University of Tehran
College of Engineering
School of Electrical and Computer Engineering (ECE)
School of Mechanical Engineering (ME)

Mechatronics & Robotics

Homework 6:
Deep learning in Robotics

Teaching Assistants:

Zeynab Ezzati

Deadline: 14 June 2024 (25 Khordad), 23:59

List of Problems

Problem 1: Back Propagation (15 points)

Back propagation is a common method for training neural networks. The main goal of this method is to optimize the weights of the network for the correct mapping of input to output. In this question, we want to check how to calculate it for a simple network.
According to Fig. 1, the network includes an input layer, a hidden layer, and an output layer; each layer includes two neurons. Also, the hidden layer and output include bias and the sigmoid activation function. To minimize errors during propagation, the cost function is the sum of squared errors (SSE). The initial values of weights, biases, input values, and target outputs are shown in Fig. 1.

a. Write the forward path for the network and calculate total error.

b. Update the $W5$ and $W4$ weights on the backward path.

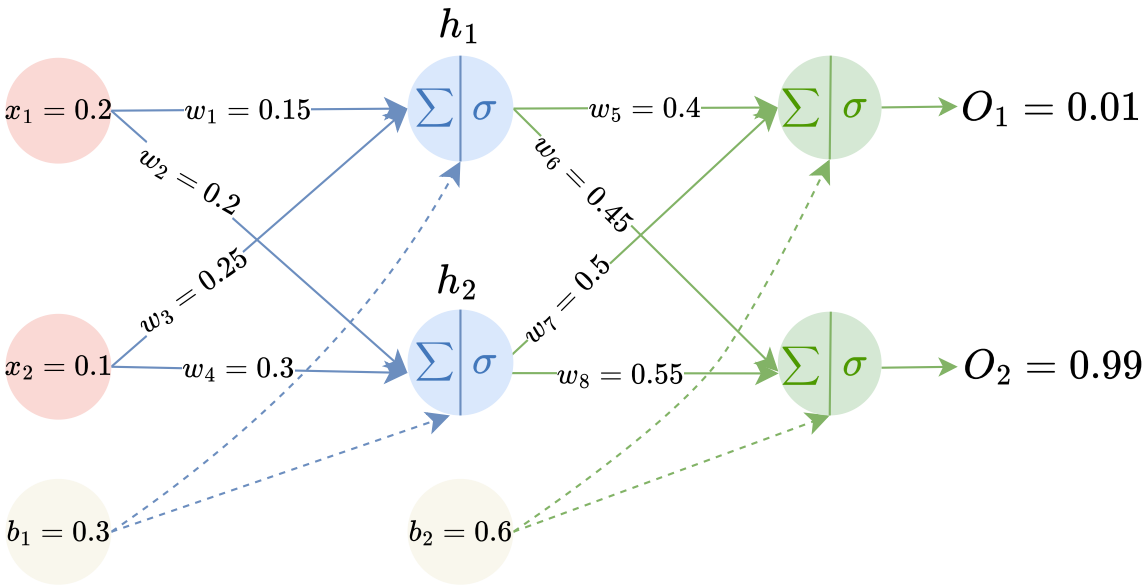

Figure 1: MLP Network

Problem 2: Perceptron Learning (10 points)

a. Create a perceptron manually to classify the given data with proper weight values. Use your intuition of geometry to choose appropriate weights.

Training Example	x_1	x_2	Class
a	1	-1	-1
b	0	2	-1
c	2	1	+1

b. Implement the Perceptron Learning Algorithm on the provided data, with a learning rate of 0.5 and initial weight values of:
$w_0 = -1$
$w_1 = 1$
$w_2 = 0.5$

Provide the updated weight values after each training step in your response.

Problem 3: SVM (15 points)

Consider positive and negative data as Fig. 2:

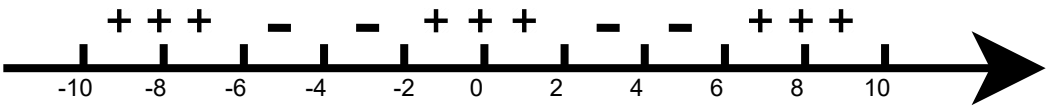

Figure 2: Samples.

a. If we decide to solve this problem with SVM, is it linear separable or nonlinear separable? What is solution of SVM for this problem?

b. If this function is offered to solve this problem,

$$K(x_1, x_2) = \cos\left(\frac{\pi}{4}x_1\right)\cos\left(\frac{\pi}{4}x_2\right) + \sin\left(\frac{\pi}{4}x_1\right)\sin\left(\frac{\pi}{4}x_2\right).$$

find ϕ function.

c. Display the transformed points of the problem using the answer of part b and determine the support vectors.

d. Report the equation of the separator line of the points and display it on the coordinate axis.

Problem 4: CNN forward process (20 points)

In this question, we aim to perform a forward step and use mathematical relations (without simulation) in accordance with the structure defined for a convolutional neural network.

