Journal of Mechatronics and Robotics

Research Article

A GUI-Enabled, Automated LSTM-Based Inverse Kinematics Pipeline for 6-DOF Robotic Arms Using MATLAB and CoppeliaSim

Waleed Abdulrahman Saleh Al-Akwa and Mohammed Abdulwahab Ahmed Daba

Department of Mechatronics Engineering, University of Sana'a, Sana'a, Yemen

Article history
Received: 18-01-2025
Revised: 02-04-2025
Accepted: 25-04-2025

Corresponding Author:
Waleed Abdulrahman Saleh Al-Akwa
Department of Mechatronics Engineering, University of Sana'a, Sana'a, Yemen
Email: waleed.alakwa@su.edu.ye

Abstract: Existing deep learning-based inverse kinematics (IK) solutions often target specific robotic arms and require significant modifications when applied to different configurations. To support early-phase design and testing of 6-degree-of-freedom (6-DOF) robotic arms, this study presents a fast and adaptable IK solution through a user-friendly interface. Unlike traditional numerical methods that are computationally intensive, sensitive to initial conditions, and may not generalize to custom designs, the proposed approach allows users to input Denavit-Hartenberg (DH) parameters and quickly generate a first-draft IK solution. This solution is built on a deep learning-based pipeline using a Long Short-Term Memory (LSTM) neural network integrated with a MATLAB-based graphical user interface (GUI) for automated dataset generation and model training. To enhance performance, this approach applies various data preprocessing techniques, including MinMaxScaler, Normalizer, RobustScaler, and StandardScaler. It also incorporates K-Fold cross-validation for performance evaluation and an early stopping mechanism to prevent overfitting. Multiple 6-DOF robotic arms are tested using MATLAB and CoppeliaSim by performing tasks, such as trajectory tracking of letters and words on planar and non-planar surfaces, to ensure a flexible solution across diverse robotic configurations and task environments.

Keywords: Inverse Kinematics, LSTM, Neural Networks, Graphical User Interface

Introduction

Robotic manipulators are integral to various industries, including military, healthcare and aerospace (Ibarra-Pérez *et al.*, 2022). These systems consist of rigid links interconnected by either rotational or prismatic joints. While Forward Kinematics (FK) determines the position and orientation of the end-effector based on specified joint angles (Lu *et al.*, 2022), Inverse Kinematics (IK) calculates the joint configurations required to achieve a desired end-effector position and orientation (Aggogeri *et al.*, 2022; Martínez-Blanco *et al.*, 2021), which is typically more complex because it is computationally expensive (Sharkawy *et al.*, 2022) and often has multiple valid solutions with potential singularities (Kvernberg, 2015; Bouzid *et al.*, 2024).

Despite extensive research, existing IK approaches face significant limitations regarding flexibility across diverse robotic configurations. Approaches based on analytical techniques, while computationally efficient, require tedious work of derivation and adjustments when switching from one configuration to another, where a different configuration may not have a solution using such techniques (Bouzid *et al.*, 2024; Zhao *et al.*, 2024). Geometric techniques similarly struggle to scale effectively for highly redundant manipulators (Lu *et al.*, 2022; Sharkawy *et al.*, 2022). Recent advances utilizing deep learning, including Multilayer Perceptron (MLP) networks (Lu *et al.*, 2022; Cagigas-Muñiz, 2023), Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) (Volinski *et al.*, 2022), Convolutional Neural Networks (CNNs) (Elkholy *et al.*, 2020), Deep Deterministic Policy Gradient (DDPG) (Surriani *et al.*, 2024) and Bidirectional Long Short-Term Memory (BiLSTM) and Gated Recurrent Unit (GRU) models (Wagaa *et al.*, 2023), have improved accuracy and efficiency. However, these approaches still face challenges in maintaining flexibility across diverse robotic arms. For instance, MLP networks were enhanced by different approaches such as joint space segmentation (Lu *et al.*, 2022) and bootstrap sampling (Cagigas-Muñiz, 2023), showing improved efficiency

DOI: 10.3844/jmrsp.2025.13.23

and accuracy compared to traditional methods, but each of these approaches was limited to only one specific robotic arm—such as Xarm6 and Scorbot ER VII, analyzed by Lu *et al.* (2022); Cagigas-Muñiz (2023), respectively—and required significant effort to be adapted to other robotic platforms.

Approaches like deep learning-enhanced damped least squares (Wang *et al.*, 2021; Wu & Ren, 2020), Long Short-Term Memory (LSTM) networks (Wang *et al.*, 2023), Backpropagation Neural Network (BPNN) models (Ibarra-Pérez *et al.*, 2022; Gao, 2020) and AI-based techniques such as Neural Network Genetic Algorithm (NNGA) and Particle Swarm Optimization (PSO) (Khaleel & Humaidi, 2024), have improved computation time and accuracy, but they don't fully explore their adaptability across different robotic tasks. For instance, Ibarra-Pérez *et al.* (2022) and Gao (2020) achieved high prediction accuracy with low error margins using BPNNs, but their experiments were limited to specific tasks and basic IK predictions.

To utilize learning-based IK approaches while providing an easy-to-use interface, this study introduces an automated pipeline for 6-DOF robotic arms. An interactive MATLAB-based Graphical User Interface (GUI) was designed to automate dataset generation, preprocessing and model training based on user-defined Denavit-Hartenberg (DH) parameters. To test the automated pipeline's flexibility across different robot configurations, systematic validations were conducted on multiple industrial robotic arms, including ABB IRB 4600-40/2.55, KUKA KR5 Arc and KUKA KR60-3, in realistic simulation environments provided by MATLAB and CoppeliaSim. Adaptability across different robotic tasks was demonstrated through trajectory-tracking tasks on both planar and non-planar surfaces, exemplified by tracing complex paths such as the word "Yemen" on spherical surfaces and the letter "W" on planar surfaces. The proposed method achieved consistent accuracy ranging from 88.47-92.75%, employing efficient optimization techniques, including preprocessing scaling methods, K-Fold cross-validation and an early stopping mechanism.

Unlike existing approaches that require significant manual effort for each new robot, our model supports direct integration with different 6-DOF robotic arms, making it more practical for early-phase robot design.

