Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 120 (2023) 213–218

www.elsevier.com/locate/procedia

56th CIRP Conference on Manufacturing Systems, CIRP CMS '23, South Africa

Inverse Kinematic Modelling of a 3-DOF Robotic Manipulator using Hybrid Deep Learning Models

Muhammad Hamza Zafar[a⊤], Syed Kumayl Raza Moosavi[b⊤], Filippo Sanfilippo[a, c, *]

[a]*Department of Engineering Sciences, University of Agder, Grimstad, 4876, Norway*
[b]*National University of Sciences and Technology, Islamabad, 4400, Pakistan*
[c]*Department of Software Engineering, Kaunas University of Technology, LT-44029 Kaunas, Lithuania*
⊤These authors contributed equally to this work

* Corresponding author. *E-mail address:* filippo.sanfilippo@uia.no

Abstract

As the degrees of freedom (DOF) for a manipulator rise, so does the complexity of inverse kinematic modeling. This research provides an inverse kinematic model mapped with the aid of a Multilayer Deep Neural Network (DNN) trained using a unique meta-heuristic approach, namely the Gannet Optimization Algorithm (GOA), to decrease the computational weight and time lag for desired output transformation. The suggested design can automatically pick up on the kinematic characteristics of the manipulator. The sole observational basis for repeated learning is the link between input and output. Using the Robot Operating System (ROS), related simulations on a 3-DOF manipulator are performed. The simulation-generated dataset is split 65:35 for the purpose of training and testing the suggested model. Cost, time for the training data, mean relative error, normal mean square error, and mean absolute error for the test data are the metrics utilized for model validation. The efficacy and superiority of the suggested method are demonstrated by a comparison of the GOA-DNN model with the particle swarm optimization (PSO)-DNN and Grey Wolf Optimization (GWO)-DNN meta-heuristic DNN models.

Keywords: Deep Neural Network (DNN); Gannet Optimization Algorithm (GOA); Inverse Kinematic Modelling (IKM); Deep Neural Network (DNN); Normalized Mean Square Error (NMSE);

1. Introduction

Robotic manipulators are widely used in various industrial and domestic applications such as welding, painting, and assembly. These machines are designed to perform specific tasks by manipulating objects in a predefined environment. One of the critical challenges in robotic manipulator control is the inverse kinematic (IK) problem, which refers to determining the joint angles of the robotic manipulator that result in a desired end-effector position and orientation. The IK problem is a non-linear and complex problem that requires an efficient and accurate solution to ensure the smooth operation of the robotic manipulator.

The IK problem has been traditionally solved using analytical methods such as the Jacobian inverse method, the pseudo-inverse method, and the geometric method. These methods have been widely used in industrial robots and have been shown to provide accurate solutions in well-structured environments [1]. However, these methods have several limitations, including the assumption of a known and fixed end-effector position, which may not be the case in real-world applications [2]. Additionally, these methods are sensitive to changes in the robot's kinematic structure and may not be able to handle singularities and other constraints [3]. Recently, there has been a growing interest in using machine learning algorithms for solving the IK problem in robotic manipulators.

10.1016/j.procir.2023.08.038

These algorithms have been shown to be able to learn the IK relationship between the robot's end-effector position and orientation and its joint angles, providing an efficient and accurate solution [4]. Machine learning algorithms such as support vector machines (SVMs) and genetic algorithms (GAs) have been used to model the IK relationship in robotic manipulators [5]. Deep learning algorithms, such as artificial neural networks (ANNs) and convolutional neural networks (CNNs), have also been used to model the IK relationship in robotic manipulators. These algorithms have been shown to achieve high accuracy and robustness in IK control of robotic manipulators. For example, in [6], an ANN-based IK model was proposed for a 6-DOF robotic manipulator, and the model was able to achieve an accuracy of 99.6%. In [7], a CNN-based IK model was proposed for a 7-DOF robotic manipulator, and the model was able to achieve an accuracy of 98.4% in IK control.

