Neural Network Approach to the Inverse Kinematic Control of Robotic Manipulator

Agbonoga Dauda Okumede	**ENG1704257**
Ahekobuwa Nosamudianaede Praise	**ENG1704258**
Ahueansebhor Emmanuel Osemuahun	**ENG1704259**
Aihie Daniel Aisosa	**ENG1704261**

PROJECT SUPERVISOR

Engr. Ebu-Nkamaodo O. T

IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE BACHELOR OF ENGINEERING

CERTIFICATION

This is to certify that the project work titled "Neural Network Approach to the Inverse Kinematic Control of Robotic Manipulators" was carried out by:

Agbonoga Dauda Okumede	**ENG1704257**
Ahekobuwa Nosamudianaede Praise.	**ENG1704258**
Ahueansebhor Emmanuel Osemuahun.	**ENG1704259**
Aihie Daniel Aisosa	**ENG1704261**

Of the Department of Mechanical Engineering, Faculty of Engineering, University of Benin, in partial fulfillment of the requirement for the award of Bachelor of Engineering, (B. Eng.) in Mechanical Engineering

……. …….

ENGR. EBU-NKAMAODO O. T DATE

(PROJECT SUPERVISOR)

……. …….

DR. I.B OWUNNA DATE

(PROJECT CO-ORDINATOR)

……. …….

PROF. GODFREY OMONEFE ARIAVIE DATE

(HEAD OF DEPARTMENT)

DEDICATION

We dedicate this project to the Almighty God and all who have supported us throughout our academic journey.

ACKNOWLEDGEMENTS

We express our sincere gratitude to our project supervisor, **ENGR. EBU-NKAMAODO O.T**, for his immense support and contributions towards the success of our project. We also wish to appreciate all the members and staff of the Department of Mechanical Engineering, headed by **PROFESSOR GODFREY OMONEFE ARIAVIE**, at the University of Benin in Benin City. Our sincerest thanks also go to our beloved parents for their financial and emotional support, prayers, and encouragement throughout our studies at the University of Benin.

Acknowledgements

ABSTRACT

Robots are becoming more relevant across multiple industries; therefore, it is important for engineers to innovate in all aspects of this technology. This Project explores the concept of inverse kinematic control in robotics, its significance, difficulties and explores a solution using Neural networks. It presents an analysis of the forward kinematics of various robot arms-using the Denavit -Hartenberg method- as a way to generate the data set for training and testing the Neural network, until a suitable performance benchmark was reached.

The information on the robot arms that were analyzed was gotten from manufacturers publications, open to the public. Some of the robot arms considered includes; ABB IRB 1600; a 6dof robot arm, FANUC LR Mate 200iC; a 6dof robot arm, The Universal Robots UR10; a 6dof robot arm, among others.This project considers one of the simplest machine learning models for regression analysis, the Artificial Neural Network (ANN)

The trained model displayed high accuracy in predicting the suitable joint angles, with all trained models having a R squared value above 0.70

Table of Contents

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

1.1. Background to study

A robot manipulator can be referred to as a mechanical component that is designed to manipulate various objects in its environment with the aim of carrying out various tasks. In present day Robotic manipulators are used in many various fields such as medicine, machine manufacturing, space exploration, military, etc. In these fields, robots are given tasks that are too complex, dangerous or impossible for humans to do. The introduction of robotics into an industry is usually marked by improved productivity and profitability of the industry.

Robot manipulators can perform various tasks depending on the industry and problem application. These tasks can include material handling, welding, painting, assembling of parts, packaging, handling hazardous materials, operations.

Robotic manipulators for a task are chosen based on parameters such as work envelope (which is the region or space were the robot can operate in) and the degree of freedom of the robotic manipulator, the degree of freedom determines the type of motion the end effector/tool of the robotic arm can under go and also the type of position the tool/end effector can assume in space, with a low degree of freedom robotic arm having limited types of motion and poses, and a high degree of freedom robotic arm has higher variability in the type of motion it can undergo and poses it can assume.

Figure 1-1 A robotic manipulator holding a camera

1.2. Statement of problem

Robots are becoming more relevant across multiple industries; therefore, it is important for engineers to innovate in all aspects of this technology.

The concept of inverse kinematic control in robotics is significant as it greatly aids the automation of processes using Robots, this is an important problem as most of the times in the industry robots are required to do tasks that can only be accomplished with inverse kinematics (only the end effector details as known), tasks which may include, surgery, painting, machining, 3d printing, etc. However, inverse kinematic equations are difficult to solve analytically therefore it becomes important that we have methods that give inverse kinematics solutions quickly with a high degree of accuracy.

Methods that can be used in solving inverse kinematics problems of robot manipulators include;

- Abridged analytical methods
- Numerical analysis methods
- Machine learning techniques

1.3. Aim

The aim of this project is to build an Artificial Neural Network model capable of solving the inverse kinematics problem of 6 DOF Robotic Manipulators and implement in their control

1.4. Objectives

- Identifying neural network models relevant to the solution of inverse kinematics of complex robotic manipulators
- Build a neural network capable of solving the inverse kinematics of complex robotic manipulators

1.5. Scope of Project

This project covers

- Making a forward kinematic analysis of the robotic arm to generate joint angles and end effector data pairs.
- Building, Training and testing an Artificial Neural Network model for the inverse kinematics of the selected robotic arms with the data gotten from their forward kinematics analysis.
- Analyzing and studying the results from the ANN model

1.6. Methodology

- Research and selection of popular robotic arms with useful applications
- Research on methods for forward kinematics and inverse kinematic analysis of robot arms
- Research on suitable Artificial Intelligence models for Inverse kinematics of selected robot arms
- Analyze and generate data sets on the selected robot arms
- Build and train Artificial Intelligence on the datasets generated
- Test and evaluate the performance of the models

- Draw conclusions based on the results

Chapter 2

Literature Review

2.1. Robotic manipulators: significance and uses

Robotics is a rapidly growing and versatile field with widespread applications in manufacturing, healthcare, transportation, and various other industries. Robots effectively undertake dangerous, repetitive, or arduous tasks, enhancing productivity, efficiency, and safety. Additionally, they offer opportunities for companionship, education, and entertainment, enriching our lives in multiple dimensions.(Wagaa et al., 2023)

In manufacturing, robots aid in welding, assembly, and painting, augmenting productivity and product quality while minimizing workplace injuries. In healthcare, they contribute to surgery, rehabilitation, and companionship for elderly patients, improving care quality and reducing healthcare costs. Robots play vital roles in transportation through self-driving cars, delivery services, and airport maintenance, enhancing safety, efficiency, and convenience. In education, robots engage students in subjects like science, engineering, and mathematics, making learning more interactive and captivating. Moreover, robots perform entertaining tasks, from show performances to providing companionship and interactive gaming experiences.

Robotic technology continues to advance, making robots more sophisticated, versatile, and affordable. As they expand their capabilities, robots will find even more extensive applications in various aspects of our lives. However, challenges such as cost, safety, and public acceptance remain to be addressed. Despite these challenges, the benefits of robotics, including increased productivity, enhanced safety, reduced costs, and improved product and service quality, outweigh the drawbacks. As robots continue to evolve, they hold the potential to revolutionize multiple industries and positively impact our daily lives.

A Robotic manipulator is a machine that is used to manipulate objects in the environment. It is a type of robot that has a structure consisting of several rigid links connected by joints. The joints allow the links to move relative to each other, which gives the manipulator the ability to move its end-effector (the part of the manipulator that interacts with the environment) in a variety of ways. The number of links in a robotic manipulator can vary. The links are typically made of metal, such as aluminum or steel, but they can also be made of other materials, such as carbon fiber. The joints in a robotic manipulator can be either rotary or linear. Rotary joints allow the links to rotate around a fixed axis, while linear joints allow the links to move back and forth in a straight line. The type of joints that are used in a robotic manipulator determines the range of motion that the manipulator has. For example, a manipulator with rotary joints can only rotate the links, while a manipulator with linear joints can only move the links back and forth. The type of joints also affects the speed and accuracy with which the manipulator can move.

The end-effector of a robotic manipulator is the part of the manipulator that interacts with the environment. The end-effector can be a variety of different devices, such as a gripper, a welding torch, or a camera(fig 1-1).

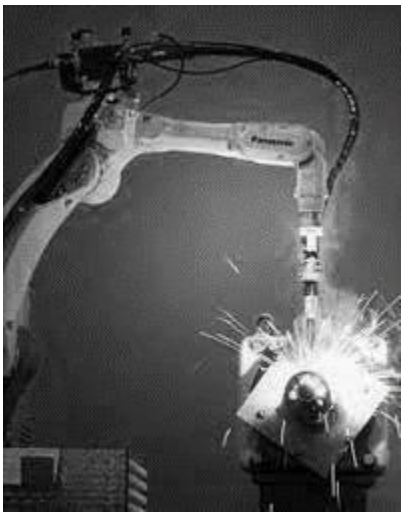

Figure 2-1 A robotic manipulator welding

The motion of a robotic manipulator is controlled by its control system. The control system receives input from the manipulator's sensors, such as cameras and proximity sensors, and uses this input to generate commands for the actuators. The control system must be able to track the position and orientation of the end-effector, as well as the forces and torques that are acting on it.

Some common applications for robotic manipulators include:

- Manufacturing: Robotic manipulators are used in manufacturing to automate tasks such as welding, painting, and assembly. (Filipescu et al., 2020)
- Medical: Robotic manipulators are used in medicine to perform surgery, deliver medication, and assist with rehabilitation.
- Search and rescue: Robotic manipulators are used in search and rescue to locate and extract victims from hazardous environments.
- Space exploration: Robotic manipulators are used in space exploration to assemble and maintain spacecraft, as well as to collect samples from other planets.

Robotic manipulators are versatile machines that can be used in a wide variety of applications. As the field of robotics continues to develop, we can expect to see even more innovative and sophisticated robotic manipulators being developed in the future.

- The number of degrees of freedom (DOF) of a robotic manipulator is the number of independent ways that the end-effector can move. The DOF of a robotic manipulator is determined by the number of links and joints in the manipulator.
- The reach of a robotic manipulator is the maximum distance that the end-effector can reach from the base of the manipulator. The reach of a robotic manipulator is determined by the length of the links in the manipulator.
- The payload of a robotic manipulator is the maximum weight that the end-effector can carry. The payload of a robotic manipulator is determined by the strength of the links and joints in the manipulator

2.1.1. Parallel manipulators

Parallel manipulators are made up of a series of parallel links and joints. The end-effector is attached to the end of the parallelogram formed by the links. Parallel manipulators have a smaller range of motion than serial manipulators, but they are easier to control. (Hu, 2014)

2.1.2. Serial manipulators

A serial manipulator is a type of robotic manipulator that is made up of a series of links and joints that are connected in a chain. The end-effector is attached to the last link in the chain. The links are typically made of metal, such as aluminum or steel, but they can also be made of other materials, such as carbon fiber. The joints in a serial manipulator can be either rotary or linear. Rotary joints allow the links to rotate around a fixed axis, while linear joints allow the links to move back and forth in a straight line. The motion of a serial manipulator is achieved by the rotation of the joints in the chain. The rotation of each joint causes the next link in the chain to rotate, and so on. This causes the end-effector to move in a specific direction. The range of motion of a serial manipulator is limited by the number of links and joints in the chain. Serial manipulators are the most common type of robotic manipulator. They are relatively simple to control and they are versatile.(Hu, 2014),

2.2. Forward Kinematics

The process of determining the end-effector's position and orientation (pose) in relation to a robot's base, given the joint angles and lengths of its links, is described by forward kinematics. A series of geometric transformations, typically involving the Denavit-Hartenberg parameters or transformation matrices, is used to calculate the transformation from one joint to the next, eventually reaching the end-effector. This critical calculation forms the basis for robot control, allowing the accurate navigation and manipulation of robotic arms in various applications, from manufacturing and surgery to space exploration and more. A fundamental understanding of

forward kinematics is deemed essential for programming robots to perform specific tasks with precision and efficiency.(Mohammad Amin, 2023)

2.3. Degrees of Freedom

Degrees of Freedom represents the number of independent parameters that define the configuration of a mechanical system. In the context of serial manipulators (also known as serial robotic arms), the DOF refers to the number of independent movements or joints the manipulator possesses.(Shah et al., 2013)

For a serial manipulator, the formula to calculate the DOF is given by:

DOF = N - (J + 1)

where:

- DOF is the Degrees of Freedom of the serial manipulator,

- N is the total number of links in the manipulator (including the end-effector or tool),

- J is the total number of closed loops in the manipulator's kinematic chain.

The term (J + 1) accounts for the constraint introduced by the closed kinematic loops. In a serial manipulator, each joint introduces one degree of freedom (rotation or translation) except for the last joint (end-effector) which has one less DOF as it is restricted to follow the motion of the last link.

It's important to note that this formula assumes that the manipulator is in its fully constrained state, i.e., all joints are actuated and controllable. In some cases, manipulators may have passive or redundant degrees of freedom, which are not powered or controlled. In such instances, the DOF calculation may need to be adjusted accordingly.(Fang & Tsai, 2003)

Forward kinematics is achieved by considering the joint angles and link lengths to sequentially determine the pose of each joint, eventually yielding the overall pose of the end-effector. In essence, forward kinematics establishes a mathematical relationship that enables us to map joint

configurations to the position and orientation of the robot's working end, serving as a crucial component in robot control and path planningDH convention

The Denavit-Hartenberg (DH) parameters convention is a systematic method used in robotics to describe the geometry and kinematics of robotic manipulators. Four parameters are assigned to each joint-link connection: the link length (a), the link twist (α), the link offset (d), and the joint angle (θ). These parameters establish a standard reference frame for each joint, allowing for consistent and efficient forward and inverse kinematics calculations. The DH convention simplifies the representation of complex robot structures and enables the construction of transformation matrices that describe the relative pose of successive joints in a manipulator, facilitating precise control and motion planning.

The rationale underlying the Denavit-Hartenberg (DH) parameters lies in the idea that these four parameters [the link length (a), link twist (α), link offset (d), and joint angle (θ)] represent the essential types of motion needed to align one axis with the previous one in a robotic manipulator. By systematically defining these parameters the DH convention establishes a standard framework that allows for the precise description of relative motion between adjacent links.

2.4. Inverse kinematics

The necessity of inverse kinematics in robotic manipulators lies in its role of calculating the joint angles or configurations required to attain a desired end-effector position and orientation, It is crucial for robot control and precise motion planning.

Inverse kinematics involves using the desired end effector position and pose to calculate the joint angles that will lead to the positions, It is difficult to solve due to its inherent complexity arising from the non-linear and often highly coupled relationships between joint angles and the end-effector's position and orientation. Inverse kinematics becomes particularly challenging for robots with complex geometries and multiple degrees of freedom, as solving for the required joint angles involves solving non-linear equations with multiple variables, which can have multiple solutions or even be unsolvable in certain configurations.

2.5. Analytical Solutions

Analytical solutions of the inverse kinematics of robot arms involve the use of geometric principles to get an expression or a series of expressions that maps the end effector position to the Robot parameters (A. Raheem et al., 2016)(joint variables).

Advantages:

- Speed and efficiency: Analytical methods can be faster and more efficient than numerical methods, especially for simple robotic arms. This is because analytical methods do not require iterative calculations, which can be time-consuming.
- Range of applicability: Analytical methods can be used to solve a wider range of IK problems than numerical methods. This is because analytical methods are based on the underlying geometry of the robotic arm, which can be used to derive closed-form solutions for the inverse kinematics problem.
- Robustness to errors: Analytical methods can be more robust to errors in the input data than numerical methods. This is because analytical methods do not rely on iterative calculations, which can be sensitive to errors.

Disadvantages:

- Limited applicability: Analytical methods may not be able to solve all IK problems. This is because analytical methods are based on the underlying geometry of the robotic arm, and some robotic arms may have geometries that cannot be represented by analytical methods.
- Complexity: Analytical methods can be more complex to implement and use than numerical methods. This is because analytical methods require a deeper understanding of the underlying geometry of the robotic arm.
- Accuracy: Analytical methods may not be as accurate as numerical methods. This is because analytical methods are based on simplified models of the robotic arm, and these models may not be accurate enough for some applications.

Although computationally tasking Analytical solutions of arms may exist such as
But they are outside the scope of this report

2.6. Neural Networks

Inverse kinematics (IK) is a challenging problem in robotics, involving determining the joint angles of a robotic arm to position the end-effector at a desired location. Traditional methods like analytical and numerical approaches can be computationally expensive and sensitive to errors in input data, making real-time solutions difficult. However, machine learning methods such as neural networks (NNs) present a promising alternative for IK problem-solving.(Tarkhov et al., 2023)

ML algorithms can be trained on known IK solutions, enabling them to efficiently predict joint angles for new target positions in real-time applications. Moreover, ML's robustness to input data errors allows for accurate results even with imperfect data. The flexibility of ML algorithms also permits them to be adapted to various robotic arms and applications, as they can learn from diverse datasets.(A. Raheem et al., 2016)

Advantages of ML and NNs in IK include improved efficiency, making them suitable for applications requiring rapid responses to environmental changes. Their robustness enhances reliability, overcoming limitations of traditional methods. ML's ability to adapt to various IK problems further broadens their potential applications.

The increasing popularity of ML and NNs for IK in robotics is evident in successful applications, such as controlling robotic arms in manufacturing, leading to heightened productivity and precision. In surgical settings, ML algorithms enable more accurate and less invasive procedures. Moreover, in search and rescue missions, ML-controlled robotic arms have played a crucial role in saving lives during disaster situations.

As ML and NN technologies continue to advance, their significance in IK problem-solving in robotics is expected to grow. This development will expand the capabilities of robotic arms, leading to their application in a wider range of fields and enhancing the efficiency and reliability of robotics as a whole

2.7. Similar work done in the past

(Šegota et al., 2021) utilized a Feed Forward Multilayer Perceptron to calculate the inverse kinematics of robotic manipulators, the model was trained using Homogenous transformation matrices gotten with the aid of the Denavit-Hartenberg Convention. Multiple models, with varying combinations of hyperparameters (), were built and trained to determine the optimum combination of parameters,

(Kamlesh Shah et al., 2020) employed the use of MATLAB to build and train a Deep Artificial Neural Network to solve the inverse kinematics of a 5 Axis (or 5dof) serial Robot, with rotary joints, (an articulated robot arm) The data set for training the model was generated using the forward kinematics transformation matrix of the robot arm, an experimental model of the robot arm was built to validate the solutions generated by the DANN model, end effector positions were measured in real time with a smart camera system and image processing software. The experiment recorded a deviation of ±0.1 rads.

