applied sciences

MDPI

Article

Neural Network Learning Algorithms for High-Precision Position Control and Drift Attenuation in Robotic Manipulators

Arkadiusz Mystkowski [1,*], Adam Wolniakowski [1], Nesrine Kadri [2], Mateusz Sewiolo [1] and Lorenzo Scalera [3]

[1] Faculty of Electrical Engineering, Bialystok University of Technology, Wiejska 45D, 15-351 Bialystok, Poland; a.wolniakowski@pb.edu.pl (A.W.); mateusz.sewiolo@sd.pb.edu.pl (M.S.)
[2] Institut Supérieur des Sciences Appliquées et de Technologie Mahdia, Rejiche Road, Mahdia 5121, Tunisia; nesrinekadri721@gmail.com
[3] Polytechnic Department of Engineering and Architecture, University of Udine, 33100 Udine, Italy; lorenzo.scalera@uniud.it
* Correspondence: a.mystkowski@pb.edu.pl

Abstract: In this paper, different learning methods based on Artificial Neural Networks (ANNs) are examined to replace the default speed controller for high-precision position control and drift attenuation in robotic manipulators. ANN learning methods including Levenberg–Marquardt and Bayesian Regression are implemented and compared using a UR5 robot with six degrees of freedom to improve trajectory tracking and minimize position error. Extensive simulation and experimental tests on the identification and control of the robot by means of the neural network controllers yield comparable results with respect to the classical controller, showing the feasibility of the proposed approach.

Keywords: robotics; neural network; learning algorithm; trajectory tracking; UR5 robot

Citation: Mystkowski, A.; Wolniakowski, A.; Kadri, N.; Sewiolo, M.; Scalera, L. Neural Network Learning Algorithms for High-Precision Position Control and Drift Attenuation in Robotic Manipulators. *Appl. Sci.* **2023**, *13*, 10854. https://doi.org/10.3390/app131910854

Academic Editor: Vincent A. Cicirello

Received: 6 September 2023
Revised: 22 September 2023
Accepted: 27 September 2023
Published: 29 September 2023

1. Introduction

Nowadays, collaborative robotics is applied to boost performance and add value in countless industries. With scientific advancements, particularly in the fields of electronics computing, and mechanics, robotic technology has emerged as a focal point and thus a source of interest. Present-day robots are often equipped with intelligent algorithms that provide them with autonomy and allow them to spread into new areas.

In the context of Industry 4.0, artificial intelligence (AI), and in particular machine learning and deep learning, provides advanced solutions that can address problems in different application scenarios with complex and nonlinear robot dynamics, obtaining better performance in comparison to classical methods [1]. Especially, Artificial Neural Networks (ANNs) have been widely used in robotics, for example in solving the inverse kinematics (IK) problem. In [2], ANNs and genetic algorithms are used jointly to solve the IK problem of a Stanford robotic manipulator to minimize the positioning error. In particular, a hybrid approach is proposed using a fitness function with the genetic algorithm that results in an improvement of ten orders of magnitude where the error is reduced to micrometre levels.

A new method for the solution of the inverse kinematics problem of a robot arm equipped with a gripper controlled by an ANN is reported in [3]. The presented results show a reduction in positioning error, an increase in robot joint state estimation accuracy, and an optimization of robot motion control. In [4], an Adaptive Neuro-Fuzzy Inference System (ANFIS) and genetic algorithm techniques are used to successfully solve the inverse kinematics problem of a five-degree-of-freedom (DOF) robot arm. Optimization based on an ANN for trajectory control of a 3-DOF robot is illustrated in [5], and indicates that the ANN controller increases the trajectory tracking performance for certain manipulative tasks. An application with a novel visual servoing of a 6-DOF manipulator is presented

In [6], where the IK solution Is found using an ANN. This process Is comprised of four main stages: object recognition, determination of the pose of the object, IK problem solution, and the trajectory planning of the robot arm. The task is performed using a robot arm in an industrial warehouse setting. The results achieved with the ANN control are compared with the radial basis function (RBF) network approach, and with the multi-layer perceptron (MLP) approach.

In [7], classical and ANN controllers with a parallel structure are applied for the trajectory tracking control of industrial commercial manipulators. In [8], an investigation of the feasibility of the use of deep neural networks for the control of robotic arms is carried out. To meet the desired requirements imposed by the real-time control, a multi-layer feed-forward ANN architecture is proposed. Training and testing data are collected, and the artificial intelligence techniques employed in that study allowed for tracking error minimization. The validation results show that the ANN method has a better absolute fit to the reference and a smaller control error overall.

A more advanced, adaptive genetic algorithm (AGA) method of feedback gain tuning of a PD controller with application to the modelling of a single-link flexible arm is presented in [9]. The AGA approach utilizes standard genetic algorithm operators of reproduction and crossover with modifications applied to the mutation operation. Furthermore, three new objective functions are introduced: HAISE (Hub-Angle Integral of the Squared Error), TDISE (Tip-Deflection Integral of the Squared Error), and ADISE (Angle and Deflection Integral of the Squared Error). This new genetic algorithm approach is used successfully to achieve optimal angular positioning of the arm. The results are compared with the approach utilizing direct evaluations of the objective function, and it is shown that the newly developed adaptive mechanism is faster in convergence and effectiveness in error minimization, and in controlling multiple-link flexible robots.

