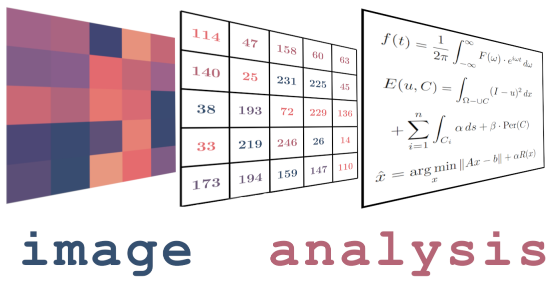
UNIVERSITY OF CAMBRIDGE



MPhil in Data Intensive Science

Submission: 11:59pm on 22nd June 2025

- Coursework Assignment

Dr. A. Biguri, Dr. P. Cañizares, Dr. S. Mukherjee, Dr. M. Roberts, Dr. A. Breger

The coursework consists of 3 Modules. The final report should explain how you have arrived to the conclusions of your answers and the choices for each step. The report should include your analysis, presenting the illustrations generated, a discussion and what you have learned doing this coursework. **This has to be done for each module separately**. *The report is a total of [60] 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.

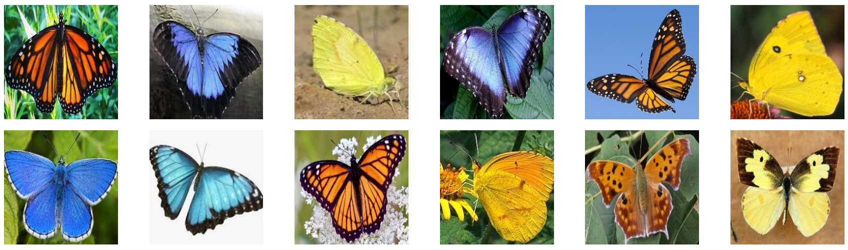
The code associated with the coursework should be written in Python and follow best software development practice as defined by the Research Computing module. This should include:

- *Writing clear readable code that is compliant with a common style guide and uses suitable build management tools.*
- *Providing appropriate documentation.*
- *The project must be well structured (sensible folder structure, README.md, licence etc.) following standard best practice.*

- *Uses appropriate version control best practice, including branching for development and testing, and commit hooks to protect "main”.*

Module 1: Classical Image processing

An entomologist has come to you with a collection of images of butterflies (attached with the coursework) and have asked you to help them clean up the data they acquired in their latest trip. They want help with some tasks.



They have only shown you a small sample of their dataset, so its important that however you help them is completely automatic, and does not rely in manual selection. You can assume any finetuned value that works for these images will work for all the new images, but the process should be automatic. However, this entomologist dad was insulted by ChatGPT once, so using any type of machine learning solution to help them is strictly forbidden. You can only use classical image processing techniques, from skimage or your own.

1.1 - Color classification. They have captured Butterflies of 3 different colors. Can you classify them by color? there should be 4 pictures of each type. [5]
1.2 - Background removal. They are not interested in the background, so can you please remove it and only keep the butterflies? [5]
1.3 - Collection display. It would be great if we can make an image where all the butterflies of each type are displayed with a single background (your choice). Note that entomologist are very picky, so they want to be able to put as many images of butterflies they want (we have now 4, but your function should take N images of each class) and also be able to chose the resolution of the final image (in pixels). You can chose which arrangement makes more sense. [5]
1.4 - Odd one out. Your extensive experience of looking at butterflies is quite useful! You noticed that in each of the 4 images in every class, one of the butterflies is of a different species than the others! Can you find a way to differentiate it from the other 3 automatically? [5]

Please remember to explain to the entomologist why you chose the methods you used, and not different solutions.

Note: there is no unique correct answer to this exercise, nor a ground truth that your results will be compared against. This is about you exploring methods, choosing them carefully and showcasing your image processing skills. If you convince us that your method is a sensible way of solving the task, you get full marks, even if the results are not perfect.

