

RESEARCH

Open Access



Development of safety method for a 3-DOF industrial robot based on recurrent neural network

Khaled H. Mahmoud^{1*} , Abdel-Nasser Sharkawy^{2,3}  and G. T. Abdel-Jaber⁴

*Correspondence:
eng.khaledhma@gmail.com

¹ Mechatronics Department,
Faculty of Industry and Energy
Technology, New Cairo
Technological University NCTU,
Cairo, Egypt

² Mechatronics Engineering,
Mechanical Engineering
Department, Faculty
of Engineering, South Valley
University, Qena 83523, Egypt

³ Mechanical Engineering
Department, College
of Engineering, Fahad Bin Sultan
University, Tabuk 47721, Saudi
Arabia

⁴ Mechanical Engineering
Department, Faculty
of Engineering, South Valley
University, Qena 83523, Egypt

Abstract

In this paper, a safety method for a 3-DOF industrial robot is developed based on recurrent neural network (RNN). Safety standards for human robot interaction (HRI) are taken into accounts. The main objective is to detect the undesired collisions on any of robot links. Since most of industrial robots are not collaborative, the dependence of the method on torque sensors to detect collisions makes its ability to use very restricted. Therefore, only the position data of joints are collected to be the data inputs of the proposed method in order to detect the undesired collisions. These data are aggregated from KUKA LWR IV robot while no collisions and in another time when applying collisions. These data are used to train the proposed RNN using Levenberg-Marquardt LM algorithm. KUKA robot is configured to act as a 3-DOF manipulator that moves in space and under the effect of gravity.

The results show that the modelled and trained RNN is sensitive and efficient in detecting collisions on each link of robot separately. Studying the resulted error from the developed model reveals clearly that the method is reliable.

Keywords: Recurrent neural network (RNN), Safety method, 3-DOF industrial robot, Training and testing, Comparisons

Introduction

When the human operator interacts with a robot within a work area, the safety procedures are very crucial and become an urgent necessity. Many of industrial robots are not collaborative. In other words, they do not have torque sensors in their joints whereas all of them have sensors having capabilities of identifying position and angular velocity of their joints. So, there is a need to find a method to use data of position and other data of kinematics of robot to predict its dynamics data, e.g., the external torque.

It is recommended that when designing a HRI system, safety must be taken into account [1]. To integrate an industrial robot or a robotic system in manufacturing processes, there are safety requirements to do so. These requirements are provided by ISO 10218-1 [2], and ISO 10218-2 [3]. Yamada et al. discussed safety when human interacts with robots work area. They provided limits of pain [4]. Industrial settings pose unique challenges that can be interesting dimensions for future research.

Because of the competitive nature and economizing issues of industry, safety systems of industrial robots, based on machine learning, require the robots to be more equipped in order to be more efficient and accurate. Achieving this goal may lead to over-fitting that can revoke the main purpose of adaptable systems [5]. This is a notable reason to search for a method uses the current equipment (its own sensors without redesigning or fixing additional sensors) of robot enabling it from responding with inputs intelligently.

Based on above, the main objective of this paper is to design an accurate approach to detect collisions on any of the robotic manipulator links. The intended approach uses the RNN model that designed and trained to estimate external torques which affect the robot links, only by using joint position signals as known inputs. These signals can be commonly identified and aggregated from most of industrial robots.

When analysing the feasibility of applying of the intelligent control in robotics, it can be concluded that intelligent control enhances quality control and safety in workspace. Moreover, it helps to reduce energy consumed during operation [6].

Literature review

Many research attempts were tried and conducted to guarantee safety when a human in an interaction with robots. Some researchers created their safety methods based on dynamics of robot model. Methodologies for developing the collision detection approach that followed by researchers were as follows: (1) vision approach or sensing of kinematic properties (e.g., torque and position sensors), (2) based on robot model, or (3) data collected from sensors.

Sharkawy et al. [1] designed a NN-based approach to detect collisions by investigating the joint position and torque of the robot as inputs of the developed method. They provided a good model performance. Sharkawy et al. [7] proposed a method based on NN to detect collisions between human and robot. They examined their method on a 2-DOF manipulator makes its motions in a horizontal plan. Sharkawy and Mostafa proposed [8] three types of NN architecture to detect collisions on a one-DOF robot. The three architectures were MLFFNN-1 “multi-layer feedforward neural network having one hidden layer”, MLFFNN-2 “having two hidden layers”, CFNN “cascade forward neural network”, and RNN. They used position parameters of joint as inputs of NNs, where the output of NNs was the measured external torque. Sharkawy and Aspragathos [9] tested a multilayer feedforward neural network using Levenberg-Marquardt algorithm to detect the undesired human-robot collisions. They designed their neural network using the following parameters as inputs, the current position error, the previous position error, the actual velocity, and the measured joint torque of joint. Using the Levenberg Marquardt algorithm for the training process, Sharkawy et al. [10] designed an NN-based approach to detect collisions by investigating only the positioning parameters “position and velocity data which are provided by position sensors” of robot as inputs of model.

Based on fuzzy method, Dimeas et al. [11] built a model to identify collisions affect the robot links. They compared the model performance with another one which built based on time series. Their method considered the position error and the torque of joints as inputs of model. They concluded that the model built based on fuzzy method was more accurate than that built based on time series.

Zhang et al. [12] presented a deep learning-based method using the RNN to analyze the human actions in an assembly setting, and to predict the future trajectory of human operator motion for online robot action planning and execution. Lasota et al. [13], based on a phase-space motion capture system and current angle of joints, designed a system to detect the motions that possibly cause collisions so as to improve the latency of robot to avoid these collisions.

Based on dynamic model of the robot, Heinzmann and Zelinsky defined the torque vector space for all joints and consequently the working space which fulfills constraints. They presented a control scheme of collision force and tested it in simulation and on a real robot. Their test revealed the feasibility of their control scheme [14]. Mu et al. [15] introduced an intelligent demolition robot. Based on robot model, they considered Newton–Euler method to create the dynamic model. They used the modified oriented bounding box method to predict the real-time collisions.

Kaonain et al. [6] studied the safety of human when interacting with robot works in home. They designed a model with Gazebo based on ROS to simulate the interaction between human and robot. They modified the model by integrating physical components to improve its ability of detecting and avoiding collisions when interacting with human.

Yuan et al. [16] proposed a method based on a gated recurrent unit-recurrent neural network GRU-RNN model to optimize the path planning of a mobile robot. They used an improved artificial potential field APF and improved ant colony optimization ACO algorithms to generate the sample sets which are deriving inputs and tags. Inputs was dimensional distances and one angular dimension. The outputs were velocity and angle of the mobile robot.

The previous methods, which were based on robot model, apply a control approach on the dynamic model of the robot. This model becomes much complicated whenever robots' DOF and workspace coordinates increase. In order to improve the results of these methods, some researchers (e.g., [14]) use additional vision equipment. Even when using of AI methods like ANN, some researchers use additional vision and torque sensors. This results in raising the cost of operation. Moreover, additional equipment may contribute to restricting generalization and application of their methods.

The main contribution

In this paper, an approach is designed and trained based on a multilayer RNN to detect human–robot collisions and identify the collided link. A KUKA robot was used to test this model. KUKA robot used here is a collaborative robot “COBOT”. This robot has position and torque sensors on all its joints, and only the joints' positions signals are used in our current approach. This allow the method to be applied to any industrial robot. This robot consists of seven joints, and only three of them was identified to act as a 3-DOF robot.

