ECE 520.438 & 520.638: Deep Learning
Homework 4: Auto-encoders
Spring 2024

1 Part 1: Auto-encoder for image reconstruction (40 points)

In this problem, you will design a convolutional auto-encoder network for image reconstruction. You will use the CIFAR10 dataset for training and testing. Note that Pytorch has pre-defined data loader for CIFAR10. The input to the auto-encoder will be an image and you will try to reconstruct the same image as output. You will design just a two block encoder and a two block decoder. Effectively, your auto-encoder will thus have four convolutional layers. Note that the convolutional layers in decoder will actually be transpose convolution layers. For all these layers, use a kernel size of 3, stride of 1 and padding of 0. The number of channels in the auto-encoder should be of the following order: (input channels, output channels) ==> (3,8), (8,8), (8,8), (8,3). For example, the first layer has an input channels of 3 and output channels of 8. The second layer has an input channel size of 8 and output channel size of 8 and so on. Make sure to use a suitable activation function at required places. Use the L2 loss function.

1. Plot the convergence of the loss function.
2. Test if the auto-encoder works on the CIFAR10 test set. Visualize 10 inputs with their corresponding 10 reconstructions.

2 Part 2: Denoising Auto-encoder (60 points)

An auto-encoder is a model which can be tweaked to solve many real-world tasks. In this part, we will focus on performing denoising. You will use the same auto-encoder design and the CIFAR10 dataset as Part 1. Instead of performing reconstruction, you will train the auto-encoder this time to perform denoising. To get the noisy image, simply apply Gaussian noise (mean 0 and variance of 0.1) directly to the image tensor before passing it to the auto-encoder. While doing this, please make sure that the image has values in the range of -1 to 1. You will then train the auto-encoder to output the original image from the noisy image. You can use the same L2 loss function.

(Hint: An easy way to add Gaussian noise would be to use torch.randn. torch.randn actually outputs a tensor with elements drawn from a Gaussian distribution of zero mean and unit variance. So, an easy way would be to tweak this accordingly to change the variance to 0.1 and add to the image. There are other ways to add Gaussian noise as well.)

1. Plot the convergence of the loss function.
2. Test if the denoising auto-encoder works on the CIFAR10 test set. Note that you would need to add noise to the test data as well. Visualize 10 noisy inputs with their corresponding 10 denoised images.

3. Report the mean peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) obtained by your model on the test set.

Grading: You will be graded based on the code you develop, plus your homework report summarizing your findings. If possible, please write your report using LaTeX. Note that you are allowed to use Pytorch or any open-source library.