Inverse Problems CW2

Student 23188618

February 2024

This report focuses on inverse problems in the context of computed tomography. Traditional approaches to solving these inverse problems involve analytical methods, such as filtered backprojection (FBP) and iterative reconstruction algorithms like Haar wavelet denoising. As this problem constitutes a forward-adjoint pair, the data cannot be exactly recovered. This is further elucidated by the fact that frequently these reconstructions take place in the context of limited data, missing segments, and noise. These methods will all be discussed in this report, and will be compared with their modern deep learning counterparts.

1 Part A

1.1 Radon Transform and Back-Projection

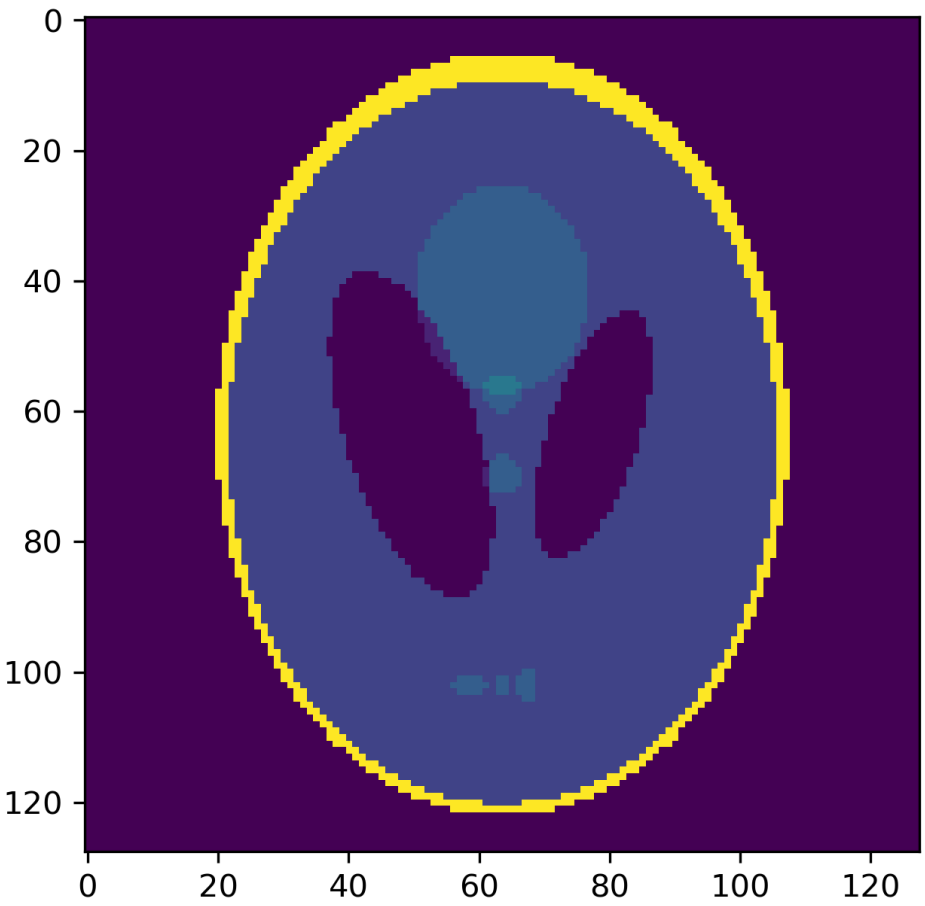

Figure 1: Shepp-Logan phantom of size 128×128, which is f_{true} in this investigation.

Sinograms were obtained by applying the Radon transform to the Shepp-Logan phantom image and can be seen in Fig 2. The sinograms have a distinct pattern, with high-intensity regions representing

the edges and boundaries of the original phantom image.

The number of projection samples (`det count`) is related to the granularity with which data is collected, such that more samples per projection can be obtained either by improving the detector's ability to record data or by changing the scan parameters to allow for finer resolution in data collection. It can be seen that lowering the number of samples to 95 (see Figure 2) crops out information that would be important for image reconstruction.

With 95 samples, each projection (also known as a sinogram row) corresponds to a vector of 95 values. Since there are 180 angles, each angle contributes one row to the sinogram. Therefore, the sinogram matrix has dimensions of 180×95. Similarly, with 120 samples, the sinogram matrix will be of size 180×120.

The regions with 0 intensity can be considered unimportant for reconstruction. To mimic situations where data might be corrupted or lost, we used 95 samples in latter parts of this investigation.

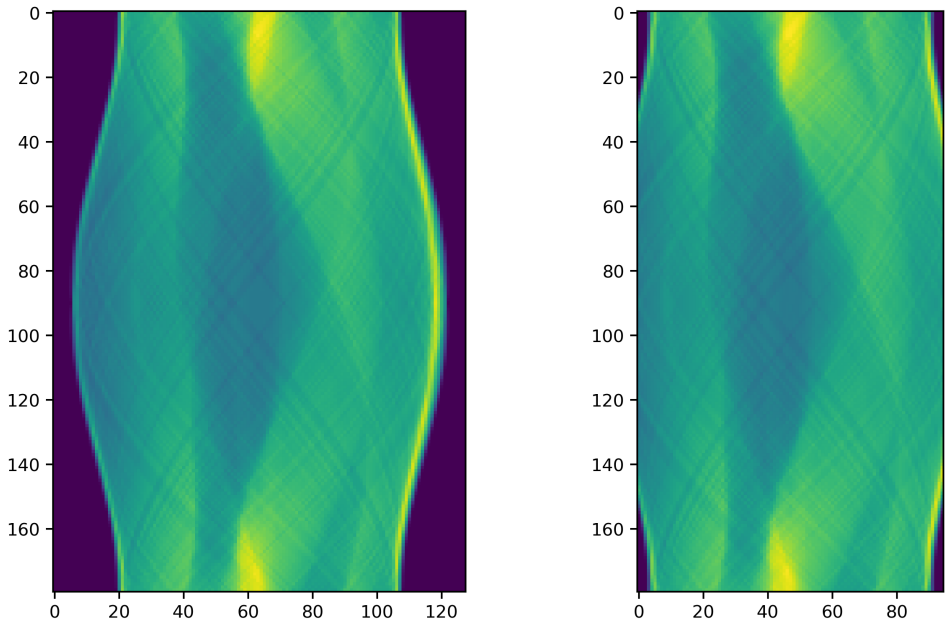

Figure 2: Sinograms generated by applying the Radon transform to the Shepp-Logan phantom image for 180 degrees at 1-degree intervals with the number of detector samples = 128 (left) and 95 (right).

In the sinogram (data) domain, the difference between images will be referred to as the residual, $Af_{true} - g$, whereas in the image domain the difference $f_{true} - f_{recon}$ will be referred to as the delta.

1.1.1 Unfiltered Back-Projection

The unfiltered back-projection is computed by applying the inverse Radon transform to the sinogram data and is shown in Figure 3. The size of the reconstructed image is the size of the original image, 128×128, The resulting image provides an estimate of the original phantom. However, the image has a halo effect and lacks sharpness, indicating the loss of high-frequency information. To counteract this effect, filtered backprojection uses a high pass filter in the frequency domain.

The corresponding sinogram, which was used to calculate the residual, is also seen to have the characteristic halo effect. This can be seen in the Appendix.

1.1.2 Filtered Back-Projection

To improve the reconstruction quality, a filtered back-projection was performed. This filter identifies sharp edges in the projection (and consequently, in the underlying slice) while disregarding flat regions.

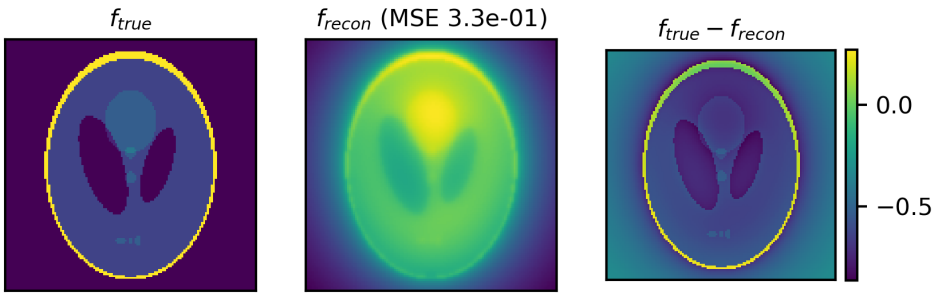

Figure 3: Unfiltered back-projection reconstruction obtained by applying the inverse Radon transform to the sinogram of f_{true}. The delta between the original image (left) and the transform (centre) is displayed on the right.

