EE371Q Digital Image Processing

Homework #2

Instead of submitting Homework 2, you have an alternative option to receive 100% on this assignment: participate in an interesting **image (aesthetic) quality assessment human study!**

Description of the Human Study

This study is attempting to learn the best way to convert from landscape mode to portrait mode. The subjects are instructed to select the best portrait crop from each displayed image using the mouse cursor. The main requirement of participation is that you have normal or corrected-to-normal vision.

You will be shown a series of images on a monitor interface. You can interactively drag a bounding box on the image to crop the desired region.

This study is divided into two separate test sessions. There will be a short instruction video before the test session. Each test session lasts approximately an hour. Given each image, you will be asked to provide a bounding box for your desired crop.

This data will be used for research purposes. No release of the data will contain any identifiable information that could associate you with it.

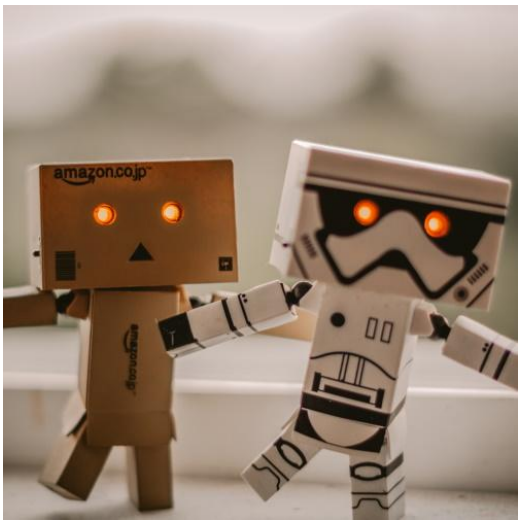
To sign up for the study, a Google sheet will be shared with you (via Canvas or Sidharth, the TA) that will have 1-hour slots spread over the next two weeks (Monday, Oct 2nd – Sat, Oct 14th, 2023).

Contact Cheng-Han (chenghan.lee@utexas.edu) or Mani (mmandal@utexas.edu) if you have any doubts or questions.

- This homework is due two weeks from now, on **October 11th at 11:59pm**.
- You are encouraged to work with other students to understand the course material. However, sharing source code, images, or other answers from any homework is strictly forbidden. You are to submit your own work.
- Please include your written answer and/or output images for each problem.
- Please show all steps for full credit and attach the code to each problem. If using Python, Jupyter Notebooks can allow you to generate a PDF for submission quite easily.
- **All the images needed for this homework can be downloaded as .jpg files from Canvas under the HW2 folder in the Files tab.**
- The homework should be formatted as a PDF file. You should also submit your source code file (if there are multiple files, submit it as a zipped file)

Problem 1 - Discrete Fourier Transform (30 points)

The goal of this problem is to use DFT and understand the roles of magnitude and phase components of an image in the frequency domain (Helpful functions: np.fft.fft2, np.fft.ifft2, np.fft.fftshift, np.fft.ifftshift, np.abs, np.angle, real)



figures.jpg and taj_mahal.jpg

a) (5 points) Read both the images, figures.jpg and taj_mahal.jpg, and convert them to grayscale. Display both the colored and the corresponding gray-scale images in a 2x2 grid.

b) (5 points) Compute the DFT of the gray-scale version of figures.jpg. Calculate the DFT phase component and the logarithm of the DFT magnitude component, and perform full-scale contrast stretch (FSCS) on both these components. Show the results side by side. Please move the low frequency part to the center (helpful functions: np.fft.fft2, np.fft.fftshift).

c) (5 points) Add “π” to the original DFT phase component of the gray-scale version of figures.jpg. Perform the inverse DFT to reconstruct the image using the modified DFT phase (after adding “π”) and the original DFT magnitude components (use the components before FSCS). Show the original and the reconstructed scotland.jpg side by side. Can you explain the result mathematically? (Hint: Keep only the real part of the inverse DFT, because the imaginary parts may be generated due to rounding errors.)

d) (5 points) Increase the original DFT magnitude component of the gray-scale version of figures.jpg to the following powers: 1.5 and 0.5. Perform the inverse DFT to reconstruct the image using the modified DFT magnitude and the original DFT phase components

(before the FSCS). Show the two reconstructed images and the original image in a 1x3 grid. Can you explain the effects you observe after the exponentiation operation?

e) (5 points) Truncate the shifted DFT component of the gray-scale version of figures.jpg by a truncation window (you can determine the size and shape) to keep ~25%, ~12.5%, and ~6.25% of the DFT coefficients in center. Perform the inverse DFT to reconstruct the image using the modified DFT (remember to undo the FSCS if required). Show the original and the three reconstructed figures.jpg images in a 2x2 grid with the appropriate labels. (Hint: Keep only the real part of the inverse DFT, because the imaginary parts may be generated due to rounding errors.)

f) (5 points) Let the original DFT magnitude component of the gray-scale version of figures.jpg be the same, and replace its DFT phase component with that of the gray-scale version of taj_mahal.jpg. Perform the inverse DFT to reconstruct this phase-off image. Show the resulting image, with full-scale contrast stretch if necessary, and the original figures.jpg image side by side. Write a few lines about what you observe, regarding what information the magnitude and phase components carry in the frequency domain.

