Journal of Artificial Intelligence & Control Systems

CoSci Press

Research Article

Performing the Inverse Kinematics of an ABB 6 DOF Robotic Arm with LSTM, Feedforward and Recurrent Neural Networks

Christopher O. Okoh [1,2], Neil Buckley [1], Emanuele Lindo Secco [2,*]

[1]AI Lab, School of Computer Science and the Environment, Liverpool Hope University, Liverpool L16 9JD, UK
[2]Robotic Lab, School of Computer Science and the Environment, Liverpool Hope University, Liverpool L16 9JD, UK
*Corresponding author: seccoe@hope.ac.uk

Article History: Received 9 May 2026; Revised 17 September 2026; Accepted 26 October 2026; Published 13 November 2026

Abstract: Conventional numerical Inverse Kinematics (IK) methods—such as the Jacobian Pseudo-inverse, Levenberg-Marquardt (LM), and Newton-Raphson algorithms—are known to be applied when solving the non-linear IK problems of a robot arm. This is done by taking an initial guess of the joint angles and iteratively or sequentially calculating small adjustments using the robot's Jacobian matrix until the end-effector converges onto the target 6-DoF pose. While highly accurate, these traditional techniques suffer from several fundamental limitations such as high computational cost and slower inference, high dependency on initial guess and inability to handle multi-modal solutions and thus make them challenging to deploy in dynamic, real-time, or highly complex robotic environments. This paper leverages on the dataset of an ABB IRB120 6 Degree Of Freedom (DOF) robot arm from Kaggle where end-effectors' positions obtained by means of Forward Kinematics (FK) are fed into the neural network models. And the derived joint angles were mapped to the end-effectors' position thereby showing a faster method of providing IK solutions. The datasets were trained individually on *Long-Short Term Memory* (LSTM) neural network, *Feed-forward Neural Network* (FNN), and *Recurrent Neural Network* (RNN) using Keras on Tensor Flow. According to a performance analysis of the results, the LSTM model showed an MSE of 0.0001 rad^2 with an execution time of 0.4 secs which translates into a highly precise inverse mapping capable of accurately guiding the robot through its Cartesian trajectory while satisfying latency constraints during robot deployment. Other experimental evaluations made are the MAE, R^2 score and the End-effector error (*Mean*/*Max*) to show a high precision and minimal Euclidean distance error of the End-effector's 3D Cartesian coordinates (*x, y, z*). Comparisons were made with previous results that performed IK evaluations using neural networks.

Keywords: 6-DoF Manipulator Kinematics, Configuration Space Mapping, Spatial End-effector Coordinates, Neural Networks for Kinematics

1. Introduction

Robotic arms are becoming more and more attractive in the context of the industrial revolution and the domestic applications due to their capability to perform repetitive tasks and increasingly adapt to the environment where they operate. To make such devices work properly, direct or Forward and Inverse Kinematics need to be implemented (FK and IK, respectively): Tammishetty et al. [1] refer to FK and IK as the tools to determine the pose (i.e., position and orientation) of the end-effector of the robotic arm. While the FK allows to calculate the values of the coordinates of the end-effector vs. the joint positions, the IK requires finding the solution of the kinematics in terms of finding which values of the joint coordinates allow the arm to reach a specific pose of the tip or end-effector. IK is usually performed by means of iterative techniques or other methods with a possible limited efficiency [2], moreover when the number of Degrees Of Freedom (DOF) of the arm is significantly high stability [3,4].

However, deep-learning literature frequently treats this static kinematic data as an unstructured regression mapping, overlooking two critical friction points: the strict sequential dependency of the physical manipulator chain and the numerical scale imbalances between translational and rotational spaces. Traditional network designs often suffer from gradient instability and loss-function competition when learning these mixed geometric boundaries simultaneously, as the unbounded Euclidean nature of translation spaces inherently conflicts with the periodic manifolds of rotation [5]. According to these limitations, research has also been focusing on utilizing advanced neural networks to map the workspace topology, though the optimal structural paradigm for balancing accuracy, geometric scale conflicts, and execution latency remains an open question.

This paper presents a neural network-based system to perform the IK of an ABB 6 DOF robotic arm in order to calculate the joint coordinates of the robot vs. the required pose of its end-effector with low positioning and orientation errors. Specifically, this study evaluates three core research questions: First, how effectively can recurrent models exploit the sequential nature of the physical kinematic chain to improve configuration accuracy over flat regression? Second, to what extent can gated internal channels isolate and resolve the scaling discrepancies between translation and orientation elements? Third, what are the precise trade-offs between model parameter efficiency and real-time inference latency?

