BATCH A

3 Group

ABB IRB 140 INDUSTRIAL ROBOT

Introduction to AI Robotics

Batch A Group 3

1. Pranav G - (CB.EN.U4AIE22016)
2. Ganesh Sundhar S - (CB.EN.U4AIE22017)
3. Gadila Siri Reddy - (CB.EN.U4AIE22019)
4. Hari Krishnan N - (CB.EN.U4AIE22020)

Agenda

01	Introduction
02	Objective
03	Forward and Inverse Kinematics
04	Simulation
05	Sequential Methodology
06	Validation with IK Simulation.

Introduction:

The ABB IRB-140 robot is a compact industrial manipulator with 6 degrees of freedom (DOF)

This robot has a capacity load of 6 Kg and a scope of 810 mm

It also has the possibility of backward dub

High precision drilling operations, taking advantage of the 6 DOF of ABB IRB-140 robot arm.

Objective

machine learning (ML) sequential methodology for robot IK modeling to improve computational efficiency and accuracy

Using : Artificial Neural Networks

Work Flow:

Website Creation:

Offline web page that includes Kinematic information of the robot.

Simulation:

Simulation of the robot using blender and css

Neural Network (FFN)

Implement an Sequential Artificial Neural Network to obtain the inverse kinematics

IK Simulation

Using the obtained Inverse Kinematics coordinates to simulate the drawing

Forward and Inverse Kinematics:

DH Parameter Table:

Used in robotics to describe the robot properties like axis orientations and arm lengths

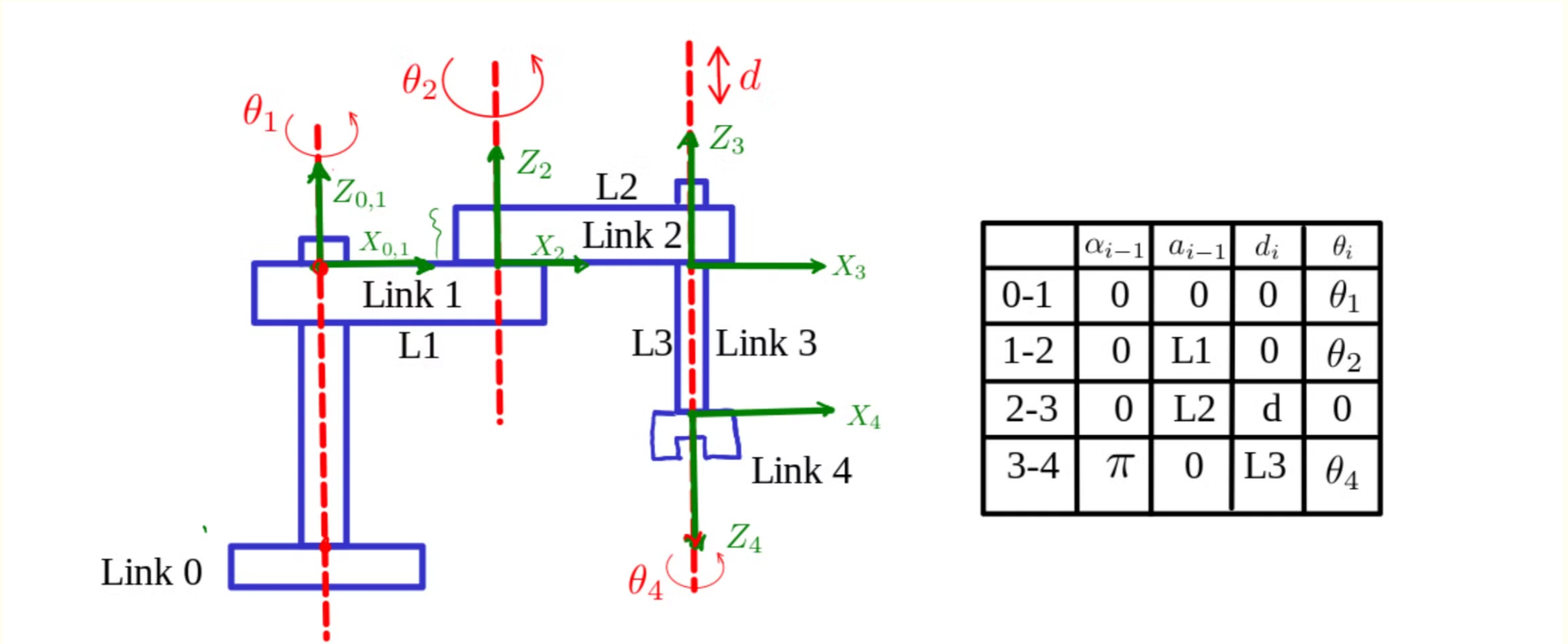
Forward Kinematics:

process of obtaining position of end effector, given the known joint angles.

Inverse Kinematics:

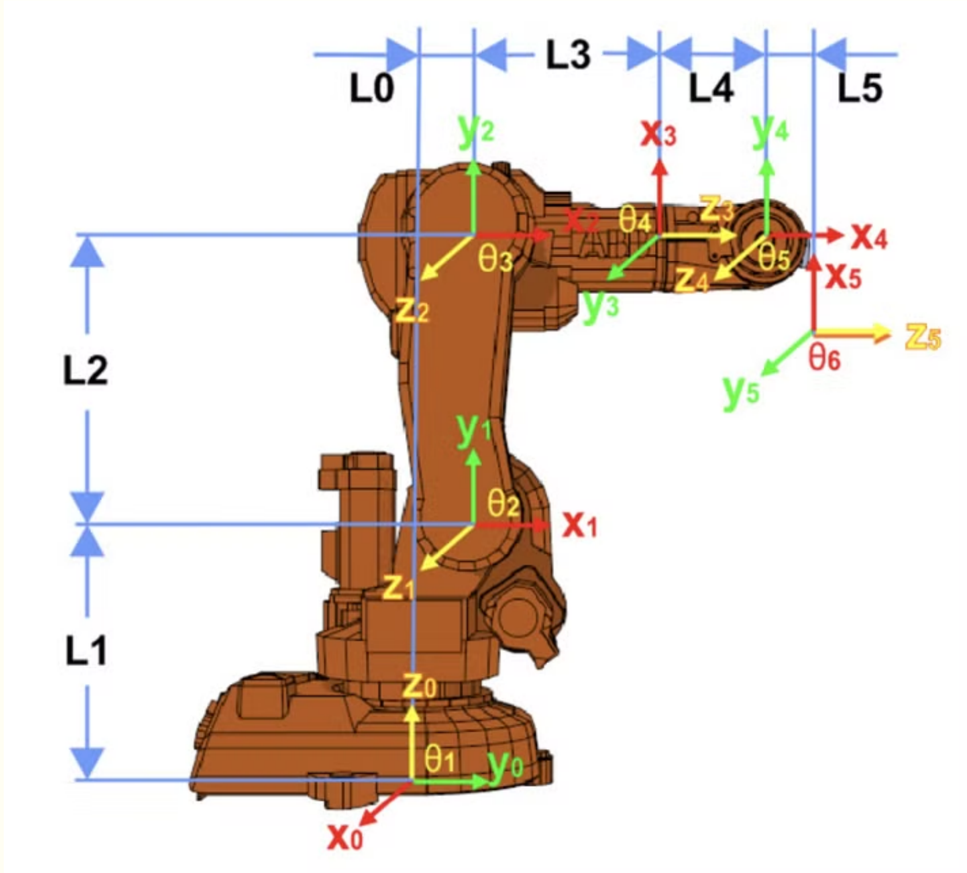
process of obtaining joint angles from known coordinates of end effector.

