Solving Inverse Kinematics Using Artificial Neural Networks

By

Maaz Mohsin

170381

Supervisor:

Dr.Hasan Sajid

Robotics and Intelligent Machines Engineering

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

May, 2019

Solving Inverse Kinematic Using Artificial Neural Networks

Maaz Mohsin

170381

A thesis submitted in partial fulfillment of the requirements for the degree of

MS(Robotic and Intelligent Machine Engineering)

Thesis Supervisor:

Dr.Hasan Sajid

Thesis Supervisor's Signature: ________________________

Robotic and Intelligent Machine Engineering

SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,

ISLAMABAD

May, 2019

Declaration

I certify that this research work titled “*Solving Inverse Kinematics using Artificial Neural Networks”* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Maaz Mohsin

2016-Nust-MS-RIME-00000170381

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism. Turnitin report endorsed by Supervisor is attached.

Signature of Student

Maaz Mohsin

00000170381

Dr. Hasan Sajid

(Supervisor)

Copyright Statement

Acknowledgements

I am thankful to the only Creator Allah Subhana-Watala to have given me strength and knowledge to carry out this work at every step and for every new thought which **You** setup in my mind to improve it. Indeed **You** are the most beneficial and merciful. I am helpless and could not have any done anything without **Your** priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was **Your** will, so indeed none be worthy of praise but You.

I would also like to express special thanks to my supervisor **Dr.Hasan Sajid** for **his** help throughout my thesis and also for teaching the key concept of machine learning and deep learning. I can safely say that I haven't learned any other subject in such depth than the ones which he has taught. I am extremely obliged to him for his guidance, advice and the motivation he gave me in the time of need for the accomplishment of my research work I would also like to mention his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help I wouldn’t have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

I am profusely thankful to my beloved parents (**Mrs. & Mr. Dr. Abdul Qayyum Mohsin**) who raised me when I was not capable of walking and continued to support me throughout in every department of my life, specially my mother who believed in me and always was there to support and provide constructive criticism. I am also thankful to my sisters for their prayers, encouragement and moral support and also to my dear friend (**Raja Muhammad Saad Bashir**) who has always been there to help me.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my exceptional parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment.

Abstract

Finding the inverse kinematic solution is among one of the most important task in the field of robotics i.e. provided the position and orientation of the end factor compute the set of joint angles which would represent the desired position of end effector. For this purpose there exist many solutions such as geometric, algebraic and iterative as the complexity of the joint structure of the manipulator increases the computation becomes more difficult and time consuming. In order to elude cumbersome task of calculating manually a Neural Network based approach is proposed to solve Inverse kinematics. In this study inverse kinematics of six-degrees-of-freedom PUMA 560 was solved using ANN. Comprehensive study was conducted to find the finest configuration for this problem.

Key Words: *Artificial Neural Network, Inverse Kinematic, PUMA 560*

Table of Contents

List of Figures

List of Tables

CHAPTER :1 INTRODUCTION

In this era of Digitalization and machines data has grown exponentially and its near to impossible to hard code all those patterns and feature to perform analyses and understand to learn. We need machine learning tool that can learn through examples that we provide and perform detail feature analyses and moment recognitions for future and on the bases of those predict outcomes perform particular tasks .In order to design a robot's movements, we have to recognize the connection between the actuators that we can control and the robot's position. In inverse kinematics we are given the position of the end effector and we compute the joint configuration of the robot as shown figure 1.

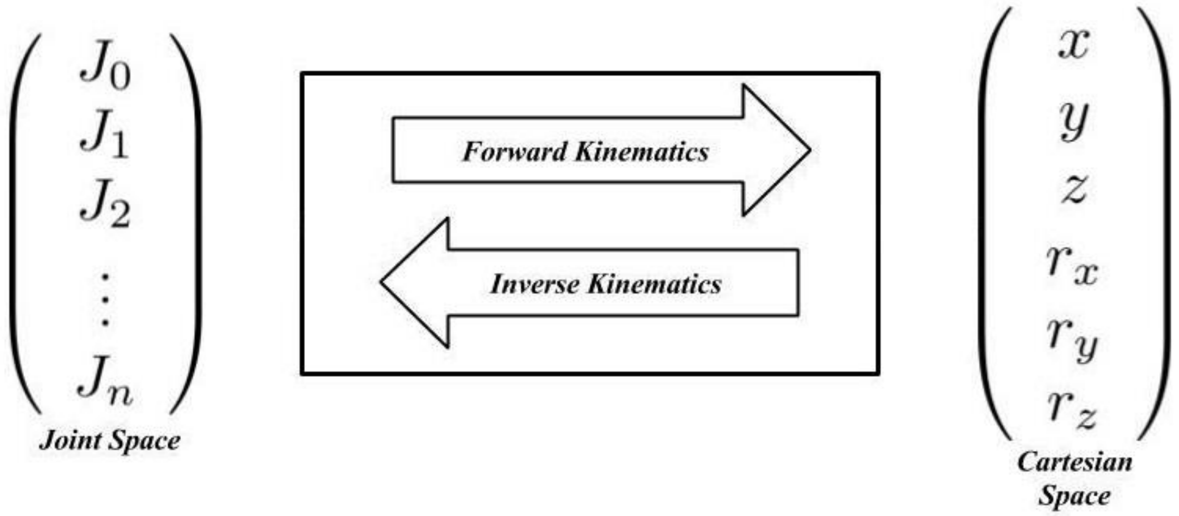


Figure 1:Kinematics Relation

1.1 Problem Definition

To calculate the inverse kinematics solution of a robot there are many methodology such as geometric, algebraic and iterative. As complexity of the joint structure of the manipulator increases the computation becomes more difficult and time consuming. In order to elude cumbersome task of calculating manually Neural Network based solution is proposed. Artificial Neural network mimic biological brain. So keeping in mind the capability of neural networks to solve complex tasks we find it suitable to solve the problem of inverse kinematics.

We chose PUMA560 which is a six degree of freedom robot. As we are using artificial neural network which are a part of supervised machine learning technique we have to provide a suitable data set for learning the non-linear function of inverse kinematics.

We illustrate the problem using the figure 2 for better understanding of the proposed problem. We will give input the position and rotation matrix value to the neural network. The trained neural network will give us the joints of against the provided input.

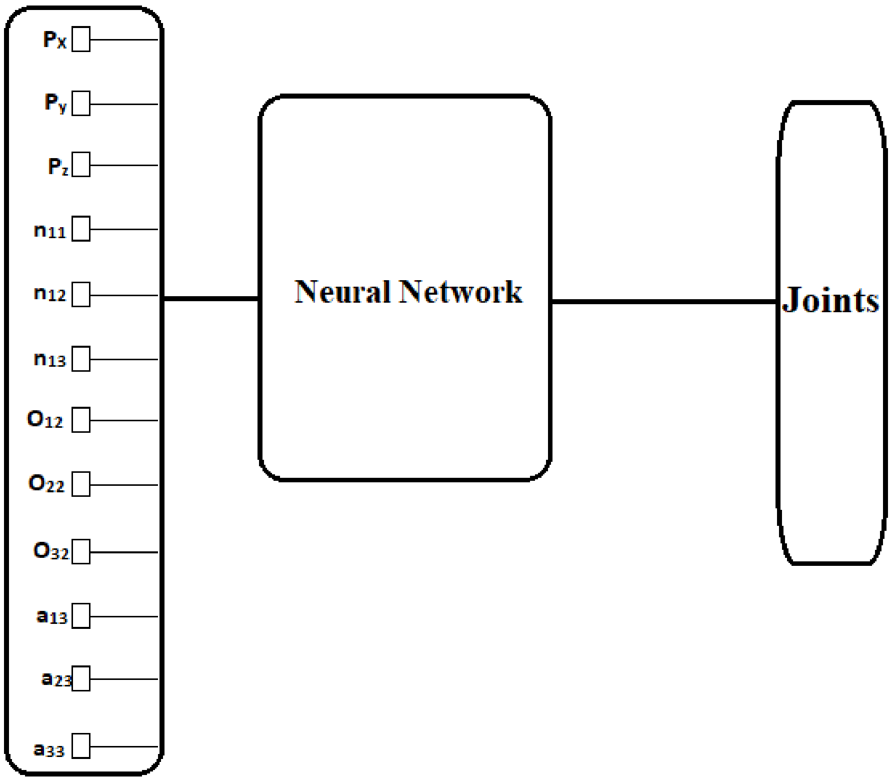


Figure 2 ANN Relationship With Input Output

To this end our contributions for this thesis are:

- Solving Inverse kinematics using learning
- Compile the data set for our robot i.e. Puma 560
- Devise the best configuration for solving the problem.