(Ibarra-Pérez et al., 2022) uses a dataset optimization technique based on the Taguchi optimization technique(a technique that…) on the dataset used to train an Artificial Neural Network on the inverse kinematics of a 6 DoF robotic manipulator. The model reached a high percentage of accuracy in prediction, maintaining a margin of error of less than 5%

(Lu et al., 2022) proposes a strategy called “joint space segmentation” in building and training a Multilayer Perceptron (MLP) for the inverse kinematics of 6Dof robot arms

They use an architecture where two models are trained simultaneously, one model is responsible for 5 outputs (representing 5 joint angles) and the other model is responsible or the remaining 1 output (representing the last joint angle), The method showed higher accuracy compared to other models

(Tagliani et al., 2022) uses scaled conjugate as a method of training an ANN for the Inverse kinematics of Robotic manipulators, the dataset used to train the model was generated using the Denavit-Hartenberg Parameters conventions, the experiment was validated with the simulation of

a 6dof robot (IRB140) with the ABB Robot studio software and experimentally with a 6 DOF robot (FANUCLRMate200iC) The project recorded a mean absolute error of 0.370-0.699mm

(Castellani et al, 2008) tests the effectiveness of the Bees Algorithm (a parameter optimization algorithm motivated by the foraging patterns of Honey Bees) in training Multilayer Perceptron Neural Networks to solve the inverse kinematics of an Articulated Robotic Arm. The experiment was compared with back propagation algorithm and an evolutionary algorithm. The results prove the remarkable robustness of the Bees Algorithm, which consistently trained the neural networks to model the kinematics data with very high accuracy

Chapter 3

Methodology

3.1. Kinematic Analysis of Robots using The Denavit-Hartenberg (DH) convention

The Denavit-Hartenberg (DH) convention is a useful method for calculating the forward kinematics of a robotic arm. It standardizes the representation of a robotic arm's geometry, simplifying forward kinematics calculations. Understanding forward kinematics is fundamental in robotic arm design, control, and programming, enabling engineers to grasp arm movement and optimize its performance effectively. Rather than attaching reference frames to each link in an arbitrary fashion, in the Denavit–Hartenberg convention a set of rules for assigning link frames is observed

The DH convention defines four parameters for each link in a robot:

1. The link twist αi−1 is the angle from ˆzi−1 to ˆzi, measured about xˆi−1. •
2. The link offset di is the distance from the intersection of xˆi−1 and ˆzi to the origin of the link-i frame (the positive direction is defined to be along the ˆzi-axis). •
3. The joint angle φi is the angle from xˆi−1 to xˆi, measured about the ˆzi-axis.**
4. The length of the mutually perpendicular line, denoted by the scalar ai−1, is called the link length of link i−1.

The DH parameters are usually listed in a table, with one row for each link in the robot. The table is then used to construct a transformation matrix for each link, which can be used to calculate the position and orientation of the end-effector of the robot.

consider a DH table for a 6-DOF serial manipulator:

Table 3-1 example of DH parameter table

Link	Link Length (a)	Link Twist(α)	Link Offset (d)	Joint Angle (θ)
1	a1	α1°	d1	θ1
2	a2	α2°	d2	θ2
3	a3	α3°	d3	θ3
4	a4	α4°	d4	θ4
5	a5	α5°	d5	θ5
6	a6	α6°	d6	θ6

The DH table specifies the location and orientation of the coordinate frames attached to the links of the robot. The first column lists the link number, the second column lists the distance between the *z*-axes of the two links, the third column lists the angle between the *z*-axes of the two links, the fourth column lists the distance between the *x*-axes of the two links, and the fifth column lists the angle between the *x*-axes of the two links.

3.2. Materials

The Robot arms considered for this project and their Denavit-Hartenberg tables are presented in the following:

3.2.1. ABB IRB 1600

The first arm considered is the ABB IRB 1600. It is an industrial robot arm designed and manufactured by ABB Robotics, one of the world's leading robotics companies.

The ABB IRB 1600 is a 6-degree-of-freedom (6-DOF) industrial robot arm,i.e it has six rotational joints that allow it to move in multiple directions. It is designed for a wide range of manufacturing and automation applications, including welding, material handling, assembly, and more.It has compact design, high precision, and flexibility, making it suitable for various industries and production environments.

A picture of the arm can be shown below.

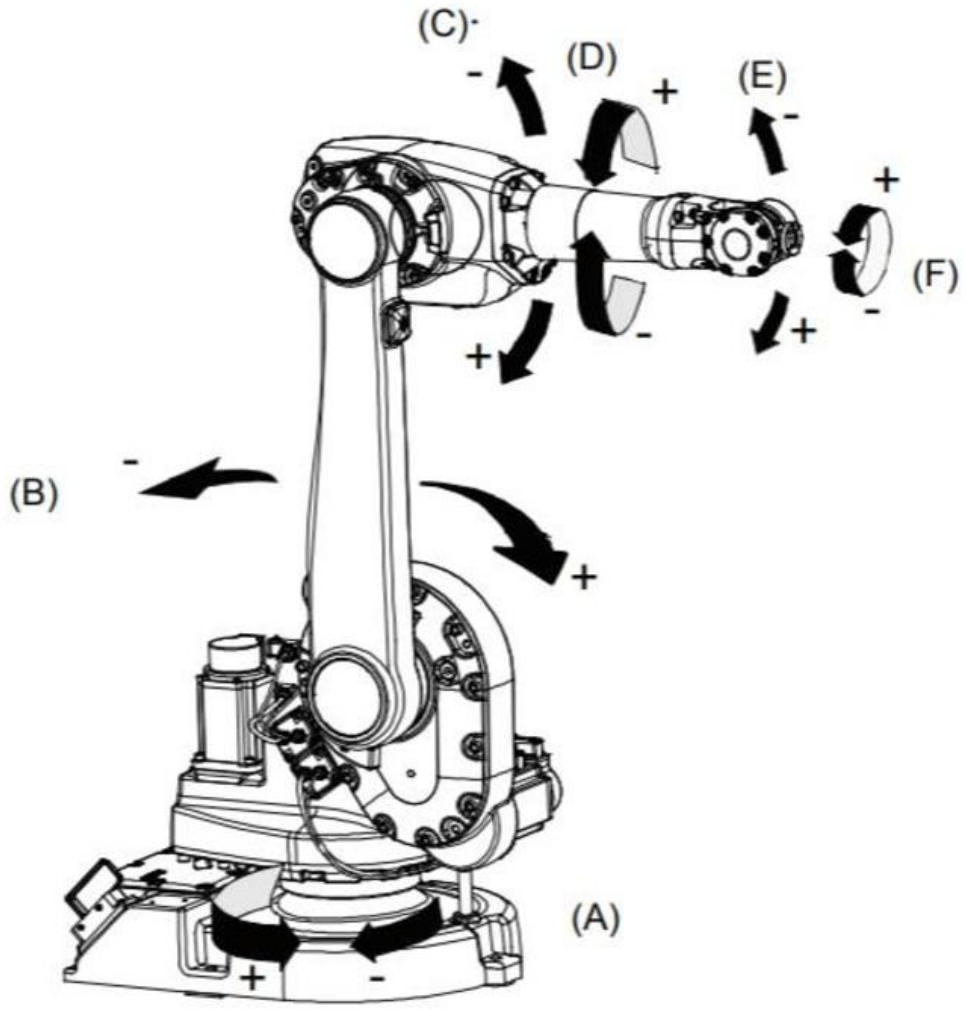

Figure 3-1 The ABB IRB 1600

Table 3-2 DH table for The ABB IRB 1600

s/n	**Θ**	**α**	**r**	**d**
1	**Θ1**	**90**	**0.3**	**o.649**
2	**Θ2**	**0**	**1.2**	**0**
3	**Θ3**	**90**	**0.250**	**0**
4	**Θ4**	**90**	**0**	**1.320**
5	**Θ5**	**90**	**0**	**0**

6	**Θ6**		**0**	**0.290**

3.2.2. THE FANUC LR MATE 200IC

The FANUC LR Mate 200iC is an industrial robotic arm developed by FANUC Corporation, a Japanese robotics company. It is known for its compact size, high precision, and versatility in a wide range of industrial applications. The LR Mate 200iD is designed to perform various tasks, including material handling, assembly, and machine tending.

A picture of the arm can be shown below.

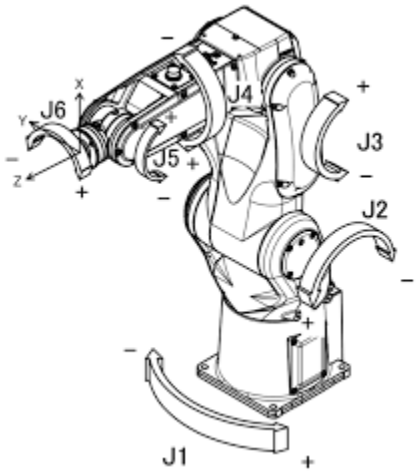

Figure 3-2 The FANUC LR Mate 200iC

Table 3-3 DH table for The FANUC LR Mate 200iC

s/n	**Θ**	**α**	**r**	**d**
1	**Θ1**	**-90**	**75**	**330**
2	**Θ2**	**180**	**300**	**0**
3	**Θ3**	**90**	**-75**	**0**
4	**Θ4**	**-90**	**0**	**-320**

5	**Ɵ5**	**90**	**0**	**0**
6	**Ɵ6**	**180**	**0**	**-80**

3.2.3. THE UNIVERSAL ROBOT UR10

The Universal Robots UR10 is a collaborative robotic arm designed and manufactured by Universal Robots, a Danish robotics company. The UR10 is part of Universal Robots' line of collaborative robots, also known as COBOTS. These robots are intended to work safely alongside humans in various industrial and commercial applications, enhancing productivity and flexibility in manufacturing environments.

A picture of the arm is shown below;

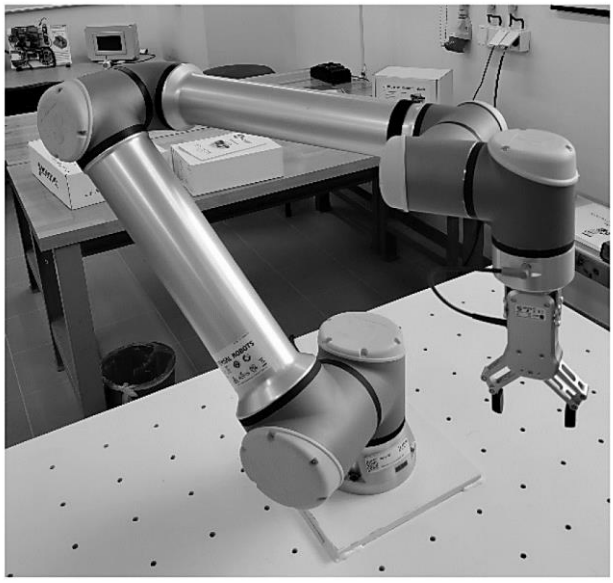

Figure 3-3 The Universal robot Ur10

Table 3-4 DH table for The Universal Robot Ur10

s/n	**Ɵ**	**α**	**r**	**d**

1	**Θ1**	**90**	**0**	**127**
2	**Θ2**	**0**	**612**	**0**
3	**Θ3**	**0**	**572**	**0**
4	**Θ4**	**90**	**0**	**164**
5	**Θ5**	**-90**	**0**	**115**
6	**Θ6**	**0**	**0**	**92**

3.2.4. The Kuka KR Agilus industrial robot.

The KUKA KR AGILUS is a series of compact industrial robot arms manufactured by KUKA Robotics, a German robotics company known for its innovative automation solutions. They were initially designed to be fast, precise, and versatile, making it well-suited for various applications in industries like automotive, electronics, plastics, and more. Despite its compact size, the KR AGILUS robot offers high payload capacities and exceptional performance, making it ideal for tasks that require agility and accuracy.

A picture of the arm can be shown below;

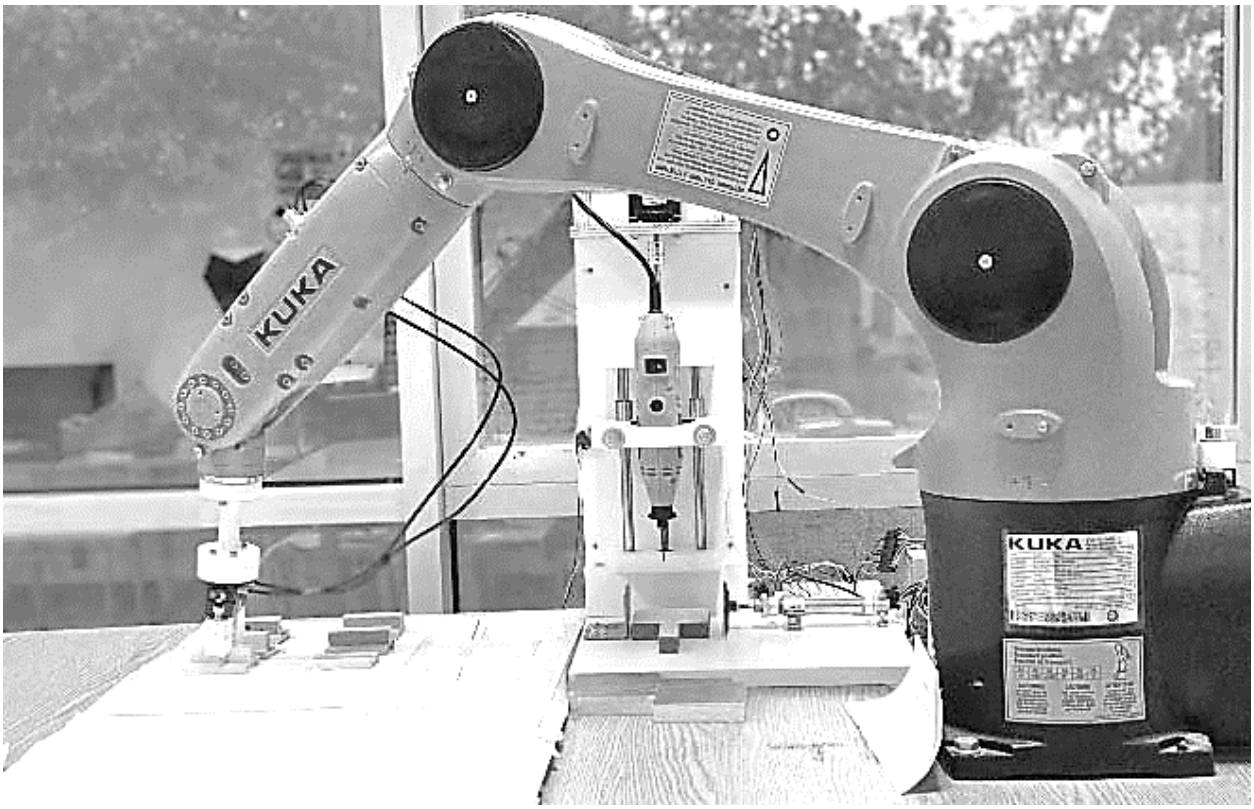

Figure 3-4 The KUKA KR AGILUS

Table 3-5 DH table for The KUKA KR AGILUS

s/n	Θ	α	r	d
1	Θ1	90	180	400
2	Θ2	180	600	135
3	Θ3	-90	120	135
4	Θ4	90	0	620
5	Θ5	-90	0	0
6	Θ6	0	0	115

3.2.5. The Yasakawa Motoman GP series.

The Yasakawa Motoman GP series is a family of industrial robot arms that are designed and manufactured by Yaskawa Electric Corporation's Motoman Robotics division. They can used for various automation applications in industries such as automotive, electronics, metalworking, and more. This is because they are mostly known for its versatility, high performance, and reliability, making it a popular choice for industrial automation tasks.

This analysis focuses on the Yasakawa Motoman GP7 model of the GP series. A picture of the arm can be shown below.

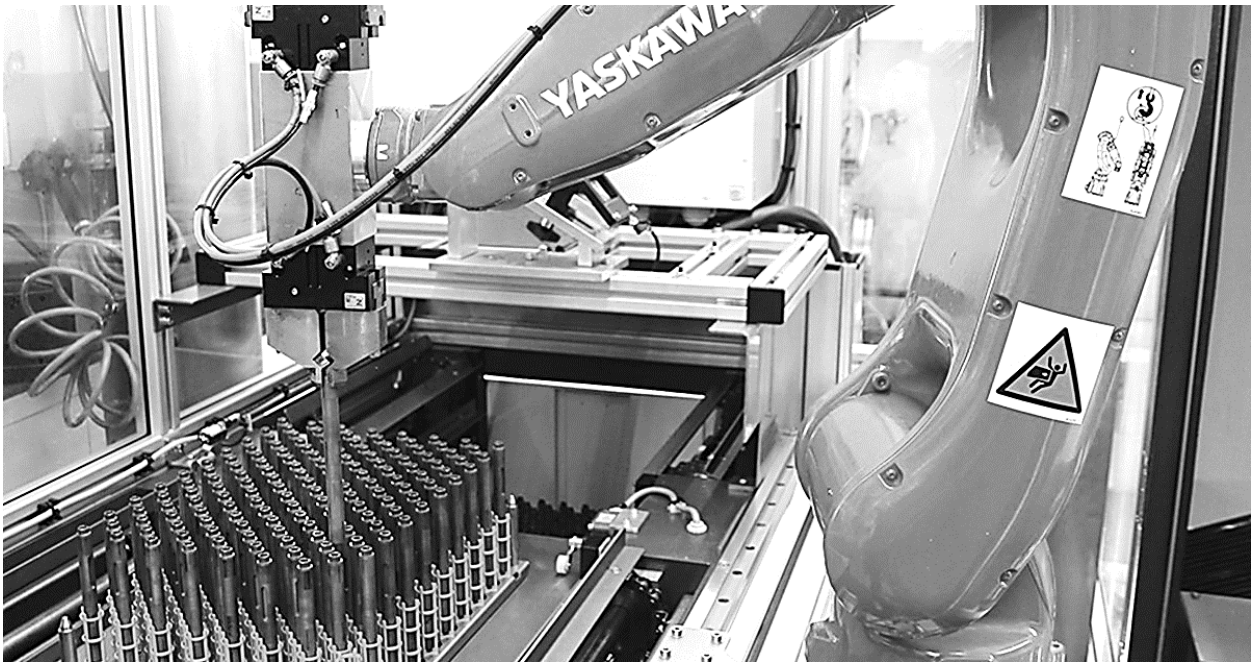

Figure 3-5 The Yasakawa Motoman GP

Table 3-6 DH table for The Yasakawa Motoman GP

s/n	Θ	α	r	d
1	Θ1	0	40	330
2	Θ2	0	445	0
3	Θ3	0	40	0

4	**Θ4**	**90**	**0**	**440**
5	**Θ5**	**-90**	**0**	**0**
6	**Θ6**	**-90**	**0**	**80**

3.2.6. 4DOF ROBOT ARM

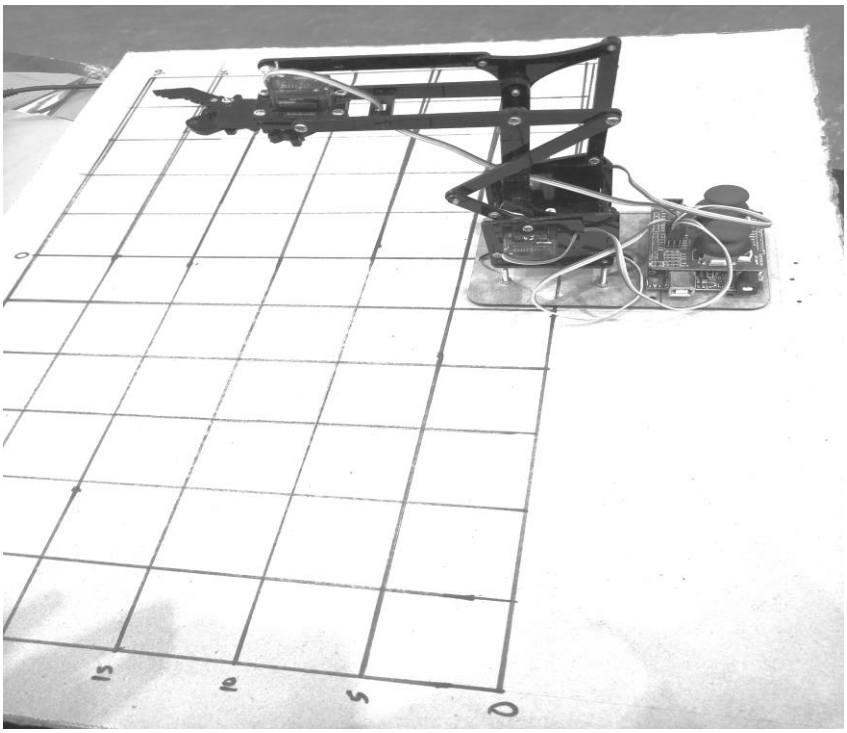

Figure 3-6 A 4 DOF Robot Arm

3.3. Neural Network Models

The model takes seven inputs which are a representation of the desired joint angles and outputs

- 6 values of the joint angles, in the case of the 6dof robots OR
- 4 values of joint angles in the case of the 4dof robot.