Kinematics

In this study, the ABB IRB4600-40/2.55, which is a 6-DOF robotic arm with revolute joints, is the primary case study whose kinematics will be deeply analyzed. The other tested robotic arms (KUKA KR5 Arc and KUKA KR60-3) are used only to further verify our flexible proposed approach. The position and orientation of each ABB IRB4600-40/2.55 joint are represented using Denavit-Hartenberg (DH) transformation matrices, which can be written as Eq. (1). These matrices are products of four key parameters: Joint angle (θ_i), link offset (d_i), link length (a_i) and link twist (α_i) (Denavit and Hartenberg, 1955):

$$T_i = Rot_{z,\theta i} Trans_{z,di} Trans_{x,ai} Rot_{x,\alpha i} \# \quad (1)$$

The overall transformation matrix for the robot's end-effector relative to the base is obtained by multiplying all the joint transformation matrices and can be written as Eq. (2):

$${}^0_6T = {}^0_1T \times {}^1_2T \times {}^2_3T \times {}^3_4T \times {}^4_5T \times {}^5_6T \quad (2)$$

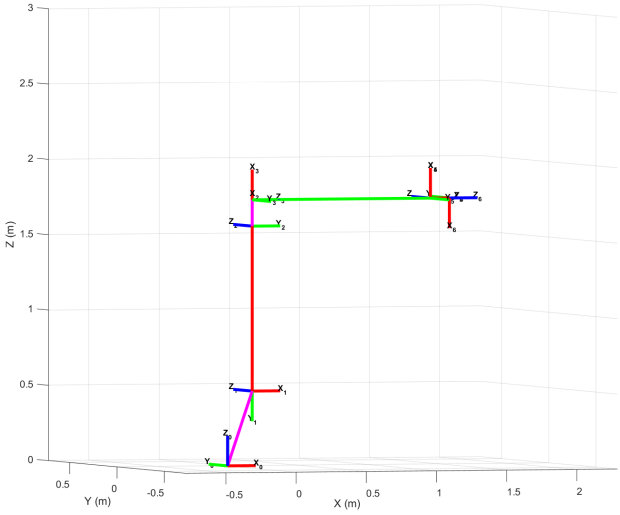

Fig. 1: ABB IRB4600-40/2.55 coordinate system

By defining the first frame to the origin X_0, Y_0, Z_0, as shown in Fig. (1), the frames are rotated and translated according to the Denavit-Hartenberg (DH) convention (Murray *et al.*, 1994) as shown in Table (1).

Table 1: DH Parameters for ABB IRB4600-40/2.55

Joint	θi (rad)	Range (deg)	αi-1 (rad)	di (m)	ai (m)
1	θ1	-180 ~ 180	$-\frac{\pi}{2}$	0.495	0.175
2	θ1 - $\frac{\pi}{2}$	-90 ~ 150	0	0	1.095
3	θ3	-180 ~ 75	$-\frac{\pi}{2}$	0	0.175
4	θ4	-400 ~ 400	$+\frac{\pi}{2}$	1.270	0
5	θ5	-125 ~ 120	$-\frac{\pi}{2}$	0	0
6	θ6 + π	-400 ~ 400	0	0.135	0

The task of solving the Inverse Kinematics (IK) problem is very important and quite challenging due to the geometry of the robot and the resulting nonlinear trigonometric equations (Almusawi *et al.*, 2016). In this study, the proposed approach not only solved the IK problem but also introduced an automated pipeline, flexible to directly integrate different robotic configurations.

DOI: 10.3844/jmrsp.2025.13.23

Materials and Methods

Graphical User Interface (GUI)

To enhance the flexibility of our proposed solution by making it more user-friendly, we developed a MATLAB Graphical User Interface (GUI) to input the Denavit-Hartenberg (DH) parameters specific to the robotic arm configuration, as shown in Figure (2).

Once the parameters are entered (in this trial, ABB IRB 4600-40/2.55), as shown in Figure (3), the system generates a kinematic dataset by computing end-effector positions and orientations for different joint movements. Additionally, the GUI initiates the LSTM model training process by executing an integrated Python script that utilizes TensorFlow to train the proposed model. Real-time feedback during training updates the user on model progress and output values. Figure (4) illustrates the GUI workflow, from DH parameter entry to result visualization. Designed for flexibility, the GUI accommodates different robotic arm configurations without requiring code modifications. Input fields dynamically adjust based on the selected joint type, ensuring compatibility across multiple robot models.

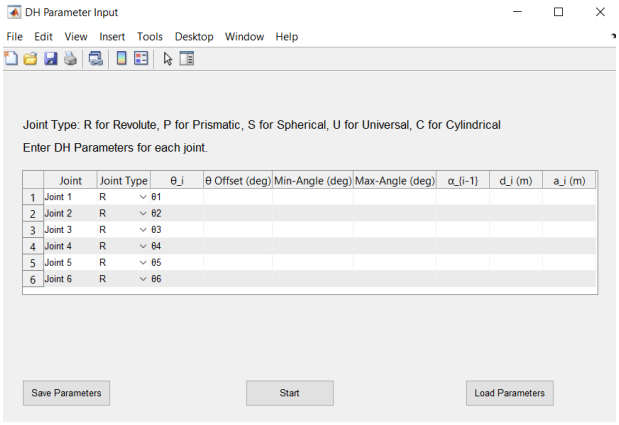

Fig. 2: DH table GUI parameters entry

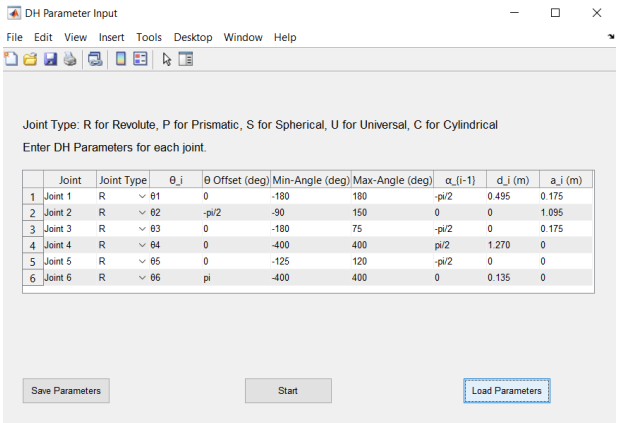

Fig. 3: Loading ABB IRB 4600-40/2.55 DH parameters

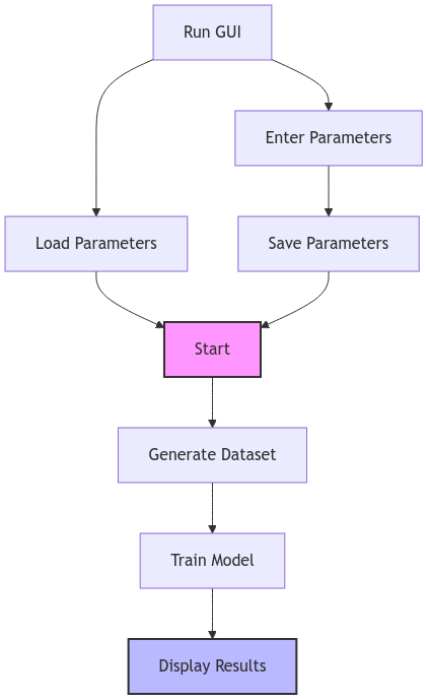

Fig. 4: GUI workflow

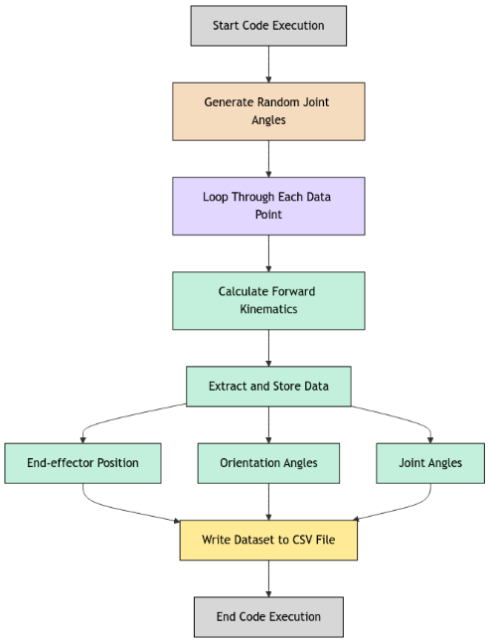

Fig. 5: Dataset generation workflow

Data Collection

The dataset was generated using MATLAB to simulate the robotic arm's workspace kinematics, similar to the approach taken by Ibarra-Pérez *et al.* (2022). This involved randomizing joint angles within their specified operational ranges to model diverse motion scenarios and capture the corresponding end-effector positions and orientations. The dataset composition for the ABB

