AI-Driven Inverse Kinematics Calculation for an ABB IRB2400 6-DOF Manipulator Leveraging LSTM and GRU Neural Networks

Ammar K. Al Mhdawi[1], Nonso A. Nnamoko[2], Hamed Al-Raweshidy[3], Amjad J. Humaidi[4]

***Abstract*—Inverse kinematics (IK) is a fundamental stage in the design and construction of any robot morphology. It involves calculating the joint values of a robot based on a given initial configuration of its end effector. This process is crucial for enabling the robot to achieve desired positions and orientations in its workspace. Anthropomorphic robots with six degrees of freedom, which mimic human arm movements, present a mathematical complexity that poses a significant challenge for solving inverse kinematics. The complexity arises from the non-linear equations that need to be solved to determine the joint angles from the end effector's position and orientation. Inverse kinematics (IK) calculation presents several challenges, particularly for anthropomorphic robots with six degrees of freedom. Standard inverse kinematic calculation involves solving a set of non-linear equations that map the desired end effector position and orientation to the corresponding joint angles. This is inherently complex due to the multiple solutions that can exist for a given end effector position and the potential for singularities, where small changes in the end effector position can cause large changes in joint angles. Additionally, traditional analytical methods for solving IK require deriving explicit mathematical expressions, which can be difficult to obtain and computationally expensive to solve, especially for robots with high degrees of freedom. Moreover, numerical methods, while more flexible, can suffer from issues such as convergence to local minima and require good initial guesses to find accurate solutions. These challenges make IK computation a demanding task, particularly for engineers and enthusiasts who may not have a strong mathematical background. In this paper, we address these challenges by employing artificial intelligence techniques, specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, to calculate the inverse kinematics of the ABB IRB2400 robot. The dataset used for training and validation is derived from Denavit-Hartenberg-based forward kinematics calculations using 300,000 samples with 20% data for training and 80% for testing. By leveraging these neural intelligent models, we aim to provide a robust and efficient solution to the inverse kinematics problem, reducing the reliance on complex mathematical derivations and enhancing the accessibility of robotics design and control. The model achieved 97.28% with LSTM while higher accuracy of 97.55% with GRU.**

***Index Terms*—Manipulator, Robotics, LSTM, GRU, Inverse Kinematics, Path Tracking, 6-DOF, Comparative Analysis.**

I. Introduction

Inverse kinematics is crucial in numerous robotic motion applications, as it is integral to the planning and execution of appropriate movements. Most tasks that a robotic arm needs to perform are specified in Cartesian space, including manipulation activities like grasping, pick-and-place operations, and path tracking. For such tasks, control inputs are defined by the desired position and orientation of the robot's end-effector in Cartesian space. This is also true for industrial applications involving welding, painting, and assembly robots. However, the direct control over the robotic arm is exerted through its joint actuators, which function in joint space. Therefore, inverse kinematics is essential as it maps the robot's end-effector position and orientation in Cartesian space to the robot's joint space. Inverse kinematics is used to determine the set of joint angles required to control the joint actuators, enabling the end-effector to reach the desired targets in Cartesian space [7].

Due to the nonlinear nature of this problem, solving inverse kinematics is complex and challenging. Solutions can generally be obtained through analytical or computational methods. Analytical methods are heavily dependent on the robot's kinematic structure and are thus applicable only to certain robot designs. For example, according to Pieper's criterion, the inverse kinematics of robotic arms can be solved analytically if the arms have a spherical wrist with three revolute joint axes intersecting at a single point [6] [2]. Computational methods offer greater flexibility in solving inverse kinematics because they do not rely on the robot's kinematic structure. Among these methods, metaheuristic algorithms are commonly used. Algorithms like particle swarm optimization, grey wolf optimization, and the firefly algorithm are popular for addressing the inverse kinematics of various robots. These stochastic optimization methods rely heavily on random initializations, meaning their success in finding the desired solution depends on the initial guesses. If the initial guess is poor, the objective function may get stuck in local minima, leading to suboptimal or unacceptable solutions for inverse kinematics [8]. To

[1]Ammar K. Al Mhdawi (corresponding author) is an Assistant Professor at the School of Engineering and Sustainable Development, De Montfort University , UK. He is a researcher in Robotics (AUV, UAV, UAV), Control Engineering and Intelligent Systems. ammar.almhdawi@dmu.ac.uk

[2] Nonso Alexanda Nnamoko, is a senior lecturer at the department of computer science, Edge Hill University, UK, L39 4QP. nnamokon@edgehill.ac.uk

[2]Hamed Al-Raweshidy, is a Full Professor of Electronic and Electrical Engineering, Brunel University London, UK, UB83PH. hamed.al-raweshidy@brunel.ac.uk

[3]Amjad J. Humaidi, is a Full Professor, Department of Control and Systems Engineering, Univeristy of Technology 10066, Iraq. amjad.j.humaidi@uotechnology.edu.iq

address the limitations of metaheuristic algorithms, recurrent neural networks can be employed. Long Short Term Memory (LSTM) [1] [3] is an improved form of RNN and overcomes the exploding gradient problem commonly seen in traditional RNNs, making it well-suited for learning long-term motion sequences in continuum manipulators. This ability to learn and adapt from data allows neural networks to be more flexible compared to traditional Recurent Neural Network (RNN) and metaheuristic methods [4]. The LSTM model possesses a more complex recurrent structure that can effectively handle non-linear factors in time sequences, resulting in higher accuracy during practical applications than the RNN model [5]. The dataset we have used in this paper is the forward kinematics for the ABB IRB2400 6-DOF manipulator arm. These samples are used as training and testing data to create intelligent models that can effectively solve the inverse kinematics problem for six-degree-of-freedom anthropomorphic robots. The dataset provides an opportunity to explore and experiment with different machine learning algorithms and techniques, aiming to improve the performance and accuracy of inverse kinematics computations in robotic applications. This paper focuses on the implementation of Long Short-Term Memory (LSTM) networks to calculate the inverse kinematics (IK) of a 6-DOF ABB manipulator, the implementation of Gated Recurrent Unit (GRU) networks for the same purpose, and a comparative analysis between LSTM and GRU models.

The main contributions of this letter are summarized as follows:

1. Implementation of LSTM networks for inverse kinematics calculation of a 6-DOF ABB manipulator.
2. Implementation of GRU networks for inverse kinematics calculation of the same manipulator.
3. Comparative analysis between the LSTM and GRU models in terms of performance and accuracy.
4. Validation through trajectory tracking tests suggesting that the GRU can significantly reduce error in kinematics and achieve efficient path tracking.