Problem 2 - Image Filtering (40 points)

The goal of this problem is to understand some types of image noise, compare linear and non-linear image filtering, and explore using the DFT to perform filtering. (Useful functions: getGaussianKernel, filter2D, median, np.random.normal, np.random.randint)

joey.jpg

a) (5 points) You need to write a function, “average_filter()”. The function will have the parameters as follows: average_filter(image, window_size). The function should implement the average filtering on an image with square windows (helpful function: filter2D).

b) (5 points) You need to write a function, “median_filter()”. The function will have the parameters as follows: median_filter(image, window_size). The function should implement the median filtering on an image with square windows (helpful function: median).

c) (5 points) You need to write a function, “gaussian_filter()”. The function will have the parameters as follows: gaussian_filter(image, window_size, sigma). The function should implement the Gaussian filtering on an image with different window sizes and σ (standard deviation) (helpful function: getGaussianKernel).

d) (10 points) Read the image, joey.jpg, and convert it into gray-scale. Add 5% salt-and-pepper noise to the image. This means on an average, 2.5% of the pixels are changed to 255 and 2.5% of the pixels are changed to 0 (helpful function: np.random.randint). Apply the three different filters you implement above to the noisy image. Show the noisy image and the best de-noised results from the three filters in terms of MSE (Mean Squared Error) in a 2x2 grid (with appropriate labels), and report both the resulting MSE and the parameters you picked for each filter. Write a few lines explaining the results and why the filter that works best does so.

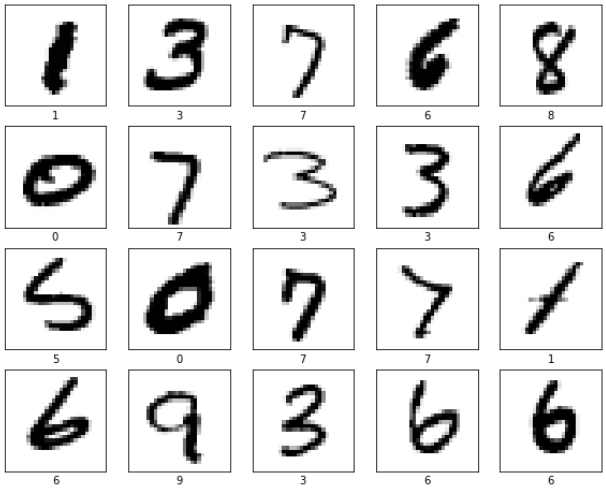
e) (10 points) Read the image, joey.jpg, and convert it into gray-scale. Add Gaussian white noise of zero mean and σ= 0.1 to the image (helpful function: np.random.normal). Apply the three different filters you implement above to the noisy image. Show the noisy image and the best de-noised results from the three filters in terms of MSE in a 2x2 grid (with appropriate labels), and report both the resulting MSE and the parameters you picked for each filter. Write a few lines explaining the results and why the filter that works best does so. (Hint: Add the Gaussian white noise to the normalized image, i.e., intensity scaled to [0 1], or alternatively, scale the Gaussian white noise based on the range of image intensity.)

f) (5 points) We will now try doing linear filtering using the DFT. You need to write a function, “linear_filter(image, h)”. This function should filter an image by pointwise multiplication in the DFT domain with a frequency response template ‘h’ (Hint: don’t forget to account for the wrap-around effect). Now, let’s use a simple linear filter to blur the image. Perform this operation twice, once using “average_filter()” with a 15x15

window, and once using “linear_filter()” and a corresponding 15x15 kernel. Display the results of both in a 1x2 window. Explain the similarities and differences if any in the images.

Problem 3 - Convolutional Neural Networks (30 points)

The goal of this problem is to get you used to machine learning for image processing. We will use CNNs to create a classifier for images of digits. In other words, we want our model to tell us whether the digit in the image is one of 0, 1, 2, …, 9. We recommend that you use PyTorch for this question as it is one of the most popular machine learning frameworks and is a good choice for implementing your projects (if they are related to machine learning). We have provided starter code (q3.py) that uses PyTorch that should make this problem much more approachable for those that do not have any experience with machine learning. However, you are not required to use the provided template.



Sample images from the MNIST dataset, along with corresponding labels

a) (5 points) We will be using the MNIST dataset for this problem. This dataset contains a large collection of 28x28 grayscale images corresponding to handwritten digits (0-9). It is usually split into two sets: the training set (60,000 images) and the testing set (10,000 images). Download this dataset and visualize 6 of the images in a 2x3 grid, with labels indicating the corresponding values that are displayed in the images.

b) (5 points) Implement a CNN architecture that you will train to classify these handwritten digits. In the starter code we provide guidance on how to implement the LeNet5 architecture, however you are free to research and implement a model of your choosing

c) (5 points) Initialize the network and visualize 6 testing images along with labels corresponding to predictions made by your **untrained** network.

d) (10 points) Train the model for at least 1 epoch (meaning 1 pass through the entire training dataset). We recommend that you use stochastic gradient descent (SGD) for optimizing the network and use cross entropy loss as your loss function. Feel free to play around with training hyperparameters, here are some that you can start with: batch_size=100, learning_rate=0.001, momentum=0.9. Report the accuracy of your model on the testing dataset once you are done training

e) (5 points) Visualize 6 testing images along with labels corresponding to predictions made by your **trained** network.