BME 470/570 Project

Category-selection deadline: November 11, 2022 11:59 pm
Preference-presentation deadline: November 17, 2022, 11:59 pm
Report due date: December 19, 2022 11:59 pm

Instructions

- In HW 1, we learned concepts in functional analysis for imaging through solving mathematical problems. In HW 2, we are learning about some of the computational tools for analysis of imaging systems such as singular-value decomposition and eigen analysis. In HW 3, we will learn about Fourier theory. Having learned these theoretical and computational tools, in this assignment, our objective will be to use these tools to analyze two imaging systems! Further, we will use these tools to conduct artificial-intelligence (AI)-based operations on the data obtained with these systems. We will conduct this analysis through a step-wise procedure.
- The assignment will be graded based on a presentation and a report.
- For each of the two imaging systems mentioned below, there are seven sub-parts. The sixth and seventh sub-part involve concepts from artificial intelligence (AI). However, these two questions are more challenging than the other questions due to multiple reasons such as this not being a course on AI, the AI-related questions being more involved and longer and that question would require more computations. To ensure fairness while providing students with flexibility, we offer the following option to choose the questions a student wants to attempt:
  - The seven sub-parts are divided into two categories, sub-parts (1)-(5) and sub-parts 6-7. You are required to choose between one of these two categories by Nov. 11, 2022.
  - If you choose sub-parts (1)-(5), in your presentation, you would be randomly assigned to present one of these sub-parts. You would have to then present that sub-part individually. In your written report, you would have to provide solutions to all 5 sub-parts and for both the systems.
  - If you choose sub-part 6-8, you will be assigned one of these sub-parts for one of the two systems. You can collaborate and present that sub-part as a team of 3 people. In your written report, you would need to provide solution only to sub-part and only for the system that you present.
  - You need to inform the instructor and the TAs if you want to work on sub-parts (1)-(5), or on sub-part (6)-(8). If you choose to work on sub-part (6)-(8), you would need to inform the composition of your team and if you have a preference for System 1 vs. System 2.
  - In case we do not hear from a student, we will assume that the student has chosen subparts (1)-(5) for their presentation and written report.
- The presentation and report components of this assignment are described below.

- **Presentation:**
  - The objective of the presentation is to learn how to communicate mathematically complex ideas. This is a very important skill while presenting work on theoretical and computational imaging at venues such as conferences.
  - The presentations will occur over the last four days of the class (Nov. 28, Nov. 30, Dec. 5, and Dec. 7).
  - In several of the questions, we ask that the student display the results for five objects. The requirement of five object functions is only for the presentation. Also, **one of these objects should be the Shepp-Logan phantom**. The other four object functions can be your choice.
  - If the student chooses to attempt sub-parts (1)-(5), they will be assigned one sub-part within this category to present. The student will be assigned 8 minutes, of which 5-6 minutes are assigned to presentation and the remaining time for a question-answer session. The questions will be asked by the students, TAs, and the instructor.
  - For sub-part (6)-(7), the team of students will get a total of 20 minutes, with 15 minutes assigned for presentation and 5 minutes for questions. All members of the team need to present.
  - In the presentation, the student should focus on outlining the concept they learned in the sub-part they are presenting. A guideline to organize the presentation is as follows:
    * Introduce and very briefly go through how the problem was solved.
    * Discuss the results.
    * Describe the concept that was learned.

    More detailed guidelines have been provided in presentation format on Canvas.
  - We have tried to ensure that sub-parts (1)-(5) have equal levels of difficulty. In case you choose this category of question to attempt, the students can mail their preferences on which of these sub-parts they would like to present to the instructor and teaching assistants by Nov. 17. To ensure that different students present different sub-parts, the sub-parts will be assigned to the students through a random-selection procedure that will account for the student preferences. The sub-parts will be assigned by around Nov. 20. Students who choose their own imaging system will need to mention this in their email and those students will be assigned one sub-part for that system.
  - Students who have taken **BME 470** and choose between subparts (1)-(5) can provide their preferences between sub-parts $1-4$. Mention in your email that you have taken BME 470.
  - The question-answer session at the end of each presentation is highly encouraged and will be part of the grading (both for the presenter and the rest of the students).
- **Report**:
  - The report for this assignment is due Dec. 19, 2022 at 11:59 pm.
  - In the report, first answer all the sub-parts for one system before proceeding to the next system.
  - Start each sub-part on a new page.