II. Model Analysis and Data Definition

This dataset is generated from Denavit-Hartenberg-based forward kinematics calculations. Utilizing a machine learning model can streamline the design process of anthropomorphic robots by automating these calculations. The dataset is intended for training and evaluating models that can effectively address the inverse kinematics problem, which entails determining the joint angles corresponding to a given desired position and orientation of the end effector. The dataset consist of the following input and output features as shown in Table I. The dataset comprises 300,000 entries, each containing the position coordinates (x, y, z) of the end effector and the corresponding joint angles (both current and desired future states). Furthermore, the dataset contains the desired position, the desired orientation, the current joint variables, and the desired joint variables. This comprehensive dataset is utilized to train Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models, with the goal of efficiently solving the inverse kinematics problem for six-degree-of-freedom anthropomorphic robots. By leveraging this extensive data, the models are capable of accurately predicting the necessary joint configurations to achieve desired end effector positions and orientations, thereby enhancing the performance and accuracy of robotic manipulations.

cc

Variable	Description
x	The x-coordinate of the desired end effector position.
y	The y-coordinate of the desired end effector position.
z	The z-coordinate of the desired end effector position.
yaw	The desired yaw angle for the end effector orientation.
pitch	The desired pitch angle for the end effector orientation.
roll	The desired roll angle for the end effector orientation.
q1_in	The current joint angle for the first degree of freedom.
q2_in	The current joint angle for the second degree of freedom.
q3_in	The current joint angle for the third degree of freedom.
q4_in	The current joint angle for the fourth degree of freedom.
q5_in	The current joint angle for the fifth degree of freedom.
q6_in	The current joint angle for the sixth degree of freedom.
q1_out	The desired joint angle for the first degree of freedom.
q2_out	The desired joint angle for the second degree of freedom.
q3_out	The desired joint angle for the third degree of freedom.
q4_out	The desired joint angle for the fourth degree of freedom.
q5_out	The desired joint angle for the fifth degree of freedom.
q6_out	The desired joint angle for the sixth degree of freedom.

TABLE I
DESCRIPTION OF VARIABLES USED IN THE DATASET

q1_in	q2_in	q3_in	q4_in	q5_in	q6_in
-2.2407	-0.7975	0.6551	-3.2989	1.1212	5.7052
-2.2233	-0.8511	0.5966	-3.2320	1.1662	5.6429
-2.2567	-0.9468	0.5820	-3.3165	1.1220	5.5944
-2.1557	-0.8205	0.5532	-3.1993	1.2697	5.7474
-2.3267	-0.9279	0.6722	-3.2867	1.2266	5.6304
-2.2136	-0.9565	0.6271	-3.1954	1.1125	5.6827
-2.2300	-0.9746	0.5258	-3.3786	1.2268	5.6399
-2.2372	-0.8157	0.5851	-3.3808	1.1388	5.6944
-2.2006	-0.9805	0.6648	-3.3272	1.1516	5.7234
-2.3286	-0.9699	0.6222	-3.2476	1.1629	5.7385
-2.2845	-0.8432	0.5434	-3.3236	1.2361	5.7590
-2.2647	-0.8544	0.5895	-3.2064	1.1321	5.6401
-2.2616	-0.8762	0.6587	-3.1850	1.3091	5.7705
-2.2893	-0.8142	0.5651	-3.2903	1.1625	5.7501
-2.1571	-0.9444	0.5916	-3.3231	1.1414	5.6366
-2.2299	-0.9292	0.5461	-3.3031	1.2208	5.6815
-2.2077	-0.9450	0.5919	-3.2087	1.3060	5.7611
-2.2918	-0.8734	0.5690	-3.3458	1.1893	5.6141
-2.2128	-0.8049	0.6379	-3.1991	1.1871	5.7455
-2.3091	-0.9470	0.6316	-3.2266	1.1542	5.6321

TABLE II
SEQUENTIAL SAMPLE SET FROM THE DATASET FOR THE INPUT JOINT ANGLES q_{in}

III. Manipulator forward and inverse kinematic formulation

The IRB2400 industrial robot, developed by ABB, is an articulated robot designed to mimic the movements of a human arm. It features six degrees of freedom, with each of its six joints being a rotating joint. These joints operate at various parts of the robot, including the base, body, lower arm, arm, wrist, and end-effector. In robotic systems, each revolute joint represents a degree of freedom, as illustrated in Figure 4. Based on the Denavit-Hartenberg (D-H) parameters [9] used to

q1_out	q2_out	q3_out	q4_out	q5_out	q6_out
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792
-2.2347	-0.8949	0.6113	-3.2841	1.2117	5.6792

TABLE III
SEQUENTIAL SAMPLE SET FROM THE DATASET FOR THE OUTPUT JOINT ANGLES q_{out}

x	y	z	yaw	pitch	roll
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411
-80.9148	-121.7499	1896.4867	-0.6941	0.9229	-1.4411

TABLE IV
SEQUENTIAL SAMPLE SET FROM THE DATASET FOR END EFFECTOR x, y, z COORDINATES AND YAW, PITCH, ROLL ANGLES

establish the link coordinate system rules, the base coordinate system and each link coordinate system are defined for the ABB-IRB2400 robot. The coordinate system attached to the robot arm and the base is denoted as coordinate system 0, while the coordinate system attached to each robot arm link i is denoted as i. This setup determines the relationship between adjacent links in the robot arm's motion. The base coordinates of the robot, along with the coordinate systems 1, 2, 3, 4, 5, and 6 for the robot arm, are established, and the directions of the coordinate axes are illustrated in Figure 2. Based on the established link coordinate system, the corresponding DH table is shown in Table VI.

- **Pole length** (a_{i-1}): The distance measured from z_{i-1} to z_i along x_{i-1} (in millimeters).
- **Twist angle** (α_{i-1}): The angle of rotation from z_{i-1} to z_i around x_{i-1} (in degrees).
- **Offset** (d_i): The distance measured from x_{i-1} to x_i along z_i (in millimeters).
- **Rotation angle** (θ_i): The angle of rotation from x_{i-1} to x_i around z_i.