By generating negative pixels at the edges, the high-pass filter eliminates the additional blurring induced by backprojection.

The resultant image exhibits enhanced sharpness and reduced blurriness compared to the unfiltered back-projection. The filtered back-projection effectively suppresses the low-frequency artifacts and recovers the high-frequency details, leading to a more accurate reconstruction of the original phantom. However, there is a clear inability to perfectly reconstruct the very highest intensity pixels such as the circular border, which can be observed in the difference between the original and reconstructed image Fig. 4 c. There are also subtle streaks that can be seen in the reconstruction (and corresponding delta). The streaks could be a form of aliasing artifact, which occurs when the number of projection samples is too low to accurately capture the structure of the object being imaged.

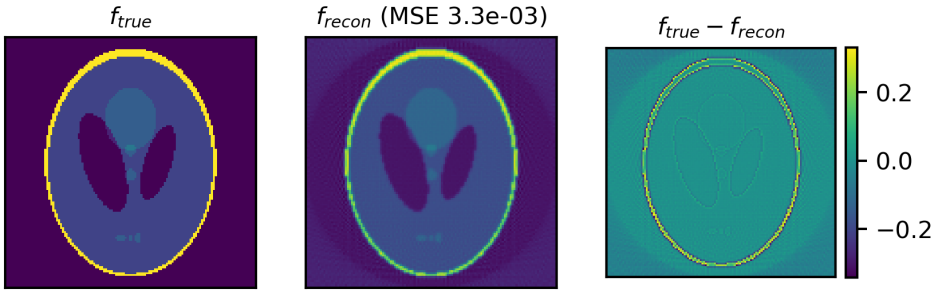

Figure 4: Filtered back-projection reconstruction obtained by applying the inverse Radon transform to the sinogram of f_{true}. The delta between the original image (left) and the transform (centre) is displayed on the right.

The corresponding sinogram, which was used to calculate the residual, resembles a conventional sinogram, however there are still some areas such as the sinogram borders which show a loss of information. This can be seen in Appendix.

1.1.3 Noisy measurements

Poisson noise was added to the sinogram data before applying the filtered back-projection. The relationship between the parameter θ and the Poisson noise can be understood through the properties of the Poisson distribution. The probability mass function (PMF) of the Poisson distribution is given by:

$$P(X = k) = \frac{e^{-\theta} \cdot \theta^k}{k!}, \quad \text{for } k = 0, 1, 2, \ldots$$

Here, X represents the random variable following a Poisson distribution, θ is the parameter controlling

the rate of occurrence, and k is the number of events. As θ increases, the mean signal intensity increases, leading to a higher signal-to-noise ratio (SNR) in the reconstructed images. We can see how varying this parameter impacts reconstruction quality in Figure 5. At the lowest θ value, $\theta = 30$, the reconstructed image is severely corrupted, with the phantom's features almost entirely obscured by noise.

We can also observe that for increasing noise, the MSE values increase, but in all cases the filtered backprojection most closely resembles the original phantoms.

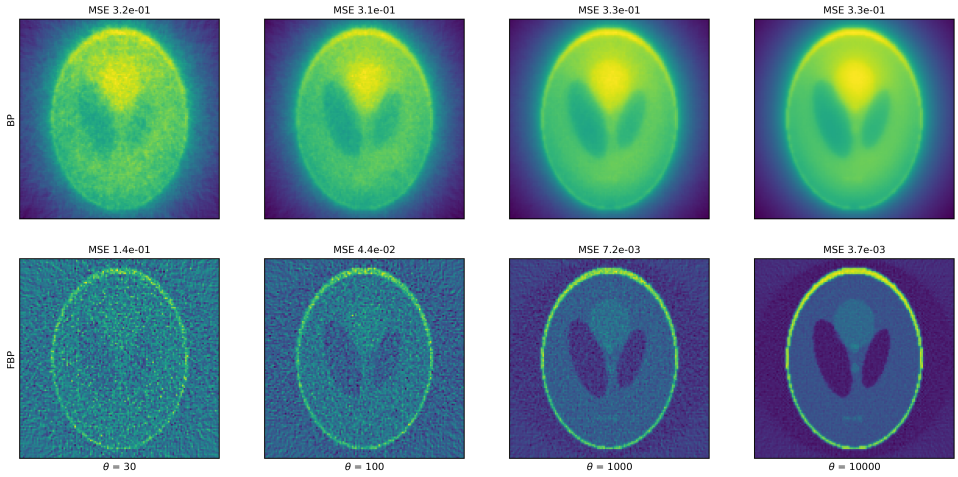

Figure 5: Unfiltered (top) and filtered (bottom) backprojections of noisy sinograms for varying levels of Poisson noise.

1.2 Matrix form of the Radon Transform

To calculate the explicit Radon Transform for a 64×64 Shepp-Logan image, the `explicitradon()` function was implemented (see Appendix).

The explicit Radon transform matrices, and their full corresponding singular value spectra, are plotted for one specific sparse angle case (every 4th angle in a range of 180 degrees) and one limited angle case (45 angles in a range of 45 degrees). These can be seen in Figure 6 and Figure 7. The sinogram

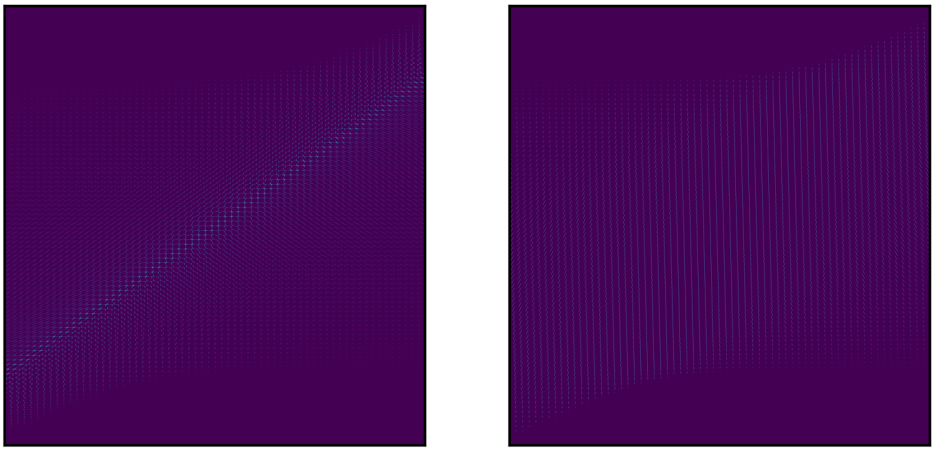

Figure 6: Explicit Radon transforms for sparse and limited angle cases.

matrices, calculated with 45 angles and 95 samples has 4275 elements, where each row in the sinogram corresponds to a different angle, and each column represents a different sample along the detector.

In the sparse angle case, the matrix appears to have more sparsely distributed values across the image but with a lower intensity. In contrast, the limited angle case shows a more concentrated band

of values with higher intensity in a narrower range of pixels along a diagonal. These observations intuitively make sense in the context of the different angle ranges.