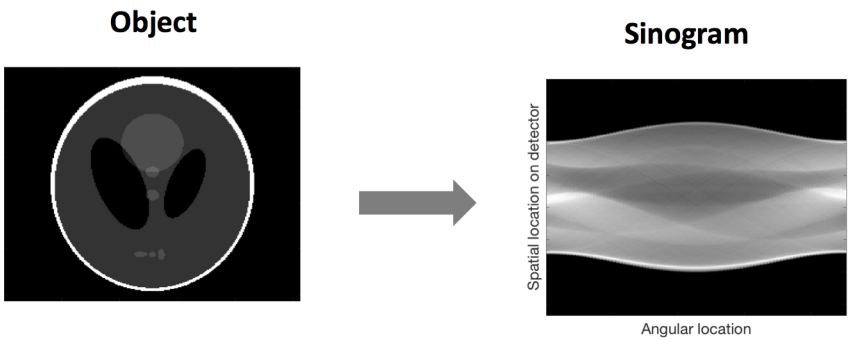

Figure 1: Demonstrating the procedure to display a sinogram

  - In several of the questions below, we ask that the student display the results for five object functions. For the report, the results need to be displayed only for **one object function: the Shepp-Logan phantom**.
  - Collaboration on the assignment is fine, but any written material, including code, should be your own.
  - We recommend for the report to be typed.
  - Organize your solutions so that they are easy to follow. Some suggestions include providing titles for plots, plotting legible figures, and referring to the plots by figure numbers in your solution. For example, in questions that require plotting and/or providing comments on the plots, the template below could be used as a guideline:
    * Provide theoretical details of the solution and conceptual understanding.
    * Provide any implementation details. This is a great place to provide the code.
    * Plot results. For each plot, provide title and caption.
    * Provide comments on the results. In each comment, refer to the plot by Fig. number (e.g. Fig. 3a)

    A general thought to keep in mind while organizing your solution is putting yourself in the shoes of the grader.

- In some of the questions below, you are asked to display the data vectors as a sinogram. In this sinogram image, the pixels for each detector position are stacked in a vertical strip, with a grey level in each pixel corresponding to the data for that pixel, and the strips for successive angles are placed next to each other. An example is shown in Fig. 1.

- Your entire submission should be a single file containing your answers to all the questions in PDF format. **Do not submit multiple files or zip files.**

- All the code used for this assignment must be submitted. If you prefer to not include the code along with your solution, then include the code as an Appendix in your submitted file.

The key purpose of this project is to become familiar with analyzing imaging systems using the tools learned in class. We will consider two imaging systems in this project.

(A) **System 1: 2-D tomographic system**: Consider a simplified 2-D tomographic system. The field of view (FOV) is the unit disk in the plane (diameter = 1). The kernel for the imaging system is given by

$$h_m(x,y) = \begin{cases} 1 \text{ for } \left(-\frac{1}{2} + \frac{m-1}{16}\right) < y <= \left(-\frac{1}{2} + \frac{m}{16}\right) \\ 0, \text{otherwise} \end{cases} \tag{1}$$

In other words, each sensitivity function is equal to 1 in a horizontal strip of width 1/8, and zero outside this strip. The strips are non-overlapping and stack up to cover the whole unit disk. This corresponds to a 16 pixel detector oriented vertically with perfect collimation. To get the next 16 sensitivity functions we rotate these by $\pi/8$. We continue rotating by $\pi/8$ to generate a total of $M = 16 \times 8$ sensitivity functions.