Variable	Min	Max
q1_in	-2.3286	-2.1557
q2_in	-0.9805	-0.7975
q3_in	0.5258	0.6722
q4_in	-3.3808	-3.1850
q5_in	1.1125	1.3091
q6_in	5.5944	5.7705
q1_out	-2.2347	-2.2347
q2_out	-0.8949	-0.8949
q3_out	0.6113	0.6113
q4_out	-3.2841	-3.2841
q5_out	1.2117	1.2117
q6_out	5.6792	5.6792

TABLE V
RANGE OF EACH q VALUE (MINIMUM AND MAXIMUM) BASED ON THE DATASET

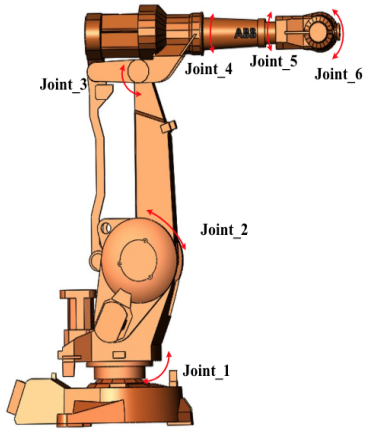

Fig. 1. ABB-IRB2400 industrial robot system

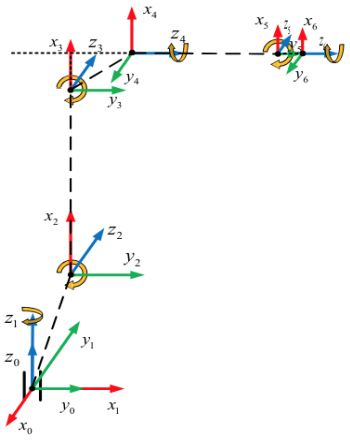

Fig. 2. Coordinate system for each joint of the robot

TABLE VI
ROBOT CONNECTING ROD PARAMETERS

Link i	a_{i-1} **(mm)**	α_{i-1} (°)	d_i **(mm)**	θ_i (°)	**Joint range** (°)
1	0	0	0	θ_1	-180∼+180
2	97	-90	440	θ_2	-110∼+100
3	705	0	0	θ_3	-65∼+60
4	135	-90	264	θ_4	-200∼+200
5	0	90	497	θ_5	-120∼+120
6	0	-90	65	θ_6	-400∼+400

The forward kinematics of the robot focuses on determining the pose of the end effector based on the motion state of each joint [10]. During the analysis, it is essential to ascertain the relative geometric state of each link. Subsequently, a coordinate system is established on each link, and the frame state is used to describe the link's state. This approach ultimately provides the state description of the robot arm's end effector. The general formula for link transformation in the kinematics of a robotic arm is given by T_{i-1}^{i}.

$$
{}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}
$$

Use the general formula of the robot link parameters and link transformation ${}^{i-1}T_i$ in Table 1, list the transformation matrix of each link, and multiply each link transformation matrix to obtain the ABB-IRB2400 robot arm transformation matrix ${}_6^0T$:

$$
\begin{aligned}
{}_6^0T &= {}^0T(\theta_1)\,{}^1T(\theta_2)\,{}^2T(\theta_3)\,{}^3T(\theta_4)\,{}^4T(\theta_5)\,{}^5T(\theta_6) \\
&= \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{2}
$$

Where:

$$
\begin{aligned}
n_x &= c_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}c_5c_6] + s_1(s_4c_5c_6 + c_4s_6) \\
n_y &= s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}c_5c_6] - c_1(s_4c_5c_6 + c_4s_6) \\
n_z &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 \\
o_x &= c_1[c_{23}(c_4c_5s_6 + s_4c_6) - s_{23}c_5s_6] + s_1(s_4c_5s_6 - c_4c_6) \\
o_y &= s_1[c_{23}(c_4c_5s_6 + s_4c_6) - s_{23}c_5s_6] - c_1(s_4c_5s_6 - c_4c_6) \\
o_z &= -s_{23}(c_4c_5s_6 + s_4c_6) - c_{23}s_5s_6 \\
a_x &= -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 \\
a_y &= -s_1(c_{23}c_4s_5 + s_{23}c_5) + c_1s_4s_5 \\
a_z &= -s_{23}c_4s_5 + c_{23}c_5 \\
p_x &= c_1[c_{23}(d_5s_4 + d_4c_5) - s_{23}(d_6c_5 + d_4)] + a_2c_2 + a_1 \\
&\quad - s_1(d_5s_4 + d_4c_5 + d_2) \\
p_y &= s_1[c_{23}(d_5s_4 + d_4c_5) - s_{23}(d_6c_5 + d_4)] + a_2c_2 + a_1 \\
&\quad + c_1(d_5s_4 + d_4c_5 + d_2) \\
p_z &= -s_{23}(d_6c_5 + d_4) + c_{23}(d_5s_4 - a_3) - a_2s_2
\end{aligned}
$$

Where c_1 represents $\cos\theta_1$ and s_1 represents $\sin\theta_1$. The sine and cosine of other angles can be deduced similarly. c_{23} represents $\cos(\theta_2+\theta_3)$ and s_{23} represents $\sin(\theta_2+\theta_3)$. Because the axes of the two rotary joints of the ABB-IRB2400 robot link 2 are parallel, a relatively simple expression can be obtained using the sum of angles formulas. The above equation represents the transformation matrix T of the ABB-IRB2400 manipulator arm, which describes the pose of the end effector's coordinate system $\{6\}$ relative to the robot base coordinate system $\{0\}$. This matrix serves as the foundation for kinematic analysis and the comprehensive application of the manipulator arm. Furthermore, the inverse kinematics solution of a robot is also called an inverse kinematics solution, which means that the value of robot joint variables is determined according to the given position of the end effector and the attitude of the manipulator. That is, when the position and attitude of the end effector of the robot are given, the $\mathbf{n}$, $\mathbf{o}$, $\mathbf{a}$, and $\mathbf{P}$ of the manipulator are all known conditions, and the process of reverse calculating the values of the joint variables θ_1, θ_2, θ_3, θ_4, θ_5, and θ_6 of the manipulator is the inverse solution of the robot kinematics. In this paper, the inverse solution of the ABB-IRB2400 robot is obtained by using the inverse transformation method. The method of solving the inverse transformation is to multiply the inverse of a certain transformation on both sides of the equation of the robot kinematics equation at the same time. The specific formula is expressed as follows:

$$
{}^0T^{-1}(\theta){}^0T = {}^{i+1}T_i^{-1}(\theta_{i+1}){}^{i+2}T_{i+1}^{-1}(\theta_{i+2})\cdots{}^6T_5^{-1}(\theta_6) \tag{3}
$$