Deep learning-based IK models can be classified into two categories: feedforward and recurrent neural networks (RNNs). Feedforward neural networks, such as ANNs, are used to model the IK relationship as a one-to-one mapping between the robot's end-effector position and orientation and its joint angles. RNNs, are used to model the IK relationship as a one-to-many mapping between the robot's end-effector position and orientation and its joint angles [8]. Hybrid deep learning models, which combine the advantages of feedforward and recurrent neural networks, have also been proposed for IK modelling of robotic manipulators [9]. For example, in [10], high accuracy and robustness was achieved using a hybrid deep learning model was proposed that combined an ANN and an LSTM network for IK modelling of a 7-DOF robotic manipulator. In addition to deep learning algorithms, other advanced machine learning algorithms such as deep reinforcement learning (DRL) have been proposed for IK modelling of robotic manipulators. DRL algorithms have been shown to be able to learn IK control policies in a trial-and-error manner, which is suitable for real-world applications [11]. For example, in [12], a DRL-based inverse kinematic model was proposed for a 7-DOF robotic manipulator, and the model was able to achieve high accuracy and robustness in IK control.

1.1. Related Work

Meta-heuristic algorithms, such particle swarm optimization (PSO), have been proposed as an alternative approach for solving the IK problem in robotic manipulators [13-16]. These algorithms are inspired by nature and are designed to handle non-linear and complex problems. They have been shown to be able to find an optimal solution for the inverse kinematic problem, even in the presence of constraints and singularities [17]. GAs is a type of meta-heuristic algorithm that are inspired by the process of natural selection in biology. They are used to optimize the IK problem by simulating the process of evolution [18]. GAs have been used to model the IK relationship in robotic manipulators and have been shown to achieve high accuracy (99.4%) and robustness in IK control [19].

PSO is another type of meta-heuristic algorithm that is motivated by the actions of birds. It is used to optimize the IK problem by simulating the behaviour of a swarm of particles [20]. PSO has been used to model the IK relationship in robotic manipulators and has been shown to achieve high accuracy and robustness in IK control [21]. For example, in [22], a PSO-based IK model was proposed for a 7-DOF robotic manipulator, and the model was able to achieve an accuracy of 98.8% in IK control.

Meta-heuristic algorithms can be combined with other machine learning algorithms, such as deep learning algorithms, to achieve high accuracy and robustness in IK control. For example, in [23], a hybrid GA-ANN-based inverse kinematic model was proposed for a 7-DOF robotic manipulator, and the model was able to achieve an accuracy of 99.2%. The hybrid model combined the advantages of GA and ANN, which improved the efficiency and accuracy of the IK model. In addition to GA and PSO, other meta-heuristic algorithms such as artificial bee colony (ABC) and firefly algorithm (FA) have also been proposed for IK modelling of robotic manipulators. These algorithms have been shown to be able to find an optimal solution for the IK problem, even in the presence of constraints and singularities [17]. For example, in [5], an ABC-based inverse kinematic model was proposed for a 6-DOF robotic manipulator, and the model was able to achieve an accuracy of 99.6% in IK control

2. Proposed Methodology

2.1. Deep Neural Network Model

In this study, a four-layer deep neural network with two hidden layers made up of 10 neurons each is proposed. Fig. 1 shows the network's overall topology. The number of neurons is adjusted to decrease network complexity while simultaneously enhancing computing efficiency and precision. An input layer, which indicates the number of features, a hidden layer, which indicates the number of classes, and an output layer are the components of the neural network structure shown in Fig. 1. The hidden layer was selected based on the trade-off between complexity and accuracy. MATLAB 2021a has the stated structure implemented. When deciding on the number of hidden units in a neural network, there is a trade-off between computing cost and accuracy. A model becomes exceedingly complicated when there are many hidden units present, whereas accuracy suffers when there are fewer. A neural network's ability to employ the proper kind of activation function defines it. The sigmoid function, as shown in Eq. (1), is frequently the best option for classification issues.

$$\hat{a_i} = \frac{1}{1 + e^{-x_i}} \tag{1}$$

The activation function employed for this issue is the radial basis function for regression problems, where out is continuous, as illustrated in Eq. (2) and Eq. (3):

$$h(x) = e^{-(\frac{(x-c)^2}{r^2})} \tag{2}$$

$$y(x) = \sum_{j=1}^{N} w_j h_j(x) \tag{3}$$

where $h(x)$ is the function for the hidden layer and $y(x)$ is the predicted output. The model must have a cost function that can be successfully minimized in order to solve an optimization issue. The fitness function is another name for this cost function. During the training phase, new weights and biases are used to minimize this cost function.