In the experimental work, a sinusoidal motion is applied on the three joints simultaneously under the effect of gravity. The frequencies of this motion are having variable and constant values. The parameters of the NN model are extracted from the controller of robot, which is called KUKA robot controller, KRC. The values of current position $\theta_i(k)$, previous position $\theta_i(k - 1)$, and angular velocity $\dot{\theta}_i(k)$ of the three joints were aggregated

and used as inputs of the designed neural network. The corresponding external torques τ of the three joints are aggregated and used for the training process. The LM algorithm is used for the training process.

The main aim of our method is the application to any conventional robot. In addition, this method enables operator from observing the collided link specifically.

Outline of the paper

The next sections (from 2 to 6) show the research methodology. “[Methods](#)” section shows the methods of conducting experiments to collect the data being analyzed. This includes the dynamics of the 3-DOF robot and why RNN method was selected to detect collisions exerting on the robot. Also, it shows the design of RNN architecture and how the model is trained. This section shows the governing equations of the main parameters of training model, as well. “[Experimental work](#)” section shows the experimental work which shows how data were collected from KUKA robot and how the RNN was trained. “[Experimental results](#)” section shows the experimental results. “[Discussion and comparison](#)” section shows the discussion of results through conducting a comparison between our proposed method and other previous methods proposed by other researchers. Finally, “[Conclusions](#)” section records the conclusions and proposes the future work.

Methods

In this section, the dynamics of the 3-DOF robotic manipulator are presented and discussed. In addition, the design requirements of the proposed RNN including the inputs and outputs are illustrated. All of these information are described in “Dynamics of 3-DOF robot” and “[RNN design](#)” sections.

Dynamics of 3-DOF robot

From the basic knowledge of robotics, the robot is coupled mechanically. So, any collision on any of its links can make effect on others.

The dynamics equation of the robotic manipulator can be given by

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) \quad (1)$$

where $M(\theta)$ is then $n \times n$ inertia matrix of the manipulator, $V(\theta, \dot{\theta})$ is an $n \times 1$ vector of centrifugal and Coriolis terms, and $G(\theta)$ is an $n \times 1$ vector of gravity terms. τ is actuator torque [17]. Therefore, the dynamics of 3-DOF serial link robot, as Murray et al. mentioned [18], can be written as the following:

$$\tau + \tau_{ext} = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) \quad (2)$$

where τ_{ext} is the external torque which affecting the actuator on joint. $M(\theta)$ and $(\theta, \dot{\theta}) \in \mathbb{R}^{3 \times 3}$. $G(\theta) \in \mathbb{R}^3$.

When applying collision forces on links having great values, Coriolis or inertia forces can be considered as external forces. As the robot is dynamically coupled, these forces have obvious detectable effect on inputs of NN (current position, previous position and current velocity) [1]. It is notable that from anatomy of KUKA robot used in this paper,

inertial forces of links 2 and 3 do not have appreciable effect on joint 1 because it rotates in a plane parallel to the ground. Joint 1 axis is perpendicular to the ground.

The KUKA robot is used in the current work has 7-DOF, but it has been configured to act as a 3-DOF manipulator. Figure 1 shows active joints in the proposed configuration. These joints are A1, A4, and A6 which represent joints J1, J2, and J3 respectively. Link 2 and link 3 are affected by gravity whereas link 1 is not. The figure shows the joints and the links in which external forces such as F1, F2, and F3 are applied. These forces affect the joints as external torques applied on links from many orientation randomly.

There are a wide range of AI approaches can be used to predict the collisions on manipulator's links. As mentioned in the literature review, the approaches include fuzzy logic inference and some types of NN. In fuzzy, defining the linguistic rules for input and output parameters is difficult. Therefore, in exploring a contradictory between inputs and logical corresponding outputs, the rule base must be tuned. This is the most prominent challenge when using fuzzy logic-based approach to solve the problem [19]. Other prominent challenge is the time. Time is an effective parameter in defining the parameters of robot joints, e.g., joint velocity. NN is preferable as an intelligent method compared with other methods, e.g., fuzzy logic. So, the ANN is considered very appropriate to deal with this problem. Using of NN models is more optimistic to achieve higher accuracy. The NN models included, for example, MLFFNN, CFNN, and GRU-RNN. The results of using NN types was a good motivation to investigate RNN method in this paper to detect the collisions on the robot arm. ANN has advantage of dealing with many various problems. In our current work, RNN is considered due to many properties compared with other types of NNs, as follows:

1. RNNs are designed to deal with data having time series or data that in sequence.
2. One of properties of RNNs is universal approximation [20]. It can make a well approximation for non-linear dynamic systems, having random precision and irregular working conditions, by achieving a well-constructed mapping from input to

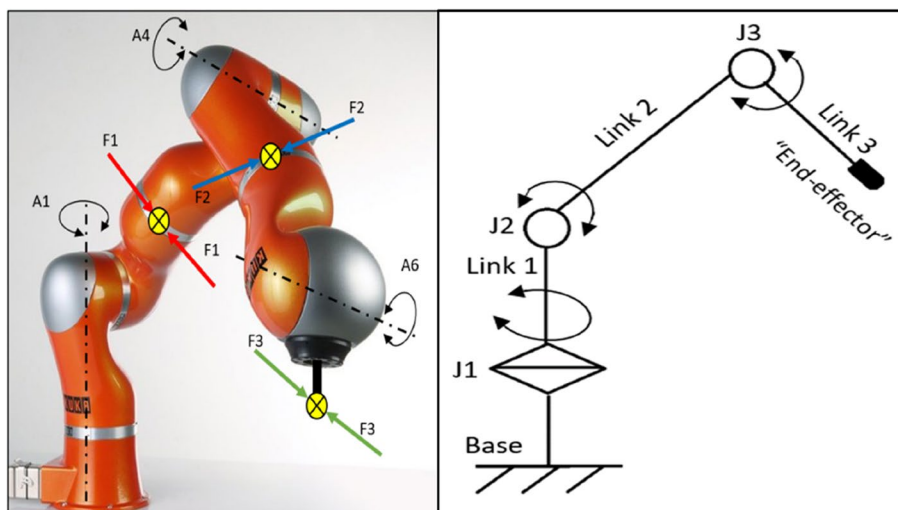


Fig. 1 Configuration of KUKA robot as 3-DOF manipulator

output sequences [21]. The robotic system is a non-linear dynamic system and the random collisions on the robot arm make it working irregularly. In addition, its precision goes random. So RNN is suitable for this system.

3. As Zhao X et al. [22] and Schydlo P et al. [23] presented, when HRI takes place, use of trained RNN becomes the preferable choice and the widely used.

When comparing RNN with ordinary feed-forward NN, it can be noted that ordinary feed-forward NN are just able to deal with data points, which are independent of each other. Traditional feed-forward networks are not appropriate to be used when it is concerned with time series data or with sequential data. Some have argued that since time series data may have autocorrelation or time dependence, the recurrent neural network models which take advantage of time dependence may be useful. Recurrent neural networks “RNNs” are a type of neural networks having a dynamic structure can retain the last data and predict next, or future, output values [24]. Comparing with feedforward networks, the size of RNN is significantly compact when demand the same approximation accuracy of dynamic systems. RNN can be used as associative memories to build attractors from input-output association [25]. The dynamic structure means that the NN model follows completely (or partially) differential equations. It may also follow algebraic equations. Whereas, static NN models follow only algebraic equations [26].

RNN design

The RNNs principle is to preserve the outputs of certain hidden layer and feed them back to the input layer to estimate/predict the desired output of system. The main criteria followed for the design of the proposed RNN is achieving the high performance which is obtaining the lowest/smallest MSE and training error.