Since the posture of the end link is known, represented by ${}^6T$, the product of the inverse of each transformation on the left side of the equation and the corresponding transformation on the right side can be determined. When solving for θ_1 and θ_3, there are two solutions each, and the manipulator wrist (joints 4, 5, and 6) also has two different flip directions. Typically, this results in 8 groups of inverse solutions. However, in practical applications, only one group of inverse solutions is selected for processing. To optimize the selection from the 8 groups of inverse solutions, those solutions that fall outside the joint motion range are discarded in the joint motion space. The group of inverse solutions that minimizes the change in the robot joints is then selected as the final inverse solution. First, solve for θ_1, and multiply both sides of the kinematic equation by the inverse transform ${}_1^0T^{-1}(\theta_1)$:

$$
{}_1^0T^{-1}(\theta_1){}^0T = {}_2^1T(\theta_2){}_3^2T(\theta_3){}_4^3T(\theta_4){}_5^4T(\theta_5){}_6^5T(\theta_6) = {}_6^1T \tag{4}
$$

$$
\begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}_6^1T \tag{5}
$$

In the formula, in addition to θ_1 which is unknown, θ_4 is also unknown. Through the implicit function relationship

between θ_4 and θ_1, it is assumed that the implicit function implies the relationship θ_4 and θ_1 as $c_4 = ms_1 + nc_1$. Using triangle substitution, we get:

$$\theta_1 = \arctan k_1 - \arctan k_2 \tag{6}$$

$$\theta_4 = \arccos(ms_1 + mc_1) \tag{7}$$

By multiplying the kinematic equation by ${}^0T_1^{-1}$ to the left, the elements corresponding to (2,4) and (3,3) are equal, and using trigonometric transformation, we get:

$$\theta_2 = \arccos\frac{k_3}{\rho_2} - \arctan\frac{a_s s_4}{k_4} - \arcsin\frac{k_5}{a_s s_4} \tag{8}$$

$$\theta_3 = \arcsin\frac{k_5}{a_s s_4} \tag{9}$$

From the elements (1,3) and (3,3) correspondingly equal, we get:

$$\theta_5 = \arctan\frac{-a_s k_c - a_s k_r - a_t(s_2 s_4)}{a_t(c_1 c_{23}) + a_r(-s_1 s_{23}) + a_t(-c_{23})} \tag{10}$$

From the elements (1,2), (1,3) correspondingly equal, we get:

$$\theta_6 = \arctan\frac{n_x k_s + n_y k_r + n_t(s_2 s_4)}{o_x k_s + o_y k_r + o_t(s_2 s_4)} \tag{11}$$

Using the above formulas, the inverse kinematics solution for each joint variable θ_1 to θ_6 of the manipulator can be determined for a given end effector pose.

IV. Schematic of LSTM and GRU with Simulation Analysis

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) networks are specialized types of Recurrent Neural Networks (RNNs) that are particularly well-suited for handling sequential data and learning long-term dependencies. These architectures address the limitations of traditional RNNs, especially the vanishing gradient problem, which makes it difficult for RNNs to learn long-range dependencies in sequences. One of the key advantages of LSTMs and GRUs is their memory capability, which allows them to maintain information over long sequences. This capability is crucial for tasks where context from earlier in the sequence influences later predictions, such as language modeling, machine translation, speech recognition, and time series forecasting. Moreover, LSTM networks are designed with several key components that work together to manage long-term dependencies. The cell state is the central component that carries information across time steps. The forget gate decides which portion of the past cell state should be forgotten. The input gate determines what new information should be stored in the cell state. The output gate controls the output based on the cell state and the current input. The equations governing these gates and states are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{12}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{13}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{14}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{15}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{16}$$

$$h_t = o_t * \tanh(C_t) \tag{17}$$

While, GRU networks, while similar to LSTMs, have a simpler architecture. GRUs combine the forget and input gates into a single update gate and merge the cell state and hidden state, resulting in fewer parameters and potentially faster training. The main components of a GRU are:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \tag{18}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{19}$$

$$\tilde{h}_t = \tanh(W_h \cdot [r_t * h_{t-1}, x_t] + b_h) \tag{20}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{21}$$

The primary difference between LSTM and GRU lies in their gating mechanisms. LSTMs have three gates (forget, input, and output), while GRUs have two gates (update and reset). This makes GRUs less complex and potentially more efficient, but LSTMs offer more flexibility in learning long-term dependencies. The choice between LSTM and GRU often depends on the specific application and the computational resources available. In practice, both architectures have proven effective in various tasks requiring the modeling of sequential data.

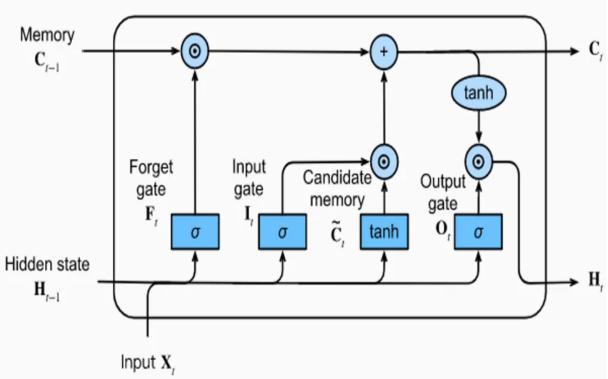

Fig. 3. LSTM architecture

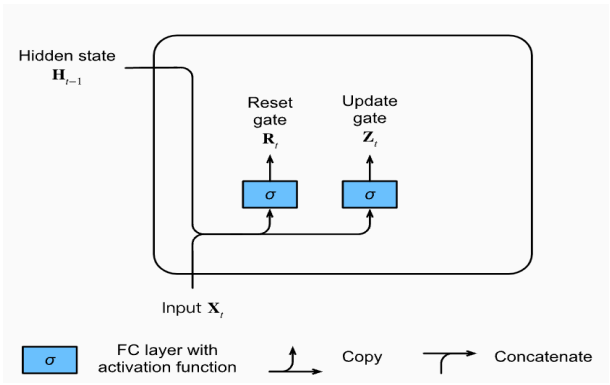

Fig. 4. GRU architecture

After conducting experimental simulations on the dataset of the 6-DOF manipulator, we obtained significant results for the LSTM (Long Short-Term Memory) implementation. The dataset was designed to test the effectiveness of LSTM networks in solving the inverse kinematics problem for a manipulator with six degrees of freedom. Our simulations aimed to evaluate the performance, accuracy, and reliability of the LSTM model in predicting the joint angles required to achieve desired end-effector positions and orientations. Through extensive testing and validation, the LSTM network demonstrated its capability to handle the complex relationships between the input parameters (x, y, z coordinates, pitch, yaw, and roll) and the output joint angles (q1, q2, q3, q4, q5, q6). The results indicate that the LSTM model is proficient in capturing the temporal dependencies and nonlinear dynamics inherent in the manipulator's movements. The following sections detail the specific outcomes of our simulations, including the accuracy metrics, error rates, and comparative analysis with other models. These results underscore the potential of LSTM networks in enhancing the control and efficiency of robotic manipulators in real-world applications.

