This coursework contains three modules. Modules 1 and 3 are common, and module 2 has two options (A and B). Each option is self-contained. ***You have to submit your answers for the three modules, but only one of the options for module 2, either A or B.*** *Do not submit work for both modules 2A and 2B, as you will not be assessed for both. The weighting of marks for each part is indicated in the right margin in square brackets, up to a maximum of 100 marks.*

The coursework will be submitted via a GitLab repository which will be created for you. You should write a report of no more than 3000 words to accompany the software you write to solve the problem. You should place all your code and your report in this repository. The report should be in PDF format in a folder called "report". You will be provided access to the repository until the deadline above. After this you will lose access which will constitute submission of your work.

The code associated with the coursework should be written in Python and follow best software development practice as defined by the Research Computing module. This should include:

- *Writing clear readable code that is compliant with a common style guide and uses suitable build management tools.*
- *Providing appropriate documentation that is compatible with auto-documentation tools like Doxygen/Sphinx.*
- *The project must be well structured (sensible folder structure, README.md, licence etc.) following standard best practice.*
- *Appropriate and robust unit tests for automatic validation of your code.*
- *Uses appropriate version control best practice, including branching for development and testing, and commit hooks to protect "main”.*
- *Appropriate containerisation to ensure portability of the project to other computers and operating systems if necessary.*

MODULE 1: Handling DICOM data

Converstion to numpy Your code needs to automatically convert the DICOM dataset (3D volume) of the 12 cases of the LCTSC Lung CT Segmentation Challenge shared with you via GitHub into a numpy array per case (patient).

[*Hint: you may want to use the* `pydicom` *methods seen in Practical2.*]
[*Remember that the GitHub repository is https://github.com/loressa/DataScience_MPhill_practicals/tree/master/Dataset*]

[20]

MODULE 2A: Working with CT backprojections

(a) Choose the central 2D slice for one of the patients from Module 1 and modify it to be able to do CT simulations, as seen in Practical 1. This will involve reshaping the data, but also converting it from Hounsfield Units (HUs) to linear attenuation coefficients (µ). In µ, air has value = 0, and water = 1.022 g/cm^3. This image will be used for the remaining of this module. [5]

(b) The most common algorithm to reconstruct tomography is Filtered Backprojection (FBP). This consists of a high-pass filter using a ramp filter on the sinogram, followed by a backprojection.

Implement the sinogram filter yourself and backproject using the techniques seen in Practical 2 (i.e. implement FBP). Use different filter choices, e.g. cosine, Hamming, Shepp-Logan, Hann, etc. The standard one "ram-lak" is a ramp filter. Compare the result with python's `iradon`, which with default inputs is the FBP algorithm. Provide an analysis with image quality metrics and different noise levels, implementing at least 3 filters (including "ram-lak"). [20]

[*Hint: In the following exercises, use either your FBP solution or* `iradon`]

(c) Reducing the dose of CT images is an active field of research. There are two clear ways to reduce the dose delivered: reducing the source intensity and reducing the total number of projections (rows in the sinogram). Simulate these two effects separately and together. Assume clinical dose is $I_0 = 10^5$ for comparison. Visualize, study, compare and discuss the results. [7½]

(d) Sometimes, the machine can not rotate the full circle. This is common in dental CT, breast tomosynthesis and image guided surgery. Simulate this effect, limiting the range of the rotation of the machine. Change it from 360 to 180, 100 and 60 degrees. Visualize, study, compare and discuss the results. [7½]

(e) In the lectures we learned that there are other ways to reconstruct, based on the variational description of the problem. Recall that the desired solution is:

$$\hat{\mu} = argmin_{\mu} D(\mu, p) + \alpha R(\mu),$$

i.e. we want to find a $\mu = \hat{\mu}$ such that the equation $D(\mu, p) + \alpha R(\mu)$ has the lowest value possible. For now, lets assume $R = 0$. Compute the gradient descent (GD) solution of the equation above, for $D(\mu, p) = \|A\mu - p\|^2$.

How does it perform compared to FBP? What about when we have low dose and low number of projections? Compare results visually and with image differences, and set the visualization color limits to the same, for a fair comparison. [15]

(f) Research has shown that updating the gradient descent algorithm using subsets (mini-batches) may produce faster convergence (i.e. we may get to a solution in less iterations). Modify the gradient descend algorithm implemented in the previous step into a subset GD. Split the sinogram in 10 subsets, and update the image using a loop, rather than a single line as in GD. How do these two algorithms compare? If you compute the distance function $D(\mu, p)$ per full sinogram update, which one is faster? How many iterations of the subseted GD you need to obtain the same image quality as 30 iterations of GD? And how long do each iteration take? [5]

MODULE 2B: UNet-based segmentation

Alongside with the images in the GitHub repository used for the Practicals and mentioned in Module 1, you can now also access the corresponding segmentations in numpy format (folder Segmentations). You will use them to train your own UNet-based model to segment lungs in CT images. You need to:

(a) Write a training method for the UNet architecture you created in Practical 3, with a custom loss function combining binary cross entropy and the soft Dice loss described in the practical session. [*Hint: Remember that you need to apply the activation function (use sigmoid) to the output of the UNet. If using* `nn.BCEWithLogitsLoss()`*, this method already adds the sigmoid function internally.*] [20]

(b) Divide the dataset provided into train (2/3) and test datasets (1/3). [5]

(TURN OVER

(c) Train your UNet for a maximum of 10 epochs. Use batch size of 3 (if enough memory, otherwise reduce) and a learning rate of 0.1. Make a plot showing the loss and the accuracy per epoch. Take into account that this step requires several hours of running and that you need GPUs, so you need plan when to do this step accordingly and use the HPC job submission system. [*Hint: For the accuracy you will need to make predictions with the model (use part e) below), and you may want to use* `from torchmetrics.classification import BinaryAccuracy`] [15]

(d) Write a method to make the prediction (i.e. evaluating the trained UNet model) for the test dataset and to compare the prediction to the ground truth segmentations, computing the Dice Similarity Coefficient (DSC) between the two. You may use either your trained model from previous questions or the pre-trained model shared with you in Practical 3.

- -Plot the accuracy and DSC per 2D slice, separating the results for the train and test datasets. [12]
- -Find examples of 2D slices in different patients with i) best DSC values, ii) worst DSC values and iii) in-between values. Illustrate at least 3 examples of each category in a figure showing the real image (left), the ground truth segmentation (middle) and the prediction (right). [8]

[*Hint: After applying the sigmoid to the prediction, remember to convert it into a binary mask. You can use for example a threshold of 0.5.*]

MODULE 3: Final Report

Your final report should present a clear, scientific summary of the following points that you have worked on during the previous modules. These marks relate to the analysis, critical discussion and presentation of results, rather than programming or implementation as before.

(a) Report and justify your choices for each step of the previous modules. [8]

(b) Create an analysis report presenting the illustrations generated in the previous modules and what you have learned doing this coursework. Discuss your results. [8]

(c) Were your results acceptable? What could be the potential causes for imperfect results? What could you do to improve them in the future? [4]

END OF PAPER