RNNs are based on the Hopfield networks with containing feedback paths. Figure 2 shows the proposed model architecture which is a multilayer fully connected RNN with feedback delay (z^{-1}). The network contains three layers: the input layer, the hidden layer, and the output layer. The input layer includes the inputs which are nine inputs: current position $\theta_i(k)$, previous position $\theta_i(k-1)$, and angular velocity $\dot{\theta}_i(k)$ occur at time (kT) where the target “output” is external torque which is predicted at time $(k+1)T$. The hidden layer is a non-linear layer and its activation function is the hyperbolic tangent (tanh). The output layer estimates the external torques of the joints of the 3-DOF robot. Figure 3 shows the flow of signals during training the proposed RNN model. The model is training using LM algorithm. T is the torque observed by KRC and q_d is the desired position (joint angle).

A discrete-time RNN follows a set of non-linear discrete time recurrent equations. Through the composition of the neuron’s functions and the time delays [27]. The equation of a discrete time RNN dynamical system of the state variable $h(k)$ (the state at time k) is the following:

$$h(k+1) = g(Jh(k) + Bx(k) + b) \quad (3)$$

where J and B are weight matrices and b is the bias vector. g is a non-linear function, which is the “tanh” function. k represents the time index [28].

So, the network can be represented in matrix form by:

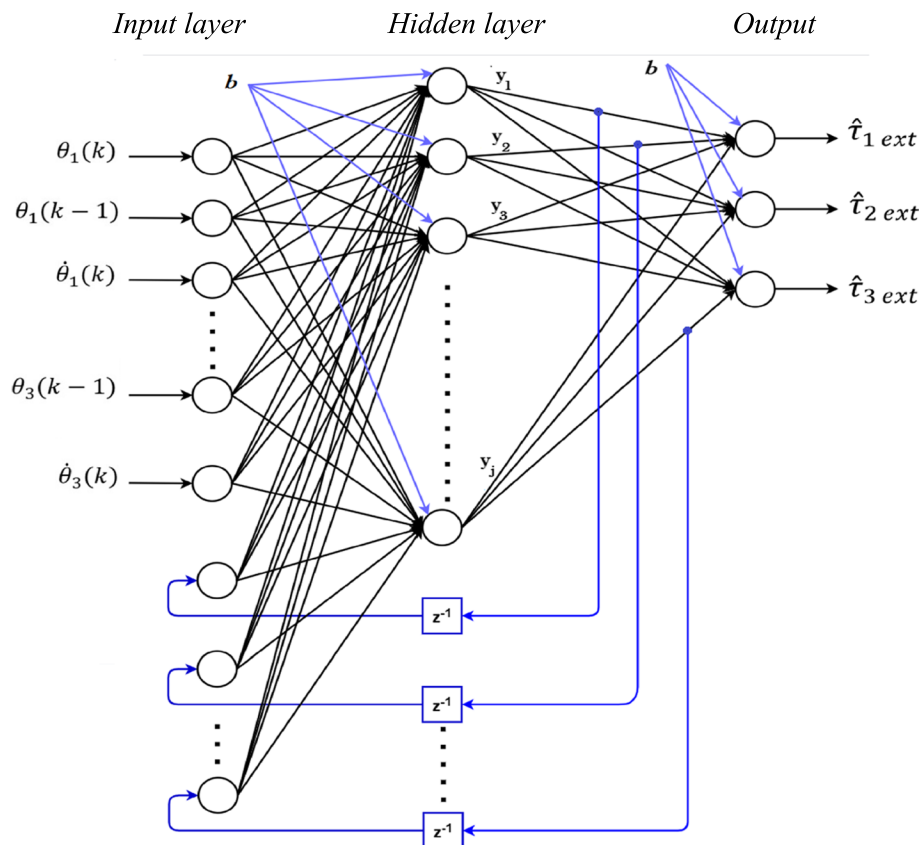


Fig. 2 The proposed multilayer RNN coupling the joints

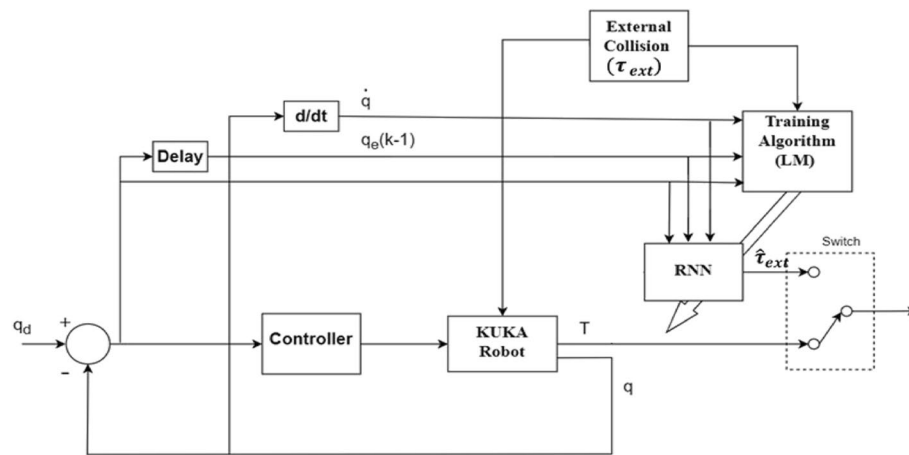


Fig. 3 Block diagram of training of the proposed RNN model

$$\tau(k+1)T = g(W_{ji}y_j(kT) + b) \tag{4}$$

where W_{ji} is a weight matrix, b is the bias vector having a weight unity, and g is activation function, tan-sigmoid “ $\tanh(y_j)$ ”:

$$\tanh(y) = \frac{e^y - e^{-y}}{e^y + e^{-y}} \quad (5)$$

It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface.

LM algorithm is a type of the second-order optimization techniques that have a strong theoretical basis and provide significantly fast convergence, and it is considered as an approximation to Newton's method. According to LM algorithm the optimum adjusted weight can be updated as follows:

$$\Delta w_{ij} = [H + \lambda I]^{-1} g \quad (6)$$

where \mathbf{H} is Hessian of the second order function and \mathbf{g} is the gradient vector of the second order function. \mathbf{I} is the identity matrix of the same dimensions as \mathbf{H} and λ [25].

Error of training $e(t)$ can be estimated by the following [11]:

$$e(t) = \tau_{ext} - \hat{\tau}_{ext} \quad (7)$$

where τ_{ext} is the external torque obtained by KUKA controller, and $\hat{\tau}_{ext}$ is the external estimated torque by the used recurrent neural network. Therefore, the error of training on each joint can be determined by:

$$\begin{aligned} e_1(t) &= \tau_{1ext} - \hat{\tau}_{1ext}, \\ e_2(t) &= \tau_{2ext} - \hat{\tau}_{2ext}, \text{ and} \\ e_3(t) &= \tau_{3ext} - \hat{\tau}_{3ext} \end{aligned} \quad (8)$$

Experimental work

In this section, the executed experiments with KUKA LWR robot are presented in which the data are collected for training the proposed RNN. In addition, the training process is discussed.

Data collection and RNN training

Training the neural network means analyzing its parameters to accomplish accurately the task that neural network assigned to. So, the training strategies of NN can be classified as: supervised training and unsupervised training [26].

- a- Supervised training: training occurs by supplying the neural network with a range of input data and their desired corresponding outputs. While training, the weights are continuously adjusted to reach the minimum error value between the actual and desired targets.
- b- Unsupervised learning: sometimes called "open-loop adapting" because there is no feedback information helping the network to update its parameters.

The strategy that followed in training the proposed RNN is the supervised training strategy.

The data of training, positioning data as inputs and external torques as outputs, were aggregated from the 3-DOF robot during two situations: while no collisions and while exerting collisions. First of them is the robot's motion is happened without applying collisions on its links. In the other experiments, the robot's motion is happened with applying number of random collisions on each link. The data collected from both situations were combined and prepared to be used for training the RNN model. The number of inputs of data combined is 70775 inputs.

