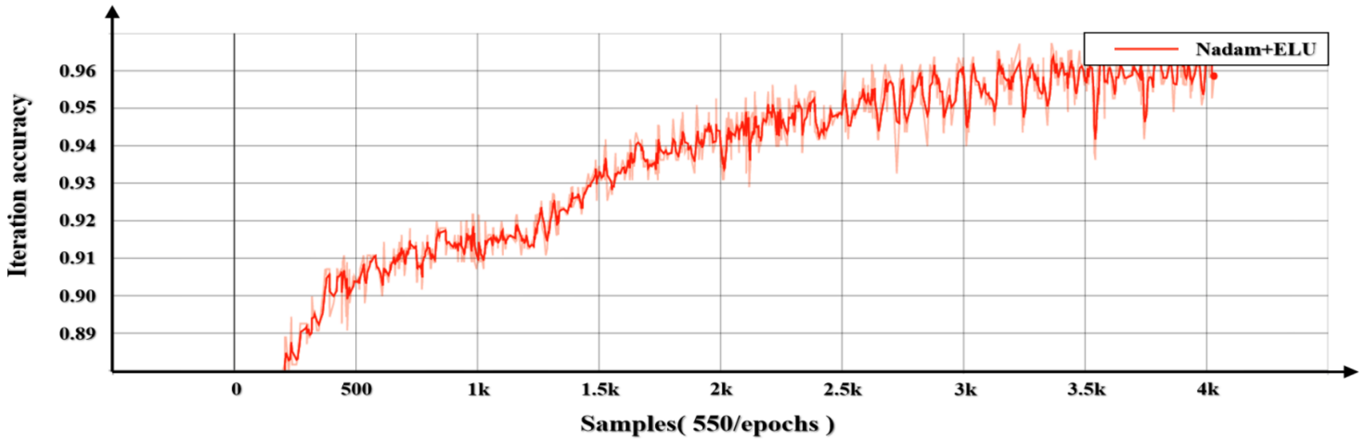

Figure 15. The precision for the solution of Nadam + ELU.