The trajectory tracking of a robotic arm utilizing a feed-forward ANN, which maps between a dataset of numeric inputs and a set of numeric targets, is given in [10]. Training data for the ANN are calculated using forward kinematics equations of the given robot. The ANN is trained to solve an inversion problem. The efficacy of this approach is confirmed using Matlab. In [11], the dynamics identification and multi-link industrial manipulator control are given using Runge–Kutta–Gill Neural Networks (RKGNNs). The neural network method can accurately grasp the changing rates of the states; this method can be effectively used for long-term prediction of the states of the multi-link robot manipulator dynamics. Once identified, the gathered dynamics data can also be used to control the robot for a new trajectory. In [12], researchers designed an industrial robot manipulator controller using two parallel subsystems. First, the PD controller is designed to minimize trajectory tracking errors. Then, the ANN controller is used to generate the required torque by a given dynamic trajectory. Data used for ANN training and online weights update are calculated based on the simplified dynamic model of the robot. The proposed method is tested in simulation and experimentally, and both methods confirm its usefulness. The ANN is implemented in a 2-DOF industrial-grade manipulator [13], in which position controllers utilize neural networks and provide satisfactory performance of trajectory tracking for sinusoidal references. The ANN can be effectively used to adjust PID parameters [14]. Thanks to this, the accuracy of the tracking control is improved over the standard PID algorithm.

More advanced adaptive ANN solutions are utilized in complex systems. For example, in [15], an adaptive neural network controller is proposed for robot manipulators with multiple fingers and uncertain kinematics, dynamics, and Jacobian matrix. Utilizing ANNs allows one to limit the position error despite the presence of uncertainties. Simulations are carried out to show the performance of the proposed controller. In [16], researchers propose global adaptive neural control with finite-time convergence learning performance for a general class of nonlinear robot manipulators. What is more, an adaptive learning algorithm driven by estimated weights error is used to guarantee the neural weights to converge to optimal values in a finite amount of time. The method performance is proven in a simulation environment against conventional methods. In [17], a robust adaptive

control method based on dynamic structure Fuzzy Wavelet Neural Networks (FWNNs) is proposed for trajectory tracking control of industrial robot manipulators with uncertainties and disturbances via adaptive sliding mode control. Good control performance is achieved by utilizing an adaptive learning algorithm that allows FWNNs' weights to be modified online, and the Lyapunov stability theorem. The researchers present a mathematical proof to confirm the functionality and parameters of the given controller.

In [18], the authors present a neural network tracking controller for a SCARA manipulator. A three-layer neural network is trained based on reinforcement learning methodology. In [19], a feed-forward NN controller with robust integral of the sign of the error (RISE) feedback is proposed to control a multiaxial hydraulic manipulator. The performance of the control system is verified using a simulation. An interesting approach using the auto-encoder method is used in [20] for kinematic control of a manipulator. The model is built directly from the kinematic parameters of the robot model rather than from a training dataset. Deep learning's suitability for robot control is explored in [21], where the authors present a framework for progressive neural network training for a robot with an unknown kinematic model.

The main goal of this work Is to p"esen' deep learning-based approaches for the trajectory control of an industrial robot as well as to estimate the position and the velocity of robot joints. Furthermore, we replace the classical PID/PI controller with a controller based on an ANN to improve the trajectory control and optimize the position error. We present the structure of the network based on our own collected datasets. We have used the same datasets for training and testing, so the results can be compared directly.

In more detail, we present a complete ANN controller structure for the trajectory control of a UR5 industrial manipulator. The neural network is subsequently trained on the reference data obtained from sample trajectory executions with the classical control scheme. The ANN training is performed using the Levenberg–Marquardt and Bayesian Regression methods. These designs address trajectory tracking and the joint position drift problem that occurs in the method of robot-control box commanding. The obtained networks are tested by executing a sample trajectory with a real UR5 arm setup. To summarize, the main contributions of this paper are the following:

- the development of different Artificial Neural Networks (ANNs) controllers for the trajectory tracking of an industrial 6-DOF manipulator;
- the verification of the Levenberg–Marquardt and Bayesian Regression learning algorithms for the trajectory control;
- the development of a scheme for the generation of the training and testing data for trajectory tracking using a classical control approach;
- a comparison of numerical neural network training algorithms for the task of trajectory tracking;
- the experimental verification of the proposed approach on a real UR5 robot.

The paper is organized as follows. The methods are described in Section 2, including the kinematic modelling of the robot arm, the trajectory planning approach, the classical online trajectory control, and the proposed neural network controller. The experimental results are reported in Section 3, and the conclusions follow in Section 4.

2. Methods

2.1. Kinematic Modelling of the Robot Arm

In this work, we are interested in the trajectory control of a 6-DOF industrial robot that can be used for collaborative applications. These requirements can be fulfilled with a UR5 robot arm by Universal Robots, shown in Figure 1. This manipulator is a small collaborative table-top robot, designed mostly for application in light assembly tasks and in automated workbench scenarios. The robot can lift a payload of 5 kg within its workspace, with a radius of 850 mm. The manufacturer claims a repeatability of 0.1 mm [22].

Figure 1. The UR5 manipulator by Universal Robots used for the experiments.

To develop a control system for a robot arm, it is necessary to provide a kinematic model of the robot first. A comprehensive derivation of the kinematic and dynamic model of the UR5 manipulator is presented in [23]. The Denavit–Hartenberg (D-H) parameters of the kinematic chain are reported in Table 1.

Table 1. The D-H parameters of the UR5 robot arm.