As you can see in Fig. 3, the network has a 2 channel 3×3 image.

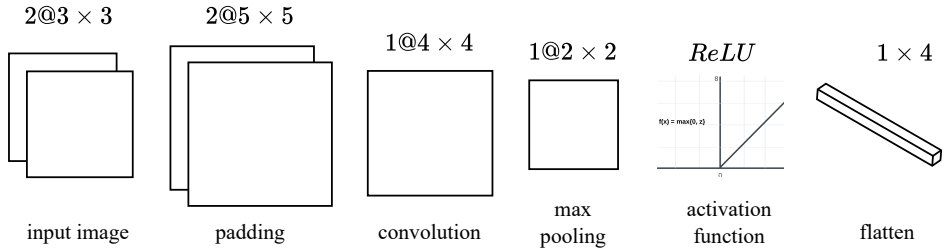

Figure 3: CNN Network

a. To increase the dimensions of the input image in the first step, use padding = 1. Then the size of each 2 channel will be 5×5.

b. Convolute the resulting representation using a 2×2 convolutional filter with *stride* = 1. The output dimensions will be 4×4 single channel.

c. Then do max pooling with a 2×2 filter and *stride* = 2. The output will be 2×2.

d. Pass the output of the previous step through a non-linear ReLU function.

e. Flatten this representation and get a 4×1 feature vector.

f. With the given 4 weights and obtained feature vector, find the final output of the network.

For access to input values and weights of the network layers, you should first run the attached python file (`init.ipynb`). You must give the last 3 digits of your student number as input to the function and call it. The raw input and all the required weights and biases will be generated.

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.

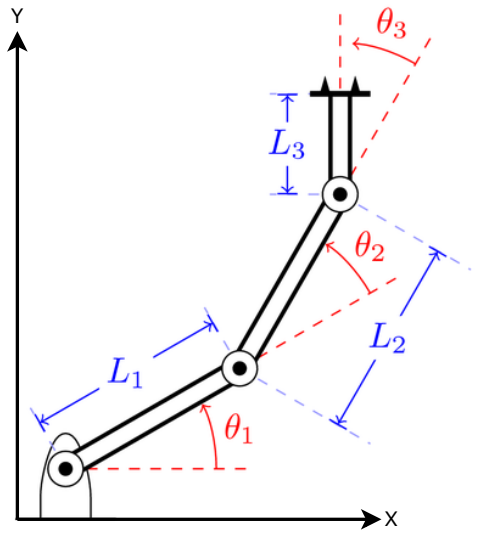

Figure 4: 3-DOF planar robot

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 6: CNN Classification (40 point Bonus + Reward)

The purpose of this exercise is to familiarize you with convolutional networks and how to do an image classification practically from start to finish. The main task is to classify 5 classes of robots using CNNs.

Note: In order to ensure that results are fair with respect to the hardware capabilities of the students, all of the questions in this question must be implemented in **Colab** with **PyTorch** library.

Note: A cash **reward of 2,000,000 Tomans** has been considered for **one** student with the most suitable report in terms of network accuracy, quality, appearance, and the completion of all parts.

To do this exercise, do the following steps:

Step 1: Data Collection and Preparation

Data Collection

The goal is to categorize class of robots based on their images. Therefore, we need images of various robots. We consider five categories of robots that are not difficult to recognize (Fig. 5):

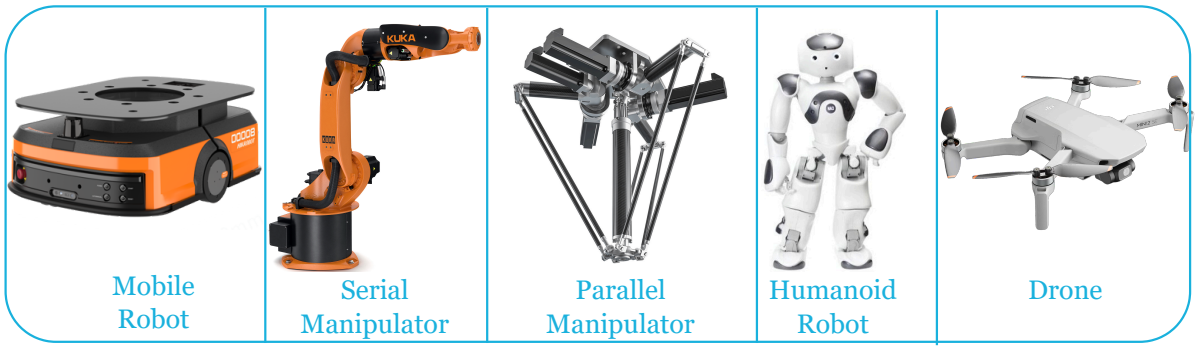

Figure 5: Five classes of robots

In the end, you will have five classes. For each of these classes, collect a sufficient number of images from the internet and other sources. The number of images is up to you, but keep in mind that, based on what you have learned so far, the number of images significantly affects the learning of the network and improves accuracy. Aim for a balanced dataset with a reasonable number of images per class.