(B) **System 2: 2-D radial MRI**: Consider a highly simplified 2D radial MRI system. The kernel for the imaging system is given by:

$$h_m(\boldsymbol{r}) = \exp(-2\pi i \boldsymbol{k}_m \cdot \boldsymbol{r}) \tag{2}$$

and the FOV is the disk centered at the origin with diameter 1. The first 16 $\boldsymbol{k}_m$ are $(m-8, 0)$ for $m = 1, 2, \ldots, 16$. To get the next 16 $\boldsymbol{k}_m$ we rotate the first 16 by $\pi/8$. We continue rotating by $\pi/8$ to generate a total of $M = 16 \times 8$ sensitivity functions.

For each of the systems above, **or for a system of your choice**, perform the following:

1. **System modeling:** Simulate the forward model for this system using the following expansion functions for representing the object:
   (a) 16×16 pixels
   (b) 32×32 pixels
   (c) Fourier-basis functions for the square of side-length 1 units for system A and system B. These would be of dimensions 32×32.

   In the process, you will obtain the $\boldsymbol{H}$ matrices for each of the expansion functions. For 5 object functions in the unit disk and using these $\boldsymbol{H}$ matrices, display and compare the data vectors that you get from the three different discretizations.

2. **SVD analysis of DD operator:** For each of the $\boldsymbol{H}$ matrices generated in part 1, perform the following:
   (a) Compute the SVD and plot the singular value spectra.
   (b) Plot some of the singular vectors in object space as images.
   (c) For each singular vector plotted above, plot the corresponding singular vectors in data space in the sinogram format.
   (d) For five object functions on the unit disk, use the SVD analysis to determine and then display the measured and null components of the object.
   (e) Comment on all the results.

3. **Reconstruction with DD operator:** For five object functions of your choice on the unit disk and each $\boldsymbol{H}$ matrix obtained in 1, generate the data vector and perform the following:

(a) Use the SVD to perform a pseudoinverse reconstruction for each $\boldsymbol{H}$ matrix for five object functions of your choice.

(b) Repeat with Poisson noise added to the data. To add Poisson noise in Matlab, you can use the poissrnd command. Remember that the Poisson noise must be added to the data, not to the object.

(c) You will find that adding noise leads to poor-quality reconstruction when all the SVD values are used. Explain why.

(d) Reduce the number of SVD values by discarding the values with smaller magnitude and repeat the reconstruction. This type of process is referred to as regularization.

4. **SVD analysis of CD operator:** In this sub-part, we work directly with the original continuous-to-discrete (CD) operator $\boldsymbol{\mathcal{H}}$, the kernel of which is given by Eq. (1) and Eq. (2) for the two systems. Note that the operator **cannot** be discretized in the problems below.

   (a) Describe the procedure to compute the SVD. Hint: Use similar approach as problem 3 in Homework 2.

   (b) Plot the singular value spectra.

   (c) Plot the some of the singular vectors in object space as images. Note that you will need to discretize the singular vectors.

   (d) Plot some of the singular vectors in data space in the sinogram format.

   (e) For five object functions on the unit disk, use the SVD analysis to determine and then display the measured and null components of the object. Again you will need to discretize the object functions.

5. **Reconstruction with CD operator:** Here we will perform reconstruction with the original CD operator. Again note that the operator cannot be discretized in the problems below.

   (a) Use the SVD to perform a pseudoinverse reconstruction of the CD $\boldsymbol{\mathcal{H}}$ operator for five object functions on the unit disk.

   (b) Repeat with Poisson noise added to the data. Remember that the Poisson noise must be added to the data, not to the object.

   (c) The results from part (b) above will likely be poor-quality reconstruction. Explain why that is the case.

   (d) Implement the regularization-based strategy described above to address the issue of poor-quality reconstruction.

   In the next question, we will explore the application of the concepts we learned in class in the emerging era of AI, and more specifically, deep learning. We will use the tool of convolutional neural network (CNN) in these questions, and some working knowledge of how to train and test CNNs will be assumed. There are multiple references to this though, and several open-source codes. The emphasis will not be on AI, but instead on the tools we learned in the class.