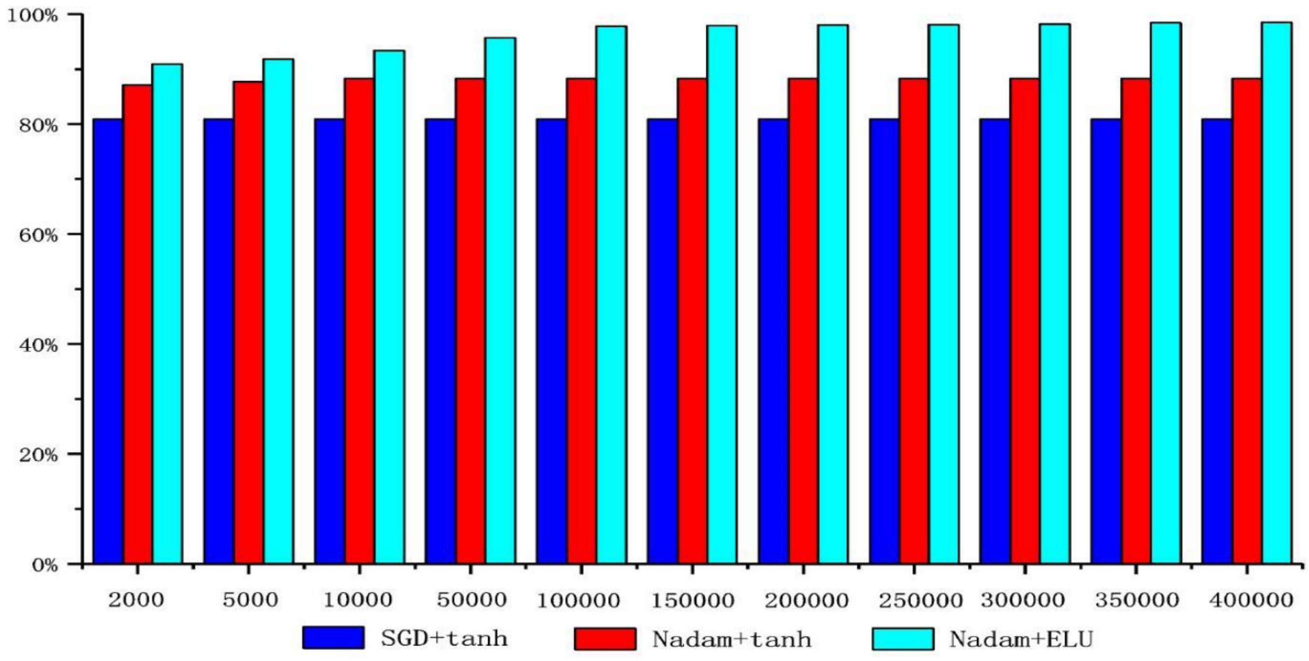

Figure 16. The solving speed comparison of the three solution methods.

Table 5. The partial test results of three solution methods.

Method + Iteration	2000	10,000	100,000	400,000
SGD + Tanh	80.9%	80.9%	80.9%	80.9%
Nadam + Tanh	87.1%	88.3%	88.3%	88.3%
Nadam + ELU	90.9%	93.4%	97.8%	98.5%

Different methods consume different times when reaching different iteration times. With the increase in iteration times, different methods consume more and more time. The time consumed by different methods during iteration is shown in Table 6.

Table 6. The time consumed by different methods during iteration.

Method + Iteration + Consumption Time	SGD + Tanh	Nadam + Tanh	Nadam + ELU
2000	80.9%	87.1%	90.9%
Single iteration time consumption	20 ms	22 ms	22 ms
Total time consumption	40 s	45 s	45 s
10,000	80.9%	88.3%	93.4%
Single iteration time consumption	20 ms	22 ms	22 ms
Total time consumption	3 min 21 s	3 min 47 s	3 min 44 s
100,000	80.9%	88.3%	97.8%
Single iteration time consumption	20 ms	22 ms	22 ms
Total time consumption	33 min 16 s	33 min 51 s	33 min 57 s
400,000	80.9%	88.3%	98.5%
Single iteration time consumption	20 ms	22 ms	22 ms
Total time consumption	2 h 14 min 26 s	2 h 27 min 06 s	2 h 28 min 25 s

4.3. Nadam Optimization

Nadam optimizer is used to optimize the processing results of DNN. The optimization strategy for Nadam (Nesterov accelerated adaptive moment estimation) is to update model parameters according to the following formula [36,37].

$$g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1}) \tag{9}$$

$$\hat{g}_t = \frac{g_t}{1 - \prod_{i=1}^{t} \alpha_i} \tag{10}$$

$$m_t = \alpha_t m_{t-1} + (1 - \alpha_t) g_t \tag{11}$$

$$\hat{m}_t = \frac{m_t}{1 - \prod_{i=1}^{t+1} \alpha_i} \tag{12}$$

$$n_t = \beta n_{t-1} + (1 - \beta) g_t^2 \tag{13}$$

$$\hat{n}_t = \frac{n_t}{1 - \beta^t} \tag{14}$$

$$\bar{m}_t = (1 - \alpha_t) \hat{g}_t + \alpha_{t+1} \hat{m}_t \tag{15}$$

$$\theta_t - \theta_{t-1} = \Delta\theta_t = -\eta \frac{\bar{m}_t}{\sqrt{\hat{n}_t} + \varepsilon} \tag{16}$$

where g_t refers to gradient, θ refers to Initial parameters, α refers to momentum factor, m_t refers to first order moment estimation, n_t refers to second order moment estimation, β refers to gradient cumulant factor, η refers to learning rate, ε refers to learning speed rate and make sure the denominator is not 0.