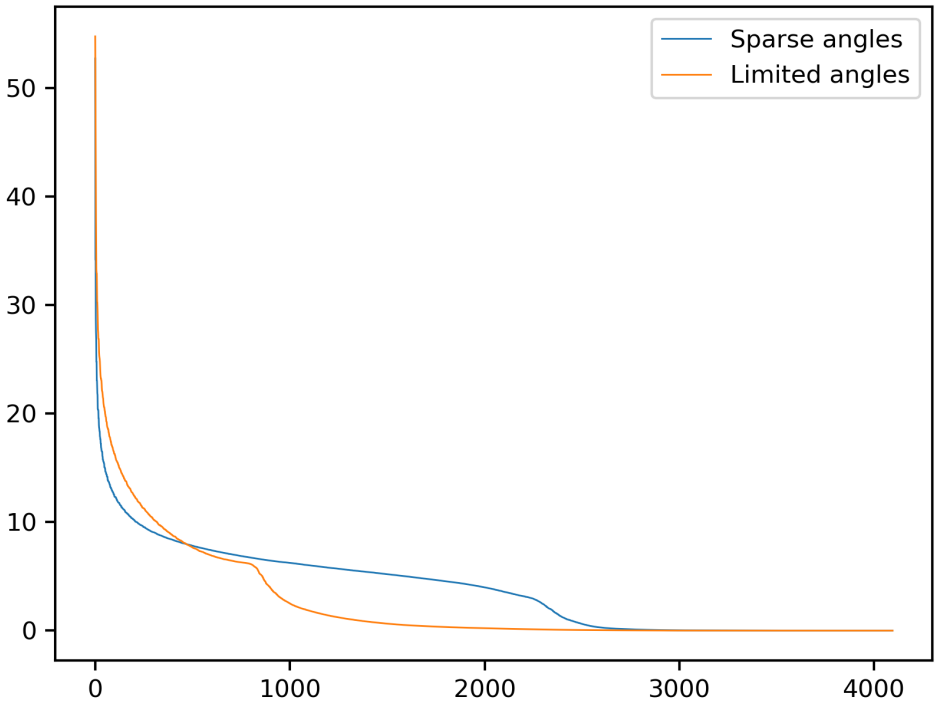

Figure 7: SVD singular values for the explicit Radon transform in the sparse and limited angle cases.

For the sparse angle case, the singular values decrease more gradually. There is a slower decay from the largest to the smallest singular values, and the first 2000 or so also have non-negligible values, suggesting that the information is more evenly distributed across the different singular vectors in this case.

However, for the limited angle case, there is a much sharper initial decay in the singular values; the first few singular values are significantly larger than the rest, which quickly drop to near zero before reaching the 1000th value. This implies that most of the information in the limited angle Radon transform is concentrated in the first few singular vectors corresponding to these dominant singular values.

Next, we show how the first 6 singular values of the explicit Radon transform change when the range and number of angles are varied.

In Figure 8, the range of degrees is held constant at 180, however the number of angles is varied from 1 angle every 4 degrees, one every 2 degrees, 3 angles every 4 degrees, and finally the full angle case which is shown for comparison's sake. There is a clear trend where increasing the number of angles leads to larger singular values across the board, with each line decaying at the same rate but offset from the rest.

In Figure 9 we can see that the first singular value is lower than that of the sparse case, meaning it carries less information overall, and we can also observe that while the first singular value for all the different angle ranges starts at roughly 52, they decay at different rates, with the larger angle range decaying fastest compared with an angle of 90 degrees for example.

1.3 Matrix-free regularised least-squares solver for the Radon Transform

To use the Krylov GMRES solver for the regularised least-squares problem, we firstly modify our forward operator to the context of Radon transforms. We also define a new adjoint oeprator, as this operation is not self-adjoint, which is displayed in the `ATy()` function in the Appendix.

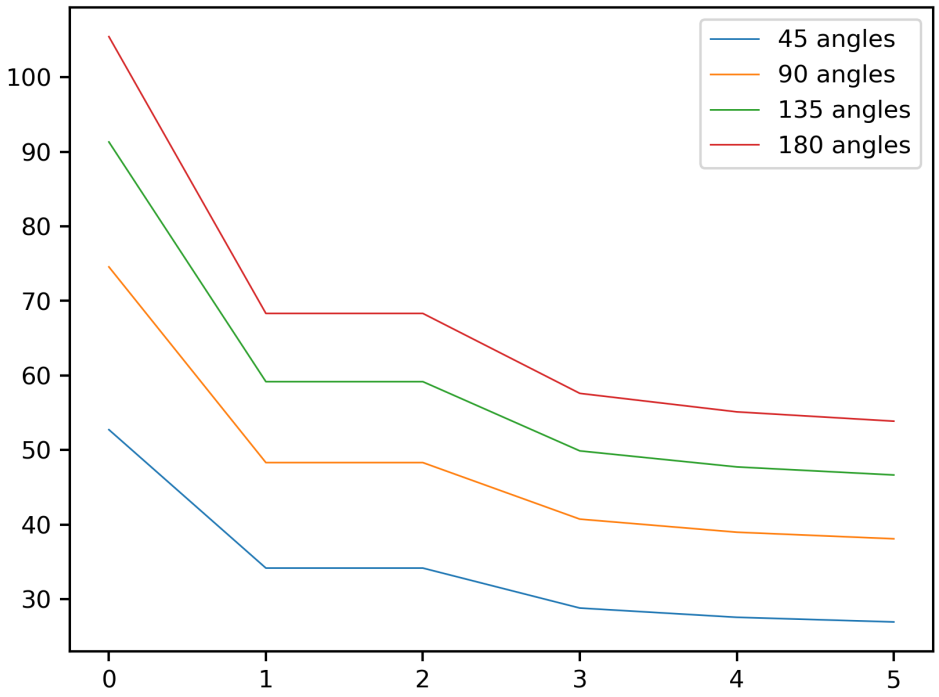

Figure 8: First 6 singular values for the explicit Radon transform in the sparse angle case.

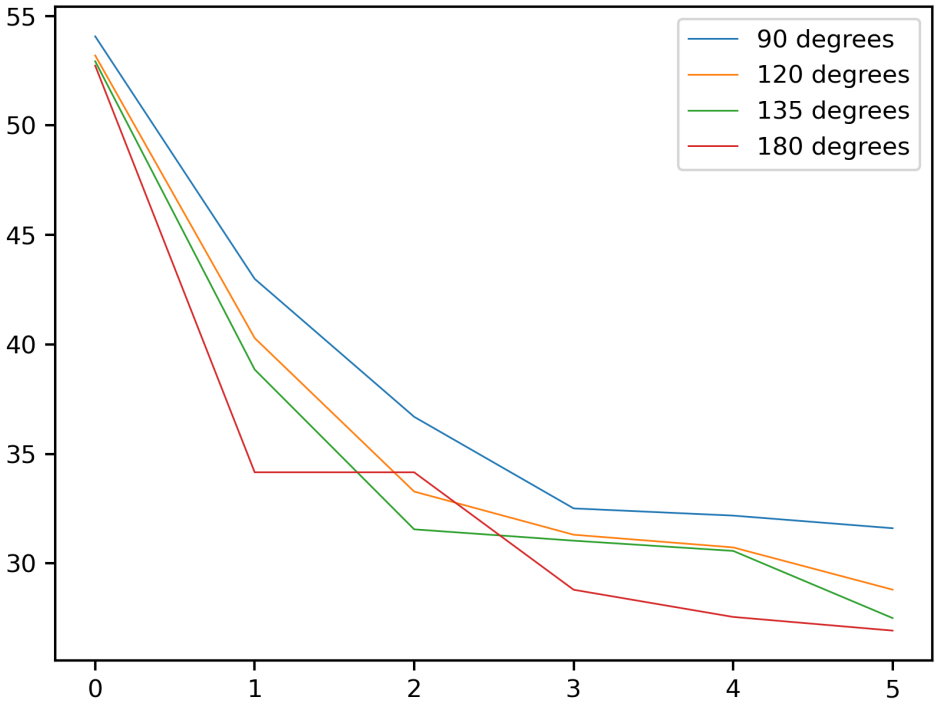

Figure 9: First 6 singular values for the explicit Radon transform in the limited angle case.

The final step is to choose an appropriate α parameter for regularisation. The method chosen was the L-curve, for its geometric interpretation and simple translation to several problems involving forward-adjoint operations. The optimal α was significantly larger than for a deconvolution problem, ranging between 0.32 and 1 as shown in Figure 10. To ensure the best results, the best α values were stored separately for the sparse and limited angle cases.