6. **Image segmentation:**

   Background: Images typically consist of multiple regions. For example, a medical image of a patient with a disease may consist of normal organs and abnormal organs. Similarly, a image of a road may consist of different cars, traffic signals, road signs and so on. Image

segmentation is the process of partitioning an image into these different regions. This helps in analyzing the image to extract relevant information. For example, segmentation of a tumor in a medical image of a patient with cancer can quantify the volume of the tumor.

In this question, we will develop a supervised CNN-based approach to segment lesions from images. Supervised deep-learning-based approaches for image segmentation are broadly based on the idea that if a neural network is provided a population of images, and for each image, is provided the corresponding label that specifies the region to which each pixel in the image belongs, then the approach can learn how to segment the images. More specifically, the network is "trained" to classify each pixel and thus assign that pixel a label. By minimizing the distance between the true labels and the estimated labels (using a loss function), through an iterative process, the network becomes trained to segment a new image.

As we see, this is a multi-step process. In this question, we will go through each of these steps one by one.

(a) **Population generation:** The first step is to generate a population of objects. Note here that it is important that there should be variability in your population, else the network may just memorize how to segment the tumor in your training set, and would fail when given a new image with a new tumor.

The object will be assumed to consist of two parts, signal and background. The signal and background will be denoted by $f_s(\boldsymbol{r})$ and $f_b(\boldsymbol{r})$. If we denote the object as $f(\boldsymbol{r})$, then we can write

$$f(\boldsymbol{r}) = f_s(\boldsymbol{r}) + f_b(\boldsymbol{r}) \tag{3}$$

We consider a simple circular model of the signal, $f_s(\boldsymbol{r})$. This is given by

$$f_s(\boldsymbol{r}) = A_s \text{rect}(\boldsymbol{r}) \text{circ}\left(\frac{\boldsymbol{r}-\boldsymbol{c}}{\sigma_s}\right) \tag{4}$$

where A_s denotes the signal amplitude and where $\text{circ}\left(\frac{\boldsymbol{r}-\boldsymbol{c}}{\sigma_s}\right)$ denotes a function that is unity inside a circle centered at $\boldsymbol{c} = (c_{xs}, c_{ys})$ and of radius σ_s. Mathematically

$$\text{circ}\left(\frac{\boldsymbol{r}-\boldsymbol{c}}{\sigma_s}\right) = \begin{cases} 1, & |\boldsymbol{r}-\boldsymbol{c}| \leq \sigma_s. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

For the background $f_b(\boldsymbol{r})$ we will use a simplistic version of the lumpy background model. This model denotes the objects as a collection of Gaussian lumps, and is given by

$$f_b(\boldsymbol{r}) = \text{rect}(\boldsymbol{r}) \sum_{n=1}^{N} \frac{a_n}{(2\pi\sigma_n^2)} \exp\left[-\frac{(x-c_{x,n})^2}{2\sigma_n^2} - \frac{(y-c_{y,n})^2}{2\sigma_n^2}\right] \tag{6}$$

where N denotes the total number of lumps, a_n and σ_n denote the amplitude and width of the lump, and $(c_{x,n}, c_{y,n})$ denote the center of the lump. We will consider that $c_{x,n}$ and $c_{y,n}$) are both sampled from a uniform distribution. Consider that N and σ_n are fixed.

To generate an object, we follow the following process:

i. Generate signal: Sample the signal center (c_{xs}, c_{ys}) from a uniform distribution with range $(-1/4, 1/4)$. Sample σ_s from a uniform distribution with range $(0, 1/2)$. Fix $A_s = 3$. Insert these sampled values into Eq. (8). This leads to a continuous representation of the signal. Discretize this over 64×64 pixels. This yields the discretized signal.

ii. Generate background: Fix the number of lumps $N = 10$ and for each lump, fix σ_n to σ_s and $a_n = 1$. Next, for each lump, sample $c_{x,n}$ and $c_{y,n}$, each from a uniform distribution with range $(-1/2, 1/2)$. Insert these values into Eq. (10). This yields the continuous version of the background. Discretize this over 64×64 pixels.

iii. Add the signal and the background. This yields the object over a 64×64 grid.

iv. Generate the segmentation labels: Label all the pixel values as 1, other than the pixels occupied by the signal, which will be labeled as 2. This yields the true segmentation.