Considering dynamical system models, the ability to apply universal approximators, and weight adjustments during learning by using gradient descent and backpropagation, Long Short-Term Memory (LSTM [6]), Feed-forward (FNN [7]), and Recurrent (RNN [8]) Neural Network models are used to train the dataset in this work. Then, a systematic architectural comparison is made across these structural paradigms to identify the model with the lowest error, highest stability near workspace boundaries, and optimal computational efficiency for physical deployment.

2. Aims and Objectives

A 6-DoF inverse kinematic solution with high precision, accuracy, and reduced computational latency is designed with data-driven neural network models. By systematically analysing the performance of different network topologies, this research aims to

identify how direct functional regression compares against sequential and gated architectures in minimizing training and validation errors, mitigating translation-rotation scale imbalances, and optimizing real-time prediction stability.

To achieve this aim, the following objectives are targeted:

- Set up the dataset of the IK as a CSV file containing the joint angles and end-effector positions for the ABB IRB120 6-DOF robot arm—defining the targets and features [9]—sourced from Kaggle.
- Utilize the Google Colab environment to preprocess the data using MinMaxScaler normalization [10,11] to reconcile unit scale discrepancies and individually train the dataset on the FNN, RNN, and LSTM models using the TensorFlow (Keras) learning framework paired with the Adam optimizer [12].
- Perform a rigorous validation and benchmarking analysis on the trained models to ascertain and compare their performance based on end-effector coordinate error (mean and maximum), *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), and R^2 score, thereby mapping the explicit engineering trade-offs between architectural complexity, structural tracking precision, and computational latency.

3. Related Works

The optimization of Inverse Kinematics (IK) utilizing neural networks is a highly active research domain driven by the continuous demand for enhanced positioning precision and angular orientation tracking in robotic manipulators. A core benefit of this data-driven shift is highlighted by Athulya et al. [13], who demonstrate that Artificial Neural Networks (ANNs) effectively eliminate the need for explicit, highly complex analytical modelling of nonlinear systems by yielding multiple joint configurations close to the target space. To evaluate these mappings methodically, recent deep learning frameworks have transitioned from standard linear models to specialized network optimization schemes. For example, Vaishnavi et al. [14] proposed a deep learning system that is explicitly trained on a vector-based MAE loss function. Their analysis of joint-angle predictions revealed accelerated computing times and minimal tracking discrepancies across both position and orientation spaces.

Building upon this, comparative benchmarking by Navya et al. [15] verified that Feed forward Neural Network (FNN) models significantly outperform traditional linear regressions and decision tree algorithms across standardized metrics, including MSE, Root Mean Squared Error (RMSE), MAE, and R^2 scores, while demonstrating robust global generalization. Despite the rapid inference capabilities typical of standard feedforward regression, architectural trade-offs remain a primary concern; Dorman & Gupta [16] observed that while FNN structures provide immediate, single-pass spatial solutions, they consistently encounter performance bottlenecks when balancing rigorous accuracy constraints against execution timing over complex pathways. This limitation is particularly pronounced as workspace geometry scales in complexity.

Nevertheless, data-driven approaches have shown notable versatility when extended to highly non-linear kinematics, such as tracking continuum robots with constant curvature, where real-time trajectory tracking achieved elevated execution efficiency [17]. Furthermore, when benchmarked directly against traditional numerical solvers, neural network formulations demonstrate distinct advantages. Bezzout et al. [18] revealed that while learning-based models achieve accuracy metrics comparable to theoretical, physics-based IK algorithms, the data-driven models possess significantly higher robustness and adaptability when subjected to structural variations in the robot's physical parameters and ambient environment variables.

To address the limitations of standard architectures in complex configuration spaces, alternative structural topologies have emerged. Garg et al. [19] argued that despite the marginal performance gains of standard multi-layer networks, they ultimately fall short when navigating kinematic redundancies and the severe non-linearities of continuum structures; to resolve this, they introduced Invertible Neural Networks (INNs), which achieved a significant R2 value of 0.92 and an RMSE value of 0.0058 mm, outperforming conventional learning models. Concurrently, researchers have investigated the structural sensitivity of neural models to varying robot link dimensions. Bouzid & Narayan [20] analysed the specific challenge of variable manipulator arm lengths—a factor that drastically impacts a system's global functionality—and demonstrated that systematically tuning network hyperparameters is an effective methodology for stabilizing IK solutions across diverse geometric scales. In terms of functional path execution, Esmat et al. [21] proposed an ANN architecture focused specifically on trajectory tracking, proving its capability to successfully guide a manipulator along dynamic reference paths.

Similarly, Peng et al. [22] utilized Back Propagation (BP) networks to govern a 6-DoF robot arm moving toward a target pose, discovering that optimizing the network's weight updates significantly minimized Cartesian workspace errors while accelerating global convergence speeds compared to standard BP baselines. Finally, the literature validates the scalability of these networks across both serial and parallel configurations. While Hlavac [23] and Manolescu & Secco [24] established the foundational viability of utilizing neural networks to map 3D serial manipulator coordinates, Khattab & El-Habrouk [25] successfully expanded this scope to parallel manipulators by implementing a dual-ANN architecture to simultaneously estimate forward and inverse kinematic mappings, yielding superior predictive and localization capabilities.

