Assignment 5: CS 754, Advanced Image Processing

Due: 18th April before 11:55 pm

Remember the honor code while submitting this (and every other) assignment. All members of the group should work on and understand all parts of the assignment. We will adopt a zero-tolerance policy against any violation.

Submission instructions: You should ideally type out all the answers in Word (with the equation editor) or using Latex. In either case, prepare a pdf file. Create a single zip or rar file containing the report, code and sample outputs and name it as follows: A5-IdNumberOfFirstStudent-IdNumberOfSecondStudent.zip. (If you are doing the assignment alone, the name of the zip file is A5-IdNumber.zip). Upload the file on moodle BEFORE 11:55 pm on the due date. **There will be no late extensions for this assignments as the due date is the last day of classes**. Note that only one student per group should upload their work on moodle. Please preserve a copy of all your work until the end of the semester. If you have difficulties, please do not hesitate to seek help from me.

Note: This pdf has hyperlinks in it.

1. Your task here is to implement a deep autoencoder neural network to denoise images. You don't have to implement the network from scratch. You will modify a MATLAB sample code for this purpose. The purpose of this question is to make you familiarize with deep neural networks. This question requires MATLAB 2019 to be installed in your system.

   (a) This page is on creating a deep autoencoder for denoising application, in MATLAB. It walks you through the process of writing an autoencoder-training-code to denoise images corrupted by salt and pepper noise. Go through this and understand the code and explanations well. If you have any difficulty in following the section, feel free to contact the TA, Jerin Geo James at jeringeo@cse.iitb.ac.in.

   (b) Modify the code to incorporate the following changes and train your network

      i. Change the noise model to the Poisson model. The noisy image should be generated in the following manner: *img = poissrnd(img*N); img = img/max(img(:));* where $N = 16$ and img is a double array with intensities ranging from 0 to 1. [1 point]

      ii. Adjust the minibatch size and number of epochs and train your model at least for 30 mins on your system. [2 points]

      iii. As you can see in the code, the autoencoder network has a set of encoding layers and equivalent set of decoding layers. Your job here is to replace those layers with your own set of layers. Play around with different combinations to produce a good restoration. Your submission will not be evaluated if you do not make any changes to the existing architecture. Also, while creating the convolutional layers, name each of those layers. Google search how to do that. You will need this for part (c) below. [3 points]

   (c) Include the following things in your report

      i. Briefly describe the architecture of your network. Including the number of layers, activation functions etc. Also mention the observations you made during your trials to zero in to the final architecture. [3 points]

      ii. Submit your training code as *training.m*

iii. MATLAB produces a live plot during the training process, displaying the loss and error decays as the training progresses. Take a screenshot of this graph after the training is over and include it in the report. [2 points]

iv. The variable 'net' is your trained model. Save this and submit it.

(d) For the following sub-questions, evaluations will be done *only* by executing your code and not from your report. If you have to include anything in report, it will be explicitly stated. Your code should load 'net' variable from the submission folder and do the following things:

i. Restore any 12 images from the test dataset. Display the following items side by side for those images. [3 points]

1.Original image 2.Noisy image 3.Restored image

Also report the RMSE (i.e. $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2/\|\boldsymbol{x}\|_2$ for original image $\boldsymbol{x}$ and restored image $\hat{\boldsymbol{x}}$) and SSIM values for noisy and restored images for each case. SSIM has been implemented in MATLAB and code is also available at http://www.cns.nyu.edu/ lcv/ssim/.

ii. Write code to make a collage of filters in each layer. You can refer this page to see how to do that. Include the collages for all layers in the report. In the report, comment on the nature of filters as you go deep into the network. [5 points]

iii. Take any one image from the test dataset. Make a code to generate a collage of activations for each filter in each layer. If you have a filter of n channels in a layer, you will get n images for that layer. Submit the input image and the collages for all layers. Refer this page to see how to do that. What do you observe as you go deep into the network ? Include the observations in the report. [5 points]

(e) This page shows an example for denoising images using a pre-trained network. This network is trained for Gaussian noise, still it works fairly well for Poisson noise also. The first two sub-questions will be evaluated directly by executing the code.

i. Denoise any 12 images using this network. This network requires the image size to be larger than 50X50. Hence, concatenate the noisy images suitably, before feeding it to the network. [3 points]

ii. You can load *'cameraman.tif'* image in matlab by calling *imread('cameraman.tif');*. Corrupt this image using Poisson noise and denoise it using your network. Note that your network is designed to take 32X32 sized images. Hence, break the image into non overlapping patches and restore each patch individually. DO NOT average the patches. [3 points]

For both the above tasks you should corrupt the images in the following fashion: *img = poissrnd(img*16); img = img/max(img(:));* where *img* is a double array with intensities in the range [0,1].