See appendix for code snippets

3.3.1. The NN Model for the 6DOF Manipulators

The NN Model for the 6DOF Manipulators was split in to two with each part responsible for 3 outputs only

Table 3-7 model architecture for the 6dof robotic arms (a)

layers	Input layer	Hidden layer1	Hidden layer2	Hidden layer3	Hidden layer4	Hidden layer5	Hidden layer6	Hidden layer7	Hidden layer8	Hidden layer 9	Output layer
Number of nodes	**7**	**512**	**256**	**256**	**128**	**128**	**64**	**64**	**32**	**32**	**3**
Type of activation function		**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	

Table 3-8 model architecture for the 6dof robotic arms (b)

layers	Input layer	Hidden layer1	Hidden layer2	Hidden layer3	Hidden layer4	Hidden layer5	Hidden layer6	Hidden layer7	Hidden layer8	Hidden layer 9	Output layer
Number of nodes	**7**	**512**	**256**	**256**	**128**	**128**	**64**	**64**	**32**	**32**	**3**
Type of activation function		**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	

3.3.2. The model architecture for the 4d0f model

Table 3-9 model architecture for the 4dof robot arm

layers	Input layer	Hidden layer1	Hidden layer2	Hidden layer3	Hidden layer4	Hidden layer5	Hidden layer6	Hidden layer7	Hidden layer8	Hidden layer9	Output layer
Number of nodes	**7**	**512**	**256**	**256**	**128**	**128**	**64**	**64**	**32**	**32**	**4**
Type of activation function		**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	**ReLU**	

Chapter 4

Result

The Result of all the models built are presented below;

Dataset generated for arms :- The dataset generated for all the arms(the 6 dof arm models and the 4dof live testing model) was gotten by generating random angles while strictly following the specified range limit of each joint according to the manufacturer's manual. A total of 100,000 different data entries were generated using a function to generate the dataset. For the 6dof models, a total of 6 different joint angle combinations were generated for each entry while for the 4dof model, a total of four different joint angle combinations were generated for each entry. Foward kinematics was done using this different joint angle combinations to obtain the quarternion representation and x,y,z position of the end effector for each joint angle. For the 6dof manipulators, the combination of the 6 joint angles, the quarternion representation, and the end effector position formed the total dataset (13 columns). While for the 4dof manipulator, the four joint angles, the end effector position, and the quarternion representation formed the total dataset (11 columns).
This data was then split into training and test data for the neural networks model. Each of these angles are generated in radians for the models to be trained and evaluated on. The total generated dataset was split into 80,000 for training and 20,000 for test. Twenty out of the complete 100,000 data sets generated (training and test) for all the arm models can be shown below.

4.1. Arm 1

Table 4-1 Result comparison for each predicted vs actual (first 20 values of table)

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual
2409.8541	**190.7337**	**1157.5796**	**2387.0929**	**269.3849**	**1379.6588**

-1184.5587	**2285.9261**	**140.2789**	**-1277.1028**	**2272.6844**	**46.3644**
1828.2322	**1396.2749**	**1636.1265**	**1719.9136**	**1294.9423**	**1867.5192**
-1904.5516	**1019.6171**	**-68.2975**	**-1980.5305**	**1055.0535**	**13.9894**
-280.8689	**2000.4092**	**1557.7839**	**-353.7540**	**1995.8426**	**1606.6902**
-1171.8058	**2253.8726**	**192.6211**	**-1070.1199**	**2203.8941**	**130.0086**
1480.2687	**1102.3807**	**1656.6080**	**1501.4472**	**987.8685**	**1773.2111**
915.5779	**1823.1846**	**2088.8874**	**875.1902**	**1843.0737**	**2147.1465**
-709.5761	**1082.3103**	**2998.1515**	**-648.5154**	**874.6369**	**2939.2040**
2141.3110	**276.6836**	**2106.7832**	**2055.7201**	**310.2088**	**2296.2594**
-2295.3000	**401.3176**	**72.8277**	**-2377.1927**	**356.4353**	**170.3101**
439.3049	**2072.6757**	**1817.9119**	**352.9935**	**2026.0235**	**1919.8006**
-744.8202	**632.5968**	**2872.0891**	**-439.4544**	**441.0260**	**2941.2973**
1856.3986	**1238.7889**	**-244.7906**	**1856.6838**	**1467.6098**	**-18.7268**
1115.3468	**2377.6561**	**439.3190**	**930.2274**	**2222.1807**	**345.4716**
-875.0212	**2384.3806**	**1581.4169**	**-975.5007**	**2140.3540**	**1954.8657**
1618.0011	**913.1367**	**-235.7239**	**1604.1660**	**953.7861**	**-217.1425**
1592.6160	**1255.5635**	**-0.3503**	**1457.4524**	**1346.4441**	**60.8205**
1755.5468	**99.9585**	**2095.2109**	**1647.6111**	**182.8979**	**2317.9147**

-10.6199	**2213.7995**	**278.1757**	**-11.6187**	**2237.8111**	**374.2108**

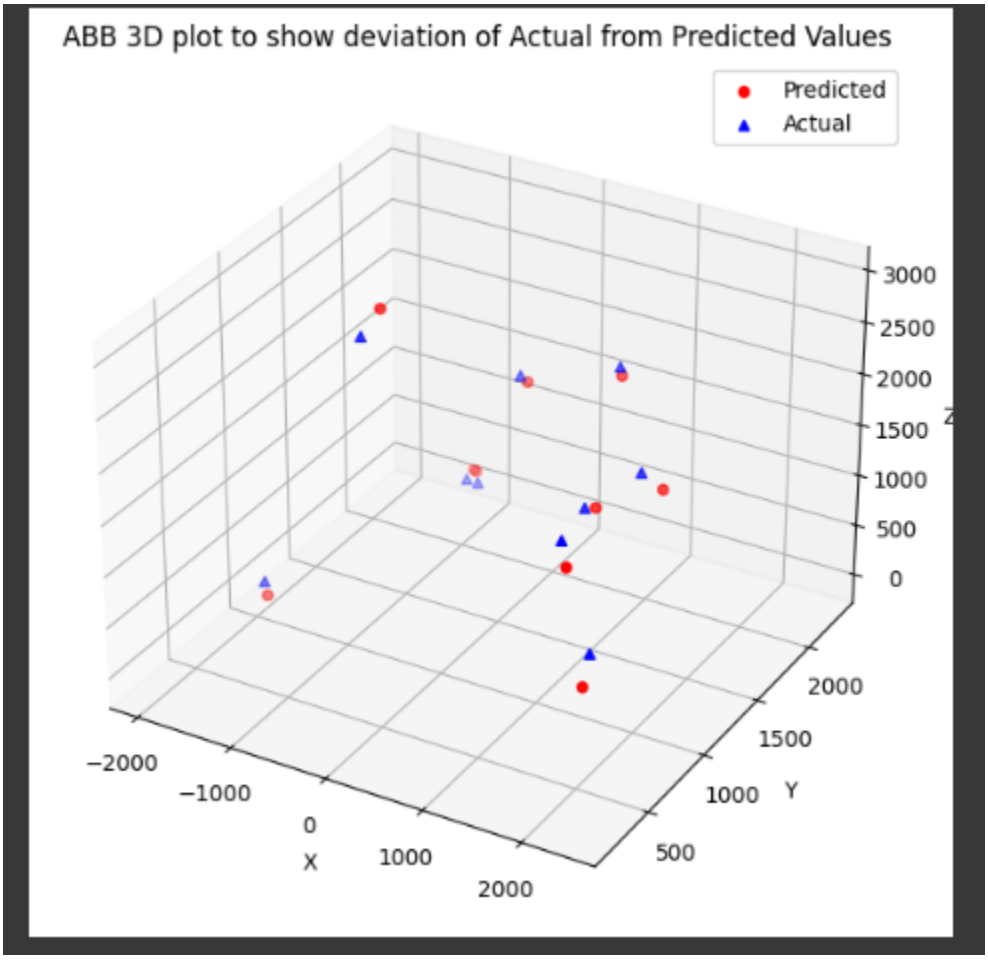

Figure 4-1 Result comparison for each predicted vs actual(graph)

4.1.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.2. Arm 2

Table 4-2 Result comparison for the fanuc arm first 20 predicted vs actual values.

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual
302.8865	**111.2329**	**456.7143**	**318.7981**	**111.6372**	**462.5276**
-62.5091	**63.7427**	**543.5907**	**-63.1501**	**59.1925**	**548.3676**
128.3358	**205.2312**	**449.7896**	**122.9626**	**196.4552**	**456.3627**
-146.5845	**88.4684**	**653.8550**	**-143.6651**	**70.1168**	**648.1994**
-141.6141	**400.3419**	**427.4375**	**-137.2669**	**400.5091**	**424.8362**
-108.7351	**189.7162**	**607.0702**	**-116.8793**	**191.1013**	**615.7218**
334.0468	**273.8032**	**312.0277**	**332.5049**	**292.3125**	**321.2323**
10.9525	**96.8377**	**467.9763**	**25.9097**	**104.3026**	**475.3968**
-176.2844	**79.8613**	**386.1423**	**-188.0324**	**85.7720**	**377.4940**
-12.6643	**78.7978**	**280.4204**	**-45.7883**	**59.6553**	**284.4882**
-155.5138	**-38.1914**	**578.1656**	**-140.7503**	**-48.4510**	**570.0675**
5.9535	**229.7219**	**490.3152**	**3.0095**	**216.8364**	**495.7398**
-128.8176	**31.5402**	**415.3602**	**-122.7231**	**12.7120**	**410.8432**
17.9192	**101.2634**	**533.7480**	**17.6708**	**100.0176**	**539.7666**
94.6406	**243.9582**	**599.1063**	**99.2731**	**254.0748**	**619.1675**

-102.9502	**35.1999**	**388.4167**	**-102.1033**	**29.0420**	**388.9069**
225.3638	**209.2584**	**659.9805**	**216.3452**	**193.4980**	**677.6174**
259.7779	**284.3523**	**642.9933**	**244.7312**	**295.5448**	**644.6179**
308.3297	**27.1843**	**445.0004**	**300.0858**	**46.4011**	**443.0746**
-99.5246	**321.2684**	**562.8790**	**-90.3326**	**353.0322**	**569.9731**

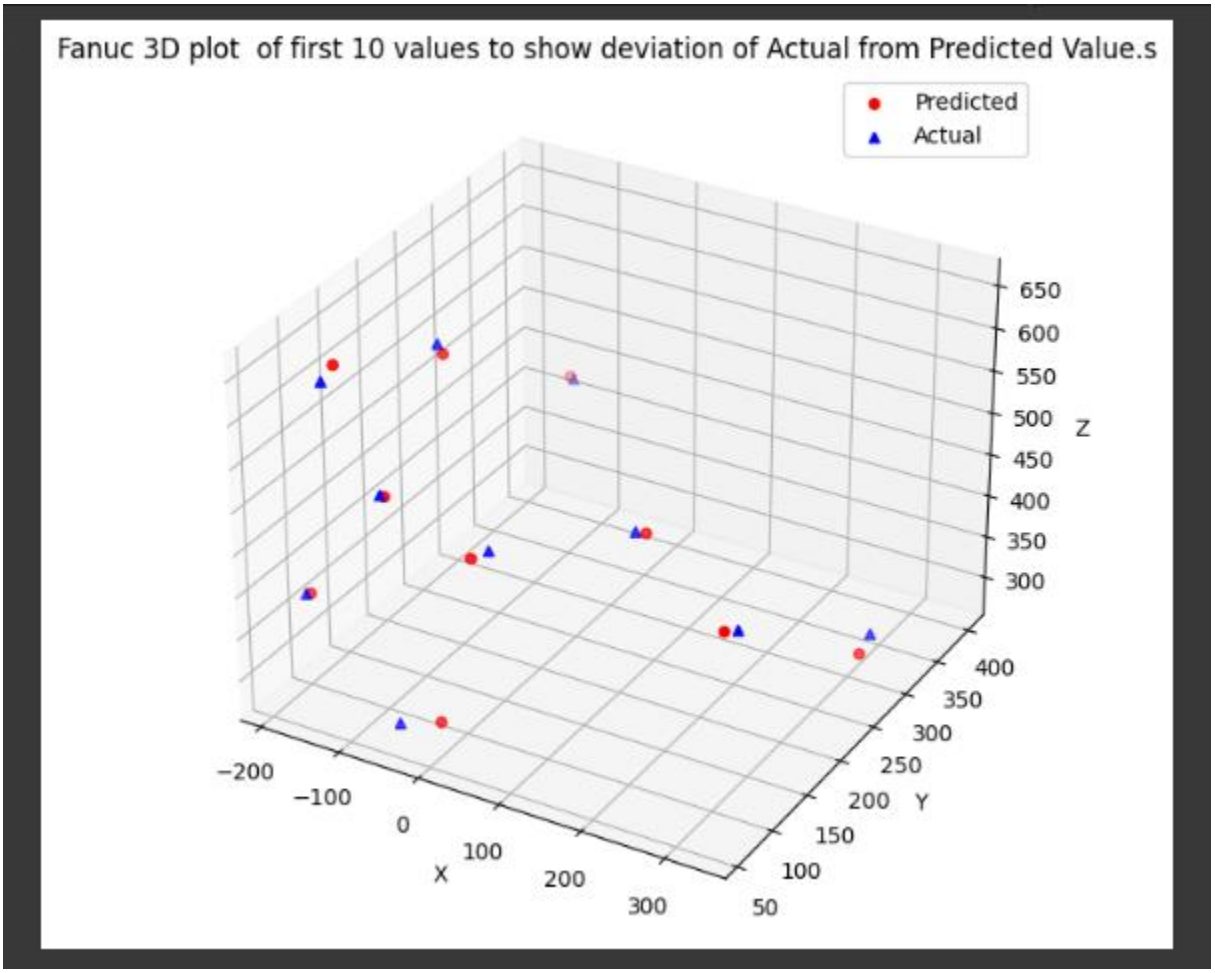

Figure 4-2 Result comparison for each predicted vs actual(graph).

4.2.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.3. Arm 3

Table 4-3 Result comparison for each predicted vs actual (table).

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual
-188.9601	**-207.6755**	**1257.8822**	**-143.7640**	**-219.1075**	**1296.8362**
-431.4633	**1002.9546**	**459.6879**	**-420.1675**	**1021.9228**	**463.4497**
-434.1977	**-622.1768**	**1071.6883**	**-407.3294**	**-644.0693**	**1101.0744**
-900.2541	**787.3260**	**578.6576**	**-921.0602**	**768.4131**	**571.3808**
410.1213	**-570.9638**	**1145.9366**	**390.1220**	**-544.7256**	**1202.6824**
-287.8428	**991.1510**	**613.7645**	**-289.6986**	**985.1771**	**668.3930**
-808.7300	**-795.4385**	**712.7552**	**-780.6692**	**-803.7963**	**814.7537**
23.6890	**-577.5500**	**1069.4068**	**23.0089**	**-586.6013**	**1105.0071**
1015.8080	**-576.5372**	**-66.5664**	**1004.5107**	**-566.0583**	**-77.9419**
-561.5195	**-330.9341**	**893.3660**	**-487.2781**	**-323.5224**	**935.8870**
-946.3534	**321.7446**	**872.5754**	**-1000.4454**	**270.2718**	**824.1568**

130.3898	**-718.7301**	**1065.1433**	**159.0338**	**-701.3426**	**1125.4518**
775.0253	**-273.0519**	**-603.7808**	**883.8818**	**-339.4422**	**-468.6864**
996.4304	**518.4072**	**676.0017**	**971.8302**	**554.7162**	**682.9887**
559.2887	**746.1535**	**817.1806**	**545.7414**	**736.4701**	**841.4485**
430.8566	**-473.5041**	**1061.2394**	**409.3032**	**-395.4083**	**1115.4109**
1140.5578	**470.3159**	**500.5583**	**1172.8548**	**477.9504**	**460.9613**
944.8362	**588.8271**	**728.2520**	**928.6147**	**616.0387**	**746.6273**
-1050.7377	**-355.9790**	**625.1597**	**-1023.8335**	**-393.6229**	**698.0759**
105.3389	**709.3994**	**1111.0325**	**120.8587**	**768.2789**	**1090.7692**

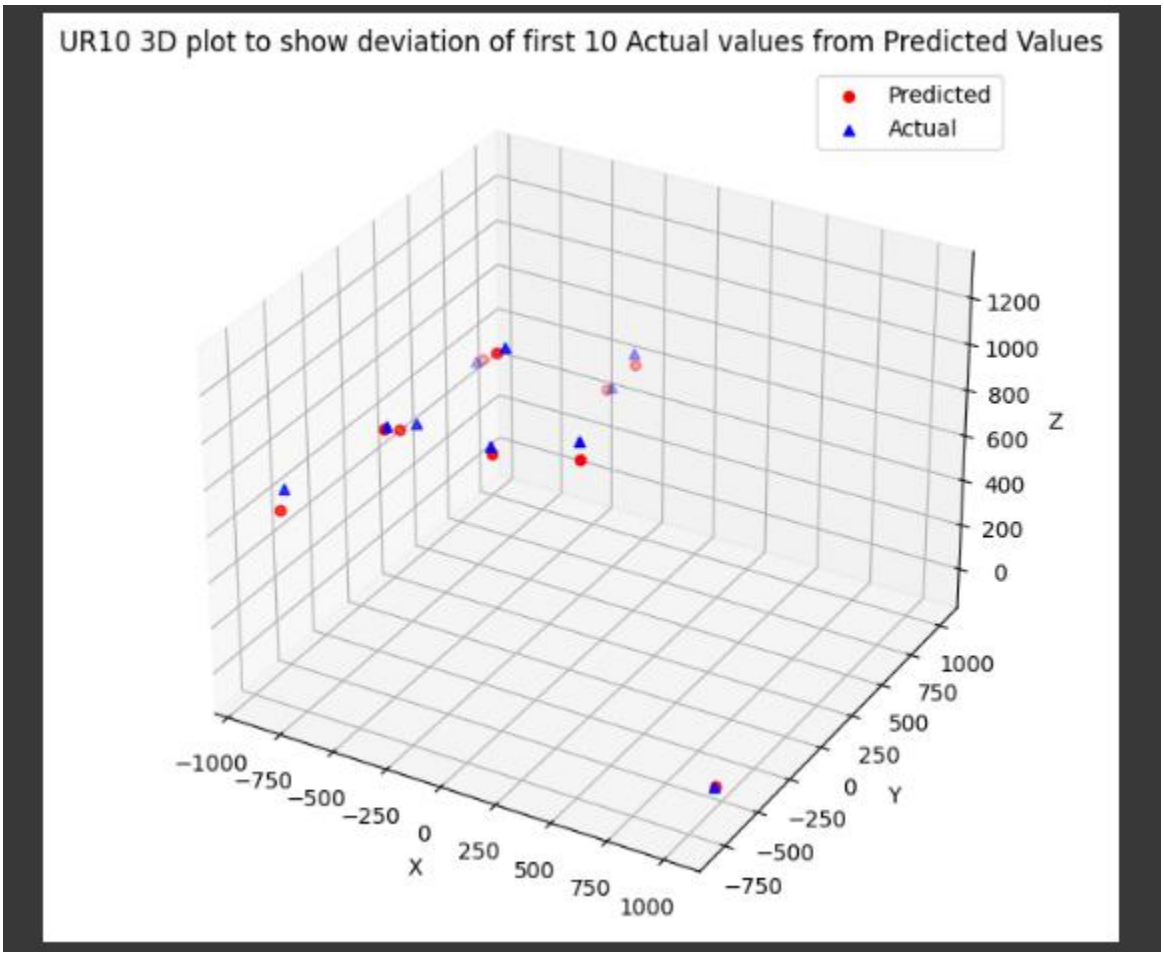

Figure 4-3 Result comparison for each predicted vs actual(graph).