As mentioned, the main aim a trained method can detect collisions on each link using only three parameters as inputs: current position, previous position, and angular velocity. MATLAB is used for the training and testing processes of the designed RNN model.

Many trials were attempted to obtain an optimal possible trained model. The trials use 20, 40, 60, and 80 neurons in the hidden layer. The best number of hidden neurons is 80 which give the high performance of the RNN. The training performance is examined by estimating the mean square error of training set and the linear regression. This is discussed in the following subsections.

The mean square error (MSE)

The mean square error on the training set (TMSE) is given by the following equation:

$$\text{TMSE} = \sqrt{\frac{1}{N_T} \sum_{k=1}^{N_T} [y^k - g(x^k, w)]^2} \quad (9)$$

where N_T is the number of observations present in the training set. y^k expresses the measurements of the quantity to be modeled “ y^k ” and $g(x^k, w)$ is the output value of the model with weights vector “ w ” for the vector of variables x^k [26].

About 200 h are spent for the training process and 1000 epochs “iterations” are used and tried to create this trained model. It is known that training neural networks by a fixed learning rate would lead to achieve better performance using MSE approach [29]. Therefore, the learning rate used to train the NN model was 0.01.

As seen in Fig. 4, the best performance “lowest MSE” was observed at value 0.03173. This value is very small and close to zero which means that the designed RNN is trained very well. Figure 5 shows the MSEs resulted from trials conducted to train a model using number of hidden neurons.

Linear regression

Linear regression analysis is the commonly standard approach to test the performance of a model in approximation applications. Let Y_p is an affine function of variables which have been selected in an earlier step, so $Y_p = \zeta^T w_p + B$, where ζ is the vector of the variables of the model, having known dimension q , w_p is the vector of the parameters of the model, and B is a random vector which has expectation value equals zero. From this, the regression function is linear respecting to the model variables. Therefore, the regression function for a model can be determined by [26]:

$$E(Y_p) = \zeta^T w_p \quad (10)$$

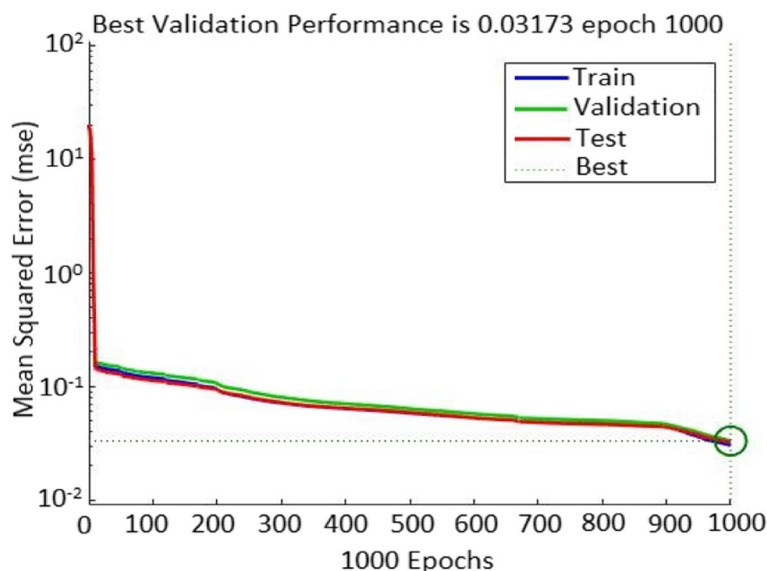


Fig. 4 The lowest MSE for training of RNN of the coupling joints

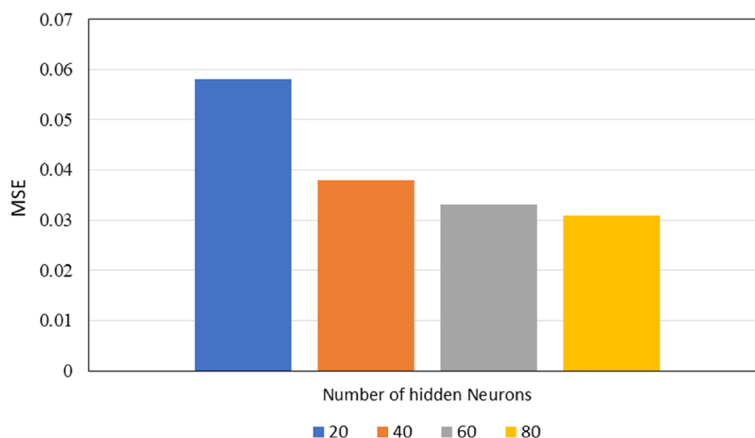


Fig. 5 Number of hidden neurons which have been used for training and their corresponding MSE

Here, the neural network estimated outputs and the corresponding data set targets' outputs for the testing epochs (or instances) are plotted. Figure 6 shows the regression of the trained model. The value of regression is 0.96796 which is very close to value of one. This means that training occurred sufficiently. In addition, the convergence/approximation between the estimated outputs by the trained RNN and the targets is very good.

Experimental results

After the proposed RNN is completely trained, two cases are presented as follows: the first case is that the trained RNN is tested and verified using the data without collisions and the collision threshold is determined. The second case is testing and verified the trained RNN using data with collisions and based on the determined collision threshold. This is followed to show the effectiveness of the method to detect the collisions correctly and efficiently. The following subsection presents that in detail.

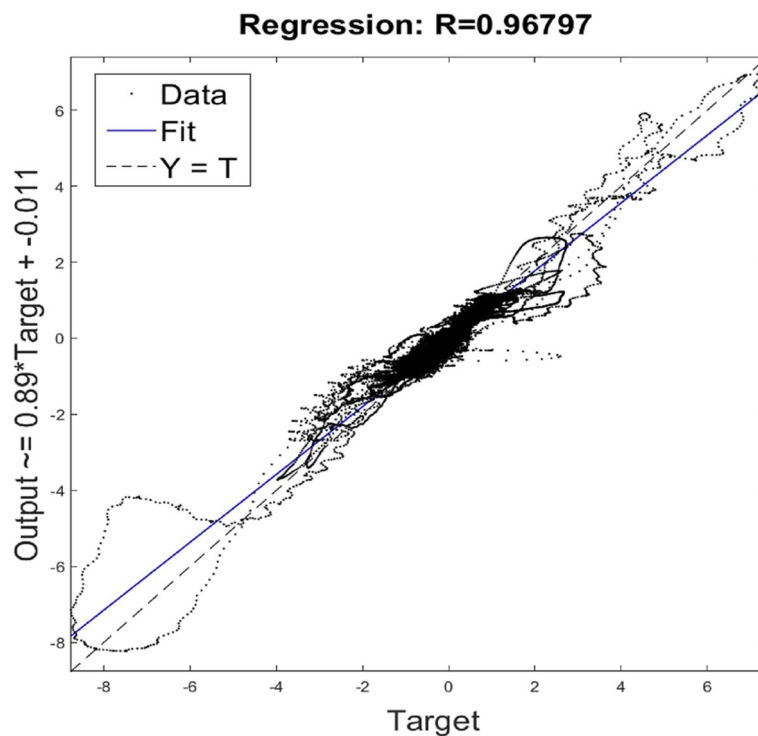


Fig. 6 The resulted regression of trained RNN model

RNN testing and verification

Investigating the trained NN for the data without collision

The collected data without collision are used to investigate the trained RNN. The input data (current position, previous position, and current velocity) are used by the trained RNN to estimate the external torque of each joint. These estimated torques are compared by the given external torques by KRC, as shown in Fig. 7. The main purpose of this test is to determine the maximum of absolute error values for the joints which consequently determine the collision/torque threshold, as discussed in ref. [1, 9, 10]. The torque threshold is the torque value that is used to determine the true collision. The intended error is revealed by observing the difference between the desired external torque and the estimated one by trained RNN. This error was plotted as shown in Fig. 8. From which, the torque thresholds are estimated as follows:

- The torque threshold for joint 1, Thr1 = maximum absolute error + average absolute error, of joint 1.
- The torque threshold for joint 2, Thr2 = maximum absolute error, of joint 2.
- The torque threshold for joint 3, Thr3 = maximum absolute error, of joint 3.