Joint	a [m]	α [rad]	d [m]	θ [rad]
1	0	$\pi/2$	0.089159	q_1
2	-0.425	0	0	q_2
3	-0.39225	0	0	q_3
4	0	$\pi/2$	0.10915	q_4
5	0	$-\pi/2$	0.09465	q_5
6	0	0	0.0823	q_6

Based on this description, the solution to the forward kinematics problem can be presented as a sequence of coordinate frame transformations between the base and the end frame of the robot. While an explicit solution for the IK problem can be adapted from [23], in our work we have opted for a numerical approach, using the inverse Jacobian method, as in [24].

2.2. Trajectory Generation

For our experiments on trajectory control, we have defined a sample trajectory for the end-effector of the UR5 robot. The trajectory is designed in the Cartesian space in the parametric form of a Lissajous curve:

$$\begin{cases} x = x_0 + cos\theta \cdot A_x \cdot cos\omega_x t - sin\theta \cdot A_y \cdot sin\omega_y t \\ y = y_0 + sin\theta \cdot A_x \cdot cos\omega_x t + cos\theta \cdot A_y \cdot sin\omega_y t \\ z = z_0 \end{cases} \quad (1)$$

In Equation (1), x, y, and z are the coordinates of the curve, x_0, y_0, and z_0 are the location of the curve origin in the workspace of the robot, A_x and A_y are the major and minor axis length parameters, respectively, ω_x and ω_y are the angular frequency parameters, θ is the curve tilt, and $t = [0, 2\pi]$ is the parameter of the curve. In our case, we have defined $A_x = A_y = 0.1$ m, $\omega_x = 36$ rad/s, $\omega_y = 72$ rad/s, $\theta = 1$ rad, and $(x_0, y_0, z_0) = (0.36, 0.24, 0.1)$ m. The obtained curve is shown in Figure 2.

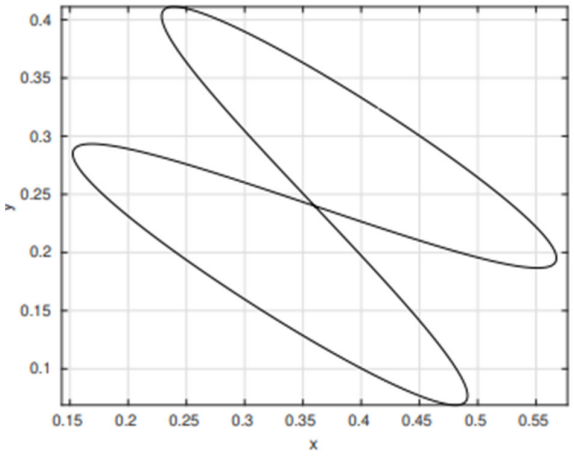


Figure 2. The designed trajectory in the *x*-*y* plane.

To fix the TCP position in space we have assumed a constant TCP frame orientation expressed as *RPY* (π, 0, 0). Furthermore, we have scaled the obtained trajectory in time, such that the duration of the trajectory is T = 10 s.

The Cartesian space trajectory described above is subsequently expressed in the joint space of the robot. This is achieved in the following manner. First, the Cartesian trajectory is sampled at 50 points τ = [0:0.2:10] s. The sampled Cartesian trajectory can be presented as a list of pairs $TP_i = (t_i, P_i)$, where P_i denotes the desired pose of the TCP frame at time t_i.

Assuming the initial robot configuration q_0 = [3.73, −1.28, 0.99, −1.28, −1.57, 2.18], we have obtained IK solutions for all the subsequent points of the trajectory using a Jacobian IK solver. We have thus obtained a sequence of trajectory waypoints in the configuration space expressed as a list of pairs $TQ_i = (t_i, q_i)$, where q_i denotes the robot configuration achieved at time t_i.

Based on the trajectory waypoints TQ_i, we have constructed cubic spline interpolators CS_j for each of the *j*-th joint of the UR5 robot. We can denote the spline interpolator CS_j as a function that generates the desired joint position $q_j(t)$ and the desired joint velocity $\dot{q}_j(t)$ given as:

$$CS_j = \left\{ q_j(t), \dot{q}_j(t) \right\} \tag{2}$$

2.3. Classical Online Trajectory Control

Typically, the commercially available manipulators have few available protocols for controlling the movement of the robot. Usually, there is a simplified interface, where simple point-to-point (PTP) motions can be executed. In that case, the responsibility for the trajectory execution is delegated to the proprietary robot controller. To implement a custom controller, the availability of a lower-level interface to the robot control box is necessary.

The robot driver communicates with the robot control box and sends encoded desired velocity commands over a TCP/IP connection. In UR robots, the online control of robot movement can be achieved with *servoq* and *speedq* URScript instructions [25], which allow for the online position and velocity control, respectively, of the robot joints. These commands are sent to the robot with a control loop frequency of 125 Hz. In our work, we use the *speedj* interface to implement a custom position-velocity control loop. The diagram of the robot control system is shown in Figure 3.

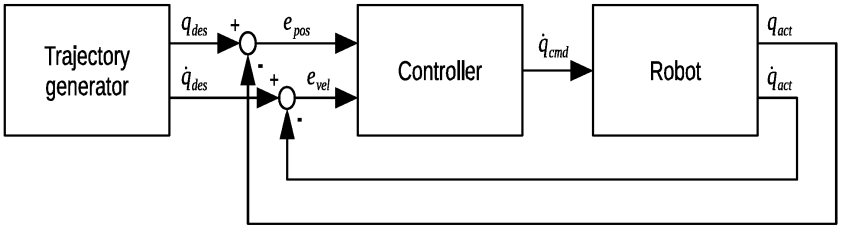


Figure 3. The online trajectory control scheme for the UR5 robot; in case of classical control using PID/PI controller.