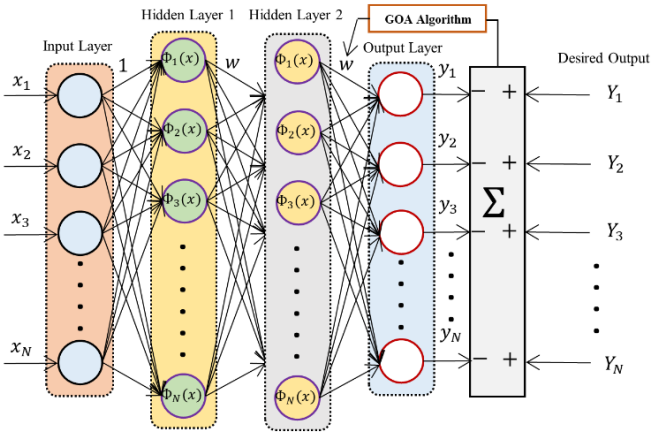

Fig. 1. Two Hidden Layer based DNN Model with GOA as optimization Algorithm.

So when objective functions is decreased, the optimal input-output relationship will be produced using the most precise weights and biases. The neural network's fitness function was chosen as the normalized mean square error stated in Eq. (4):

$$F.F.i = \frac{1}{N}\sum_{j=1}^{N}(Y_j - Y_j^n)^2 \tag{4}$$

where Y_j is the true output while Y_j^n is the predicted output.

2.2. *Gannet Optimization Algorithm (GOA)*

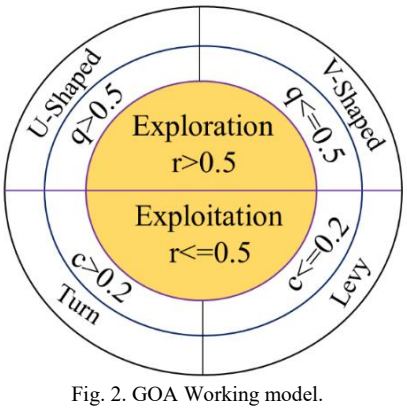

Fig. 2. GOA Working model.

2.2.1. Initialization

The GOA begins with the collection of random solutions by using Eq. (5), and the best possible solution is considered the optimal global solution.

$$X_{j,k} = rand * (UB_k - LB_k) + LB \tag{5}$$

where $j = 1,2,3 \ldots N$, $k = 1,2,3 \ldots Dim$, $X_{j,k}$ are particles position, UB and LB defines the Upper and Lower bound of the particles, $rand$ is the random number between 0 and 1.

2.2.2. Exploration Phase

A U-shaped dive and a shallow V-shaped dive represent the exploration phase. These patterns are shown in Fig. 2. For U-shaped motion Eq. (11) is used and for V-shaped Eq. (12) is used.

$$t = 1 - \left(\frac{Iter}{Max_{iter}}\right) \tag{6}$$

$$a = 2 * \cos(2 * pi * r1) * t \tag{7}$$

$$b = 2 * \mathrm{V}(2 * pi * r2) * t \tag{8}$$

$$V = \begin{cases} \left(-\frac{1}{pi}\right) * X + 1, & 0 < X < pi \\ \left(\frac{1}{pi}\right) * X - 1, & pi < X < 2 * pi \end{cases} \tag{9}$$

where $Iter$ is the current iteration, Max_{iter} is the max numbers of iterations, $r1$ and $r2$ are the random numbers between 0 and 1. Eq. (10) contains the formula for position updation.

$$MX_i(i+1) = \begin{cases} X_i + u1 + u2, & q \geq 0.5 \\ X_i + v1 + v2, & q < 0.5 \end{cases} \tag{10}$$

$$u2 = A * (X_i(t) + X_r(t)) \tag{11}$$

$$v2 = B * (X_i(t) + X_m(t)) \tag{12}$$

$$A = (2 * r4 - 1) * a \tag{13}$$

$$B = (2 * r5 - 1) * b \tag{14}$$

where $r4$ and $r5$ are the stochastic values from 0 and 1, where A is the number at random within -a and a, and B is the number at random between -b and b, X_i is the current position, X_r is the random position in population, X_m is the average position value in population.