Repeat the above three steps for $J = 500$ times will yield 500 object realizations. In each realization, in addition to saving the object, also save the true segmentations.

(b) **Image generation**: Using the imaging system model for System A or System B, conduct the following:

i. Generating the system matrix: Simulate the forward model for the system using the expansion function of the object as pixels with the following dimensions:

- 32×32 pixels
- 16×16 pixels
- 8×8 pixels

ii. Computing the SVD: Perform the SVD of this system matrix to obtain the singular values and the singular vectors.

iii. Obtaining the pseudo-inverse of the system operator: Use the SVD to obtain the pseudoinverse of the system operator.

iv. Generating projection data: Apply the generated system matrix to the generated objects to compute 500 instances of the projection data.

v. Reconstructing the image: Apply the pseudoinverse to the projection data to obtain 500 instances of the reconstructed images. Conduct this process for all three dimensions of the number of pixels mentioned in subpart 6(b)i.

We will split this dataset of 500 images into three parts, a training dataset, consisting of 300 images, a validation dataset consisting of 100 images, and a testing dataset, consisting of the remaining 100 images. For each of these reconstructed images, we have the true segmented object.

(c) **Training the CNN:** Using any generic CNN-based segmentation code, train a CNN to segment the 300 images in the training dataset, using the 100 images in the validation test for ensuring that there has been no overfitting. One such code is available on the following website: link. The link directs you to a github page, where the file of interest is cnn_model.py. You would be required to train one CNN for each of the three dimensions mentioned in subpart 6(b)i. For the sake of notation, we will denote the CNN trained with images of size $K \times K$ pixels by CNN_K.

(d) **Studying the effect of pixel size:** In this step, we will evaluate the performance of the three CNNs on the corresponding images in the testing set. To quantify segmentation performance, we typically use the figure of merit of dice similarity coefficient (DSC). In this question, we will also quantify performance on a simpler version of a clinical task, namely quantifying tumor area. Perform the following operations for all the three dimensions mentioned in subpart 6(b)i.

i. Apply the trained CNN_K to the 100 test images of dimension $K \times K$ pixels.

ii. Compute the DSC between the true segmentation and the segmentation estimated using the CNN. Plot the variation in DSC as a function of the number of pixels. Comment on your results.

iii. For each image, compute the true signal area (This would be $\pi\sigma_s^2$). Compare that to segmentation obtained with the CNN. Plot this as a function of the signal radius, σ_s. Show four plots on the figure, corresponding to the true area, and the three area values estimated at the three pixel values. Comment on your results.

You will see that the pixel size affects your accuracy of the estimate. One approach to address this is to instead estimate the fractional region that the signal occupies in each pixel, as in this study link. This is optional in case you want to try this approach.

7. **Image denoising:**

Background: Images are typically corrupted by noise. This noise could be due to various sources such as photon noise and electronic noise. The corruption by noise impacts image quality and thus, in the imaging community, there are several efforts on making the image look visually less noisy. This operation is referred to as image denoising. In this question, we will developed a supervised CNN-based approach for image denoising. These are based on the idea that if a neural network is provided a population of noisy images, and for image, is provided a less noisy version, then the algorithm can learn how to predict the less noisy images given a more noisy image.

A key question is how do we know if a denoising method is effective and better compared to other methods. To answer this question, we recall that in scientific and medical imaging, images are acquired for a certain task. An example of this could be that a medical image of a patient exhibiting symptoms of say cancer is acquired so that a radiologist could assess if the patient has a tumor. This is referred to as the detection task. Another task could be measurement of the tumor volume. This is referred to as a quantification task. Ideally, a denoising should be evaluated based on how well it performs in that task. In this question, we will also evaluate the performance of this denoising technique on the task of detecting a certain signal.

As we see, this is a multi-step process. In this question, we will go through each of these steps one by one.

