CSCE 421: Machine Learning (Spring 2025)

Homework #4

Due 4/8/2025, 11:59 pm

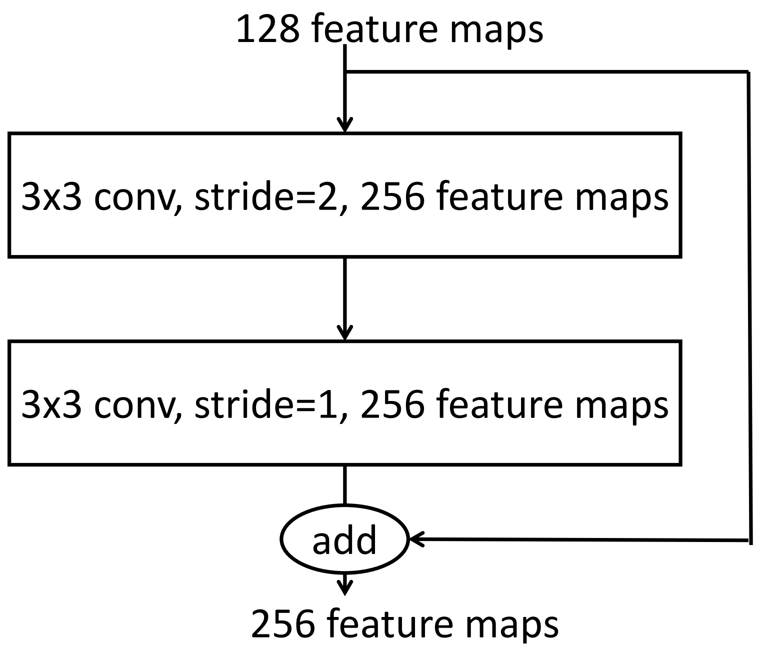
1. You need to submit (1) a report in PDF and (2) your code files, both to Canvas.
2. Your PDF report should include (1) answers to the non-programming part, and (2) results and analysis of the programming part. For the programming part, your PDF report should at least include the results you obtained, for example the accuracy, training curves, parameters, etc. You should also analyze your results as needed.
3. Please name your PDF report "HW#_FirstName_LastName.pdf". Please put all code files into a compressed file named "HW#_FirstName_LastName.zip". Please submit two files (.pdf and .zip) to Canvas (i.e., do not include the PDF file into the ZIP file).
4. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE
5. LFD refers to the textbook "Learning from Data".
6. All students are highly encouraged to typeset their reports using Word or LaTeX. In case you decide to hand-write, please make sure your answers are clearly readable in scanned PDF.
7. Unlimited number of submissions are allowed and the latest one will be timed and graded. If you make a resubmission after the due date, it will be considered late.
8. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.
9. Please start your submission to Canvas at least 15-30 minutes before the deadline, as there might be latency. We do NOT accept E-mail submissions.

---

1. (10 points) A single $(15 \times 15 \times 3)$ image is passed through a convolutional layer with 28 filters, each of size $(3 \times 3 \times 3)$. The padding size is 1 (1 unit at top, bottom, left, and right) and the stride size is also 1. What is the size of the output feature map volume? What is the number of parameters in this layer (including bias)? Note that, for simplicity, we consider filters of $3 \times 3 \times 3$ as one filter, instead of three filters of size 3×3.
2. (20 points) In this question, you can (a) assume padding of appropriate size is used in convolutional layers, and (b) ignore batch normalization. Given the residual block as below:


128 feature maps
3x3 conv, stride=2, 256 feature maps
3x3 conv, stride=1, 256 feature maps
add
256 feature maps

(a) (10 points) What projection shortcut operations are required on the skip connection?

(b) (10 points) What is the total number of trainable parameters in the block (you can ignore bias terms, but need to consider the skip connection)?

3. (20 points) Using batch normalization in neural networks requires computing the mean and variance of a tensor. Suppose a batch normalization layer takes vectors $z_1, z_2, \cdots, z_m$ as input, where m is the mini-batch size. It computes $\hat{z}_1, \hat{z}_2, \cdots, \hat{z}_m$ according to

$$\hat{z}_i = \frac{z_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

where

$$\mu = \frac{1}{m}\sum_{i=1}^{m} z_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \mu)^2.$$

It then applies a second transformation to obtain $\tilde{z}_1, \tilde{z}_2, \cdots, \tilde{z}_m$ using learned parameters γ and β as

$$\tilde{z}_i = \gamma \hat{z}_i + \beta.$$

In this question, you can assume that $\epsilon = 0$.