4.3.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.4. Arm 4

Table 4-4 Result comparison for each predicted vs actual (table)

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual
925.4178	**322.3457**	**354.6286**	**908.1530**	**259.6541**	**327.1963**
-329.2310	**413.8270**	**-181.7480**	**-289.7602**	**396.3276**	**-147.2705**
480.1234	**532.3448**	**235.0839**	**493.7699**	**565.8419**	**248.2363**
-597.8412	**288.5849**	**-239.9229**	**-559.0185**	**284.4924**	**-270.9445**
-343.1659	**932.2661**	**477.2992**	**-306.3788**	**996.3567**	**450.3053**
-460.8177	**748.5527**	**-177.5895**	**-386.6402**	**684.3652**	**-128.3826**
771.3447	**666.7681**	**655.8230**	**797.9556**	**636.2261**	**671.7149**
219.4756	**513.3239**	**100.9062**	**184.2060**	**505.2507**	**122.6684**
-568.9542	**366.8871**	**508.4564**	**-550.9643**	**347.4680**	**519.8716**
142.1805	**140.7974**	**316.3166**	**157.4335**	**141.1511**	**300.7163**
-636.5676	**-120.1371**	**-68.8848**	**-597.3166**	**-47.5429**	**-139.0785**
50.6859	**694.1969**	**189.5256**	**59.5847**	**728.0925**	**160.1787**
-453.8705	**205.9908**	**377.1029**	**-455.7606**	**211.1686**	**374.2455**
247.2144	**339.9363**	**-157.9272**	**268.9929**	**354.5296**	**-154.3256**
264.7360	**813.3901**	**-115.2256**	**322.0790**	**807.0565**	**-90.2988**

-288.6690	**350.6774**	**205.7233**	**-305.6720**	**342.6642**	**196.9839**
611.4609	**524.6727**	**-233.1362**	**702.6635**	**533.3352**	**-237.9968**
635.9691	**735.8211**	**-87.6838**	**695.5513**	**785.1782**	**-88.9020**
807.3777	**161.4211**	**370.0009**	**839.7944**	**120.9949**	**360.4083**
-202.3010	**990.4088**	**55.5553**	**-187.6301**	**996.8218**	**72.2784**

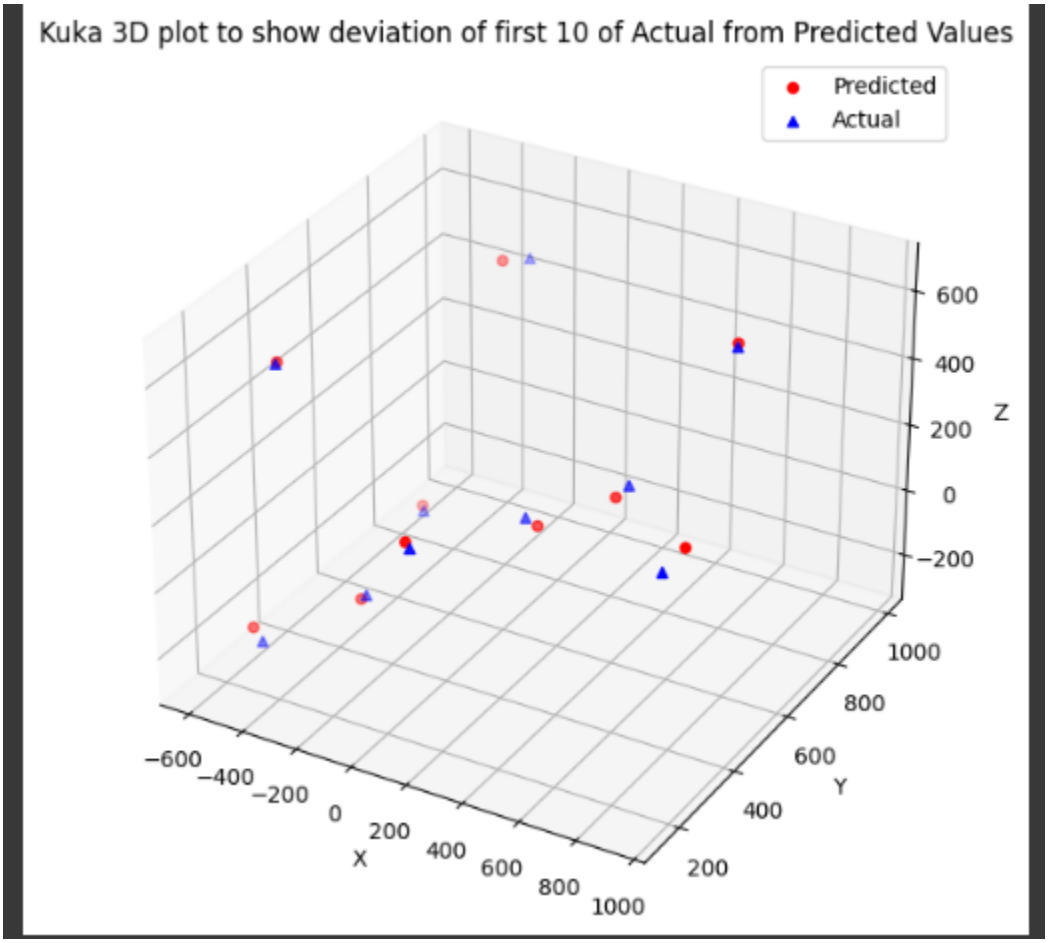

Figure 4-4 Result comparison for each predicted vs actual(graph)

4.4.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.5. Arm 5

Table 4-5 Result comparison for each predicted vs actual (table)

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual
298.1205	**396.3232**	**697.5022**	**312.1497**	**377.0167**	**696.8390**
229.6466	**424.5335**	**714.1481**	**217.7731**	**432.5360**	**709.9100**
197.9537	**486.6595**	**723.0829**	**193.5747**	**488.1848**	**727.6765**
57.1475	**507.4120**	**848.5765**	**71.6975**	**512.7340**	**849.2028**
-54.4360	**470.8718**	**786.0307**	**-86.9176**	**470.9041**	**801.8525**
181.9480	**431.2457**	**725.7670**	**160.4170**	**433.4368**	**727.0453**
149.6694	**515.1637**	**768.2568**	**119.3350**	**510.4969**	**770.7723**
180.1302	**461.3538**	**813.2920**	**129.2242**	**471.3350**	**815.4503**
-419.1582	**226.2533**	**698.6273**	**-419.6445**	**222.5643**	**698.0190**
319.1106	**414.7111**	**705.8304**	**352.5034**	**387.6142**	**710.1508**
1.7865	**525.0684**	**727.8872**	**-34.0545**	**519.1155**	**737.1595**

51.9032	**521.2339**	**823.9893**	**25.0531**	**517.1021**	**829.5405**
-488.7267	**193.3726**	**763.8372**	**-485.4926**	**200.1909**	**755.6182**
507.8893	**218.6235**	**739.6401**	**517.1032**	**195.0192**	**741.8377**
310.5755	**344.1245**	**771.8792**	**286.4044**	**341.5282**	**768.9430**
-107.0624	**496.0582**	**693.8727**	**-111.3466**	**495.0346**	**693.3472**
515.8535	**142.6576**	**791.3405**	**501.6887**	**140.9579**	**798.5898**
423.8982	**246.2733**	**825.2599**	**406.9852**	**266.9146**	**819.2256**
183.7126	**453.4012**	**846.7043**	**182.5207**	**452.8649**	**847.4857**
186.6761	**445.9553**	**722.0487**	**178.6495**	**445.8303**	**730.1937**

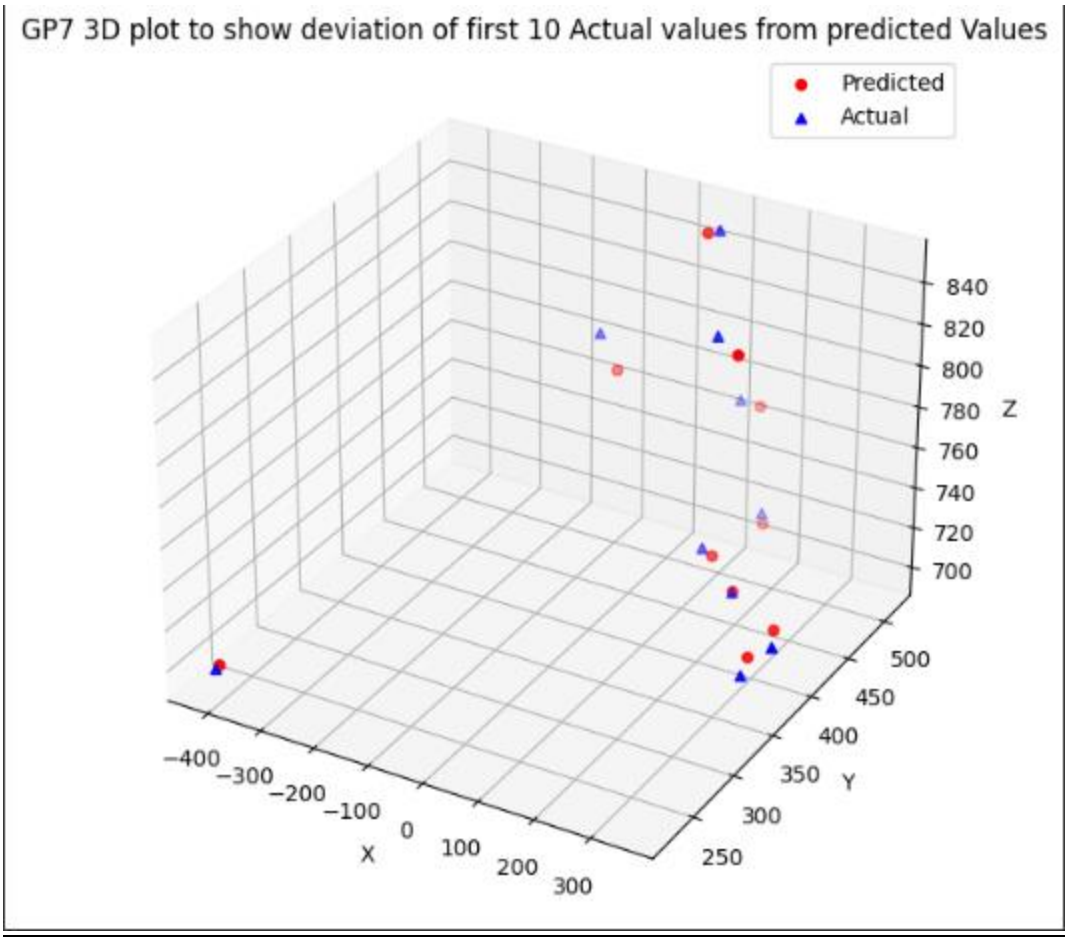

Figure 4-5 Result comparison for each predicted vs actual(graph)

4.5.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.6. Arm 6

Table 4-6 Result comparison for each predicted vs actual (table)

x_predicted	y_predicted	z_predicted	x_actual	y_actual	z_actual

8.7748	**-13.6719**	**-0.2667**	**8.5066**	**-13.8398**	**-0.2145**
10.2711	**-13.0384**	**2.2972**	**10.0157**	**-13.2776**	**2.2848**
16.2447	**0.2331**	**0.3478**	**16.2441**	**-0.3481**	**0.4555**
1.2836	**16.6613**	**2.2205**	**1.2308**	**16.6824**	**2.2006**
16.4008	**-4.4554**	**1.6149**	**16.4267**	**-4.3539**	**1.6191**
16.9819	**-1.4384**	**-1.4406**	**16.9642**	**-1.5916**	**-1.4566**
10.1415	**13.2748**	**2.2262**	**10.2314**	**13.2047**	**2.2270**
9.5867	**-13.1288**	**0.8233**	**9.2112**	**-13.3984**	**0.9091**
17.0607	**2.2216**	**-0.1755**	**17.1292**	**1.6169**	**-0.1491**
9.9133	**-12.9556**	**1.6701**	**9.5485**	**-13.2365**	**1.7302**
8.1314	**14.7857**	**1.9485**	**8.2655**	**14.7276**	**1.9156**
9.5289	**13.1736**	**-0.8914**	**9.5316**	**13.1681**	**-0.8040**
14.4492	**8.5611**	**2.1033**	**14.6416**	**8.2567**	**2.0787**
6.7601	**-15.1874**	**2.2884**	**6.5605**	**-15.2825**	**2.2850**
9.5086	**-14.1228**	**1.5076**	**9.3480**	**-14.2271**	**1.5151**
6.0344	**15.1265**	**1.3912**	**6.2319**	**15.0467**	**1.3979**
15.2581	**6.7127**	**2.2598**	**15.5092**	**6.1035**	**2.2617**
15.2544	**5.5872**	**-0.2543**	**15.2678**	**5.5494**	**-0.2002**

13.8570	**8.8781**	**2.2197**	**14.0172**	**8.6393**	**2.2323**
16.2796	**4.2510**	**-2.0489**	**16.5650**	**2.7900**	**-2.0976**

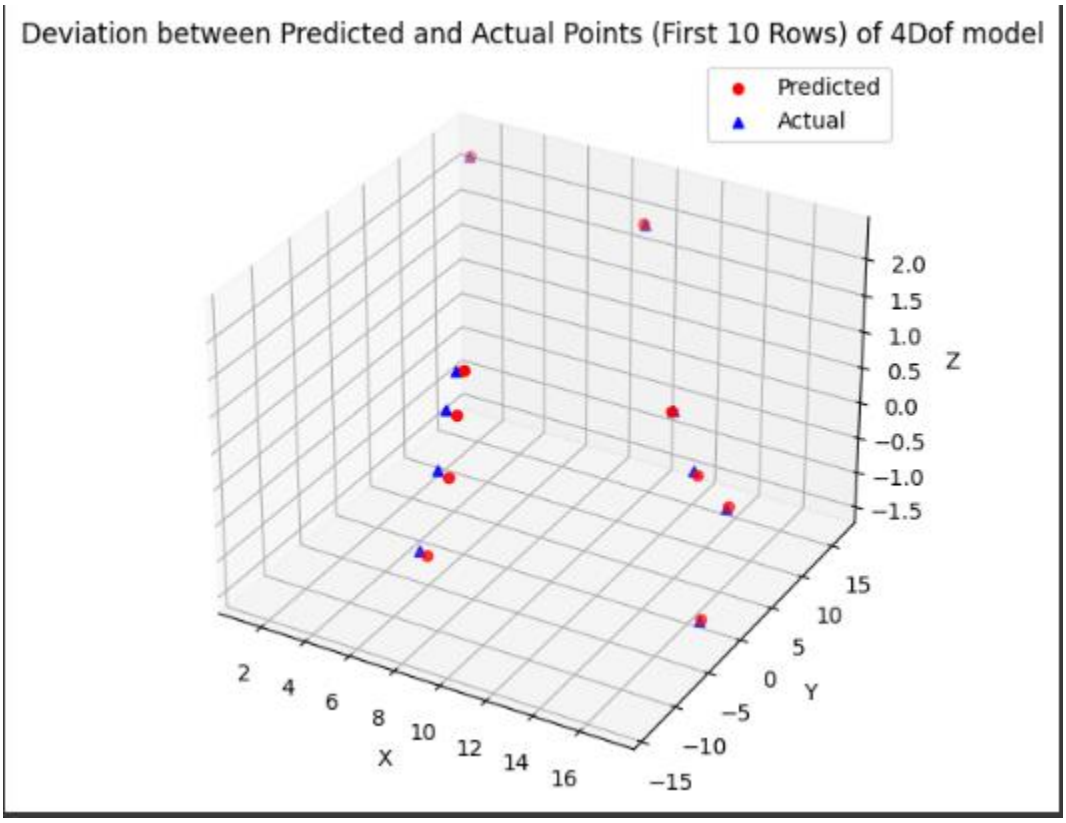

Figure 4-6 Result comparison for arm 6 predicted vs actual(graph)

4.6.1. Discussion

Form the tables and the graphs of the result above it was observed that the Neural Network model could predict the desired joint angles to a significant degree of accuracy, the worst performing models having errors within 1 to 2 decimal places

4.7. Model Performance

Table 4-7 Model Performance

Arms	Model name	Mean squared error	R squared value

1	ABB IRB 1600	0.0819	0.8402
2	Fanuc LR Mate 200iD	0.0209	0.9490
3	Universal Robots UR10	0.0240	0.9412
4	KUKA KR AGILUS	0.0896	0.7813
5	Yasakawa Motorman GP series	0.0860	0.7908
6	Model no.5(4Dof model)	0.2045	0.7288

With performance like these, models can be applied in fields requiring moderate precision.

4.8. Problems faced

- Deployment of the neural networks code from python environment to Arduino environment is proving to be very difficult and the angles generated by the neural networks has to be passed in manually on the Arduino side.
- Normal neural networks approach proved to be less effective so more advanced techniques needed to be used especially for the 6 degree of freedom robot arms.

Chapter 5

Conclusion and Recommendations

5.1. Conclusions

The effectiveness of neural networks approach in solving the inverse kinematics problem of a neural manipulator largely depends on the complexity and problem type it is being applied in. Neural networks work better for 6dof manipulators and below.

Neural networks are effective enough in solving the inverse kinematics problem of robot manipulators although more research and improvements have to be made to make them deployable and used more commonly and effectively in the industry.