The trajectory generator provides desired joint configuration and joint velocity vectors q_{des} and $q_{\cdot des}$. The controller input signals are calculated as $e_{pos} = q_{des} - q_{act}$ and $e_{vel} = q_{\cdot des} - q_{\cdot act}$, where q_{act} and $q_{\cdot act}$, are the actual joint positions and velocities, respectively. The control system presented above is implemented in Matlab, using a custom ROS (Robot Operating System) node to send the velocity commands to the robot and to read the joint states.

As a baseline, the following system is implemented as the controller. The velocity command signal is generated according to the following rule:

$$\dot{q}_{cmd} = k_p \cdot e_{pos} + k_v \cdot e_{vel} \tag{3}$$

where $\dot{q}_{cmd}$ is the controller output, e_{pos} and e_{vel} are the position and velocity errors, respectively, and k_p and k_v are the controller coefficients. In our work, these control law coefficients are set to k_p = 4.0 and k_v = 0.2.

2.4. Neural Network Controller

In this work, we investigate a robot system where the controller is replaced with an Artificial Neural Network. The diagram of this control system is shown in Figure 4. Let us define the input vector of the NN controller as:

$$X = [x_1, \ldots, x_m] \tag{4}$$

where x_i represents the i-th input and m is the number of the inputs.

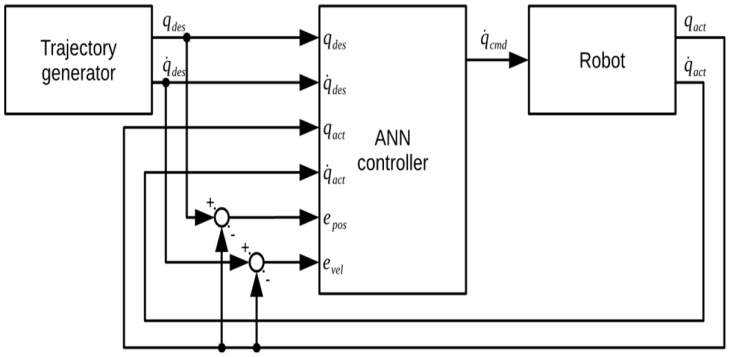


Figure 4. The online trajectory control with ANN controller inputs.

According to the scheme presented in Figure 4, the vector X can be written as:

$$X = \left[q_{des}, \dot{q}_{des}, q_{act}, \dot{q}_{act}, e_{pos}, e_{vel}\right]_{1x36} \tag{5}$$

The controller output Y is defined as follows:

$$Y = \left[\dot{q}_{cmd}\right]_{1x6} \tag{6}$$

where $\dot{q}_{cmd}$ is the velocity command sent to the robot.

Let us denote a neural network N as a tuple of a weight matrix W and the bias matrix ϑ:

$$N = (W, \vartheta) \tag{7}$$

The activation A of the i-th neuron in the l-th layer of the network can be calculated as:

$$A_{li} = \sum W_{lij} O_{l-1,j} + \vartheta_{li} \tag{8}$$

where W_{lij} is the weight of the connection between the neurons I and j, O_{l-1} is the output of the preceding layer, and ϑ_{li} is the neuron bias. The output of the i-th neuron in the l-th layer is calculated based on its activation as:

$$O_{li} = f_{li}(A_{li}) \tag{9}$$

where O_{li} is the output of the i-th neuron in the l-th layer, f_{li} is the activation function of the respective neuron, and A_{li} is its activation value.

In this work, we have constructed the NN architecture as presented in Figure 5. The neural network controller has 36 inputs, 6 outputs, and two hidden layers: (1) 10 neurons with *satlin* activation function, and (2) 6 neurons with *purelin* activation function.

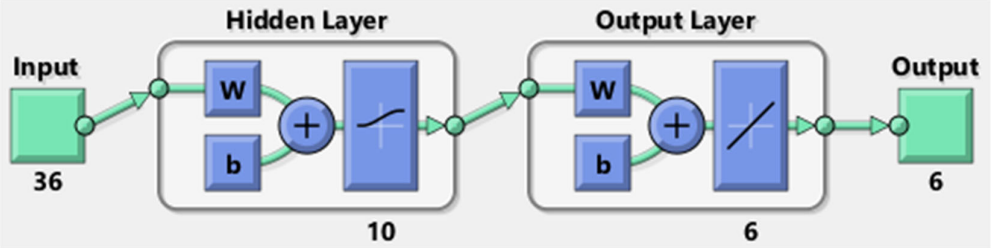


Figure 5. Neural network architecture.

The ANNs are optimized and compared using the Levenberg–Marquardt and Bayesian Regression learning methods, which are described in the following.

The Levenberg–Marquardt (*LM*) algorithm is a popular optimization method used for nonlinear regression problems and is commonly used to train artificial neural networks due to its ability to efficiently optimize complex nonlinear functions. It is an extension of the Gauss–Newton algorithm and adds a damping term to the Hessian matrix to improve its stability. The *LM* algorithm is used in the context of UR5 trajectory control to minimize the differences between the predicted and actual joint angles during robot motion along a trajectory. The algorithm iteratively updates the weights and biases of the ANN until convergence is achieved. During each iteration, the Jacobian matrix and the error vector are computed to determine the weight update. The damping parameter λ is a hyperparameter that controls the trade-off between the Gauss–Newton and steepest descent directions. A smaller damping parameter results in a more Gauss–Newton-like update, while a larger damping parameter results in a steeper descent-like update. The weight and bias updates in *LM* can be computed mathematically with the equation [26]:

$$\Delta w = \left[J^T J + \mu I \right]^{-1} J^T e \tag{10}$$

where Δw is the weight update, J is the Jacobian matrix, e is the error vector, μ is the learning rate which is to be updated using the β depending on the outcome. μ is multiplied by the decay rate β ($0 < \beta < 1$), and I is the identity matrix. The weight and bias values are then updated using the computed weight update.