(a) **Population generation:** The first step is to generate a population of objects. Note here that it is important that there should be variability in your population, else the network may just memorize how to denoise a specific set of noisy images, and would fail when given a new image.

As in the previous subpart, the object will be assumed to consist of two parts, signal and background. The signal and background will be denoted by $f_s(\boldsymbol{r})$ and $f_b(\boldsymbol{r})$. If we denote the object as $f(\boldsymbol{r})$, then we can write

$$f(\boldsymbol{r}) = f_s(\boldsymbol{r}) + f_b(\boldsymbol{r}) \tag{7}$$

We consider a simple circular model of the signal, $f_s(\boldsymbol{r})$. This is given by

$$f_s(\boldsymbol{r}) = A_s \text{rect}(\boldsymbol{r}) \text{circ}\left(\frac{\boldsymbol{r} - \boldsymbol{c}}{\sigma_s}\right) \tag{8}$$

where A_s denotes the signal amplitude and where $\text{circ}\left(\frac{\boldsymbol{r}-\boldsymbol{c}}{\sigma_s}\right)$ denotes a function that is unity inside a circle centered at $\boldsymbol{c} = (c_{xs}, c_{ys})$ and of radius σ_s. Mathematically

$$\text{circ}\left(\frac{\boldsymbol{r}-\boldsymbol{c}}{\sigma_s}\right) = \begin{cases} 1, & |\boldsymbol{r}-\boldsymbol{c}| \leq \sigma_s. \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

For the background $f_b(\boldsymbol{r})$ we will use a simplistic version of the lumpy background model. This model denotes the objects as a collection of Gaussian lumps, and is given by

$$f_b(\boldsymbol{r}) = \text{rect}(\boldsymbol{r}) \sum_{n=1}^{N} \frac{a_n}{(2\pi\sigma_n^2)} \exp\left[-\frac{(x-c_{x,n})^2}{2\sigma_n^2} - \frac{(y-c_{y,n})^2}{2\sigma_n^2}\right] \tag{10}$$

where N denotes the total number of lumps, a_n and σ_n denote the amplitude and width of the lump, and $(c_{x,n}, c_{y,n})$ denote the center of the lump. We will consider that $c_{x,n}$ and $c_{y,n}$) are both sampled from a uniform distribution. Consider that N and σ_n are fixed.

To generate an object, we follow the following process:

i. Generate signal: Let the signal be at the center of the object, i.e. if the coordinate system was designed such that the origin falls at the center, then the center would be $0, 0$). Fix $\sigma_s = 1/4$ and $A_s = 3$. Insert these sampled values into Eq. (8). This leads to a continuous representation of the signal. Discretize this over 64×64 pixels. This yields the discretized signal.

ii. Generate background: Fix the number of lumps $N = 10$ and for each lump, fix σ_n to σ_s and $a_n = 1$. Next, for each lump, sample $c_{x,n}$ and $c_{y,n}$, each from a uniform distribution with range $(-1/2, 1/2)$. Insert these values into Eq. (10). This yields the continuous version of the background. Discretize this over 64×64 pixels. This provides the background object, also referred to as the signal-absent object.

iii. Add the signal and the background. This yields the object with the signal over a 64×64 grid. We refer to this as the signal-present object.

Repeat the above three steps for $J = 500$ times will yield 500 object realizations. You will need to perform the subsequent steps with both the signal-absent and signal-present images.

(b) **Image generation**: Using the imaging system model for System A or System B, conduct the following:

i. Generating the system matrix: Simulate the forward model for the system using the expansion function of the object as 32×32 pixels.

ii. Computing the SVD: Perform the SVD of this system matrix to obtain the singular values and the singular vectors.

iii. Obtaining the pseudo-inverse of the system operator: Use the SVD to obtain the pseudoinverse of the system operator.

iv. Generating noiseless projection data: Apply the generated system matrix to the generated objects to compute 500 instances of the projection data. Let us refer to the generated noiseless projection data as $\bar{\mathbf{g}}$.

v. Adding noise: In this step, we will be generating projection data at four different noise levels. For this purpose, we will be scaling the projection data generated in the above step.