DOI: 10.3844/jmrsp.2025.13.23

IRB4600-40/2.55 robotic arm encompassed around 70 million data points stored in a CSV file named *IRB4600_Dataset.csv*. Additionally, each data point captured the end-effector's three-dimensional position (X, Y, Z), orientation vectors (n, o, a) and Roll, Pitch and Yaw (RPY) angles (Wang *et al.*, 2023). Figure (5) illustrates the workflow of generating the dataset for our specified model.

Model Development

A study by Wagaa *et al.* (2023) considered different deep-learning algorithms, including CNN, LSTM and BILSTM. It suggested that LSTM and BILSTM (which handle both forward and backward LSTM information) are more accurate and efficient than other approaches despite their simpler and faster training process. Similarly, Zhang (2024), in his review, emphasized that LSTM exhibits superior capabilities and provides more accurate joint angle estimation, making it one of the ideal choices for solving IK problems. Given these advantages, LSTM was selected as the foundation of our proposed model to develop a learning-based IK solution.

The LSTM neural network architecture was designed to effectively process the complex kinematics data associated with 6-DOF robotic arms. As shown in Figure (6), the model consists of four LSTM hidden layers with increasing units from 64 to 256, followed by dense layers and an output layer using a linear activation function to predict joint angles. The ReLU activation function was applied to the LSTM layers to introduce non-linearity (Korol *et al.*, 2023). The selection of four LSTM layers and the specific number of units (64, 128, 256, 256) was determined through hyperparameter tuning experiments to balance model accuracy, convergence speed and overfitting prevention. There is no specific method to accurately determine the optimum number of layers (Wagaa *et al.*, 2023). Initial tests with fewer layers (two or three LSTM layers) resulted in higher validation loss and unstable predictions, while deeper architectures (more than four layers) did not provide significant accuracy improvements but increased training time.

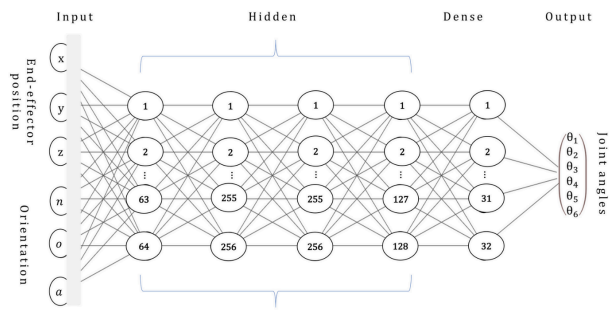

Fig. 6: Architecture for the proposed LSTM neural network

Additionally, K-Fold cross-validation (K = 2) was employed to assess model robustness across different data partitions, ensuring that the chosen architecture generalized well to unseen data (Hernandez *et al.*, 2021). The final model demonstrated consistent accuracy improvements across all validation folds, confirming the effectiveness of the selected LSTM configuration.

For data preprocessing, various scaling techniques such as MinMaxScaler (Choi *et al.*, 2024), Normalizer (Elkholy *et al.*, 2020), RobustScaler (Choi *et al.*, 2024) and StandardScaler (Carneros-Prado *et al.*, 2024) were implemented for both input and output data. The model also incorporated the early stopping mechanism during training to prevent overfitting by halting the training process if the validation accuracy wasn't improved for a set number of epochs (Volinski *et al.*, 2022).

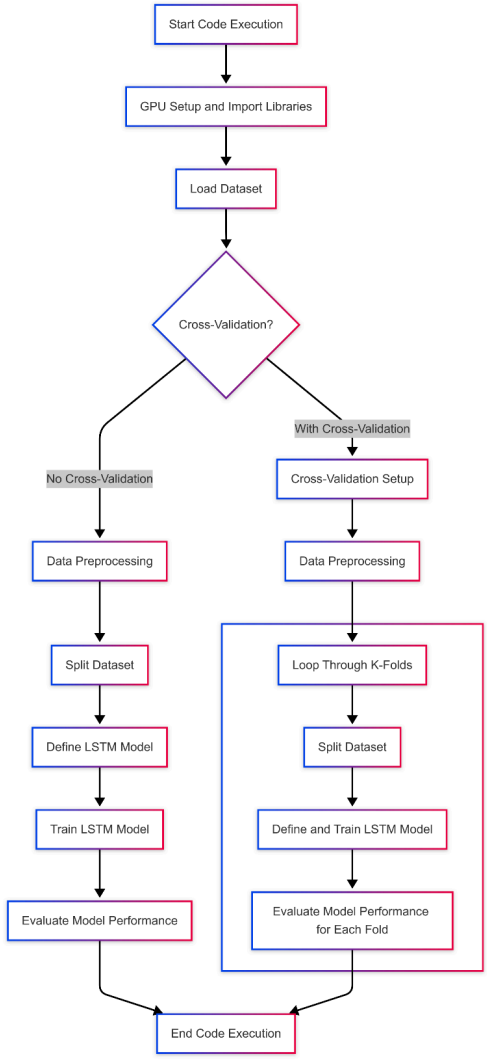

Fig. 7: Model with and without cross-validation

Model development, as can be seen in Figure (7), starts with setting up the Graphics Processing Unit (GPU) to be used in the proposed model training due to its highly efficient speed compared with the Central Processing Unit (CPU). Then, necessary libraries are imported for data handling and neural network

DOI: 10.3844/jmrsp.2025.13.23

operations. After that, the dataset is loaded, scaled and split into training and testing sets. A sequential LSTM neural network is built and trained, with model performance evaluated through loss and accuracy metrics. If cross-validation is used, the process is iterated over k-folds, with each fold's model trained and evaluated independently, aggregating the performance metrics for a comprehensive analysis.

Results

The ABB IRB4600-40/2.55's testing phase involved training the LSTM model using different scaling techniques with K-Fold cross-validation to better evaluate model performance. Overall, the results in Table (2) showed consistent improvements in model accuracy, with final validation accuracies ranging from 88.47-92.75%, depending on the chosen scaling technique and the incorporation of K-Fold cross-validation.

