Research Reports on Computer Science
https://ojs.wiserpub.com/index.php/RRCS/

UNIVERSAL WISER PUBLISHER

Research Article in Special Issue: Selected Papers from the 4th International Conference on Machine Learning, Image Processing, Network Security and Data Sciences (MIND-2022)

Determining Homogenous Transformation Matrix from DH Parameter Table Using Deep Learning Techniques

Vijay Bhaskar Semwal*, Yash Gupta

Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, India
E-mail: vsemwal@gmail.com

Received: 8 March 2023; **Accepted:** 28 March 2023

Abstract: One of the most popular ways of representing any robotic model mathematically is through Denavit-Hartenberg (DH) parameter table. And the most common way of finding a forward kinematics solution to any robotic model is by finding its homogenous transformation matrix, which is obtained from the DH parameter table by a certain set of steps or algorithms. In this research work, we have tried solving this problem in just a single step by deep learning method and thus finding forward kinematics of almost any kind of manipulator. This research work shows not just this problem but many more such complex problems which require a certain set of steps or algorithms that can be solved by deep learning techniques in a single step. The results obtained are very close to accurate and show the ability of deep learning techniques for solving different kinds of such problems.

Keywords: robotics, forward kinematics, DH parameter, homogenous transformation matrix, neural networks

1. Introduction

Robotic technology has an impact on every element of life, both at work and at home. Robotics can improve people's lives and work by performing them in a more efficient way and at the same time increasing safety with better service. We all know, robotics is set to be the leading technology for the next generation of gadgets, by their learning capability, and ability to interact with the surrounding, bridging the digital and physical worlds. In big industrial businesses, it has already become essential for flexibility and competitiveness. On the other hand, in non-manufacturing businesses like agriculture, healthcare, security, and transportation services, robots will have a significant impact. With the growth in these industries, it is expected that expansion in this domain will be even more.

Figure 1 shows different domains of robotics study. The majority of robotics challenges necessitate modelling. Kinematics analysis and dynamic analysis are both required for robotic modelling. Most of the robotics implementations, such as frame transformation [1], movement of hand [2, 3], stability and control of skeleton or posture [4, 5], three-dimensional (3D) transformation [6], bipedal robot trajectory analysis or prediction [7, 8], and manipulators for industrial applications [9, 10], trajectory tracing [11] necessitate solving the inverse kinematics issue to determine joint trajectories. The kinematics model examines the system's basic geometry, on the other hand, the dynamics of a model examines the model when it is in motion. The association between forward kinematics and inverse kinematics is

DOI: https://doi.org/10.37256/rrcs.2320232627

depicted in Figure 2.

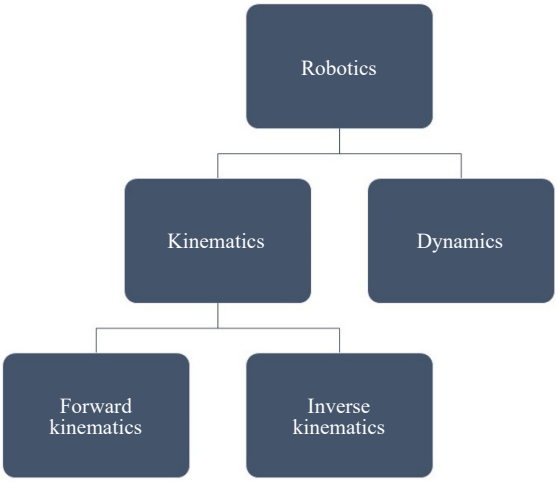

Figure 1. Domains of robotics

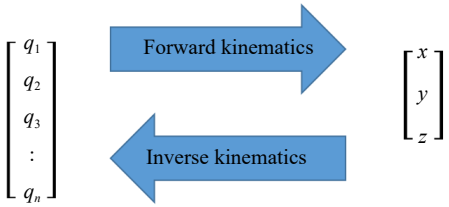

Figure 2. Forward and inverse kinematics relationship

The most common representation of any type of robot is the Denavit-Hartenberg (DH) representation [12]. It can aid in the detection of both forward and inverse kinematics issues. The DH parameter is a mathematical representation of the robot link structure. It can be used to obtain forward and inverse kinematics solutions by calculating the homogenous matrix. In this research, we will first explain a step-by-step way for obtaining a homogenous transformation matrix from a DH parameter table, and then use deep learning techniques to solve the same problem in a single step.