CHAPTER :2 LITRETURE REVIEW

Due to the strong capability of solving complex problems, Neural Networks solution have attracted the attention if many researchers in recent years [1-6]. Applying Neural Networks to the application of robotics in solving inverse kinematics give us an edge over the traditional methods. Neural Network trained solution will manifest high efficiency no matter what the kinematic structure is and have robust mapping capability but they have slow convergence speed, the weights might stuck in local minima and there is this difficulty of deciding the number of layers and neurons which comes with experience and experimentation. Researchers have used various methodologies to improve the results of the problem some of them are enlisted with detail below.

2.1 Kalman Filter

In literature many few researchers have used filters with neural networks to solve the problem of inverse kinematics. Hoai-Nhan Nguyen and colleagues [14] proposed a calibrated method for improvement in accuracy using extended kalman filter and an artificial neural network. Extended Kalman filter was used remove the geometric parameter errors. After that a 3 layer ANN with 40 neuron and tan-sigmoid activation function was used to model the non-geometric error sources.

2.2 Adaptive Learning Strategy

Hasan, A.T., et al. [2] controlled the motion of the robotic manipulator using an adaptive learning strategy. They used artificial neural network to surpass the problems in finding the inverse kinematics solution. In their approach, a network was trained to learn the joint configuration from the positions of end-effector

2.3 Inclusion of Current Joint

Almusawi, A.R.J., et al., [13] proposed that by including the current joint configuration of the robot significant improvement in the accuracy was observed. Robotic Arm Denso VP6242 was used to collect data set for training. Evaluation of the work was done by giving the desired position to the robot and it achieved the required joint configuration necessary to attain those particular points. The percentage error in position was 0.17, 0.36, 0.12 in position in X,Y and Z respectively.

2.4 Modular Neural Network

Oyama and colleague [8] presented modular neural network architectures for learning the inverse kinematics model. Their technique is established on DeMers' method, which includes a number of experts, an expert selector, an expert generator, and a feedback controller that can accommodate the nonlinearities in the kinematic system. In their method they have achieved an error of less than 10 mm in the hand position

2.5 4 DOF Surgical Robot

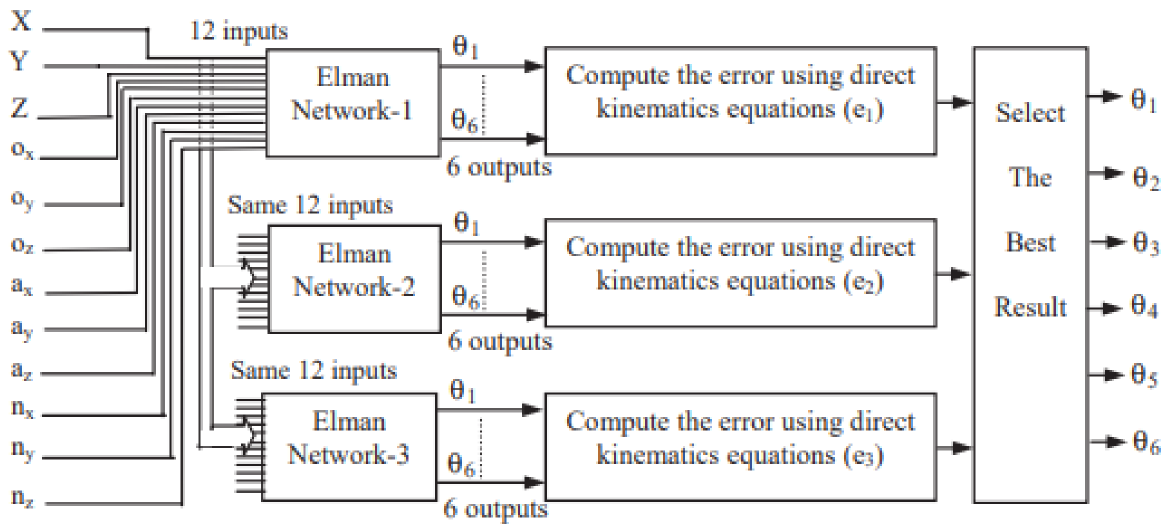
Nihat Çabuk et al., [16] presented a solution for Inverse kinematics problem of a four degree of freedom manipulator using ANN, manipulator was used to for lightning system that is used in surgery room ,The manipulator was designed in CAD and later used with Simulink. The author used a 2 layer neural network, changed the number of neurons in the hidden layer in the order of 12, 24,36,72,96,110,120 and computed the results .The enlisted error were around 1.5 m.

2.6 Optimization Algorithm

Mustafa Ayyıldız et al., [15] Used 4 different algorithms among which are genetic algorithm, particle swarm optimization, quantum particle swarm optimization and gravitational search algorithm.100 randomly selected workspace points were defined in order to calculate inverse kinematics which was calculated using these optimization algorithms. Comparison was performed between these 4 algorithms on the basis of execution time and position of the end effector. Results showed that quantum particle swarm optimization calculated the inverse kinematics in less time, iteration and average error.

2.7 Reliability-based Neural Network

Koker [11] proposed the study in which he used parallel neural networks to compute the inverse kinematics of the robots. He proposed a method in which he selected the best output from three parallel networks to obtain the most consistent solution for a 6-degree-of-freedom robot. Three networks were trained in order to reduce the error. RNN based model called Elman network was used to map the state space of the system. Three Elman neural networks used “sigmoid” as an activation function in order to deduce a reliable inverse kinematics solution. The detail pictorial



$$F(X, Y, Z, o_x, o_y, o_z, n_x, n_y, n_z, a_x, a_y, a_z) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$$

representation of their methodology is represented below in figure 3

Figure 3 Elman Filter Architecture

2.8 Jacobian Solution

Hasan et al., [12] used Artificial Neural Network to overcome singularities and uncertainties. They used a 6 degree of freedom robotic manipulator and managed to achieve a low error rate which was for joints 1–6 were 0.915%, 0.135%, 0.57%, 4.79%, 4.81%, and 1.11%, respectively. They also calculated angular velocities with angular position .The error for angular velocities 1-6 were 1.265%, 2.02%, 1.205%, 1.41%, 1.15%, and 1.175% respectively. They also enlisted their error in Cartesian position which was 3.34%, 6.72%, and 0.35% in P_X, P_Y, P_Z respectively.

2.9 Committee Machine

Koker Hasan et al., [4] proposed a neural network based committee machine to solve the 6 DOF redundant robotic manipulator. The committee machine consisted of ten neural networks. All neural networks were trained to solve the problem independently. In this work the researchers stated that using six networks simultaneously reduces the error range from (5.76 - 13.44) mm to (0.39-0.74) mm they also mentioned in their finding that using more the 6 networks simultaneously had no significant effect on the output.

2.10 Particle Swarm Optimization

B. Durmuş et al., [18] calculated Inverse kinematics problem using particle swarm optimization. As it is evident from the name PSO (Particle Swarm Optimization) is an optimization algorithm which depends on the behavior of swarm. Like genetic algorithm it starts by generating an initial population. Random selection of each individual is performed from the search space. After processing each individual is evaluated on the basis of a fitness function which is predefined according to the problem.

CHAPTER :3 DATA SET

For the purpose of calculating inverse kinematics we chose a 6 DOF PUMA 560 and collected different data points in the workspace to construct the data set.

3.1 Kinematics Of Puma560

In order to define the motion of the joints of a robot regardless of the forces causing it kinematics is used. Kinematics is usually divided into two parts i.e. forward Kinematics and Inverse Kinematics.