4. Methods

This work adapts the dataset by Baressi Šegota et al. [26,27]. In their dataset, they left out the roll, pitch and yaw of the tip to test for a method that will simplify the neural network training. In their experiment, they achieved a satisfactory R2 and MAE using Multilayer Perceptron (MLP) Regression Model. And it's on that basis that x, y, z for the end-effector is used instead of $(x, y, z, \alpha, \beta, \gamma)$.

4.1. Methodology

4.1.1. Inverse Kinematics (IK)

For a 6 DOF robotic arm, the IK calculates the values of the joint coordinates vs the desired pose of the end-effector in the 3D space (Figure 1). It holds:

$$\boldsymbol{T}_G = \boldsymbol{T}_0^6(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_e) \quad (1)$$

where

$\boldsymbol{T}_G$ = desired position and orientation of the tip;

$\boldsymbol{T}_0^6$ = forward kinematics transformation;

$\theta_1 \ldots\ldots \theta_e$ = Joint angles;

$\theta_6 = \theta_e$ (angle at the end-effector).

When the six joint angles are passed into T_0^6 function the resulting matrix takes the standard form—4 × 4 rigid-body transformation structure:

$$\boldsymbol{T}_G = \begin{bmatrix} R_{11} & R_{12} & R_{13} & x \\ R_{21} & R_{22} & R_{23} & y \\ R_{31} & R_{32} & R_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The upper-left 3 × 3 submatrix (R) represents the final 3D of the gripper, and the fourth column vector (x, y, z) represents the exact Cartesian position of the gripper center point in 3D space.

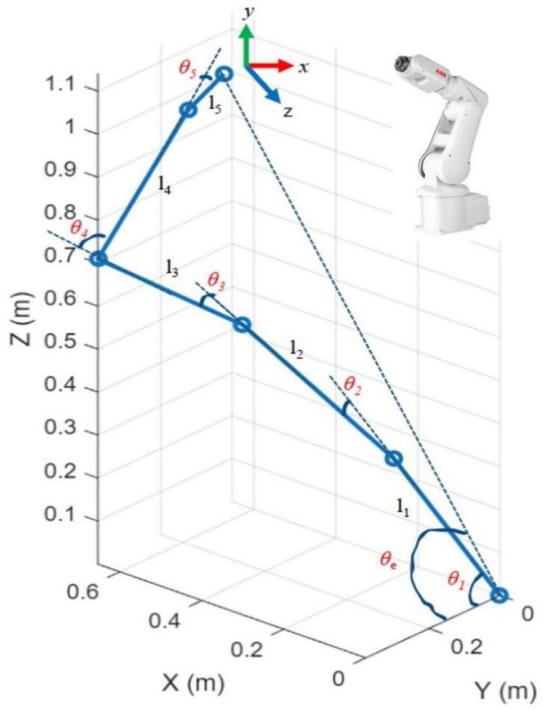


Figure 1. Representation of the 6 DOF ABB Robotic Arm.

4.1.2. Neural Networks

Neural Networks are a privileged approach when non-linear mappings are required: here this work models the relationship between the pose of the robotic arm's tip vs. its 6 joint angles. A set of training data for the NN is prepared through the implementation of the forward kinematic of the robot; then a position-only IK is performed as the input or features which is the end-effector's pose is mapped into the neural network to the output or target which are the joint angles, namely:

$$(x, y, z) \longrightarrow (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_e) \quad (3)$$

The methodology roughly mimics the human brain where artificial neurons replace the brain cells [28]. Neurons are implemented models where the feeding inputs are tuned through weights and then processed (Figure 2). When inputs are fed into the neuron, such inputs pass through two units, the summation and activation unit, and the result is then fed to other neurons on the next layer.

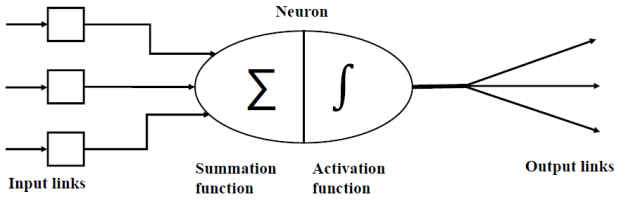


Figure 2. Mathematical Model of an artificial neuron.

The connections between neurons are balanced with numeric weights w_i^j, where *i* refers to the layer and j refers to the number of the neuron within that layer. These weights reinforce or weaken the strength and signal within a particular connection (Figure 3).