DH Parameter Table:



	α_{i-1}	a_{i-1}	d_i	θ_i
0-1	0	0	0	θ_1
1-2	0	L1	0	θ_2
2-3	0	L2	d	0
3-4	π	0	L3	θ_4

DH Parameter Table:



Axis (i)	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	$d_1 = 352$	θ_1
2	-90	$a_1 = 70$	0	θ_2-90
3	0	$a_2 = 360$	0	θ_3
4	-90	0	$d_4 = 380$	θ_4
5	90	0	0	θ_5
6	-90	0	0	θ_6

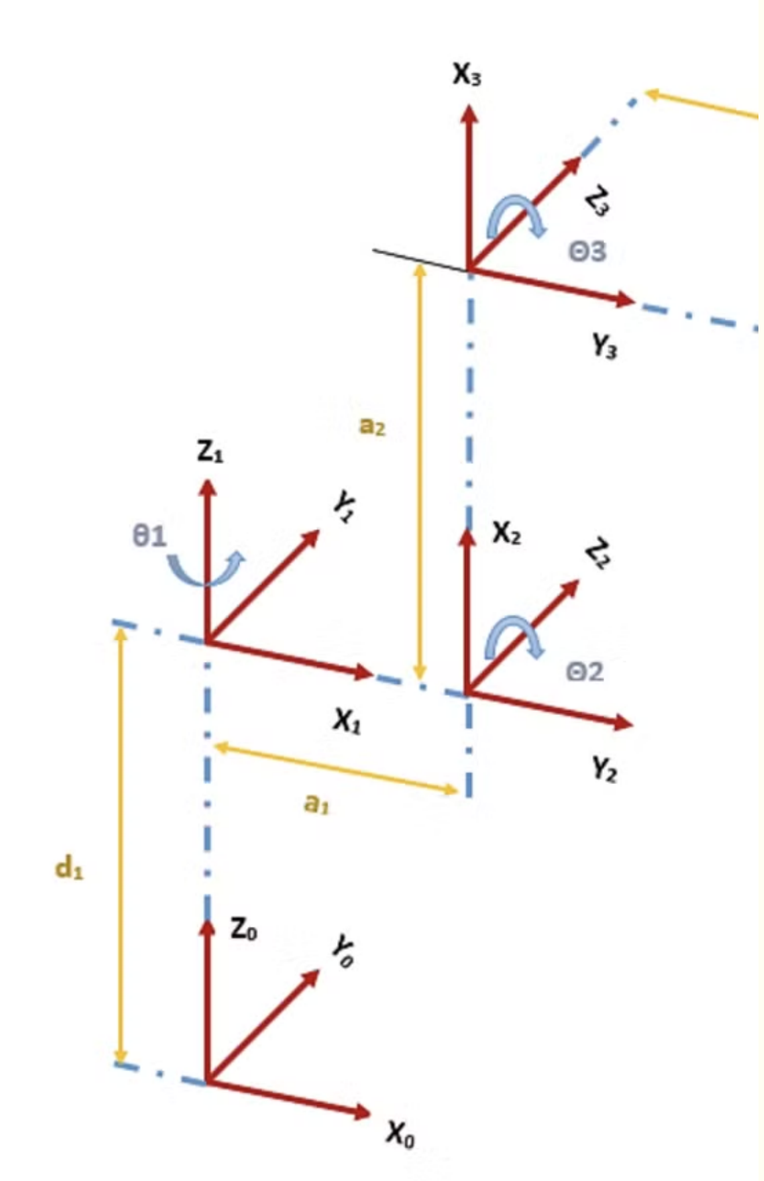
Forward Kinematics:

$$ {}^{0}_{1}T = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ T = T_1 * T_2 * T_3 * T_4 * T_5 * T_6 $$

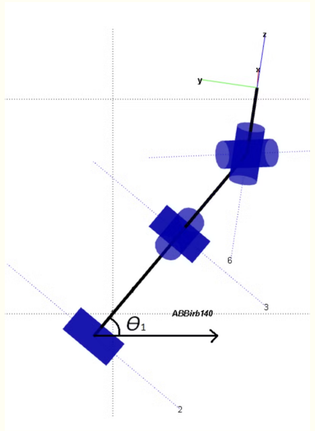
$$ \mathrm{P_{tcp}} = {}^{0}_{6}T \; X \; \mathrm{P^6} $$

$$ \mathrm{P_{tcp}} = \begin{pmatrix} r_{11} & \mathrm{r_{12}} & \mathrm{r_{13}} & \mathrm{x} \\ \mathrm{r_{21}} & \mathrm{r22} & \mathrm{r_{23}} & \mathrm{y} \\ \mathrm{r_{31}} & \mathrm{r_{32}} & \mathrm{r_{33}} & z \\ 0 & 0 & 0 & 1 \end{pmatrix} X \begin{pmatrix} 0 \\ 0 \\ \mathrm{d_6} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{d_6\, r_{13} + x} \\ \mathrm{d_6\, r_{23} + y} \\ \mathrm{d_6\, r_{33} + z} \\ 1 \end{pmatrix} $$



Inverse Kinematics:

Graphical Solution:



$$\theta 1 = \text{atan2}\,(Py, Px),$$
$$\theta 11 = \Pi + \theta 1.$$

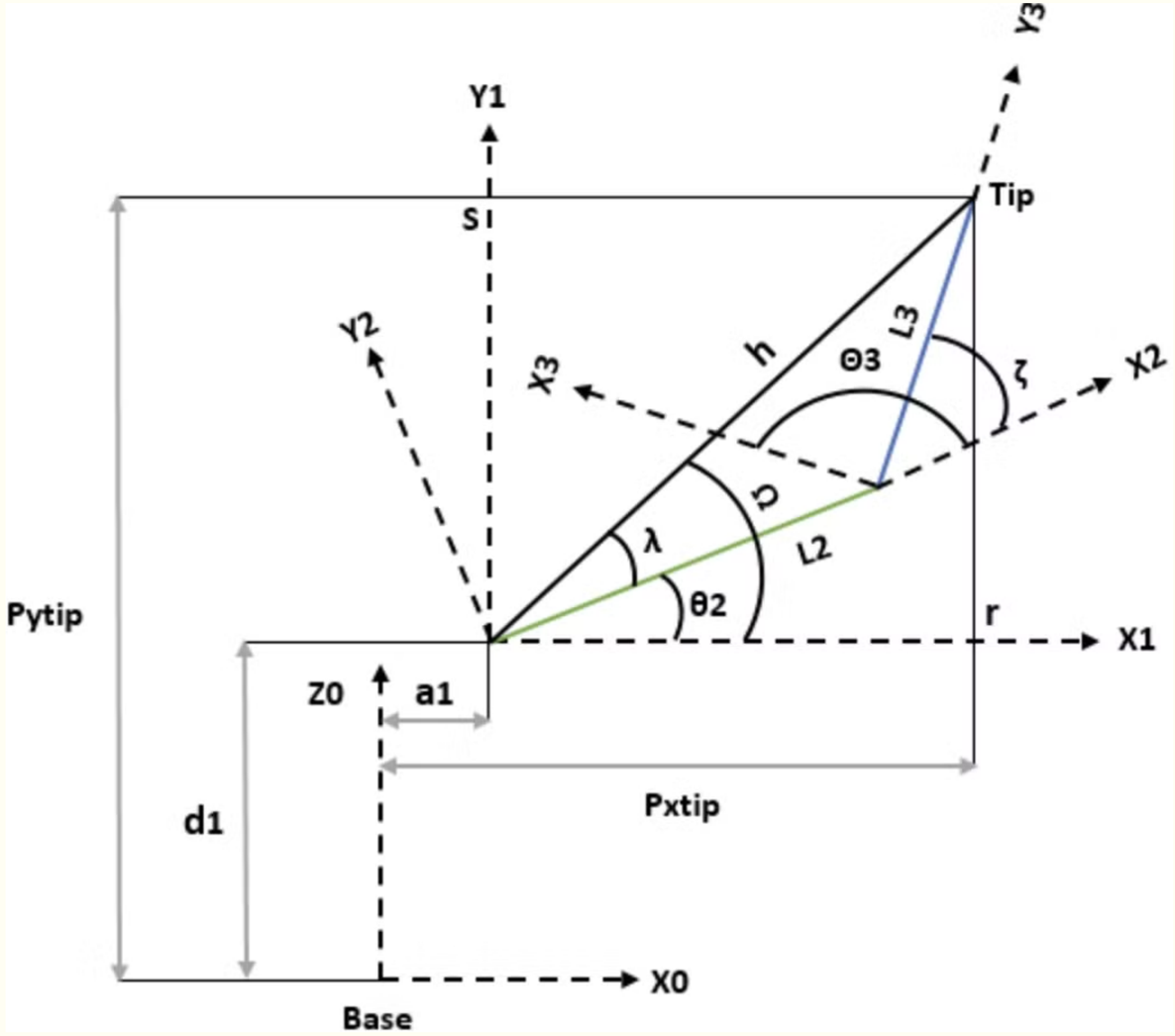
$$\theta 2 = \text{atan2}\,[(P_{ztip} - d1), \pm\sqrt{(Pxtip - a1\cos(\theta 1))2 + (Pytip - a1\sin(\theta 1))2)}] - \text{atan2}\,[(d_4+d_6)\times\sin(\zeta), a_2+(d_4+d_6)\cos(\zeta)].$$

$$\text{Cos}\,(\zeta) = \frac{[(Pztip - d1)2 + (Pxtip - a1\cos(\theta 1))2 + (Pytip - a1\sin(\theta 1))2 - (a2)2 - (d4+d6)\,2]}{2 \times a2 \times (d4+d6)}$$

$$\text{Sin}\,(\zeta) = \pm\sqrt{1 - \text{Cos2}\,(\zeta)}$$

$$\zeta = \text{atan2}\,(\text{Sin}\,(\zeta), \text{Cos}\,(\zeta))$$

Finally, $\theta 3 = -(90 + \zeta)$



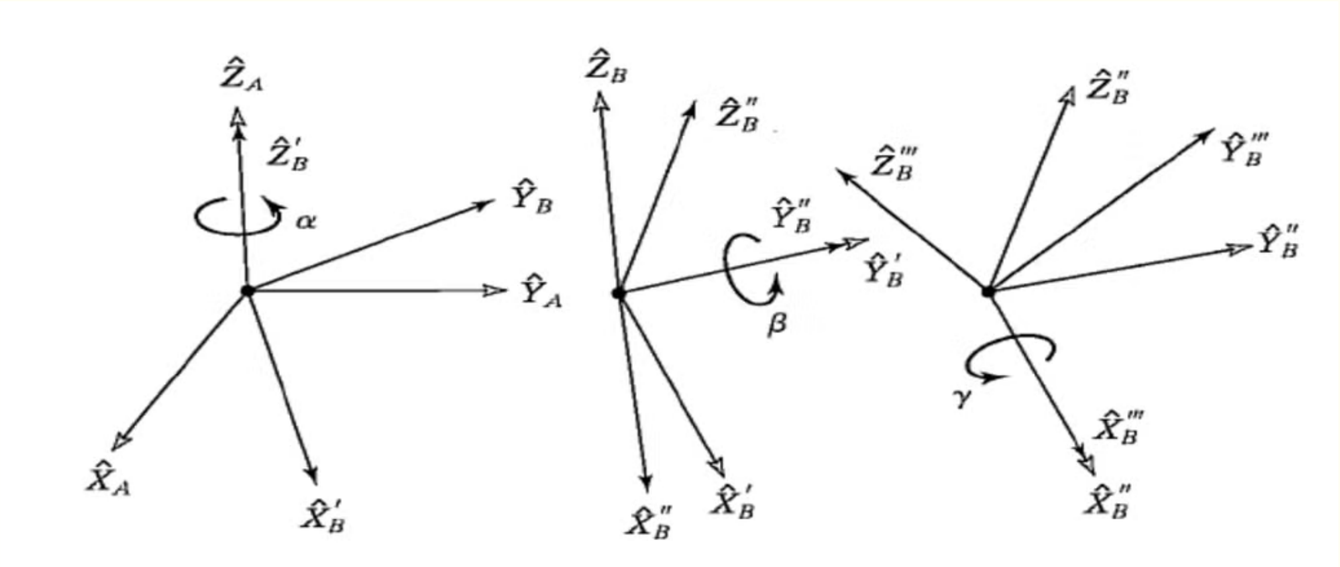
Try Pitch

Inverse Kinematics:

Analytical Solution:

$$ {}^{0}_{3}R = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 \\ -s_{23} & -c_{23} & 0 \end{pmatrix} $$

$$ {}^{3}_{6}R = ({}^{0}_{3}R)^{T} \; {}^{0}_{6}R $$



$$ {}^{3}_{6}R = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} $$

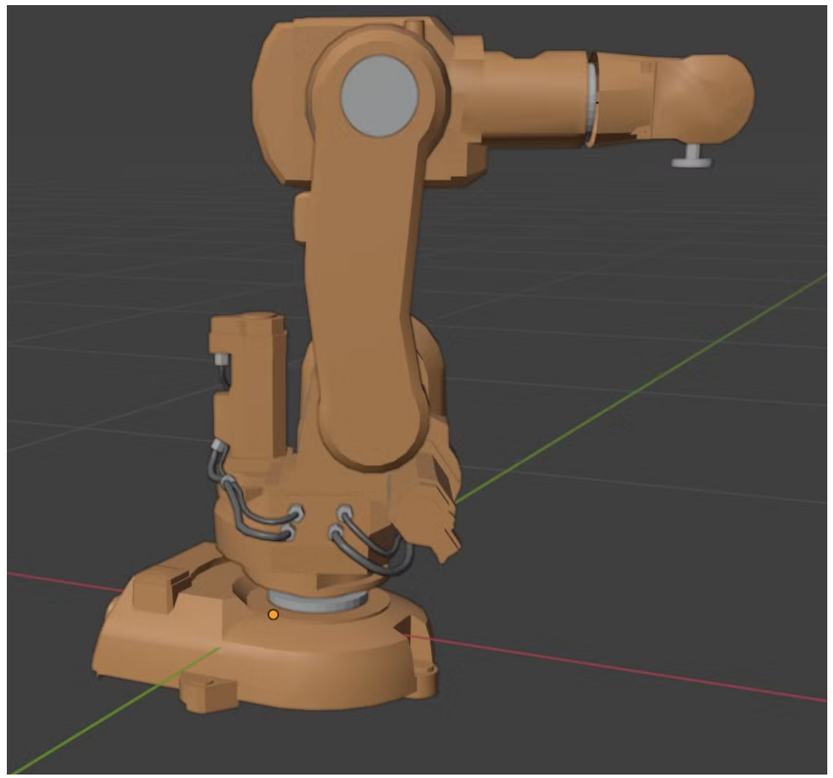
$$ {}^{3}_{6}R = \begin{pmatrix} c_\alpha c_\beta c_\gamma - s_\alpha s_\gamma & -c_\alpha c_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta \\ s_\alpha c_\beta c_\gamma + c_\alpha s_\gamma & -s_\alpha c_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta \\ -s_\beta c_\gamma & s_\beta s_\gamma & c_\beta \end{pmatrix} $$

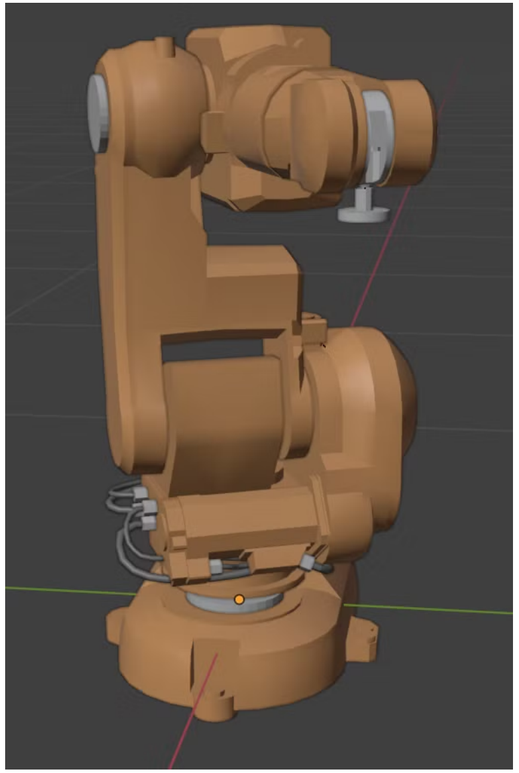
$$ \theta_5 = \beta = atan2\left(+\sqrt{g_{31}^2 + g_{32}^2}, g_{33}\right) $$

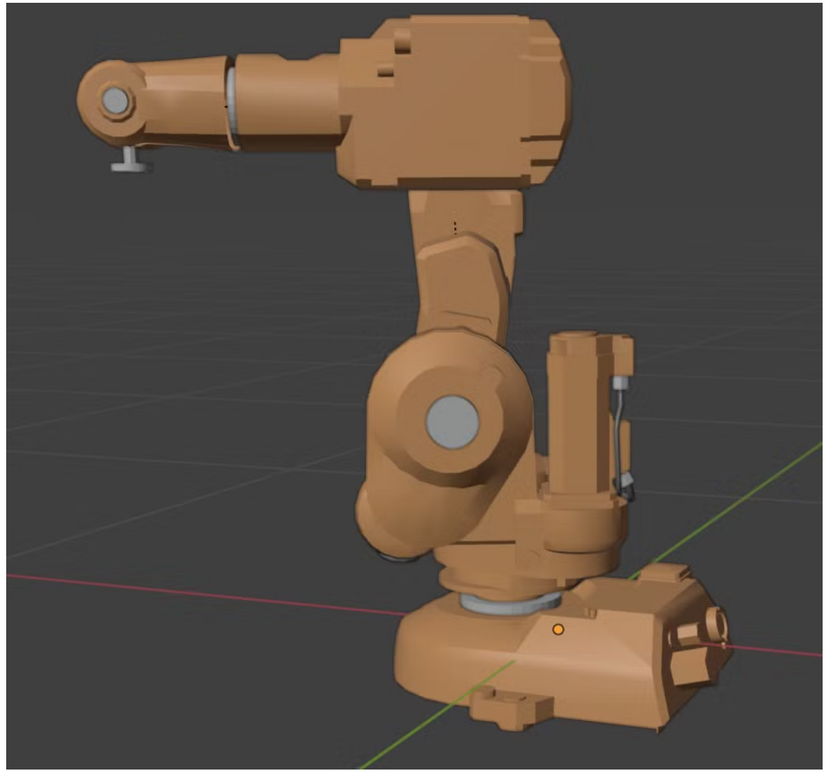
$$ \theta_4 = \alpha = atan2\left(\frac{g_{32}}{s_\beta}, \frac{-g_{31}}{s_\beta}\right) $$

$$ \theta_6 = \gamma = atan2\left(\frac{g_{23}}{s_\beta}, \frac{g_{13}}{s_\beta}\right) $$