Forward Kinematics is the process of computing the orientation and position of end-effector from the set of joint angles .In other words transformation from joint space description into Cartesian space description is known as forward Kinematics. Inverse Kinematics is the process of computing the set of joint angles from the position and orientation of end-effector.

To compute Inverse Kinematics there exist many solutions such as geometric, algebraic and iterative as the complexity of the joint structure of the manipulator increases the computation becomes more difficult and time consuming. In order to elude cumbersome task of calculating manually a Neural Network based approach is more appropriate to solve Inverse kinematics.

In order to design a robot’s movements, we have to recognize the connection between the actuators that we can control and the robot’s position. In our study we have computed the inverse kinematics of puma560 robot which has 6 DOF, the coordinate frame assignment has been performed and assigned to the links which can be seen in figure 4**.** The DH (Denavit-Hartenberg) parameters of the robot are shown in table 1.The joint ranges of puma560 are shown in table 2.

Link	theta	D	A	Alpha	Offset
1	q1	0	0	0	0
2	q2	0	0	1.5708	0
3	q3	0.15005	0.4318	0	0
4	q4	0.4318	0.0203	-1.5708	0
5	q5	0	0	1.5708	0
6	q6	0	0	-1.5708	0

Table 1 DH Parameters Of PUMA 560

Links	q_min	q_max
A_1	-2.7925	2.7925
A_2	-0.7854	3.9270
A_3	-3.9270	0.7854
A_4	-1.9199	2.9671
A_5	-1.7453	1.7453
A_6	-4.6426	4.6426

Table 2 Joint Limits of Puma 560

The Homogeneous transformation matrix is used to represent the position and orientation of the end effector with respect to the base of the robot. The homogeneous transformation matrix for all joints for 0 to 6 is represented in equation 1

$$\boldsymbol{T_6^0} = \begin{vmatrix} \boldsymbol{R_6^0} & \boldsymbol{P_6^0} \\ \boldsymbol{0} & \boldsymbol{1} \end{vmatrix} \qquad \mathbf{1}$$

R_6^0 Represents the rotational matrix 3 X 3 and P_6^0 represents the position vector of end effector. This robot has six links thus we calculate transformation matrix for each link and then the homogeneous transformation matrix is calculated by multiplication of each transformation.

$$\boldsymbol{A_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5} = \begin{bmatrix} \boldsymbol{n11} & \boldsymbol{O12} & \boldsymbol{a13} & \boldsymbol{Px} \\ \boldsymbol{n21} & \boldsymbol{O22} & \boldsymbol{a23} & \boldsymbol{Py} \\ \boldsymbol{n31} & \boldsymbol{O32} & \boldsymbol{a33} & \boldsymbol{Pz} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix} \qquad \mathbf{2}$$

Here “n”, “o”, and “a” defines the orientation of the end effector in the “X”, “Y” and “Z” respectively. The Cartesian coordinates are calculated using the combination of joint angles lying in the ranges shown in table 2 .Using these joint angle values a data set is constructed which is explained in section below.

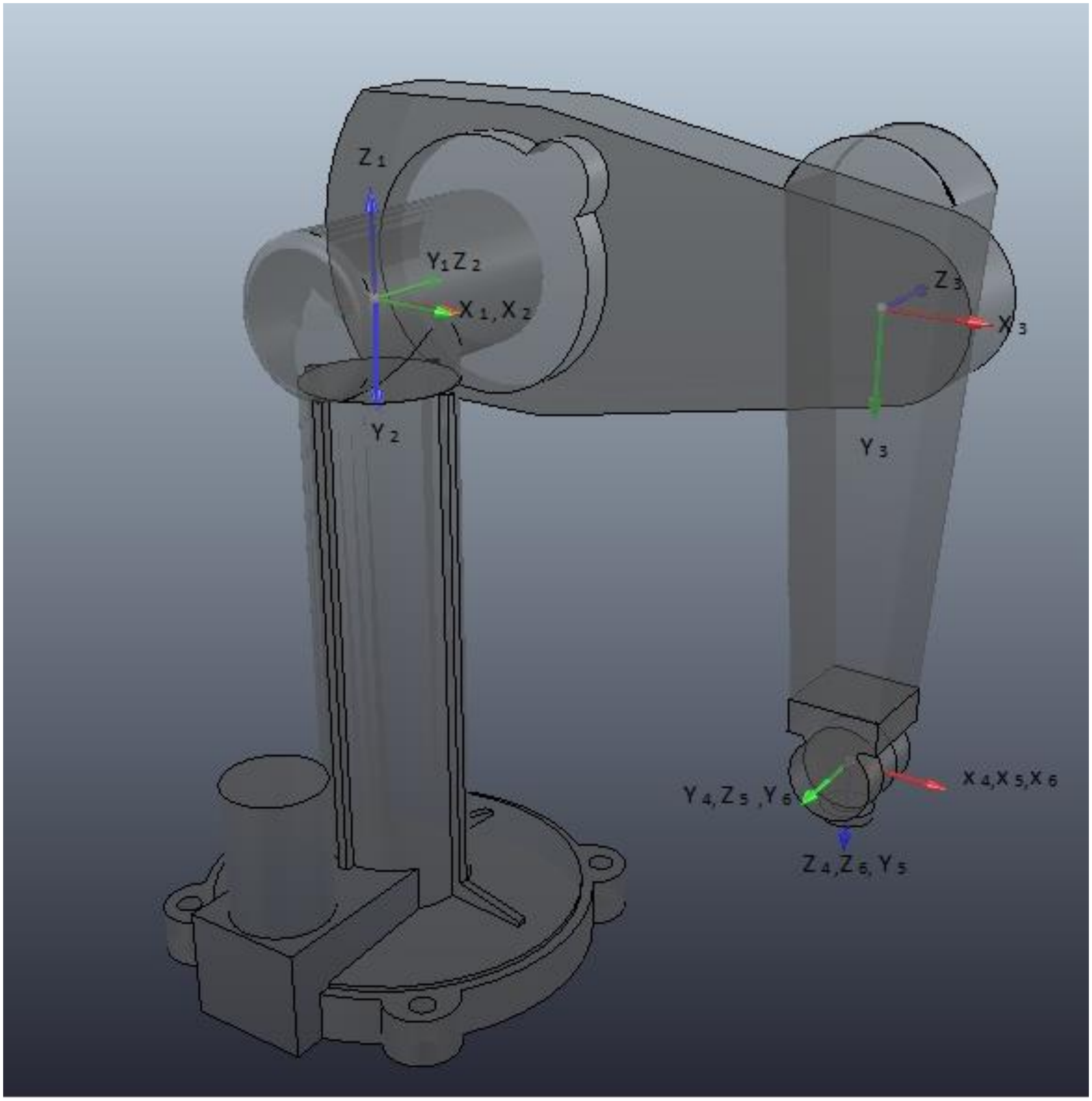


Figure 4 Puma 560 Assignment of frame

3.2 Data Set Collection

In order to map the function by neural network it is important to generate a good dataset. For this study the data set was created using MATLAB software. Puma 560 model from Robotic toolbox by peter corke [17] was used which is shown below in table1for acquiring the set of Cartesian space and joint configuration. Random data sampling technique was used to obtain the values of joint angles keeping in mind the configuration of robot mentioned in table 2.To compute the forward kinematics "fkine" function was used and for Inverse kinematics "ikine6s" function of robotic toolbox was used. To ensure the data consistency duplicate values were removed from the data set. The data set consisted of 50000 unique data points which contained the Position orientation and joint configuration of the robot. The input for ANN is shown in table 3 which has the position and orientation and the output against this input is shown in table 4. Which consist of the joint configuration of each link. This data set was divided in to training set, test set and validation set with the ratio is shown in figure 5