(TURN OVER

Module 2: Data-driven regularization for inverse problems

After insisting, you have convinced the entomologist that using some machine learning is not that bad. Particualrly given how some of the entomologist images are not just noisy, but really blurry and acquired with a broken camera. The entomologist agrees on using machine learning, but only if used appropriately under some rigorous mathematical framework, like PnP.
Thus, you will leverage a powerful image denoiser (which is based on a pre-trained deep neural network) to solve more challenging inverse problems such as image deblurring and inpainting, and hopefully restore the butterfly images.

Alternating directions method-of-multipliers (ADMM)

Consider the variational problem

$$\min_x \frac{1}{2} \|Ax - y\|_2^2 + g(x) \tag{1}$$

for image reconstruction. Note that (1) can be reformulated as

$$\min_{x,v} \frac{1}{2} \|Ax - y\|_2^2 + g(v) \quad \text{subject to} \quad x = v. \tag{2}$$

The augmented Lagrangian for (3) is given by

$$L_\eta(x, v, u) := \frac{1}{2} \|Ax - y\|_2^2 + g(v) + u^\top (x - v) + \frac{\eta}{2} \|x - v\|_2^2, \tag{3}$$

where $\eta > 0$ and u denotes the dual variable. Then, recall that the ADMM iterative updates for (1) consist of the following three steps:

$$\begin{aligned}
&x\textbf{-update:} \quad x^{k+1} = \arg\min_x L_\eta(x, v^k, u^k) = \left(A^\top A + \eta I\right)^{-1} \left(A^\top y + \eta\, v^k - u^k\right), \\
&v\textbf{-update:} \quad v^{k+1} = \arg\min_v L_\eta(x^{k+1}, v, u^k) = \operatorname{prox}_g^{\frac{1}{\eta}} \left(x^{k+1} + \frac{1}{\eta} u^k\right), \quad \text{and} \\
&u\textbf{-update:} \quad u^{k+1} = u^k + \eta \left(x^{k+1} - v^{k+1}\right).
\end{aligned}$$

Here, $\operatorname{prox}_\gamma^g(z) = \arg\min_v \frac{1}{2}\|v - z\|_2^2 + \gamma\, g(v)$. After redefining the dual variable as $\frac{1}{\eta} u$, we note that the ADMM algorithm, in its scaled form, entails the following iterative updates:

$$\begin{aligned}
&x\textbf{-update:} \quad x^{k+1} = \left(A^\top A + \eta I\right)^{-1} \left(A^\top y + \eta\, (v^k - u^k)\right), \\
&v\textbf{-update:} \quad v^{k+1} = \operatorname{prox}_g^{\frac{1}{\eta}} \left(x^{k+1} + u^k\right), \quad \text{and} \\
&u\textbf{-update:} \quad u^{k+1} = u^k + \left(x^{k+1} - v^{k+1}\right).
\end{aligned}$$

Plug-and-play (PnP)-ADMM

In the PnP variant of the ADMM algorithm, we replace the proximal operator in the v-update step with an off-the-shelf image denoiser D. We will use a pre-trained deep denoiser (based on a U-net architecture, available at `https://drive.google.com/file/d/1FFuauq-PUjY_kG3iiiHfDpHcG4Srl8mQ/view?usp=sharing`) for this purpose. Use the same U-net implementation as available in `https://github.com/facebookresearch/fastMRI/blob/main/fastmri/models/unet.py` and load the pre-trained model using the following lines of code:

```
model = Unet(3, 3, chans=64).to(device)
model.load_state_dict(torch.load('denoiser.pth', map_location=device))
```

You may use one of the butterfly images from Module 1 as the ground-truth for the exercises in this module, to simulate the image degradation that happened to the entomologist.