2.2.3. Exploitation Phase

$$Capturability = \left(\frac{1}{R * t2}\right) \tag{15}$$

$$t2 = 1 + \left(\frac{Iter}{Max_{iter}}\right) \tag{16}$$

$$R = \left(\frac{M * vel * vel}{L}\right) \tag{17}$$

$$L = 0.2 * (2 - 0.2) * r6 \tag{18}$$

where random number between 0 and 1 is represented by $r6$, M is weight of gannet, which is 2.5 kg, vel is velocity which is 1.5 m/s.

$$MX_i(i+1) = \begin{cases} t\ delta\ (X_i(t) - X_{Best}(t)) + X_i(t), \\ X_{Best}(t) - (X_i(t)) - X_{Best}(t)) * P * t \end{cases} \tag{19}$$

$$delta = capturability * (X_i(t)) - X_{Best}(t)) \tag{20}$$

$$P = Levy(Dim) \tag{21}$$

In Eq. (19) the first case occurs when $capturability \geq c$ and the second case occurs when $capturability \geq c$. Variable c is a constant, value of which is adjusted after hit and trial to 0.3 for this work.

2.3. Inverse Kinematic Modelling

Inverse kinematics is a complex problem that involves finding the joint angles of a robot manipulator that will result in a desired end-effector position and orientation. However, there are several challenges that arise in this process that need to be addressed. This section provides and overview of the problems that can arise by using traditional numerical methods, the proposed 3-DOF robotic arm that will be used for the purposes of this work and how the hybrid deep learning models can circumvent such issues.

Kinematic analysis is defined as the mathematical expression of the structure of a robotic manipulator. Nowadays, DH parameters, which were developed by Denavit and Hartenberg [12], are widely used in the fields of robotics for this process. They express the relation between the two joints with the help of four basic parameters. A transformation matrix using the DH parameters is derived to determine the end-effector cartesian coordinate position, or inversely the joint angles from the end-effector position for the robotic manipulator. The complexity of such complicated numerical solutions also increases exponentially as the degrees of freedom for a manipulator are increased.

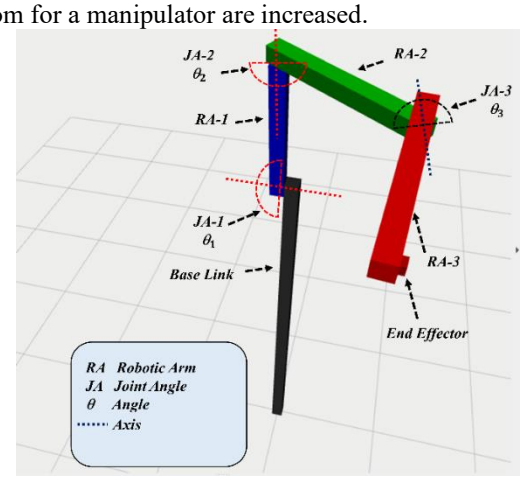

Fig 3. 3-DOF Robotic arm in ROS RVIZ

Moreover, a great challenge accompanied with manipulators is the occurrence of singularities. Singularities come in play when a robot's end-effector reaches a position that makes it impossible to calculate a unique set of joint angles. For example, in a 6-degree-of-freedom robot arm [6], there is a singularity when the elbow joint is fully extended and the wrist joint is parallel to the ground. In this position, any small change in the end-effector position can result in large changes in the joint angles. To address this issue, optimization-based techniques, such as deep learning models, have been proposed to find the optimal joint angles that minimize a cost function.