P_x	P_y	P_z	n_{11}	n_{12}	n_{13}	O_{12}	O_{22}	O_{32}	a_{13}	a_{23}	a_{33}
-0.65	-0.21	0.53	-0.45	-0.89	0.01	-0.65	0.34	0.68	-0.61	0.3	-0.74
0.57	-0.07	0.47	-0.04	0.22	0.97	-0.22	-0.95	0.21	0.97	-0.21	0.09
-0.16	-0.48	0.39	-0.27	-0.21	-0.94	0.67	0.66	-0.33	0.69	-0.72	-0.04
0.05	-0.72	0.47	0.65	0.16	0.74	-0.74	0.32	0.59	-0.15	-0.93	0.33
-0.08	0.22	0.72	-0.48	-0.02	-0.88	-0.87	-0.15	0.48	-0.14	0.99	0.05
-0.17	-0.38	0.26	0.86	-0.21	0.46	-0.34	-0.91	0.23	0.37	-0.36	-0.86
0.28	0.23	0.01	-0.24	0.45	0.86	0.09	-0.87	0.48	0.97	0.19	0.17
-0.17	-0.34	0.43	0.23	0.92	0.31	-0.79	0.36	-0.5	-0.57	-0.13	0.81
-0.04	-0.65	-0.4	-0.06	-0.78	0.62	-0.99	0.13	0.07	-0.14	-0.61	-0.78
0.39	-0.03	0.64	0.61	-0.58	-0.55	0.68	0.02	0.73	-0.41	-0.82	0.4
-0.25	0.16	-0.58	0.67	-0.6	0.44	-0.65	-0.76	-0.05	0.37	-0.25	-0.9

Table 3 Input For The Network

Θ_1	Θ_2	Θ_3	Θ_4	Θ_5	Θ_6
0.09	2.28	-1.15	-0.37	1.35	-1.05
2.76	1.85	-0.37	1.61	0.18	-1.8
0.95	1.69	0.02	1.56	1.38	1.09
1.43	2.34	-1.12	-1.01	0.02	0.34
4.39	1.28	-0.45	0.67	0.8	-3.12
0.79	1.55	0.48	0.59	1.18	-2.3
3.4	1.95	0.83	-3.08	1.38	2.61
0.71	1.39	0.22	-2.82	1.06	-2.69
1.28	-3.1	-0.49	2.94	0.23	-3.11
2.69	1.54	-0.43	-1.95	1.4	-1.07
5.19	-2.73	0.04	-1.42	0.21	-2.06

Table 4 Ground Truth For The Network

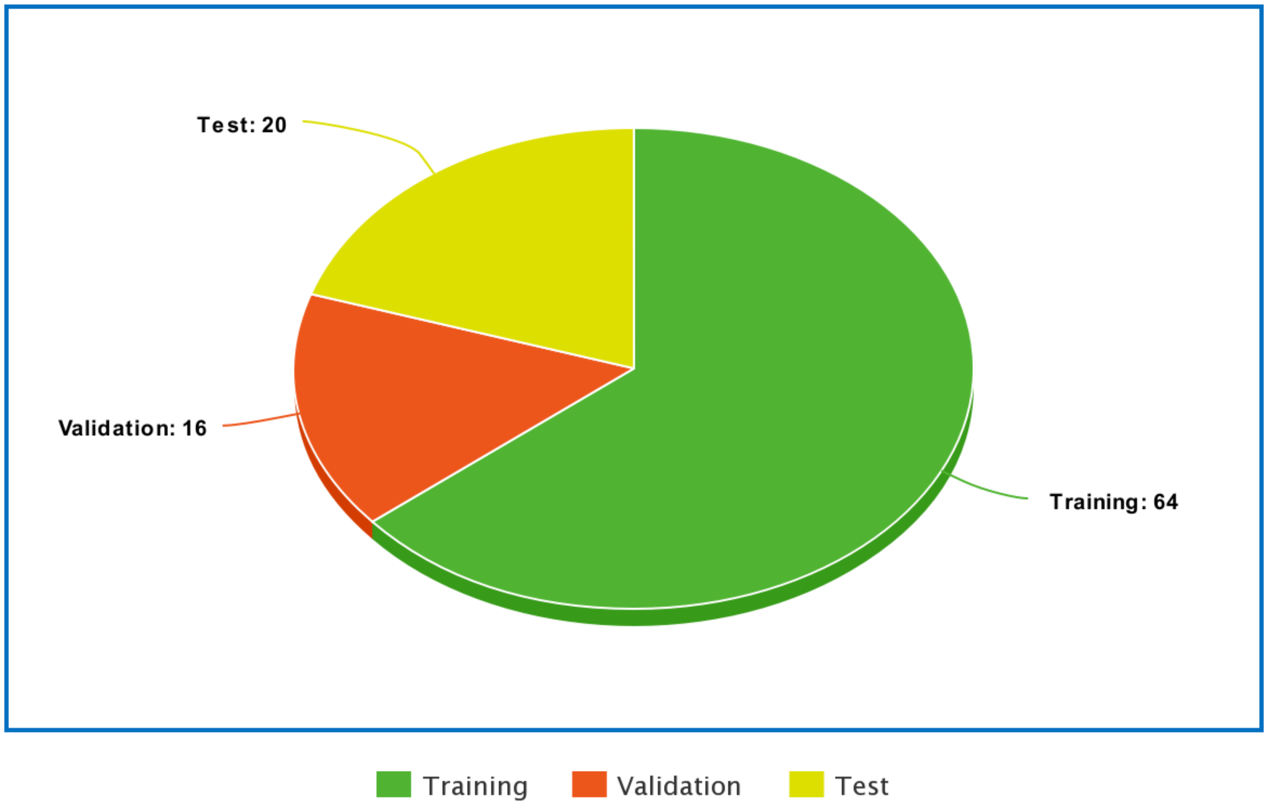


Figure 5 Data Set Distribution

CHAPTER :4 PROPOSED METHODOLOGY

4.1 Methodology

The aim of this work is to train a set of neural networks to deliver inverse kinematics solution and conclude which configuration is well suited. Given the orientation and position of the end-effector the neural network should be able to output the set of joint angles of the robot. These joint angles would be responsible to move the end-effector of robot to the desired position. So the below mentioned techniques will have the position and orientation as input and the output of the neural network will be joint angle (or set of joint angle).As there is no unique solution for the inverse kinematics and mathematical derivation are time consuming and complex so it is better to find out the solution through neural network. In our work we used an ANN to solve the problem of inverse kinematics. We used two techniques and then compared the results of each technique and with the results shown in literature.

4.2 First Configuration

In this configuration we computed the joint angle explicitly using a dedicated network for each joint. The network consisted of 12 inputs and one output. The configuration of ANN is shown in figure 6.