2. Methodology

2.1 *DH frames*

DH frames assist us in deriving the equations necessary to control a robotic arm. We use four rules to draw the frames (i.e., the x, y, and z axes), which will allow us to take a shortcut while deriving the robot's mathematics. The DH convention is the name given to all of these rules.

The following are the four rules for drawing DH coordinate frames:

(a) A revolute joint's rotation axis is called the z-axis.

(b) Both the current z-axis and the prior z-axis must be perpendicular to the x-axis.

(c) Using the right-hand coordinate system, the y-axis is calculated from the x- and z-axes.

(d) The preceding z-axis must intersect the x-axis (the rule does not apply to frame 0).

Figure 3 shows the Universal Robot-UR5 is shown in the zero position or a flat orientation parallel to the plane in the illustration above. An end effector, such as a hand, gripper, or suction cup, is attached to the robotic arm's end.

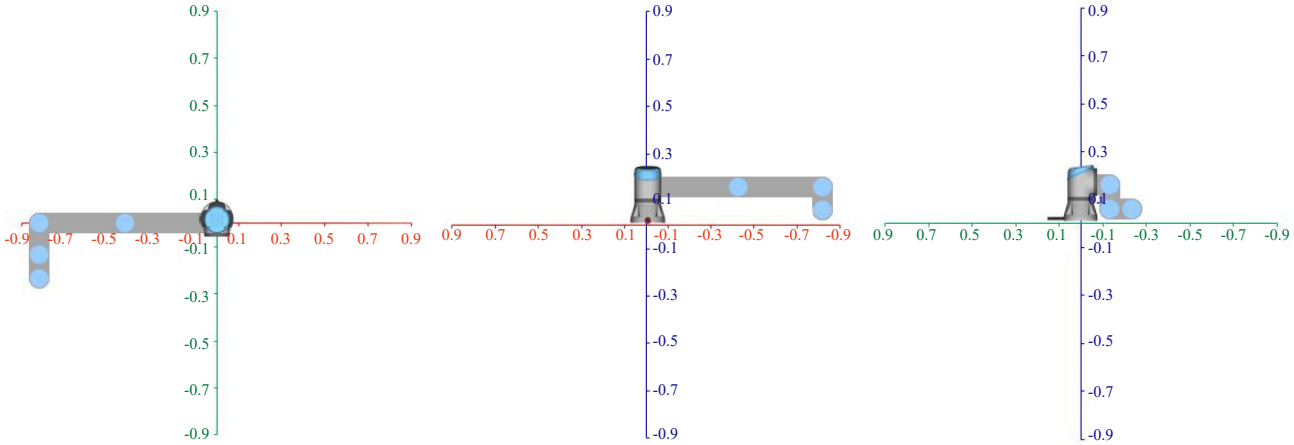

Figure 3. View of a collaborative robot having 6 degree of freedom (DOF) from different angles

Let's try drawing DH frames of Universal Robot-UR5 using the principles above. It's a well-known 6 DOF industrial robot that's been utilized as a standard in numerous experimental researches.

The DH frames of the aforesaid UR5 robot at zero position will be as shown in Figure 4 using the four rules of the DH convention.