5.2. Recommendations

Due to limited time, data from live experiment could not be presented in this report, this could be addressed in future research

This project focused on the use of a relatively simple Neural Network model, other more complex models exist and there use in inverse kinematic problems should be explored

References

A. Raheem, F., R. Kareem, A., & J. Humaidi, A. (2016). Inverse Kinematics Solution of Robot Manipulator End- Effector Position Using Multi-Neural Networks. *Engineering and Technology Journal*, *34*(7), 1360–1368. https://doi.org/10.30684/etj.34.7A.9

Fang, Y., & Tsai, L.-W. (2003). Inverse Velocity and Singularity Analysis of Low-DOF Serial Manipulators. *Journal of Robotic Systems*, *20*(4), 177–188. https://doi.org/10.1002/rob.10079

Filipescu, A., Mincă, E., Filipescu, A., & Coandă, H.-G. (2020). Manufacturing Technology on a Mechatronics Line Assisted by Autonomous Robotic Systems, Robotic Manipulators and Visual Servoing Systems. *Actuators*, *9*(4), 127. https://doi.org/10.3390/act9040127

Hu, B. (2014). Complete kinematics of a serial–parallel manipulator formed by two Tricept parallel manipulators connected in serials. *Nonlinear Dynamics*, *78*(4), 2685–2698. https://doi.org/10.1007/s11071-014-1618-4

Ibarra-Pérez, T., Ortiz-Rodríguez, J. M., Olivera-Domingo, F., Guerrero-Osuna, H. A., Gamboa-Rosales, H., & Martínez-Blanco, Ma. D. 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

IEEE INDIN 2008 6th International Conference on Industrial Informatics, July 13-16, 2008, Daejeon, Korea. (2008). IEEE.

Kamlesh Shah, S., Mishra, R., & Samprit Ray, L. (2020). Solution and validation of inverse kinematics using Deep Artificial neural network. *Materials Today: Proceedings*, *26*, 1250–1254. https://doi.org/10.1016/j.matpr.2020.02.250

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

Mohammad Amin, F. (2023). Analysis of Forward Kinematics of 2R Robotic Arm. *Engineering and Applied Sciences*. https://doi.org/10.11648/j.eas.20230802.11

Šegota, S. B., Anđelić, N., Mrzljak, V., Lorencin, I., Kuric, I., & Car, Z. (2021). Utilization of multilayer perceptron for determining the inverse kinematics of an industrial robotic manipulator. *International Journal of Advanced Robotic Systems*, *18*(4), 172988142092528. https://doi.org/10.1177/1729881420925283

Shah, J. A., Rattan, S. S., & Nakra, B. C. (2013). End-Effector Position Analysis Using Forward Kinematics For 5 Dof Pravak Robot Arm. *IAES International Journal of Robotics and Automation (IJRA)*, *2*(3), 112–116. https://doi.org/10.11591/ijra.v2i3.2015

Tagliani, F. L., Pellegrini, N., & Aggogeri, F. (2022). Machine Learning Sequential Methodology for Robot Inverse Kinematic Modelling. *Applied Sciences*, *12*(19), 9417. https://doi.org/10.3390/app12199417

Tarkhov, D., Lazovskaya, T., & Malykhina, G. (2023). Constructing Physics-Informed Neural Networks with Architecture Based on Analytical Modification of Numerical Methods by Solving the Problem of Modelling Processes in a Chemical Reactor. *Sensors*, *23*(2), 663. https://doi.org/10.3390/s23020663

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

APPENDIX

Code Snippet For 6dof Model

```python
# Import the necessary libraries
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping

# Load the CSV data
data = pd.read_csv('/content/drive/MyDrive/joint_positions_abb (3).csv')

# Split features (input) and labels (output)
features = data.iloc[:, -7:].values  # Last 7 columns are features
labels = data.iloc[:, :-7].values    # First 6 columns are labels

# Split data into training, validation, and test sets
x_train, x_test, y_train, y_test = train_test_split(
    features, labels, test_size=0.2, random_state=42)

# Define the first TensorFlow model for the first four outputs
abb_model1 = tf.keras.Sequential([
    tf.keras.layers.Dense(512, activation='relu', input_shape=(7,)),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3)  # Output the first four values
])
```

```python
# Define the second TensorFlow model for the last three outputs
abb_model2 = tf.keras.Sequential([
    tf.keras.layers.Dense(512, activation='relu', input_shape=(7,)),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(3)  # Output the last three values
])

# Compile both models
optimizer_1 = tf.keras.optimizers.Adam(learning_rate=0.001)
optimizer_2 = tf.keras.optimizers.Adam(learning_rate=0.001)
abb_model1.compile(optimizer=optimizer_1, loss='mean_squared_error')
abb_model2.compile(optimizer=optimizer_2, loss='mean_squared_error')

# Define early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)

# Train model1
print("Training abb model1...")
abb_model1.fit(x_train, y_train[:, :3], epochs=20, batch_size=32, validation_data=(x_test, y_test[:, :3]), callbacks=[early_stopping])
print("Training of abb model1 completed.")

# Train model2
print("Training abb model2...")
abb_model2.fit(x_train, y_train[:, 3:], epochs=20, batch_size=32, validation_data=(x_test, y_test[:, 3:]), callbacks=[early_stopping])
print("Training of abb model2 completed.")
```

```python
# Make predictions using both models
y_preds1 = abb_model1.predict(x_test)
y_preds2 = abb_model2.predict(x_test)

# Combine the predictions to form the final prediction variable
abb_final_predictions = np.hstack((y_preds1, y_preds2))

# Evaluate the combined model's performance
mse = np.mean(np.square(abb_final_predictions - y_test))
print('Mean Squared Error (Combined Model):', mse)

# Calculate the mean of the actual target values from the test set (mean_y_test)
mean_y_test = np.mean(y_test, axis=0)

# Calculate the total sum of squares (TSS)
TSS = np.sum(np.square(y_test - mean_y_test))

# Calculate the RSS (residual sum of squares)
RSS = np.sum(np.square(y_test - abb_final_predictions))

# Calculate R-squared (R²)
R_squared = 1 - (RSS / TSS)

print("R-squared value:", R_squared)
```

Code Snippet for Generation of Dataset

```python
# Import the necessary libraries
import numpy as np
import math  # For mathematical operations.
import csv  # To save generated training and test data into a CSV file.

# Function to generate random angles within specified ranges.
def generate_joint_angles():
    '''
    This function generates random angles within specified ranges.
    The angle ranges should be specified in radians form.
    '''
    joint1 = np.random.uniform(0, np.pi)  # 0-180 degrees
    joint2 = np.random.uniform(0, np.pi * 0.5)  # 0-90 degrees
    joint3 = np.random.uniform(0, np.pi * 0.5)  # 0-90 degrees
    joint4 = np.random.uniform(0, np.pi)  # 0-180 degrees
    joint5 = np.random.uniform(0, np.pi * 0.5)  # 0-90 degrees
    joint6 = np.random.uniform(0, np.pi)  # 0-180 degrees
    return [joint1, joint2, joint3, joint4, joint5, joint6]

# Create a function that performs the forward kinematics of the robot manipulator.
def forward_kinematics(theta):
    '''
    This function takes in the theta values for the specific problem and calculates the corresponding end effector position.
    The theta values should be passed into the function in radians form too.
    '''
    # DH parameters for the ABB IRB 1600
    a = [300, 1200, 250, 0, 0, 0]  # link lengths (mm)
    alpha = [np.pi/2, 0, np.pi/2, np.pi/2, np.pi/2, 0]  # link twists
    d = [649, 0, 0, 1320, 0, 290]  # link offsets (mm)

    # Homogeneous transformation matrices
    T = np.eye(4)  # initial transformation matrix (identity matrix)

    for i in range(len(theta)):
        # Call parameters for the transformation matrix.
        cos_theta = np.cos(theta[i])
```

```
                # Call parameters for the transformation matrix.
                cos_theta = np.cos(theta[i])
                sin_theta = np.sin(theta[i])
                cos_alpha = np.cos(alpha[i])
                sin_alpha = np.sin(alpha[i])

                # Denavit-Hartenberg transformation matrix
                A = np.array([[cos_theta, -sin_theta * cos_alpha, sin_theta * sin_alpha, a[i] * cos_theta],
                              [sin_theta, cos_theta * cos_alpha, -cos_theta * sin_alpha, a[i] * sin_theta],
                              [0, sin_alpha, cos_alpha, d[i]],
                              [0, 0, 0, 1]])

                # Update the transformation matrix
                T = np.dot(T, A)

                # Extract the end effector position from the transformation matrix
                end_effector_pos = T[:3, 3]

                # Create the quaternion representation of the XYZW-rotation
                m_00 = T[0, 0]
                m_01 = T[0, 1]
                m_02 = T[0, 2]
                m_10 = T[1, 0]
                m_11 = T[1, 1]
                m_12 = T[1, 2]
                m_20 = T[2, 0]
                m_21 = T[2, 1]
                m_22 = T[2, 2]

                w = math.sqrt((1 + m_00 + m_11 + m_22) / 2)
                x = (m_21 - m_12) / (4 * w)
                y = (m_02 - m_20) / (4 * w)
                z = (m_10 - m_01) / (4 * w)

                quart = np.array([w, x, y, z])
```

```
        w = math.sqrt((1 + m_00 + m_11 + m_22) / 2)
        x = (m_21 - m_12) / (4 * w)
        y = (m_02 - m_20) / (4 * w)
        z = (m_10 - m_01) / (4 * w)

        quart = np.array([w, x, y, z])

        final_pos = np.concatenate((end_effector_pos, quart))
    return final_pos

# Generate 100,000 random sets of joint angles.
np.random.seed(123)  # For reproducibility.
joint_angles = np.array([generate_joint_angles() for _ in range(100000)])  # Generate angles separately for each joint.


# Calculate forward kinematics for each set of joint angles.
final_positions = []
for angles in joint_angles:
    position = forward_kinematics(angles)
    final_positions.append(position)

final_positions = np.array(final_positions)

# Combine joint angles and end effector positions.
data = np.hstack((joint_angles, final_positions))  # Horizontally stack up the joint parameters with the corresponding end effector position

# Save joint angles and end effector positions to a CSV file
with open('joint_positions_abb.csv', 'w', newline='') as file: #The `newline` argument is used to ensure that the CSV file is written with consistent newline characters.
    writer = csv.writer(file)
    writer.writerow(['Joint1', 'Joint2', 'Joint3', 'Joint4', 'Joint5', 'Joint6', 'X', 'Y', 'Z', 'q1', 'q2', 'q3', 'q4'])  # Write header with xyz being the end effector p
    writer.writerows(data)  # Writes rows of entries to the csv file from the data we stacked using `np.hstack()`

print("CSV file generated successfully.")
```

```
CSV file generated successfully.
```

Arm 1 (ABB IRB 1600):

For the ABB IRB 1600 robot arm, the data was generated

Testing data set for arm (First 20 values)

Table A-1 Testing data set for arm 1 (First 20 values)

0.4504	**1.5686**	**0.4212**	**0.2400**	**1.3908**	**1.3005**	**1151.7841**	**481.5875**	**2846.4774**	**0.8440**	**-0.0089**	**-0.2233**	**0.5215**
1.4024	**0.5211**	**0.3419**	**0.7640**	**1.0954**	**2.8027**	**599.3081**	**2460.7707**	**806.0137**	**0.4075**	**-0.2163**	**0.0828**	**-0.6363**
0.9177	**0.6286**	**0.4214**	**1.9236**	**1.3900**	**2.7618**	**1698.6959**	**1779.8229**	**855.2038**	**0.1724**	**0.3836**	**-0.3927**	**0.4373**
0.5474	**0.2539**	**1.1320**	**2.7063**	**0.6006**	**2.2408**	**2206.9311**	**1264.3018**	**851.4381**	**0.7918**	**0.3394**	**-0.4720**	**0.0726**
1.5085	**0.7038**	**0.8909**	**0.1103**	**0.6740**	**1.9869**	**162.9725**	**2291.9524**	**1881.5308**	**0.1902**	**0.0400**	**0.3131**	**-0.6256**
0.2618	**0.2962**	**1.1643**	**1.2764**	**0.2965**	**0.6337**	**2449.5759**	**572.4547**	**1157.4722**	**1.0881**	**0.2610**	**-0.3668**	**-0.0370**
1.5952	**0.1700**	**0.3185**	**1.3379**	**0.2451**	**1.5884**	**14.7261**	**2206.2506**	**59.7949**	**0.7954**	**0.1575**	**-0.0668**	**0.5591**
1.2902	**0.1264**	**0.5964**	**2.2128**	**1.5034**	**3.0704**	**889.5925**	**2249.8633**	**-124.3034**	**0.1228**	**0.4506**	**-0.3649**	**0.4001**
2.9228	**0.3100**	**0.1505**	**0.0342**	**0.5325**	**0.9220**	**-2218.8130**	**498.6159**	**232.8981**	**0.4667**	**0.0183**	**-0.0186**	**-0.6670**
1.5352	**0.2933**	**0.7106**	**1.4589**	**1.3611**	**3.0711**	**376.4051**	**2650.8748**	**557.0078**	**0.4284**	**-0.3788**	**0.2356**	**-0.5051**
0.7537	**0.4455**	**1.0376**	**2.9413**	**0.6215**	**1.2615**	**1824.3513**	**1666.0079**	**1155.1821**	**0.6583**	**0.5841**	**-0.1821**	**-0.1318**

2.7018	0.7627	0.3434	1.2878	1.1984	1.8280	-2060.3413	1256.1968	1224.8289	0.4576	-0.3404	-0.1877	-0.5446
2.3200	0.4907	0.6846	0.4765	1.0956	0.5622	-1710.4983	2012.6517	1199.1708	0.3171	0.1176	0.0990	0.6717
1.7585	0.7673	0.8003	1.3710	0.6726	2.2953	-247.2941	2251.2054	1764.4056	0.0846	-0.3257	0.3352	-0.5289
1.9179	1.1730	0.4394	3.1087	1.4911	1.0241	-692.6326	1942.5464	1770.2826	0.0291	0.6378	0.3049	0.0102
1.4895	0.6493	0.3783	1.1687	1.4949	0.5596	472.6560	2524.6359	1014.5862	0.7992	0.2709	0.2886	0.4285
3.0948	0.5555	0.5733	2.1807	1.3440	0.9842	-2477.9378	347.8283	824.7005	0.0737	0.2614	0.5327	0.3828
2.9533	0.6630	0.5800	0.7905	0.4120	0.4484	-2293.7105	521.1243	1362.1352	0.2106	0.3129	0.1079	0.6159
2.1831	0.5740	0.6437	1.8889	0.3126	1.4590	-1289.5491	1983.2666	1147.9411	0.6160	0.4353	-0.0289	0.4635
2.2097	0.9512	0.1624	2.0398	0.0106	2.7996	-1208.8335	1631.9045	1394.2645	0.0995	0.2509	-0.2789	0.5973
2.8495	1.2631	1.0325	0.2848	0.0434	1.0998	-1206.5478	366.4408	2671.8961	0.1646	-0.5442	0.3342	-0.2923

Predicted data set for arm :- The predicted dataset is gotten from the neural networks model after it is trained on the training dataset. After the training on the training dataset, the model is tested on

the x,y,z position of the end effector, and the quaternion representation of the end effector frame (7 columns in total) from the test dataset, and it makes predictions of the 6 joint angles of the test dataset. The deviation of the predicted values from the actual values tells about the model performance and accuracy. The first 20 predicted values can be shown below;

Table A-2 Predicted data set for arm 1

joint1	**joint2**	**joint3**	**joint4**	**joint5**	**joint6**
0.1777	**0.6774**	**0.4429**	**1.4027**	**0.9855**	**0.7552**
2.1103	**0.0606**	**0.6624**	**0.7517**	**0.9240**	**2.7758**
0.6943	**0.7943**	**0.8676**	**2.7349**	**1.0053**	**2.4963**
2.6941	**0.1025**	**0.3547**	**1.8231**	**0.3451**	**1.2765**
1.7601	**0.9982**	**0.3063**	**2.1477**	**0.4262**	**0.3526**
2.0995	**0.1370**	**0.5326**	**0.5897**	**0.8884**	**0.6413**
0.6573	**1.2122**	**0.2261**	**2.9961**	**0.8549**	**1.1190**
1.1506	**0.8456**	**0.9783**	**1.1725**	**0.3519**	**1.4819**
2.3501	**1.2815**	**1.0904**	**1.0862**	**1.4910**	**1.0163**
0.2182	**0.8429**	**1.1536**	**2.2997**	**1.1067**	**1.2317**
3.0355	**0.1175**	**0.5510**	**1.8246**	**0.5897**	**0.2946**
1.4090	**0.8713**	**0.7778**	**2.2582**	**0.4605**	**2.5704**
2.6140	**1.4033**	**1.2232**	**2.2542**	**0.8678**	**1.6267**

0.6647	**0.0749**	**0.5335**	**2.4713**	**1.2319**	**2.9766**
1.1557	**0.1726**	**0.6638**	**0.2889**	**0.8437**	**2.4944**
1.9747	**0.6392**	**1.1050**	**2.6647**	**1.4727**	**0.8738**
0.5396	**0.1143**	**0.1430**	**2.8454**	**0.6015**	**0.4539**
0.7749	**0.2975**	**0.0243**	**1.6932**	**0.8550**	**2.8482**
0.1193	**1.2107**	**0.4273**	**1.5425**	**0.3880**	**2.5178**

Training data set for arm 1(First 20 values)

Table A-3 Training data set for arm 1(First 20 values)

Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	X	Y	Z	q1	q2	q3	q4
2.188022477	**0.449466616**	**0.35633743**	**1.732006428**	**1.130139215**	**1.329228147**	**-1170.868173**	**2097.054687**	**491.8783786**	**0.43119878**	**0.372620247**	**0.251971711**	**0.501179919**
3.081161601	**1.075728038**	**0.755446064**	**1.231873515**	**0.539062767**	**2.290377205**	**-1816.210854**	**250.5438329**	**2270.078582**	**0.692286211**	**-0.483138625**	**0.174878641**	**-0.340851112**
1.377815342	**0.093741821**	**0.625246454**	**2.318480945**	**0.28665734**	**0.551197948**	**505.3079003**	**2272.344212**	**105.3099843**	**1.248147547**	**0.310373421**	**0.069871037**	**-0.096531354**
1.669917891	**0.83539282**	**0.996514695**	**2.668568684**	**1.137971763**	**1.919586972**	**-104.1798316**	**2259.548536**	**1863.353779**	**0.045742606**	**0.672545071**	**-0.141178219**	**0.165011337**
2.269622823	**0.507302676**	**0.568296291**	**0.717110089**	**0.461364945**	**1.982269955**	**-1509.155658**	**1927.892821**	**1033.699149**	**0.362105559**	**-0.199351799**	**0.173415312**	**-0.630401365**