Bayesian Regression (*BR*) is a probabilistic approach to regression analysis that allows for the estimation of posterior distributions for the model parameters. In the context of neural networks, *BR* can be used to estimate the uncertainty associated with the network predictions. *BR* involves computing the posterior distribution of the model parameters

given the training data and any prior knowledge about the parameters. This can be expressed using Bayes' rule as [27]:

$$p(\Theta \vee D) \propto p(\Theta)p(D|\Theta) \tag{11}$$

where Θ represents the model parameters, D represents the training data, while $p(D|\Theta)$ and $p(\Theta)$ represent the likelihood and prior distributions, respectively. The posterior distribution can be used to make predictions and estimate the uncertainty associated with those predictions. Bayesian regression can be particularly useful in the context of trajectory control of the UR5 robot, where it can estimate the uncertainty in the predicted joint angles during the robot's motion along a trajectory. The algorithm updates the weights and biases of the neural network using a probabilistic model that incorporates prior distributions over the parameters. These prior distributions can be set to reflect any prior knowledge or assumptions about the parameters.

Our proposed approach is summarized in Figure 6. We first implement a trajectory generation module, where a planar trajectory is designed in the task space of the robot and translated into the configuration space representation. We perform a set (N = 10) of experiments using a real UR5 with a classical control scheme in order to collect the training dataset for the network. As the control cycle of the robot is executed at a frequency of 125 Hz and the trajectory length is 10 s, we obtain a set of $10 \times 10 \times 125 = 12{,}500$ training samples. The collected data are used to train a neural network model using the selected training algorithms (LM and BR). The obtained NN models are used for the experimental verification of the control scheme in simulation and using the real robot setup.

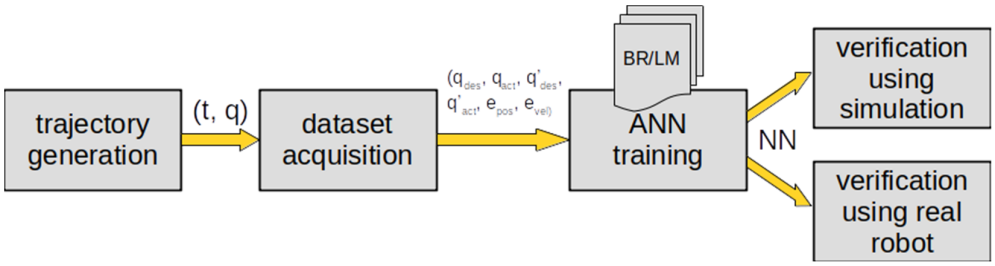


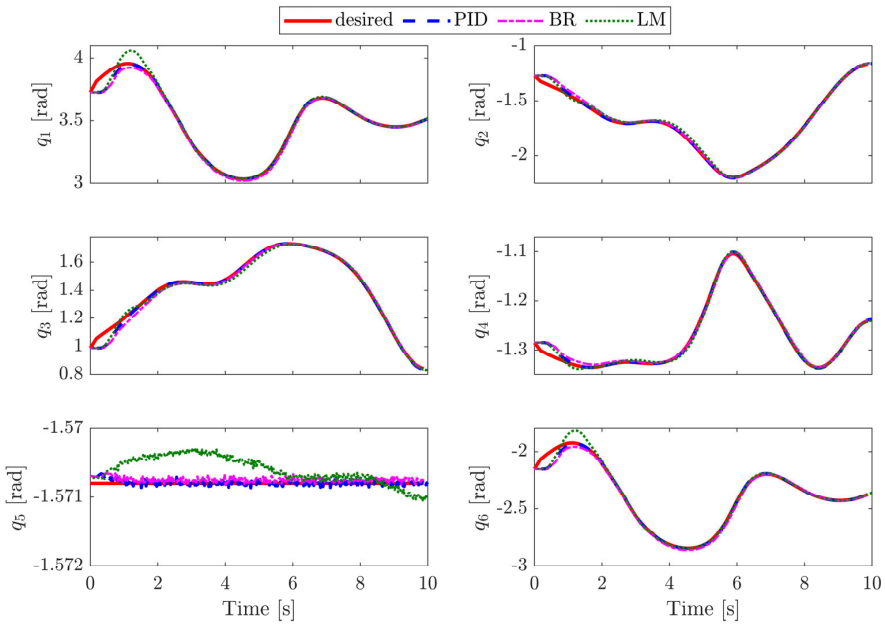
Figure 6. Summary of the proposed approach.