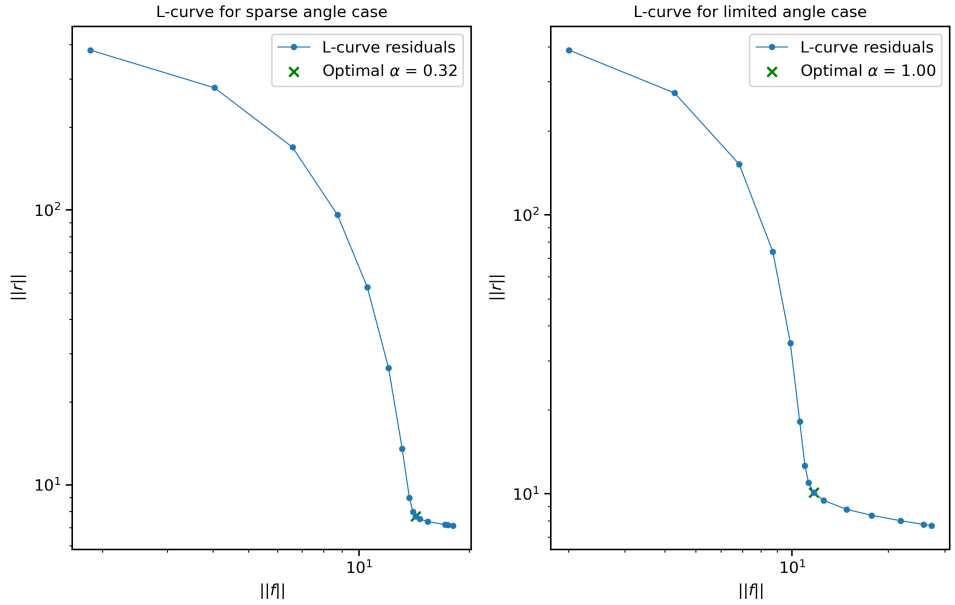

Figure 10: L-curve residuals for the sparse angle case (left) and limited angle case (right). The optimal α is chosen for the residual at the elbow of the curve.

1.3.1 Sparse case

For these experiments, the noise level was fixed at $\theta = 10,000$. The iterative solver took a large number of iterations, between 200-400, to converge.

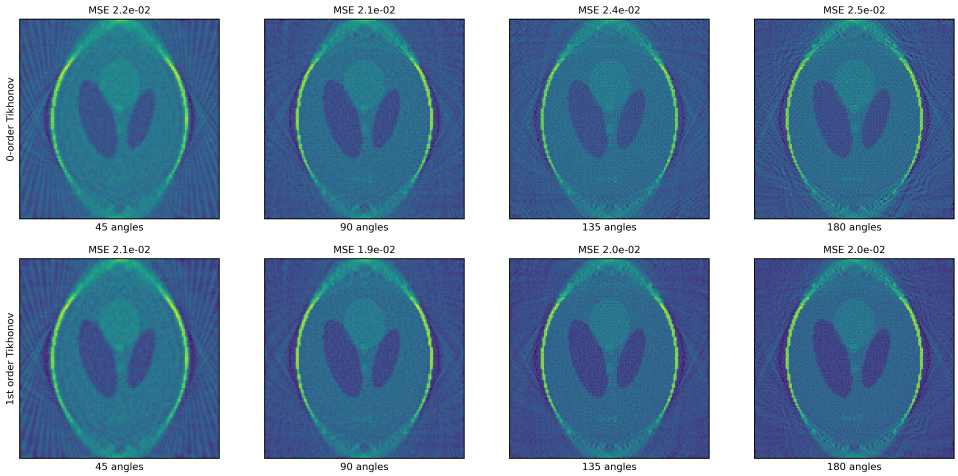

Figure 11: Unfiltered back-projections solved by a least-squares solver in a sparse angle case. The top row has 0-order Tikhonov regularisation while the bottom row has 1st order Tikhonov (gradient) regularisation.

In the sparse angle case (Figure 11), the reconstructed images demonstrate a clear improvement in quality and detail as the number of angles increases from 45 to 180. In the best case scenario (rightmost phantom, with 180 angles and a 180-degree range), the solver reached an MSE value of 2×10^{-2}. In the leftmost image, the detector turning round the phantom is evident from the lines around the phantom suggesting that that is how each projection might have been captured.

Interestingly, though the image becomes more crisp with more angles, the MSE values increase for 0-order Tikhonov regularisation. The 1st-order regularization consistently yields smoother images and lower MSE compared to the 0-order approach, indicating its effectiveness in suppressing noise and performing an acceptable backprojection.

1.3.2 Limited angle case

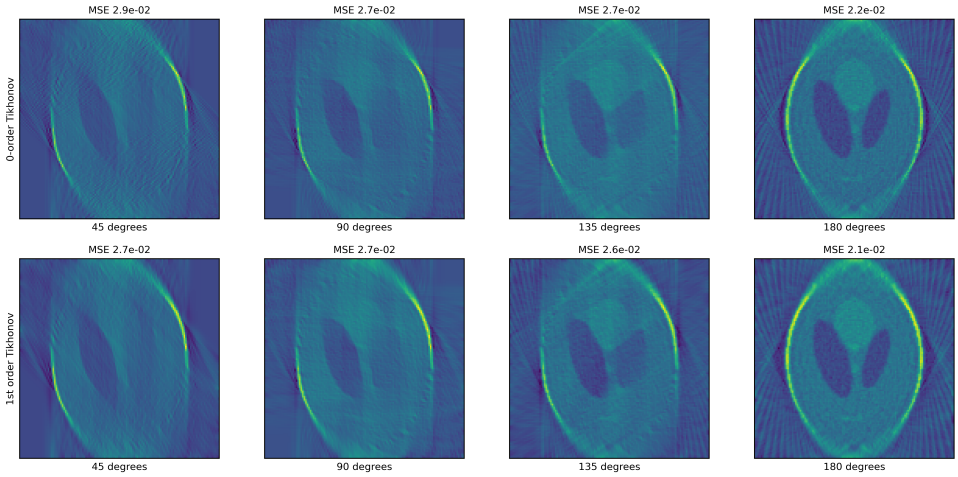

Figure 12: Unfiltered back-projections solved by a least-squares solver in a limited angle case. The top row has 0-order Tikhonov regularisation while the bottom row has 1st order Tikhonov (gradient) regularisation.

Limited angle reconstructions are more challenging than sparse angle cases, purely because of the irretrievable and often large missing wedge of data. Even with only 25% of angles missing, the GMRES solver shows heavy streaking across the phantom. We can also see high definition in the regions along the off diagonal, suggesting that these regions were within the limited angles range. There is a tradeoff between resolution and image quality for the varying angle ranges, as for the 45-degree range, when we have 45 angles (leftmost phantom) we can observe that the parts of the image which we *can* see are very crisp, albeit completely dominated by streaks. As the angle range increases, the high resolution areas along the off diagonal become broader while the overall resolution for those areas goes down because there are fewer angles in that range.

By fixing the number of angles to 45 in this case, the situation may seem worse than it actually is, as in the sparse case, the maximum angle was always the best case scenario (180 degrees). Whereas in ours, even with 135 degrees we still only have 45 angles, which is more of a mix between limited and sparse scenarios. Thus, It may be more illuminating to study the limited angles in a scenario where there is 1 degree per angle vs what is presented here.

1.4 Haar wavelet denoiser

1.4.1 Haar Wavelet Coefficients

The Haar wavelet decomposition involves applying a scaling function and wavelet to a given signal at different scales and translations, resulting in a set of approximation (ie. levels of granularity) and detail (vertical, horizontal and diagonal) coefficients. The coefficients are organized in an order of increasing coarseness, with each level corresponding to a specific scale or frequency range, and are defined as

$$c_{j,k} = \frac{1}{\sqrt{2^j}} \int_{-\infty}^{\infty} f(t)\psi(\frac{t - 2^j k}{2^j})dt$$

where j and k are the scale and translation parameters, respectively. The coefficients from the Haar wavelet decomposition of the Shepp-Logan phantom are visible in Figure 13. We can see that the coefficients go from highly detailed, capturing details that are difficult to pick up by eye, to very coarse, such that for coefficient 7 there is only a 1-pixel value for each detail coefficient. Coefficient 1 captures information along the border of the phantom, which is very high frequency and needs a high level of detail to remain sharp and strongly differentiated from the rest of the image. Of note is coefficient 3, which captures the circular border of the phantom at the least granular level. It is

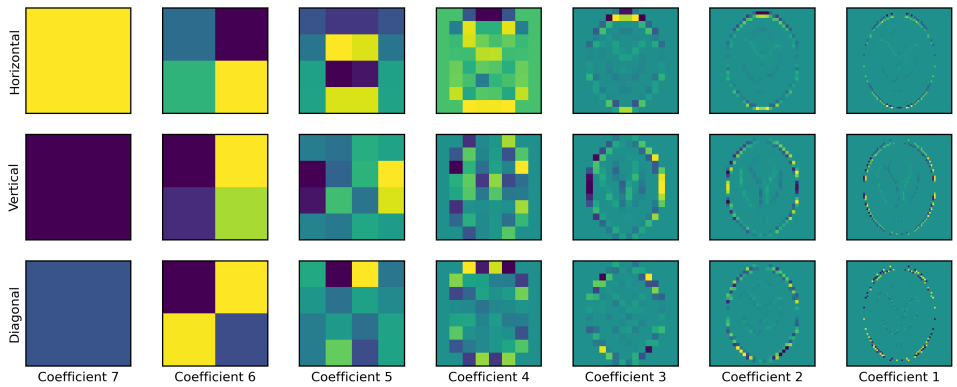

Figure 13: The 7 Haar wavelet coefficients for the Shepp-Logan phantom, with corresponding detail coefficients.

observed that indeed, horizontal, vertical and diagonal pixels have a higher intensity for each of the 3 detail coefficients.