A. Low-noise version: Scale the generated projection so that the sum of $\bar{\mathbf{g}}$ is 50,000.

B. Noisy version 1: Medium-noise: Scale the generated projection so that the sum of $\bar{\mathbf{g}}$ is 25,000.

C. Noisy version 2: High-noise: Scale the generated projection so that the sum of $\bar{\mathbf{g}}$ is 10,000.

D. Noisy version 3: Very high-noise: Scale the generated projection so that the sum of $\bar{\mathbf{g}}$ is 2,500.

. Next add Poisson noise to this data. In Matlab, this can be done by using the command POISSRND(GBAR).

vi. Reconstructing the image: Apply the pseudoinverse to the projection data at each noise level to obtain 500 instances of the reconstructed images.

We will split this dataset of 500×2 images into three parts. The first will be a training dataset, consisting of 300×2 images (i.e 300 images from the signal-present set and 300 images from the signal-absent set. Similarly, the validation dataset will consisting of 100×2 images, and the testing dataset will consisting of the remaining 100×2 images.

Typically, in imaging, we never have access to the noiseless data. Thus, usually, a network is trained to estimate an image at the low noise levels, given the images at higher noise levels. Thus, in the above setup, for each of these reconstructed images, the low-noise version will be considered as the ground truth for the training process.

(c) **Training the CNN:** Using any generic CNN-based denoising code, train a CNN to estimate the low-noise version given the (medium/high/very high) noisy image. You would use 300×2 images in the training set for training and the 100×2 images in the validation set for ensuring that there has been no overfitting. One such code is available at the following link. You would be required to train one CNN for each of the three noise levels (medium/high/very high). For the sake of notation, we will denote the CNN trained for the images at "Noise level K" by CNN_K.

(d) **Evaluating the denoising operation:** In this step, we will evaluate the performance of the three CNNs on the corresponding images in the testing set. As mentioned above, the performance will be evaluated on a signal-detection task. Perform the following operations for all the three noise levels mentioned in Q. 7(b)v.

i. Apply the trained CNN_K to the remaining 100 test images from the signal-absent set and 100 test images in the signal-absent set acquired at noise level K. Let us refer to the resultant images as "denoised" images.

ii. Visually, indicate your perception of whether the "denoised" images are less noisy and look more similar to the low-noise image. Do you think the CNN is effective in suppressing noise? You can also study this using quantitative metrics such as root mean square error between the low-noise image and the denoised image.

iii. We are providing you with code to evaluate performance on a signal-detection task at this link. To this code, you will input the 200 signal-present and signal-absent images (noisy or denoised) from the test dataset. The code outputs the metric of area under the receiver operating characteristics curve (AUC). The values of the AUC lie between 0.5 and 1, and the higher the value, the more accurate the performance on the detection task. Compute and plot the AUC for all four noise levels mentioned in Q. 7(b)v. Does performance on the signal-detection task show a similar performance as the visual perception results?

Note: If generating and processing 500 signal-present and 500 signal-absent images poses computational challenges, you can just generate 250 of each and split into training/validation/testing in the same ratio as with the 1000-patient dataset.

8. **Neural Fields**

In this class we have explored that image reconstruction algorithms rely on some form of discretization of an imaging system

$$g = \mathcal{H}f + e,$$

where $\mathcal{H} : \mathbb{U} \to \mathcal{D}$ is a continuous to discrete operator and e is an additive noise term. This is because neither humans or computers can deal with an infinite amount of information. Instead we approximate the infinite dimensional space $\mathbb{U}$ with a finite dimensional analog $\mathbf{U}$. Often $\mathbf{U}$ is a finite dimensional vector space. However, this vector space representation $\mathbf{U}$ usually needs to be extremely high dimensional to accurately represent arbitrary images and requires a lot of memory.

One way to avoid this memory is to use a **neural field**, a particular class of neural networks that represent an image as a continuous function. Mathematically we can think of our neural field as a neural network $\Phi_\xi \in L^2(\Omega)$ parameterized by a set of weights $\xi \in \mathbb{R}^n$ and arranged with some network architecture.