The obtained experimental results after optimization of the Nadam optimizer, the training time required by the corresponding method to achieve the corresponding accuracy is significantly reduced compared with that before optimization. The results show that the proposed method can greatly improve the accuracy and efficiency of the robot kinematics solution. The optimized time results and efficiency are shown in Table 7.

Table 7. The optimized time results and efficiency.

Method + Accuracy	**Time Required for 80% Accuracy**	**Time Required for 85% Accuracy**	**Time Required for 90% Accuracy**	**Time Required for 95% Accuracy**
SGD + Tanh	About 30 s	N/A	N/A	N/A
Nadam + Tanh	About 20 s	About 30 s	N/A	N/A
Nadam + ELU	About 20 s	About 30 s	About 45 s	About 10 min

(1) The relationship between joint angle and terminal position and orientation can be understood as nonlinear regression of x and y, the nonlinear fitting is carried out by the full connection of layer to layer of DNN, represented as $y = f(x)$. After running the program, the predicted result of terminal position and orientation can be obtained, and then compared with the theoretical analytical value, the absolute mean error of orientation forward kinematics solution is about 0.02°, the absolute mean error of position forward kinematics solution is about 0.3 mm, they are all within the margin of error. It shows that there is little difference between the predicted value and the analytical solution, which can meet the precision requirement of engineering applications. The comparison curves between the analytic solution and the predicted solution of forward kinematics are shown in Figure 17, the selected sample data is shown in Table 8.

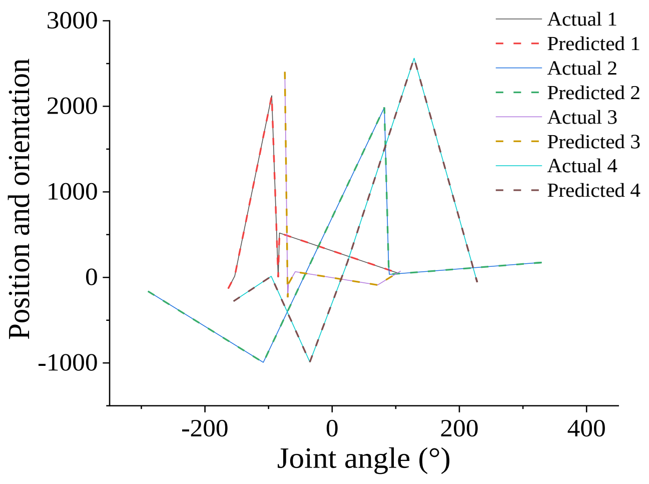

Figure 17. Comparison curves of analytic and predictive solutions of forward kinematics.

(2) Similarly, the relationship between terminal position and orientation and joint angle can be understood as nonlinear regression of y and x. The deep neural network is combined with the inverse kinematics solving program, and the unique inverse kinematics solution of the robot is determined by filtering and limiting conditions. After running the program, the predicted result of joint angle can be obtained and then compared with the theoretical analytical value, the absolute mean error of orientation inverse kinematics solution is about 10^{-5}, and it is almost close to 0. It shows that there is little difference between the predicted value and the analytical solution (negligible), which can meet the precision requirement of engineering applications. The comparison curves between the analytic solution and the predicted solution of inverse kinematics are shown in Figure 18, the selected sample data is shown in Table 9.

Table 8. Comparison of analytic and predictive solutions of forward kinematics.

1	**Joint angle actual value**	103.157	−82.834	−95.024	−163.470	−84.854	−153.296
	Position and orientation actual value	51.512	519.156	2124.172	−130.236	4.834	16.073
	Position and orientation predicted value	52.048	519.625	2124.669	−130.266	4.825	16.080
2	**Joint angle actual value**	88.812	−108.298	82.113	329.419	89.898	−289.636
	Position and orientation actual value	142.725	−993.218	1986.104	174.860	35.362	−161.150
	Position and orientation predicted value	143.089	−993.617	1986.581	174.353	35.718	−161.664
3	**Joint angle actual value**	−69.850	−68.648	−74.447	71.387	−58.072	107.855369
	Position and orientation actual value	−231.802	−71.347	2403.578	−87.774	66.774	75.165
	Position and orientation predicted value	−232.019	−71.836	2403.117	−88.013	66.384	75.881
4	**Joint angle actual value**	−155.081	−35.073	128.912	23.640	−95.757	228.066
	Position and orientation actual value	−278.397	−988.946	2562.402	171.416	12.991	−55.852
	Position and orientation predicted value	−278.662	−989.034	2562.831	171.956	12.448	−55.725

