011110COM 526000 Deep Learning
Homework 3: CNN Implementation

Announced: 2022 /12 /05 (一)
Deadline: 2023 /01 /06 (五) at 23:59 PM
TA:

張君豪 (howard88315@gmail.com)

Introduction:

In this homework, you need to build a convolutional neural network for image classification. You need to implement the feedforward network and backpropagation in order to realize the CNN learning process. Particularly, you can utilize any related functions in Python and **PyTorch,** which are recommended to be applied to this problem.

Rules:

- Built-in machine learning libraries (ex. Sklearn, PyTorch …) **are allowed** to use for CNN.
  Note: The framework please use pytorch, instead of keras, tensorflow.
- Please properly comment your code to let us understand your train of thought.
- Discussions are encouraged, but plagiarism is strictly prohibited.

Data:

The Finger signs dataset, encapsulated in Dataset.zip, has images of fingers with six classes (zero to five) as shown in Figure 1. Images and labels are saved in files called ‘Signs_Data_Training.h5’ and ‘Signs_Data_Testing.h5’’. The dataset contains 1080 and 120 images for training and testing, where the size of each image is 64×64.

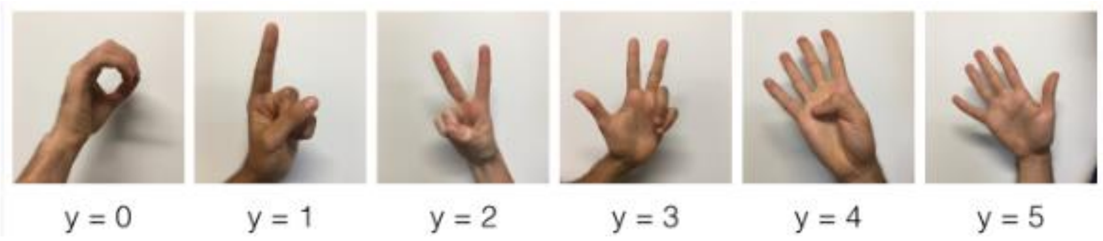

Fig. 1: Illustration of the data set containing six finger signs. You need to build a neural network that is capable of performing classification.

Implementation [50%]:

1. Read the finger signs dataset and convert the numpy array to tensor. (5%)
2. Implement 5-fold cross-validation to create 5 training and validation sets. (10%)
3. Design your own CNN model for finger sign image classification. (10%)
   (We show a simple CNN model in Fig. 2.)
4. Choose optimizer and loss function. (5%)
5. Final testing accuracy (your own CNN model): (20%)
   Reach over 93% to get 20 points, 90%~93%: 15 points, 88%~90%: 10 points, 60%~88%: 5 points, otherwise: zero.
   Note: Please remember to save your model weights (.pth or .pt) and write code to load your weights during testing.

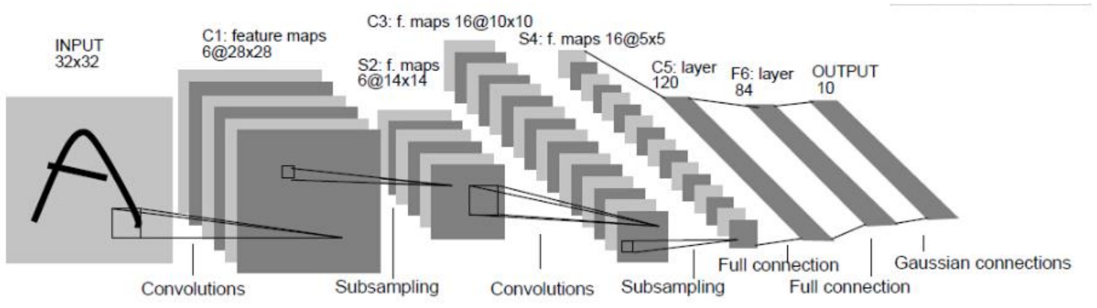

Fig. 2: Architecture of LeNet, you can modify it as you wish

Submission:

Please put all files below into a folder, then compress the folder into a **ZIP** file (hw3_studentID.zip) and finally upload the **ZIP** file on eeclass before the deadline.

- Code (.py)
- readme.txt (Explain how to run your code, environment, and libraries in detail.)
- Report (hw3report_studentID.pdf)
- Training weights (model_weights.pth or .pt)
- We will reproduce your results based on codes, weights, and readme files. Please make sure we can reproduce the results, otherwise, you would not get the scores.

Report [50%]:

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

1. Show your model architecture, loss function, and the final testing accuracy. Please briefly explain the structure of the model and the motivation of the designed architecture. (10%)

2. Show the training and validation accuracy of each fold (5-fold in total) and explain the advantage of cross-validation. (10%) (Only need to show the training and validation accuracy of 5-fold of the best model on the test set)

3. In the training process, you need to adjust hyperparameters (epochs, batch size, …). How do you adjust hyperparameters according to the cross-validation? Please conduct experiments (in **Table** form) to verify it. (10%)

4. By using 5-fold cross-validation, you would obtain 5 models after the training process. How do you obtain a final model for testing? Please describe how you choose the final model for testing. (10%)

5. Instead of training from scratch, we could also utilize pre-trained models to achieve better performance. Please utilize at least one **pre-trained model** to train, including AlexNet, VGG-16, ResNet-18, ResNet-34, etc. Show the results of using pre-trained models and explain the advantage of using pre-trained models. (Please compare the pre-trained models with your own architecture.) (10%)