iii. Comment on the performance of each network in both cases. List out the reasons why you think one network outperforms the other ? Include this part in the report. [5 points]

**Note: Once you have zeroed in to your architecture, you may leave your model to get trained overnight, to produce better results. This question has 35 points in all.*

2. We have studied Non-negative matrix factorization (NMF) and non-negative sparse coding (NNSC) in class. We have seen their applications in image denoising and in face recognition. Your job is to do a google search and find out a paper which explores any other application of either NMF or NNSC or some modification of these techniques. The application should be in the domain of image processing, machine learning, computer vision, audio processing or data mining. In your report, you should briefly describe the application, the main objective function being optimized in the paper, and a summary of the results. Mention the title, author-list and venue of the paper and put in a link to it. Recommended venues are IEEE Transactions on Image Processing/Signal Processing/Pattern Analysis and Machine Intelligence/Computational Imaging/Geoscience and Remote Sensing; Journal of Machine Learning Research; or conferences such as CVPR, ICCV, ECCV, NIP (NeurIPS), ICML. [20 points]

3. Consider an inverse problem of the form $\boldsymbol{y} = \mathcal{H}(\boldsymbol{x}) + \boldsymbol{\eta}$ where $\boldsymbol{y}$ is the observed degraded and noisy image, $\boldsymbol{x}$ is the underlying image to be estimated, $\boldsymbol{\eta}$ is a noise vector, and $\mathcal{H}$ represents a transformation operator. In case of denoising, this operator is represented by the identity matrix. In case of compressed sensing,

it is the sensing matrix, and in case of deblurring, it represents a convolution. The aim is to estimate $\boldsymbol{x}$ given $\boldsymbol{y}$ and $\mathcal{H}$ as well as the noise model. This is often framed as a Bayesian problem to maximize $p(\boldsymbol{x}|\boldsymbol{y},\mathcal{H}) \propto p(\boldsymbol{y}|\boldsymbol{x},\mathcal{H})p(\boldsymbol{x})$. In this relation, the first term in the product on the right hand side is the likelihood term, and the second term represents a prior probability imposed on $\boldsymbol{x}$.
With this in mind, we refer to the paper 'User assisted separation of reflections from a single image using a sparsity prior' by Anat Levin, IEEE Transactions on Pattern Analysis and Machine Intelligence. Answer the following questions:

- In Eqn. (7), explain what $A_{j\rightarrow}$ and b_j represent, for each of the four terms in Eqn. (6).
- In Eqn. (6), which terms are obtained from the prior and which terms are obtained from the likelihood? What is the prior used in the paper? What is the likelihood used in the paper?
- Why does the paper use a likelihood term that is different from the more commonly used Gaussian prior? [5+10+5=20 points]

4. Consider compressive measurements of the form $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{\eta}$ under the usual notations with $\boldsymbol{y} \in \mathbb{R}^m, \boldsymbol{\Phi} \in \mathbb{R}^{m\times n}, m \ll n, \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{\eta} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_{m\times m})$. Instead of the usual model of assuming signal sparsity in an orthonormal basis, consider that $\boldsymbol{x}$ is a random draw from a zero-mean Gaussian distribution with known covariance matrix $\boldsymbol{\Sigma_x}$ (of size $n \times n$). Derive an expression for the maximum a posteriori (MAP) estimate of $\boldsymbol{x}$ given $\boldsymbol{y}, \boldsymbol{\Phi}, \boldsymbol{\Sigma_x}$. Also, run the following simulation: Generate $\boldsymbol{\Sigma_x} = \boldsymbol{U\Lambda U}^T$ of size 128×128 where $\boldsymbol{U}$ is a random orthonormal matrix, and $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues of the form $ci^{-\alpha}$ where $c = 1$ is a constant, i is an index for the eigenvalues with $1 \leq i \leq n$ and α is a decay factor for the eigenvalues. Generate 10 signals from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma_x})$. For $m \in \{40, 50, 64, 80, 100, 120\}$, generate compressive measurements of the form $\boldsymbol{y} = \boldsymbol{\Phi x} + \boldsymbol{\eta}$ for each signal $\boldsymbol{x}$. In each case, $\boldsymbol{\Phi}$ should be a matrix of iid Gaussian entries with mean 0 and variance $1/m$, and $\sigma = 0.01\times$ the average absolute value in $\boldsymbol{\Phi x}$. Reconstruct $\boldsymbol{x}$ using the MAP formula, and plot the average RMSE versus m for the case $\alpha = 3$ and $\alpha = 0$. Comment on the results - is there any difference in the reconstruction performance when α is varied? If so, what could be the reason for the difference? [25 points]