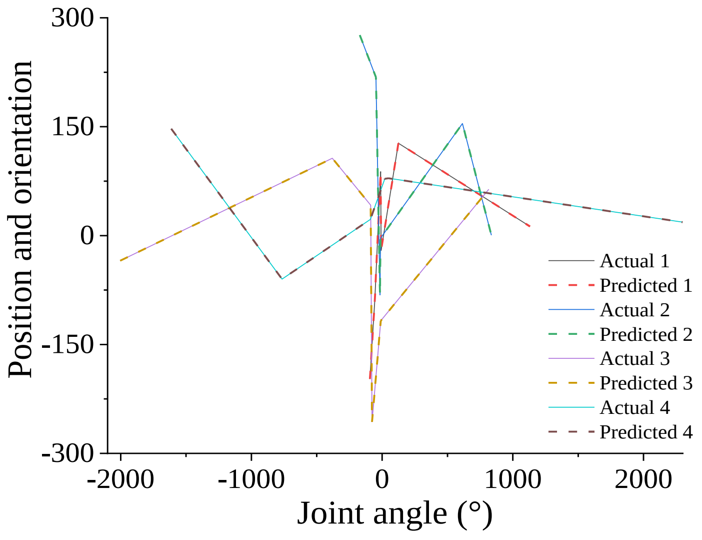

Figure 18. Comparison curves of analytic and predictive solutions of inverse kinematics.

Table 9. Comparison of analytic and predictive solutions of inverse kinematics.

1	**Position and orientation actual value**	1131.414	−7.778	124.19	−11.456	−55.673	−93.113
	Joint angle actual value	12.630900	−20.476474	127.226347	88.156338	−88.722097	−197.538884
	Joint angle predicted value	12.630894	−20.476461	127.226353	88.156347	−88.722106	−197.538896
2	**Position and orientation actual value**	835.82	−16.01	614.392	−47.505	−11.666	−170.864
	Joint angle actual value	0.474678	−81.717642	154.329096	218.121807	−3.515341	276.113429
	Joint angle predicted value	0.474703	−81.717651	154.329104	218.121796	−3.515348	276.113437
3	**Position and orientation actual value**	−380.804	−2003.691	815.821	−76.990	−10.079	−87.533
	Joint angle actual value	106.601969	−34.660064	63.840859	−256.547298	−117.218808	41.888494
	Joint angle predicted value	106.602017	−34.660035	63.840873	−256.547304	−117.218824	41.888486
4	**Position and orientation actual value**	−1613.768	−765.244	2298.126	48.712	20.206	−90.383
	Joint angle actual value	147.210303	−59.896149	18.556023	78.966712	78.258449	22.322419
	Joint angle predicted value	147.210309	−59.896154	18.556041	78.966708	78.258453	22.322422

Table 8 shows the prediction results of forward kinematics, reflecting the terminal pose obtained by giving six joint angles, which is represented as $[\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6] \rightarrow [X, Y, Z, A, B, C]$. Table 9 shows the prediction results of inverse kinematics, reflecting the six joint angles obtained by the given position and orientation, which is represented as $[X, Y, Z, A, B, C] \rightarrow [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]$, where $[X, Y, Z]$ represents position (first three columns) and $[A, B, C]$ represents orientation (last three columns).