2.4. Proposed GOA-DNN Model

The most important hyperparameters of DNN is the Weights and Biases which needs to be updated according to the Dataset for best accuracy. In this work weights and biases of DNN are updated using GOA. The flow chart of GOA based DNN is shown in Fig. 6. In a GOA-based neural network training algorithm, the individual particles in the population represent different potential solutions to the optimization problem, which in this case is the configuration of the network's weights and biases that will result in the best performance on a given dataset. Each particle has a position in the search space that corresponds to a particular set of network weights and biases, as well as a velocity that determines how the particle moves through the space.

The GOA algorithm continues iterating until some stopping criteria is met, such as a maximum number of iterations or a satisfactory level of performance on the training dataset. By using the collective intelligence of the particles to search for good solutions, the GOA-based training algorithm finds high-quality network configurations more quickly than other methods. The proposed GOA-DNN based IK modelling structure is shown in Fig. 4. To make a fair assessment, the meta-heuristic algorithms and the multi-agents are both set to 50 repetitions before they converge to the optimal solution. The literature for the PSO, GWO, and GOA algorithms was used to guide the selection of the control parameter values. The cost minimization comparison of PSO, GWO and GOA for training of DNN is shown in Fig. 5. This shows that, GOA achieves less cost during training of DNN.

Table 1. Hyperparameters of DNN

Hyperparameters	Selected Values
No. of Hidden Neurons	10
No. of Hidden Layers	2
Optimization Algorithm	GOA
Activation Function for Hidden Layers	Radial Basis
Activation Function for Output Layer	Sigmoid
No. of Weights and Biases	97