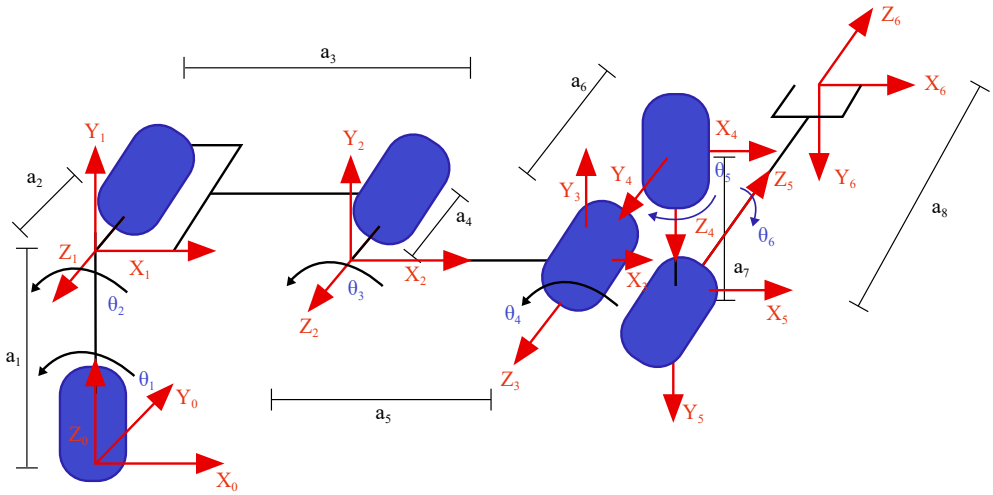

Figure 4. DH frames for UR5 robot

2.2 *Homogenous transformation matrices*

It allows combining rotation matrices (3 rows, 3 columns) with displacement matrices (3 rows, 1 column) into a single matrix. They're a crucial part of forward kinematics. Let's talk about rotation and displacement matrices first, and then we'll talk about transformation matrices.

Rotation matrices: The angles of a robotic arm can be described using these matrices (i.e. to indicate the direction of the robotic arm). It aids us in detecting how a robot's end effector (e.g., robotic gripping hand, paintbrush, suction cup, etc.) will change its orientation with changes in servo motor angles.

In other words, it is a mathematical representation of a robotic system's orientation. The three coordinate axes (x, y, and z) indicating the position of an object in 3D in one frame are transformed to another frame coordinates via rotation matrices.

Rotation matrices differ depending on which axis the frame rotated. Equations (1), (2), and (3) show rotation

matrices for rotation along the x, y and z axes respectively.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

$$R_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2}$$

$$R_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Displacement vectors: A displacement vector is a list of three values in a single column that represents the displacement (or displacement in position) of a frame with respect to some other frame in coordinate axes.

To indicate the change in the position of coordinate frame *n* with respect to coordinate frame *m*, we'll use the notation as in Equation (4).

$${}^{0}_{n}P = \begin{bmatrix} {}^{0}_{nx}P \\ {}^{0}_{ny}P \\ {}^{0}_{nz}P \end{bmatrix} \tag{4}$$

We can simply multiply the rotation matrices of each joint to determine the orientation of the end effector with respect to the base frame as represented by Equation (5).

$${}^{0}_{n}R = {}^{0}_{1}R\,{}^{1}_{2}R \ldots \ldots \ldots \ldots {}^{n-1}_{n}R \tag{5}$$

However, this is not the case for displacement vectors. For determining the position of the end effector frame with respect to the base frame, we can't just combine displacement vectors as in Equation (6).

$${}^{0}_{n}P \neq {}^{0}_{1}P\,{}^{1}_{2}P \ldots \ldots \ldots \ldots {}^{n-1}_{n}P \tag{6}$$

A homogenous transformation matrix [13] is required to solve this problem. The rotation matrix is augmented with the displacement vector to form a single matrix called a homogeneous transformation matrix. Equation (7) is an example of a homogeneous transformation.

$$T = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \tag{7}$$

The reason for calculating transformation matrices is that, like rotation matrices, we can multiply two homogeneous matrices together. Let's say the transformation matrix from frame 0 to frame 2 is homogen 0 2. To do so, multiply the transformation matrix from frame 0 to 1 by the transformation matrix from frame 1 to 2 as in Equation (8).

$${}^{0}_{n}T = {}^{0}_{1}T\,{}^{1}_{2}T \ldots \ldots \ldots \ldots {}^{n-1}_{n}T \tag{8}$$

DH parameter: We can use DH frames to find DH parameters, and we can also use them to find homogeneous

transformation matrices. The DH parameter is a set of four variables that aid in the mathematical representation of each joint. According to [14], the DH parameters are as follows.

a_i = distance from Z_i to Z_{i+1} measured along X_i

α_i = angle measured around X_i from Z_i to Z_{i+1}

d_i = distance between X_{i-1} and X_i as measured along Z_i

θ_i = angle measured around Z_i from X_{i-1} to X_i

The DH parameters for UR5 are given in Table 1.

Table 1. DH parameters for UR5

Link	α_{i-1}	a_i	d_i	θ_i
1	0	0	d_1	θ_1
2	90	0	0	θ_2
3	0	a_2	0	θ_3
4	0	a_2	d_4	θ_4
5	90	0	d_5	θ_5
6	-90	0	d_6	θ_6

DH characteristics of a UR5 robot, modified to correspond to the frames in Figure 4. The first parameter, *i*, is variable, whereas the rest are constants.

