Problem 5: IKP with MLP (40 points)

The objective of this question is to develop a deep neural network model that will address the inverse kinematics problem for the planar 3-DOF manipulator depicted in Fig. 4.

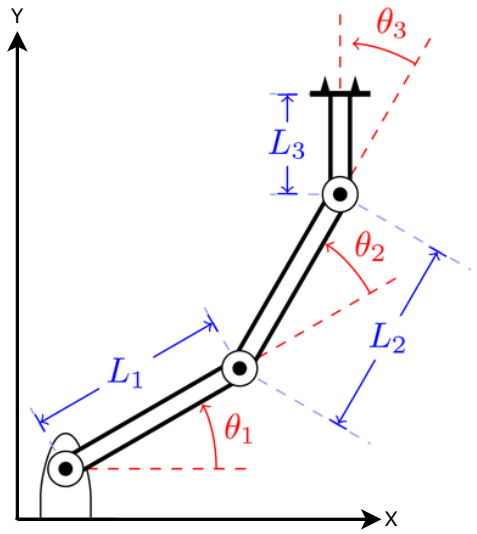

Figure 4: 3-DOF planar robot

In order to train the neural network, a large dataset of joint angles and their corresponding end effector positions and orientations is necessary. Applying the robot's forward kinematic equations, we will generate this dataset. Subsequently, we will train a neural network on the generated data and evaluate its efficacy on unseen test data. In order achieve the ultimate objective, follow to the steps described below.

a. **forward kinematics of robot**
Consider Q1 of Homework 4, where you calculated FKP for robot:

$$\begin{aligned} x &= L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ y &= L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \\ \phi &= \theta_1 + \theta_2 + \theta_3 \end{aligned} \tag{1}$$

b. **Data generation**
Utilizing the forward equations and accounting for angle variations ($[-\pi,\pi]$ for θ_1, θ_2, θ_3):

- Develop a program to generate corresponding coordinates (x, y, z) for changes in joint angles. Given 3 joints, the joint space expands to n^3. Discrete joint space and workspace into 20 points along each X-Y axis. Thus, you obtain 20^2 samples encompassing the robot's workspace. Subsequently, every sample point, characterized by its unique joint parameters and End Effector (EE) position.
- Now these positions will be the input of the model and the angles of the joints will be the output of the model. Save these points (input positions, output joints) in a `csv` file as a dataset.

c. **Train and Test model**
Construct a fully connected neural network with the following architecture:

- Two hidden layers, each comprising 50 neurons with ReLU activation functions.
- An output layer consisting of 3 neurons with linear activation.

The training hyper parameters are specified as follows:

- Cost function: Mean Square Error (MSE).
- Optimizer: SGD.
- Learning rate: 0.01.
- train data: 80%, test data: 20%.
- Number of epochs: 200.

plot the loss diagram for the training data.

d. **Comparison**
Compute the result of Network in time step = 0 with abtained result in Q1 of Homework 4.

Problem 5 – IKP with MLP

I generated 400 random data points (20 for each theta), saved in an excel file you can find attached.

Then I used this data to train my model. The training loss per epoch plot is shown below. It has a reasonable form. Increasing as I expected.

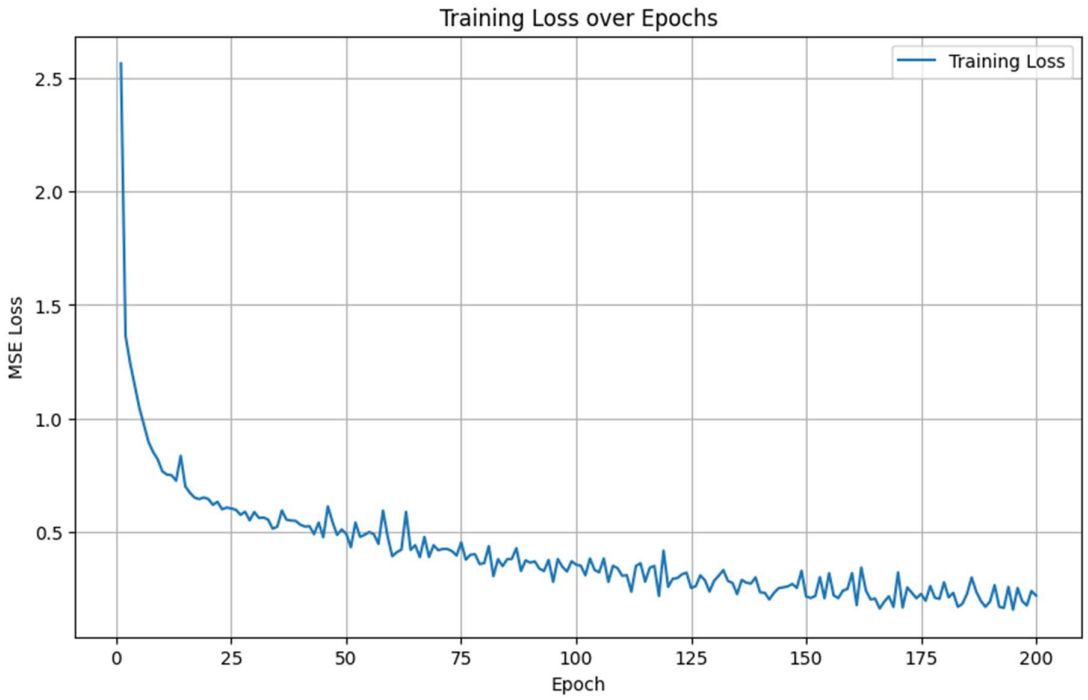

Figure 5-1: Training loss for IKP problem

While doing this exercise I learned changing the learning rate plays a huge role in CNN problems.