1.4.2 Reconstructing the original image

The original phantom was successfully reconstructed by applying a Haar wavelet decomposition for 7 levels, then reversing it through an inverse decomposition (or recomposition). Wavelet decomposition is very useful in the context of image compression, as it allows for compact image storage in its coefficients which can easily be recovered through the recomposition. This can be seen in Figure 14. To the naked eye, f_{recon} looks identical to the true image. However, likely due to floating point errors there is very small MSE error here.

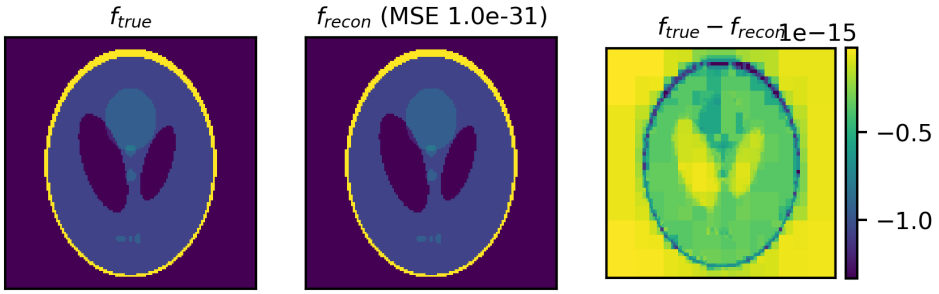

Figure 14: Reconstructing the Shepp-Logan image using inverse Haar wavelet decomposition.

We know that this is just a rounding error as calling `np.allclose()` on the function confirms that the reconstruction is within rounding error precision of the original:

```
np.allclose(f,f_rec)
>>> True
```

1.4.3 Denoising via wavelet coefficient thresholding

By selectively preserving and/or discarding certain coefficients, we aim to achieve efficient image representation while removing noisy image components. To do so, we create a function which sets coefficients with values above a specific parameter to 0. See Appendix for iterating thresholding function.

The `pywt.threshold()` function also stores the version of the image that has been thresholded and therefore we had to ensure that we were reloading the image upon every call of the function. Thresholding is particularly useful for keeping all the information from certain chosen coefficients while

thresholding others. We find that the reconstructed images from the coefficients have varying levels of detail, noise, and overall smoothness, depending on how we varied the range of thresholded coefficients and the parameter itself. This thresholding allows us to discard parts of the image deemed

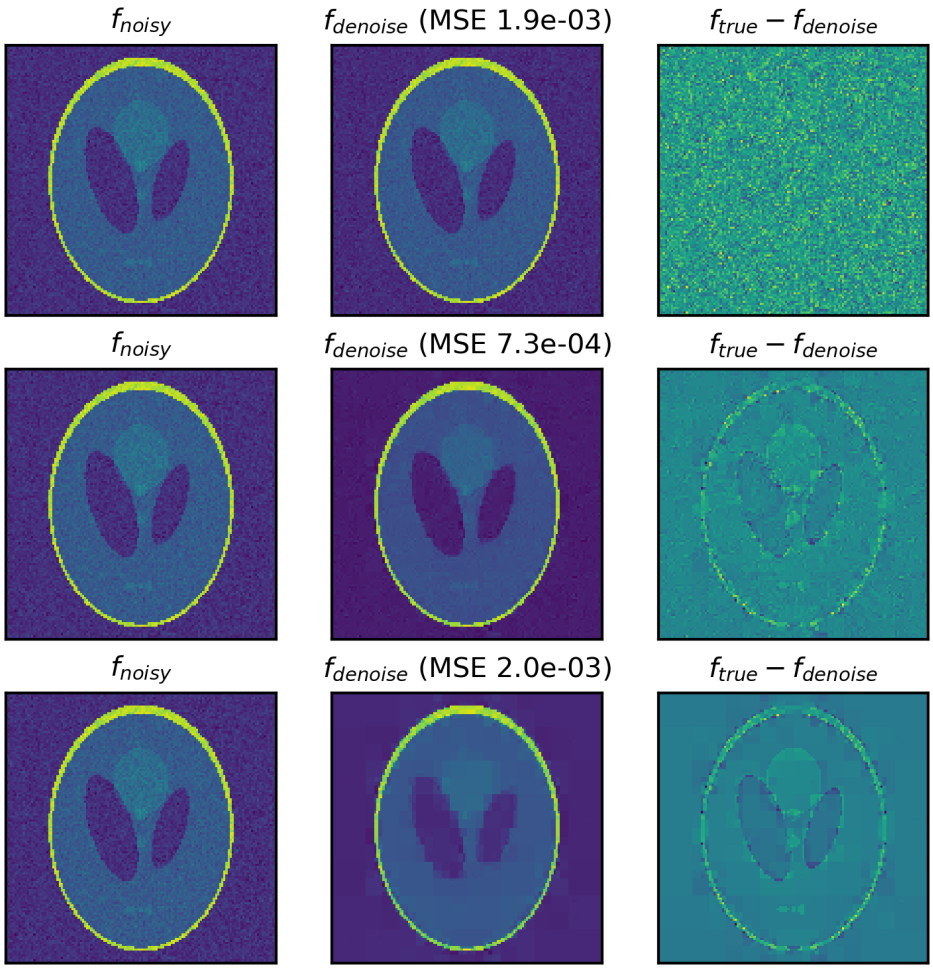

Figure 15: Reconstructing the Shepp-Logan image using inverse Haar wavelet decomposition with a thresholding function. The results for varying thresholding parameters to 0.01 (top), 0.05 (centre) and 0.1 (bottom) are shown.

unnecessary, such as the first coefficient which, because of its fine-scale granularity, also carries a lot of noise. We first varied the thresholding parameter `tVal` $= [0.01, 0.05, 0.1]$ in Figure. In the rightmost column of the figure, looking at the difference between the target and denoised images we can see that a parameter of 0.01 discards the least amount of valuable information (like edges and details) and $f_{true} - f_{denoise}$ just returns pure noise. However the reconstruction for `tVal` $= 0.01$ shows that clearly not all of the noise was adequately removed from f_{noisy} and further thresholding should have been done. We found that a value of 0.05 produced an image with high fidelity to the original, while also removing the most noise, and the lowest MSE between all examples of 7.3×10^{-4}.

We chose to vary the tRange parameter by sliding over 3 coefficients at a time, such that we evaluated the impact of thresholding coefficients [5,4,3], then [4,3,2], then [3,2,1], to see the overall effect of thresholding a range of coefficients. We found, as expected, that thresholding the higher coefficients (with finer details) led to a higher noise reduction, at the cost of losing some definition.

