EEP 596 Computer Vision

Assignment #4

The purpose of this homework is to begin to explore computer vision concepts using PyTorch. You are free to work with fellow students but should turn in your own work. Specifically, you should be able to explain anything in your assignment if asked.

You should turn in your assignment as **"Assignment4.py" and "Report.pdf"** in Gradescope. You should develop code with the skeleton provided, including the names of functions, inputs, and outputs. **You could use print() and cv2.imsave() to generate outputs for the report, but these should be commented out while turning in.**

1. **CIFAR-10 dataset.** Read through the following tutorial, which is the basis for the following questions. CIFAR10 tutorial
   https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
   Use TorchVision to download the CIFAR-10 train/test dataset to a directory (let's call it "cifar10") your local machine. Navigate to "cifar10/cifar-10-batches.py" directory. Open the `readme.html` file, and read the first part of the file. Considering the following question while you are reading the 'readme.html'', and **report these values in your pdf report.**
   i. The number of training batches (report in pdf with name num_train_batches)
   ii. The number of test batches (report in pdf with name variable num_test_batches)
   iii. The number of images per batch (report in pdf with nam variable num_img_per_batch)
   iv. The number of training images total (report in pdf with name num_train_img),
   v. The number of test images total (report in pdf with name num_test_img)
   vi. The size of each batch in terms of number of bytes (report in pdf with name *size_batch_bytes* in unit KB)
   vii. The size of memory required for uncompressed 32x32x3 image (report in pdf with name size_image_bytes in unit KB).
   viii. The size of memory required for a batch that has 10k 32x32x3 images (report in pdf with name size_batchimage_bytes in unit KB)

   a. **(10 points)** After you read the 'readme.html', write the code to grab a single mini-batch of 4 images from the training set, at random. Display the 4 images side-by-side, with their ground-truth labels overlaid (this is for your visualization and is not counted for final grades). **Include the plot in your report.** (Hint: As in the tutorial above, remember to transform the dataset to torch tensor and normalize each channel with mean (0.5, 0.5, 0.5) and standard deviation (0.5, 0.5, 0.5)., use DataLoader(.., batch_size=4, shuffle=True), iter(), and next() to get a single mini-batch of images. Unlike the tutorial, use plt.text() to overlay the text. See

example below.)

For this function, please return:

ix. **A batch of images as a torch array with type torch.FloatTensor**. The first dimension of the array should be batch dimension, the second channel dimension, followed by image height and image width.

x. **Labels of the images in a torch array**.

2. Train classifier (30 points). Create a CNN with the same structure as in the tutorial. Feel free to copy/paste code from the tutorial.

Step 1. Create the network. Run the network on a mini-batch of images. Think about what size the output is and what the output represents.

Step 2. Write the code to train the network. (Your code will be essentially the same as in the tutorial.) Use cross-entropy loss, and SGD with momentum. Train the network for 2 epochs, as in the tutorial. Ensure training loss steadily decreases and is comparable to the loss in the tutorial. Save the weights as **'cifar_net_2epoch.pth'** in the current folder

Step 3. Load the weight saved in **'cifar_net_2epoch.pth'** back and run the network on the test dataset. Check the accuracy you achieve on the test data.

When submitting the assignment, please remember to submit the saved weight file 'cifar_net_2epoch.pth'. *Ensure when you load the weights from the file, you can achieve around 50% accuracy on the full testing set. The final grading will be based on the test accuracy.*

3. Visualize weights. Retrieve the weights of the convolution kernels from the previous question. Ignore bias. These weights will be graded according to their

a. (10 points) Return the weights of the first layer as PyTorch tensors.

b. (10 points) Return the weights of the second layer as PyTorch tensors.

4. Hyperparameter sweep. Modify the code from Question #2 to store the training loss for later visualization. Train the model for 2 epochs and modify the code to compute the accuracy on both the training and test datasets after every 2000 iterations. (For speed, subsample 1000 images randomly from each of these, rather than computing on the entire dataset). *Please include the results in your report.pdf, the question will be graded manually.*

c. (30 points) Retrain the network using three different learning rates: 0.01, 0.001, and 0.0001. (The middle value is the same as what you used above.) Produce 3 separate plots:
    i. One plot shows the **training loss** as a function of iteration, for each of the 3 learning rates.
    ii. One plot shows the **training error** as a function of iteration, for each of the 3 learning rates.
    iii. One plot shows the **test error** as a function of iteration, for each of the 3 learning rates.
d. (10 points) Describe the results.

Note. The slow, correct way would be as follows: Every 2000 iterations compute the error on the entire 10k test set. But for this HW, you can use the faster, approximate way: Every 2000 iterations randomly sample 1k images from the entire 10k test set and compute the error only on those 1k images.