The transformation for each connection can be written using the DH parameters by Equation (9).

$$ {}^{n-1}_{n}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & \alpha_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & \sin\alpha_{i-1} d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1} d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9} $$

Multiplying all six transformation matrices as in Equation (8) yields the complete transformation from base to end-effector. Just like transformation matrices can be combined, it can be broken down into a series of transformations as in Equation (10).

$$ {}^{0}_{6}\boldsymbol{T}(\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5\ \theta_6) = {}^{0}_{6}\boldsymbol{T}(\theta_1)\,{}^{0}_{6}\boldsymbol{T}(\theta_1)\,{}^{0}_{6}\boldsymbol{T}(\theta_1)\,{}^{0}_{6}\boldsymbol{T}(\theta_1)\,{}^{0}_{6}\boldsymbol{T}(\theta_1)\,{}^{0}_{6}\boldsymbol{T}(\theta_1) \tag{10} $$

2.3 *Deep learning*

All of the preceding mathematics can be avoided by using deep learning techniques to solve the problem. Deep learning is a part of machine learning which in turn is a part of artificial intelligence. Artificial intelligence refers to methods that allow machines to emulate human-like intelligent behaviour.

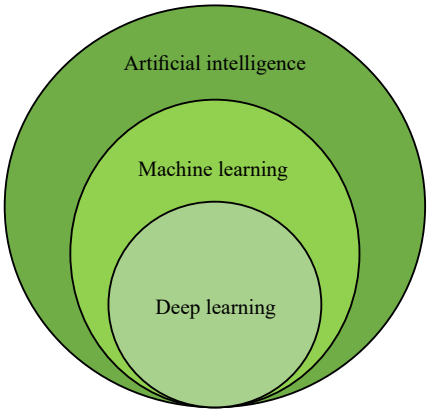

Figure 5. Deep learning is a part of artificial intelligence

Deep learning is a part of machine learning, Figure 5 that is influenced by human brain structure. Deep learning algorithms analyse inputs with a predetermined logical structure in order to reach similar conclusions as humans. Deep learning achieves this by employing a multi-layered neuron structure known as neural networks as shown in Figure 6.

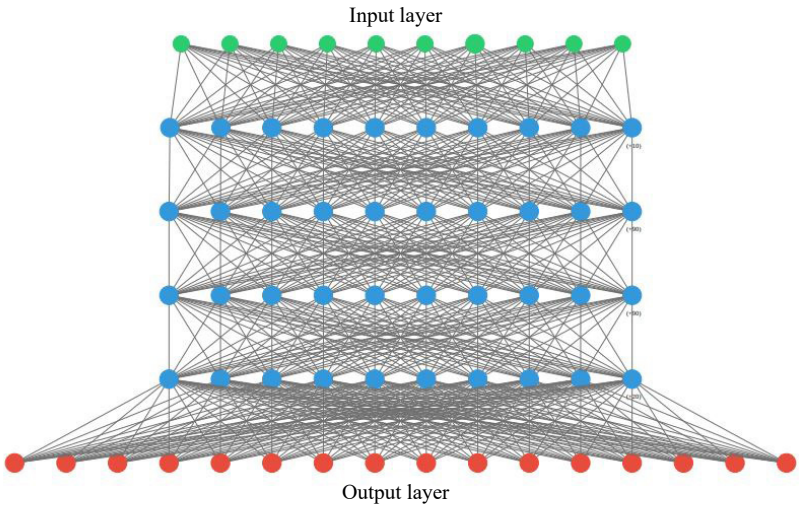

Figure 6. Neural network architecture

Different layers of neural networks are nothing but a kind of filter which works from the most obvious to the most subtle, improving the possibility of predicting the right result. These kinds of neural network models are capable of solving problems that machine learning models can't.

The lack of necessity for so-called feature extraction gives an edge to deep learning. Feature extraction is usually quite difficult and necessitates a thorough understanding of every part of a problem. For best results, pre-processing is a must, it should be customised, tested, and optimised. Deep learning, on the other hand, doesn't require feature extraction.

The layers can directly and independently learn an implicit likeness of the raw input. Over successive layers of neural networks, which helps in forming a more compressed and more abstract form of raw data. The result is then generated using this compact form of the data. The result could then be used for the classification of data into different classes. Consider the case shown in Figure 7.