3. Experimental Results

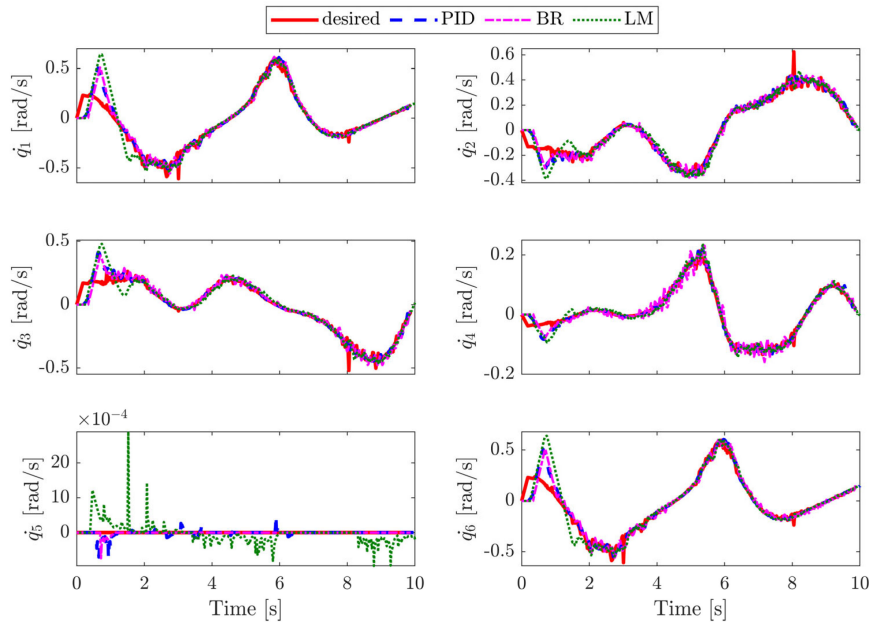
In this section, we describe the experimental setting and the tests executed to validate the performance of the considered controllers. The experiments were performed using two platforms: the URSim simulator (developed by Universal Robots) and a real UR5 robot available at the Bialystok University of Technology, Robotics Laboratory (Figure 1). The robot motion is presented as a sequence of (t, x, y, z) tuples representing the desired TCP frame position at the given time t. For the purpose of the direct robot control, an Inverse Kinematics model was used to map the task-space coordinates to the desired joint space coordinate sequence.

The experiments using the classical control serve a two-fold purpose: first to provide a baseline against which we compare the results achieved with ANN control, and secondly to generate trajectory, control, and feedback signals used to train the ANN controllers based on the results. The training and testing set for the training of the networks was created based on the data collected in the classical control experiment on the real UR5 robot. We repeated this experiment two times, using different state-of-the-art ANN learning methods: Levenberg–Marquardt (*LM*) and Bayesian Regression (*BR*). For both these approaches, we performed the experiment in simulation and on the real robot. Both the classical and the ANN control scheme were implemented in Matlab for our experiment. The implementation performed well, and we had no issue with matching the desired control cycle frequency of the robot (125 Hz).

The experimental results are shown in Figure 7, where the desired and actual joint positions and velocities during the trajectory execution using the classical controller and the developed ANN controllers are reported. The trajectory tracking performance achieved with the ANN controllers is nearly the same with respect to the classical PID controller, showing the feasibility of the proposed approach and the good tracking capabilities of the proposed controllers based on neural networks. Some noise can be observed in both position and velocity. It is worth pointing out the apparent discrepancy eminent in the case of the q_5 joint, for which no movement was defined in the testing scenario. The discrepancy is an artifact of the automatic zoom level of the plot, and the error is not notably larger than for the other joints.



(**a**)



(**b**)

Figure 7. Desired and actual joint positions and velocities during the trajectory execution using the classical controller and the developed ANN controllers: (**a**) joint positions; (**b**) joint velocities.

Tables 2 and 3 report the root mean square error (RMSE) of the position and velocity tracking for the compared controllers, respectively. As can be seen from the values in the tables, the performance of the compared controller is similar and comparable both for the position and for the velocity tracking. In particular, the mean RMSE value of the position tracking for the classical PID controller is equal to 0.0117 rad, whereas the *BR* and *LM* controller show values of 0.0148 rad and 0.0159 rad, respectively. With regard to velocity tracking, the PID control shows a RMSE value equal to 0.0361 rad/s, the *BR* controller to 0.0364 rad/s, and the *LM* controller to 0.0533 rad/s. The ANN learning based on the *BR* algorithm gives slightly better performance compared with the *LM* algorithm, according to both position and velocity tracking.

Table 2. RMSE of the position tracking for the compared controllers (in rad).

	Controller		
	PID	**BR**	**LM**
q_1	0.0197	0.0221	0.0291
q_2	0.0121	0.0177	0.0171
q_3	0.0154	0.0213	0.0165
q_4	0.0036	0.0051	0.0038
q_5	3.16×10^{-5}	5.72×10^{-5}	2.83×10^{-4}
q_6	0.0197	0.0225	0.0288
mean	0.0117	0.0148	0.0159

Table 3. RMSE of the velocity tracking for the compared controllers (in rad/s).

	Controller		
	PID	**BR**	**LM**
$\dot{q}_1$	0.0593	0.0597	0.0984
$\dot{q}_2$	0.0398	0.0400	0.0505
$\dot{q}_3$	0.0446	0.0413	0.0550
$\dot{q}_4$	0.0135	0.0167	0.0175
$\dot{q}_5$	8.56×10^{-5}	5.20×10^{-5}	2.68×10^{-4}
$\dot{q}_6$	0.0594	0.0606	0.0984
mean	0.0361	0.0364	0.0533

4. Conclusions

In this paper, different learning methods based on Artificial Neural Networks (ANNs) have been examined to replace the default speed controller for high-precision position control and drift attenuation in robotic manipulators. ANN learning methods including Levenberg–Marquardt and Bayesian Regression have been implemented and compared using a UR5 robot with six degrees of freedom to improve the trajectory tracking and minimize the position error. Extensive simulation and experimental tests on the identification and control of the robot by means of the neural network controllers yielded comparable results with respect to the classical controller, showing the feasibility of the proposed approach. It can be noted that while we have not achieved improvement in the positional accuracy of the TCP using the NN-based control, the strength of the proposed approach lies rather in the power of generalization and adaptation. As the kinematic parameters of individual robot arms vary ever so slightly between different units, we expect the NN controller to compensate for these differences and perform without the need for tuning which

could be the case with the classical control scheme. We admit that such generalization can only be achieved with a large enough dataset, which will be our focus in our further work.