Table 2: ABB IRB4600-40/2.55 model's scaling technique estimated accuracies

Scaling technique	Start validation accuracy (%)	End validation accuracy (%)	Epochs[1] to End (Early Stopping)
MinMaxScaler	17.59	88.47	86/100
Normalizer	32.00	90.16	88/100
RobustScaler	42.12	91.91	74/100
StandardScaler	36.34	92.75	76/100
StandardScaler with K-Fold Cross-Validation	Fold 1: 38.82 Fold 2: 36.85	Fold 1: 91.43 Fold 2: 92.30	50/50 50/50

[1] *The epochs were ended early in each case using early stopping to prevent overfitting and ensure efficient training.*

StandardScaler achieved the highest final validation accuracy (92.75%) and improved efficiency, leading to a faster training process and better generalization (Carneros-Prado *et al.*, 2024). RobustScaler also exhibited strong performance, achieved a final accuracy of 91.91% and maintained a low Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) (Choi *et al.*, 2024). Normalizer and MinMaxScaler resulted in slightly lower validation accuracies of 90.16% and 88.47%, respectively. Applying scaling features by Normalizer and MinMaxScaler to a fixed range, often between 0 and 1, led to slower convergence and lower final accuracies compared to StandardScaler and RobustScaler (Elkholy *et al.*, 2020; Choi *et al.*, 2024).

The training and validation curves for accuracy and loss, as shown in Figure (8), further illustrate the performance resulting from the applied scaling techniques. StandardScaler exhibited the closest alignment between training and validation. RobustScaler also converged rapidly while maintaining a high final accuracy. In contrast, MinMaxScaler and Normalizer showed slower convergence, although they still reached high validation accuracies. The loss curves for each scaling technique demonstrate a steep initial decrease followed by stabilization. RobustScaler and StandardScaler show a slightly faster decline in loss and stabilize earlier compared to MinMaxScaler and Normalizer.

The utilization of the cross-validation technique, as shown in Figure (9), shows that the accuracy curves for both Fold 1 and Fold 2 start from moderate initial values (38.82% for Fold 1 and 36.85% for Fold 2) and increase consistently over the training epochs, reaching above 91% by Epoch 50. Both training and validation accuracy curves are closely aligned throughout, resulting in high final validation accuracies (91.43% for Fold 1 and 92.30% for Fold 2). The loss curves for both folds show a steep initial decline in the first 10 epochs, followed by gradual stabilization at low loss values.

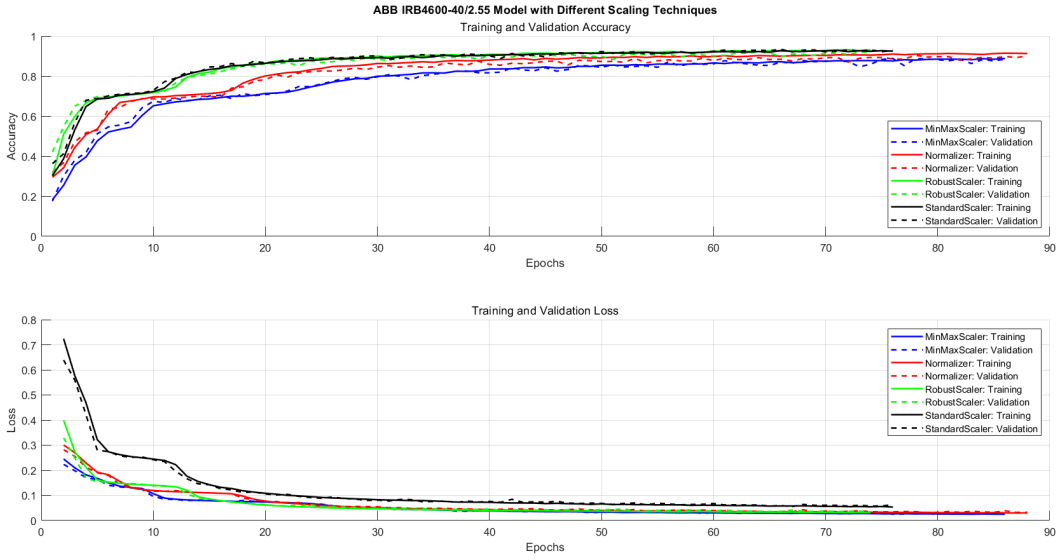

Fig. 8: Accuracy and loss curves of ABB IRB4600-40/2.55 model with different scaling techniques

DOI: 10.3844/jmrsp.2025.13.23

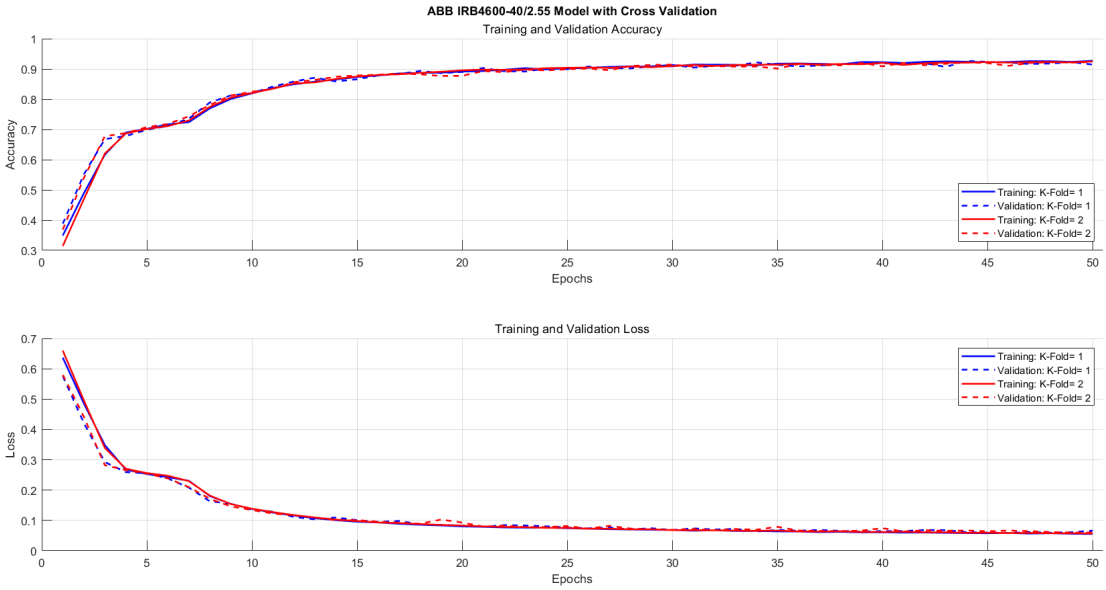

Fig. 9: Accuracy and loss curves of ABB IRB4600-40/2.55 model with cross-validation

