EE5179 : Deep Learning for Imaging

Programming Assignment 4: Autoencoders

Due by: October 28, 2022

Instructions

- Program in python for this assignment.
- Post any doubts you have on moodle. This will be helpful to your peers as well.
- Submit the codes (the actual notebook and a PDF version) and your assignment report in a zip file titled PA4_RollNumber.zip in the submission link provided on moodle.

Preliminaries

- It is recommended to use Google Colab (as in the tutorials) or Jupyter/iPython notebooks for the assignment. The notebook format is very convenient to work (and evaluate!) with and additionally, Colab provides GPU access as well. Click here to learn more about how GPUs can be accessed using Colab.
- The dataset can be downloaded from here or you can make use of the inbuilt dataset from Pytorch.
  Note: Check if the labels are in one-hot format, or appropriately convert them to one-hot format before training and testing the network.

---

1 Comparing PCA and Autoencoders

Load MNIST dataset . **Flatten the (28*28) image to 784**. Do PCA on it and take only the first 30 eigenvalues with their corresponding eigenvectors .Now , project the data onto these eigenvectors and reconstruct them . Next , train an autoencoder with the following architecture:

- **Encoder**
  - input (784)
  - fc (512)
  - fc (256)
  - fc (128)
  - fc (30)
- **Decoder**
  - fc (128)
  - fc (256)
  - fc (784)

Use ReLU as the activation function. Show the training and validation accuracy plot. Compare the Reconstruction Accuracy. Also Compare visually by plotting outputs for each of the class for both PCA and autoencoder

2 Experimenting with hidden units of varying sizes

Train a standard auto-encoder with the following architecture:

- fc(x)-fc(784)

Here, x is the size of the hidden unit. The architecture consists of only a hidden layer and the output layer. Using x = [64, 128, 256], do the following:

1. Plot the Training and validation accuracy plot for the 3 cases
2. Test the network on any one of your testset images and compare the quality of reconstruction for different values of x. This comparison should be done visually.
3. What outputs do you get if you pass a non-digit image (Try Fashion MNIST) and random noise images through the network?

3 Sparse Autoencoders

Design an over-complete autoencoder with Sparsity regularization (Check L1 Penalty in torch). We impose sparsity by adding L1 penalty on the hidden layer activation. L1 penalty is nothing but L1 norm on the output of hidden layer. Here, the parameter controls the degree of sparsity (the one you pass to L1 Penalty function while defining the model). Higher the value, more sparser the activations are. You can vary the value of this parameter and observe the change in filter visualizations. Also, if the sparsity is too much, it could lead to bad reconstruction error.

1. Plot the training-validation plot for the different sparsity values chosen
2. Compare the average hidden layer activations of the Sparse AutoEncoder with that of the Standard AutoEncoder (in the above question). Also compare visually the differences in the output for different sparsities. What differences do you observe?
3. Now, try to visualize the learned filters of this Sparse AutoEncoder as images. What difference do you observe in the structure of these filters from the ones you learned using the Standard AutoEncoder?

4 Denoising Autoencoders

Design a denoising autoencoder with just one hidden unit (Take hidden size as 256).

1. What happens when you pass images corrupted with noise to the previously trained Standard Autoencoders (From Q2) ? Compare it with Denoising autoencoders
2. Change the noise level (typical values: 0.3, 0.5, 0.8, 0.9) and repeat the above experiments. What kind of variations do you observe in the results? (Both Visually and by MSE)
3. Visualize the learned filters for Denoising Autoencoders. Compare it with that of Standard Autoencoders. What difference do you observe between them?

5 Convolutional Autoencoders

AE can also be implemented as fully convolutional networks with the decoder consisting of upsampling operations of any of these variants - i) Unpooling or ii) Unpooling + Deconvolution or iii) Deconvolution.

1. Train a Convolutional AE for the MNIST data with 3 convolutional layers for encoder and the decoder being the mirror of encoder (i.e a total of 7 convolutional layers for AE with the final convolutional layer mapping to the output). Architecture for the encoder part:
    - Input-Conv1(8 3x3 filters with stride 1)
    - 2x2 Maxpooling
    - Conv2(16 3x3 filters with stride 1)
    - 2x2 Maxpooling
    - Conv3(16 3x3 filters with stride 1)

- 2x2 Maxpooling

At the output of the final 2x2 Maxpooling we have the **encoded representation**. This needs to followed by the decoder network. Experiment with all the three types of upsampling. Refer to this report for good description on deconvolution and unpooling in deep learning.
Keeping all the other parameters the same, report on reconstruction error and convergence with the different types of upsampling. Also visualize the decoder weights for the three cases. What do you observe?