The Nesterov momentum included in the Nadam optimizer is capable of producing a better gradient update than classical momentum (Adam). Take advantage of this and train the model with RMSProp (Root Mean Square Prop algorithm, which can adjust the running distance of the model in this direction according to each parameter to accommodate different depths.) containing Nesterov momentum. It can produce a significant effect of consistent dimensions in the training process along the gradient direction, the next accuracy is predicted in advance and accelerating gradient descent learning.

As can be seen from the results in Figures 17 and 18, Tables 8 and 9, the Nadam optimizer is used to optimize the processing results of DNN, and more accurate solutions can be obtained. It can be shown that this method can accelerate the convergence rate of trajectory prediction error, improve the accuracy of trajectory prediction, and realize the process of deep learning.

DNN can be used to enhance robots' ability to carry out human commands, it is particularly good at acquiring representations of linguistic expressions, often requiring training them on large data sets that include robot movements, verbal descriptions, and information about different environments. Therefore, efficient prediction and real-time motion control of robot kinematics can be realized.

5. Conclusions

In this paper, a method based on a multi-layer forward propagation deep neural network is proposed to solve the kinematics equation of a grinding robot, to achieve the goal of fast, efficient, and accurate online grinding operation. The working process and kinematics model of grinding robots are introduced. Based on the proposed method, simulations of the end position and orientation, and joint angle of the grinding robot are given. The calculation examples and simulation shows that the proposed method can meet the requirements of trajectory predictive control and efficient grinding. Compared with the traditional method, the proposed method can obtain higher computational efficiency when solving the end-effector pose of the grinding robot. Finally, the Nadam optimizer is used to optimize the processing results of DNN. The results show that the mean absolute errors of forward and inverse kinematics are in a reasonable range. The optimization results show that this method can accelerate the convergence rate of trajectory prediction error and improve the trajectory prediction accuracy. Compared with the traditional analytical method, this method has a shorter training time, stronger replaceable ability, and shorter response time to solve faults. If the hardware configuration of the computer is higher, the processing speed of this method is faster, and the advantage is more obvious. The actual response time of this method is much shorter than the decision time of mechanical motion. The method proposed in this paper can achieve the goal of fast, efficient, and accurate online grinding operation of the grinding robot. In the future, the author will use DNN to conduct in-depth research on real-time trajectory tracking and visual recognition of grinding robots.

Author Contributions: Conceptualization, W.S., J.Z. and L.L.; methodology, W.S. and L.L.; validation, W.S., J.Z., L.L., T.W. and Y.Z.; software, W.S., L.L. and Z.L.; formal analysis, W.S., J.Z., L.L. and Y.Z.; investigation, W.S. and L.L.; writing—original draft preparation, W.S. and L.L.; writing—review and editing, W.S., J.Z., L.L., T.W. and Y.Z.; supervision, X.X., T.W. and Q.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China, grant number 2018YFB1308702; the Graduate Education Innovation Program of Shanxi Province, grant number 2020BY142; the Special Funding for Guiding Local Scientific and Technological Development of the Central, grant number YDZX20191400002149 and the National Natural Science Foundation of China, grant number 51905367, 51905372, 52105557.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to each of the authors for their contributions to the manuscript and for the support of all funding projects.