The values of Thr1, Thr2, and Thr3 that resulted from training are 1.3 Nm, 1.2931 Nm, and 0.6394 Nm respectively. The average absolute error between the two external torques on joints J1, J2, and J3 are 0.084 Nm, 0.12 Nm, and 0.067 Nm.

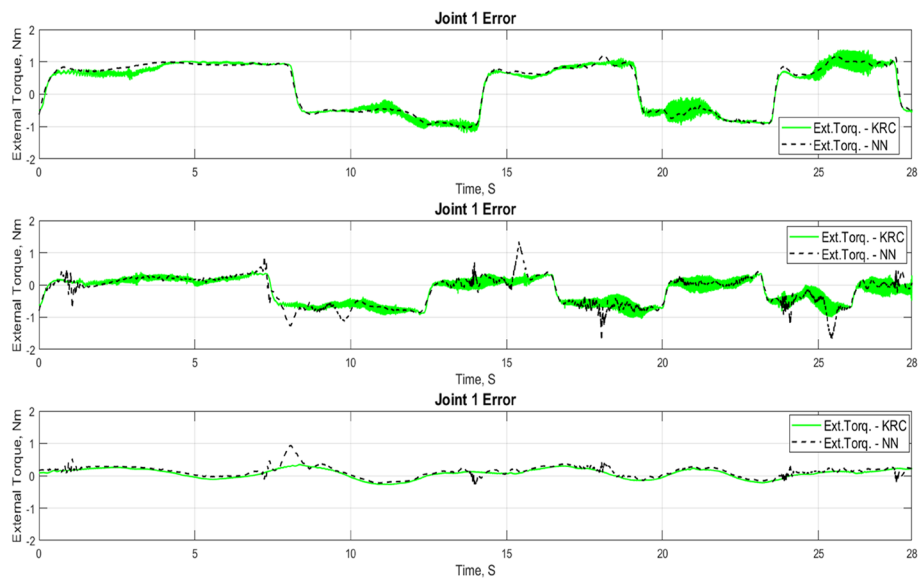


Fig. 7 The three external torques detected on each joint compared to the estimated torques by the RNN, in case of there is no collision

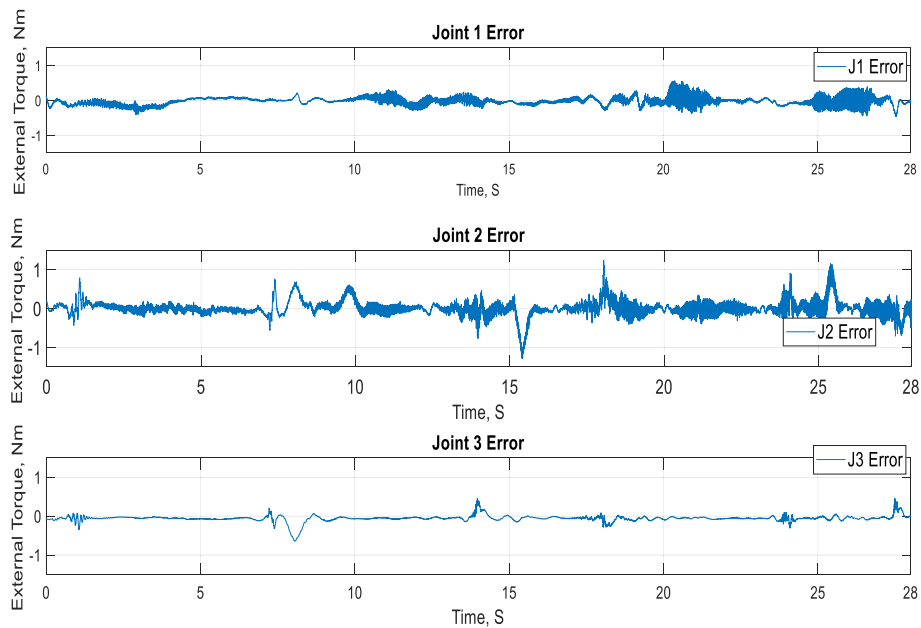


Fig. 8 The error between the detected external torque and the estimated one by trained RNN in case of there is no collision

The error histogram for the trained model is shown in Fig. 9. It shows obviously that error observed, between real measured torques and those estimated by the RNN model, for most of instances is low and close to zero.

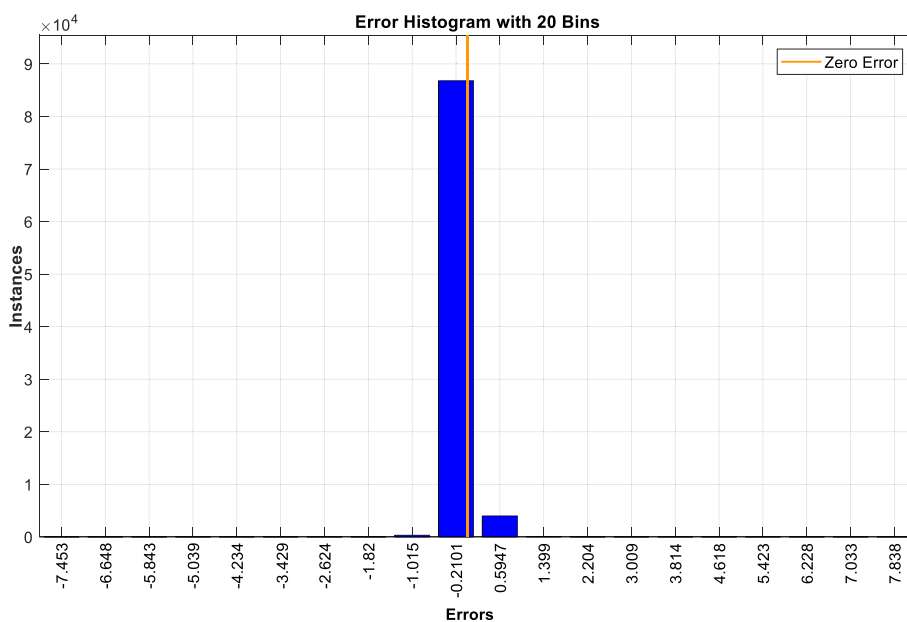


Fig. 9 The error histogram for the trained model, in case of there is no collision

