EEP 596 Homework 2 (100%) due: 10/26 11:59 pm

Submission Instructions

You will need to submit the following materials:

1. **Questions and Answers** in PDF format. Remember to put your name in your report.
2. A folder containing all ipynb files with your results for the the problems. Please remember to keep all results and logs in the submitted ipynb files to get full credit.

All submitted files should be put into one single zip file named as HW#_xxx.zip, e.g. HW2_George_Clooney.zip**,** including all ipynb files with answers (both results and discussions), and the **Questions and Answers** report.

Problem 1: Practices of PyTorch (25%)

In this problem, we will use what we learned from the PyTorch tutorial on class to try some PyTorch's tensor operations.

Problem 2: Digital Image Classification by MLP and CNN (25%)

We used SVM to classify digital images on MNIST dataset. In this problem, we will do classification on MNIST dataset in deep learning approaches.

(a) Prepare MNIST dataset

Same as HW1, download MNIST dataset mnist.mat from the provided google drive link and follow the instruction in HW2.ipynb to load and visualize the dataset.
In this part, you will practice how to load files from your Google Drive to the Google Colab Notebook. To achieve this, you will need to:

- Upload file(s) to your own Google Drive.
- Mount your Google Drive to the corresponding Colab Notebook.
- Find the path of your uploaded file(s) and access them.
- Dowsample the images and split the validation set from the training set (train: 50000, valid: 10000, test: 10000).

(b) Multilayer Perceptron (MLP) (10%)

Use MLP to achieve image classification.

1) Define the following **MLP** using PyTorch to do image classification on MNIST dataset according to Fig. 2:
    - # of hidden layers: 3
    - # of neurons in the layers: [100, 50, 20]
    - Activation functions: ReLU()
    - Dropout for all hidden layers: 30%
    - Output layer + softmax

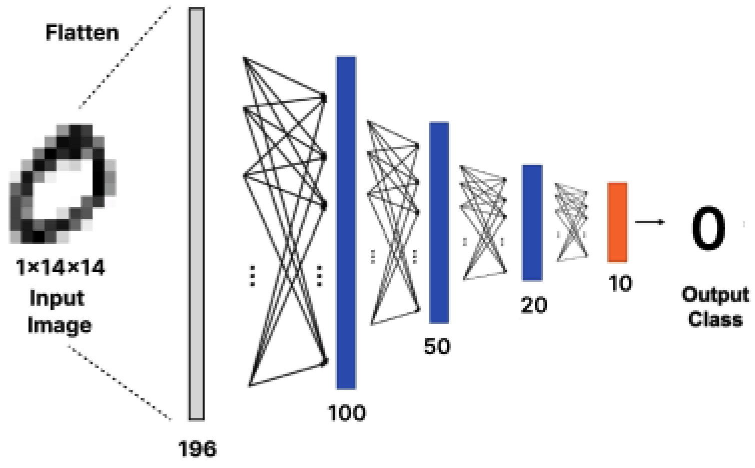


Fig. 2: Architecture of the MLP.

(c) Convolution Neural Network (CNN) (15%)

Built a LeNet-5 using PyTorch to do image classification. Develop the CNN from the PyTorch NN tutorial introduced in class.

Note: The input image dimension shown in Fig. 3 is different from the one we have in this problem. Use kernel size=3 for both convolutions (LeNet-5 uses kernel size=5)