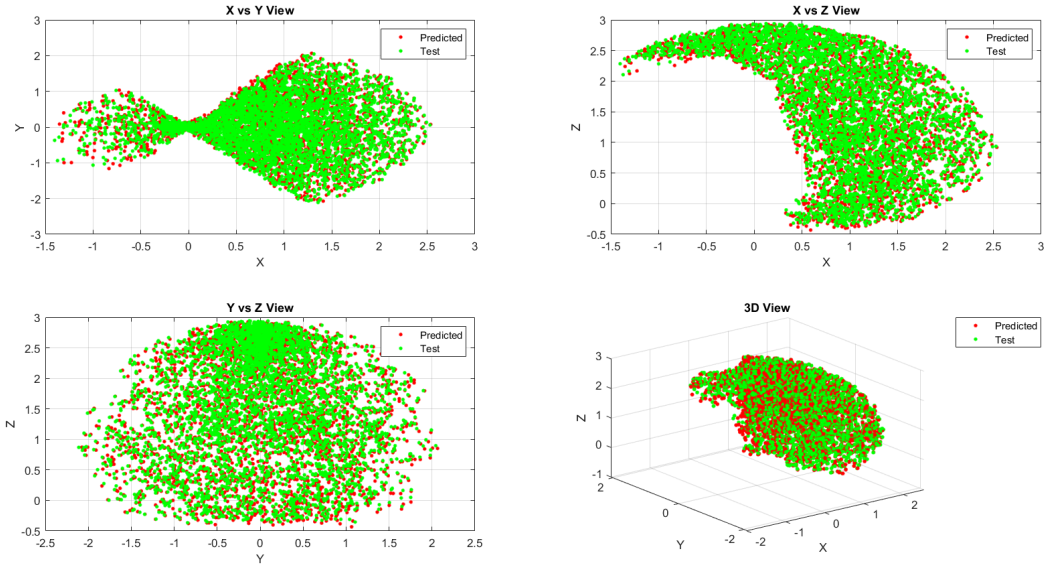

Fig. 10: ABB IRB 4600-40/2.55 predicted and actual end-effector positions

Table 3: ABB IRB4600-40/2.55 model comparison between predicted and actual joint angles; Average Accuracy 92.72%

Samples	θi	θ1	θ2	θ3	θ4	θ5	θ6	Accuracy
1	Actual	-38.1521	-14.4687	48.05278	28.96949	9.873431	50.01912	92.76%
	Predict	-37.1103	-13.4717	47.04272	27.03358	7.61423	48.94568	
2	Actual	7.622861	-48.7378	-48.5768	46.79007	31.13482	48.0959	92.20%
	Predict	8.19713	-51.3646	-52.5853	42.16608	28.27274	44.95103	
3	Actual	-56.3424	-8.71451	16.70616	36.33426	22.83993	-12.5774	93.01%
	Predict	-54.397	-9.00147	15.73885	34.38157	23.85653	-10.1125	
4	Actual	-38.6552	53.60645	-30.5521	6.897025	21.89236	-22.1092	93.38%
	Predict	-37.3684	50.96294	-28.8911	5.767998	22.09929	-20.1844	
5	Actual	3.734295	-22.4269	-16.5102	10.62862	25.09474	-3.0021	92.26%
	Predict	3.207422	-22.0958	-16.0432	10.01813	25.87689	-3.5773	

DOI: 10.3844/jmrsp.2025.13.23

A visual inspection of the model's predicted versus actual end-effector positions was conducted using a 3D scatter plot, as shown in Figure (10), where green and red dots represented the actual and predicted positions, respectively. The high degree of overlap between these points indicated the model's accuracy in predicting joint configurations that achieve the desired end-effector positions. Additionally, a comparative analysis between actual and predicted joint angles for the ABB IRB4600-40/2.55 robotic arm is shown in Table (3). Five test samples were taken for this comparison, where each sample consists of six joint angles (θ1-θ6). The actual joint angles represent the desired values, while the predicted joint angles are the values estimated by the trained LSTM model.

The overall average accuracy across all samples was 92.72%, indicating that the LSTM-based inverse kinematics model is highly precise in estimating joint angles for the ABB IRB4600-40/2.55 robotic arm. Additionally, the low RMSE and MAE metrics in Table (4) demonstrated minimal discrepancies between predicted and actual values.

Table 4: RMSE and MAE for ABB IRB 4600-40/2.55 trained model

Metric	1	2	3	4
RMSE	0.000565	0.000586	0.000484	0.000266
MAE	0.0177	0.0192	0.0175	0.0128

To validate the model's practical applicability, the ABB IRB 4600-40/2.55 robotic arm was commanded to write the letter "W" in CoppeliaSim with added Gaussian noise to simulate sensor inaccuracies and real-world mechanical variations in order to assess the model's ability to maintain accurate predictions under realistic operating conditions (Korol *et al.*, 2023). The trajectory was closely followed by the robot without noticeable deviation, as shown in Figure (11). Further simulations were conducted using MATLAB, where the ABB IRB 4600-40/2.55 was commanded to write the letter "W" on a planar surface and the word "Yemen" on a spherical surface, as shown in Figs. (12-13). These simulations indicated that the model is flexible and precise in dealing with planar and non-planar environments.

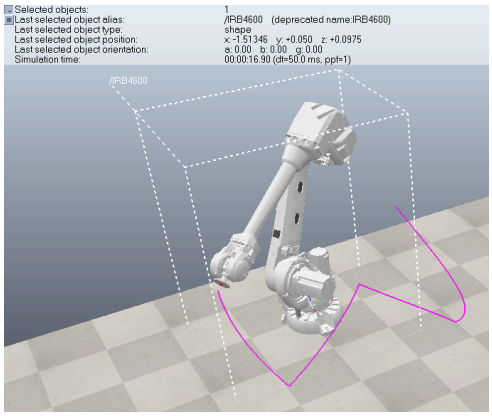

Fig. 11: ABB IRB 4600-40/2.55 model test to write the letter "W" using CoppeliaSim

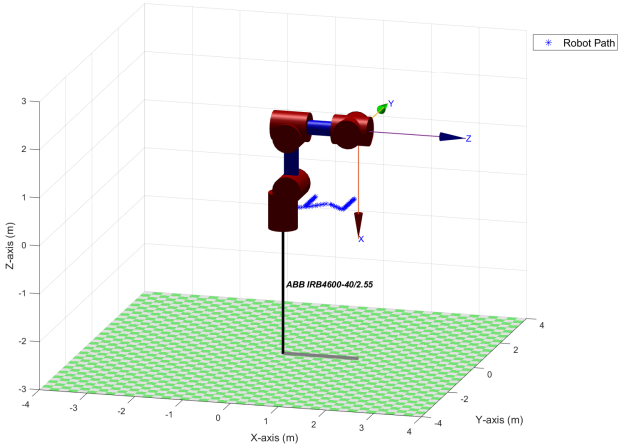

Fig. 12: ABB IRB 4600-40/2.55 model test to write the letter "W" using MATLAB

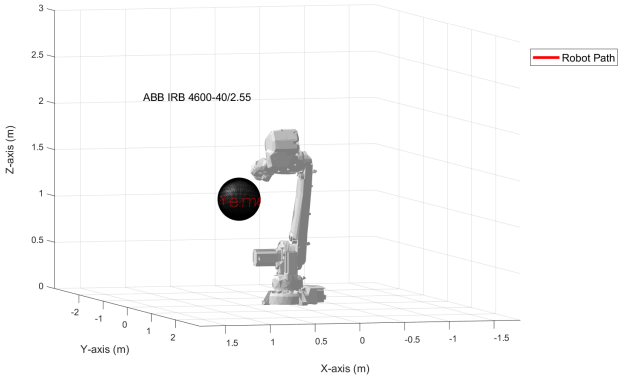

Fig. 13: ABB IRB 4600-40/2.55 model test to write the word "Yemen" on a sphere using MATLAB