0.2893 56203	**0.6812 56209**	**0.6767 97646**	**1.5509 57476**	**0.6688 92656**	**0.9809 97564**	**2307.0 01029**	**499.22 38358**	**1421.9 03801**	**1.0758 14887**	**0.3374 18414**	**- 0.3028 73213**	**0.0712 19472**
1.3394 22134	**1.4033 32416**	**1.4830 83089**	**1.5765 66414**	**0.9801 03005**	**0.3632 25901**	**360.99 32747**	**481.94 22109**	**3015.9 40986**	**0.1130 60149**	**0.6047 11129**	**- 0.1391 13323**	**0.3343 29604**
0.9967 81739	**0.6516 0749**	**1.3607 95243**	**0.7868 28736**	**0.7587 48448**	**3.0962 27382**	**1253.6 23701**	**1678.4 59486**	**2204.3 54473**	**0.3957 11921**	**0.1996 14321**	**0.4217 74897**	**- 0.4930 64983**
1.6320 10634	**0.9627 3247**	**0.1894 83065**	**2.5960 26188**	**0.9472 84635**	**1.7123 81645**	**- 3.8823 86522**	**2060.3 0066**	**1210.7 36682**	**0.6361 00609**	**0.5893 37356**	**- 0.0431 9722**	**0.2228 44082**
1.0768 24342	**0.4777 11818**	**0.6550 56957**	**2.1403 69481**	**1.3751 64391**	**1.6035 39066**	**1420.3 99672**	**2132.4 26773**	**752.19 97303**	**0.5303 18044**	**0.5798 19461**	**- 0.1138 33484**	**0.2837 99267**
2.1027 11264	**0.9203 86984**	**0.9815 96126**	**2.1195 98166**	**1.3231 48407**	**0.2613 64764**	**- 895.92 4302**	**1995.3 41336**	**2107.8 24038**	**0.2035 81298**	**0.4515 60573**	**0.4307 97078**	**0.3164 5788**
2.3991 80404	**0.3827 50246**	**0.3050 84713**	**1.7984 26572**	**0.1503 4487**	**2.7813 36253**	**- 1632.7 74329**	**1555.5 35863**	**451.33 26349**	**0.1929 2876**	**- 0.1989 69038**	**0.1574 78071**	**- 0.6529 2159**
1.9705 60763	**1.1363 39758**	**0.0253 35699**	**1.8674 62826**	**0.8745 96135**	**0.4993 8645**	**- 550.90 47198**	**1850.6 0946**	**1456.0 89947**	**0.8378 08422**	**0.3990 71061**	**0.2800 21083**	**0.2947 02805**
0.4808 85206	**1.0925 35229**	**0.5007 17132**	**2.1738 88797**	**0.8708 23172**	**1.2219 24266**	**1842.0 61141**	**755.02 2904**	**1863.9 60772**	**0.7184 5382**	**0.5082 07359**	**- 0.3221 33082**	**0.0944 01188**
2.9063 89433	**1.3220 9214**	**0.5613 98785**	**0.1369 46622**	**0.4787 2857**	**1.2509 37213**	**- 1443.5 9332**	**364.69 44116**	**2502.7 57772**	**0.4916 16076**	**- 0.3658 87248**	**0.2753 80482**	**- 0.4794 48141**
2.2146 93483	**1.5635 05447**	**0.5590 69764**	**2.3956 1461**	**0.9317 60122**	**2.1730 45289**	**- 620.52 00619**	**1089.8 41975**	**2517.5 62491**	**0.2023 24569**	**- 0.6607 27655**	**0.1307 83598**	**- 0.1900 0213**

0.4747 80894	**0.6265 53415**	**0.3783 35559**	**1.0789 98891**	**0.8060 1982**	**2.0942 62789**	**2222.2 56859**	**934.84 3708**	**1046.9 56683**	**0.7711 64015**	**0.1534 80275**	**- 0.2484 6283**	**0.5157 87304**
0.3327 21319	**0.2056 09308**	**0.5057 65954**	**2.0783 6566**	**1.3296 88869**	**1.7381 0921**	**2327.0 18625**	**543.75 50597**	**20.448 44444**	**0.8966 55117**	**0.4669 50548**	**- 0.2525 45482**	**0.1310 73894**
2.6843 41658	**0.6045 0182**	**0.4976 09265**	**1.1129 55302**	**0.2687 34709**	**2.6047 34162**	**- 2072.5 80558**	**1096.7 47065**	**1114.4 60648**	**0.6894 62721**	**- 0.2459 04012**	**0.2334 0974**	**- 0.5159 5678**

Arm 2 (Fanuc LR mate 200iC):-

First 20 testing data set for the Fanuc arm.

Table A-4 First 20 testing data set for arm 2

0.4504	**1.5686**	**0.4212**	**0.1200**	**1.3908**	**1.3005**	**281.76 42**	**146.70 30**	**306.99 20**	**1.2661**	**0.0399**	**-0.0860**	**-0.3004**
1.4024	**0.5211**	**0.3419**	**0.3820**	**1.0954**	**2.8027**	**17.439 3**	**260.77 00**	**556.67 49**	**0.8133**	**0.3051**	**0.0868**	**-0.4837**
0.9177	**0.6286**	**0.4214**	**0.9618**	**1.3900**	**2.7618**	**112.17 30**	**252.85 40**	**505.51 79**	**0.2652**	**0.3992**	**-0.1330**	**-0.5526**
0.5474	**0.2539**	**1.1320**	**1.3532**	**0.6006**	**2.2408**	**- 10.830 7**	**45.094 5**	**435.93 80**	**0.1172**	**- 0.3765**	**0.0050**	**0.5957**
1.5085	**0.7038**	**0.8909**	**0.0552**	**0.6740**	**1.9869**	**4.0956**	**109.84 81**	**488.46 55**	**1.2391**	**0.2912**	**0.0473**	**-0.1706**
0.2618	**0.2962**	**1.1643**	**0.6382**	**0.2965**	**0.6337**	**-4.8040**	**13.128 0**	**427.13 26**	**1.0235**	**0.2162**	**-0.3057**	**-0.3128**
1.5952	**0.1700**	**0.3185**	**0.6690**	**0.2451**	**1.5884**	**- 17.457 8**	**222.26 86**	**659.13 05**	**1.3116**	**0.1308**	**-0.0085**	**-0.2296**

1.2902	**0.1264**	**0.5964**	**1.1064**	**1.5034**	**3.0704**	**-33.5445**	**141.3065**	**532.1221**	**0.0845**	**-0.5166**	**-0.1368**	**0.4611**
2.9228	**0.3100**	**0.1505**	**0.0171**	**0.5325**	**0.9220**	**-301.1424**	**66.2615**	**640.8259**	**0.7600**	**0.1238**	**0.0432**	**0.5817**
1.5352	**0.2933**	**0.7106**	**0.7295**	**1.3611**	**3.0711**	**-48.4226**	**105.6353**	**496.9953**	**0.2866**	**0.4581**	**0.2578**	**-0.4507**
0.7537	**0.4455**	**1.0376**	**1.4706**	**0.6215**	**1.2615**	**15.4830**	**78.0917**	**475.7723**	**0.5956**	**0.2491**	**-0.1656**	**-0.5673**
2.7018	**0.7627**	**0.3434**	**0.6439**	**1.1984**	**1.8280**	**-300.5108**	**91.9728**	**496.3907**	**1.2552**	**0.2994**	**0.0385**	**0.1226**
2.3200	**0.4907**	**0.6846**	**0.2382**	**1.0956**	**0.5622**	**-100.4636**	**83.3625**	**510.7782**	**0.8325**	**0.4137**	**-0.0909**	**0.3839**
1.7585	**0.7673**	**0.8003**	**0.6855**	**0.6726**	**2.2953**	**-61.7483**	**156.0172**	**500.3616**	**1.0855**	**0.2395**	**0.0343**	**-0.3832**
1.9179	**1.1730**	**0.4394**	**1.5543**	**1.4911**	**1.0241**	**-195.0858**	**304.8733**	**346.9411**	**1.0344**	**0.1412**	**-0.4610**	**0.0045**
1.4895	**0.6493**	**0.3783**	**0.5844**	**1.4949**	**0.5596**	**-22.3537**	**267.5781**	**500.6597**	**1.1068**	**0.2506**	**-0.3365**	**0.1329**
3.0948	**0.5555**	**0.5733**	**1.0904**	**1.3440**	**0.9842**	**-215.8523**	**-59.1023**	**507.7512**	**0.9660**	**0.4415**	**-0.0294**	**0.2663**
2.9533	**0.6630**	**0.5800**	**0.3952**	**0.4120**	**0.4484**	**-237.9054**	**32.7826**	**545.9806**	**0.6838**	**0.1170**	**-0.0190**	**0.6075**

2.1831	**0.5740**	**0.6437**	**0.9444**	**0.3126**	**1.4590**	**-137.0956**	**160.4968**	**556.0363**	**1.3825**	**0.1256**	**-0.0081**	**-0.0796**
2.2097	**0.9512**	**0.1624**	**1.0199**	**0.0106**	**2.7996**	**-286.7136**	**384.7631**	**421.1895**	**0.9079**	**-0.0313**	**-0.2681**	**-0.4702**
2.8495	**1.2631**	**1.0325**	**0.1424**	**0.0434**	**1.0998**	**-173.4062**	**51.6207**	**451.3524**	**0.9769**	**-0.0579**	**-0.0322**	**0.5070**
0.8605	**1.5499**	**0.6088**	**1.0385**	**1.2188**	**2.6724**	**143.6777**	**266.2806**	**326.1699**	**0.4898**	**0.1435**	**-0.2870**	**-0.5806**

First 20 predicted data set for the fanuc arm.

Table A-5 First 20 predicted data set for arm 2

joint1	joint2	joint3	joint4	joint5	joint6
0.2309	**0.8476**	**0.2950**	**0.5094**	**1.5229**	**0.6811**
2.1359	**0.1101**	**0.5669**	**0.2509**	**1.2215**	**2.8967**
0.7225	**0.9540**	**0.7178**	**1.2059**	**1.1793**	**2.1609**
2.5765	**0.1001**	**0.4235**	**0.7127**	**0.0724**	**1.6260**
1.8526	**1.0231**	**0.2452**	**0.6035**	**0.5754**	**0.2237**
2.0492	**0.2353**	**0.2303**	**0.1218**	**1.2363**	**1.0756**
0.5342	**1.2646**	**0.2661**	**1.4380**	**0.9735**	**0.8211**
1.2836	**0.8310**	**0.9983**	**0.3683**	**0.6282**	**1.2386**

2.4601	**1.4687**	**0.8602**	**0.6794**	**1.3490**	**1.1104**
0.3053	**0.5210**	**1.5382**	**1.4412**	**1.4745**	**0.5967**
2.9965	**0.1762**	**0.5048**	**1.0076**	**1.0996**	**0.3184**
1.4175	**0.9375**	**0.7854**	**1.2334**	**0.3972**	**2.5369**
2.4352	**1.4211**	**1.1965**	**1.3828**	**0.8612**	**1.7682**
0.7460	**0.1261**	**0.7448**	**1.2339**	**0.9683**	**2.9499**
1.1435	**0.3795**	**0.2553**	**0.2302**	**0.9603**	**2.7257**
2.0350	**0.9048**	**1.1823**	**1.3124**	**1.4060**	**0.9049**
0.5848	**0.1833**	**0.0988**	**1.1277**	**0.7655**	**0.2269**
0.7553	**0.3032**	**0.0114**	**0.9890**	**0.4483**	**2.7164**
0.0453	**1.1634**	**0.6685**	**0.9769**	**0.2002**	**1.9354**
1.7362	**0.4914**	**0.1424**	**0.6553**	**1.1901**	**0.1700**

First 20 training data set for arm 2.

Table A-6 First 20 training data set for arm 2

Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	X	Y	Z	q1	q2	q3	q4
2.1880	**0.4495**	**0.3563**	**0.8660**	**1.1301**	**1.3292**	**-193.5687**	**177.5034**	**563.5767**	**1.2160**	**0.3443**	**-0.1080**	**0.0112**

3.0812	**1.0757**	**0.7554**	**0.6159**	**0.5391**	**2.2904**	**-237.8887**	**-9.3751**	**469.0757**	**1.3864**	**0.1126**	**0.0267**	**0.0782**
1.3778	**0.0937**	**0.6252**	**1.1592**	**0.2867**	**0.5512**	**-1.1387**	**102.2679**	**601.3242**	**1.3030**	**0.2203**	**-0.0952**	**-0.1340**
1.6699	**0.8354**	**0.9965**	**1.3343**	**1.1380**	**1.9196**	**-82.9927**	**121.1035**	**441.7458**	**0.7817**	**0.3732**	**-0.1256**	**-0.4384**
2.2696	**0.5073**	**0.5683**	**0.3586**	**0.4614**	**1.9823**	**-141.5758**	**149.0589**	**568.5565**	**1.3659**	**0.1659**	**0.0734**	**-0.0261**
0.2894	**0.6813**	**0.6768**	**0.7755**	**0.6689**	**0.9810**	**181.1213**	**90.1578**	**524.3184**	**0.9933**	**0.0556**	**-0.2243**	**-0.4472**
1.3394	**1.4033**	**1.4831**	**0.7883**	**0.9801**	**0.3632**	**-51.7102**	**-14.0138**	**387.8859**	**1.2244**	**0.1704**	**-0.3063**	**0.0482**
0.9968	**0.6516**	**1.3608**	**0.3934**	**0.7587**	**3.0962**	**-33.0089**	**-12.1891**	**353.0168**	**0.2730**	**0.4145**	**0.2097**	**-0.5154**
1.6320	**0.9627**	**0.1895**	**1.2980**	**0.9473**	**1.7124**	**-89.1206**	**431.6797**	**410.8145**	**1.0991**	**0.0815**	**-0.3173**	**-0.3011**
1.0768	**0.4777**	**0.6551**	**1.0702**	**1.3752**	**1.6035**	**20.6131**	**183.4812**	**502.4875**	**0.6831**	**0.3684**	**-0.2953**	**-0.4005**
2.1027	**0.9204**	**0.9816**	**1.0598**	**1.3231**	**0.2614**	**-120.7700**	**71.8326**	**423.3151**	**1.0281**	**0.2804**	**-0.3427**	**0.1992**
2.3992	**0.3828**	**0.3051**	**0.8992**	**0.1503**	**2.7813**	**-228.9042**	**197.2945**	**622.2501**	**1.1337**	**0.0419**	**0.0014**	**-0.4206**
1.9706	**1.1363**	**0.0253**	**0.9337**	**0.8746**	**0.4994**	**-234.0325**	**427.1245**	**322.5728**	**1.1632**	**-0.1581**	**-0.2271**	**0.2918**

0.4809	**1.0925**	**0.5007**	**1.0869**	**0.8708**	**1.2219**	**271.4831**	**202.7364**	**429.7287**	**0.9203**	**-0.0976**	**-0.2391**	**-0.4707**
2.9064	**1.3221**	**0.5614**	**0.0685**	**0.4787**	**1.2509**	**-328.7517**	**76.1888**	**399.5131**	**0.9758**	**-0.0799**	**-0.0606**	**0.5019**
2.2147	**1.5635**	**0.5591**	**1.1978**	**0.9318**	**2.1730**	**-248.7436**	**231.7730**	**310.3455**	**1.1765**	**0.1744**	**-0.2791**	**-0.2138**
0.4748	**0.6266**	**0.3783**	**0.5395**	**0.8060**	**2.0943**	**243.9528**	**158.7309**	**548.5709**	**0.6679**	**0.1532**	**-0.1439**	**-0.5868**
0.3327	**0.2056**	**0.5058**	**1.0392**	**1.3297**	**1.7381**	**128.5517**	**115.2735**	**558.8730**	**0.2475**	**0.2960**	**-0.3765**	**-0.5053**
2.6843	**0.6045**	**0.4976**	**0.5565**	**0.2687**	**2.6047**	**-248.7411**	**109.8840**	**564.2822**	**1.3695**	**0.0581**	**0.0310**	**-0.1637**
1.0640	**0.8677**	**0.9088**	**0.8192**	**0.0042**	**3.1050**	**85.8817**	**155.1932**	**497.7225**	**0.1982**	**0.0103**	**0.0118**	**-0.7000**

Arm 3 (ur10 arm)

First 20 testing data set for the arm.

Table A-7 First 20 testing data set for the arm.3

0.4504	**3.1372**	**0.4212**	**0.1200**	**1.3908**	**0.6502**	**-926.2012**	**-648.2927**	**43.2559**	**1.1761**	**-0.1776**	**0.3044**	**-0.1731**
1.4024	**1.0423**	**0.3419**	**0.3820**	**1.0954**	**1.4014**	**294.2685**	**501.1685**	**1159.6570**	**0.3010**	**-0.4724**	**0.4949**	**-0.0964**
0.9177	**1.2572**	**0.4214**	**0.9618**	**1.3900**	**1.3809**	**302.4802**	**98.2398**	**1335.2139**	**0.7100**	**-0.2910**	**0.5324**	**0.0764**

0.5474	**0.5079**	**1.1320**	**1.3532**	**0.6006**	**1.1204**	**606.2283**	**88.5687**	**1101.2958**	**0.8398**	**-0.0378**	**0.5191**	**-0.2297**
1.5085	**1.4077**	**0.8909**	**0.0552**	**0.6740**	**0.9934**	**225.5165**	**-173.4660**	**1198.3754**	**0.6889**	**-0.4499**	**0.3977**	**-0.1440**
0.2618	**0.5925**	**1.1643**	**0.6382**	**0.2965**	**0.3169**	**547.9659**	**-114.0420**	**1097.0307**	**0.0335**	**-0.2356**	**0.4941**	**-0.4473**
1.5952	**0.3400**	**0.3185**	**0.6690**	**0.2451**	**0.7942**	**225.5127**	**1141.4246**	**631.7666**	**0.2197**	**-0.5326**	**0.1588**	**-0.4232**
1.2902	**0.2527**	**0.5964**	**1.1064**	**1.5034**	**1.5352**	**471.3444**	**1020.8674**	**667.5371**	**0.2462**	**-0.4233**	**0.5496**	**0.0604**
2.9228	**0.6200**	**0.1505**	**0.0171**	**0.5325**	**0.4610**	**-881.4258**	**445.2291**	**766.7743**	**0.2217**	**-0.4544**	**-0.3653**	**-0.3844**
1.5352	**0.5865**	**0.7106**	**0.7295**	**1.3611**	**1.5356**	**211.7510**	**800.1526**	**986.2604**	**0.3308**	**-0.4725**	**0.4963**	**0.0555**
0.7537	**0.8909**	**1.0376**	**1.4706**	**0.6215**	**0.6308**	**314.3769**	**-32.4506**	**1263.5655**	**0.9553**	**-0.1181**	**0.4461**	**-0.2428**
2.7018	**1.5255**	**0.3434**	**0.6439**	**1.1984**	**0.9140**	**87.0919**	**177.2582**	**1327.7411**	**0.5278**	**-0.6220**	**-0.0080**	**0.2083**
2.3200	**0.9814**	**0.6846**	**0.2382**	**1.0956**	**0.2811**	**-136.0169**	**448.8411**	**1165.4757**	**0.3602**	**-0.6656**	**-0.1303**	**-0.0866**
1.7585	**1.5346**	**0.8003**	**0.6855**	**0.6726**	**1.1476**	**288.3194**	**-253.4795**	**1258.8094**	**0.9478**	**-0.3290**	**0.4007**	**0.0810**
1.9179	**2.3459**	**0.4394**	**1.5543**	**1.4911**	**0.5121**	**514.2488**	**-917.8833**	**890.9521**	**1.2668**	**-0.1282**	**0.0384**	**0.2844**
1.4895	**1.2987**	**0.3783**	**0.5844**	**1.4949**	**0.2798**	**190.79**	**236.26**	**1287.7**	**0.4809**	**-0.6525**	**0.1283**	**-0.0014**

						00	71	849				
3.0948	1.1109	0.5733	1.0904	1.3440	0.4921	-322.8285	199.9942	1318.9171	0.5360	-0.5261	-0.2463	0.3013
2.9533	1.3259	0.5800	0.3952	0.4120	0.2242	-22.7295	257.1036	1310.2028	0.8140	-0.5382	-0.1868	-0.0990
2.1831	1.1480	0.6437	0.9444	0.3126	0.7295	92.5171	305.9103	1337.7279	0.9248	-0.4632	0.2565	-0.0765
2.2097	1.9023	0.1624	1.0199	0.0106	1.3998	481.5031	-218.8992	1324.0693	0.9776	-0.2125	0.4528	0.1046
2.8495	2.5262	1.0325	0.1424	0.0434	0.5499	1108.2913	-65.9803	348.1190	0.9207	-0.3836	0.3116	0.2094
0.8605	3.0999	0.6088	1.0385	1.2188	1.3362	-641.8369	-1046.4028	-72.4760	1.3232	0.1120	0.0541	0.2164

Predicted data set for arm.