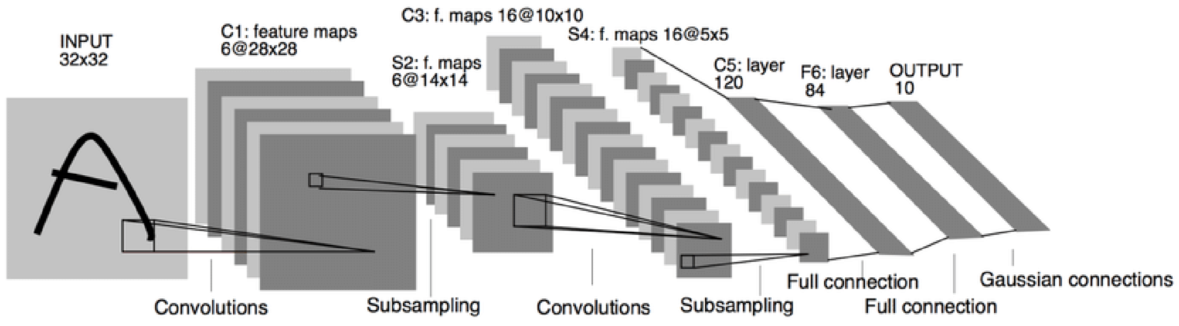


Fig. 3: LeNet-5 architecture

Problem 3: Image Classification by CNN (25%)

In this problem, we will solve a simple image classification problem on the CIFAR-10 dataset. It has the classes: ‘airplane’, ‘automobile’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’, ‘ship’, ‘truck’. The images in CIFAR-10 are of size 3x32x32, i.e. 3-channel color images of 32x32 pixels in size. There are 50000 images in the training set, and 10000 images in the testing set.

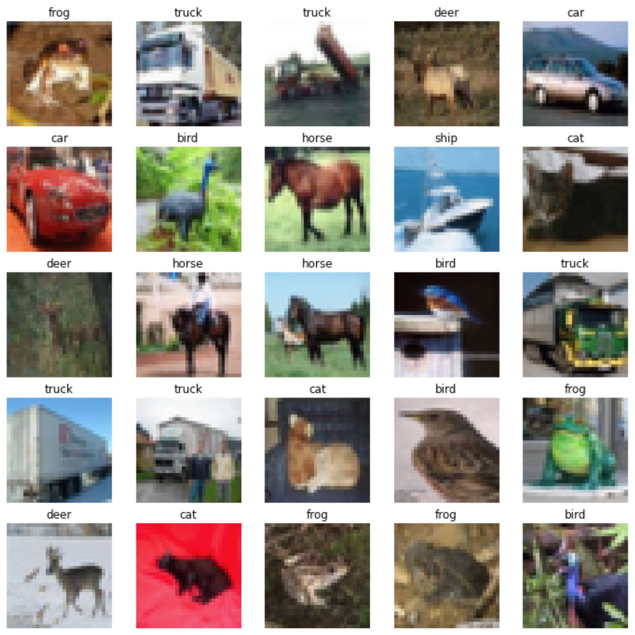


Fig. 1: Some sample images in CIFAR-10 dataset.

(a) First, we use torchvision to load and visualize some built-in datasets. Please write code to **load and visualize the CIFAR-10** dataset. Note that we typically **normalize** the data during the loading process. The code for this transform is written in HW2.ipynb.

(b) Use the following **data augmentation** techniques to the training set:
- Shifting: randomly shift the images up/down and left/right by within 10%.
- Rotating: randomly rotate the images by some angles.
- Flipping: horizontally flip the images.
- Adding Noise: randomly add some small Gaussian noise to the images.

Thus, after the above data augmentation, you will get in total 50000x5 training images. You will use the augmented training images to train the CNN for the next question. (Hint: you can use PyTorch's built-in function for data augmentation.

(c) Define a **CNN** using PyTorch to do image classification. Develop the CNN from the previous PyTorch NN tutorial.