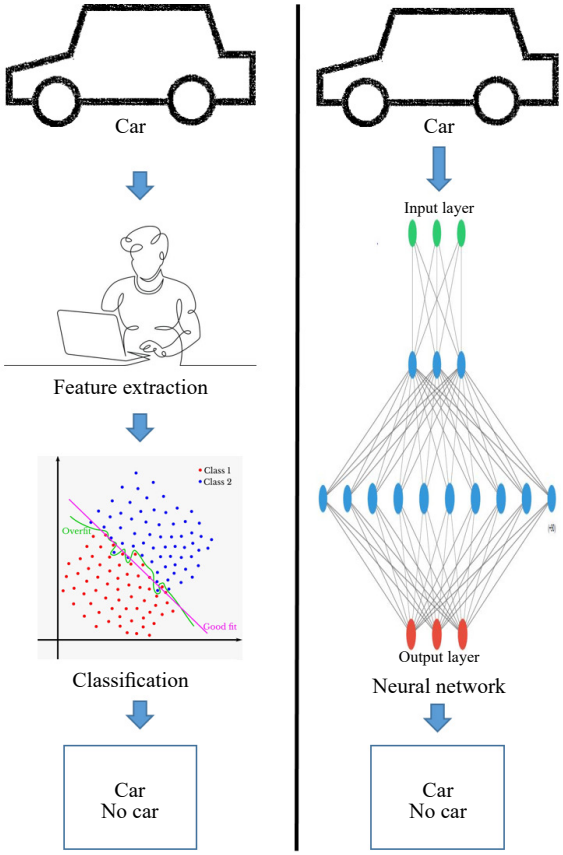

Figure 7. Difference between machine learning and deep learning method

Thus, we can also say that feature extraction is included in the artificial neural network's operation. In reality, avoiding data characteristics extraction applies to every single activity you'll ever perform using neural networks. Simply feed the raw inputs to the network, and the model will handle the rest.

3. Result and verification

3.1 *Preparation of dataset*

As discussed in the last section, we need to provide data for training our neural network. For this, we require some datasets. There is no publicly available dataset for training neural networks for finding homogenous transformation matrix. So, we implied the mathematical approach discussed in Section 2 to generate a large size of data so that we can train our model properly.

At first, we randomly assigned values to different DH parameters (i.e., a, d, r, and θ). Based on the number of links and DOF, multiple sets of DH parameters are required. For example, a 3 DOF, 3 link manipulator required 3 sets of DH parameters as shown in Table 2.

Table 2. DH parameters for a sample of 3 DOF manipulator

Link	α_{i-1}	a_i	d_i	θ_i
1	0	0	d_1	θ_1
2	90	0	0	θ_2
3	0	a_2	0	θ_3

We generated 10,000 such values and then using Equation (9), we calculated the homogenous transformation matrix. As discussed earlier, the homogenous transformation matrix is a 4 × 4 matrix, but we have stored it in a single row, thus our output is these 16 values and the input is 4 × n, where n is the number of links. In the above case, 3 DOF 3 link manipulator, it will be 12.

3.2 *Architecture*

There is no specific rule in neural networks about what kind of neural network should be used for what purpose. It is broadly of three types Feed Forward Neural Network, Recurrent Neural Network and Convolutional Neural Network. Each one is further categorized into different kinds. For our research work, we found out by experiment that a feed-forward multi-layered neural network is best suited [15, 16]. Thus, we tried different feed-forward neural network architectures having different layers of neurons, different activation functions and tested different drop-out and regulariser methods to find the best-suited neural network for our work.

Neural network structure varies for different robotic models as it will have a different number of input nodes. The neural network for 6 DOF, 6 link manipulator is shown in Figure 6. By trying out different activation functions, we found out that the sigmoid (11) and a rectified linear unit (ReLU) activation function (13) is most suited. In the hidden layer, we have employed the sigmoid function while in output neurons we have employed the ReLU function.

$$y = \frac{1}{1+e^{-x}} \tag{11}$$

$$y = \max(0, x) \tag{12}$$

The neural network needs to start with some weights and then iteratively update them to better values. In our case, we have used uniform distribution to initialize all of the weights. During the training of the neural network, these initialized weights are modified in order to reduce the losses. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rates. There are different types of optimizers: Gradient Descent, Stochastic Gradient Descent, Adagrad, Adadelta, Adam, etc. We have found out experimentally that the Adam optimizer is most suitable for our work. It gives minimum losses.