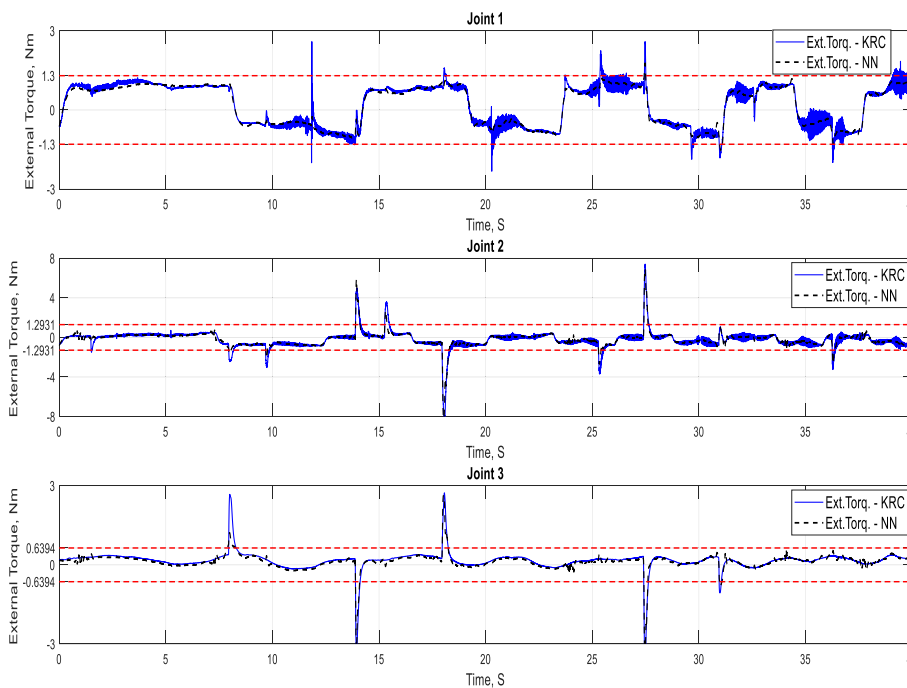


Fig. 10 The external torques detected by NN and KRC, in case of there are collisions

Investigating the trained NN for data with collision

Data collected during the sinusoidal motion of the robot in case of there are collisions, are tested using the RNN model. Figure 10 shows a comparison between external

torque values estimated by the RNN and those measured by KRC. It can be observed that the RNN model almost predicted how external torque behaves sufficiently.

The robot is dynamically coupled. Therefore, the link affected by collisions can be identified according to the following rule:

- If $\hat{\tau}_{ext1} > Thr1$, $\hat{\tau}_{ext2} > Thr2$, and $\hat{\tau}_{ext3} > Thr3$, then link 3 “end-effector” is the collided link.
- If $\hat{\tau}_{ext1} > Thr1$ and $\hat{\tau}_{ext2} > Thr2$, then link 2 is the collided link.
- If $\hat{\tau}_{ext1} > Thr1$ only, then link 1 is the collided link.

As presented from the error histogram in Fig. 11, it can be detected that the number of instances close to zero are slightly lower than those in case of no collisions. But still obviously most of instances are close to zero. Figure 12 shows the error between the external torques in case of the applied collisions to the robot links. The resulted average absolute error on joints J1, J2, and J3 were 0.1 Nm, and 0.14, and 0.067 Nm respectively. The resulted error is low and satisfactory which means that the RNN is trained well and efficient in estimate the external torque or collision in a correct way.

Discussion and comparison

The main aim in developing a method for the robot collision detection is to minimize and reduce the errors or the faults that can result by this method. This section discuss the results through a comparison study to check the effectiveness of our proposed method. This study included methods proposed in references [1, 7–11, 16, 30]. The research work in these references is very related to our proposed work.

The comparison starts by investigating the model design and implementation. This step is identifying inputs and outputs of the model; and the algorithm used to implement this model. Table 1 shows a comparison that clarifies this step. This comparison shows that our current proposed method and the previous study in [10] as well, are

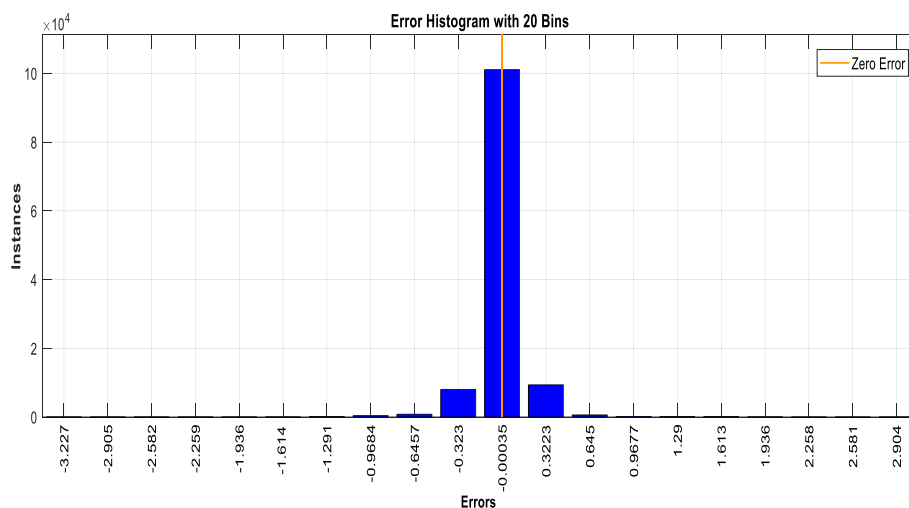


Fig. 11 The error histogram for the trained model, in case of collisions

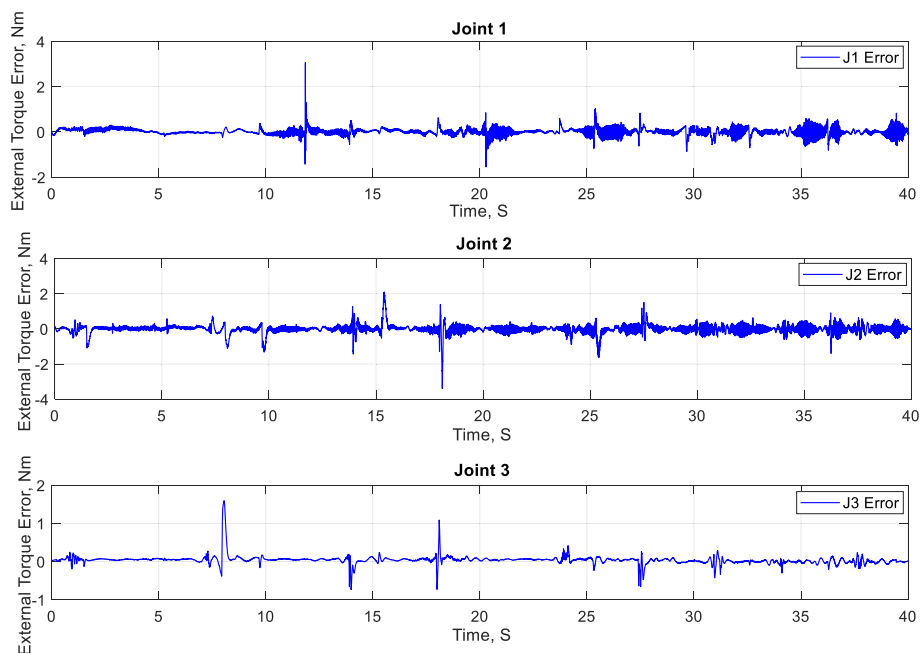


Fig. 12 The error between the detected external torque and the estimated one by trained RNN, when there are applied collisions

Table 1 Comparison among proposed methods according to inputs, outputs, and used algorithm

Algorithm	Inputs	Outputs	Application
MLFFNN using LM algorithm–3-DOF robot [1],	Positioning and torque parameters	External toque	Collaborative robots
2-DOF robot			
MLFFNN using LM algorithm, [7]	Positioning and torque parameters	External toque	Collaborative robots
MLFFNN using LM algorithm, [10]	Positioning parameters only	External toque	Any conventional robot
1-DOF robot			
MLFFNN-1 using LM algorithm, [8]	Positioning and torque parameters	External toque	Collaborative robots
MLFFNN-2 using LM algorithm, [8]	Positioning parameters only	External toque	Any conventional robot
CFNN using LM algorithm, [8]			
RNN using LM algorithm, [8]			
MLFFNN using LM algorithm, [9]	Positioning and torque parameters	External toque	Collaborative robots
Fuzzy using LM algorithm, [11]	Positioning and torque parameters	External toque	Collaborative robots
GRU-RNN using APF and ACO algorithms, [16]	Dimensional distances and one angle	velocity and angle	Any conventional mobile robot
NARXNN-1-DOF, [30]	Positioning parameters only	External toque	Any conventional robot
Our proposed method, RNN-3-DOF robot	Positioning parameters only	External toque	Any conventional robot

applicable for any industrial or conventional robot because it does not need to torque sensors on the robot joints.