1.5 Iterative soft-thresholding for X-Ray tomography

This algorithm iteratively minimizes the discrepancy between the measured projections and the forward-projected estimate of the reconstructed image while reducing noise through soft thresholding

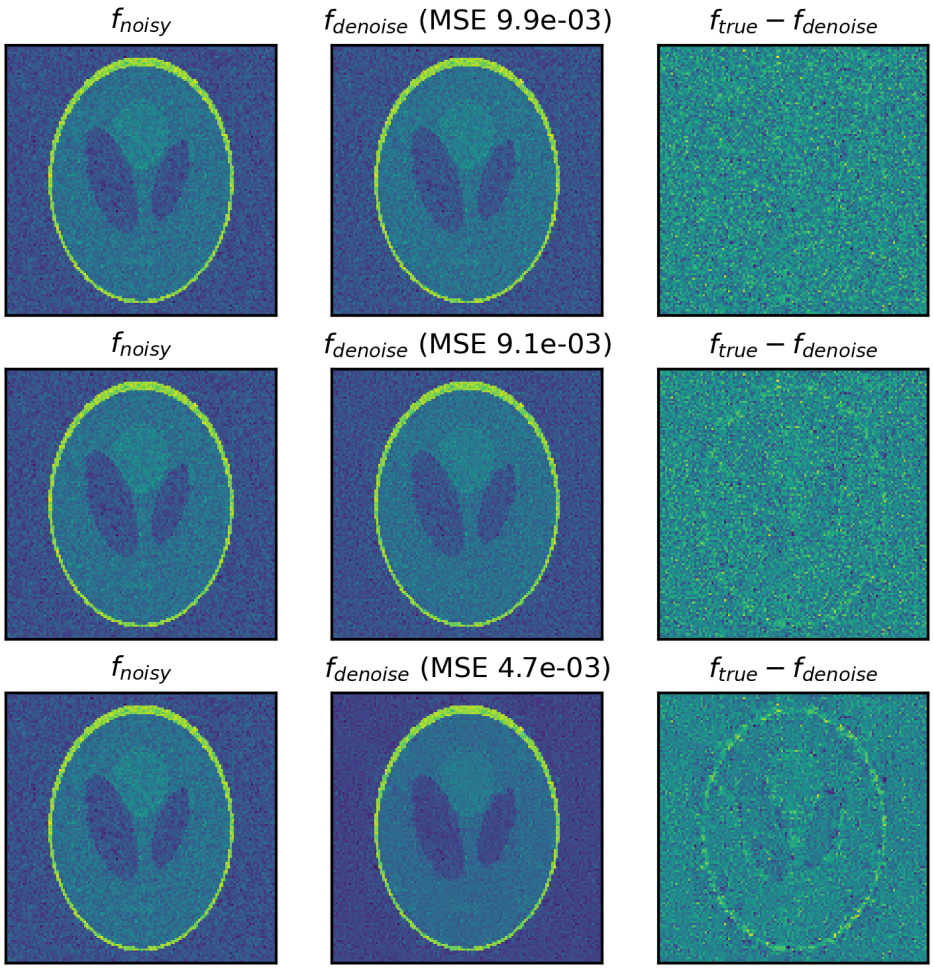

Figure 16: Reconstructing the Shepp-Logan image using inverse Haar wavelet decomposition with a thresholding function. The results for varying the range of thresholded coefficients to [2,3,4] (top), [3,4,5] (centre) and [4,5,6] (bottom) are shown.

of the wavelet coefficients.

The initial iterate f_0 was chosen such that it is an unfiltered backprojection of a noisy sinogram of f_{true} as this seemed like a good starting point to judge the algorithm's ability to both reduce noise and artifacts from the initial BP reconstruction. The best step size was found through cross validation to be $\lambda = 3.9 \times 10^{-4}$. The stopping criterion was determined based on the difference in MSE loss between two consecutive iterates, such that if f_{k+1} didn't significantly improve WRT f_k, the loop would stop. This threshold was chosen as 0.0001. One of the drawbacks of having the thresholding

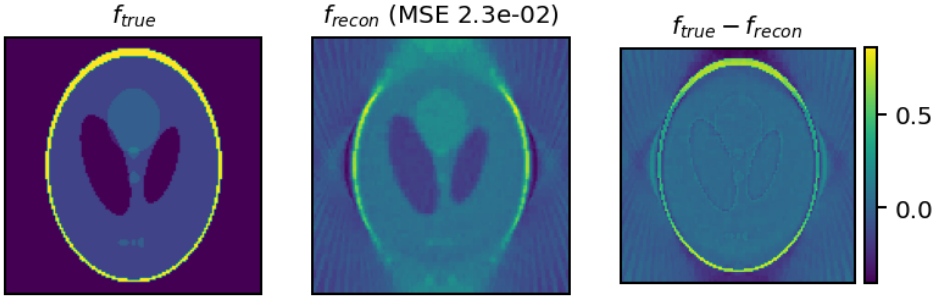

Figure 17: Result of performing iterative soft thresholding on the sparse angle case.

parameter μ predetermined based on the regularisation parameter and the step size is that we could

not control how the thresholding occured. For example for $\lambda = 10^{-5}$ and $\alpha = 0.32$ we had a very low thresholding parameter with an order of magnitude of 10^{-6} which led to most values for the thresholded coefficients getting cut off. One benefit of this model is that, even for the sparse case (45 angles in a range of 90 degrees), the model was successful in reconstructing some of the lost information from the phantom, as can be seen in Figure 18. As the number of iterations went on, we noticed that the iterative soft-thresholding reconstruction approaches that of filtered backprojection, which shows that filtered backprojection is an inexpensive and powerful tool to apply to this kind of problem.

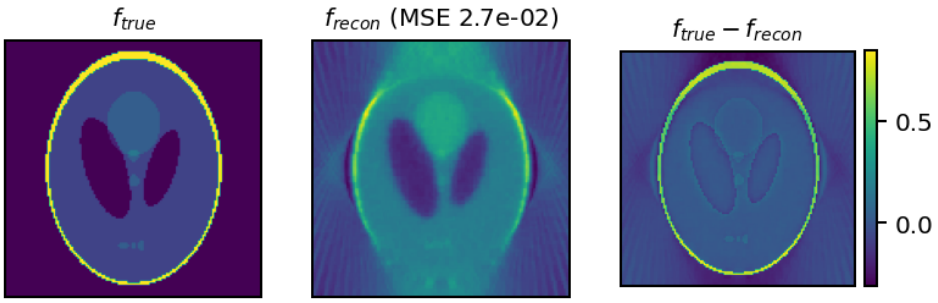

Figure 18: Result of performing iterative soft thresholding on the limited angle case.

2 Part B

2.0.1 Advanced Topic 3: Learning Based Reconstruction

In this section, we will focus on deep learning based reconstruction methods. The goal is to compare the generalization capabilities of two approaches for learned image reconstruction:

1. Initial reconstruction followed by learned post-processing
2. Learned model-based iterative reconstructions

2.1 Learned Post-Processing

A synthetic dataset was created for the training process. 64×64 images containing 5 to 20 overlapping ellipses of varying sizes were generated as the f_{true} target set. Then, noisy sinograms were created for these images which were reconstructed using filtered backprojection. The reconstructed images f_{BP} were the samples for the ResNet model. The dataset consisted of 10,000 total ellipse images, split 80/20 for training and validation. An example of a "phantom" from our dataset can be seen in Figure 19. The test image that the model is evaluated on is called f_{recon}.

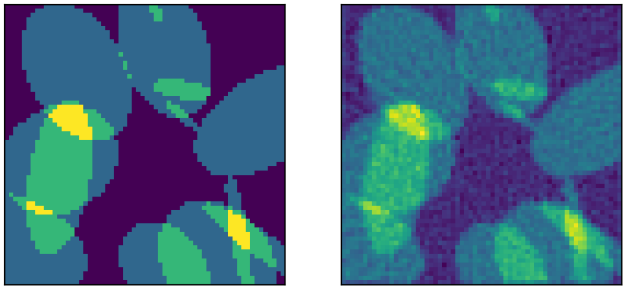

Figure 19: An example of a 64×64 pixel data pair generated for ResNet model training. The noisy reconstruction (sample) is on the right and the true image (target) is on the left.

In the first approach, initial reconstruction and learned post-processing, a ResNet G_θ was trained to map the initial reconstruction f_{BP} to the true phantom f_{true}, learning to both remove artifacts from backprojection and denoise the image. Mathematically, this can be expressed as:

$$\hat{f} = G_\theta(f_0)$$

where $\hat{f}$ (or f_{recon}) is the final reconstructed image.

To correct the initial reconstruction, we trained a ResNet G_θ. The loss function used for training the CNN is:

$$loss(\theta; g) = ||G_\theta(f_0) - f_{true}||_p$$

Training was performed using a 5-block ResNet created in PyTorch. This number of layers was chosen as it created a model with 6343 parameters, which we hoped would not lead to overfitting on our toy dataset. Each ResNet block consisted of:

1. A convolution layer followed by a BatchNorm and ReLU
2. A second convolution followed by a BatchNorm and ReLU
3. A linear projection to ensure that the original input has the same size as the output of the convolutions
4. A skip connection, which adds the output of the linear projection to the convolution output

5. A final ReLU layer

These blocks were connected sequentially in the final model, followed by three sequences of ConvTranspose2D, BatchNorm and ReLU activations to finally upsample the image back to its original shape. The learning rate was varied between 0.05, 0.01, 0.001 and 0.005 until it was kept fixed at 0.001, and the batch size was 32.