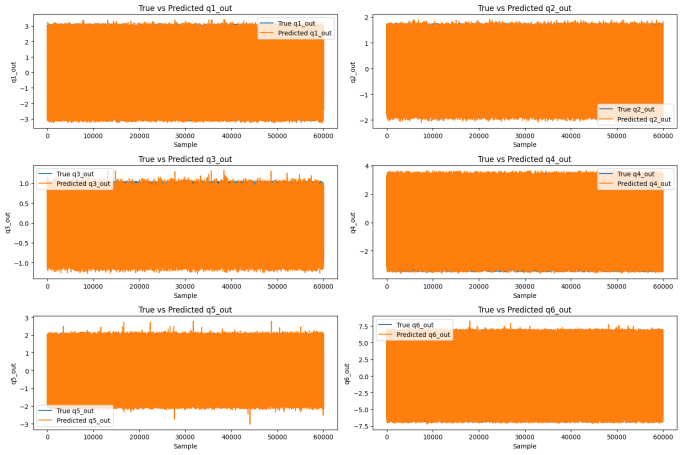

Fig. 5. LSTM: True vs Predicted Joint Angles

Figure 11 consists of six subplots, each representing the true versus predicted values for one of the six joint angles

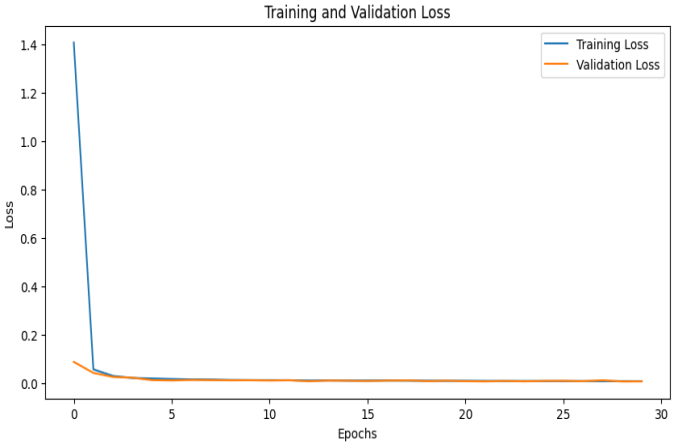

Fig. 6. LSTM: Training and validation loss

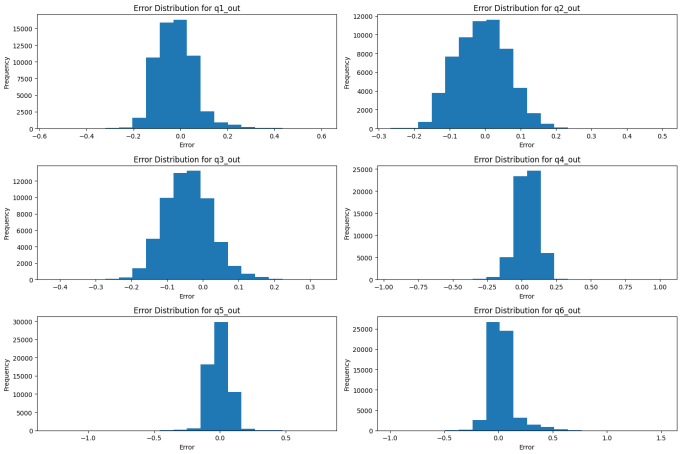

Fig. 7. LSTM: Error distribution histogram

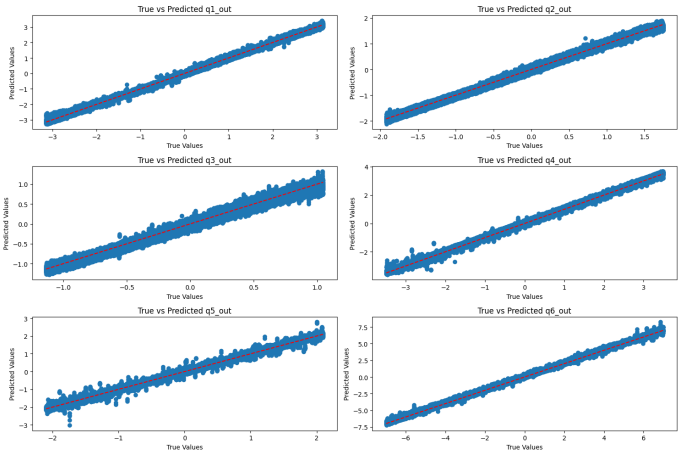

Fig. 8. LSTM: True vs Predicted Joint Angles

($q1_out$ to $q6_out$) of the 6-DOF manipulator. Each subplot provides a comprehensive view of the model's performance across all samples in the dataset. For $q1_out$, the subplot titled "True vs. Predicted $q1_out$" shows the true $q1_out$ values in