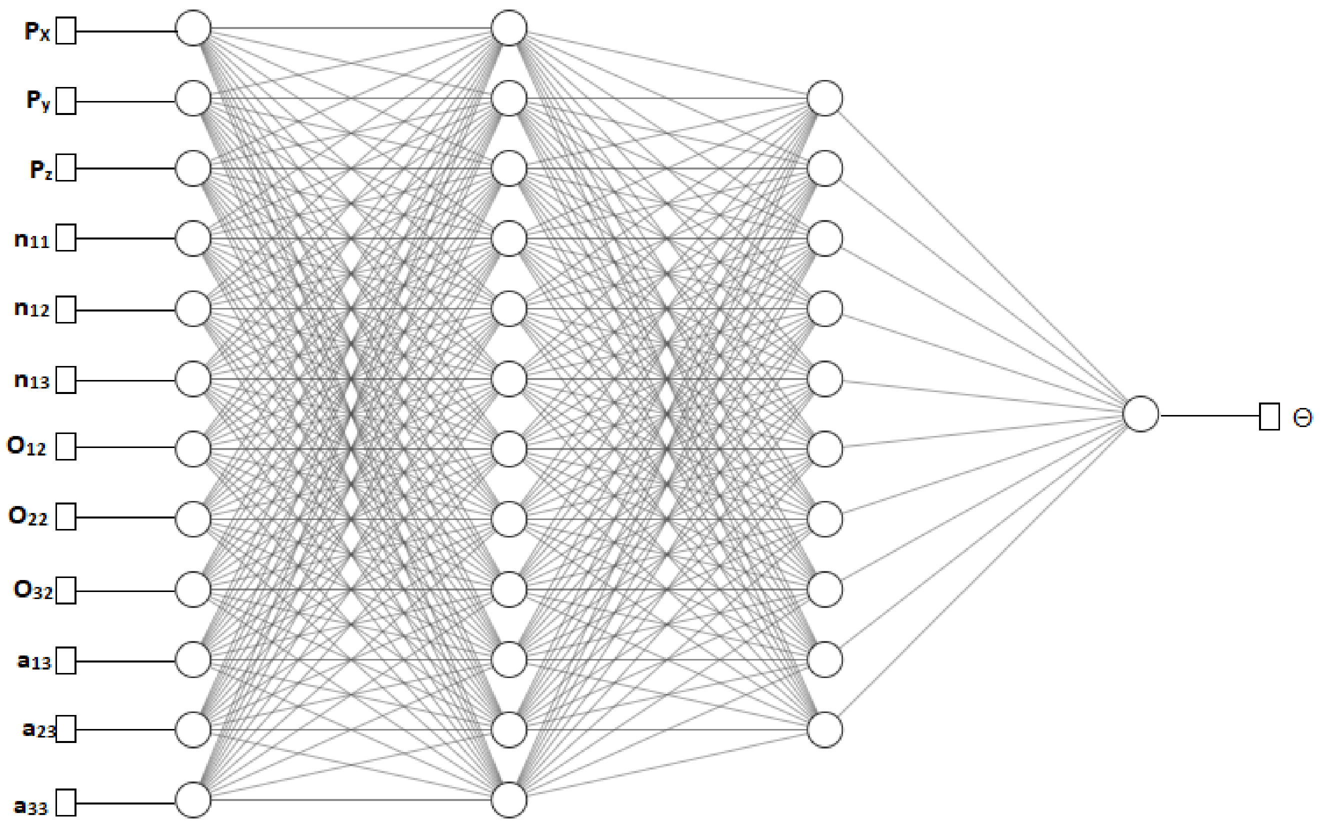


Figure 6 Configuration With Single Joint Output

For experimentation purpose we used two configuration one with three layers and second with four layers. Rectified Linear Unit (RELU) was used as activation function in these layers. We used 50 neurons in the first layer and then increased it with the multiple of two for the next layers. The last layer consisted of only one neuron that represented our output i.e. Theta. The layer details against the networks are shown in table6.

Layers/Networks	Layer 1	Layer 2	Layer 3	Layer4
Network 1	50	100	**Output**	**-**
Network 2	50	100	200	**Output**

Table 5 Network Configuration For Single Joint Output

4.3 Second configuration

In this configuration all joints were calculated simultaneously. The network consisted of 12 inputs and 6 outputs representing the six joints of Puma 560 as shown in figure 7 .For gaining better results we increased the number of layers starting from 3 and going on to 7.The number of neuron in the layer started from 50 and increased with the multiple of 2 in the next layer. Different layer configuration were used which are shown in table 7. Rectified Linear Unit (RELU) was used as activation function.

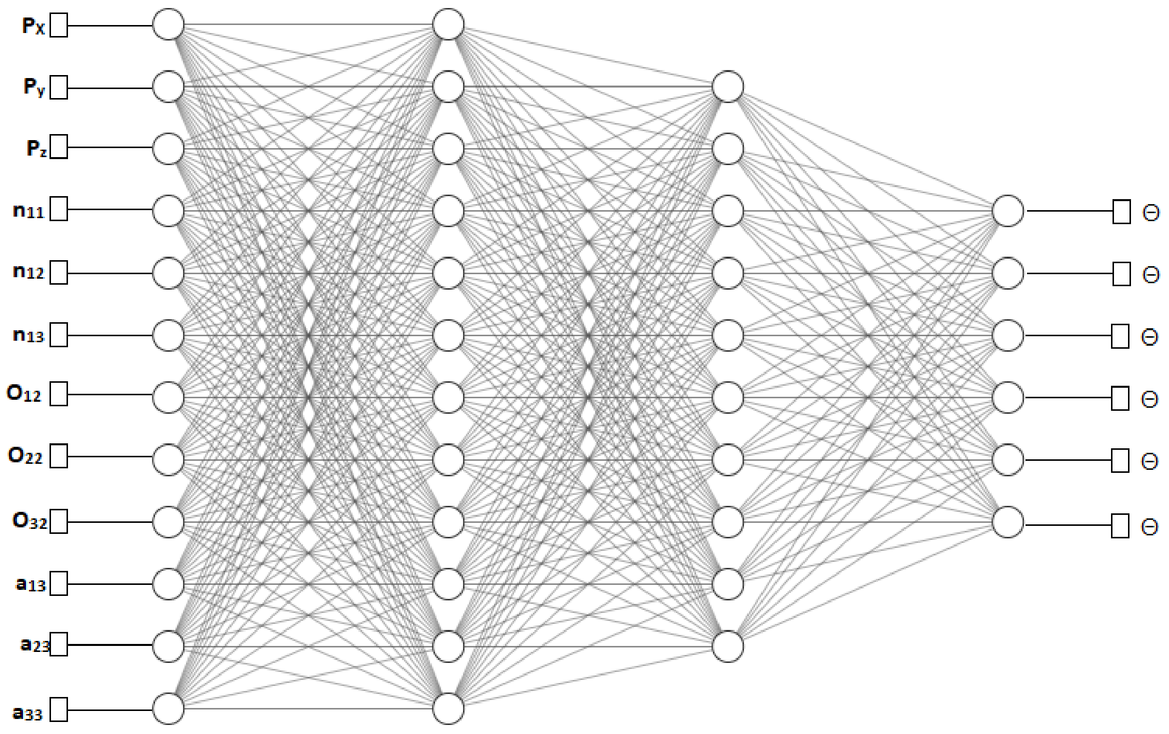


Figure 7 Configuration With Six Joint As Outputs

Layers/Networks	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
Network 1	50	100	**Output**	**-**	**-**	**-**	**-**
Network 2	50	100	200	**Output**	**-**	**-**	**-**
Network 3	50	100	200	400	**Output**	**-**	**-**
Network 4	50	100	200	400	600	**Output**	**-**
Network 5	50	100	200	400	600	800	**Output**

Table 6 Network Configuration For Six Joint Output

4.4 Experimental Evaluation

In order to come up with some conclusive results we focused on the error of actual and predicted value. For this purpose of error calculation we calculated the difference of actual and predicted value by the network using equation 3.

$$\boldsymbol{Error} = (\boldsymbol{Actual\ value}) - (\boldsymbol{Pridicted\ value}) \qquad \mathbf{3}$$

4.5 First Configuration

Focusing on a single theta as an output ANN works quite efficiently. We got good results in just the second configuration of network which only had 4 layers out of which one was the output layer. After training the network for 50000 epochs we evaluated the predicted joints and the difference of the joint values of predicted and actual were plotted as an error in figure 8. Most of the researcher focused on minimizing the MSE (Mean square Error) but we focused on minimizing the position error, for this purpose we computed the forward kinematics and attained the position values in X,Y,Z respectively. After getting the position values we took the difference with the actual values and plotted the histogram representing the error in figure 9.