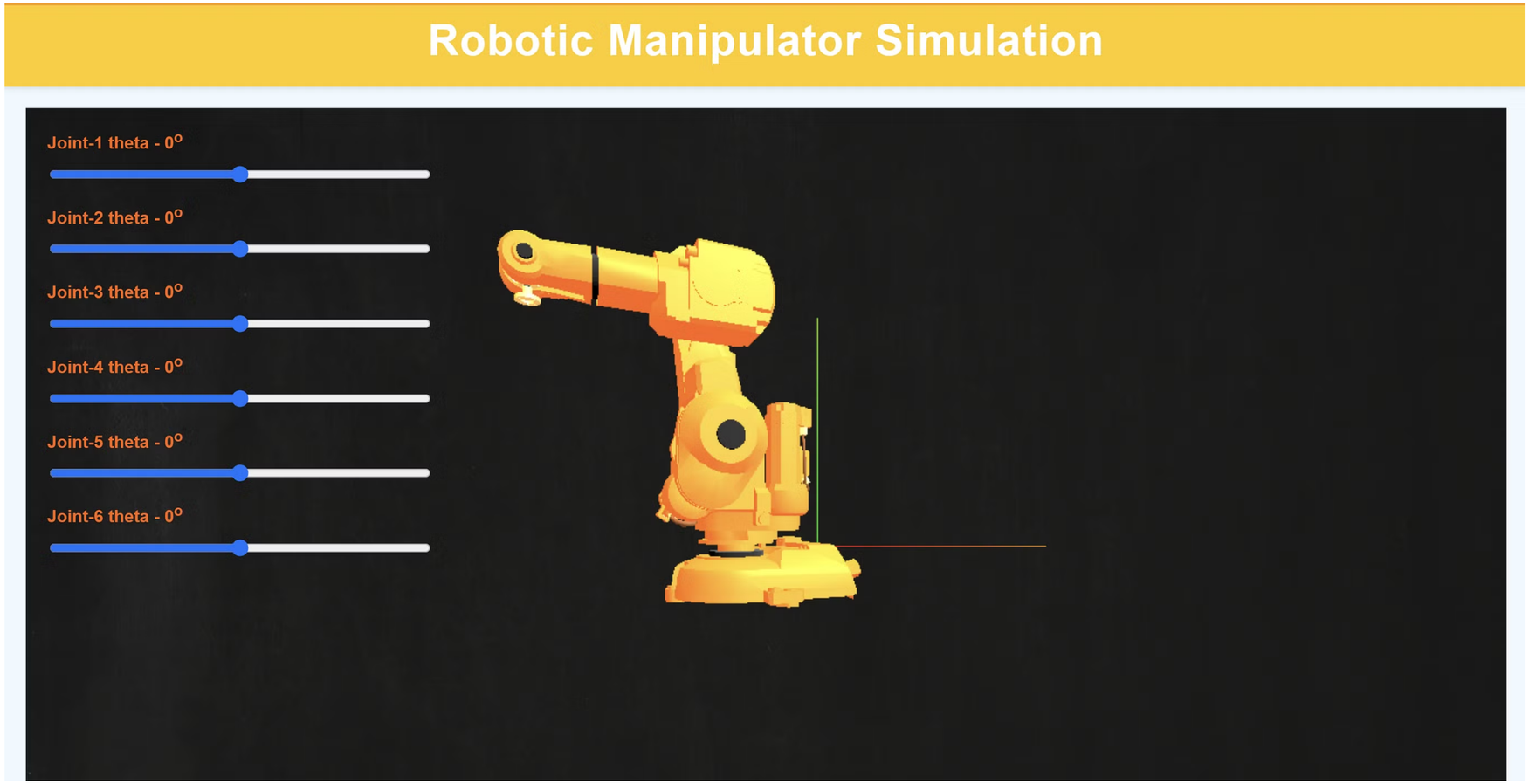
IRB - 140 Simulation model overview:







Simulation:



- Joint 1 : -180° to 180°
- Joint 2 : -45° to 45°
- Joint 3 : -30° to 90°
- Joint 4 : -45° to 45°
- Joint 5 : -100° to 60°
- Joint 6 : -90° to 90°

INVERSE KINEMATICS USING ARTIFICIAL NEURAL NETWORKS

INPUTS

- x - coordinate of the end effector
- y - coordinate of the end effector
- z - coordinate of the end effector

OUTPUTS

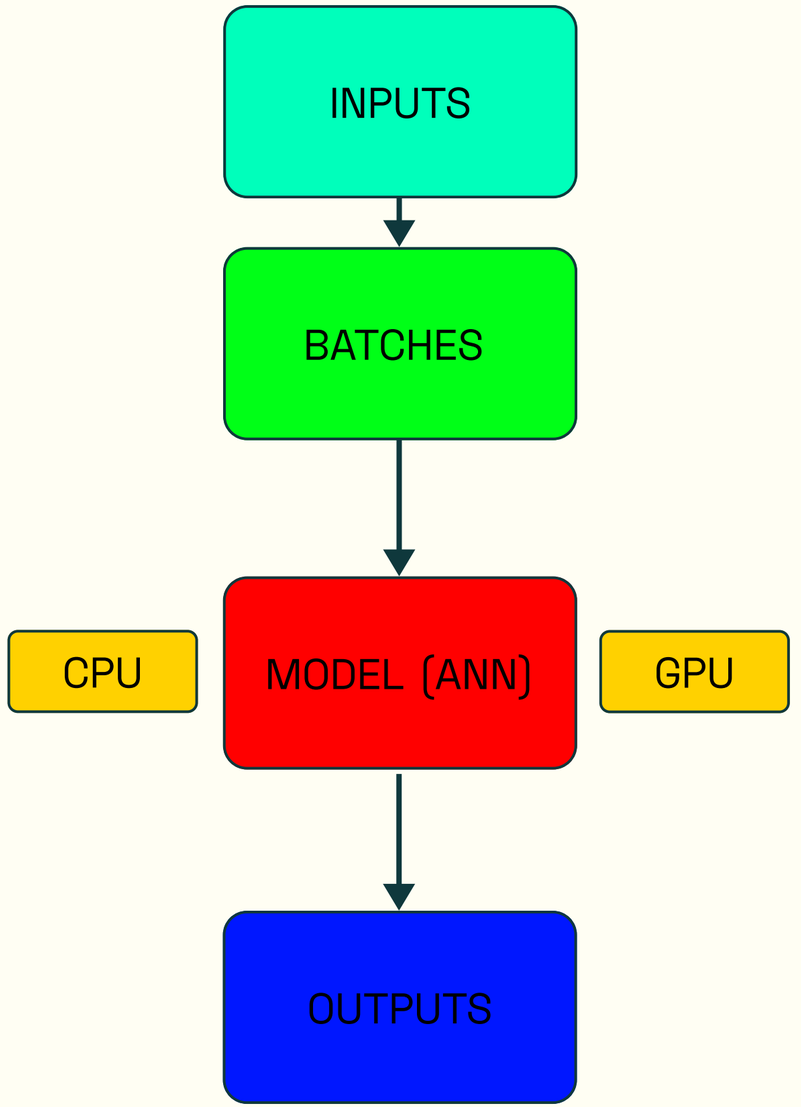
- joint_0 - First joint angle
- joint_1 - Second joint angle
- joint_2 - Third joint angle
- joint_3 - Fourth joint angle
- joint_4 - Fifth joint angle
- joint_5 - Sixth joint angle