Meanwhile, when this neuron is connected to many other neurons it forms an ANN. Usually, ANNs can have 2 or more layers, that is an input layer and a hidden layer which can be made of one or more layers. The hidden layer is usually the place where "learning and processing" are embedded together with the output layer.

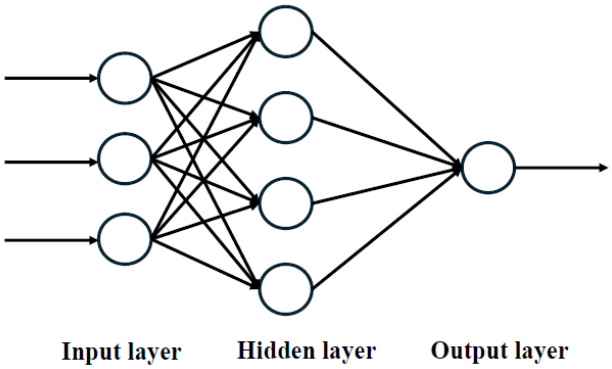


Figure 3. Multi-layer Neural Network Model.

(1) *Long-Short Term Memory (LSTM)*

LSTM networks are a type of RNN with the capability to learn and retain long-term information of sequential data. For the IK problem, the LSTM is mapping each pose of the tip to the corresponding joint angles with respect to time.

(2) *Feed-forward Neural Network (FNN)*

FNNs are a type of ANN allowing their signals to flow only in one direction—from the input layer, where information is received, to the hidden layer, that resolves the non-linear transformation, and then to the output layer, that predicts the target.

(3) *Recurrent Neural Network (RNN)*

Like LSTM, the traditional RNN is applied to time-dependent mapping of the robot arm end-effector's pose to its equivalent joint angles. Unlike FNNs and LSTMs, RNNs' feedback connections allow information from previous time steps, which is the only memory they maintain to influence their current state.

4.2. The Research Method

The IK dataset was downloaded from Kaggle and trained on *Google Colab* notebook using TensorFlow (Keras). Then the following 4 steps are applied:

(1) *Dataset Generation*—Each set of joint angles per time step of the ABB robot is calculated vs the pose of the end-effector.
(2) *Loading and Pre-processing of Data*—the CSV file containing the IK dataset is loaded. The features were identified as the input (x, y, z) and the target angles were identified as output $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$. Preprocessing of the dataset was done with the MinMaxScaler normalization to bound the input range between 0 and 1 then match the active high-gradient regions of the neural network activation functions such as sigmoid and tanh. The data is also split into training and validation sets (80/20) due to the independent static datasets without temporal dependencies. The validation sets directly measure the generalization capability vs other arbitrary points across the robot's workspace.
(3) *Perform Training (LSTM, FNN and RNN)*—FNN, as a standard baseline, provides fast direct mapping for real-time 6-DoF IK while RNN and LSTM leverage structural weight sharing and gated sequence loops respectively to better mirror the physical kinematic chain and resolve the scaling mismatches between translation and rotation. Each training model was defined with similar hyperparameters. The MSE loss function and the Adam optimizer were used because of the default learning rate (0.001) that helps to provide balance between speed and stability. The batch size = 32, dropout rate = 0.2, weight initialization = Xavier and LSTM sequence return =

False. The number of hidden layers, and epochs were defined and indicated on the table below.

(4) *Evaluation*—This measures the performance of the training vs the models' dataset, according to the following metrics:

i. *Mean Squared Error (MSE)*—The *MSE* represents the average square error between the predicted and true angular values in the dataset and returns a single value, in squared radians. In supervised learning, the goal of training with MSE is to reduce loss and it is calculated as:

$$MSE_{inner\text{-}product} = \sum_{i=1}^{N} \frac{(\theta_{pred} - \theta_{true})^2}{N} \quad (4)$$

ii. *Mean Average Error (MAE)*—The *MAE* reports the average magnitude of the absolute errors between the predicted joint angle configurations and the true target joint angle configurations, measured in radians. The error is defined as follows:

$$MAE = \sum_{i=1}^{N} \frac{(\theta_{pred} - \theta_{true})}{N} \quad (5)$$

iii. R^2—The R^2 refers to the best fit on the model. It measures how the predicted joint angles correctly align with the true joint angles. The formula for the R^2 is described below:

$$R^2 = 1 - \frac{\sum(\theta_{true} - \theta_{pred})^2}{\sum(\theta_{true} - \bar{\theta}_{true})^2} \quad (6)$$

iv. *End-effector error* (e_{pos})—The e_{pos} known as the Cartesian position error or the Euclidean distance error is a measure of the predicted joint angle's deviation of the end-effector in space from its desired pose. This deviation is measured in meters, and it is calculated with the expression below:

$$e_{pos} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \quad (7)$$

where Δ is the difference between true position and the predicted position.