To test the flexibility of the proposed approach across different robotic configurations, other 6-DOF robotic arms were tested, including the KUKA KR 5 Arc and KUKA KR60-3. The designed GUI was used to input the Denavit-Hartenberg (DH) parameters of these robotic arms and upon clicking the GUI start button, the following steps occurred sequentially:

1. Dataset generation: The dataset was generated automatically in MATLAB
2. Model training: The MATLAB Engine API was used to connect with Python (Jupyter Lab). Then, the generated dataset was loaded by the LSTM model for the training stage.
3. Model prediction: After training, the model began predicting joint angles for any point within the robot's workspace to achieve the desired end-effector position and orientation

Figure (14) shows the training and validation curves for the resulting accuracy and loss for these robotic arms. The KUKA KR 5 Arc and KUKA KR60-3 exhibited stable learning with validation accuracies of 90.62% and 90.47%, respectively. Similarly, the loss curves were consistent with those observed in the ABB IRB 4600-40/2.55 model, demonstrating a steep initial decrease

DOI: 10.3844/jmrsp.2025.13.23

followed by stabilization. Tables (5-6) show a comparative analysis between the actual and predicted joint angles for both KUKA KR 5 Arc and KUKA KR60-3. The average accuracies were 90.6 and 90.4%, respectively. These results indicated that the proposed model was able to estimate precise joint angles for both robotic arms. Low RMSE and MAE values, as shown in Table (7), further indicated the minimal discrepancies between predicted and actual values.

Moreover, to validate the model's accuracy in a simulated environment, we conducted another test using MATLAB. Both robotic arms were commanded to write the letter "W", as shown in Figs. (15-16). KUKA KR 5 Arc demonstrated precise trajectory tracking with minimal deviation due to its compact structure, while the KUKA KR60-3 maintained accuracy over a larger workspace but exhibited slight dynamic effects due to its extended reach.

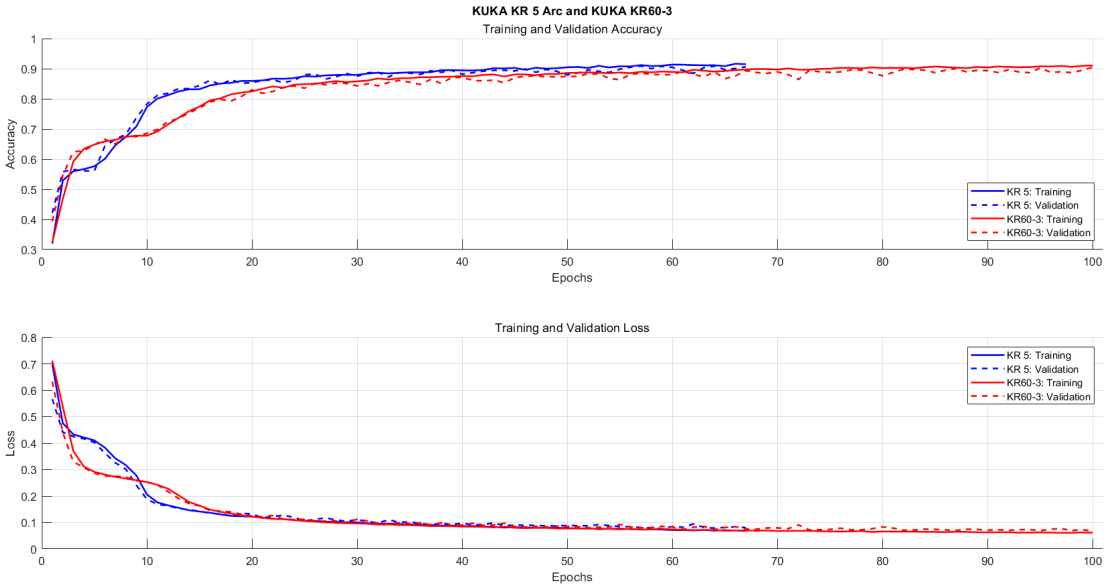

Fig. 14: Accuracy and loss curves of KUKA KR 5 Arc and KUKA KR60-3 models

Table 5: KUKA KR 5 Arc model comparison between predicted and actual joint angles; Average Accuracy 90.6%

Samples	θi	θ1	θ2	θ3	θ4	θ5	θ6	Accuracy
1	Actual	-9.10511	-7.35384	12.38574	-87.5915	31.17037	82.58307	90.07%
	Predict	-10.9627	-8.93495	13.61338	-88.3488	29.87487	80.31445	
2	Actual	29.11507	54.66291	6.266068	-65.7562	49.64762	9.596348	90.78%
	Predict	28.85066	56.28456	4.810633	-66.2897	44.46794	11.22401	
3	Actual	46.07554	15.77651	-7.57522	-8.18301	33.16003	-11.0968	91.33%
	Predict	45.75831	15.26789	-8.80765	-7.52399	33.56766	-13.602	
4	Actual	39.38837	13.39209	-12.984	83.17668	34.05664	-45.1423	91.05%
	Predict	41.81043	15.32537	-10.702	85.37033	36.81911	-47.3155	
5	Actual	-12.5971	-1.13944	-6.91542	6.771719	30.81741	10.01993	90.19%
	Predict	-12.5047	-1.37869	-6.36012	6.802795	30.78708	12.87984	

Table 6: KUKA KR60-3 model comparison between predicted and actual joint angles; Average Accuracy 90.4%

Samples	θi	θ1	θ2	θ3	θ4	θ5	θ6	Accuracy
1	Actual	47.46161	-18.7959	57.0603	88.81509	-44.8883	46.31631	90.75%
	Predict	50.87112	-22.4236	60.08445	84.56924	-49.1375	50.68897	
2	Actual	-83.674	21.64952	32.20376	68.7404	10.04532	-27.0067	90.07%
	Predict	-85.0301	23.98203	29.85171	63.30581	8.492323	-22.5427	
3	Actual	-3.1581	-26.6831	69.45905	-76.3869	-9.58206	-65.7873	90.56%
	Predict	-4.01884	-28.4315	65.25291	-80.0226	-9.89739	-60.0279	
4	Actual	-58.0358	-12.7951	-56.0422	13.67581	12.55865	-47.9691	90.11%
	Predict	-56.4978	-13.6215	-56.7346	18.17967	13.62612	-51.614	
5	Actual	-53.8807	10.36354	-74.6653	-6.01011	-69.3135	-42.6721	90.02%
	Predict	-52.1509	11.12066	-72.9175	-7.90856	-71.1573	-45.5663	