BATCH SIZE 32

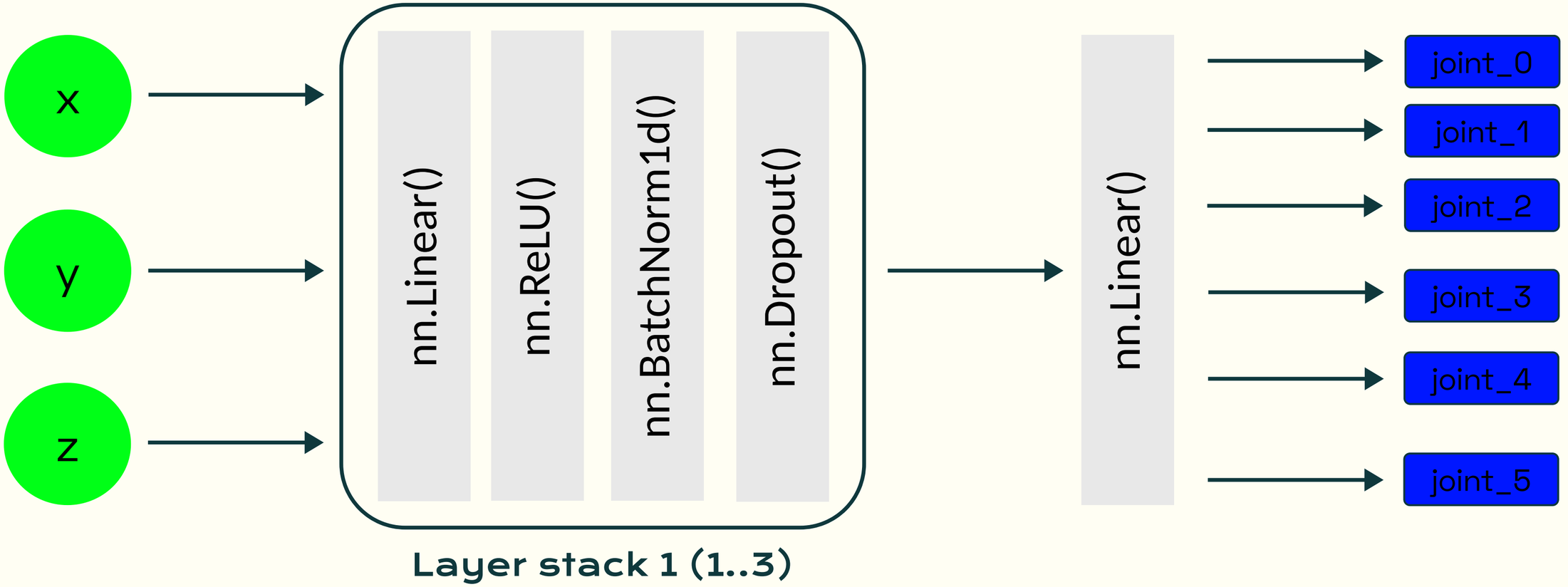
HIDDEN NEURONS 128

TWO TYPES OF MODELS

1. Global Model
2. Sequential Model

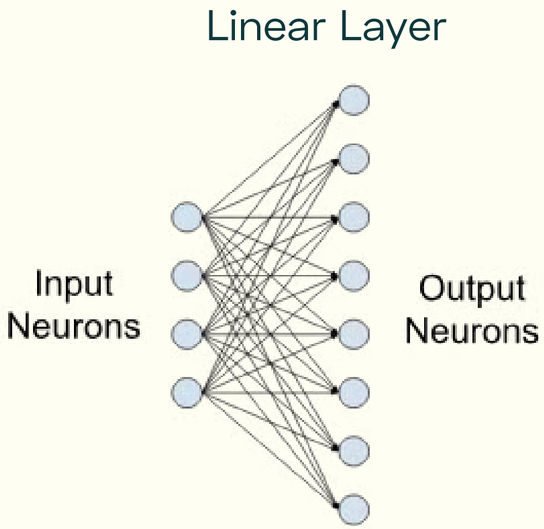


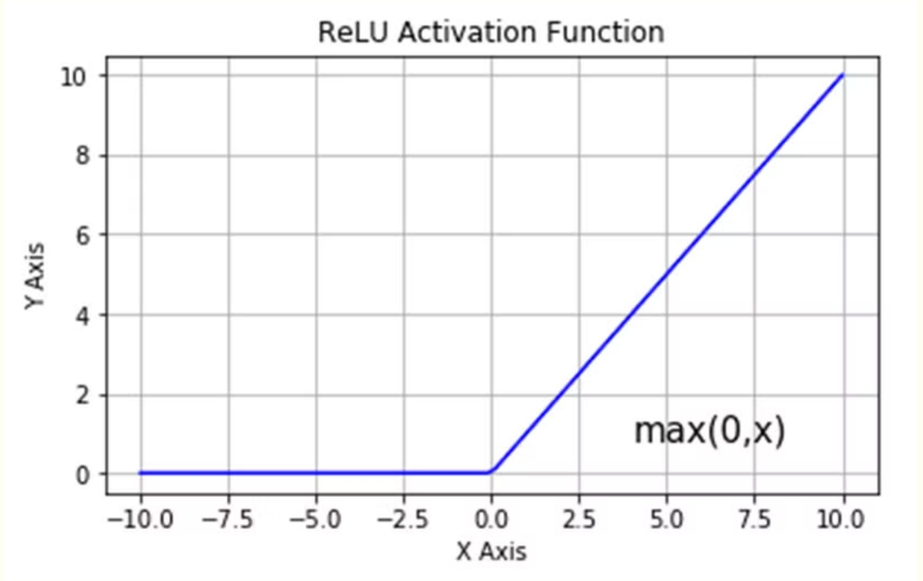
GLOBAL MODEL ARCHITECTURE



Layer stack 1 (1..3)