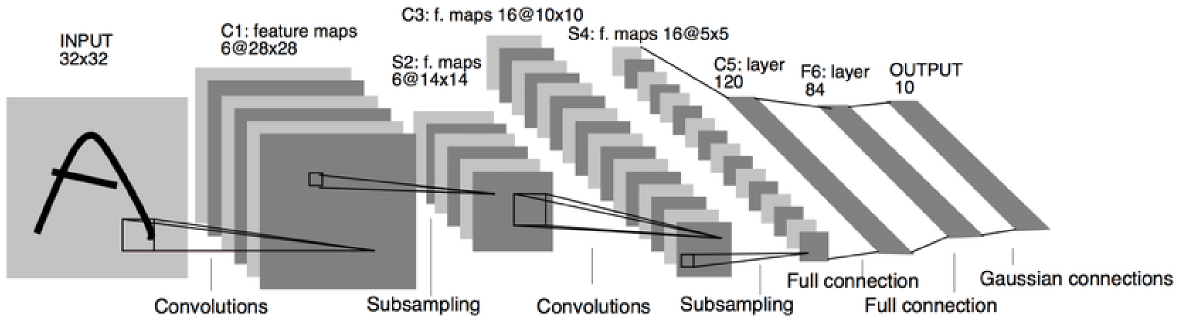


Fig. 3: LeNet-5 architecture for you to start from.

1) Try different network **parameters/configurations** (at least 5 combinations):
    - # of conv: 2, 5, 9
    - Kernel size (conv): 3x3, 5x5
    - Stride (cov): 1x1, 3x3
    - Dilation: 1x1, 3x3
    - Dropout in FC: 0%, 30%

   Please include data augmentation for your CNN. You should try to achieve **at least 80%** accuracy on the CIFAR-10 testing set by using different network configurations (can use other configurations and techniques not mentioned above).
2) Evaluate and report the accuracies using different configurations in a table.

(b) Discussions (5%)

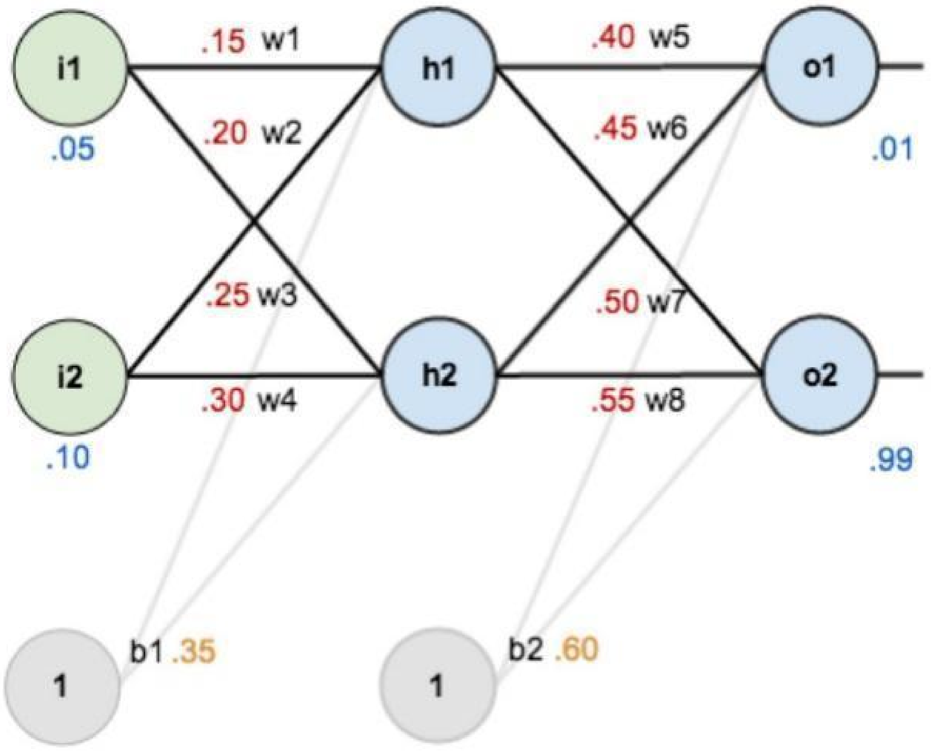
From the results in P2 and P3, discuss:

1) Which framework can achieve higher accuracy, MLP or CNN? Briefly explain the reason.
2) Which parameter can potentially affect your performance most?