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

References

1. Woodside, M.R.; Fischer, J.; Bazzoli, P.; Bristow, D.A.; Landers, R.G. A kinematic error controller for real-time kinematic error correction of industrial robots. *Procedia Manuf.* **2021**, *53*, 705–715. [CrossRef]
2. Nubert, J.; Koehler, J.; Berenz, V.; Allgower, F.; Trimpe, S. Safe and fast tracking on a robot manipulator: Robust MPC and Neural Network Control. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3050–3057. [CrossRef]
3. Pham, D.T.; Nguyen, T.V.; Le, H.X.; Nguyen, L.; Thai, N.H.; Phan, T.A.; Pham, H.T.; Duong, A.H. Adaptive neural network based dynamic surface control for uncertain dual arm robots. *Int. J. Dyn. Control* **2020**, *8*, 824–834. [CrossRef]
4. Huang, P.; Huang, H.Z.; Li, Y.F.; Li, H. Positioning accuracy reliability analysis of industrial robots based on differential kinematics and saddlepoint approximation. *Mech. Mach. Theory* **2021**, *162*, 104367. [CrossRef]
5. Agbaraji, C.E.; Udeani, U.H.; Inyiama, H.C.; Okezie, C.C. Robust control for a 3-DOF articulated robotic manipulator joint torque under uncertainties. *J. Eng. Res. Rep.* **2020**, *9*, 53565.
6. Le, Q.D.; Kang, H.J. Implementation of fault-tolerant control for a robot manipulator based on synchronous sliding mode control. *Appl. Sci.* **2020**, *10*, 2534. [CrossRef]
7. Liu, W.W. Research on Simulation and Experiment of 6-DOF Industrial Robot's Dynamic Characteristics. Master's Thesis, Northeastern University, Boston, MA, USA, 2014.
8. Xiong, Y.L.; Li, W.L.; Chen, W.B.; Yang, H.; Ding, Y.; Zhao, H. *Robotics: Modeling Control and Vision*; Huazhong University of Science & Technology Press: Huazhong, China, 2018.
9. Xie, Z.J.; Feng, C.; Wang, C.F. Kinematics positive solution of 6-PSS parallel robot based on BP neural network. *J. Mach. Des.* **2014**, *31*, 36–39.
10. Li, J.R.; Qi, L.Q.; Han, W.B. Kinematics Analysis and Trajectory Optimization of Six Degree of Freedom Manipulator. *J. Chang. Univ. Sci. Technol.* **2019**, *42*, 68–73.
11. Zhang, H.T. Trajectory simulation control of 6-DOF manipulator based on neural network. *Intern. Combust. Engine Parts* **2020**, *21*, 201–202.
12. Xie, H.; Wang, L.C.; Yuan, X.F.; Chen, H.B. Sliding mode convolutional neural network trajectory tracking control for robot manipulators. *Comput. Eng. Appl.* **2021**, 1–7.
13. Li, M.W.; Qu, G.Y.; Wei, D.Z.; Jia, H.P. Performance optimization of neural network convolution based on GPU platform. *J. Comput. Res. Dev.* **2021**, 1–10.
14. Li, H.; Yan, M.Y.; Lv, Z.Y.; Li, W.M.; Ye, X.C.; Fan, D.R.; Tang, Z.M. Survey on graph neural network acceleration arthitecctures. *J. Comput. Res. Dev.* **2021**, *58*, 1204–1229.
15. Xiao, L.; Jia, L.; Dai, J.; Tan, Z. Design and application of a robust zeroing neural network to kinematical resolution of redundant manipulators under various external disturbances. *Neurocomputing* **2020**, *415*, 174–183. [CrossRef]
16. Xu, Z.; Li, S.; Zhou, X.; Yan, W.; Cheng, T.; Huang, D. Dynamic neural networks based kinematic control for redundant manipulators with model uncertainties. *Neurocomputing* **2019**, *329*, 255–266. [CrossRef]
17. Stefan, G.; Hubert, G.; Andreas, M.; Ronald, N. Robot calibration combining kinematic model and neural network for enhanced positioning and orientation accuracy. *IFAC Pap.* **2020**, *53*, 8432–8437.
18. Alebooyeh, M.; Urbanic, R.J. Neural network model for identifying workspace, forward and inverse kinematics of the 7-DOF YuMi 14000 ABB collaborative robot. *IFAC Pap.* **2019**, *52*, 176–181. [CrossRef]
19. Zubizarreta, A.; Larrea, M.; Irigoyen, E.; Cabanes, I.; Portillo, E. Real time direct kinematic problem computation of the 3PRS robot using neural networks. *Neurocomputing* **2017**, *271*, 104–114. [CrossRef]
20. Cursi, F.; Modugno, V.; Lanari, L.; Oriolo, G.; Kormushev, P. Bayesian neural network modeling and hierarchical MPC for a tendon-driven surgical robot with uncertainty minimization. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2642–2649. [CrossRef]
21. Cursi, F.; Chappell, D.; Kormushev, P. Augmenting loss functions of feedforward neural networks with differential relationships for robot kinematic modelling. In Proceedings of the 20th International Conference on Advanced Robotics (ICAR), Ljubljana, Slovenia, 6–10 December 2021; pp. 201–207.