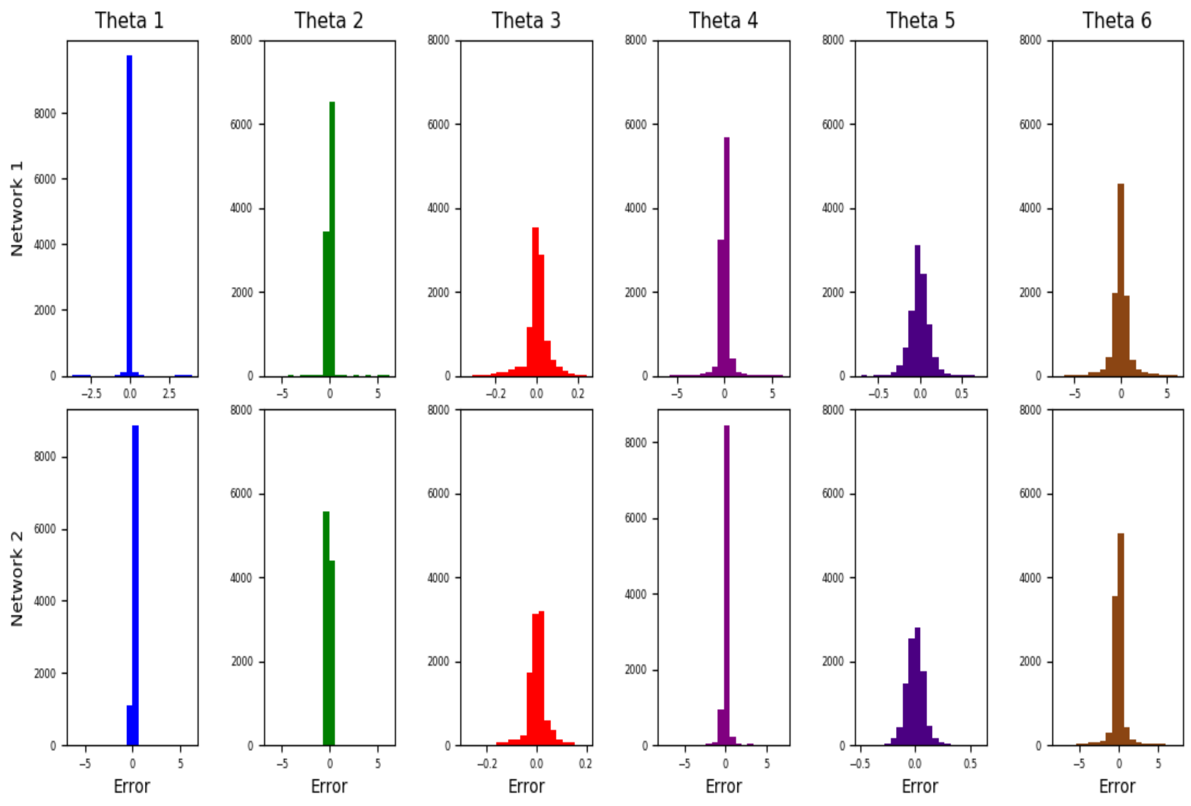


Figure 8 Joint Error Configuration 1

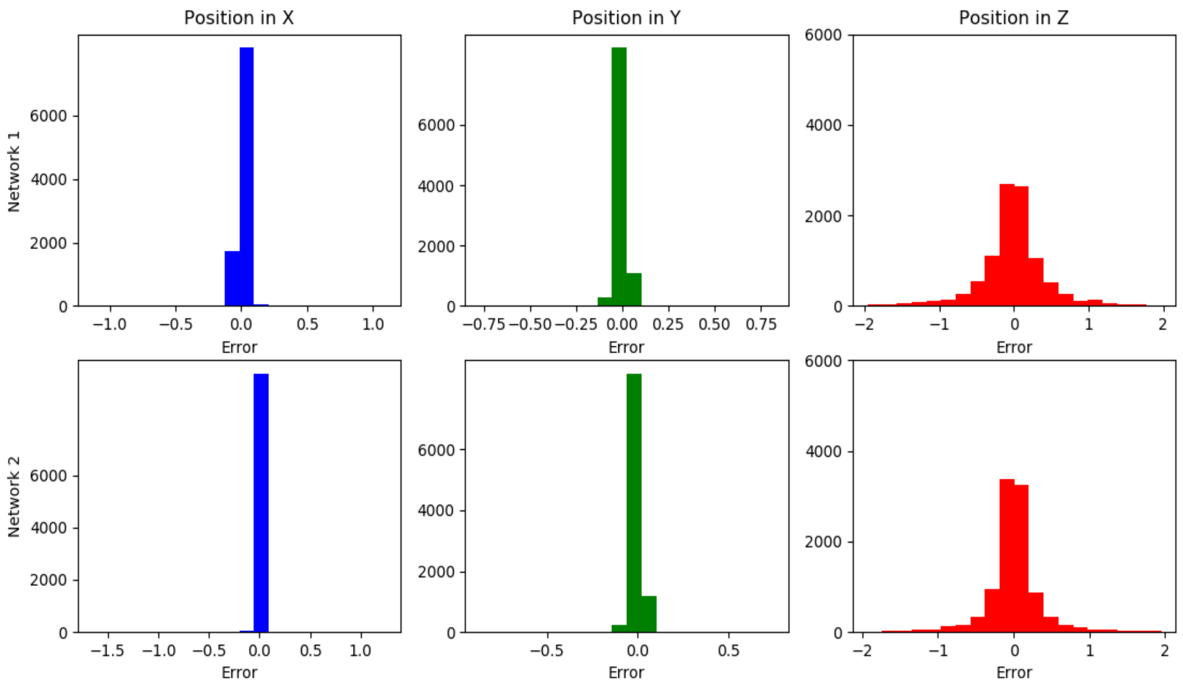


Figure 9 Position Error of configuration 1

4.6 Second Configuration

In the network giving us six joints cumulatively a comprehensive study was conducted using our dataset. Starting from network 1 results were very poor but with the increase of layers and number of neurons network started to predict somewhat near to the actual output we computed the joint error using equation 3 in the predicted values of all the joints, the error and its spread is represented against each network for every joint in figure 7**.**As it is apparent from the figure that by increasing the number of layers and neurons the error is reduced and in the last network predict values with state of the art accuracy. After calculating the results in joint space we examined the results in Cartesian space. Forward kinematic of the predicted values were calculated using the robotics toll box, after attaining the values in Cartesian space we calculated the error by taking the difference of the actual and predicted values using the same equation used prior in joint error calculation. Error in position of X ,Y and Z respectively are represented in figure 8 .