In considering the end-effector error, the mean and maximum positions are evaluated. While the end-effector error (mean) indicates the average tracking precision of each of the models across all testing configurations to prove that the models successfully learned the geometric and trigonometric manifolds of the 6-DoF physical chain, the end-effector error (maximum) represents the single worst prediction made by the models out of the entire testing dataset, i.e. it is the critical safety and boundary limits the models must not exceed after suffering a localized failure or missing its target.

The overall dataset applied to the 3 models is made of 15,000 samples of the end-effector Cartesian coordinates (x, y, z) and 6 values of the joint angles (θ_1, θ_2, θ_3, θ_4, θ_5, θ_6).

The overall architecture of the proposed neural networks is summarized in Table 1: the LSTM one is made of 2 layers with 64 neurons each combined with tanh and sigmoid activation functions, a dense output layer of 3 neurons follows. The FNN model has 3 ReLU-activated hidden layers (64, 32, and 16 neurons, respectively), and a dense output layer (3 neurons). The RNN model is made of 2 layers, namely an RNN layer (64 neurons) plus a ReLU activation function in the hidden layer, and a linear dense output layer (3 neurons). All models have 6 neurons each at their input layer. The number of layers and units was determined through a trial-and-error tuning process.

Table 1. Hyperparameter and Network Architecture.

Neural Network	Layer Type	No. of Units	Activation Function	No. of Parameters
LSTM	LSTM	64	Tanh + Sigmoid	16,896
	LSTM	64	Tanh + Sigmoid	33,280
	Dense	3	Linear	195
FNN	Dense	64	ReLU	448
	Dense	32	ReLU	2,080
	Dense	16	ReLU	528
	Dense	3	Linear	51
RNN	Simple-RNN	64	ReLU	4,224
	Dense	3	Linear	195
LSTM FNN RNN	Epochs	100		
	Learning rate	Default (0.001)		
	Optimizer	Adam		
	Loss Function	MSE		

Input data are normalized using "min-max" normalization technique. The training process is terminated once the total number of epochs reaches 100. The joint angle prediction performance for accuracy and precision by deep learning models—LSTM, FNN and RNN, and the resulting end effector mean/max are shown in Table 2 below.

Table 2. Performance Metrics of the Models.

Metrics	LSTM	FNN	RNN
MSE (rad²)	0.0001	0.000318	0.000435
MAE (rad)	0.0045	0.013849	0.016221
R^2	0.9970	0.971564	0.961267
End-effector error—Mean (m)	0.009000	0.027816	0.032528
End-effector error—Max (m)	0.046402	0.103012	0.120484

5. Results and Discussion

Results for each of the 3 models are reported in Table 2 and Figures 4, 5, and 6, respectively. According to these results, comparisons are made across the metrics and plots to establish the model that fits the dataset best.

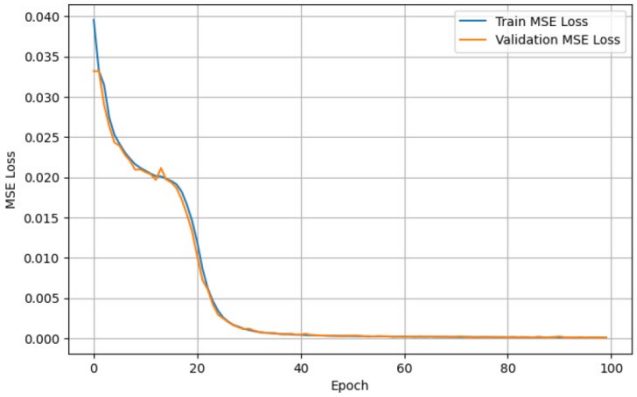


Figure 4. Training and Validation MSE Loss (LSTM).

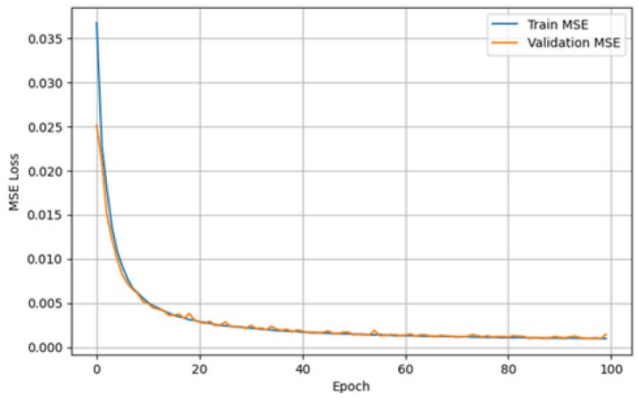


Figure 5. Training and Validation MSE Loss (FNN).