Number of hidden neurons, hidden layers, and the algorithm of training, are important parameters affect the training time and the resulted MSE. The comparison between these factors is presented in Table 2. Sharkawy et al. [7] reported that the lowest MSE of them trials was at 120 hidden neurons without mention of the MSE value. They also reported that the average absolute error when collision is higher than when no collision. As presented from Table 2 and Fig. 12, it is clear that our method uses less number of neurons to achieve the lowest MSE.

Next step, comparing the error resulted of each method in case of no collisions is presented. At this step, the average absolute error of each method is shown in Fig. 13. It is clearly found that the average absolute error of our proposed model is satisfied and competitive. It is notable that methods used in references [9, 10] were applied with one and two DOF manipulators, respectively.

The last step is to test the model when there are collisions. As shown from Fig. 14, the authors in reference [1] reported that the errors on all joints were varied from 0.8 to 1.6 Nm. Whereas, in other references [9–11], authors reported that detected errors were high. The presented figure shows obviously that our proposed method resulted very small errors (from 0.06 to 0.14 Nm) (Fig. 15).

Conclusions

This work proposes a method to detect collisions between human and robot in working area. A multi-layer RNN model is built and trained using data collected during a sinusoidal motion with a KUKA LWR IV robot. The function of this model is to estimate the external torques exerting the model links and identify the collided link. The position data are collected by KRC and used as inputs to train the RNN model to predict the external torques or the collisions. The data used for training are aggregated

Table 2 Comparison among proposed methods according to Number of hidden neurons, number of hidden layers, and MSE

Proposal	Number of hidden neurons	Number of hidden layers	MSE
MLFFNN–3DOF robot, [1]	150	1	0.034
2-DOF robot			
MLFFNN, [7]	120	1	–
MLFFNN-2DOF, [10]	120	2	0.036
With 1-DOF robot			
MLFFNN-1, [8]	90	1	0.041
MLFFNN-2, [8]	70 (35 neuron for both hidden layer)	2	0.217
CFNN, [8]	35	1	0.392
RNN, [8]	20	1	0.431
MLFFNN, [9]	90	1	0.04
GRU-RNN–Mobile robot, [16]	1st hidden layer: 40 GRU. 2nd hidden layer: 30 neurons	2	0.067
NARXNN- 1-DOF, [30]	25	3	0.34353
Our proposed method	80	1	0.031

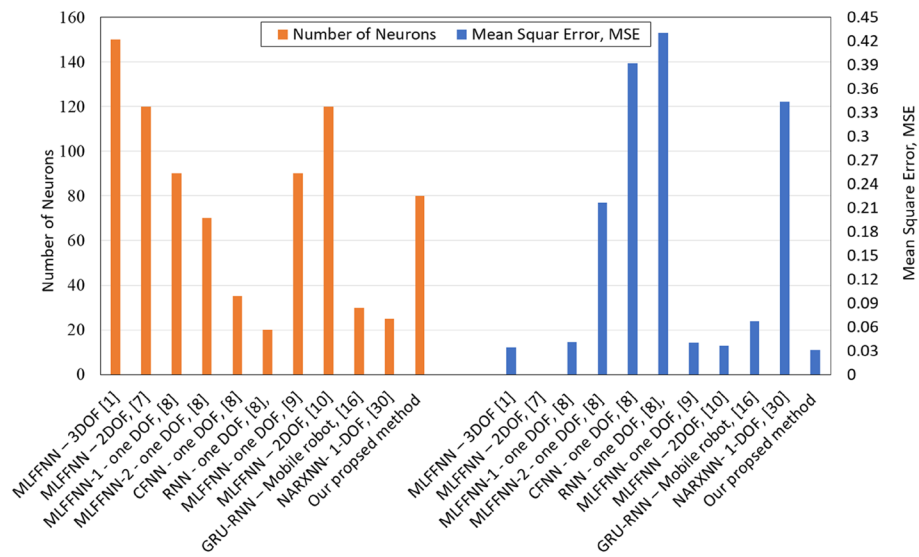


Fig. 13 Number of hidden Neurons used for each method and their corresponding MSE

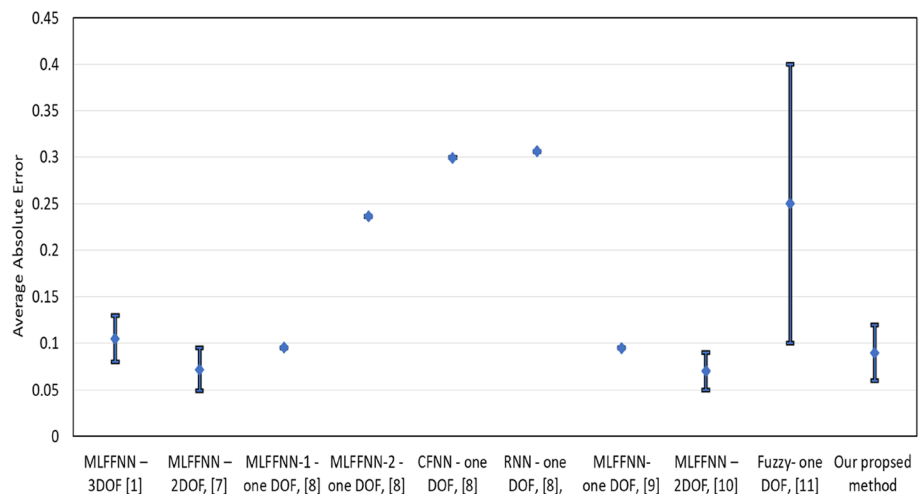


Fig. 14 Comparing the average absolute error, in case of no collisions, between our proposed method and other previous published methods

from robot during two experiments: one without any performed collision, and the other one with exerting random collisions. The robot used to conduct training and testing is the 7-DOF KUKA robot and it was configured to act as a 3-DOF robot. The dynamic coupling between the robot joints helps to identify the collided link. The torque threshold of each joint is the main result of the training process of the RNN and based on this threshold the collision is detected.

The results show that the trained model can perfectly predict the external torque behaviour. The resulted torque threshold enables the model from detecting collisions accurately. Moreover, the collided links are identified accurately. It is notable that the average absolute error of external torques on the three joints are ranged from 0.067

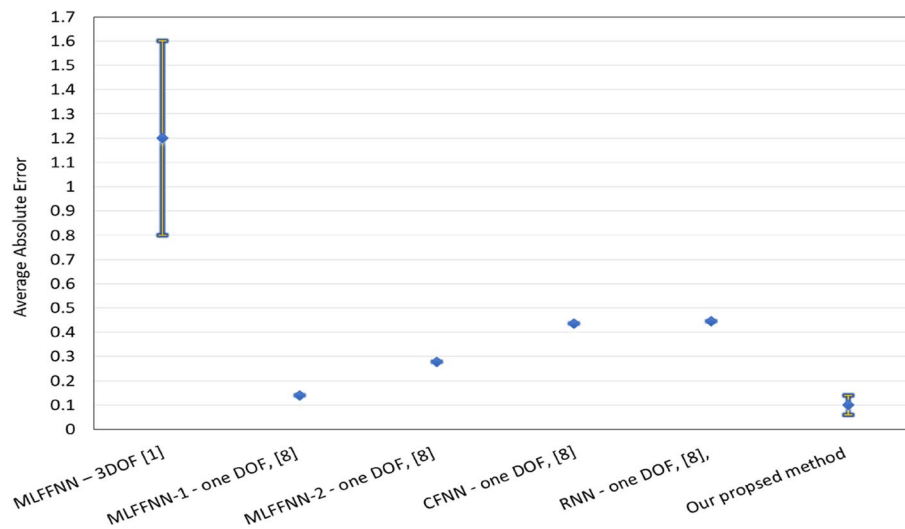


Fig. 15 Comparison of average absolute error, in case of collisions, between our proposed method and other previous published methods

to 0.12 Nm, in case of there is no collision, and from 0.067 to 0.14 Nm while applying random collisions. From which, it can be concluded that the model is reliable. Both error ranges (in case of collision and in case of NO collision) are very close from each other. In other words, the model behaves almost the same performance when collision and when no collision.