We had noted no technical difficulties of implementing the ANN controller even in the prototype configuration based on Matlab. The implementation was able to match the hardware robot control cycle frequency of 125 Hz without issue. We expect that a more optimized implementation based on a performance-focused software platform will perform even better. However, further experiments are necessary to attest that the proposed control scheme will work well enough for the new versions of the UR robots incorporating a 500 Hz control cycle.

In the future, we plan to compare the results of our approach with alternative models, as for instance, machine learning using classical methods. We will also investigate the possibility of applying ANN control to more complex robotics applications, as for instance in real collaborative tasks [28]. Furthermore, ANN control will be applied to address complex structural vibration problems in the case of lightweight and flexible robotic arms.

Author Contributions: Conceptualization, A.M.; methodology, A.W., A.M., L.S. and N.K.; software, A.W. and N.K.; validation, A.M., A.W. and M.S.; formal analysis, A.M. and A.W.; investigation, A.M., A.W., N.K. and L.S.; resources, M.S.; data curation, N.K.; writing—original draft preparation, N.K.; writing—review and editing, A.M., A.W. and L.S.; visualization, N.K., A.M., A.W. and M.S.; supervision, A.M. and L.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was supported by the Bialystok University of Technology project (no: WZ/WE-IA/4/2023) financed by a subsidy provided by the Ministry of Science and Higher Education and by National Science Centre in Poland, grant No. 2020/37/B/ST7/03280.

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

Nomenclature

ADISE	Angle and Deflection Integral of the Squared Error
AGA	Adaptive Genetic Algorithm
AI	Artificial Intelligence
ANFIS	Adaptive Neuro-Fuzzy Interface System
ANN	Artificial Neural Network
BR	Bayesian Regression
D-H	Denavit-Hartenberg
DOF	Degree Of Freedom
FWNN	Fuzzy Wavelet Neural Network
HAISE	Hub-Angle Integral of the Squared Error
IK	Inverse Kinematics
LM	Levenberg Marquardt
MLP	Multi-Layer Perceptron
PD	Proportional Derivative
PID	Proportional Integral Derivative
PTP	Point to Point
RBF	Radial Basis Function
RISE	Robust Integral of the Sign of the Error
RKGNN	Runge–Kutta–Gill Neural Network
RMSE	Root Mean Square Error
RPY	Roll Pitch Yaw

TCP	Tool Centre Point
TDISE	Tip-Deflection Integral of the Squared Error
UR	Universal Robots
List of Symbols	
a, α, d, θ	parameters of the Denavit–Hartenberg convention
A_{li}	activation of the i-th neuron of the l-th layer of the network
A_x, A_y	major and minor axis parameters of the Lissajous curve
B	decay rate
CS_i	cubic spline interpolator function
Θ	tilt parameter of the Lissajous curve
Θ	bias matrix
Θ	model parameters
D	training data
E	error vector
e_{pos}	position error
e_{vel}	velocity error
I	identity matrix
J	Jacobian matrix
k_p, k_v	controller coefficients
μ	learning rate
N	neural network as a tuple of a weight matrix and a bias matrix
O_{l-1}	output of the preceding layer of the network
$p\,(D\|\Theta)$	likelihood distribution
$p(\Theta)$	prior distribution
P_i	desired pose at time t_i
Q	robot configuration
q_0	initial robot configuration
q_{act}	actual robot joint configuration
q_{cmd}	controller output
q_{des}	desired robot joint configuration
t	time
T	total time of the trajectory
TP_i	pair of time instant and trajectory way point in the Cartesian space
TQ_i	pair of time instant and trajectory way point in the joint space
x, y, z	coordinates in the Cartesian space
x_0, y_0, z_0	coordinates of the origin in the robot workspace
X	input vector of the Neural Network controller
Y	output vector of the Neural Network controller
ω_x, ω_y	angular frequency parameters of the Lissajous curve
w	weight
W	weight matrix

References

1. Sharma, R.; Gaur, P.; Mittal, A. Performance analysis of two-degree of freedom fractional order PID controllers for robotic manipulator with payload. *ISA Trans.* **2015**, *58*, 279–291. [CrossRef] [PubMed]
2. Köker, R. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Inf. Sci.* **2013**, *222*, 528–543. [CrossRef]
3. Almusawi, A.R.J.; Dülger, L.C.; Kapucu, S. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242). *Comput. Intell. Neurosci.* **2016**, *2016*, 5720163. [CrossRef] [PubMed]
4. El-Sherbiny, A.; Elhosseini, M.; Haikal, A. A comparative study of soft computing methods to solve inverse kinematics prob-lem. *Ain Shams Eng. J.* **2018**, *9*, 2535–2548. [CrossRef]
5. Varedi-Koulaei, S.M.; Mokhtari, M. Trajectory Tracking Solution of a Robotic Arm Based on Optimized ANN. In Proceedings of the 2018 6th RSI International Conference on Robotics and Mechatronics, Tehran, Iran, 23–25 October 2018; pp. 76–81. [CrossRef]
6. Ak, A.; Topuz, V.; Ersan, E. Visual servoing application for inverse kinematics of robotic arm using artificial neural net-works. *Stud. Inform. Control* **2018**, *27*, 183–190. [CrossRef]