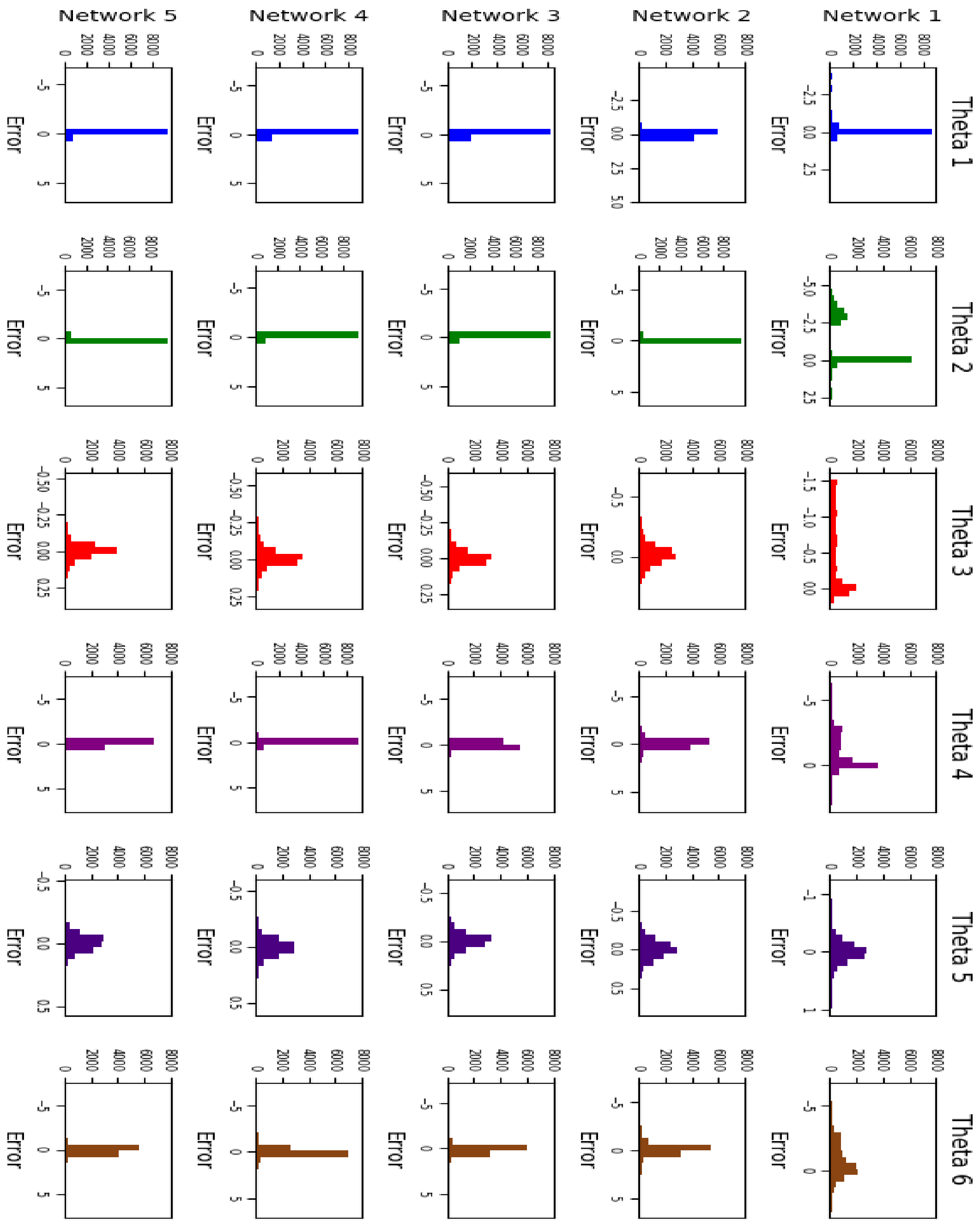


Figure 10 Joint Error Configuration 2

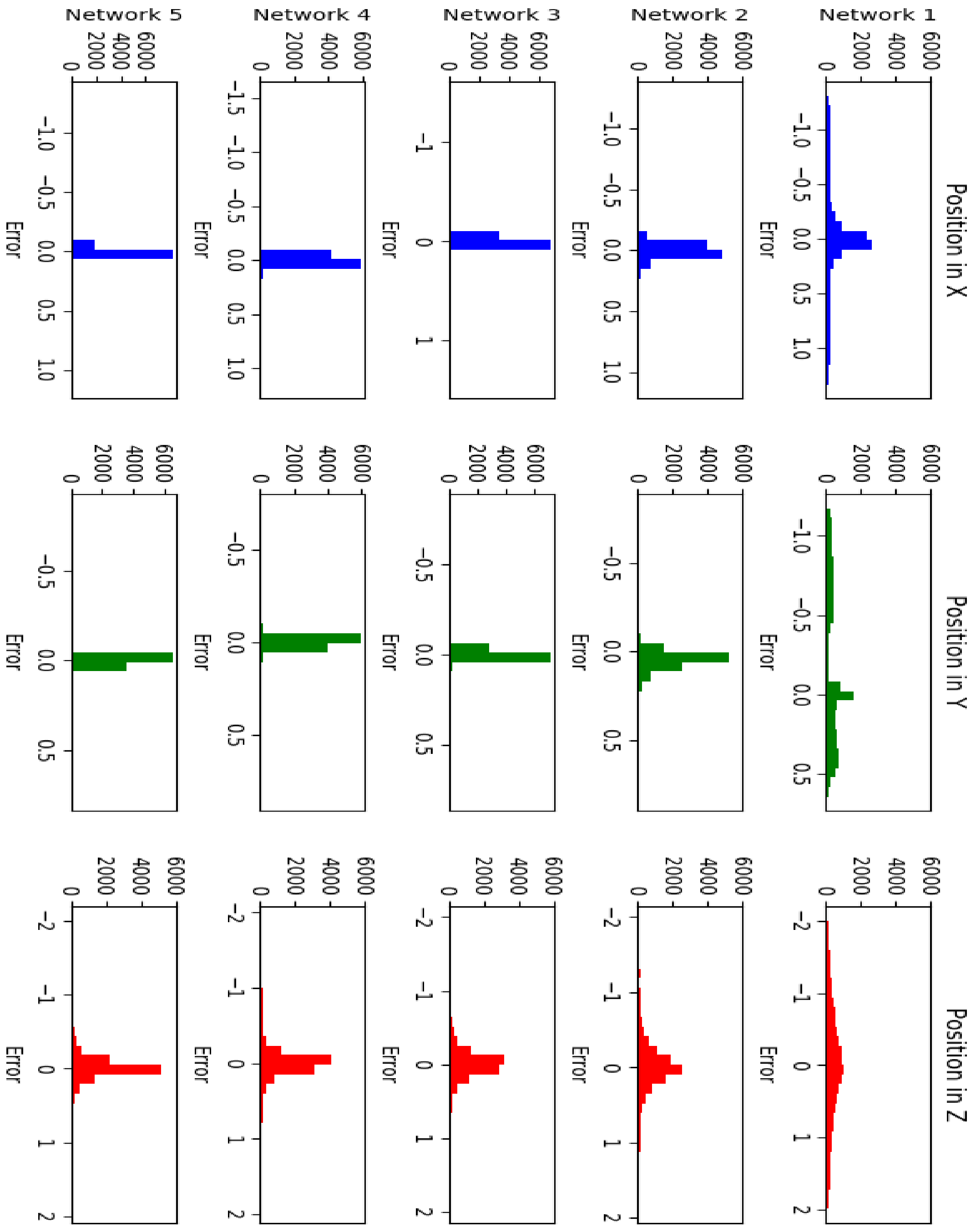


Figure 11 Position Error Configuration 2

4.7 Training Details

We split our dataset into a training set and a test set, containing 32000 and 10000 unique data points in our workspace respectively. We train over the data set for 50000 epochs for each network proposed. We used a learning rate of 0.001 for the training. For the loss function, we used the MSE(Mean Squared Error) loss which is given as:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \qquad \mathbf{4}$$

Where n is the total number of training samples y is the ground truth value and $\hat{y}$ is the predicted value of our network. Stochastic Gradient Descent (SGD) optimizer with learning rater of 0.001, decay=1e^{-6} and momentum value as 0.9

.

4.8 Rectified Linear Unit

To map the non-linear function of inverse kinematics we used Relu as an activation function. Rectified Linear Units (ReLUs) have gained admiration as a non-linearity in recent years. Using rectified linear units leads to faster training times over sigmoid, it also aids in stabilizing the gradients, resulting in more stable training. The graphical representation is shown in figure 3.4 and it is mathematically given as:

$$\boldsymbol{f(x) = \max(0, x)} \qquad \mathbf{5}$$

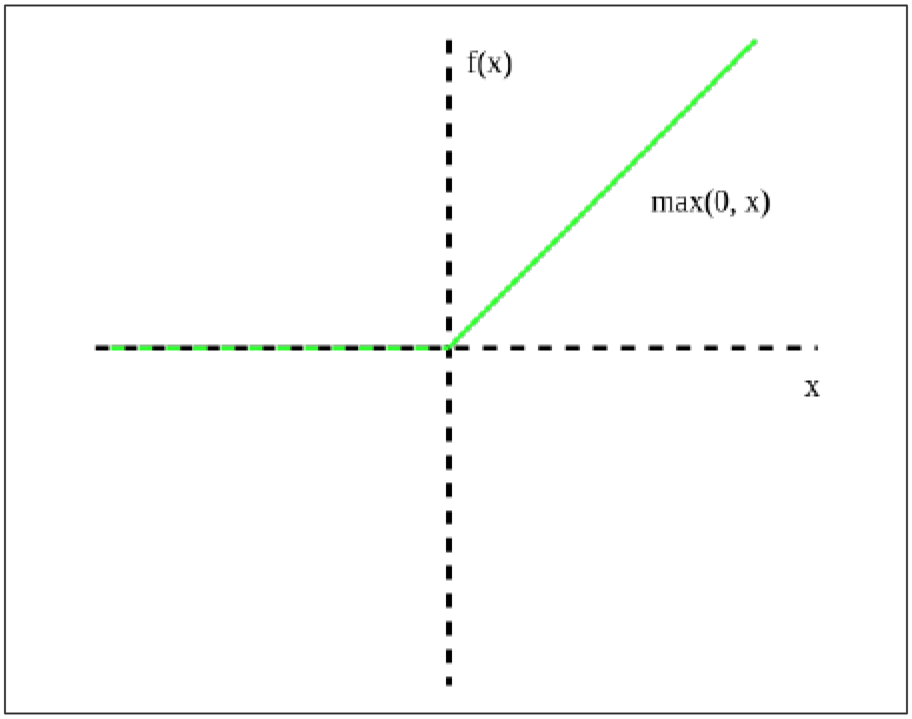


Figure 12 RELU

CHAPTER :5 CONCLUSION

In this thesis we introduced the problem of solving inverse kinematics using neural network, dataset for puma 560 with prior mentioned constraints of manipulators and baseline networks with different configurations also which configuration will be efficient to solve the problem.
We hope that this effort overlays way for forthcoming work in the domain and we propose it as an initial step towards autonomous robotics using machine learning. Below, we outline considerations for potential future work in this domain.