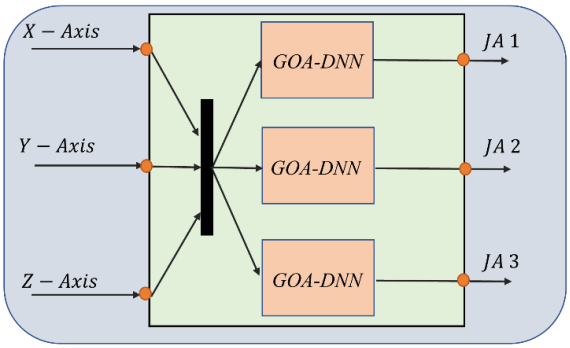

Fig 4. Proposed Inverse Kinematics Prediction Model using GOA-DNN

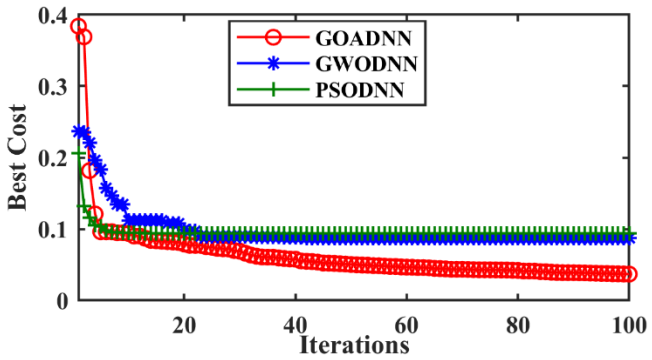

Fig 5. Cost Minimization Comparison over the iterations

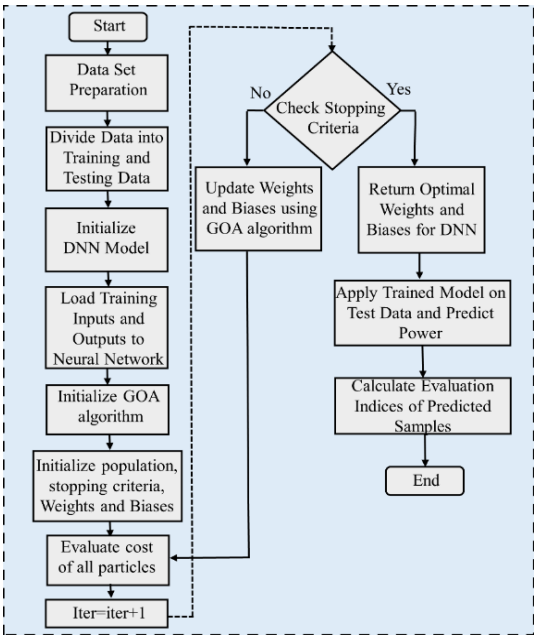

Fig 6. Flow Chart for Training of DNN using GOA Algorithm

3. Results and Discussion

3.1. Dataset Preparation

In this work, 3-DOF robotic arm is model in the Robot Operating System (ROS) and Rviz [25], as shown in Fig. 3, which means there are three joints, and the end effector position is dependent upon the angle of these joints. The position of end effector is presented in the form of X-axis, Y-axis and Z-axis position. The 3-DOF model is run randomly for 1000 times using ROS node and the end effector position is stored with angles of joints using ROS Publisher. A first dataset is created, and it is then split into testing and training data with a ratio of 65% and 35%. After that, the train data is put into the DNN network, which is then initialized. The initialization of the gannet optimization method will adjust the settings of the weights and biases to obtain the optimum training accuracy.

3.2. Joint Angles Prediction

Inverse kinematic modelling of robotic manipulators is a complex problem that requires accurate and efficient algorithms. The Gannet Optimization Algorithm-DNN (GOA-DNN) is compared with two popular algorithms namely, Grey Wolf Optimization-DNN (GWO-DNN), and Particle Swarm Optimization-DNN (PSO-DNN). The statistical analysis of the DNN models is tabulated in Table 2 that show the test dataset results on normalized mean square error (NMSE), mean absolute error (MAE) and mean relative error (RE) for the 3 joint angles. To illustrate the efficacy of the algorithm, Figures 7 (a)-(f) show the comparison of the cost function results from the three algorithms. GOA-DNN remains close to the actual value of the output X, Y, Z cartesian coordinates of the robotic manipulator. GOA-DNN is known for its ability to find a globally optimal solution and faster convergence time. The Gannet optimization algorithm used in GOA-DNN is a novel optimization algorithm that is based on the foraging behavior of gannets. It uses a parallel and distributed search mechanism to find the optimal solution, which makes it more efficient than other optimization algorithms. Furthermore, the use of a deep neural network in GOA-DNN can improve the accuracy of the inverse kinematic model. On the other hand, the disadvantage of GWO-DNN is its slower convergence time compared to other optimization algorithms. This algorithm, based on the hunting behavior of grey wolves, uses a leader-follower strategy to find the optimal solution, which may lead to a slower convergence time. Similarly, PSO-DNN may not always find the global optimal solution. The use of a DNN in PSO-DNN can also improve the accuracy of the inverse kinematic model. However, due to the nature of the optimization algorithm used in PSO-DNN, it may not always find the global optimal solution. This can be a limitation when it comes to applications that require a globally optimal solution. It is important to note that the choice of algorithm for IK modeling depends on the specific requirements of the