So basically, we will reduce this algorithmic approach of finding a homogenous transformation matrix into a single step by neural network technique. The whole concept is explained with the help of the flow diagram in Figure 8.

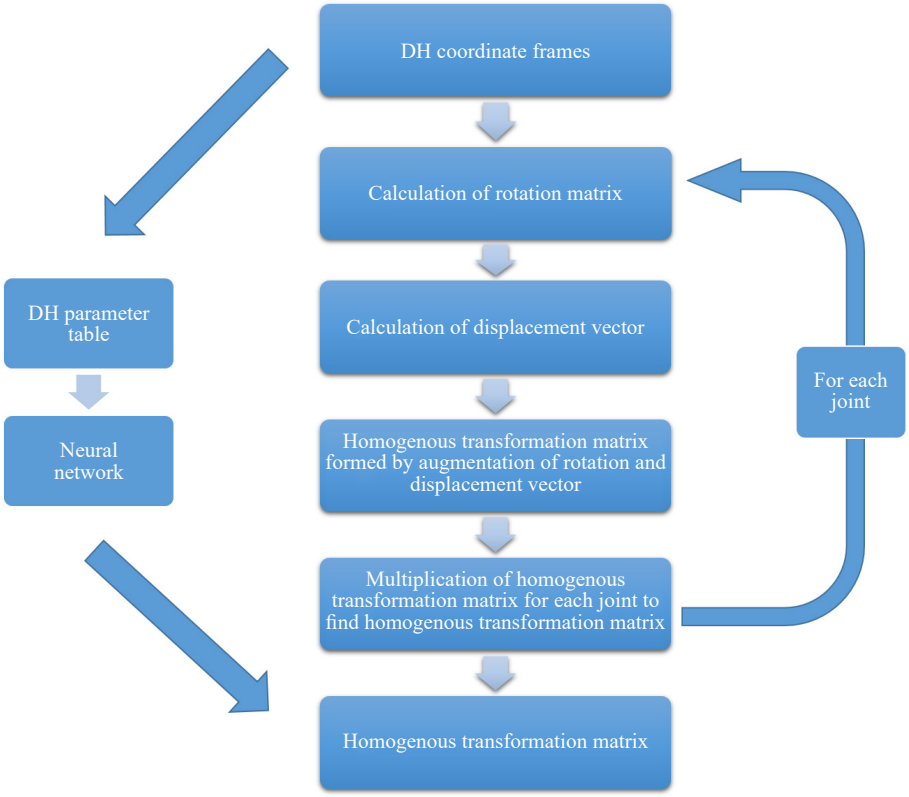

Figure 8. Different architectures of finding homogenous transformation matrix

To test the efficiency of neural network in finding homogenous transformation matrix and forward kinematics, we have taken 3 types of manipulator. First with 3 links 3 DOF, second with 5 links 6 DOF, and last with 11 links 12 DOF. We have prepared separate datasets for each manipulator. The neural network employed in all the cases is almost the same only they have different numbers of input nodes. We'll utilise 10% of the datasets we've produced for validation, 15% for testing, and the remaining 75% for training the network.

As this is a regression problem, we'll use root mean squared error (RMSE) and the cosine similarity value method to assess our model's performance. Mean squared error (13) is the most common way of analysing regression models. It determines the mean of the squares of the errors between actual output and predicted output.

$$\text{Loss} = \text{mean}\left((y_{\text{true}} - y_{\text{pred}})^2\right) \tag{13}$$

Cosine similarity (14) is a mathematical tool that computes the similarity between actual output and predicted output. It's a number that ranges from -1 to 1. When the value is a negative integer between -1 and 0, where 0 denotes orthogonality whereas numbers nearer to -1 suggest a higher resemblance. The closer the value is to 1, the higher the dissimilarity. As a result, it may be used as a loss function in a situation where the goal is to minimise the distance between targets and forecasts. Regardless of the distance between targets and predictions, cosine similarity will be 0 if either y_{true} or y_{pred} is a zero vector.

$$\text{Loss} = -\sum[\ (l_2\_\text{norm}\ (y_{\text{true}}) * l_2\_\text{norm}\ (y_{\text{pred}})] \tag{14}$$

Figures 9 and 10 show the loss curves for 6 DOF manipulator while training the model for predicting the homogenous transformation matrix. The first curve shows the RMSE loss during training and validation while the second curve shows cosine loss during training and validation of the model. As can be seen from the loss curves, the loss is very little thus our model is quite accurate in learning the hidden patterns in the problem.