The loss curves for training and validation are displayed in Figure 20. We can observe that after 20 iterations or so, the decrease in validation loss became markedly lower than the first epochs. The final validation loss value was 3×10^{-3}. We find that the model-free approach has a significant effect

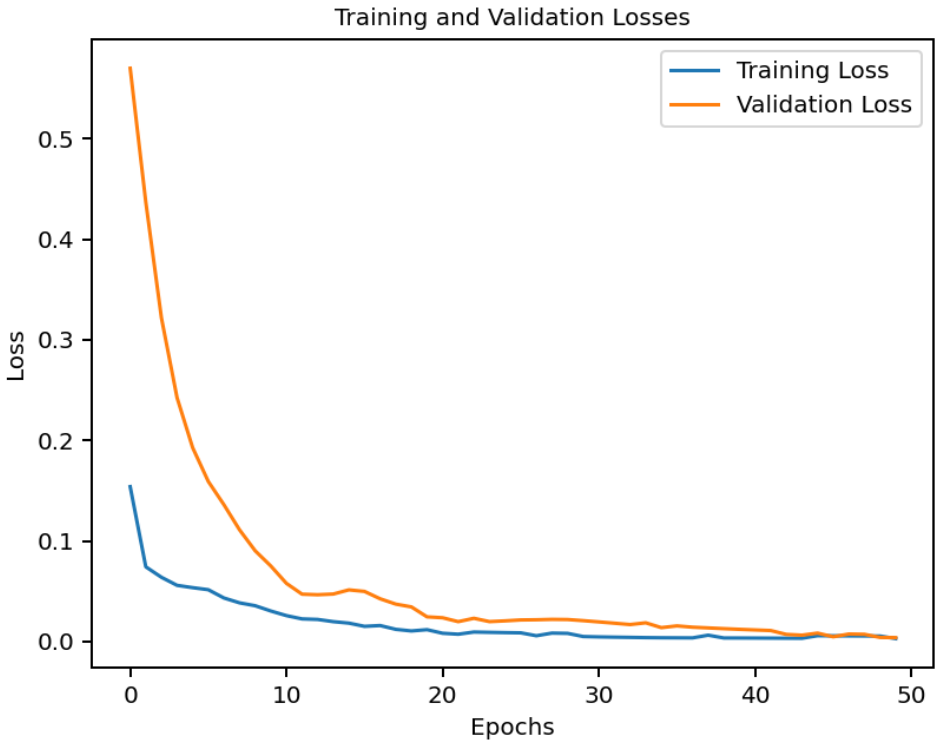

Figure 20: The MSEloss curves obtained from training the model-free ResNet on the ellipse dataset.

on reducing noise and artifacts from filtered backprojections, as seen in Figure 21. The MSE is at least one order of magnitude lower than those for most other approaches. This is especially significant considering that for Part A, we set the θ parameter to 10,000 for most of the experiment, whereas the ResNet model has been trained and tested on images with $\theta = [100, 10, 000]$. Thus, this approach to denoising, while simple and requiring no real-world sinogram data for training, is very effective.

The model wasn't able to perfectly reconstruct the bright border around the phantom, nor was it able to reliably denoise the navy blue halo that is created from the FBP process around the phantom. However, while the latter point might seem like a negative, that halo roughly has the same colour as the inner shading of the phantom. Thus, we deem it to be preferable for the outer halo to be insufficiently denoised when compared to removing the shaded region within the phantom by considering it "noise".

2.2 Learned model-based reconstructions

Instead of relying on a fixed initial reconstruction, the neural network was trained to compute the gradient of the data fit, $A^T(Af - g)$, before every residual block. This was then passed into the next residual block alongside the output of the previous residual block. By minimizing a loss function that measures the discrepancy between the reconstructed image and the true phantom, this can be formulated as an optimization problem:

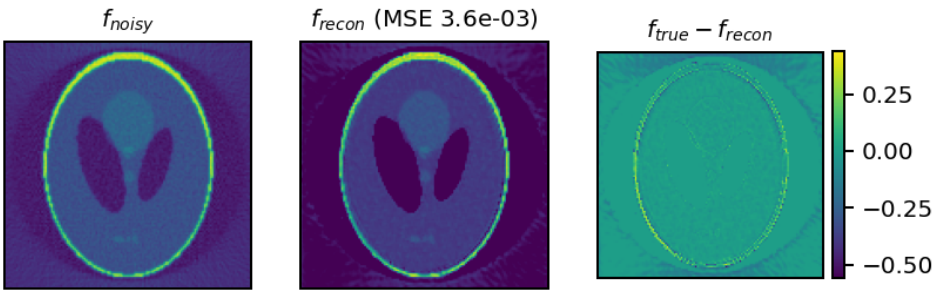

Figure 21: Result of training a model-free ResNet on the ellipses dataset for 100 epochs with a learning rate of 0.001.

$$\hat{f} = \arg\min_{f} loss(\theta; g) + A^T(Af - g)$$

Algorithm 1 Learned Model-Based Iterative Reconstruction

Input: $\{(f_{\text{BP}_1}, f_{\text{true}_1}), \ldots, (f_{\text{BP}_m}, f_{\text{true}_m})\} \in (\mathbb{R}^n \times [0,1])^m$
for epoch = 1 to epoch = maxEpochs **do**
  for $k = 1, \ldots, K$ residual blocks **do**
    Compute gradient of data fit $A^T(Af_{BP_{k-1}} - g)$
    Concatenate gradient with current reconstruction: $G_k = [f_{BP_{k-1}}, A^T(Af_{k-1} - g)]$
    Pass concatenated input G_k through residual block k
  end for
  Calculate MSE-loss between f_{recon_K} and f_{true} for backpropagation
  Update parameters p
end for
Return f_{recon}

where A is the forward operator (explicit Radon transform), and g is the noisy measurement data.

For this approach, a slightly different ResNet architecture was used. A gradient block was created to compute the data fit $A^T(Af_{BP_{k-1}} - g)$, which was sequentially followed by the ResNet block defined above. This block was also simplified to go from 2 inputs to 1 output as opposed to including the ConvTranspose2D, for simplicity. The same batch size of 32 and learning rate of 0.001 were used here, and the loss curves can be seen in Figure

The learned model-based approach was expected to generalize better to unseen data due to its ability to incorporate prior knowledge. However, as can be seen in Figure 23, there was more blurring and less definition on the features of the phantom, which was unexpected. The general MSE value was still lower than that for the non-deep learning based reconstructions, however it was not nearly as effective as its model-free counterpart. This could be due to the gradient having a very strong penalising effect on the weights, or making it difficult for the model to adequately make predictions on the unseen data. Another difficulty in evaluating this model was that the memory use was too high, likely due to the computation of multiple gradients at a time, leading the kernel to crash often.

Additionally, the bump in the loss curve in Figure 22 might indicate overtraining or perhaps an error in the implementation of the data fit term in the model.