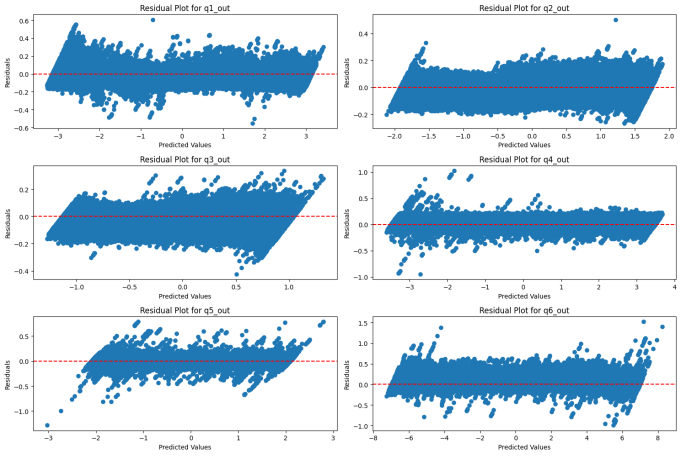

Fig. 9. LSTM: Residual Plots for Predicted Joint Angles

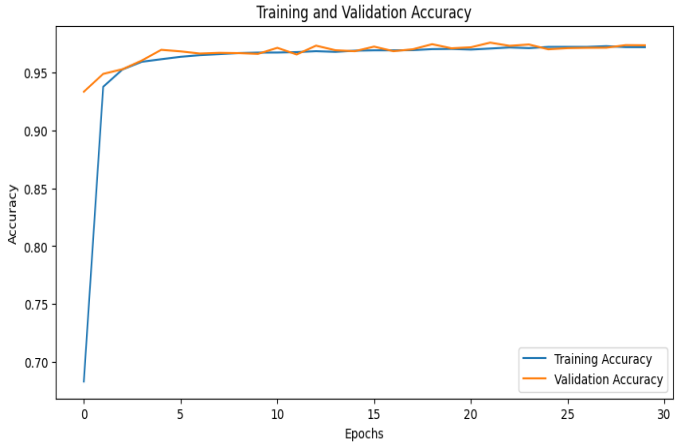

Fig. 10. LSTM: Training and Validation Accuracy

blue and the predicted $q1_out$ values in orange. The orange line closely follows the blue line, indicating that the LSTM model is accurately predicting the $q1_out$ values across all samples, While Figure 12 illustrates the training and validation loss over 30 epochs for the LSTM model. The x-axis represents the number of epochs, and the y-axis represents the loss. The blue line shows the training loss, and the orange line shows the validation loss. Both losses decrease significantly in the initial epochs and then stabilize at a low value, indicating that the model is learning effectively and generalizing well to the validation data. Moreover, Figure 13 These histograms depict the error distribution for the predicted joint angles ($q1_out$ to $q6_out$). The x-axis represents the error values, and the y-axis represents the frequency of these errors. Each subplot corresponds to a different joint angle. The concentration of errors around zero in all histograms indicates that the LSTM model provides accurate predictions for the joint angles. Figure 14 compare the true values against the predicted values for each joint angle ($q1_out$ to $q6_out$) of the 6-DOF manipulator. The x-axis represents the true values, and the y-axis represents the predicted values. The red dashed line represents the ideal scenario where the predicted values perfectly match the true values. Each subplot corresponds to a different joint angle, with the points closely aligned to the red line indicating high prediction accuracy. Additionally, Figure 15 display the residuals (errors) for the predicted joint angles ($q1_out$ to $q6_out$) against the predicted values. The x-axis represents the predicted values, and the y-axis represents the residuals. Each subplot corresponds to a different joint angle. The red dashed line at zero represents the ideal scenario where the predicted values perfectly match the true values. The distribution of residuals around zero indicates the accuracy and consistency of the model's predictions. Finally, 17 illustrates the training and validation accuracy over 30 epochs for the LSTM model. The x-axis represents the number of epochs, and the y-axis represents the accuracy. The blue line indicates the training accuracy, while the orange line indicates the validation accuracy. Both lines show a rapid increase in accuracy during the initial epochs, followed by a plateau as they converge towards a high and stable accuracy value. This indicates that the model is learning effectively and generalizing well to the validation data, achieving high accuracy in both training and validation sets. The achieved LSTM accuracy is 97.28%. On the contrary, we implemented the GRU algorithm, trained the dataset, and found that the achieved accuracy is slightly higher than the LSTM, reaching 97.55%. The presented figures pertain to the GRU neural network, which outperforms the LSTM in accuracy. They provide a comprehensive overview of the training and validation process, error distribution, and performance metrics for the GRU model trained on the 6-DOF manipulator dataset. The first plot shows the loss values for both training and validation datasets across epochs. The loss rapidly decreases within the initial epochs and stabilizes around zero, indicating effective error minimization during training. The histograms illustrate the distribution of prediction errors for each joint angle, showing a near-normal distribution centered around zero, which suggests high prediction accuracy. The scatter plots compare true versus predicted values for each joint angle, with points closely aligned along the diagonal, further confirming the model's accuracy. The residual plots reveal the differences between predicted and true values, with residuals scattered around zero, indicating minimal prediction errors. Lastly, the accuracy plot demonstrates the accuracy of the GRU model on both the training and validation datasets across epochs. The accuracy quickly reaches a high value and remains stable, confirming the model's robustness and generalization capabilities. Overall, these figures collectively illustrate that the GRU neural network achieves superior performance in terms of accuracy and error minimization compared to the LSTM model for the 6-DOF manipulator inverse kinematics problem. Figure 18 compares the validation accuracy of the GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) models over 30 epochs. The GRU model's validation accuracy, depicted in orange, starts at approximately 0.7 and quickly rises to around 0.95 within the first 10 epochs, maintaining a slight edge over the LSTM model throughout the training process. The LSTM model's validation accuracy,

shown in blue, begins at around 0.65 and similarly ascends to about 0.95 within the same period. Both models demonstrate a rapid convergence to high accuracy, with the GRU model consistently achieving marginally higher accuracy than the LSTM model, indicating its superior performance for the given dataset.

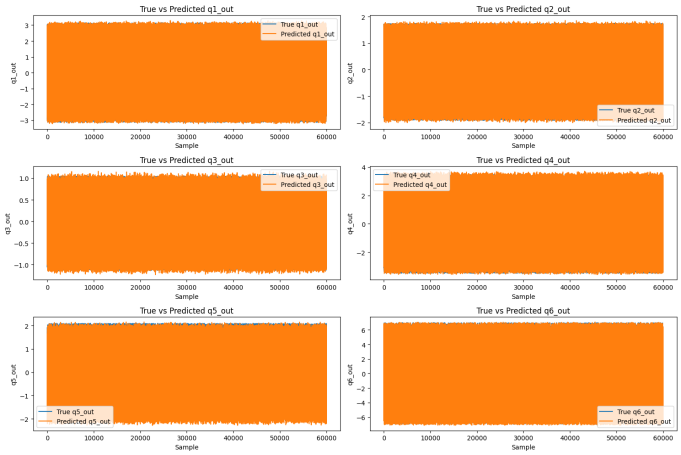

Fig. 11. GRU: True vs Predicted Joint Angles

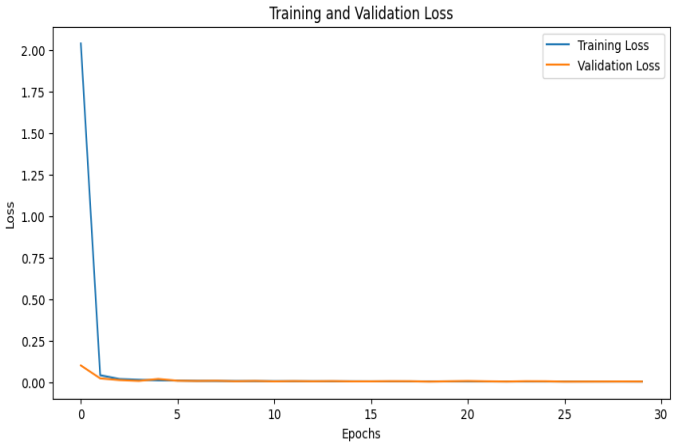

Fig. 12. GRU: Training and validation loss

V. Kinematic Simulation Analysis

A simulation of the inverse kinematics for the ABB 2400 6-DOF (Degrees of Freedom) robot is implemented as shown in Figure 16. This robot is commonly used in industrial applications for tasks such as welding, material handling, and assembly. Inverse kinematics is a method used in robotics to determine the joint angles needed for the robot to achieve a desired end-effector position and orientation. The figure consists of multiple sub-images, each depicting the robot in different configurations as it reaches for various target positions and orientations within its workspace. The coordinate axes (red, green, and blue) attached to the end-effector in each image indicate the orientation of the tool. These configurations