Problem 4: Questions and Answers (25%)

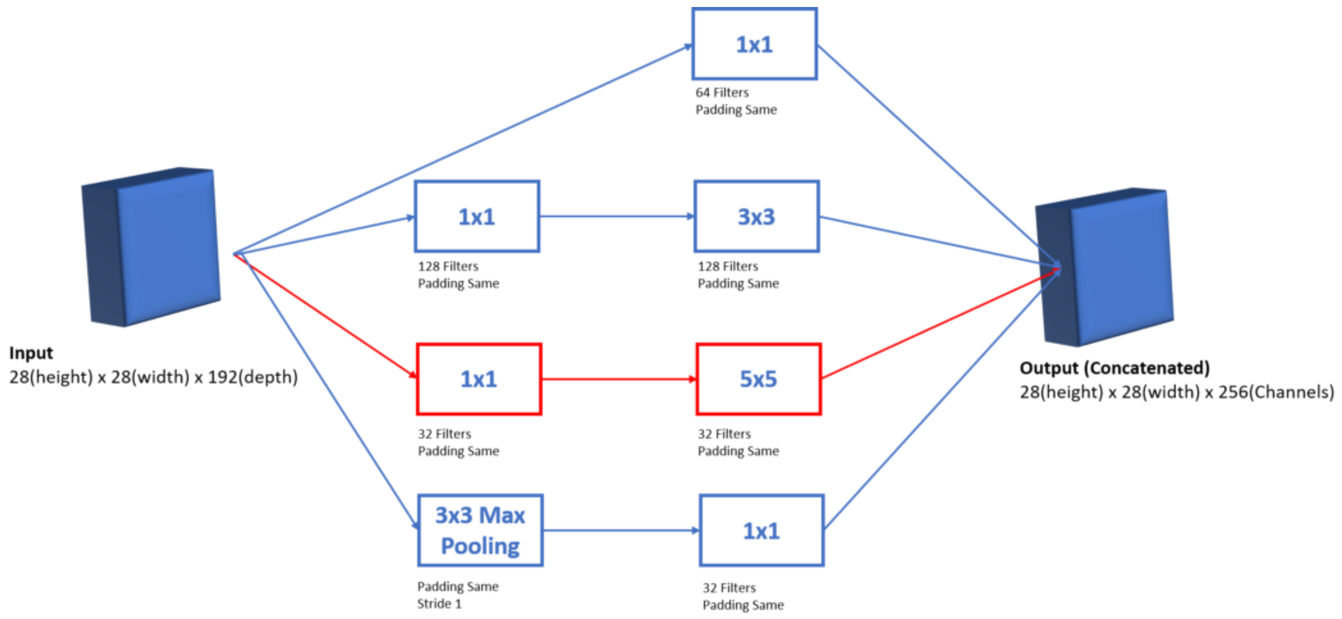
1. (15%) Based on the procedures and the values of computing the MSE minimized backpropagation updated weights w5, please compute the updated value of w2 of the following 2-layer MLP with given initialized weights (in red) and the bias offsets (in orange), where the 2-D inputs are (0.05, 0.10), and the desired/target outputs are (0.01, 0.99). All neurons are assumed to have sigmoid nonlinear activations, and the learning rate is 0.5.

   Note that you need to compute both $\frac{\partial Etotal}{\partial outo1}$ and $\frac{\partial Etotal}{\partial outo2}$ to compute $\frac{\partial Etotal}{\partial outh1}$ before you can compute $\frac{\partial Etotal}{\partial w2}$



Sigmoid function: $$g(z) = \frac{1}{1+e^{-z}}$$

2. (10%) Given one Section of Google Inception CNN architecture shown below, please compute the total number of connections and the trainable convolution parameters? In this Section, the input feature maps (previous layer) are of size 28x28, with channel sizes of 192. The output concatenated feature maps are also of size 28x28, with 256 channels. Various convolution kernels are used as shown in the Figure, please compute all the trainable parameters used in this Section.


1x1
64 Filters
Padding Same
1x1
128 Filters
Padding Same
3x3
128 Filters
Padding Same
Input
28(height) x 28(width) x 192(depth)
1x1
32 Filters
Padding Same
5x5
32 Filters
Padding Same
Output (Concatenated)
28(height) x 28(width) x 256(Channels)
3x3 Max Pooling
Padding Same
Stride 1
1x1
32 Filters
Padding Same