The PnP-ADMM algorithm is summarized in the following:

1. **Initialize**: $x, u, v = 0$
2. **Iterate until convergence**:

$$\begin{aligned} x &\leftarrow \left(A^\top A + \eta I\right)^{-1}\left(A^\top y + \eta\,(v-u)\right) \\ v &\leftarrow D(x+u) \\ u &\leftarrow u + (x - v) \end{aligned}$$

The x-update step is implemented by solving the linear system $Cx = d$ using the conjugate-gradient (CG) method, where $C := A^\top A + \eta I$ and $d = A^\top y + \eta\,(v-u)$. The code for implementing CG is given below.

```
def conjugate_gradient(A, b, x0, max_iter, tol):
"""
CG for solving Ax=b.
Here, the argument A is a function that returns Ax
"""
    x = x0
    r = b-A(x)
    d = r
    for _ in range(max_iter):
    z = A(d)
    rr = torch.sum(r**2)
    alpha = rr/torch.sum(d*z)
    x += alpha*d
    r -= alpha*z
    if torch.norm(r)/torch.norm(b) < tol:
    break
    beta = torch.sum(r**2)/rr
    d = r + beta*d
    return x
```

(TURN OVER

Exercise 2.1 (deblurring) In image deblurring, we seek to recover a sharp image x from its blurry measurement $y = Ax$, where A represents a blur operator.

1. Consider an image deblurring problem where A corresponds to a motion blur kernel of size $p \times p$ (i.e., a kernel of size $p \times p$ with all entries of the kernel equal to $\frac{1}{p^2}$). Find the mean-squared error (MSE) of the reconstructed image (w.r.t. the ground-truth image) using PnP-ADMM for different values of p (say, $p = 7, 13, 17$). The modules implementing A and $A^\top$ are given to you. You may normalize the input image to the range $[0, 1]$ and choose $\eta = 10^{-4}$. You have to write a function that implements PnP-ADMM and clip its output to the range $[0, 1]$ in the end. [4]
   The following function implements the forward (A) and the adjoint ($A^\top$) operators corresponding to a given blur kernel.

```
def conv2d_block(kernel, channels, p, device, stride=1):
    """
    Returns nn.Conv2d and nn.ConvTranspose2d modules from 2D kernel, such that
    nn.ConvTranspose2d is the adjoint operator of nn.Conv2d
    Arg:
        kernel: 2D kernel, p x p is the kernel size
        channels: number of image channels
    """
    kernel_size = kernel.shape
    kernel = kernel/kernel.sum()
    kernel = kernel.repeat(channels, 1, 1, 1)
    filter = nn.Conv2d(
        in_channels=channels, out_channels=channels,
        kernel_size=kernel_size, groups=channels, bias=False, stride=stride,
        padding=p//2
    )
    filter.weight.data = kernel
    filter.weight.requires_grad = False

    filter_adjoint = nn.ConvTranspose2d(
        in_channels=channels, out_channels=channels,
        kernel_size=kernel_size, groups=channels, bias=False, stride=stride,
        padding=p//2,
    )
    filter_adjoint.weight.data = kernel
    filter_adjoint.weight.requires_grad = False

    return filter.to(device), filter_adjoint.to(device)
```

2. Take the kernel-size to be $p = 13$. Add Gaussian noise with zero mean and $\sigma = 0.01$ to the blurry image y, and run the PnP-ADMM algorithm to reconstruct. How does the reconstructed image compare with the “no-noise in the measurement" case in terms of MSE? Write your interpretation of the result. [2]

Exercise 2.2 (inpainting) The task in image inpainting is to recover the missing pixels in an image. Here, the forward operator A involves applying a random binary mask M with the zeros representing missing pixels, and the corrupted input image is given by $y = M \odot X$, where $\odot$ denotes element-wise multiplication of the mask M with an image X. Note that this operator can equivalently be expressed as $y = Ax$, where A is a diagonal matrix with zeros corresponding to the locations of the missing pixels and ones at the remaining places, and x denotes the image in its vectorized form. The adjoint of this forward operator is clearly $A^\top = A$, which can be implemented by applying the same mask on the argument. You may use the following implementation of the forward and adjoint operators for the inpainting problem.

```python
# Inpainting
channels = 3 # for color images
mask = torch.rand(1,channels,h,w).to(device) # 3 channels and has hxw pixels
mask = mask < 0.4 # this means that approx. 40% pixels will be retained

def forward(x): # ensure that the image is of size 1x3xhxw
    return x*mask # element-wise multiplication with the image
adjoint = forward
```