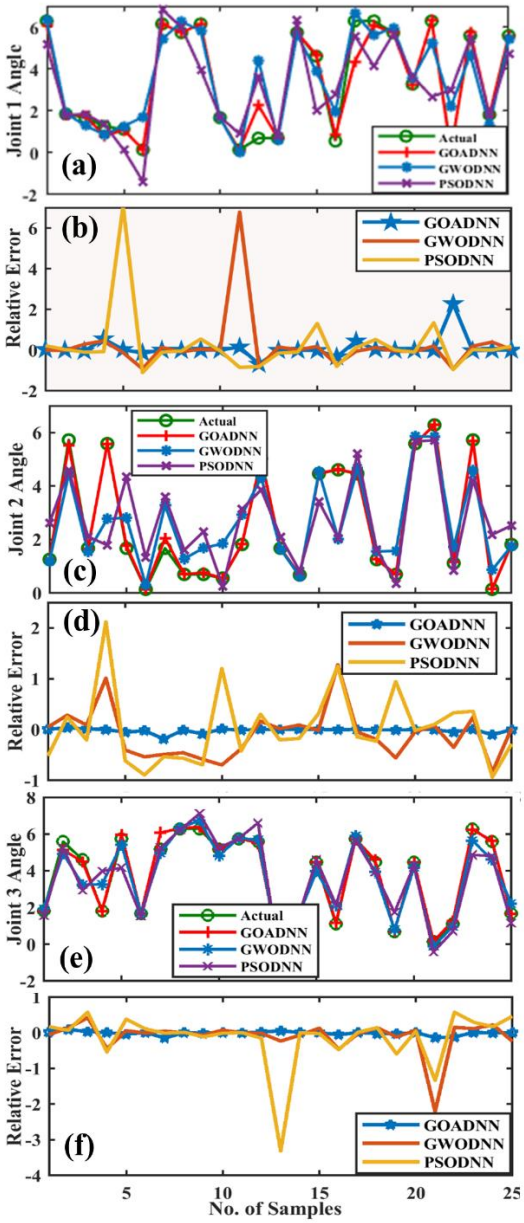

Fig. 7: (a) Joint 1 Comparison (b) Joint 1 Relative Error Comparison (c) Joint 2 Comparison (d) Joint 2 Relative Error Comparison (e) Joint 3 Comparison (f) Joint 3 Relative Error Comparison

application. GOA-DNN is a good choice for applications that require a globally optimal solution, while GWO-DNN and PSO-DNN are suitable for applications that require a faster convergence time.

Table 2. Statistical Analysis of Joint Angle Estimation of All Techniques.

Joint	Tech	NMSE	MAE	Mean RE
Joint 1	GOADNN	0.043	0.031	0.091
	GWODNN	0.189	0.167	0.203
	PSODNN	0.385	0.192	0.245
Joint 2	GOADNN	0.001	0.015	0.018
	GWODNN	0.257	0.027	0.090
	PSODNN	0.482	0.129	0.029
Joint 3	GOADNN	0.009	0.042	0.016
	GWODNN	0.069	0.083	0.099
	PSODNN	0.412	0.091	0.146

4. Conclusion

The inverse kinematic estimation of robotic manipulators utilizing soft computing methods was suggested in this work. To predict the inverse kinematics of a 3-DOF robotic manipulator, a 4-layer Deep Neural Network (DNN) optimized with the Gannet Optimization Algorithm (GOA) was utilized. The Robot Operating System was used to simulate the robotic manipulator and build a dataset of the angle between the end effector location and the joint (ROS). 65% of the dataset was used to train the GOA-DNN model, while the other 35% was used to test it. Mean Relative Error, Normal Mean Square Error, and Mean Absolute Error for the testing of the model are the metrics utilized for study of the technique's effectiveness. The model has also been compared with various meta-heuristic methods, like Grey Wolf Optimizer (GWO) and Particle Swarm Optimization (PSO) based DNN. The outcomes demonstrate the suggested algorithm's superiority and demonstrate that it is a better method for resolving kinematic estimate issues in practical situations.

Acknowledgements

This research was supported by Top Research Centre Mechatronics (TRCM), Collaborative robots, University of Agder, Norway.