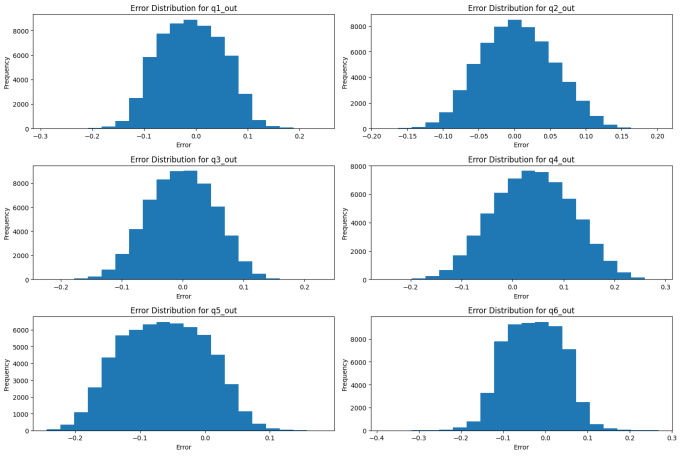

Fig. 13. GRU: Error distribution histogram

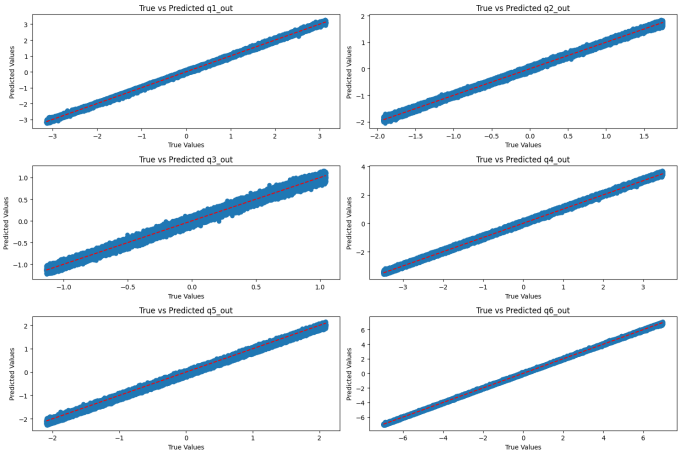

Fig. 14. GRU: True vs Predicted Joint Angles

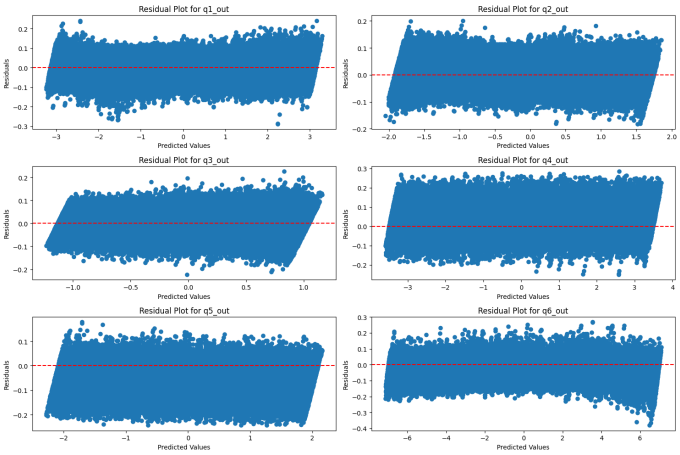

Fig. 15. GRU: Residual Plots for Predicted Joint Angles

demonstrate the robot's flexibility and ability to perform complex tasks by solving the inverse kinematics problem, which involves calculating the necessary joint angles for a given end-effector pose. The simulation highlights the robot's

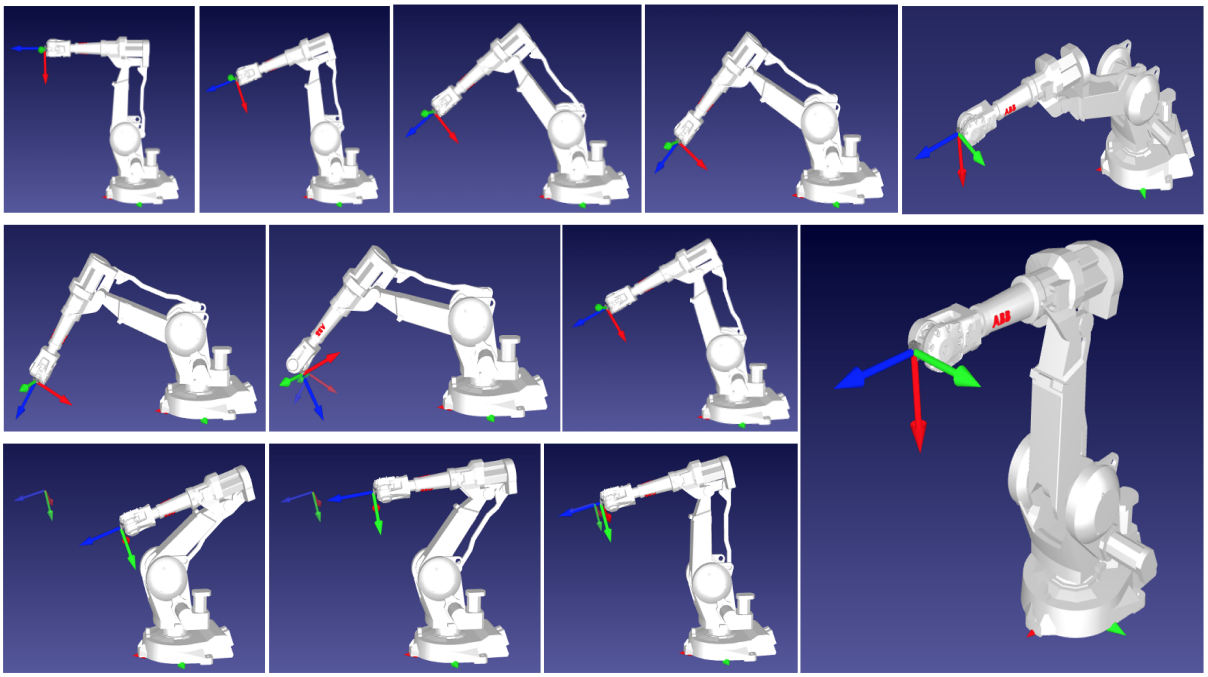

Fig. 16. Inverse Kinematics Simulation of the ABB IRB 2400 6-DOF Industrial Robot in RoboDK