(a) (5 points) You forward-propagate a mini-batch of $m = 4$ examples in your network. Suppose you are at a batch normalization layer, where the immediately previous layer is a fully connected layer with 3 units. Therefore, the input to this batch normalization layer can be represented as the below matrix:

$$\begin{bmatrix} 12 & 14 & 14 & 12 \\ 0 & 10 & 10 & 0 \\ -5 & 5 & 5 & -5 \end{bmatrix}$$

What are $\hat{z}_i$? Please express your answer in a 3×4 matrix.

(b) (5 points) Continue with the above setting. Suppose $\gamma = (1, 1, 1)$, and $\beta = (0, -10, 10)$. What are $\tilde{z}_i$? Please express your answer in a 3×4 matrix.

(c) (5 points) Describe the differences of computations required for batch normalization during training and testing.

(d) (5 points) Describe how the batch size during testing affect testing results.

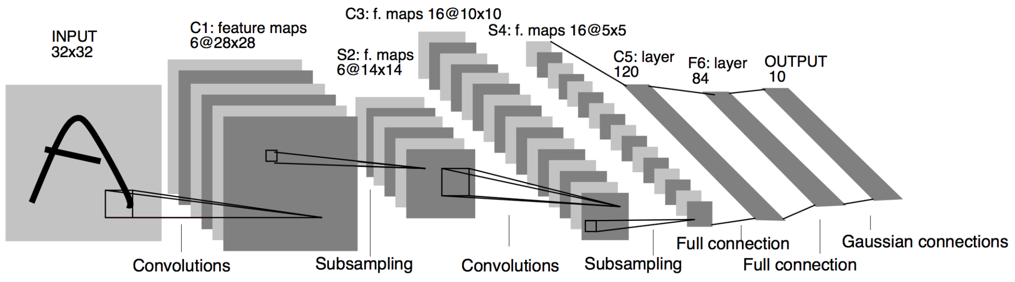
4. (50 points) **LeNet for Image Recognition:**

In this coding assignment, you will need to complete the implementation of LeNet (LeCun Network) using Pytorch and apply the LeNet to the image recognition task on Cifar-10 (10-classes classification). The access to the Cifar-10 Dataset are here (`https://www.cs.toronto.edu/~kriz/cifar.html`). In addition, you will need to install the python packages "tqdm" and "pytorch". The installation guides of PyTorch are in "readme.txt". Please read carefully and follow the instructions. You are expected to implement your solution based on the given codes. The only file you need to modify is the "solution.py" file. You can test your solution by running the "main.py" file.

(a) (10 points) Download and extract the Cifar10 Dataset from the link above. Put the data folder "cifar-10-batches-py" in the same directory of "code". Read carefully the instructions and then complete the function ***load_data()***.

(b) (10 points) Complete the function ***preprocessing()***, you need to implement two ways of preprocessing the data: (1) **rescaling**: rescales the image pixels from range 0-255 to range 0-1; (2) **normalization**: centralize and rescale each image using its mean and variance. There is a parameter *normalize* in the ***preprocessing()*** to control whether to rescale or normalize the images.

(c) (20 points) Complete the class ***LeNet()***. In particular, you need to complete functions ***__init__()*** and ***forward()*** in the class. The paper for LeNet can be found here (http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf) The network architecture is shown in the figure below.



The subsampleing is implemented by using the max pooling. And the kernel size for all the convolutional layers are 5×5. The sequential layers are:

Inputs →
Convolution (6 out channels) → BN → ReLU → Max Pooling →
Convolution (16 out channels) → BN → ReLU → Max Pooling →
Reshape to vector → Fully-connected (120 out units) → BN → ReLU →
Fully-connected (84 out units) → BN → ReLU → Dropout → Outputs
(n_classes out units)

For this part, you are only allowed to use the APIs in ***torch.nn***. Please refer to the PyTorch API documents below for the usage of those APIs before you use them: https://pytorch.org/docs/stable/nn.html

(d) (10 points) Try to read and understand the class ***LeNet_Cifar10()***. Run the main.py to train and test the model. You need to train two models: One using **normalization preprocessing and the default LeNet architecture** as defined in part (c), and another one using **rescaling preprocessing and LeNet without Dropout and Batch Normalization**. Compare the results of the two models and analyze why there is an improvement on (or why it harms) the performance. Include your training log and test accuracy, as well as a short analysis of the results in your report. The training and testing may take 10 - 30 minutes to finish on a CPU (20 epochs).