5.1 Future Work

AI (Artificial Intelligence) and technologies associated to it will emerge in every industry in coming future .Machines with the capability of perceiving and taking decision is one of the AI’s major gifts to the modern civilization. With the development in technologies in terms of hardware and software gates to the self – propelled machines are opened which has impacted our operative rules.
In the recent years, digital assistants, autonomous driving, robotic industrial staff, and smart cities have demonstrated that intelligent machines are possible. AI has transformed most industry sectors like trading, manufacturing, financing, healthcare, and media and continues to impact different industries.
Neural networks can be used in motion planning and trajectory generation. In motion planning few researchers have used PCNN (Pulse Couple Neural Network) but we would recommend to use RNN (Recurrent Neural Networks) as they maintain historical data i.e. at each step the previous state is known which would help in determining the next state so it would be beneficial in decision making and object detection. Using RNN would result in better and efficient solution to these problems.

APPENDIX A

The details of architecture of the model build in keras are shown below.

Combined Joints

- **Network with 3 layers**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                650
_________________________________________________________________
dense_2 (Dense)              (None, 100)               5100
_________________________________________________________________
dense_3 (Dense)              (None, 6)                 606
=================================================================
Total params: 6,356
Trainable params: 6,356
Non-trainable params: 0
```

- **Network with 4 layers**

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
dense_1 (Dense)           (None, 50)            650
_________________________________________________________________
dense_2 (Dense)           (None, 100)           5100
_________________________________________________________________
dense_3 (Dense)           (None, 200)           20200
_________________________________________________________________
dense_4 (Dense)           (None, 6)             1206
=================================================================
Total params: 27,156
Trainable params: 27,156
Non-trainable params: 0
_________________________________________________________________
```

- **Network with 5 layers**

```
_________________________________________________________________
Layer (type)            Output Shape            Param #
================================================================
dense_1 (Dense)           (None, 50)            650
_________________________________________________________________
dense_2 (Dense)           (None, 100)           5100
_________________________________________________________________
dense_3 (Dense)           (None, 200)           20200
_________________________________________________________________
dense_4 (Dense)           (None, 400)           80400
_________________________________________________________________
dense_5 (Dense)           (None, 6)             2406
=================================================================
Total params: 108,756
```

```
Trainable params: 108,756
Non-trainable params: 0
_________________________________________________________________
```

- **Network with 6 layers**

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_1 (Dense)             (None, 50)              650
_________________________________________________________________
dense_2 (Dense)             (None, 100)              5100
_________________________________________________________________
dense_3 (Dense)             (None, 200)              20200
_________________________________________________________________
dense_4 (Dense)             (None, 400)              80400
_________________________________________________________________
dense_5 (Dense)             (None, 600)              240600
_________________________________________________________________
dense_6 (Dense)             (None, 6)              3606
=================================================================
Total params: 350,556
Trainable params: 350,556
Non-trainable params: 0
_________________________________________________________________
```

- **Network with 7 layers**

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_1 (Dense)             (None, 50)              650
_________________________________________________________________
dense_2 (Dense)             (None, 100)              5100
_________________________________________________________________
dense_3 (Dense)             (None, 200)              20200
_________________________________________________________________
dense_4 (Dense)             (None, 400)              80400
_________________________________________________________________
dense_5 (Dense)             (None, 600)              240600
_________________________________________________________________
dense_6 (Dense)             (None, 800)              480800
_________________________________________________________________
dense_7 (Dense)             (None, 6)               4806
=================================================================
Total params: 832,556
Trainable params: 832,556
Non-trainable params: 0
_________________________________________________________________
```

Single Output

- **Network with 3 layer**

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_1 (Dense)            (None, 50)              650
_________________________________________________________________
dense_2 (Dense)            (None, 100)             5100
_________________________________________________________________
dense_3 (Dense)            (None, 6)              606
=================================================================
Total params: 6,356
Trainable params: 6,356
Non-trainable params: 0
_________________________________________________________________
```

- **Network with 4 layer**

```
_________________________________________________________________
Layer (type)              Output Shape            Param #
=================================================================
dense_1 (Dense)             (None, 50)              650
_________________________________________________________________
dense_2 (Dense)             (None, 100)             5100
_________________________________________________________________
dense_3 (Dense)             (None, 200)             20200
_________________________________________________________________
dense_4 (Dense)             (None, 6)              1206
=================================================================
Total params: 27,156
Trainable params: 27,156
Non-trainable params: 0
_________________________________________________________________
```

REFERENCES

1. Aggarwal, L., K. Aggarwal, and R.J. Urbanic, *Use of Artificial Neural Networks for the Development of an Inverse Kinematic Solution and Visual Identification of Singularity Zone(s).* Procedia CIRP, 2014. **17**: p. 812-817.
2. Hasan, A.T., et al., *An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator.* Advances in Engineering Software, 2006. **37**(7): p. 432-438.
3. Karlik, B. and S. Aydin, *An improved approach to the solution of inverse kinematics problems for robot manipulators.* Engineering Applications of Artificial Intelligence, 2000. **13**(2): p. 159-164.

4. Köker, R., T. Çakar, and Y. Sari, *A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators.* Engineering with Computers, 2014. **30**(4): p. 641-649.

5. Shi, Q. and J. Xie. *A research on inverse kinematics solution of 6-DOF robot with offset-wrist based on Adaboost neural network*. in *2017 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2017.

6. Toshani, H. and M. Farrokhi, *Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A Lyapunov-based approach.* Robotics and Autonomous Systems, 2014. **62**(6): p. 766-781.

7. Tejomurtula, S. and S. Kak, *Inverse kinematics in robotics using neural networks.* Information Sciences, 1999. **116**(2): p. 147-164.

8. Oyama, E., et al. *Inverse kinematics learning by modular architecture neural networks with performance prediction networks*. in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. 2001.

9. Mayorga, R.V. and P. Sanongboon, *Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An Artificial Neural Network approach.* Robotics and Autonomous Systems, 2005. **53**(3): p. 164-176.

10. Bingul, Z., H.M. Ertunc, and C. Oysu. *Applying Neural Network to Inverse Kinematic Problem for 6R Robot Manipulator with Offset Wrist*. in *Adaptive and Natural Computing Algorithms*. 2005. Vienna: Springer Vienna.

11. Köker, R., *Reliability-based approach to the inverse kinematics solution of robots using Elman's networks*. Vol. 18. 2005. 685-693.

12. Hasan, A.T., et al., *Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations.* Advances in Engineering Software, 2010. **41**(2): p. 359-367.

13. Almusawi, A.R.J., et al., *A New Artificial Neural Network Approach in Solving Inverse Kinematics of Robotic Arm (Denso VP6242).* Computational Intelligence and Neuroscience, 2016. **2016**: p. 10.

14. Nguyen, H.-N., J. Zhou, and H.-J. Kang, *A calibration method for enhancing robot accuracy through integration of an extended Kalman filter algorithm and an artificial neural network.* Neurocomputing, 2015. **151**: p. 996-1005.

15. Ayyıldız, M. and K. Çetinkaya, *Comparison of four different heuristic optimization algorithms for the inverse kinematics solution of a real 4-DOF serial robot manipulator.* Neural Computing and Applications, 2016. **27**(4): p. 825-836.

16. Çabuk, N., V. Bakırcıoğlu, and F. Şen, *Neural Network Approach for Inverse Kinematic of a 4-DOF Lighting Manipulator*. 2017.

17. Corke, P., *Robotics, Vision and Control*. 2017: Springer.

18. 6 th International Advanced Technologies Symposium (IATS'11), 16-18 May 2011, Elazığ, Turkey.