DOI: 10.3844/jmrsp.2025.13.23

Table 7: RMSE and MAE for KUKA KR 5 Arc and KUKA KR60-3 trained models

Metric	1	2	3	4
KUKA KR 5 Arc				
RMSE	0.00026	0.00039	0.00015	8.7E-05
MAE	0.0113	0.0123	0.0088	0.0064
KUKA KR60-3				
RMSE	0.00062	0.00075	0.0006	0.00045
MAE	0.0181	0.0203	0.0168	0.0159

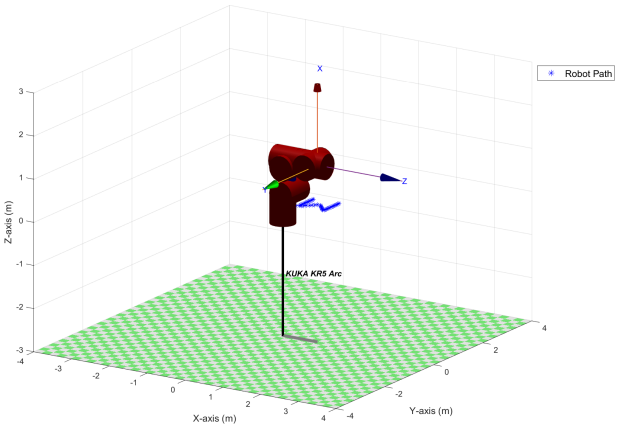

Fig. 15: KUKA KR 5 Arc model test to write the letter "W" using MATLAB

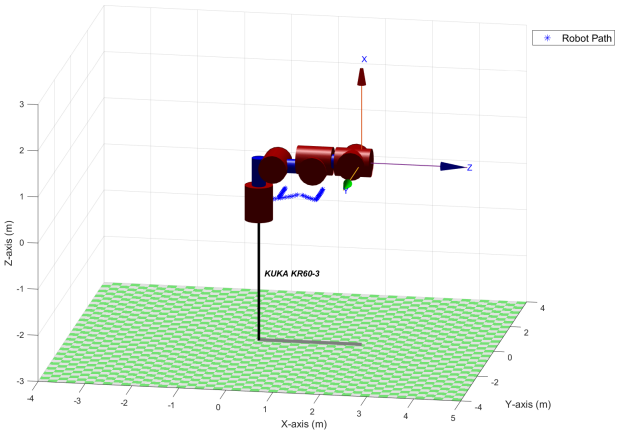

Fig. 16: KUKA KR60-3 model test to write the letter "W" using MATLAB

Discussion

This study's approach utilized an LSTM neural network to handle the sequential nature of joint data effectively and this choice resulted in high validation accuracies across different robotic arms. These accuracies are consistent with the performances reported in Zhang (2024); Wagaa *et al.* (2023), which validated our choice of the LSTM neural network over other existing approaches such as CNN and GRU (Wagaa *et al.*, 2023).

Unlike existing studies that focused solely on specific robotic arms such as Xarm6 (Lu *et al.*, 2022) and Scorbot ER VII (Cagigas-Muñiz, 2023), our approach explored multiple robotic arms such as ABB IRB 4600-40/2.55, KUKA KR 5 Arc and KUKA KR60-3, resulting in high validation accuracies of 92.75, 90.62 and 90.47%, respectively. The adaptability limitations across different robotic tasks were effectively addressed by implementing practical motion executions, such as real-time trajectory tracking on both planar and non-planar surfaces, using MATLAB and a dynamic industrial environment like CoppeliaSim, as shown in Figs. (11-13). This contrasts with studies like Ibarra-Pérez *et al.* (2022) and Gao (2020), whose improved models were simulated only in MATLAB and focused exclusively on basic inverse kinematics predictions without testing in diverse environments and tasks.

Regarding optimization techniques, our deep learning model utilized different preprocessing scaling methods to improve model convergence and accuracy, K-Fold cross-validation (K = 2) to assess model robustness across different data partitions, ensuring generalization beyond the training dataset and an early stopping mechanism to prevent overfitting by halting training when model improvements plateaued. These strategies resulted in higher accuracy compared to existing approaches. For example, Ibarra-Pérez *et al.* (2022) and Wu and Ren (2020) employed optimization techniques such as the Taguchi method and genetic algorithms to enhance training time and accuracy. However, Ibarra-Pérez *et al.* (2022) reported a prediction accuracy of 87.71%, which is lower than the validation accuracies achieved by our approach across the three tested robotic arms.

A comparative analysis was also conducted to validate the proposed approach further. The average accuracy was estimated using different testing samples of actual and predicted joint angles by the developed LSTM model for the three tested robotic arms, as shown in Tables (3, 5 and 6). These comparisons revealed minimal discrepancies between predicted and actual values and yielded high validation accuracies, confirming the superior flexibility of our approach across different robotic arms.

Conclusion

The proposed approach demonstrated efficient performance in predicting joint configurations for the ABB IRB 4600-40/2.55, KUKA KR 5 Arc and KUKA KR60-3 robotic arms. The real-time simulations conducted in CoppeliaSim and MATLAB further validated the model's practical applicability, with successful demonstrations of different tasks, including tracing complex paths on planar and non-planar surfaces. The integration of a user-friendly GUI makes our method accessible for industrial applications, especially in the early-phase designs, even for users with limited technical knowledge. However, the flexibility of the proposed model is constrained by the varying joint angle limits of different robotic arms, which may affect training efficiency since the model uses a single LSTM neural

network pre-configuration for all 6-DOF robotic arms. Future research should focus on exploring more advanced LSTM configurations or incorporating hybrid architectures that can better accommodate the unique kinematic constraints of different robotic arms. Additionally, integrating real-world testing environments and incorporating feedback mechanisms for error correction are important steps toward advancing the applicability of our method in diverse industrial settings. Finally, this study contributes significantly to the academic and industrial fields by presenting an LSTM-based inverse kinematics automated pipeline for 6-DOF robotic arms facilitated by a user-friendly GUI that provides significant flexibility.

Acknowledgement

Authors extend their sincere gratitude to Sana'a University's Department of Mechatronics Engineering. Authors also deeply appreciate their families unwavering support and encouragement.

Funding Information

There is no received funding or financial support to report.

Author's Contributions

Waleed Abdulrahman Saleh Al-Akwa: Contributed to the conception, design, development and implementation of this study. He also contributed to writing the original draft, analyzing the data, and conceptualizing this study.

Mohammed Abdulwahab Ahmed Daba: Contributed to drafting and critical revision of the manuscript as well as to the supervision and investigation.

Ethics

This study is original and has not been published before. The authors declare that no conflict of interest or any ethical issues may arise.

