CAP 4401/COT 5930: Digital Image Processing

Assignment 5: Image denoising using deep learning

Goal: Build and evaluate image denoising solutions using deep learning architectures.

Learning objectives:

- Learn how to implement an Image Processing workflow in MATLAB
- Learn how to implement and evaluate contemporary (deep-learning-based) image denoising techniques in MATLAB

Starter package

- **Datasets**:
  - Built-in dataset: DigitDataset

- **MATLAB starter code**: **DL_denoising_STARTER.mlx** (available online)

Instructions:

- Document all your findings, steps, conclusions, lessons learned, insights, etc. in your **report** *(think of it as a "lab notebook")*

- **Add your answers to the numbered questions to your report**

Procedure:

1. Download MATLAB starter code and add to your working folder, adjusting the MATLAB path if necessary.

2. Run "Part 1" of the starter code and ensure that it works as intended. This is essentially a quick review of classical of image noise types and measures of image quality (which allow a fair quantitative comparison of results produced by different types/amount of noise and will eventually be used to evaluate the performance of denoising techniques).

3. (OPTIONAL) **Your turn!**
   Expand "Part 1" of the starter code to include other test images, noise types (and their parameters) and image quality metrics.

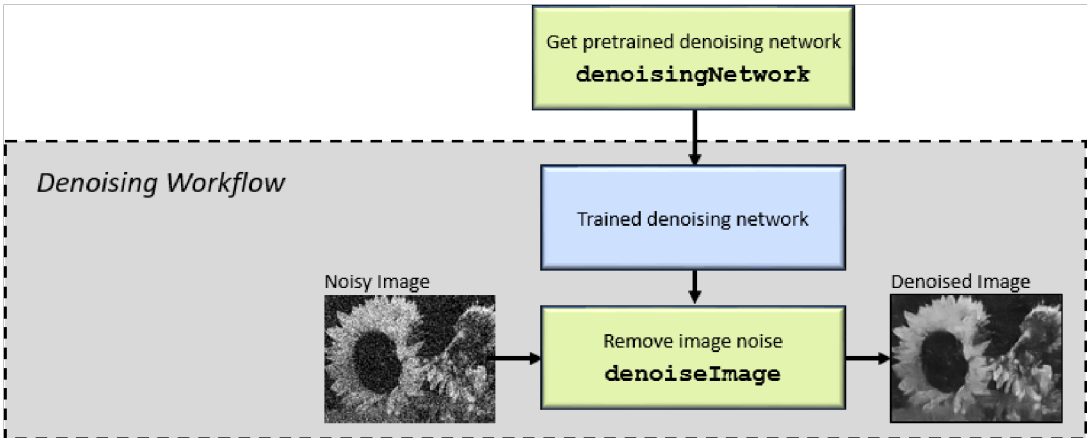

Figure 1: Denoising workflow for Part 2.

4. Run "Part 2" of the starter code and ensure that it works as intended. This is essentially the simplest workflow for removing image noise from an individual image using a pretrained denoising CNN (Figure 1).

5. **Your turn!**
   Write code to apply the DnCNN-based denoising technique of "Part 2" to at least 2 (two) different test images of your choice, using both Gaussian and salt-and-pepper noise, and compare their results using at least 2 (two) image quality metrics.

6. Summarize your results in a table (use Table 1 as an example).

Image	Noise type / amount	PSNR (denoised vs. original)	PSNR (noisy vs. original)	NIQE (original)	NIQE (noisy)	NIQE (denoised)

Table 1: Denoising using pretrained DnCNN — summary of results.

7. (OPTIONAL) **Your turn!**
   Explore the details of the denoising convolutional neural network (DnCNN) using `analyzeNetwork()`.
   See https://www.mathworks.com/help/images/ref/dncnnlayers.html and [1].