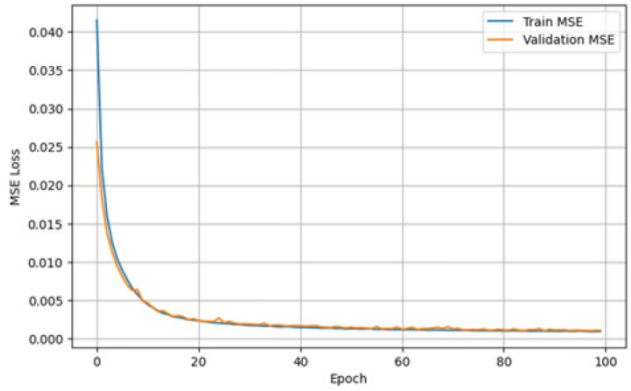


Figure 6. Training and Validation MSE Loss (RNN).

According to Table 2, the results associated with the LSTM network are better than the results of the FNN and RNN models. The table shows that the value of MSE is almost close to zero because it was computed in normalized space with a MinMax scaling, MAE value at 0.0045 rad indicates better predictive accuracy, R2 at 0.9970 shows that the model generalizes better than the other two models. On the other side, FNN performs better than RNN in all metrics, particularly in terms of the R2 value, which depicts a higher variance on the validation set. For MSE and MAE, FNN shows that it predicts joint values better than RNN. Even though RNN has the highest MSE showing a larger error, it still performs better when compared with the results of Shaar & Ghaeb [3] as indicated in Table 3 although they used a larger dataset (600,006 target values) over 15 epochs.

Meanwhile, this is not exactly the case for FNN model as Dorman & Gupta [16] reported that their model with its pre-defined hyperparameters such as the Bayesian regularization to eliminate overfitting and an MSE to serve as the error function gave an extremely low MSE value which means a better performance. More so, they applied two activation functions – the hyperbolic tangent sigmoid and the linear transfer function at the hidden and output layers respectively for optimized performance. They computed 20,000 data points at 80% training and 20% testing over 1000 epochs for the six revolute joints of the ABB IRB 6700. Their work has included the Euler angles that depict orientation, thus having their input as a 6D pose ($x, y, z, \alpha, \beta, \gamma$). While they have focused on precise trajectory planning, this work considers a generalized way of solving IK problems which are highly non-linear across global space and can be prone to MSE spikes due to trajectory ambiguity. But, in all the proposed LSTM model achieves a near zero-error value at 0.0001 rad^2 which is an improvement from the work of Vaishnavi et al. [14] although for a 5-DoF robot arm, they did not specify their number of data points but have performed their training on 150 epochs and achieved its MSE in 0.4775 s. And as such, the MSE value of the proposed LSTM model at 0.0001 rad2 shows that the model has achieved better accuracy and precision when solving IK problems with a faster execution time of 0.4 s.

Table 3. MSE (rad^2) Benchmark Comparison of Models.

Models	Year	LSTM	FNN	RNN
Vaishnavi, et al. [14]	2022	0.0492	-	-
Dorman & Gupta [16]	2024	-	3.69×10^{-10}	-
Shaar & Ghaeb [3]	2023	-	-	0.0004565
Proposed	2026	0.0001	0.000318	0.000435

Figures 4–6 show that the proposed design performs well with a learning rate and the set of the model parameters within reasonable limits. They depict a consistent and steady decline in the loss error value, apart from Figure 4, which shows an elbow at epoch 20, signifying the phase where the network's optimization path transitions from coarse global manifold mapping to localized weight refinement. During this, gradient updates have largely minimized the macro-level kinematic error, leading to a stabilization of the learning curve. And as such the networks are tasked with processing a sequential timeline of Cartesian poses to predict a corresponding sequence of joint configurations. Because sequential inverse kinematics requires mapping continuous spatial trajectories rather than isolated coordinate points, the convergence trajectories of the training and validation curves provide explicit diagnostic proof that the recurrent layers have generalized the continuous joint-space manifolds without localized overfitting. Furthermore, the close alignment of the training and validation MSE loss curve for the LSTM model in Figure 4 throughout the training period indicates that the model is not overfitting and has a good generalization capability.

6. Conclusion

This work presents 3 neural network models to perform the IK of a 6 DOF Robotic arm. The LSTM system achieved remarkable values in terms of MSE, MAE, R^2, and end-effector error (mean and max). These values will help to accomplish accuracy and precision in joint and Cartesian space predictions [29–31]. However, the study embeds the limitation of being performed in a simulated environment, which may not have a complete reflection of complexities that could be present in the real world. Also, while the LSTM model may present certain challenges in providing robust solutions for IK problems, it has performed numerically well in this experiment. These findings establish a basis for the three neural network models application in IK problems.

