COM S 6730: Advanced Topics in Computational Models of Learning
Assignment #2

1. You need to submit a report and your code to Canvas. Your hard-copy report should include (1) answers to the non-programming part, and (2) analysis and results of the programming part. Please put all your code files and report into a compressed file named "HW#_FirstName_LastName.zip"
2. Unlimited number of submissions are allowed and the latest one will be timed and graded.
3. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.
4. All students are required to typeset their reports using latex.
5. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

---

1. (60 points)(Coding Task) **Deep Residual Networks for CIFAR-10 Image Classification**: In this assignment, you will implement advanced convolutional neural networks on CIFAR-10 using *PyTorch*. In this classification task, models will take a 32×32 image with RGB channels as inputs and classify the image into one of ten pre-defined classes. The "code" folder provides the starting code. You must implement the model using the starting code. In this assignment, you must use a GPU.

   Requirements: Python 3.6, Pytorch 1.12.1, tqdm, numpy

   Required Reading Materials:

   [1] Deep Residual Learning for Image Recognition (`https://arxiv.org/abs/1512.03385`)

   [2] Identity Mappings in Deep Residual Networks (`https://arxiv.org/abs/1603.05027`)

   [3] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (`https://arxiv.org/abs/1502.03167`)

   (a) (30 points) Complete "network.py". Read the required materials carefully before this step. You are asked to implement two versions of ResNet: version 1 uses original residual blocks (Figure4(a) in [2]) and version 2 uses full pre-activation residual blocks (Figure4(e) in [2]). In particular, for version 2, implement the bottleneck blocks instead of standard residual blocks. In this step, only basic Pytorch APIs in **torch** and **torch.nn** are allowed to use.

   (b) (10 points) Add batch normalization operations after each convolution layer. Run the model by "***python main.py***" and report the testing performance as well as a short analysis of the results.

   (c) (10 points) Based on (b), add dropout operations with drop rate of 0.3 after batch normalization layer. Run the model by "***python main.py***" and report the testing performance as well as a short analysis of the results.

   (d) (10 points) Tune all the hyperparameters in "main.py" and report your final testing accuracy.

2. (20 points) Consider the standard residual block and the bottleneck block in the case where inputs and outputs have the same dimension (e.g. Figure 5 in [1]). In another word, the residual connection is an identity connection. For the standard residual block, compute the number of training parameters when the dimension of inputs and outputs is $128 \times 16 \times 16 \times 32$. Here, 128 is the batch size, 16×16 is the spatial size of feature maps, and 32 is the number of channels. For the bottleneck block, compute the number of training parameters when the dimension of inputs and outputs is $128 \times 16 \times 16 \times 128$. Compare the two results and explain the advantages and disadvantages of the bottleneck block.

3. (20 points) Using batch normalization in training requires computing the mean and variance of a tensor.

   (a) (8 points) Suppose the tensor $\boldsymbol{x}$ is the output of a fully-connected layer and we want to perform batch normalization on it. The training batch size is N and the fully-connected layer has C output nodes. Therefore, the shape of $\boldsymbol{x}$ is $N \times C$. What is the shape of the mean and variance computed in batch normalization, respectively?

   (b) (12 points) Now suppose the tensor $\boldsymbol{x}$ is the output of a 2D convolution and has shape $N \times H \times W \times C$. What is the shape of the mean and variance computed in batch normalization, respectively?