8. Answer the questions below:

> **QUESTION 1:** How did the deep-learning-based noise reduction technique perform according to the selected image quality metrics? Did you notice anything unusual?[1]
>
> **QUESTION 2:** Did you notice any significant difference between the quality of the denoised images for Gaussian versus salt-and-pepper (a.k.a. *speckle*) noise? If so, what is your explanation for the differences in quality?
>
> **QUESTION 3:** Based on your observations so far, which recommendations would you give to someone who is *new to this topic* and wants to use deep-learning-based solutions to remove/reduce noise from images for a particular application? Hint: Simpler is often better.

9. (OPTIONAL) Run "Part 3" of the starter code and ensure that it works as intended. This is the complete workflow for **training** a denoising CNN using a datastore of pristine images and eventually deploying the resulting model to be used in the way we did in Part 2 (Figure 2).

   Note: this is a very time-consuming operation that can only be successfully completed if you have powerful GPUs.

[1] An example of unusual result: NIQE score for denoised image could be better than the score for original image!

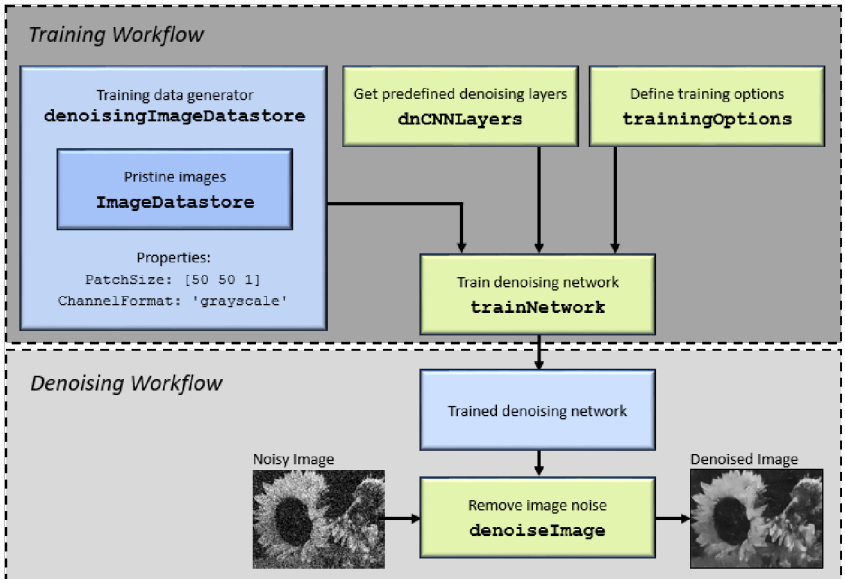

Figure 2: Denoising workflow for Part 3.

---

10. (OPTIONAL) Your turn!

Write code to apply the DnCNN-based denoising technique of "Part 3" to at least 2 (two) different test images of your choice, using both Gaussian and salt-and-pepper noise, and compare their results using at least 2 (two) image quality metrics.

11. (OPTIONAL) Summarize your results in a table (use Table 2 as an example).

Image	Noise type / amount	PSNR (denoised vs. original)	PSNR (noisy vs. original)	NIQE (original)	NIQE (noisy)	NIQE (denoised)

Table 2: Denoising using DnCNN trained from scratch – summary of results.

12. Denoising using Autoencoders

Run "Part 4" of the starter code and ensure that it works as intended. This is the workflow for training a convolutional autoencoder for denoising purposes (*see [2] for a tutorial on autoencoders*). We will use the built-in dataset called *DigitDataset* ('toolbox/nnet/nndemos/nndatasets/DigitDataset').

