CSCE 636: Deep Learning (Fall 2019)
Assignment #3
Due 11/8/2019

1. You need to submit a report in hard-copy before lecture and your code to eCampus. Your hard-copy report should include (1) answers to the non-programming part, and (2) analysis and results of the programming part. Your submission to eCampus should be your code files ONLY. Please put all your code files into a compressed file named "HW#_FirstName_LastName.zip"
2. The hard-copy is due in class before lecture and the code files are are due 11:30AM on eCampus on the due date. The timing of the homework submission is based on both the hard-copy and eCampus submissions, whichever happens later.
3. Unlimited number of submissions are allowed on eCampus and the latest one will be timed and graded.
4. LFD refers to the textbook "Learning from Data".
5. Please read and follow submission instructions. No exception will be made to accommodate incorrectly submitted files/reports.
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.
7. Only write your code between the following lines. Do not modify other parts.

   ### YOUR CODE HERE

   ### END YOUR CODE

---

1. (100 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 *Tensorflow*. 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, Tensorflow 1.10, 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) (10 points) Download the CIFAR-10 dataset (`https://www.cs.toronto.edu/~kriz/cifar.html`) and complete "DataReader.py". For the dataset, you can download any version. But make sure you write corresponding code in "DataReader.py" to read it.

(b) (10 points) Implement data augmentation. To complete "ImageUtils.py", you will implement the augmentation process for a single image using *numpy*. Corresponding *Tensorflow* functions are given.

(c) (40 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 Tensorflow APIs in tf.layers and tf.nn are allowed to use.

(d) (20 points) Complete "Model.py". Note: For this step and last step, pay attention to how to use batch normalization.

(e) (20 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 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 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 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?

4. (60 points) We investigate the back-propagation of the convolution using a simple example. In this problem, we focus on the convolution operation without any normalization and activation function. For simplicity, we consider the convolution in 1D cases. Given 1D inputs with a spatial size of 4 and 2 channels, *i.e.*,

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \end{bmatrix} \in \mathbb{R}^{2\times 4}, \tag{1}$$

we perform a 1D convolution with a kernel size of 3 to produce output Y with 2 channels. No padding is involved. It is easy to see

$$Y = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{2}$$

where each row corresponds to a channel. There are 12 training parameters involved in this convolution, forming 4 different kernels of size 3:

$$W^{ij} = [w_1^{ij}, w_2^{ij}, w_3^{ij}], i = 1, 2, j = 1, 2, \tag{3}$$

where W^{ij} scans the i-th channel of inputs and contributes to the j-th channel of outputs. Note that the notation here might be slightly different in that, one kernel/filter here connects ONE input feature map (instead of ALL input feature maps) to ONE output feature map.

(a) (15 points) Now we flatten X and Y to vectors as

$$\tilde{X} = [x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{24}]^T$$
$$\tilde{Y} = [y_{11}, y_{12}, y_{21}, y_{22}]^T$$

Please write the convolution in the form of fully connected layer as $\tilde{Y} = A\tilde{X}$ using the notations above. You can assume there is no bias term.
Hint: Note that we discussed how to view convolution layers as fully connected layers in the case of single input and output feature maps. This example asks you to extend that to the case of multiple input and output feature maps.

(b) (15 points) Next, for the back-propagation, assume we've already computed the gradients of loss L with respect to $\tilde{Y}$:

$$\frac{\partial L}{\partial \tilde{Y}} = \left[\frac{\partial L}{\partial y_{11}}, \frac{\partial L}{\partial y_{12}}, \frac{\partial L}{\partial y_{21}}, \frac{\partial L}{\partial y_{22}}\right]^T, \tag{4}$$

Please write the back-propagation step of the convolution in the form of $\frac{\partial L}{\partial \tilde{X}} = B\frac{\partial L}{\partial \tilde{Y}}$. Explain the relationship between A and B.

(c) (30 points) While the forward propagation of the convolution on X to Y could be written into $\tilde{Y} = A\tilde{X}$, could you figure out whether $\frac{\partial L}{\partial \tilde{X}} = B\frac{\partial L}{\partial \tilde{Y}}$ also corresponds to a convolution on $\frac{\partial L}{\partial Y}$ to $\frac{\partial L}{\partial X}$? If yes, write down the kernels for this convolution. If no, explain why.