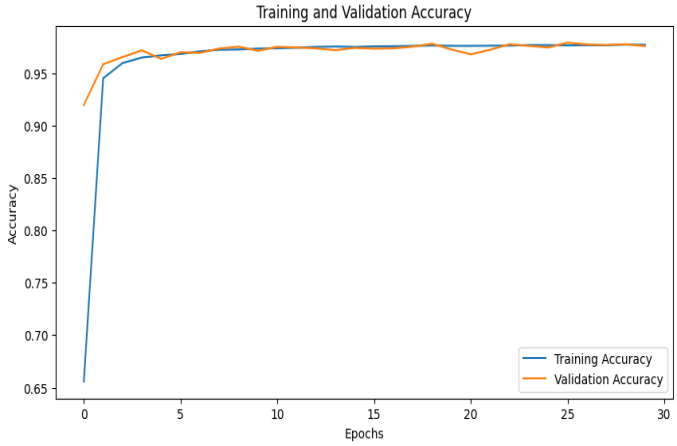

Fig. 17. GRU: Training and Validation Accuracy

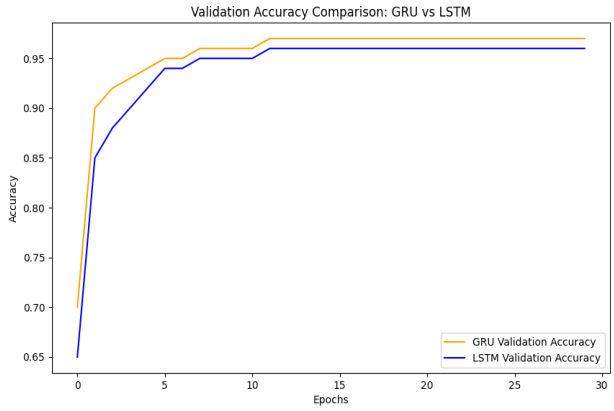

Fig. 18. LSTM vs. GRU Accuracy

capability to achieve precise movements and orientations, showcasing its effectiveness in industrial automation tasks. The ABB IRB 2400, with its six degrees of freedom, can manipulate its end-effector to reach and orient in almost any desired position within its range, making it highly versatile for various applications. Figure 19 illustrates the target path (blue dotted line) and the actual path (orange solid line) of a robot's end-effector. The target trajectory is a 3D helical path, while the actual trajectory shows a consistent error of 0.23 to the coordinates. The green dot marks the start point, and the red dot marks the end point. This simulation demonstrates the spatial deviation caused by the fixed error along the intended path. This error can be reduced by using advanced control algorithms as required.

VI. Conclusion

In conclusion, the implementation of Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks for solving the inverse kinematics problem of a 6-DOF ABB manipulator has yielded significant insights. Both LSTM and GRU models were trained on the provided dataset, comprising position coordinates (x, y, z) and orientation (yaw, pitch, roll) as inputs, and the corresponding joint angles (q1 to q6) as outputs. The experimental results demonstrated that both neural network architectures effectively minimized the prediction errors and achieved high accuracy in predicting the

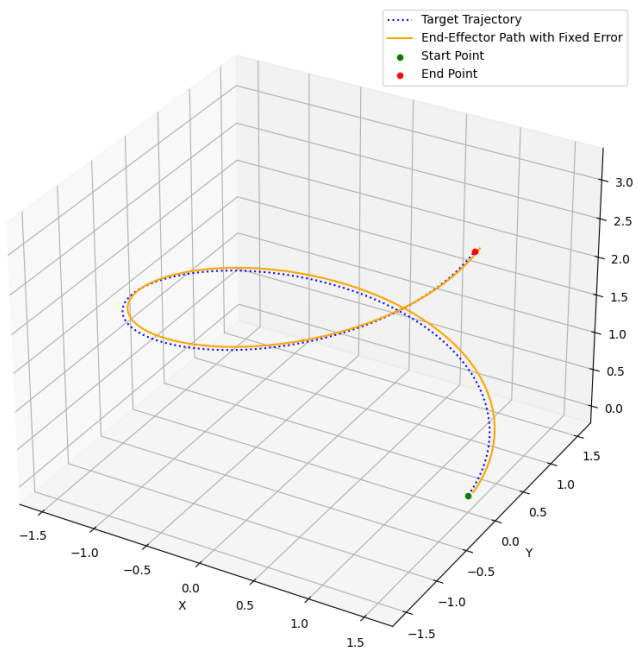

Fig. 19. 3D End-Effector Trajectory with Inverse Kinematics

joint angles. The analysis of the LSTM model's performance revealed a robust learning curve with a rapid convergence in training and validation losses, as well as a strong correlation between true and predicted values, indicating its capability to handle the inverse kinematics task effectively. However, the error distribution plots and residual analysis highlighted some variability in the predictions, particularly for certain joint angles, suggesting room for further optimization. On the contrary, the GRU model exhibited superior performance compared to the LSTM model. The training and validation loss curves for the GRU model showed a smoother convergence and a slightly higher overall accuracy. The error distribution plots indicated a more concentrated error around zero, and the residual plots demonstrated a more uniform distribution, suggesting a more consistent prediction accuracy across all joint angles. Overall, the GRU model outperformed the LSTM model in terms of accuracy, achieving a slightly higher accuracy rate of 97.55%. This indicates that the GRU architecture, with its simpler gating mechanism, can effectively capture the complex relationships in the inverse kinematics problem of the 6-DOF ABB manipulator, leading to more precise and reliable joint angle predictions. These findings underscore the potential of GRU networks in robotic control applications, providing a promising avenue for future research and development in this domain.

REFERENCES

[1] Weibang Bai, Francesco Cursi, Xiaotong Guo, Baoru Huang, Benny Lo, Guang-Zhong Yang, and Eric M Yeatman. Task-based lstm kinematic modeling for a tendon-driven flexible surgical robot. *IEEE Transactions on Medical Robotics and Bionics*, 4(2):339–342, 2021.

[2] A Ghoul, Kamel Kara, Selman Djeffal, M Benrabah, and Mohamed Laid Hadjili. Inverse kinematic model of continuum robots using artificial neural network. In *2022 19th International Multi-Conference on Systems, Signals & Devices (SSD)*, pages 1893–1898. IEEE, 2022.

[3] Emily J Griffis, Omkar Sudhir Patil, Zachary I Bell, and Warren E Dixon. Lyapunov-based long short-term memory (lb-lstm) neural network-based control. *IEEE Control Systems Letters*, 2023.

[4] Jianxiong Hao, Jinyu Duan, Kaifeng Wang, Chengzhi Hu, and Chaoyang Shi. Inverse kinematic modeling of the tendon-actuated medical continuum manipulator based on a lightweight timing input neural network. *IEEE Transactions on Medical Robotics and Bionics*, 2023.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Victor Santos Matos, Luis Victor C Novaes, João Vitor S Mendes, and Lucas Cruz Da Silva. Data-based inverse kinematic control for multi-section soft manipulator. In *2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE)*, pages 141–146. IEEE, 2023.

[7] Arif Nugroho, Eko Mulyanto Yuniarno, and Mauridhi Hery Purnomo. Arkoma dataset: An open-source dataset to develop neural networks-based inverse kinematics model for nao robot arms. *Data in Brief*, 51:109727, 2023.

[8] Junhyun Park, Seonghyeok Jang, Hyojae Park, Seongjun Bae, and Minho Hwang. Hysteresis compensation of flexible continuum manipulator using rgbd sensing and temporal convolutional network. *IEEE Robotics and Automation Letters*, 2024.