Given that the MSE for the model-free approach was of the order of 10^{-3}, it seems like a good denoiser, however as has been shown throughout this investigation, a low MSE value is not necessarily an indicator of adequate denoising or reconstruction. For example, in Figure 5, when constructing backprojections for varying noise levels, even heavily corrupted images had MSE not far from the almost noiseless reconstructions. Additionally, though the reconstruction with $\theta = 10000$ has a very

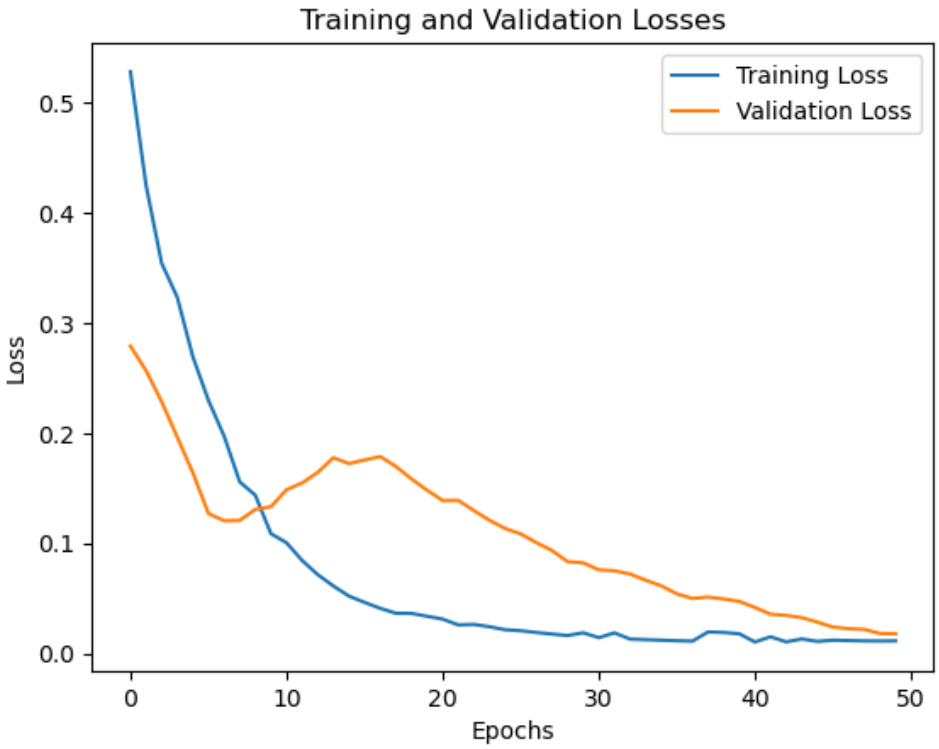

Figure 22: The MSEloss curves obtained from training the model-based ResNet on the ellipse dataset.

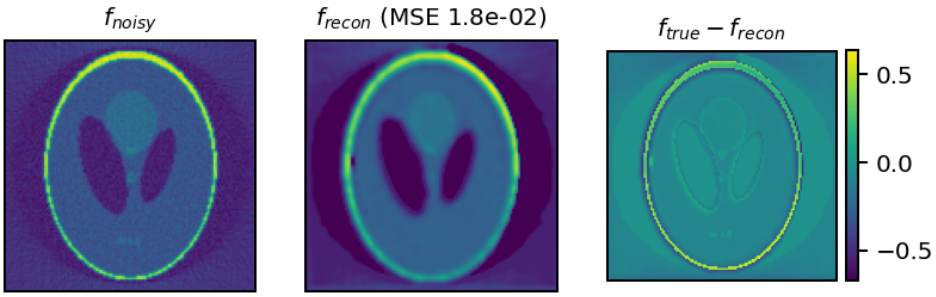

Figure 23: Result of training a model-based ResNet on the ellipses dataset for 100 epochs with a learning rate of 0.001.

low MSE value, one could argue that the visual quality of the image reconstructed by deep learning (Figure 21) is overall more faithful to the original than the filtered backprojection.

3 Appendix

```
def explicit_radon(n_angles,max_angle,n_samples,v,h):
'''
Takes Radon Transform of one pixel at a time to produce a Radon Transform matrix with size
    (v*h,n_angles*n_samples)
'''
    angles = np.linspace(0,max_angle,n_angles,endpoint=False)
    A_exp = []
    vol_geom = astra.create_vol_geom(v,h)
    proj_geom = astra.create_proj_geom('parallel',1.,n_samples,angles)
    projector_id = astra.create_projector('strip', proj_geom, vol_geom)

    row_idx = 0
```

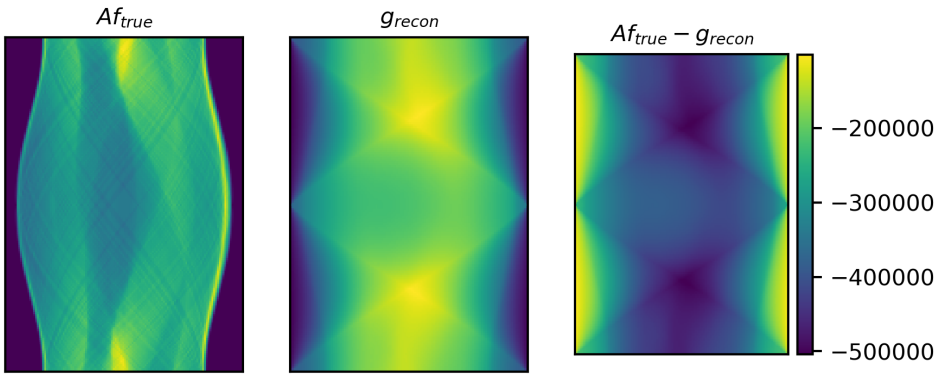

Figure 24: Sinogram obtains from unfiltered back-projection reconstruction.

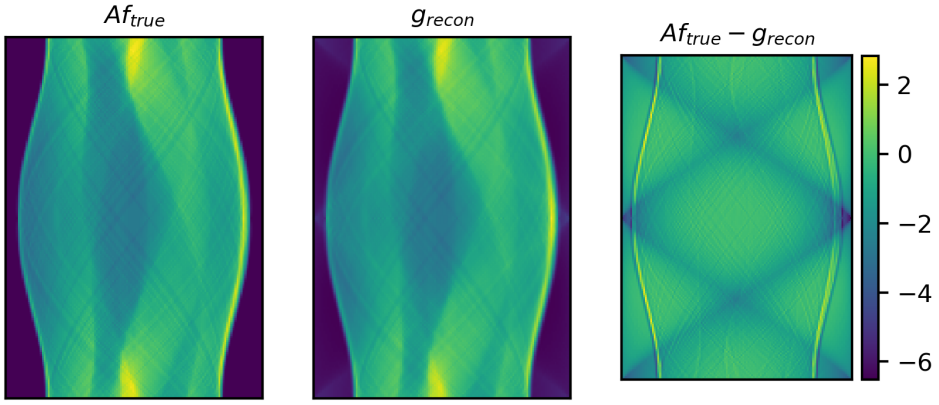

Figure 25: Sinogram obtains from unfiltered back-projection reconstruction.

```
    for col in range(v): # For each column
        for row in range(h): # For each row in the current column
            dummy_img = np.zeros((v,h))
            dummy_img[row][col] = 1
            sinogram_id, sinogram = astra.create_sino(dummy_img.T, projector_id)
            A_exp.append(sinogram.reshape(-1,1,order='F'))
        row_idx +=1
    A_exp = np.hstack(A_exp)
    return A_exp
```

```
def AT_y(vol_geom,projector_id,sinogram_id):
    """Blurs an image using a gaussian filter"""
    f_rec = BP_recon(vol_geom,projector_id,sinogram_id,'BP')
    AT_y = f_rec.flatten()

    return AT_y
```

```
def thresholdFunction(coeffs,tRange,tVal):
    for i in tRange:
        coeffs[i+1] = [tuple(pywt.threshold(comp, tVal) for comp in detail) for detail in
            coeffs[i+1]]
    return coeffs
```

```
def iterative_thres(f,n_angles,max_angle,n_samples,vol_geom,v,h,alpha,threshold,
                    step_size,tRange,maxIter=100):
```

```
vol_geom,proj_geom,projector_id,g_id,g =
    create_sinogram(f,n_angles,max_angle,n_samples,v,h)
g = astra.functions.add_noise_to_sino(g,10000)
g_id = astra.data2d.create('-sino',proj_geom,g)
f_k = normalise(BP_recon(vol_geom,projector_id,g_id,'BP')) ## create f0
for _ in range(maxIter):
    vol_geom,_,projector_id,sinogram_id,Af_k =
        create_sinogram(f_k,n_angles,max_angle,n_samples,v,h)
    # create sinogram id for Afk_g so that it can be passed into ATy
    # (because ATy takes sinogram ID, not sinogram as input)
    Afk_g_id = astra.data2d.create('-sino',proj_geom, Af_k - g)
    ATAfk_g = AT_y(vol_geom,projector_id,Afk_g_id).reshape(v,h)
    f_kplus1 = soft_threshold(f_k - step_size*ATAfk_g,7,alpha,step_size,tRange)

    if MSE(f_k,f) - MSE(f_kplus1,f) <= threshold:
        break
    else:
        f_k = f_kplus1
return f_k,MSE(f_k,f)
```