Table A-8 Predicted data set for arm 3

joint1	joint2	joint3	joint4	joint5	joint6
0.1688	1.6252	0.5523	0.4118	1.4729	0.4766
2.1461	0.3066	0.2330	0.1381	1.3554	1.1631
0.6952	1.9303	0.7992	0.9955	1.1730	1.3498

2.6385	**0.2621**	**0.3511**	**0.7005**	**0.0771**	**0.8081**
1.8480	**2.0865**	**0.3299**	**0.5978**	**0.6319**	**0.2543**
2.0467	**0.4103**	**0.3241**	**0.0949**	**1.1899**	**0.4377**
0.5749	**2.4985**	**0.4683**	**1.1516**	**0.8040**	**0.5860**
1.1874	**1.6932**	**1.0348**	**0.2580**	**0.6358**	**0.8916**
2.4635	**3.0664**	**0.7385**	**0.6776**	**1.3042**	**0.5562**
0.2268	**1.5817**	**1.5296**	**0.9786**	**1.2136**	**0.4560**
3.0191	**0.4906**	**0.5278**	**0.9662**	**1.1249**	**0.2599**
1.4033	**1.7772**	**0.9626**	**1.0041**	**0.4091**	**1.4551**
2.5230	**3.2776**	**1.2983**	**1.3201**	**0.8172**	**0.9334**
0.6741	**0.1879**	**0.6940**	**1.2178**	**0.9572**	**1.4434**
1.1779	**0.5376**	**0.4740**	**0.0922**	**0.7550**	**1.1549**
2.0203	**1.6159**	**1.1989**	**1.3004**	**1.3706**	**0.5357**
0.5833	**0.2600**	**0.2347**	**1.0961**	**0.6770**	**0.1541**
0.7820	**0.3663**	**0.4380**	**0.8310**	**0.4196**	**1.2942**
0.0952	**2.3694**	**0.8102**	**0.7509**	**0.1831**	**1.0540**
1.7219	**0.9411**	**0.2834**	**0.5933**	**1.0353**	**0.1007**

First 20 Training data set for the arm 3.

Table A-9 First 20 Training data set for the arm 3

Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	X	Y	Z	q1	q2	q3	q4
2.1880	0.8989	0.3563	0.8660	1.1301	0.6646	-239.3886	688.4526	1138.9891	0.4785	-0.6629	0.0577	-0.0071
3.0812	2.1515	0.7554	0.6159	0.5391	1.1452	904.0474	188.6987	896.0229	0.6246	-0.3914	0.2215	0.4475
1.3778	0.1875	0.6252	1.1592	0.2867	0.2756	460.5533	1041.5709	677.3964	0.2597	-0.5202	0.2109	-0.4099
1.6699	1.6708	0.9965	1.3343	1.1380	0.9598	261.2280	-579.6078	1135.5034	1.1710	-0.1590	0.2296	0.2814
2.2696	1.0146	0.5683	0.3586	0.4614	0.9911	-93.2804	493.9943	1222.1949	0.6946	-0.5745	0.1539	-0.1601
0.2894	1.3625	0.6768	0.7755	0.6689	0.4905	28.2906	-237.9965	1326.7336	0.4914	-0.1881	0.5142	-0.3740
1.3394	2.8067	1.4831	0.7883	0.9801	0.1816	-7.7346	-971.4429	-163.2623	1.3319	0.1619	0.1729	0.0209
0.9968	1.3032	1.3608	0.3934	0.7587	1.5481	45.2967	-354.9257	1089.4028	0.9695	-0.0684	0.5097	0.0240
1.6320	1.9255	0.1895	1.2980	0.9473	0.8562	245.9114	-453.3308	1321.1062	1.0772	-0.3086	0.3168	0.1196
1.0768	0.9554	0.6551	1.0702	1.3752	0.8018	379.4565	321.0353	1261.1429	0.7508	-0.4550	0.3900	-0.0012

2.1027	**1.8408**	**0.9816**	**1.0598**	**1.3231**	**0.1307**	**524.9808**	**-524.2601**	**1041.4004**	**1.2373**	**-0.2932**	**0.0219**	**0.1756**
2.3992	**0.7655**	**0.3051**	**0.8992**	**0.1503**	**1.3907**	**-436.9443**	**746.9275**	**1084.9530**	**0.9039**	**-0.4795**	**0.2336**	**-0.1060**
1.9706	**2.2727**	**0.0253**	**0.9337**	**0.8746**	**0.2497**	**483.8732**	**-572.2511**	**1142.5274**	**1.0340**	**-0.4604**	**0.1440**	**0.0024**
0.4809	**2.1851**	**0.5007**	**1.0869**	**0.8708**	**0.6110**	**-674.6019**	**-603.7913**	**1013.2746**	**1.0987**	**-0.0359**	**0.3686**	**-0.2471**
2.9064	**2.6442**	**0.5614**	**0.0685**	**0.4787**	**0.6255**	**1109.3102**	**-13.2209**	**502.0279**	**0.9080**	**-0.4693**	**0.1206**	**0.2431**
2.2147	**3.1270**	**0.5591**	**1.1978**	**0.9318**	**1.0865**	**911.7144**	**-850.0244**	**-107.2244**	**0.8385**	**0.1047**	**0.2033**	**0.5215**
0.4748	**1.2531**	**0.3783**	**0.5395**	**0.8060**	**1.0471**	**361.0344**	**-70.4458**	**1289.4900**	**0.3754**	**-0.2316**	**0.5876**	**-0.2567**
0.3327	**0.4112**	**0.5058**	**1.0392**	**1.3297**	**0.8691**	**1052.2368**	**166.8613**	**786.1121**	**0.2337**	**-0.3265**	**0.6069**	**-0.1067**
2.6843	**1.2090**	**0.4976**	**0.5565**	**0.2687**	**1.3024**	**-106.7320**	**334.1452**	**1320.7155**	**0.8892**	**-0.5131**	**0.1946**	**0.0353**
1.0640	**1.7353**	**0.9088**	**0.8192**	**0.0042**	**1.5525**	**-86.3096**	**-682.8444**	**1112.9253**	**0.9957**	**0.1960**	**0.4596**	**-0.0498**

Arm 4 (The Kuka KR Agilus robot arm)

Dataset generated for arm: - The dataset was also generated using the same procedure as the other robot arms.

First 20 testing data set for the arm

Table A-10 First 20 testing data set for the arm 4

0.4504	**1.5686**	**0.4212**	**0.1200**	**1.3908**	**1.3005**	**685.9605**	**346.7180**	**743.7690**	**0.0798**	**0.6331**	**-0.3004**	**-0.0860**
1.4024	**0.5211**	**0.3419**	**0.3820**	**1.0954**	**2.8027**	**104.0508**	**839.4131**	**41.3465**	**0.6103**	**0.4067**	**-0.4837**	**0.0868**
0.9177	**0.6286**	**0.4214**	**0.9618**	**1.3900**	**2.7618**	**443.5065**	**732.4223**	**137.2016**	**0.7984**	**0.1326**	**-0.5526**	**-0.1330**
0.5474	**0.2539**	**1.1320**	**1.3532**	**0.6006**	**2.2408**	**204.6482**	**199.0562**	**12.6213**	**0.7529**	**-0.0586**	**-0.5957**	**-0.0050**
1.5085	**0.7038**	**0.8909**	**0.0552**	**0.6740**	**1.9869**	**30.4802**	**552.0553**	**81.8308**	**0.5825**	**0.6196**	**-0.1706**	**0.0473**
0.2618	**0.2962**	**1.1643**	**0.6382**	**0.2965**	**0.6337**	**243.0052**	**85.8383**	**32.3841**	**0.4325**	**0.5118**	**-0.3128**	**-0.3057**
1.5952	**0.1700**	**0.3185**	**0.6690**	**0.2451**	**1.5884**	**-35.8174**	**759.5254**	**-236.5097**	**0.2616**	**0.6558**	**-0.2296**	**-0.0085**
1.2902	**0.1264**	**0.5964**	**1.1064**	**1.5034**	**3.0704**	**54.3086**	**558.8729**	**-115.1150**	**1.0332**	**-0.0423**	**-0.4611**	**0.1368**
2.9228	**0.3100**	**0.1505**	**0.0171**	**0.5325**	**0.9220**	**-904.5996**	**200.1571**	**-117.1034**	**0.2476**	**0.3800**	**0.5817**	**0.0432**
1.5352	**0.2933**	**0.7106**	**0.7295**	**1.3611**	**3.0711**	**-56.1921**	**528.7536**	**-29.8789**	**0.9162**	**0.1433**	**-0.4507**	**0.2578**

0.7537	**0.4455**	**1.0376**	**1.4706**	**0.6215**	**1.2615**	**258.6594**	**334.1124**	**3.2522**	**0.4983**	**0.2978**	**-0.5673**	**-0.1656**
2.7018	**0.7627**	**0.3434**	**0.6439**	**1.1984**	**1.8280**	**-854.9015**	**331.2229**	**224.0068**	**0.5987**	**0.6276**	**0.1226**	**0.0385**
2.3200	**0.4907**	**0.6846**	**0.2382**	**1.0956**	**0.5622**	**-426.1977**	**422.7834**	**18.7512**	**0.8274**	**0.4163**	**0.3839**	**-0.0909**
1.7585	**0.7673**	**0.8003**	**0.6855**	**0.6726**	**2.2953**	**-166.3942**	**633.0325**	**104.8282**	**0.4790**	**0.5428**	**-0.3832**	**0.0343**
1.9179	**1.1730**	**0.4394**	**1.5543**	**1.4911**	**1.0241**	**-421.2419**	**827.4727**	**564.9027**	**0.2824**	**0.5172**	**0.0045**	**-0.4610**
1.4895	**0.6493**	**0.3783**	**0.5844**	**1.4949**	**0.5596**	**5.9418**	**852.0233**	**163.5461**	**0.5013**	**0.5534**	**0.1329**	**-0.3365**
3.0948	**0.5555**	**0.5733**	**1.0904**	**1.3440**	**0.9842**	**-750.2927**	**-64.3697**	**69.4271**	**0.8830**	**0.4830**	**0.2663**	**-0.0294**
2.9533	**0.6630**	**0.5800**	**0.3952**	**0.4120**	**0.4484**	**-779.5909**	**130.5194**	**52.8132**	**0.2340**	**0.3419**	**0.6075**	**-0.0190**
2.1831	**0.5740**	**0.6437**	**0.9444**	**0.3126**	**1.4590**	**-444.2416**	**582.5948**	**-8.7685**	**0.2511**	**0.6913**	**-0.0796**	**-0.0081**
2.2097	**0.9512**	**0.1624**	**1.0199**	**0.0106**	**2.7996**	**-677.0635**	**909.7200**	**455.1604**	**0.0625**	**-0.4539**	**0.4702**	**0.2681**
2.8495	**1.2631**	**1.0325**	**0.1424**	**0.0434**	**1.0998**	**-614.7604**	**184.0897**	**282.7019**	**0.1159**	**-0.4884**	**-0.5070**	**0.0322**
0.8605	**1.5499**	**0.6088**	**1.0385**	**1.2188**	**2.6724**	**427.7053**	**639.9261**	**664.1185**	**0.2870**	**0.2449**	**-0.5806**	**-0.2870**

Predicted data set for arm

Table A-11 Predicted data set for arm 4

joint1	joint2	joint3	joint4	joint5	joint6
0.2276	**0.8695**	**0.2876**	**0.7867**	**1.2683**	**0.5257**
2.1485	**0.1113**	**0.5850**	**0.3914**	**1.1112**	**2.8494**
0.7169	**0.9340**	**0.7148**	**1.0844**	**1.2052**	**2.1803**
2.6688	**0.1095**	**0.4648**	**0.8075**	**0.0662**	**1.6171**
1.7962	**0.9883**	**0.2141**	**0.7613**	**0.3833**	**0.1812**
2.0428	**0.2289**	**0.2396**	**0.3253**	**1.0906**	**1.0894**
0.6040	**1.1506**	**0.2122**	**1.4455**	**0.8454**	**0.6838**
1.1744	**0.8476**	**0.9976**	**0.3873**	**0.5935**	**1.2555**
2.4638	**1.4881**	**0.8486**	**0.8653**	**1.3261**	**1.1641**
0.2155	**0.6753**	**1.5561**	**1.2541**	**1.2083**	**0.7771**
3.0900	**0.1668**	**0.5693**	**1.0100**	**0.9154**	**0.6317**
1.4056	**0.9123**	**0.7848**	**1.3350**	**0.3898**	**2.4718**
2.5342	**1.4825**	**1.2280**	**1.4049**	**0.8061**	**1.7883**
0.7236	**0.1068**	**0.7329**	**1.2950**	**1.0325**	**2.8988**

1.1557	**0.3571**	**0.2313**	**0.2195**	**0.8998**	**2.5319**
2.0551	**0.7569**	**1.1679**	**1.4925**	**1.4909**	**0.8778**
0.5899	**0.1417**	**0.0960**	**1.3296**	**0.6808**	**0.1402**
0.7930	**0.3272**	**0.0179**	**0.8782**	**0.6021**	**2.8469**
0.1236	**1.1702**	**0.6622**	**0.7549**	**0.0932**	**2.1660**
1.6977	**0.5088**	**0.1460**	**0.7858**	**1.1289**	**0.1237**

First 20 training data set for the arm 4

Table A-12 First 20 training data set for the arm 4

Joint1	**Joint2**	**Joint3**	**Joint4**	**Joint5**	**Joint6**	**X**	**Y**	**Z**	**q1**	**q2**	**q3**	**q4**
2.1880	**0.4495**	**0.3563**	**0.8660**	**1.1301**	**1.3292**	**-547.8957**	**635.0814**	**-0.5684**	**0.6886**	**0.6080**	**0.0112**	**-0.1080**
3.0812	**1.0757**	**0.7554**	**0.6159**	**0.5391**	**2.2904**	**-760.1604**	**11.8261**	**268.4276**	**0.2251**	**0.6932**	**0.0782**	**0.0267**
1.3778	**0.0937**	**0.6252**	**1.1592**	**0.2867**	**0.5512**	**66.5406**	**495.8996**	**-227.6225**	**0.4407**	**0.6515**	**-0.1340**	**-0.0952**
1.6699	**0.8354**	**0.9965**	**1.3343**	**1.1380**	**1.9196**	**-157.3612**	**556.7919**	**170.0290**	**0.7464**	**0.3908**	**-0.4384**	**-0.1256**
2.2696	**0.5073**	**0.5683**	**0.3586**	**0.4614**	**1.9823**	**-484.8523**	**549.0839**	**-34.5316**	**0.3318**	**0.6829**	**-0.0261**	**0.0734**

0.2894	**0.6813**	**0.6768**	**0.7755**	**0.6689**	**0.9810**	**674.2044**	**252.8077**	**67.9581**	**0.1112**	**0.4966**	**-0.4472**	**-0.2243**
1.3394	**1.4033**	**1.4831**	**0.7883**	**0.9801**	**0.3632**	**-2.1777**	**286.1273**	**305.5379**	**0.3407**	**0.6122**	**0.0482**	**-0.3063**
0.9968	**0.6516**	**1.3608**	**0.3934**	**0.7587**	**3.0962**	**101.9052**	**213.4472**	**199.4763**	**0.8290**	**0.1365**	**-0.5154**	**0.2097**
1.6320	**0.9627**	**0.1895**	**1.2980**	**0.9473**	**1.7124**	**-155.2350**	**1063.0871**	**466.9475**	**0.1630**	**0.5496**	**-0.3011**	**-0.3173**
1.0768	**0.4777**	**0.6551**	**1.0702**	**1.3752**	**1.6035**	**227.8441**	**631.8365**	**31.9512**	**0.7368**	**0.3415**	**-0.4005**	**-0.2953**
2.1027	**0.9204**	**0.9816**	**1.0598**	**1.3231**	**0.2614**	**-372.4061**	**441.0705**	**226.5210**	**0.5608**	**0.5140**	**0.1992**	**-0.3427**
2.3992	**0.3828**	**0.3051**	**0.8992**	**0.1503**	**2.7813**	**-674.0874**	**600.1906**	**-98.9282**	**0.0838**	**0.5668**	**-0.4206**	**0.0014**
1.9706	**1.1363**	**0.0253**	**0.9337**	**0.8746**	**0.4994**	**-487.3132**	**971.1178**	**696.9006**	**0.3163**	**-0.5816**	**-0.2918**	**0.2271**
0.4809	**1.0925**	**0.5007**	**1.0869**	**0.8708**	**1.2219**	**769.8749**	**489.4942**	**400.7571**	**0.1952**	**-0.4601**	**0.4707**	**0.2391**
2.9064	**1.3221**	**0.5614**	**0.0685**	**0.4787**	**1.2509**	**-850.9195**	**200.1860**	**504.7950**	**0.1599**	**-0.4879**	**-0.5019**	**0.0606**
2.2147	**1.5635**	**0.5591**	**1.1978**	**0.9318**	**2.1730**	**-556.0765**	**597.6322**	**703.3903**	**0.3488**	**0.5882**	**-0.2138**	**-0.2791**
0.4748	**0.6266**	**0.3783**	**0.5395**	**0.8060**	**2.0943**	**767.8277**	**442.5864**	**85.6240**	**0.3064**	**0.3340**	**-0.5868**	**-0.1439**
0.3327	**0.2056**	**0.5058**	**1.0392**	**1.3297**	**1.7381**	**570.1363**	**298.8681**	**-114.75**	**0.5919**	**0.1238**	**-0.5053**	**-0.3765**

								60				
2.6843	**0.6045**	**0.4976**	**0.5565**	**0.2687**	**2.6047**	**-765.4776**	**358.6632**	**24.3471**	**0.1162**	**0.6847**	**-0.1637**	**0.0310**
1.0640	**0.8677**	**0.9088**	**0.8192**	**0.0042**	**3.1050**	**318.7567**	**574.8554**	**118.3936**	**0.0205**	**0.0991**	**-0.7000**	**0.0118**

Arm 5 (The Yasakawa Motoman GP arm)