Data Augmentation and Preprocessing

- After collecting the images, label your images.

- Utilize data augmentation techniques such as rotation, flipping, scaling to increase dataset diversity. At this stage, mention which augmentation methods you used and why.
- As you know, input images for a CNN network must be the same size. Your dataset may consist of images with different dimensions. Resize all images to a uniform size suitable for model input, and normalize them.
- Split the dataset into training, validation, and test sets. A common split could be 70% training, 15% validation, and 15% test.

You can do all of these tasks on sites like Roboflow or you may do them manually with an algorithm or library of your choice.

Step 2: Model Architecture Design

The model's construction is split into two parts. The first part is a CNN backbone that finally creates a good feature vector from the images, and the second part is an MLP network that is used for final classification with the appropriate number of outputs, such as Fig. 6.

CNN Backbone

Design a custom CNN architecture for feature extraction with:

- Convolutional layers
- batch normalization
- activation functions
- pooling layers
- residual blocks (if needed)
- any other type of layers depending on requirements.

Only limitation of this part is to avoid using pre-trained backbones.

MLP Classifier

Implement the MLP layer on top of the CNN backbone for classification. Design the MLP with appropriate hidden layers and output units corresponding to the number of classes (5 in this case). To prevent overfitting, you can incorporate dropout for regularization.

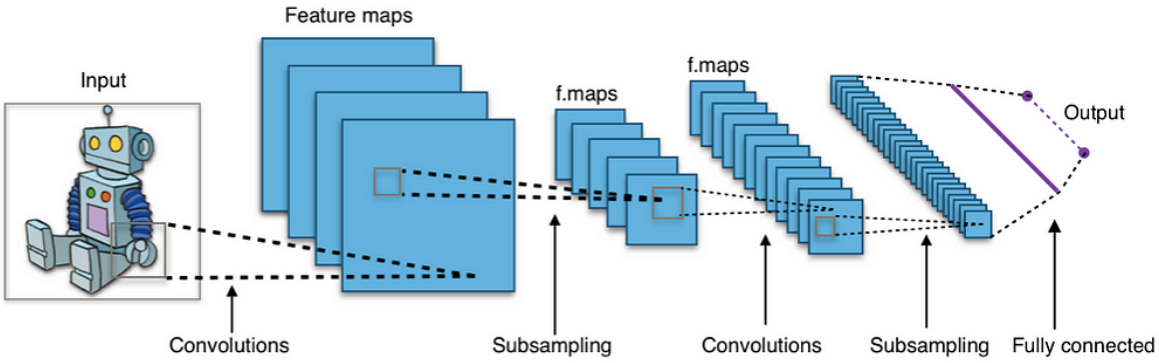

Figure 6: Model Architecture.[1]

Step 3: Model Training and Evaluation

Hyperparameter Tuning

Train the model with your chosen hyperparameters such as:

- learning rate
- batch size
- number of epochs
- using or not momentum
- initial weights
- appropriate cost function for classification

Performance Metrics

- Plot training and validation loss/accuracy curves and analyze model based on them.
- Evaluate the trained model on the test set to assess its generalization ability.
- Select 7 random samples from the test set and visualize model predictions and true labels.
- To validate and compare your model to others, we created a benchmark with four images from each class that are not accessible to you. Write a program that follows the benchmark's path and then calculates the model's accuracy. You have the option of attaching a `validation.py` file or assigning one of the Colab cells to this task.
  Note: Keep in mind that you will need to know how images are labeled. For this issue, we have included a sample image of each class. The image's name identifies its label. This sample file is attached to homework.

Step 4: Building GUI

Construct a graphical user interface (GUI) that, in accordance with your trained model, predicts the label of any desired image obtained from the user.

Homework Guidelines and Instructions

- The deadline for sending this exercise will be until the end of Friday, June 14th.
- This time cannot be extended and you can use time grace if needed.
- The implementation must be in Python programming language and your codes must be executable and uploaded along with the report.
- This exercise is done by one person.
- If any similarity is observed in the work report or implementation codes, this will be considered as fraud for the parties.
- Using ready-made codes without mentioning the source and without changing them will constitute cheating and your practice score will be considered zero.
- If you do not follow the format of the work report, you will not be awarded the grade of the report.
- Handwritten exercise delivery is not acceptable.
- All pictures and tables used in the work report must have captions and numbers.
- A large part of your grade is related to the work report and problem solving process.
- Please upload the report, code file and other required attachments in the following format in the system: `HW6_[Lastname]_[StudentNumber].zip`
  For example, the: `HW6_Ezzati_12345678.zip`
- If you have questions or doubts, you can contact the assistants through the following e-mail with the subject HW6-Mechatronics. Stay in touch educationally:
  - `Ezzati.z@ut.ac.ir` (Zeynab Ezzati)
- Be happy and healthy