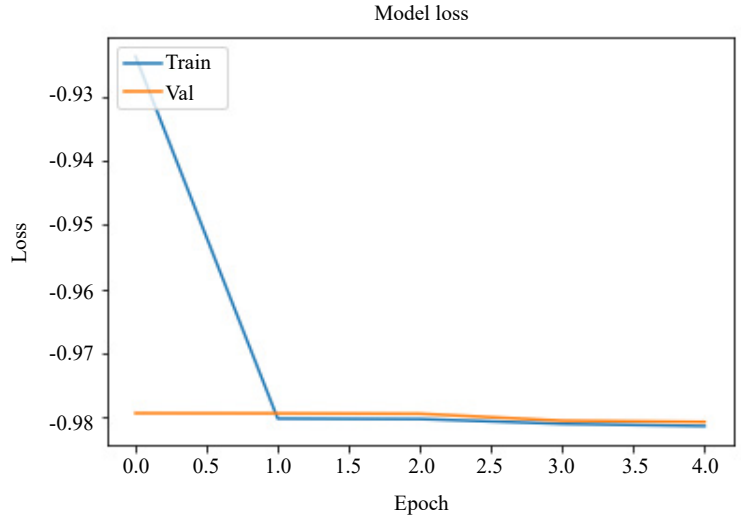

Figure 9. RMSE loss during training the model

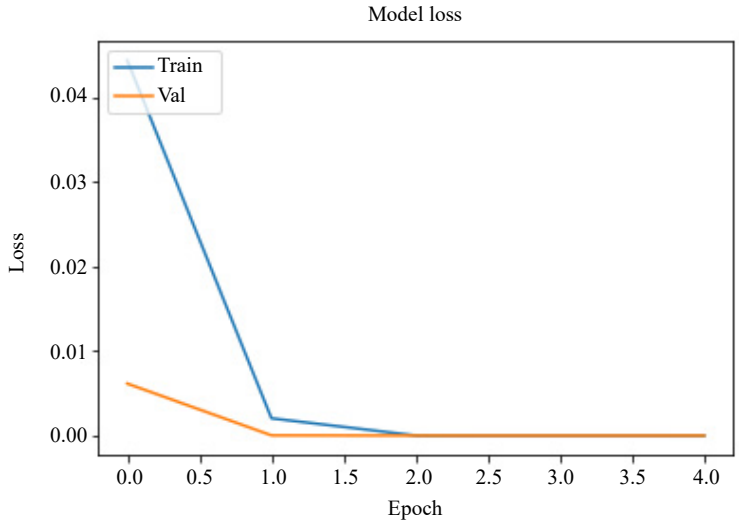

Figure 10. Cosine loss during training the model

We have used our model to predict the rotation matrix, position matrix, forward kinematics and homogenous transformation matrix from the DH parameter table. Testing results are shown in Tables 3 and 4.

Table 3. Errors in calculating position and rotation matrices

Matrix	3 DOF		6 DOF		9 DOF	
	RMSE	Cosine similarity	RMSE	Cosine similarity	RMSE	Cosine similarity
Rotation matrix	0.0013	-0.998	0.012	-0.999	0.0006	-0.999
Position matrix	0.0068	-0.998	0.000006	-0.999	0.000007	-0.999

Table 4. Errors in calculating forward kinematics and homogenous transformation matrix

Matrix	3 DOF		6 DOF		9 DOF	
	RMSE	Cosine similarity	RMSE	Cosine similarity	RMSE	Cosine similarity
Forward kinematics	0.0068	-0.998	0.000006	-0.999	0.000007	-0.999
Homogenous matrix	0.0060	-0.996	0.000002	-0.999	0.0005	-0.998

4. Conclusion

As we can see from the results, the neural network is capable of calculating the rotation matrix, displacement matrix and thus finding forward kinematics with RMSE of around 0.006 in most of the cases. Hence, we can say that predicted results are very close to actual results. Also, we can say that neural networks are capable of solving the algorithmic problem just like solving forward kinematic problems from the DH parameter table in our case.

Conflict of interest

The authors declare that there is no conflict of interest.