Limitation

Real-world deployment of trained policies usually faces significant challenges during the AI framework integration. Considerable effort was devoted to further testing the dataset, which was carried out on both CoppeliaSim and RoboDK software applications [32,33]. Initially, the raw joint angles were input into both software applications, but they gave output values (position) that were different from those in the dataset. The test was taken a step further by integrating the trained IK neural network model with RoboDK for real-time testing on the ABB IRB 120 robot simulation environment. The deployment technique involved creating a dedicated Python virtual environment, installing compatible versions of TensorFlow, NumPy, and the RoboDK API, and developing a deployment pipeline used for loading the trained model, preparing normalized pose inputs, predicting robot joint angles, and transferring motion commands to RoboDK. However, when the actual target joint angles were used, it predicted different values and did not match the position values trained. This indicates limited prediction accuracy. However, during setup, some observations were made: version incompatibility between TensorFlow/Keras, the Google Colab training environment, and the local deployment environment; errors identified during model deserialization; and weight mismatches between Dense and Neural Network layers,

which prevented effective execution of the trained model locally. These clearly show the variations that occur when translating a synthetic dataset void of noise to a real-world scenario. As such, further research should apply these three models to real-world environment while considering these challenges as a mitigation strategy to achieve the required performance accuracy.

Author Contributions: Conceptualization, C.O.O., N.B. and E.L.S.; methodology, C.O.O.; software, C.O.O.; validation, C.O.O.; formal analysis, C.O.O.; writing—original draft preparation, C.O.O.; writing—review and editing, C.O.O. and E.L.S.; supervision, N.B. and E.L.S. All the authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Ethical Approval: Not applicable

Informed Consent Statement: Not applicable.

Data Availability Statement: Data used in this paper are available from Baressi Šegota et al. [26].

Acknowledgement: This work has been presented in Coursework form in fulfillment of the requirements for the MSc Robotics Engineering for the student Christopher Okoh from the School of Computer Science and the Environment, Liverpool Hope University.

AI assisted to fine-tune the research aim before the experiment was performed. All AI-generated content has been reviewed and approved by the author. The intellectual content, design decisions, and analysis within this paper remain solely the author's own work.

We would like to express our thanks to the Team of our School for their support.

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

References

[1] Tammishetty, S.; Semwal, V.B.; Pathak, Y.; Joshi, D. Inverse kinematic solution using neural networks for multimodal inputs and optimization in constrained workspace. *IEEE Sensors Lett.* **2025**, *9*, 1–4.

[2] Srisuk, P.; Sento, A.; Kitjaidure, Y. Inverse kinematics solution using neural networks from forward kinematics equations. In *Proceedings of the 2017 9th International Conference on Knowledge and Smart Technology (KST)*, Chonburi, Thailand, 1–4 February 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 61–65.

[3] Shaar, A.; Ghaeb, J.A. Intelligent solution for inverse kinematic of industrial robotic manipulator based on RNN. In *Proceedings of the 2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, Amman, Jordan, 22–24 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 74–79.

[4] Zhang, N.; Zhang, Y.; Cheng, J.; Ma, C. Inverse kinematics solution for six-DOF serial robots based on BP neural network. In *Proceedings of the 2017 Chinese Automation Congress (CAC)*, Jinan, China, 20–22 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1154–1157.

[5] Kendall, A.; Cipolla, R. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 21–26 July 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 6555–6564.

[6] Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780.

[7] Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536.

[8] Elman, J.L. Finding structure in time. *Cogn. Sci.* **1990**, *14*, 179–211.

[9] ABB IRB120 6 DOF Robot Manual. Available online: https://library.e.abb.com/public/7aa0711a20fa41c49a8fdf3bbc3d5bb0/IRB120-Rev.J-ROBO149EN_D.pdf (accessed on 7 May 2026).

[10] Patro, S.; Sahu, K.K. Normalization: A Preprocessing Stage. *arXiv* **2015**, arXiv:1503.06375.

[11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.

[12] Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.

[13] Athulya, P.S.; Ranjith Kumar, S.; Neetha, G. A computer vision approach for the inverse kinematics of 2 DOF manipulators using neural network. In *Proceedings of the 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS)*, Thiruvananthapuram, India, 3–5 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 80–85.

[14] Vaishnavi, J.; Singh, B.; Vijayvargiya, A.; Kumar, R. Deep learning framework for inverse kinematics mapping for a 5 DoF robotic manipulator. In *Proceedings of the 2022 IEEE International Conference on Power Electronics, Drives and Energy Systems (PEDES)*, Jaipur, India, 14–17 December 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.

[15] Navya, M.; Muralidhara; Jain, N.; Pai, V.; Rao, M.; Rao, A. Estimation of inverse kinematics solutions of a 2D planar robotic manipulator using feed-forward neural network. In *Proceedings of the 2024 International Conference on IoT Based Control Networks and Intelligent Systems (ICICNIS)*, Bengaluru, India, 17–18 December 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1040–1044.