Besides being memory efficient, neural fields have a few other notable benefits.

- Neural fields, through set architecture set expectations for how an image should look leading to qualitatively better reconstructions and mitigate noise
- Neural fields are smooth and differential depending on network architectures
- Easy to implement with existing machine learning packages (Pytorch, or TensorFlow)
- Exact derivatives are computed using automatic differentiation
- Result in fully continuous representations (reconstructions that live in the infinite dimensional space)

(a) Implement a neural field architecture and use it to represent a chosen set of 5 grey-scale images, including the Shepp-Logan Phantom. First choose a neural field architecture and implement it in you chosen coding language (I would suggest doing this in Python using Pytorch.) You neural network should be of the form

$$\Phi_\xi(x) = \xi_D(\sigma(\xi_{D-1}(\dots\sigma(\xi_0(x))\dots),$$

where σ is a nonlinear function (activation function) that acts elementwise, ξ_j is an affine transformation (refereed to as a linear layer in machine learning) of the form

$$\xi_j(z) = A_j z + b_j$$

for some $A \in \mathbb{R}^{w_{j+1} \times w_j}$ and $b \in \mathbb{R}^{w_{j+1}}$. We define the **depth** to be D and the width to be $W = \max_j w_{j+1}$.

For your neural architecture, some suggestion are

- Choose your activation functions to be sinusoidal, sigmoid, hyperbolic tangent, or some other suitable choice.

- $w_0 = 2$
- Set $W = w_1 = \ldots = w_D$ to be some sufficiently large but not too large number (a good choice is 100 ish)
- $W_{D+1} = 1$
- D should be relatively shallow ($\sim$ 4-6)

Next, choose 5 square images, one of which being the Shepp-Logan Phantom, that are a couple hundred pixels on square length. Set these images to greyscale and make sure that they are a suitable datatype. Then train 5 individual neural fields to represent your chosen images by minimizing

$$\min_{\xi} \frac{1}{2} \sum_{j} |\Phi_{\xi}(x_j) - f_j|^2,$$

where x_j is the location of the j-th pixel and f_j is the value of the image in the j-th pixel.

Toy around with the parameters of your neural field for better results and compare the number of paramters in your neural field to the original number of pixels. Render your final learned images and compare them to the original image. Render the learned images with double the resolution of the original images. Discuss the steps you've taken and your analysis of the final rendered images.

(b) Use your neural field architecture for image reconstruction

i. Using your neural field architecture developed in the previous part, implement a method for noiseless image reconstruction. Choose an appropriate imaging operator, from the ones provided or one of your choice, that maps from a continuous to discrete domain. With your five images, generate measurements from your system of choice and create a neural field representation by minimizing

$$\min_{\xi} \frac{1}{2} \|g - H\Phi_{\xi}(X)\|^2.$$

How does this compare to the original image?

ii. Use your neural field architecture to implement a total-variation regularized image reconstruction from noisy measurements. Generate measurements g using the imaging system of your choice and add noise $e \sim \mathcal{N}(0, \sigma^2)$ for a relatively small standard deviation σ. For good reconstruction you will need to implement total variation (TV) regularization,

$$\mathrm{TV}(f) = \int_{[-1/2,1/2]^2} \|\nabla f(x)\| dx.$$

For neural fields this can be approximated as

$$\mathrm{TV}(f) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \|\nabla f(y)|,$$

where $\mathcal{Y}$ is a randomly chosen, discrete subset of $[-1/2, 1/2]^2$. Create a neural field representation by solving

$$\min_{\xi} \frac{1}{2} \|g - H\Phi_{\xi}(X)\|^2 + \alpha\, \mathrm{TV}(\Phi_{\xi}).$$

How does this compared to the direct representations and representations form noiseless image reconstruction?

(c) In you own words and based on this experience, draw conclusions on the following.

- What is your understanding of neural fields?
- What are their benefits?
- What are the disadvantages? Or what did you struggle with?
- What are some future applications you can think of?
- When would neural fields not be useful?