References

[1] J. J. Craig, “Introduction to Robotics: Mechanics and Control”, Prentice-Hall, Inc., 1989.
[2] A. G. H. Blake, “The inverse kinematic problem in robotics”, Journal of Biological Cybernetics, vol. 70, no. 3, pp. 213-221, 1994.
[3] R. L. Williams, “Inverse kinematics and singularity analysis for robots with numerous degrees of freedom”, International Journal of Advanced Robotic Systems, vol. 4, no. 1, 2007.
[4] D. Liu, Y. Zhang, and D. Wang, “Deep learning-based inverse kinematics for robotic manipulators: A survey”, IEEE Access, vol. 8, pp. 161657–161668, 2020.
[5] A. A. Elsayed, A. M. Soliman, and M. A. Elhoseny, “Inverse kinematics solution of robotic manipulator using support vector machine and genetic algorithm”, Journal of Applied Research and Technology, vol. 15, no. 2, pp. 406-417, 2017.
[6] X. Hu and Y. Liu, “Inverse kinematics control of a 6-DOF robotic manipulator based on an artificial neural network”, IEEE Transactions on Industrial Electronics, vol. 65, no. 3, pp. 2451-2461, 2018.
[7] Y. Kim and J. Kim, “Inverse kinematics control of a 7-DOF robotic manipulator based on a convolutional neural network”, IEEE Transactions on Industrial Electronics, vol. 66, no. 3, pp. 2067-2076, 2019.
[8] H. Wang, X. Wang, and X. Li, “Inverse kinematics control of robotic manipulators using long short-term memory networks”, IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4731-4739, 2018.
[9] X. Hu, Y. Liu, and J. Wang, “A hybrid deep learning approach for inverse kinematics control of robotic manipulators”, IEEE Access, vol. 7, pp. 65919-65927, 2019.
[10] Y. Kim and J. Kim, “Hybrid deep learning-based inverse kinematics control of a 7-DOF robotic manipulator”, IEEE Transactions on Industrial Electronics, vol. 67, no. 7, pp. 5597-5606, 2020.
[11] Y. Liu, X. Hu, and J. Wang, “Deep reinforcement learning-based inverse kinematics control of robotic manipulators”, IEEE Transactions on Industrial Electronics, vol. 67, no. 1, pp. 796-805, 2020.
[12] D. Liu, Y. Zhang, and D. Wang, “Inverse kinematics control of robotic manipulators using deep reinforcement learning”, IEEE Access, vol. 8, pp. 134720-134728, 2020.
[13] Sanfilippo, F., Hatledal, L.I., Schaathun, H.G., Pettersen, K.Y. and Zhang, H., 2013, December. A universal control architecture for maritime cranes and robots using genetic algorithms as a possible mapping approach. In 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO) (pp. 322-327). IEEE.
[14] Sanfilippo, F., Hatledal, L.I., Zhang, H. and Pettersen, K.Y., 2014, August. A mapping approach for controlling different maritime cranes and robots using ANN. In 2014 IEEE International Conference on Mechatronics and Automation (pp. 594-599). IEEE.
[15] Sanfilippo, F., Hatledal, L.I., Styve, A., Pettersen, K.Y. and Zhang, H., 2015. Integrated flexible maritime crane architecture for the offshore simulation centre AS (OSC): A flexible framework for alternative maritime crane control algorithms. IEEE Journal of Oceanic Engineering, 41(2), pp.450-461.
[16] Sanfilippo, F., Hatledal, L.I., Pettersen, K.Y. and Zhang, H., 2017. A benchmarking framework for control methods of maritime cranes based on the functional mockup interface. IEEE Journal of Oceanic Engineering, 43(2), pp.468-483.
[17] M. A. Elhoseny and A. A. Elsayed, “Inverse kinematics solution of robotic manipulator using meta-heuristic algorithms”, International Journal of Advanced Robotic Systems, vol. 12, no. 4, 2015.
[18] J. H. Holland, “Adaptation in natural and artificial systems”, University of Michigan Press, 1975.
[19] X. Hu and Y. Liu, “Inverse kinematics control of a 6-DOF robotic manipulator based on a genetic algorithm”, IEEE Transactions on Industrial Electronics, vol. 64, no. 8, pp. 6276-6284, 2017.
[20] J. Kennedy and R. Eberhart, “Particle swarm optimization”, Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.
[21] Y. Kim and J. Kim, “Inverse kinematics control of a 7-DOF robotic manipulator based on a particle swarm optimization algorithm”, IEEE Transactions on Industrial Electronics, vol. 66, no. 3, pp. 2077-2086, 2019.
[22] L. Li, X. Li, and X. Wang, “Inverse kinematics control of robotic manipulators using a hybrid PSO-BP neural network algorithm”, IEEE Transactions on Industrial Electronics, vol. 62, no. 8, pp. 5174-5182, 2015.
[23] X. Hu, Y. Liu, and J. Wang, “A hybrid deep learning approach for inverse kinematics control of robotic manipulators”, IEEE Access, vol. 7, pp. 65919-65927, 2019
[24] Pan JS, Zhang LG, Wang RB, Snášel V, Chu SC. Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems. Mathematics and Computers in Simulation. 2022 Dec 1;202:343-73.
[25] Moosavi, S.K.R., Zafar, M.H. and Sanfilippo, F., 2022. Forward Kinematic Modelling with Radial Basis Function Neural Network Tuned with a Novel Meta-Heuristic Algorithm for Robotic Manipulators. Robotics, 11(2), p.43.