I guess 200 epoch is reasonable but for the comparison part, my answers differ.

```
Predicted joint angles:
Theta1: 1.3290
Theta2: -1.2432
Theta3: -2.1534
```

```
Theta1: 0.7854 radians
Theta2: 0.6981 radians
Theta3: -0.3491 radians
```

I thought and searched a lot about how to solve this problem and how can we have a better nueral network. I also did this assigment with more hidden layers for better understanding, but the problem still exists. Meaning, the answer is now close, but not the same as is should be.

I want to briefly explain what additional job I've done and what I've learned:

- I tried larger hidden layer sizes to see if the performance improves. Sometimes deeper networks or wider can capture more complex relationships.

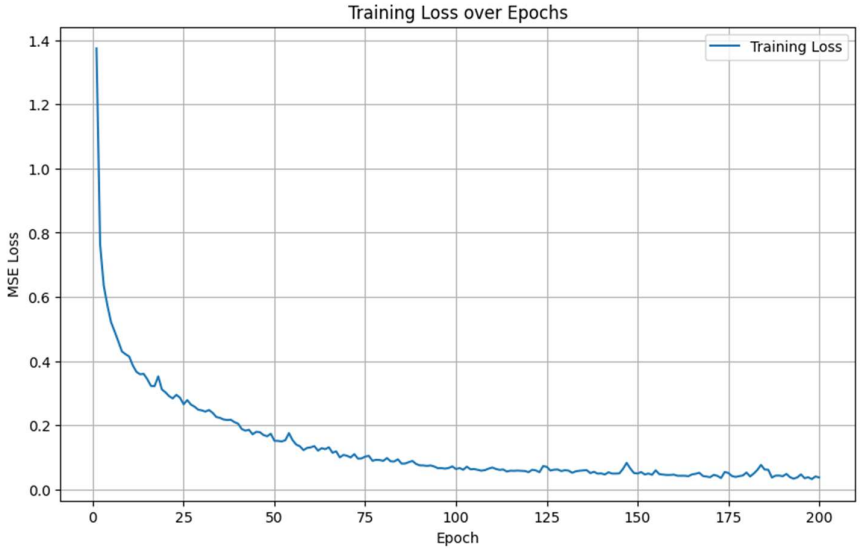

Figure 5-2: Training loss plot for 64 hidden layer size IKP, MLP problem

- I tried different learning rates such as 0.001 and 0.005. I learned tuning this parameter may lead to better convergence, but it's not certain.
- My batch size was 16, didn't really changed it.
- I tried another optimizer, named ADAM.

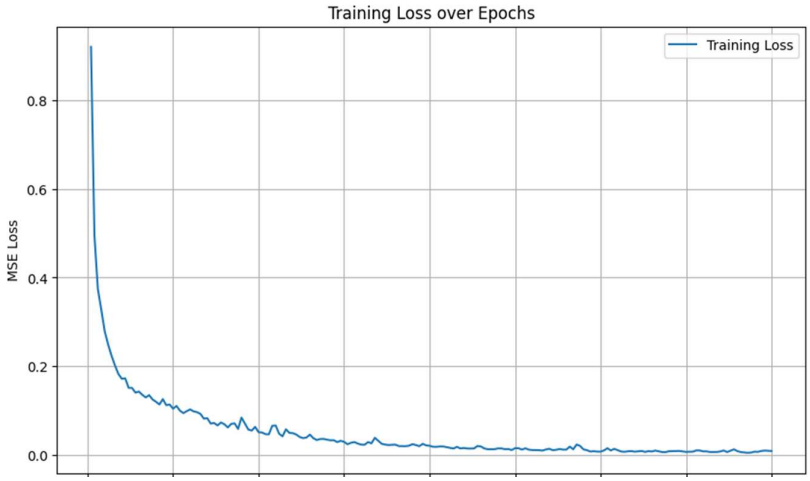

Figure 5-3: Training loss plot for IKP, MLP problem with Adam optimizer

- As mentioned, training for 200 epochs is a good starting point, but here we didn't monitor both training and test losses. We may face overfitting after more epochs, I guess overfitting can be one of the reasons my approach doesn't estimate properly.

My notebook is named "IKP.ipynb"

My additional wok notebook is named "IKP_additional.ipynb"

Generated data csv file is named "ikp_dataset.csv"