Dataset generated for arm

Testing data set for arm

Table A-13 Testing data set for arm 5

0.2252	**1.5686**	**0.4212**	**0.1200**	**2.7817**	**1.3005**	**-63.9382**	**454.5565**	**695.1263**	**0.4428**	**0.2651**	**-0.5144**	**-0.3408**
0.7012	**0.5211**	**0.3419**	**0.3820**	**2.1909**	**2.8027**	**206.6421**	**423.4890**	**723.5122**	**0.6987**	**0.0199**	**0.5632**	**0.2456**
0.4589	**0.6286**	**0.4214**	**0.9618**	**2.7799**	**2.7618**	**267.3074**	**434.0722**	**695.1755**	**0.2982**	**-0.0065**	**0.5317**	**0.4415**
0.2737	**0.2539**	**1.1320**	**1.3532**	**1.2011**	**2.2408**	**493.4206**	**265.1256**	**798.9044**	**0.5054**	**-0.4654**	**0.4649**	**0.0587**
0.7542	**0.7038**	**0.8909**	**0.0552**	**1.3480**	**1.9869**	**108.8807**	**445.5897**	**787.6728**	**0.3277**	**-0.2931**	**0.6222**	**-0.0116**
0.1309	**0.2962**	**1.1643**	**0.6382**	**0.5929**	**0.6337**	**471.2210**	**194.2114**	**836.3437**	**0.3418**	**0.0383**	**-0.5756**	**0.3716**
0.7976	**0.1700**	**0.3185**	**0.6690**	**0.4902**	**1.5884**	**305.7314**	**398.5684**	**840.5803**	**0.1494**	**-0.1189**	**0.5946**	**-0.3560**
0.6451	**0.1264**	**0.5964**	**1.1064**	**3.0068**	**3.0704**	**367.48**	**366.82**	**690.72**	**0.3558**	**0.1151**	**0.4890**	**0.4647**

						06	**69**	**52**				
1.4614	**0.3100**	**0.1505**	**0.0171**	**1.0651**	**0.9220**	**-72.8702**	**448.1005**	**808.7538**	**0.3677**	**0.0634**	**-0.6483**	**0.2048**
0.7676	**0.2933**	**0.7106**	**0.7295**	**2.7221**	**3.0711**	**264.1405**	**435.8867**	**696.9357**	**0.4703**	**0.0400**	**0.5055**	**0.4331**
0.3768	**0.4455**	**1.0376**	**1.4706**	**1.2430**	**1.2615**	**403.0306**	**393.3333**	**795.7538**	**0.0387**	**-0.5199**	**0.4530**	**-0.1554**
1.3509	**0.7627**	**0.3434**	**0.6439**	**2.3969**	**1.8280**	**-197.9534**	**443.1592**	**711.1765**	**0.2701**	**0.4186**	**-0.4885**	**-0.2606**
1.1600	**0.4907**	**0.6846**	**0.2382**	**2.1911**	**0.5622**	**7.6431**	**474.1010**	**723.4944**	**0.7523**	**0.3748**	**-0.4669**	**-0.0093**
0.8792	**0.7673**	**0.8003**	**0.6855**	**1.3453**	**2.2953**	**39.0734**	**499.4189**	**787.8899**	**0.4585**	**-0.4825**	**0.4469**	**0.1224**
0.9590	**1.1730**	**0.4394**	**1.5543**	**2.9822**	**1.0241**	**-240.4716**	**441.6702**	**691.0136**	**0.9295**	**0.5319**	**-0.0313**	**0.0114**
0.7448	**0.6493**	**0.3783**	**0.5844**	**2.9898**	**0.5596**	**108.1879**	**495.8224**	**690.9199**	**0.7887**	**0.3858**	**-0.3481**	**-0.2728**
1.5474	**0.5555**	**0.5733**	**1.0904**	**2.6880**	**0.9842**	**-232.1419**	**461.9303**	**698.0887**	**0.7967**	**0.5604**	**-0.1650**	**-0.0091**
1.4766	**0.6630**	**0.5800**	**0.3952**	**0.8239**	**0.4484**	**-213.7386**	**429.5673**	**824.3488**	**0.1974**	**0.2905**	**-0.4951**	**0.4009**
1.0916	**0.5740**	**0.6437**	**0.9444**	**0.6253**	**1.4590**	**-4.0645**	**513.3150**	**834.8652**	**0.4325**	**-0.4567**	**0.4323**	**-0.2405**
1.1048	**0.9512**	**0.1624**	**1.0199**	**0.0211**	**2.7996**	**-212.0223**	**461.4374**	**849.9821**	**0.9901**	**-0.4974**	**0.0663**	**-0.0560**

1.4248	**1.2631**	**1.0325**	**0.1424**	**0.0868**	**1.0998**	**-422.4382**	**217.3130**	**849.6990**	**0.7464**	**-0.4158**	**0.3105**	**-0.3023**
0.4302	**1.5499**	**0.6088**	**1.0385**	**2.4375**	**2.6724**	**-129.0319**	**470.0939**	**709.0239**	**0.0866**	**0.3881**	**-0.4154**	**-0.4183**

Predicted data set for arm

Table A-14 Predicted data set for arm 5

joint1	joint2	joint3	joint4	joint5	joint6
0.4681	**0.5068**	**0.6745**	**0.4077**	**2.7051**	**0.7992**
0.6075	**0.5658**	**0.5625**	**0.4097**	**2.3436**	**2.8837**
0.7430	**0.5345**	**0.9653**	**0.9342**	**2.1975**	**2.4393**
0.9986	**0.4830**	**0.5643**	**0.7551**	**0.1889**	**1.6552**
1.0873	**0.7577**	**0.4967**	**0.5206**	**1.3690**	**0.0390**
0.7432	**0.5281**	**0.4756**	**0.2609**	**2.1567**	**1.1514**
0.6509	**0.7508**	**1.0929**	**1.0069**	**1.5926**	**0.7830**
0.6878	**0.6367**	**0.7453**	**0.5975**	**0.9990**	**1.2928**
1.3339	**1.4011**	**0.9934**	**0.6016**	**2.6729**	**1.2544**
0.4035	**0.5428**	**1.3042**	**1.0358**	**2.5016**	**0.6294**
0.9419	**0.7420**	**0.8719**	**1.0173**	**2.1252**	**0.3175**

0.8823	**0.6731**	**1.0354**	**0.9781**	**0.8300**	**2.7989**
1.4424	**1.4455**	**1.0736**	**1.2163**	**1.6479**	**1.9026**
0.2208	**0.2490**	**0.9754**	**1.2598**	**1.9601**	**3.1017**
0.3854	**0.5812**	**0.6148**	**0.2569**	**1.5473**	**2.7261**
1.0557	**0.7585**	**0.9970**	**1.0601**	**2.8292**	**0.6532**
0.1780	**0.1925**	**0.7867**	**1.0464**	**1.3008**	**0.1445**
0.2894	**0.3155**	**0.7222**	**0.4150**	**0.8083**	**2.9193**
0.3520	**0.8702**	**1.0926**	**0.4644**	**0.2880**	**2.3756**
0.6651	**0.6248**	**0.6793**	**0.4332**	**2.2135**	**0.1491**

Training data set for arm 5

Table A-15 Training data set for arm 5

Joint1	Joint2	Joint3	Joint4	Joint5	Joint6	X	Y	Z	q1	q2	q3	q4
1.0940	**0.4495**	**0.3563**	**0.8660**	**2.2603**	**1.3292**	**75.0067**	**495.5743**	**719.1090**	**0.3995**	**0.3955**	**-0.5300**	**-0.1510**
1.5406	**1.0757**	**0.7554**	**0.6159**	**1.0781**	**2.2904**	**-376.0169**	**306.7781**	**807.8385**	**0.4730**	**-0.6217**	**0.1708**	**0.1686**
0.6889	**0.0937**	**0.6252**	**1.1592**	**0.5733**	**0.5512**	**389.3195**	**355.1155**	**837.2086**	**0.2503**	**0.1140**	**-0.5550**	**0.4041**

0.8350	**0.8354**	**0.9965**	**1.3343**	**2.2759**	**1.9196**	**-13.1897**	**536.8823**	**718.1483**	**0.3707**	**0.5979**	**-0.2673**	**-0.1917**
1.1348	**0.5073**	**0.5683**	**0.3586**	**0.9227**	**1.9823**	**14.9132**	**477.6612**	**818.2917**	**0.4514**	**-0.3544**	**0.5541**	**-0.1280**
0.1447	**0.6813**	**0.6768**	**0.7755**	**1.3378**	**0.9810**	**394.5396**	**313.6697**	**788.4727**	**0.3285**	**0.2104**	**-0.6387**	**0.1445**
0.6697	**1.4033**	**1.4831**	**0.7883**	**1.9602**	**0.3632**	**-192.8496**	**467.8198**	**739.6286**	**0.3656**	**0.5757**	**-0.0281**	**0.3666**
0.4984	**0.6516**	**1.3608**	**0.3934**	**1.5175**	**3.0962**	**262.2555**	**430.1056**	**774.2619**	**0.7844**	**-0.3263**	**0.3935**	**0.2913**
0.8160	**0.9627**	**0.1895**	**1.2980**	**1.8946**	**1.7124**	**-4.6899**	**510.8616**	**744.5483**	**0.1052**	**0.5166**	**-0.4667**	**-0.1122**
0.5384	**0.4777**	**0.6551**	**1.0702**	**2.7503**	**1.6035**	**292.7957**	**426.7051**	**696.0458**	**0.4184**	**0.3192**	**-0.4933**	**-0.3331**
1.0514	**0.9204**	**0.9816**	**1.0598**	**2.6463**	**0.2614**	**-168.6429**	**481.0178**	**699.6137**	**0.7932**	**0.5281**	**0.0422**	**0.2490**
1.1996	**0.3828**	**0.3051**	**0.8992**	**0.3007**	**2.7813**	**19.1353**	**512.0227**	**846.4106**	**0.9258**	**-0.4633**	**0.2479**	**-0.0982**
0.9853	**1.1363**	**0.0253**	**0.9337**	**1.7492**	**0.4994**	**-154.0110**	**441.2691**	**755.8039**	**0.5979**	**0.4384**	**-0.4191**	**0.2069**
0.2404	**1.0925**	**0.5007**	**1.0869**	**1.7416**	**1.2219**	**210.2072**	**463.3485**	**756.3984**	**0.2652**	**0.4417**	**-0.5352**	**0.0300**
1.4532	**1.3221**	**0.5614**	**0.0685**	**0.9575**	**1.2509**	**-386.8659**	**208.3977**	**816.0481**	**0.2041**	**-0.5077**	**0.4317**	**-0.2134**
1.1073	**1.5635**	**0.5591**	**1.1978**	**1.8635**	**2.1730**	**-397.0504**	**307.5946**	**746.9151**	**0.1361**	**0.6570**	**-0.1250**	**-0.2194**

0.2374	**0.6266**	**0.3783**	**0.5395**	**1.6120**	**2.0943**	**357.5256**	**307.4904**	**766.7015**	**0.3609**	**-0.0682**	**0.6795**	**0.0333**
0.1664	**0.2056**	**0.5058**	**1.0392**	**2.6594**	**1.7381**	**492.1575**	**164.2322**	**699.1224**	**0.0261**	**0.0737**	**-0.5991**	**-0.3681**
1.3422	**0.6045**	**0.4976**	**0.5565**	**0.5375**	**2.6047**	**-144.3998**	**472.8293**	**838.7206**	**0.8571**	**-0.4808**	**0.2904**	**-0.0301**
0.5320	**0.8677**	**0.9088**	**0.8192**	**0.0084**	**3.1050**	**84.0376**	**488.3809**	**849.9971**	**0.9996**	**-0.4999**	**0.0147**	**-0.0105**

Arm 6 (4dof model)

Dataset generated for arm: Though, this arm is four degrees of freedom, the dataset was also generated using the previous methods but, in this case, four joint angles were generated for the four revolute joints unlike the previous cases where six joint angles were generated.

Testing data set for arm : -

Table A-16 Testing data set for arm

10.2340	**13.0633**	**2.2979**	**1.3126**	**-0.0143**	**-0.0045**	**0.2627**
10.7149	**-12.5458**	**2.2693**	**0.9345**	**0.0565**	**-0.0120**	**-0.5276**
4.1820	**-15.7439**	**1.4422**	**1.0431**	**0.2121**	**-0.2197**	**-0.3670**
5.1425	**15.7440**	**2.2994**	**1.3019**	**0.0065**	**0.0049**	**0.2761**
7.3090	**15.5363**	**0.7075**	**1.0857**	**-0.3702**	**-0.1899**	**0.1795**
15.7798	**-3.9672**	**1.1642**	**1.1619**	**0.2760**	**-0.2175**	**0.1976**

16.9526	**1.4674**	**-1.5426**	**0.3680**	**-0.5144**	**-0.3912**	**-0.2202**
14.4589	**7.4416**	**-0.9675**	**0.5910**	**0.5705**	**-0.1725**	**0.2399**
1.2973	**16.8353**	**1.9232**	**1.3128**	**-0.1165**	**-0.1655**	**0.1679**
4.3563	**-15.7634**	**1.9249**	**0.9790**	**0.1576**	**-0.1262**	**-0.4687**
11.3336	**-11.8600**	**2.1091**	**0.6951**	**0.1436**	**0.0113**	**-0.5987**
1.7097	**16.4238**	**2.2800**	**1.2375**	**0.0334**	**0.0325**	**0.3391**
5.1493	**-15.5768**	**2.1125**	**1.3586**	**0.0498**	**-0.1338**	**-0.1349**
14.0437	**9.9244**	**-0.3770**	**0.7979**	**-0.3822**	**-0.3807**	**-0.2233**
16.9928	**1.4110**	**1.4060**	**1.0221**	**-0.2871**	**-0.1215**	**-0.3764**
16.0252	**-2.9644**	**-1.5229**	**0.5805**	**0.6069**	**-0.2172**	**0.0150**
16.8161	**-0.1873**	**2.0644**	**1.0812**	**-0.1483**	**-0.0602**	**-0.4268**
15.8458	**3.8664**	**-1.6496**	**0.4504**	**0.5934**	**-0.2778**	**0.1414**
10.5760	**12.6613**	**2.2681**	**1.3190**	**0.0558**	**0.0188**	**0.2482**
14.5802	**-7.3911**	**1.8866**	**1.1134**	**0.2120**	**-0.0007**	**-0.3810**
4.1364	**-15.7285**	**-1.0104**	**0.2835**	**0.5855**	**-0.1306**	**-0.3465**
15.7881	**5.0140**	**2.2996**	**1.2346**	**0.0052**	**0.0036**	**-0.3449**

Predicted data set for arm :-

Table A-17 Predicted data set for arm 6

Predicted_Joint1	Predicted_Joint2	Predicted_Joint3	Predicted_Joint4
0.4957	**2.3767**	**2.4519**	**0.1043**
0.4478	**1.5192**	**1.5730**	**2.1317**
1.5096	**2.2568**	**2.3037**	**2.0387**
2.8169	**1.4145**	**1.4635**	**0.3622**
0.9984	**1.1571**	**1.1921**	**0.4041**
1.1705	**0.4327**	**0.4612**	**0.3127**
2.2426	**1.4216**	**1.4660**	**0.4715**
0.5474	**2.1383**	**2.2081**	**2.5587**
1.3568	**0.7288**	**0.7656**	**1.9581**
0.5353	**1.9175**	**1.9822**	**2.4859**
2.3552	**1.2750**	**1.3065**	**1.1515**
2.4304	**2.5108**	**2.5996**	**0.0232**
1.8387	**1.3462**	**1.3788**	**2.0312**
0.1925	**1.4924**	**1.5489**	**0.3735**
0.2799	**1.1224**	**1.1632**	**1.7738**
2.6594	**2.0035**	**2.0592**	**0.7138**

1.7474	**1.4595**	**1.4948**	**2.3075**
1.8471	**2.3809**	**2.4423**	**0.7909**
1.9636	**1.6894**	**1.7172**	**1.9011**
1.5527	**0.2178**	**0.2538**	**2.0816**

Training data set for arm 6

Table A-18 Training data set for arm 6

Joint1	Joint2	Joint3	Joint4	X	Y	Z	q1	q2	q3	q4
2.1880	**2.4275**	**1.3985**	**2.5023**	**12.0268**	**11.0446**	**1.7820**	**0.9523**	**0.2344**	**-0.0371**	**0.4658**
0.8989	**1.4675**	**1.5873**	**2.4461**	**14.9819**	**-7.1882**	**2.2913**	**1.4056**	**-0.0220**	**0.0214**	**0.0717**
0.7127	**0.0627**	**1.5592**	**0.0058**	**14.9786**	**-8.4667**	**0.1175**	**0.3604**	**-0.4569**	**-0.1686**	**-0.4799**
1.7320	**2.9011**	**1.5531**	**0.5658**	**15.7851**	**3.8600**	**0.5873**	**1.0222**	**0.3602**	**0.2376**	**-0.2294**
2.2603	**0.5628**	**2.1007**	**0.3863**	**9.5999**	**13.8224**	**2.0421**	**1.3321**	**-0.0992**	**-0.1349**	**-0.1683**
1.3292	**0.6894**	**1.2333**	**1.6457**	**17.0834**	**1.6193**	**0.7928**	**1.1556**	**-0.3997**	**0.0638**	**-0.0483**
3.0812	**1.1675**	**0.1144**	**0.6192**	**-4.7159**	**16.5150**	**-0.6552**	**0.8129**	**-0.1889**	**-0.5344**	**0.1166**
2.1515	**0.5729**	**0.3330**	**2.1782**	**10.6319**	**13.3267**	**-1.4191**	**0.5129**	**-0.6357**	**0.0085**	**0.1732**
1.5109	**1.8975**	**0.2418**	**1.3176**	**16.4965**	**4.4586**	**1.2382**	**1.2251**	**-0.3381**	**-0.0328**	**-0.0967**
1.2319	**1.5883**	**0.7610**	**1.7565**	**16.9865**	**-0.5408**	**1.6151**	**1.3009**	**-0.2635**	**0.0708**	**-0.0499**

1.0781	**0.3059**	**2.7621**	**2.2568**	**16.0088**	**-4.4295**	**2.2938**	**1.4067**	**-0.0216**	**0.0145**	**0.0682**
2.2904	**1.7557**	**2.2300**	**2.0239**	**11.0099**	**12.0164**	**1.5281**	**1.0746**	**0.2871**	**0.0385**	**0.3569**
1.3778	**2.2533**	**1.6632**	**1.7506**	**16.2619**	**-1.2513**	**1.6432**	**1.3093**	**0.2626**	**-0.0495**	**-0.0043**
0.1875	**0.3318**	**1.7018**	**1.7694**	**8.4801**	**-14.8802**	**1.0269**	**0.9978**	**-0.2616**	**0.2645**	**-0.3357**
1.2505	**2.6705**	**1.5046**	**1.9329**	**15.8559**	**-3.6541**	**1.1771**	**1.2293**	**0.3292**	**-0.1169**	**0.0128**
2.3185	**0.2628**	**1.0485**	**0.9869**	**7.9876**	**15.2115**	**-0.5902**	**0.8593**	**-0.4408**	**-0.3462**	**0.0353**
0.5733	**2.0437**	**2.6568**	**0.8441**	**9.7955**	**-12.9588**	**0.0273**	**0.6547**	**0.4925**	**-0.0671**	**-0.3818**
0.5512	**0.3437**	**0.1155**	**1.8875**	**12.3342**	**-11.4338**	**-2.0617**	**0.3022**	**-0.5405**	**0.4266**	**-0.0554**
1.6699	**0.6308**	**2.5108**	**1.1605**	**15.7741**	**5.0860**	**2.3000**	**1.3971**	**0.0000**	**0.0000**	**-0.1096**
1.6708	**1.6119**	**2.6028**	**0.2166**	**15.9714**	**3.0888**	**1.0980**	**0.9842**	**0.2701**	**0.2403**	**-0.3566**