References

[1] Sao C, Lehn PW. A block diagram approach to reference frame transformation of converter dynamic models. In: *2006 Canadian Conference on Electrical and Computer Engineering*. Ottawa, Canada: IEEE; 2006. p.2270-2274. https://doi.org/10.1109/CCECE.2006.277744

[2] Gao H, He W, Zhou C, Sun C. Neural network control of a two-link flexible robotic manipulator using assumed mode method. *IEEE Transactions on Industrial Informatics*. 2019; 15(2): 755-765. https://doi.org/10.1109/TII.2018.2818120

[3] Almusawi AR, Dülger LC, Kapucu S. A new artificial neural network approach in solving inverse kinematics of robotic arm (Denso VP6242). *Computational Intelligence and Neuroscience*. 2016; 2016: 5720163. https://doi.org/10.1155/2016/5720163

[4] Cashbaugh J, Kitts C. Automatic calculation of a transformation matrix between two frames. *IEEE Access*. 2018; 6: 9614-9622. https://doi.org/10.1109/ACCESS.2018.2799173

[5] Semwal VB, Katiyar SA, Chakraborty R, Nandi GC. Biologically-inspired push recovery capable bipedal locomotion modeling through hybrid automata. *Robotics and Autonomous Systems*. 2015; 70: 181-190. https://doi.org/10.1016/j.robot.2015.02.009

[6] Mukerjee A, Mittal N. Qualitative models of 3-D frame-transformation motions. In: *Proceedings of 1995 IEEE International Conference on Robotics and Automation*. Nagoya, Japan: IEEE; 1995. p.2279-2284. https://doi.org/10.1109/ROBOT.1995.525601

[7] Nandi GC, Semwal VB, Raj M, Jindal A. Modeling bipedal locomotion trajectories using hybrid automata. In: *2016 IEEE Region 10 Conference (TENCON)*. Singapore: IEEE; 2016. p.1013-1018. https://doi.org/10.1109/TENCON.2016.7848159

[8] Semwal VB, Nandi GC. Generation of joint trajectories using hybrid automate-based model: A rocking block-based approach. *IEEE Sensors Journal*. 2016; 16(14): 5805-5816. https://doi.org/10.1109/JSEN.2016.2570281

[9] Kucuk S, Bingul Z. Robot kinematics: Forward and inverse kinematics. In: Cubero S. (ed.) *Industrial Robotics: Theory, Modelling and Control*. London, UK: IntechOpen; 2006. http://dx.doi.org/10.5772/5015

[10] Bingul Z, Ertunc HM, Oysu C. Applying neural network to inverse kinematic problem for 6R robot manipulator with offset wrist. In: Ribeiro B, Albrecht RF, Dobnikar A, Pearson DW, Steele NC. (eds.) *Adaptive and Natural Computing Algorithms*. Vienna, Austria: Springer; 2005. p.112-115. https://doi.org/10.1007/3-211-27389-1_27

[11] Duka AV. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Procedia Technology*. 2014; 12: 20-27. https://doi.org/10.1016/j.protcy.2013.12.451

[12] Hayat AA, Chittawadigi RG, Udai AD, Saha SK. Identification of Denavit-Hartenberg parameters of an industrial

robot. In: *Proceedings of Conference on Advances In Robotics*. New York, United States: Association for Computing Machinery; 2013. p.1-6. https://doi.org/10.1145/2506095.2506121
[13] VanArsdale D. Homogeneous transformation matrices for computer graphics. *Computers & Graphics*. 1994; 18(2): 177-191. https://doi.org/10.1016/0097-8493(94)90092-2
[14] Craig JJ. *Introduction to Robotics: Mechanics and Control*. 3rd ed. Beijing, China: Pearson Education International; 2005.
[15] Semwal VB, Gupta Y. Performance analysis of data-driven techniques for solving inverse kinematics problems. In: Arai K. (ed.) *Intelligent Systems and Applications. IntelliSys 2021*. Lecture Notes in Networks and Systems, vol 294. Cham, Switzerland: Springer; 2021. p.85-99. https://doi.org/10.1007/978-3-030-82193-7_6
[16] Semwal VB, Reddy M, Narad A. Comparative study of inverse kinematics using data driven and FABRIK approach. In: *Advances in Robotics-5th International Conference of The Robotics Society*. New York, United States: Association for Computing Machinery; 2021. p.1-6. https://doi.org/10.1145/3478586.3478620