References

Aggogeri, F., Pellegrini, N., Taesi, C., & Tagliani, F. L. (2022). Inverse kinematic solver based on machine learning sequential procedure for robotic applications. *Journal of Physics: Conference Series*, *2234*(1), 012007. https://doi.org/10.1088/1742-6596/2234/1/012007

Almusawi, A. R. J., Dülger, L. C., & Kapucu, S. (2016). A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242). *Computational Intelligence and Neuroscience*, *2016*, 1-10. https://doi.org/10.1155/2016/5720163

Bouzid, R., Gritli, H., & Narayan, J. (2024). ANN Approach for SCARA Robot Inverse Kinematics Solutions with Diverse Datasets and Optimisers. *Applied Computer Systems*, *29*(1), 24-34. https://doi.org/10.2478/acss-2024-0004

Cagigas-Muñiz, D. (2023). Artificial Neural Networks for inverse kinematics problem in articulated robots. *Engineering Applications of Artificial Intelligence*, *126*, 107175. https://doi.org/10.1016/j.engappai.2023.107175

Carneros-Prado, D., Dobrescu, C. C., Cabañero, L., Villa, L., Altamirano-Flores, Y. V., Lopez-Nava, I. H., González, I., Fontecha, J., & Hervás, R. (2024). Synthetic 3D full-body skeletal motion from 2D paths using RNN with LSTM cells and linear networks. *Computers in Biology and Medicine*, *180*, 108943. https://doi.org/10.1016/j.compbiomed.2024.108943

Choi, H. S., Yoon, S., Kim, J., Seo, H., & Choi, J. K. (2024). Calibrating Low-Cost Smart Insole Sensors with Recurrent Neural Networks for Accurate Prediction of Center of Pressure. *Sensors*, *24*(15), 4765. https://doi.org/10.3390/s24154765

Denavit, J., & Hartenberg, R. S. (1955). A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. *Journal of Applied Mechanics*, *22*(2), 215-221. https://doi.org/10.1115/1.4011045

Elkholy, H. A., Shahin, A. S., Shaarawy, A. W., Marzouk, H., & Elsamanty, M. (2020). Solving Inverse Kinematics of a 7-DOF Manipulator Using Convolutional Neural Network. *Proceedings of the International Conference on Artificial Intelligence and Computer Vision (AICV2020)*, 343-352. https://doi.org/10.1007/978-3-030-44289-7_32

Gao, R. (2020). Inverse kinematics solution of Robotics based on neural network algorithms. *Journal of Ambient Intelligence and Humanized Computing*, *11*(12), 6199-6209. https://doi.org/10.1007/s12652-020-01815-4

Hernandez, V., Dadkhah, D., Babakeshizadeh, V., & Kulić, D. (2021). Lower body kinematics estimation from wearable sensors for walking and running: A deep learning approach. *Gait & Posture*, *83*, 185-193. https://doi.org/10.1016/j.gaitpost.2020.10.026

Ibarra-Pérez, T., Ortiz-Rodríguez, J. M., Olivera-Domingo, F., Guerrero-Osuna, H. A., Gamboa-Rosales, H., & Martínez-Blanco, Ma. del R. (2022). A Novel Inverse Kinematic Solution of a Six-DOF Robot Using Neural Networks Based on the Taguchi Optimization Technique. *Applied Sciences*, *12*(19), 9512. https://doi.org/10.3390/app12199512

Khaleel, H. Z., & Humaidi, A. J. (2024). Towards accuracy improvement in solution of inverse kinematic problem in redundant robot: A comparative analysis. *International Review of Applied Sciences and Engineering*, *15*(2), 242-251. https://doi.org/10.1556/1848.2023.00722

Korol, A. S., Rodzin, T., Zabava, K., & Gritsenko, V. (2023). Neural networks-based approach to solve inverse kinematics problems for medical applications. *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 1-6. https://doi.org/10.1109/EMBC53108.2024.10782521

Kvernberg, P. (2015). *Developing force control scenarios on ABB IRB 4600 with camera capture of dynamic motions*.

Lu, J., Zou, T., & Jiang, X. (2022). A Neural Network Based Approach to Inverse Kinematics Problem for General Six-Axis Robots. *Sensors*, *22*(22), 8909. https://doi.org/10.3390/s22228909

Martínez-Blanco, Ma. del R., Ibarra-Pérez, T., Olivera-Domingo, F., & Ortiz-Rodríguez, J. M. (2021). Robust Design of Artificial Neural Network Methodology to Solve the Inverse Kinematics of a Manipulator of 6 DOF. *Artificial Intelligence (AI)*, 171-210. https://doi.org/10.1201/9781003005629-9

Murray, R. M., Li, Z., & Sastry, S. S. (1994). *A mathematical introduction to robotic manipulation*. https://doi.org/10.1201/9781315136370

Sharkawy, A.-N. (2022). Forward and inverse kinematics solution of a robotic manipulator using a multilayer feedforward neural network. *Journal of Mechanical and Energy Engineering*, *6*(2), 1-17. https://doi.org/10.30464/jmee.00300

Surriani, A., Wahyunggoro, O., & Imam Cahyadi, A. (2024). Inverse kinematic solution and singularity avoidance using a deep deterministic policy gradient approach. *IAES International Journal of Artificial Intelligence (IJ-AI)*, *13*(3), 2999. https://doi.org/10.11591/ijai.v13.i3.pp2999-3009

Volinski, A., Zaidel, Y., Shalumov, A., DeWolf, T., Supic, L., & Ezra Tsur, E. (2022). Data-driven artificial and spiking neural networks for inverse kinematics in neurorobotics. *Patterns*, *3*(1), 100391. https://doi.org/10.1016/j.patter.2021.100391

Wagaa, N., Kallel, H., & Mellouli, N. (2023). Analytical and deep learning approaches for solving the inverse kinematic problem of a high degrees of freedom robotic arm. *Engineering Applications of Artificial Intelligence*, *123*, 106301. https://doi.org/10.1016/j.engappai.2023.106301

Wang, S., Zhang, Y., Chen, S., Xu, M., Yu, Y., & Liu, P. (2023). Inverse Kinematics Analysis of 5-DOF Cooperative Robot Based on Long Short-Term Memory Network. *2023 IEEE 3rd International Conference on Software Engineering and Artificial Intelligence (SEAI)*, 245-249. https://doi.org/10.1109/seai59139.2023.10217721

Wang, X., Liu, X., Chen, L., & Hu, H. (2021). Deep-learning damped least squares method for inverse kinematics of redundant robots. *Measurement*, *171*, 108821. https://doi.org/10.1016/j.measurement.2020.108821

Wu, J., & Ren, X. (2020). Inverse kinematics solution and motion simulation of seven-degree-of-freedom ascending platform based on neural network. *Journal of Physics: Conference Series*, *1650*(3), 032127. https://doi.org/10.1088/1742-6596/1650/3/032127

Zhang, B. (2024). Inverse Kinematics Implementation Techniques in Robotics. *Highlights in Science, Engineering and Technology*, *81*, 109-120. https://doi.org/10.54097/vejx7557

Zhao, C., Wei, Y., Xiao, J., Sun, Y., Zhang, D., Guo, Q., & Yang, J. (2024). Inverse kinematics solution and control method of 6-degree-of-freedom manipulator based on deep reinforcement learning. *Scientific Reports*, *14*(1), 12467. https://doi.org/10.1038/s41598-024-62948-6