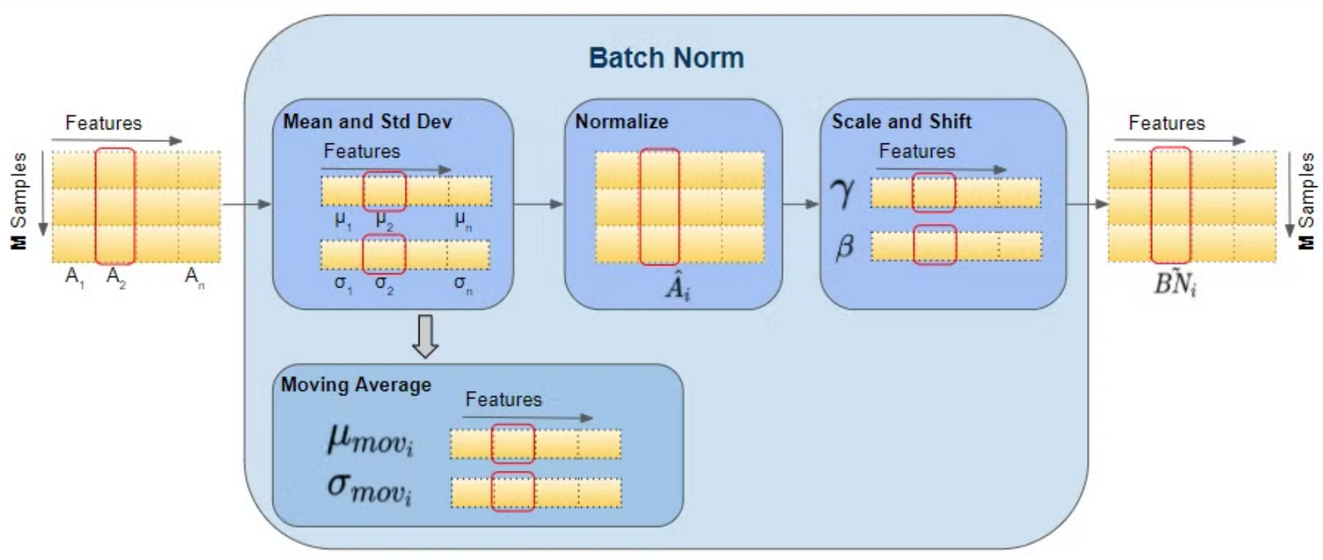
GLOBAL MODEL ARCHITECTURE



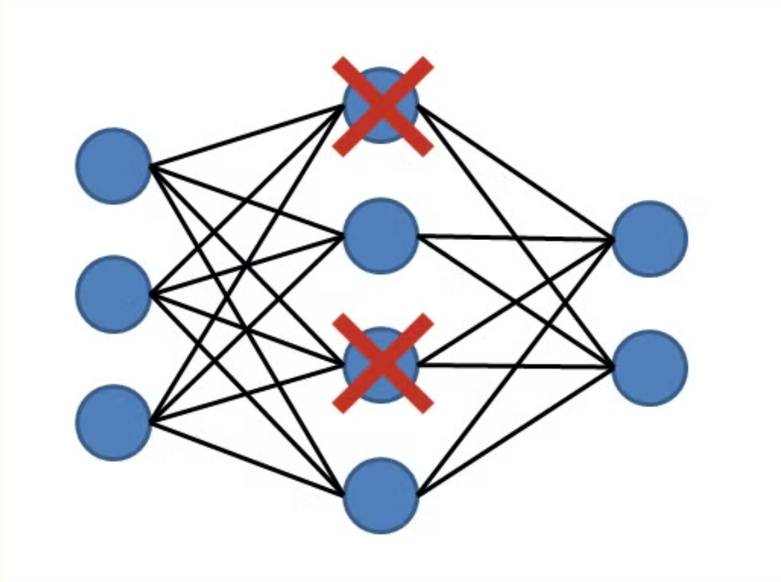


GLOBAL MODEL ARCHITECTURE

Batch Normalisation Layer



Droupout Layer

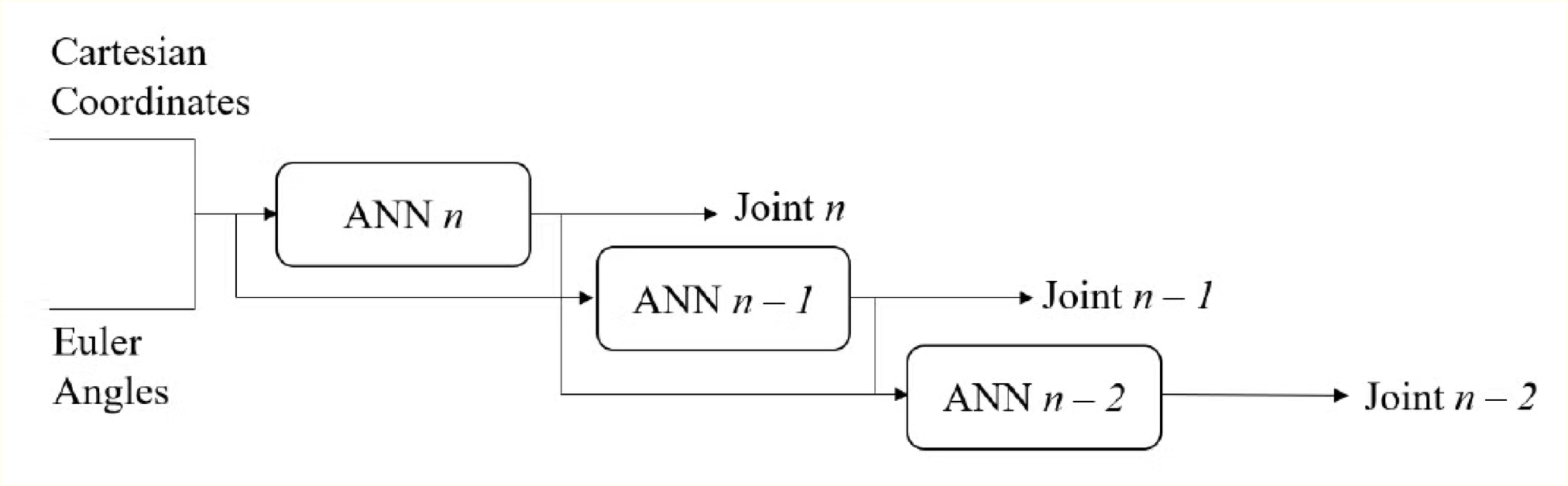


GLOBAL MODEL ARCHITECTURE (IN CODE)

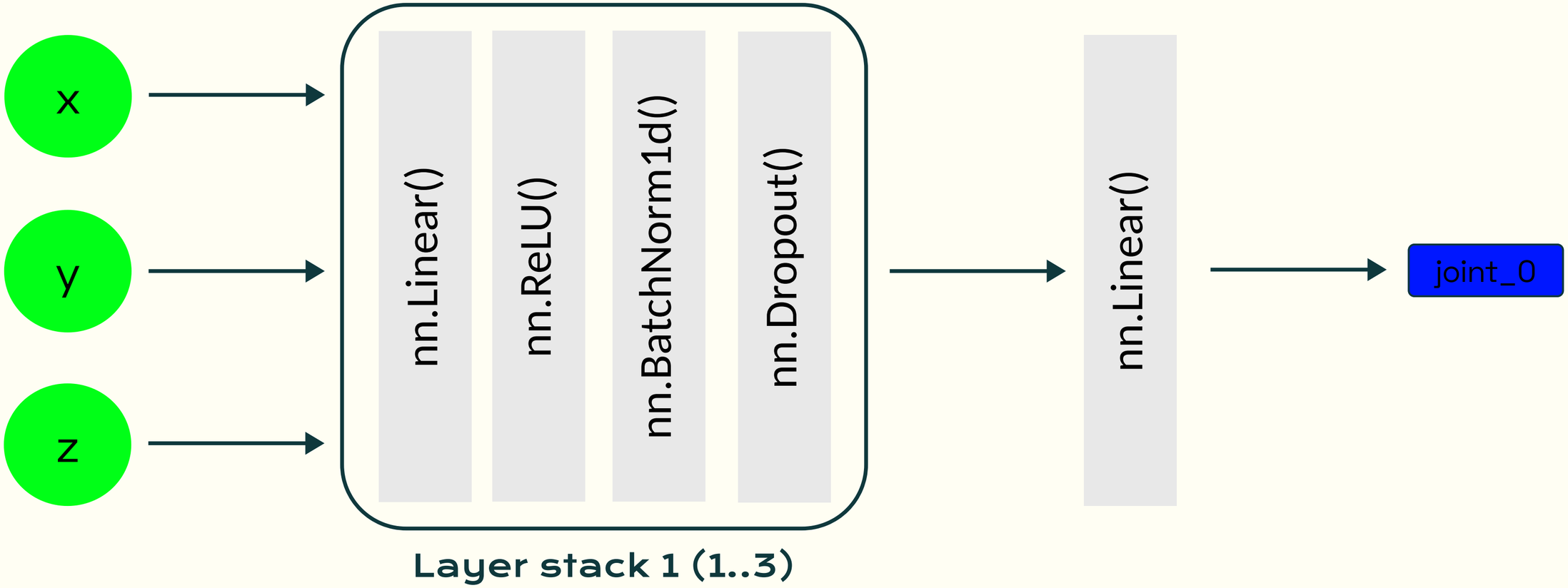
```python
def __init__(self, input_features, output_features):
    super().__init__()
    self.layer_stack_1 = nn.Sequential(
        nn.Linear(in_features=input_features, out_features=128),
        nn.ReLU(),
        nn.BatchNorm1d(num_features=128),
        nn.Dropout()
    )
    self.layer_stack_2 = nn.Sequential(
        nn.Linear(in_features=128, out_features=128),
        nn.ReLU(),
        nn.BatchNorm1d(num_features=128),
        nn.Dropout()
    )
    self.layer_stack_3 = nn.Sequential(
        nn.Linear(in_features=128, out_features=128),
        nn.ReLU(),
        nn.BatchNorm1d(num_features=128),
        nn.Dropout()
    )
    self.layer_stack_4 = nn.Sequential(
        nn.Linear(in_features=128, out_features=128),
        nn.ReLU(),
        nn.BatchNorm1d(num_features=128),
        nn.Dropout(),
        nn.Linear(in_features=128, out_features=output_features)
    )
```

```python
def forward(self, x):
    x = self.layer_stack_1(x)
    x = self.layer_stack_2(x)
    x = self.layer_stack_3(x)
    x = self.layer_stack_4(x)
    return x
```