7. Jiang, Z.-H.; Ishita, T. A Neural Network Controller for Trajectory Control of Industrial Robot Manipulators. *J. Comput.* **2008**, *3*, 1–8. [CrossRef]
8. Adar, N.G. Real Time Control Application of the Robotic Arm Using Neural Network Based Inverse Kinematics Solution. *Sak. Univ. J. Sci.* **2021**, *25*, 849–857. [CrossRef]
9. Deif, S.; Tawfik, M.; Kamal, H.A. Optimal Tuning of a PD Controller for a Single-Link Flexible Robot Arm Using Adaptive Genetic Algorithm. In Proceedings of the 14th International Conference on Aerospace Sciences and Aviation Technology (ASAT-14), Cairo, Egypt, 24–26 May 2011. [CrossRef]
10. Duka, A.-V. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Procedia Technol.* **2014**, *12*, 20–27. [CrossRef]
11. Nanayakkara, T.; Watanabe, K.; Kiguchi, K.; Izumi, K. Fuzzy self-adaptive radial basis function neural network-based control of a seven-link redundant industrial manipulator. *Adv. Robot.* **2001**, *15*, 17–43. [CrossRef]
12. Jiang, Z.H.; Ishida, T. Trajectory tracking control of industrial robot manipulators using a neural network controller. In Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics, Montreal, QC, Canada, 7–10 October 2007; pp. 2390–2395. [CrossRef]
13. Lima, T.L.d.V.; de Freitas, I.S.; Filho, J.B.d.M.; Sobrinho, C.A.O.N.; da Silva, J.F. Development and neural control of a robotic manipulator with two degrees of freedom. In Proceedings of the Electrodynamic and Mechatronic Systems, Opole, Poland, 6–8 October 2011. [CrossRef]
14. Wang, J.; Zhu, Y.; Qi, R.; Zheng, X.; Li, W. Adaptive PID control of multi-DOF industrial robot based on neural network. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 6249–6260. [CrossRef]
15. Zhao, Y.; Cheah, C.C. Neural Network Control of Multi fingered Robot Hands Using Visual Feedback. *IEEE Trans. Neural Netw.* **2009**, *20*, 758–767. [CrossRef] [PubMed]
16. Yang, C.; Teng, T.; Xu, B.; Li, Z.; Na, J.; Su, C.Y. Global adaptive tracking control of robot manipulators using neural networks with finite-time learning convergence. *Int. J. Control. Autom. Syst.* **2017**, *15*, 1916–1924. [CrossRef]
17. Yen, V.T.; Nan, W.Y.; Van Cuong, P.; Quynh, N.X.; Thich, V.H. Robust adaptive sliding mode control for industrial robot manipulator using fuzzy wavelet neural networks. *Int. J. Control. Autom. Syst.* **2017**, *15*, 2930–2941. [CrossRef]
18. Lv, Y.F.; Huang, Y.B.; Wu, X.L.; Jian, L. Neural Network Tracking Controls of SCARA Manipulator System. In Proceedings of the 33rd Chinese Control and Decision Conference (CCDC 2021), Kunming, China, 22–24 May 2021; pp. 329–333.
19. Ge, Y.W.; Zhou, J.; Deng, W.X.; Yao, J.Y.; Xie, L. Neural network robust control of a 3-DOF hydraulic manipulator with asymp-totic tracking. *Asian J. Control.* **2021**, *25*, 2060–2073.
20. Li, Z.; Li, S. Neural Network Model-Based Control for Manipulator: An Autoencoder Perspective. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 2854–2868. [CrossRef] [PubMed]
21. Nguyen, H.-T.; Cheah, C.C. Analytic Deep Neural Network-Based Robot Control. *IEEE/ASME Trans. Mechatronics* **2022**, *27*, 2176–2184. [CrossRef]
22. Universal Robots. User Manual, Ver. 1.6, UR5 with CB2, US ver. Available online: https://www.universal-robots.com/download/manuals-cb-series/user/ur5/16/user-manual-ur5-cb-series-sw16-english-us/ (accessed on 30 August 2023).
23. Kebria, P.M.; Al-Wais, S.; Abdi, H.; Nahavandi, S. Kinematic and dynamic modelling of UR5 manipulator. In Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 9–12 October 2016; pp. 4229–4234.
24. Dulęba, I.; Opałka, M. A comparison of Jacobian-based methods of inverse kinematics for serial robot manipulators. *Int. J. Appl. Math. Comput. Sci.* **2013**, *23*, 373–382. [CrossRef]
25. URScript manual (n.d.). URScript Manual. Available online: https://s3-eu-west-1.amazonaws.com/ur-supportsite/32554/scriptManual-3.5.4.pdf (accessed on 31 May 2021).
26. Suratgar, A.A.; Tavakoli, M.B.; Hoseinabadi, A. Modified Levenberg-Marquardt Method for Neural Networks Training. *World Acad. Sci. Eng. Technol. Int. J. Comput. Inf. Eng.* **2007**, *1*, 1745–1747.
27. de la Cruz, R.; Padilla, O.; Valle, M.A.; Ruz, G.A. Modeling Recidivism through Bayesian Regression Models and Deep Neu-ral Networks. *Mathematics* **2021**, *9*, 639. [CrossRef]
28. Scalera, L.; Giusti, A.; Vidoni, R.; Gasparetto, A. Enhancing fluency and productivity in human-robot collaboration through online scaling of dynamic safety zones. *Int. J. Adv. Manuf. Technol.* **2022**, *121*, 6783–6798.