Deep Learning

Homework 3: CNN Implementation

Announced: December 24
Deadline: January 12 at 11:59 pm (iLMS)
TA:

蘇翁台 (wengtai2008@hotmail.com)
林辰安 (strike860930@gmail.com)
陳冠宇 (seed0123456@gmail.com)

Introduction:

In this problem, you need to build a convolutional neural network in order to classify images into three classes. Particularly, you need to code on the feedforward network structure and error backpropagation algorithm by yourself in order to realize the CNN learning process, instead of using related functions in MATLAB or Python that do the tricks for you.

Rules:

- Built-in machine learning libraries (Sklearn, PyTorch, TensorFlow…) are not allowed to use for CNN, you need to use basic mathematical operations in NumPy to define the behavior of each layer.

- Please properly comment your code to let us understand your train of thought.

- Discussions are encouraged, but plagiarism is strictly prohibited and punishable!

Submission:

You should compress your code, data, report (hw3report_studentID.pdf) and readme.txt (explain how to run your code) into a ZIP file (hw3_studentID.zip), and submit it on iLMS before the due date.

Data:

Data, encapsulated in Data.zip, in this problem are the images of three fruits, which are respectively deposited in the three folders Carambola, Lychee, and Pear under two folders for training and testing.

Implementation [90%, 10% for each part]

1. Fruit dataset reading & splitting: You need to split the original training data into validation data and training data. (proportion: 3/7)
2. 2D Convolution layer: Implementation of learnable kernel matrices and convolution operation.
3. ReLU layer: Nonlinear activation function.
4. FC layer: Every output neuron has full connection to the input neurons.
5. Softmax output: The final layer is typically a Softmax function which outputs the probability of a sample being in different classes.
6. Cross-entropy loss calculation: Use the output of a batch of data and their labels to calculate the CE loss.
7. Backward propagation: Propagate the error backwards and update each parameter.
8. Validation: Test the performance on validation data during each epoch. 9. Testing accuracy: Reach 97% to get 10 points, 94~97% -> 7 points, 90~94% -> 5 points, zero otherwise.

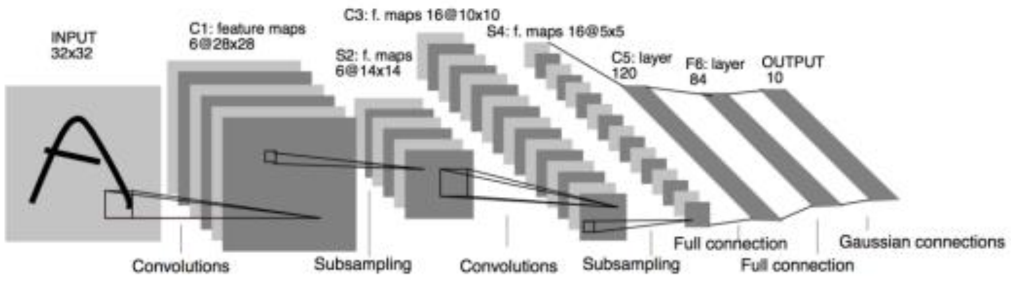

Figure 1. Architecture of LeNet, you can modify it as you wish.

Report [10%]

The format is not limited, but the following matters must be discussed in your report:

1. Show your model architecture, loss function and screenshot of testing result.

2. Plot training loss and validation loss.

3. Describe the major problem you encountered and how did you deal with it.