ECE 520.638: Deep Learning - Homework 4

Homework report written by Arushi Sinha
April 21, 2023

Link to Colab file for part 1:https://colab.research.google.com/drive/1I6dYDL19st8RfvTc62sDNxXt9UGszNLw?usp=sharing
Link to Colab file for part 2:https://colab.research.google.com/drive/17rERTNdffk2cLXgQ7Npps0Nxwk3v4qG5?usp=sharing

1. Autoencoder for image reconstruction

Objective: In this problem, you will design a convolutional auto-encoder network for image reconstruction. You will use the CIFAR10 dataset for training and testing.
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.

Observations and Results: I used the pre-defined data loader for CIFAR10 using Pytorch [1]. The auto-encoder model consists of four blocks: 2 encoder blocks (2D convolution) and 2 decoder blocks (transpose 2D convolution). The number of channels in the auto-encoder are of the following order: (input channels, output channels): (3,8), (8,8), (8,8), (8,3). For the model, Relu and Tanh activation functions are used. For training, MSE loss function and Adam optimizer is used. The model is trained for 10 epochs and it converges at loss= 0.005 at epoch 9. The plot displaying the convergence of the loss function is shown in figure 1

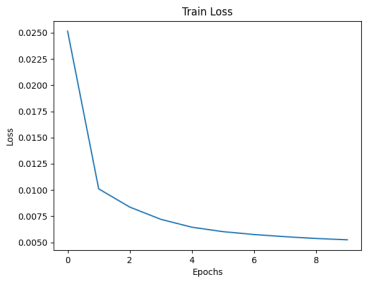

Figure 1. Loss vs epoch

After the model is trained, it is tested on the testloader. The original input images along with their corresponding reconstructed images can be seen in figure 2.

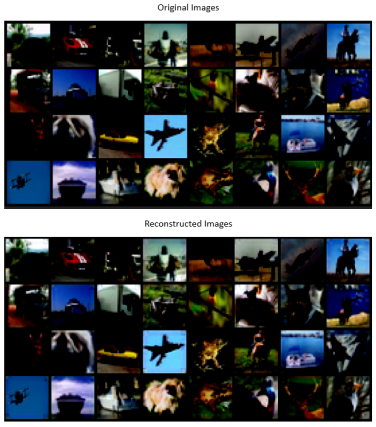

Figure 2. Original images and their corresponding reconstructed images

2. Denoising Auto-encoder

Objective: 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.
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.

Observations and Results: The same auto-encoder model is used. For training, MSE loss function and Adam optimizer is used. The input to the model is a noisy image generated by adding gaussian noise with mean 0 and variance 0.1 using torch.randn function, instead of the original image itself. The model is trained for 10 epochs and it converges at loss=0.017 at epoch 8. The plot displaying the convergence of the loss function is shown in figure 3.

After the model is trained, it is tested on the testloader. The original images along with their corresponding noised and denoised images can be seen in figure4.

The mean peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM) obtained by the model on the test set is calculated using metrics function from skim-

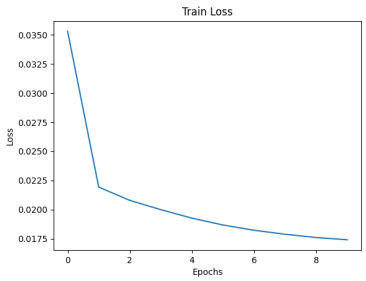

Figure 3. Loss vs epoch

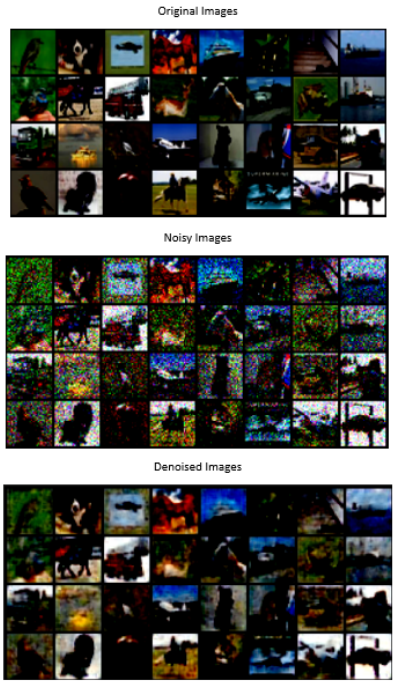

Figure 4. Original images and their corresponding noised and denoised images

age library.The observed PSNR and SSIM are reported below:

1. Average PSNR:23.860
2. Average SSIM: 0.867

References

[1] Implementing an autoencoder in pytorch. "https://www.geeksforgeeks.org/implementing-an-autoencoder-in-pytorch/#".