22. Cursi, F.; Bai, W.; Li, W.; Yeatman, E.M.; Kormushev, P. Augmented neural network for full robot kinematic modelling in SE(3). *IEEE Robot. Autom. Lett.* **2022**, *7*, 7140–7147. [CrossRef]
23. Cronin, N.J. Using deep neural networks for kinematic analysis: Challenges and opportunities. *J. Biomech.* **2021**, *123*, 110460. [CrossRef]
24. Zhao, L.; Jin, J.; Gong, J. Robust zeroing neural network for fixed-time kinematic control of wheeled mobile robot in noise-polluted environment. *Math. Comput. Simul.* **2021**, *185*, 289–307. [CrossRef]
25. Long, C.; Zhang, G.; Zeng, Z.; Hu, J. Finite-time stabilization of complex-valued neural networks with proportional delays and inertial terms: A non-separation approach. *Neural Netw.* **2022**, *148*, 86–95. [CrossRef] [PubMed]
26. Tang, S.; Zhu, Y.; Yuan, S. A novel adaptive convolutional neural network for fault diagnosis of hydraulic piston pump with acoustic images. *Adv. Eng. Inform.* **2022**, *52*, 101554. [CrossRef]
27. Zhu, Q.D.; Wang, X.L. Inverse kinematics algorithm of 6-DOF manipulator. *Robot Tech. Appl.* **2014**, *2*, 12–18.
28. Zhou, X.C.; Meng, Z.D. Algorithm for KUKA robot kinematics calculation. *Ind. Control Comput.* **2014**, *27*, 95–97+100.
29. Zhou, H.; Qin, Y.L.; Chen, H.; Ge, S.Y.; Cao, Y. Structural synthesis of five-degree-of- freedom hybrid kinematics mechanism. *J. Eng. Des.* **2016**, *27*, 390–412. [CrossRef]
30. Sze, V.; Chen, Y.H.; Yang, T.J.; Emer, J.S. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* **2017**, *105*, 2295–2329. [CrossRef]
31. Zhang, H.Y.; Liu, X.W.; Ren, C.; Zhao, B. Forward kinematics control and NURBS trajectory planning for parallel robots. *Mach. Des. Manuf.* **2021**, *4*, 282–286+292.
32. Liu, S.P.; Cao, J.F.; Sun, T.; Hu, J.B.; Fu, Y.; Zhang, S.; Li, S.J. Inverse kinematics analysis of redundant manipulators based on BP neural network. *China Mech. Eng.* **2019**, *30*, 2974–2977+2985.
33. Jin, F. *Fundamental Principles and Methods of Neural Computational Intelligence*; Southwest Jiaotong University: Chengdu, China, 2000.
34. Chen, Y.Y.; Zhang, B.; Fu, Y.X.; Fu, W.; Shen, C.J. Design and kinematics analysis of jujube pruning manipulator. *J. Agric. Mech. Res.* **2021**, *43*, 7–11.
35. Dozat, T. Incorporating nesterov momentum into adam. In Proceedings of the ICLR (2)—Workshop Track, San Juan, Puerto Rico, 24 May 2016.
36. Sutskever, I.; Marten, J.; Dah, G.E.; Hinton, G.E. On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning (ICML), Atlanta, GA, USA, 16–21 June 2013; pp. 1139–1147.
37. Degrave, J.; Felici, F.; Buchli, J.; Neunert, M.; Tracey, B.; Carpanese, F.; Ewalds, T.; Hafner, R.; Abdolmaleki, A.; Casas, D.D.L.; et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **2022**, *602*, 414–419. [CrossRef] [PubMed]