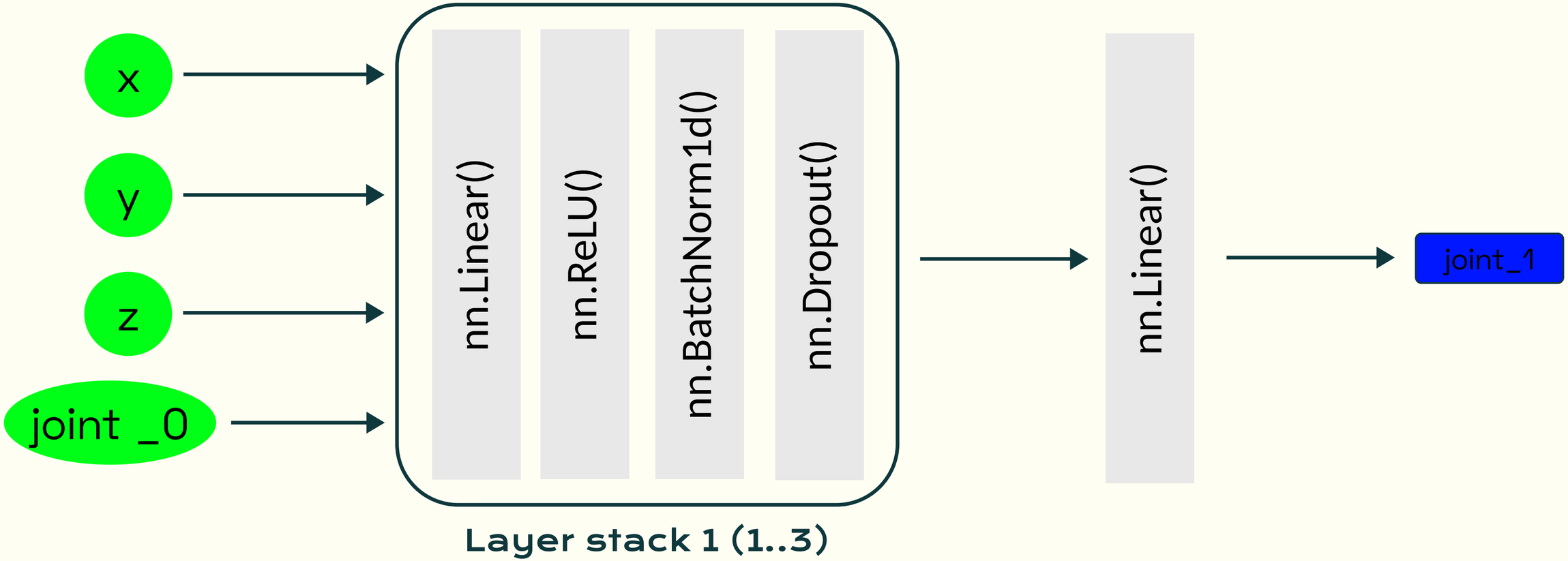
SEQUENTIAL MODEL ARCHITECTURE



SEQUENTIAL MODEL ARCHITECTURE



SEQUENTIAL MODEL ARCHITECTURE



LOSS FUNCTION AND OPTIMIZER USED

LOSS FUNCTION USED nn.MSELoss()

OPTIMIZER USED nn.Adam(params, lr=0.001)

RESULTS

GLOBAL MODEL

1. MSE Loss : 0.1127
2. L1 Loss : 0.2747

SEQUENTIAL MODEL

1. MSE Loss : 0.0620
2. L1 Loss : 0.1975

IK Simulation:

Actual Image

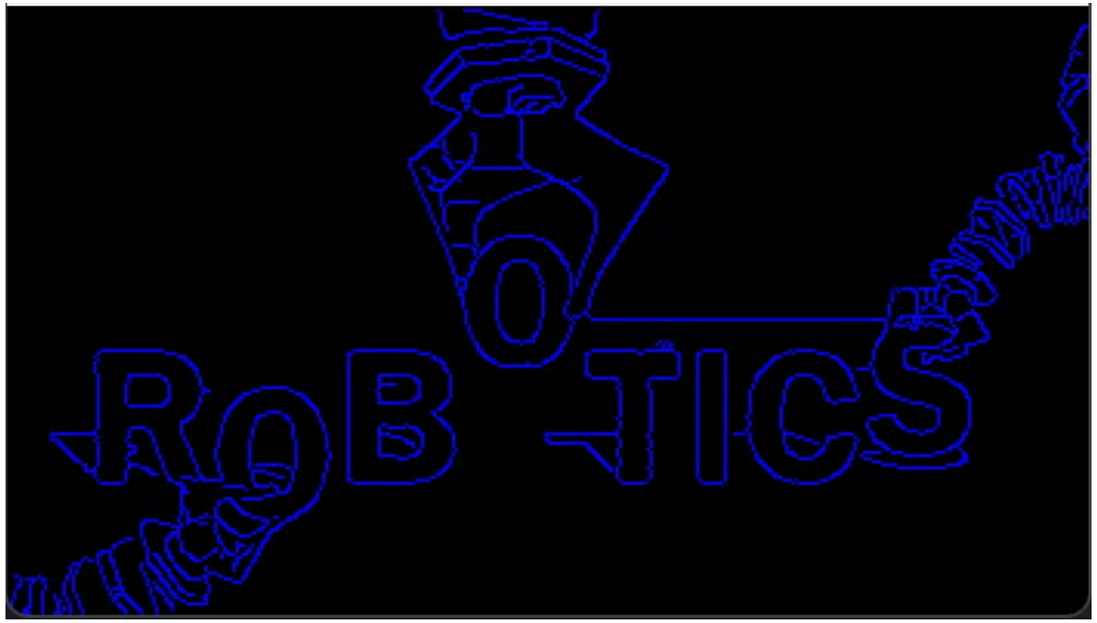


Image After IK

IK Simulation:

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)