1. Apply the PnP-ADMM algorithm to solve the inpainting problem with 40%, 60%, and 80% missing pixels, and compare the MSE of the corresponding reconstructed images w.r.t. the ground-truth. [3]

2. Recall that in class, we learned a variant of PnP algorithms (called regularization-by-denoising (RED)), where a regularizer $\rho(x)$ is formed directly using a pre-trained denoiser $D(x)$ as $\rho(x) = \frac{1}{2}x^\top(x - D(x))$. If the denoiser is Jacobian-symmetric and locally homogeneous, the gradient of this regularizer is given by $\nabla\rho(x) = x - D(x)$, i.e., the denoising residual.

   (a) Assuming the above expression for the gradient of the regularizer, implement the gradient descent algorithm for minimizing $J(x) = \frac{1}{2}\|y - Ax\|_2^2 + \lambda\,\rho(x)$, where $\lambda = 0.1$, and A is the inpainting mask corresponding to the case when 60% of the pixels are zeroed-out randomly. Use a step-size of $\eta = 1$. We will refer to this version of PnP as PnP-RED. [2]

   (b) Compare the output of PnP-ADMM and PnP-RED in terms of the MSE of the reconstructed image. [2]

   (c) For the denoiser chosen in this exercise, is it correct to set $\nabla\rho(x) = x - D(x)$? [2]

Exercise 2.3: Investigate whether the MSE increases or decreases if you run more iterations of PnP-ADMM. Plot the MSE of the reconstructed image as a function of the PnP-ADMM iterations. Explain/interpret the variation as a function of the number of

(TURN OVER

iterations. Propose a strategy to mitigate the effects that you see in the image quality. [5]

Module 3: Pitfalls and Challenges

Exercise 3.1 Image Quality Assessment

The entomologist is quite impressed with your results so far, but they are not entirely convinced that the image you say is best (the lower MSE one) is actually the best. They want you to do a much more detailed analysis on which images are really, the best ones (the exhibition is very important for the entomologist!).

[3.1.a] Discuss why PSNR and SSIM might not be sufficient image quality measures to evaluate the performance of the results in Module 2 (deblurring and inpainting). Which full-reference (FR) and no-reference (NR) IQA measures could be useful to add for these tasks? Evaluate your results again with your chosen set of FR and NR-IQA measures and compare it to the evaluation with PSNR and SSIM. [5]

[3.1.b] Choose a butterfly from the data set in Module 1 and create degraded versions where (1) PSNR and (2) SSIM give the same evaluation although the type of degradation varies highly. What happens with these values when the background pixels are removed, what can you deduce? [5]

Exercise 3.2 Pitfalls and challenges in building ML/DL systems for image analysis
To convince the entomologist that using machine learning for imaging is not that bad, you are going to show them your mastery of the tools, using the MNIST dataset. You will show them how well you understand the challenges of ML in imaging. The notebooks and data can be found here. Open `ex3.2.ipynb` and run from start to end with a GPU. This code uses the MNIST dataset to train a neural network (`case=a`) for classification of the numbers. It should converge in roughly 20-40 epochs.

[3.2.a] Explain what you see, for `case=a`, with respect to: [5]

- what this code is doing.
- how common pitfalls are mitigated.
- how data partitions are used.
- biases that may be introduced by the approach taken.
- model performance and training efficacy, in particular what metrics would be most appropriate.
- give some ideas for how you could improve training, including by potentially modifying the code - providing an explanation and motivation for them.

MNIST some images green and red, 0s are misclassified often, as all the training and validation is green, and all the test is red, for 0. others are all red or green.

[3.2.b] Implement your suggested changes for `case=a` and retrain the model to achieve a near zero loss. For the two worst performing classes can you identify key reasons performance is low and how you would make the model robust to them? [3]

[3.2.c] Now consider `case = b` and run the notebook from start to end. Explain what is different between the two cases and give detail for whether you would anticipate that the `case = b` model would perform worse, better or the same as in `case = a`. Describe any changes to the code for `case = b` that you think are needed to boost performance in the worst performing classes and report how your best model performs. [2]

END OF PAPER