13. After the training is complete you should see learning curves similar to the ones in Figure 3.

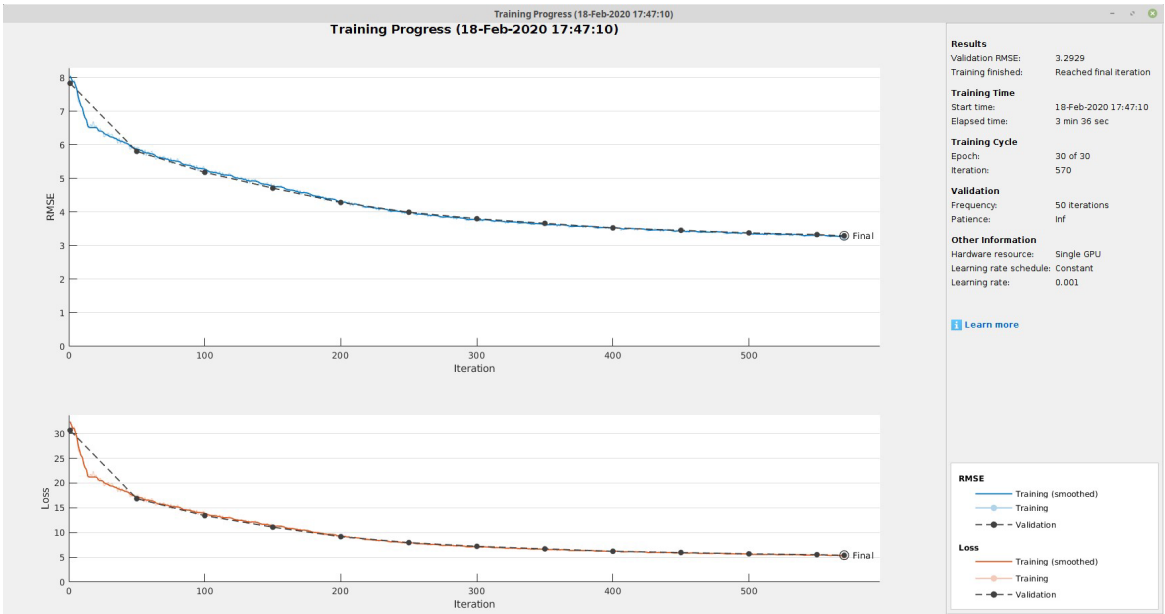

Figure 3: Learning curves for autoencoder in Part 4.

14. Answer the questions below (based on the learning curves):

> **QUESTION 4:** Was the choice of hyperparameters (learning rate, number of epochs, etc.) appropriate?
>
> **QUESTION 5:** Is the network overfitting? Explain.

15. **Your turn!** (OPTIONAL)
Write code to compute additional figures of merit for the same selected image pair.

16. Answer the questions below:

> **QUESTION 6:** Which general principle of autoencoders explains why they work so well, in general, for denoising?
>
> **QUESTION 7:** Did you notice any significant difference between the quality of the denoised images for Gaussian versus salt-and-pepper (a.k.a. *speckle*) noise? If so, what is your explanation for the differences in quality?

17. Prepare your **report**, with all relevant plots, code snippets, numerical values and – most importantly – your insights and lessons learned.

References

[1] Zhang, K., W. Zuo, Y. Chen, D. Meng, and L. Zhang. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising." *IEEE Transactions on Image Processing*. Vol. 26, Issue 7, 2017, pp. 3142–3155.

[2] Chollet, F. Building Autoencoders in Keras (Tutorial). May 2016. https://blog.keras.io/building-autoencoders-in-keras.html

Reference links:

- https://www.mathworks.com/help/deeplearning/autoencoders.html
- https://www.mathworks.com/help/deeplearning/ug/image-to-image-regression-using-deep-learning.html
- https://www.mathworks.com/help/images/image-quality-metrics.html
- https://www.mathworks.com/help/images/ref/brisque.html
- https://www.mathworks.com/help/images/ref/denoiseimage.html
- https://www.mathworks.com/help/images/ref/denoisingimagedatastore.html
- https://www.mathworks.com/help/images/ref/denoisingnetwork.html
- https://www.mathworks.com/help/images/ref/dncnnlayers.html
- https://www.mathworks.com/help/images/ref/immse.html
- https://www.mathworks.com/help/images/ref/imnoise.html
- https://www.mathworks.com/help/images/ref/niqe.html
- https://www.mathworks.com/help/images/ref/piqe.html
- https://www.mathworks.com/help/images/ref/psnr.html
- https://www.mathworks.com/help/images/ref/ssim.html
- https://www.mathworks.com/help/images/train-and-apply-denoising-neural-networks.html