[16] Dorman, J.D.; Gupta, A.; Gill, H.S. FNN-based inverse kinematics for efficient trajectory planning in industrial robots. In *Proceedings of the 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS)*, Hassan, India, 23–24 August 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 1–6.

[17] Ghoul, A.; Kara, K.; Djeffal, S.; Benrabah, M.; Hadjili, M.L. Inverse kinematic model of continuum robots using artificial neural network. In *Proceedings of the 2022 19th International Multi-Conference on Systems, Signals & Devices (SSD)*, Sétif, Algeria, 6–10 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1893–1898.

[18] Bezzout, H.; Aghoutane, B.; El Ghzaoui, M.; El Faylali, H. Inverse kinematic of differential drive robot using neural networks and Vrep simulation. In *Proceedings of the 2025 17th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Targoviste, Romania, 26–27 June 2025; IEEE: Piscataway, NJ, USA, 2025; pp. 1–5.

[19] Garg, S.; Dudeja, S.; Rastogi, V. Inverse kinematics of tendon driven continuum robots using invertible neural network. In *Proceedings of the 2022 2nd International Conference on Computers and Automation (CompAuto)*, Paris, France, 18–20 August 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 82–86.

[20] Bouzid, R.; Narayan, J.; Gritli, H. Investigating neural network hyperparameter variations in robotic arm inverse kinematics for different arm lengths. In *Proceedings of the 2024 Third International Conference on Power, Control and Computing Technologies (ICPC2T)*, Raipur, India, 18–20 January 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 351–356.

[21] Esmat, M.; Abdel-Nasser, M.; Saif, A.-W.A. Solving inverse kinematics for 3R manipulator using artificial neural networks. In *Proceedings of the 2023 20th International Multi-Conference on Systems, Signals & Devices (SSD)*, Mahdia, Tunisia, 20–23 February 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 116–120.

[22] Peng, Y.; Peng, Z.; Lan, T. Neural network based inverse kinematics solution for 6-R robot implement using R package neuralnet. In *Proceedings of the 2021 5th International Conference on Robotics and Automation Sciences (ICRAS)*, Wuhan, China, 11–13 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 65–69.

[23] Hlavac, V. Inverted kinematics of a redundant manipulator with a MLP neural network. In *Proceedings of the 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Maldives, 16–18 November 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–5.

[24] Manolescu, V.D.; Secco, E.L. Design of a 3-DOF Robotic Arm and Implementation of D-H Forward Kinematics. In *Proceedings of the 3rd Congress on Intelligent Systems (CIS 2022)*, Bengaluru, India, 5–6 September 2022; Springer: Singapore, 2022; Volume 1, pp. 569–583.

[25] Khattab, Y.; Zidane, I.F.; El-Habrouk, M.; Rezeka, S. Solving kinematics of a parallel manipulator using artificial neural networks. In *Proceedings of the 2021 31st International Conference on Computer Theory and Applications (ICCTA)*, Alexandria, Egypt, 11–13 December 2021; IEEE: Piscataway, NJ, USA, 2022; pp. 84–89.

[26] Baressi Šegota, S.; Car, Z.; Anđelić, N.; Lorencin, I.; Mrzljak, V.; Musulin, J.; et al. *Robot Kinematics Dataset*; Kaggle: San Francisco, CA, USA, 2021.

[27] Baressi Šegota, S.; Anđelić, N.; Mrzljak, V.; Lorencin, I.; Kuric, I.; Car, Z. Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator. *Int. J. Adv. Robot. Syst.* **2021**, *18*, 1729881420925283.

[28] Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2010.

[29] Secco, E.L.; Magenes, G. A Feedforward Neural Network Controlling a 3-DOF Finger. In *Proceedings of the Fifth International Conference on Cognitive and Neural Systems (ICCNS)*, Boston, MA, USA, 30 May–2 June 2001; Boston University Press: Boston, MA, USA, 2001.

[30] Secco, E.L.; Magenes, G. A feedforward neural network controlling the movement of a 3-DOF finger. *IEEE Trans. Syst. Man Cybern. Part A* **2002**, *32*, 437–445.

[31] Magenes, G.; Secco, E.L. Teaching a robot with human natural movements. In *Proceedings of the XI Winter Universiads Conference on Biomechanics and Sports*, Udine, Italy, 2003; Springer: Vienna, Austria, 2004; pp. 135–144.

[32] CoppeliaSim. CoppeliaSim Robotics Simulation Software. Available online: https://www.coppeliarobotics.com/ (accessed on 8 May 2026).

[33] RoboDK. RoboDK Industrial Robot Simulation Software. Available online: https://robodk.com/ (accessed on 8 May 2026).

Copyright: © 2026 by the author(s). *Journal of Artificial Intelligence and Control Systems* published by CoSci Press. This is an open access publication under the Creative Commons Attribution 4.0 International License (CC BY 4.0).