As this research revealed, the use of one hidden layer and 80 hidden neurons to train an NN model produced good performance. So, it is predicted that use of more than 80 neurons can result better performance. Also, use of multi-hidden layer can be investigated to achieve more realistic trained model. After making this future improvement, the ability of generalization of the NN model outside of the training ranges will be checked.

The time spent by the model to response collisions needs further study to find a method in order to reduce this time.

In future work, different NNs types, especially LSTM, can be investigated and compared with our current approach. In addition, deep learning can be considered. Application of the current method to 7-DOF robot will be also taken into accounts.

Abbreviations

- RNN Recurrent neural network
- DOF Degrees of freedom
- HRI Human-robot interaction
- KUKA LWR Kuka robot light weight
- COBOT Collaborative robot
- LM algorithm Levenberg-Marquardt algorithm
- MLFFNN Multi-layer feed forward neural network
- CFNN Cascade forward neural network
- ROS Robot operating system
- GRU-RNN Gated recurrent unit-recurrent neural network
- APF Artificial potential field
- ACO Ant colony optimization
- NARXNN Non-linear auto-regressive network with exogenous inputs neural network
- LSTM Long short term memory

Acknowledgements

Thanks a lot for Dr. Hatem Yousry, Akhbar El Youm Academy, Giza, Egypt, for his valued and appreciated guidance when preparing this paper.

Authors' contributions

Data handling, analysing, investigation, and preparation and writing the draft of manuscript were done by KH [first author]. Conceptualization, required resources, visualization, supervision, and editing-review were done by AS [second author]. GT [third author] supervised the whole work of this paper. All authors contributed in building and revising this manuscript. So all authors read and approved the final manuscript.

Funding

Authors declare that the manuscript did not receive any financial support.

Availability of data and materials

The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations**Competing interests**

The authors declare that they have no competing interests.

Received: 5 November 2022 Accepted: 3 May 2023

Published online: 13 May 2023

References

1. Sharkawy A-N, Koustoumpardis PN, Aspragathos N (2020) Human–robot collisions detection for safe human–robot interaction using one multi-input–output neural network. *Soft Comput* 24(9):6687–6719
2. ISO (2011) Robots and robotic devices—safety requirements for industrial robots—part 1: robots, 10218–1
3. ISO (2011) Robots and robotic devices—safety requirements for industrial robots—part 2: robot systems and integration
4. Yamada Y, Hirasawa Y, Huang S, Umetani Y, Suita K (1997) Human-robot contact in the safeguarding space. *IEEE/ASME Trans Mechatron* 2(4):230–236
5. Mukherjee D, Gupta K, Chang LH, Najjaran H (2022) A survey of robot learning strategies for human-robot collaboration in industrial settings. *Robot Comput Integr Manuf* 73:102231
6. Kaonain TE, Rahman MAA, Ariff MHM, Yahya WJ, Mondal K (2021) Collaborative robot safety for human-robot interaction in domestic simulated environments. In: *The 6th International Conference on Industrial, Mechanical, Electrical and Chemical Engineering - ICIMECE 2020, Solo, Indonesia*
7. Sharkawy A-N, Koustoumpardis PN, Aspragathos NA (2019) Manipulator collision detection and collided link identification based on neural networks. pp 3–12
8. Sharkawy A-N, Mostfa AA (2021) Neural networks' design and training for safe human-robot cooperation. *J King Saud Univ Eng Sci* 34(8):582–596
9. Sharkawy A-N, Aspragathos N (2018) Human-robot collision detection based on neural networks. *Int J Mech Eng Robot Res* 7(2):150–157
10. Sharkawy A-N, Koustoumpardis PN, Aspragathos N (2020) Neural network design for manipulator collision detection based only on the joint position sensors. *Robotica* 38(10):1737–1755
11. Dimeas F, Avenda L, Nasiopoulou E, Aspragathos N (2013) Robot collision detection based on fuzzy identification and time series modelling. pp 42–48
12. Zhang J, Liu H, Chang Q, Wang L, Gao RX (2020) Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly. *CIRP Ann* 69(1):9–12
13. Lasota PA, Rossano GF, Shah JA (2014) Toward safe close-proximity human-robot interaction with standard industrial robots. pp 339–344
14. Heinzmann J, Zelinsky A (2003) Quantitative safety guarantees for physical human-robot interaction. *Int J Robot Res* 22(7–8):479–504
15. Mu Z, Liu L, Jia L, Zhang L, Ding N, Wang C (2022) Intelligent demolition robot: Structural statics, collision detection, and dynamic control. *Autom Constr* 142:104490
16. Yuan J, Wang H, Lin C, Liu D, Yu D (2019) A novel GRU-RNN network model for dynamic path planning of mobile robot. *IEEE Access* 7:15140–15151
17. Craig JJ (2005) *Introduction to robotics: mechanics and control*. Pearson Education, Inc, USA
18. Murray RM, Li Z, Sastry SS (2017) *A mathematical introduction to robotic manipulation*. CRC press, USA
19. Passino KM, Yurkovich S (1997) *Fuzzy control*. Addison-Wesley Longman Publishing Co., Inc, USA
20. Maass W, Joshi P, Sontag ED (2007) Computational aspects of feedback in neural circuits. *PLoS Comput Biol* 3(1):e165
21. Siegelmann HT, Sontag ED (1991) Turing computability with neural nets. *Appl Math Lett* 4(6):77–80
22. Zhao X, Chumkamon S, Duan S, Rojas J, Pan J (2018) Collaborative human-robot motion generation using LSTM-RNN. pp 1–9
23. Schydlo P, Rakovic M, Jamone L, Santos-Victor J (2018) Anticipation in human-robot cooperation: a recurrent neural network approach for multiple action sequences prediction. pp 5909–5914

24. Brownlee J (April 2022) www.machinelearningmastery.com
25. Sharkawy A-N (2020) Principle of neural network and its main types: review. *J Adv Appl Comput Math* 7:8–19
26. Dreyfus G (2005) *Neural networks: methodology and applications*. Springer-Verlag, Berlin Heidelberg
27. Burns RS (2001) *Advanced control engineering*. Butterworth-Heinemann, Oxford, pp 325–379
28. Goodfellow I, Bengio Y, Courville A (2016) *Deep learning*. MIT Press, Cambridge
29. Irwin GW, Irwin GW, Warwick K, Hunt KJ (1995) Neural network applications in control, 53: let
30. Sharkawy A-N, Ali MM (2022) NARX neural network for safe human–robot collaboration using only joint position sensor. *Logistics* 6(4):75

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- ▶ Convenient online submission
- ▶ Rigorous peer review
- ▶ Open access: articles freely available online
- ▶ High visibility within the field
- ▶ Retaining the copyright to your article

Submit your next manuscript at ▶ [springeropen.com](https://www.springeropen.com)
