Graduation Project - Total Variational Image Denoising with U-Net
Thanh Trung Vu - 230849442

1 Introduction

In this dissertation we will describe the theory and application of the total variational (TV) method to image denoising. We will also describe the implementation of the method and an improvement using a U-Net architecture to find the regularisation parameters more efficiently.

1.1 Total Variational Denoising

The total variational (TV) method is a method for image denoising that is based on the principle of minimising the total variation of the image. The total variation of an image is a measure of the amount of variation in the intensity of the image. The TV method works by finding the image that has the smallest total variation and is closest to the noisy image in terms of the mean squared error.

The TV method is based on the idea that images that are smooth have a small total variation, while images that are noisy have a large total variation. By minimising the total variation of the image, the TV method is able to remove the noise from the image and produce a smoother image.

Denoising problems can be described as

$$\mathbf{z} = \mathbf{x}_{\text{true}} + \mathbf{e} \tag{1}$$

where $\mathbf{x}_{\text{true}} \in \mathbb{R}^{m \times n}$ is the object to be imaged, $\mathbf{e} \in \mathbb{R}^{m \times n}$ denotes some random noise component, and $\mathbf{z} \in \mathbb{R}^{m \times n}$ represents the measured data. The goal is to recover $\mathbf{x}_{\text{true}}$ or at least a good enough reconstruction given $\mathbf{z}$.

A prominent approach is to formulate the reconstruction as a minimisation problem:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} d(\mathbf{x}, \mathbf{z}) + \mathcal{R}(\mathbf{x}) \tag{2}$$

where $d(\cdot, \cdot)$ is a data fidelity term and $\mathcal{R}(\cdot)$ is a regularisation term.

Here we will look at Gaussian noise. The data fidelity term appropriate for Gaussian noise is the square of the ℓ_2 norm:

$$d(\mathbf{x}, \mathbf{z}) = \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 \tag{3}$$

The regularisation term $\mathcal{R}(\cdot)$ is the total variation of the image, which is a measure of the amount of variation in the intensity of the image. The total variation of an image can be calculated using the following formula:

$$\mathcal{R}(\mathbf{x}) = \lambda \|\nabla \mathbf{x}\|_1 \tag{4}$$

where $\nabla \mathbf{x}$ is the gradient of the image $\mathbf{x}$ and $\lambda \in \mathbb{R}^+$ is a regularisation parameter that controls the trade-off between the total variation of the image and the mean squared error between the denoised image $\mathbf{x}$ and the noisy image $\mathbf{z}$.

∇ denotes a finite-differences operator. Finite difference operators include, forward difference operator, backward difference operator, shift operator, central difference operator and mean operator. Let us focus on the forward difference operator. In particular, in the discrete case, given a matrix $\mathbf{x}$, $\nabla\mathbf{x}$ is two matrices, one for the vertical gradient $\nabla_x\mathbf{x}$ and one for the horizontal gradient $\nabla_y\mathbf{x}$.

$$\nabla\mathbf{x} = \begin{bmatrix} \nabla_x\mathbf{x} \\ \nabla_y\mathbf{x} \end{bmatrix} \tag{5}$$

For the forward gradient, the vertical gradient is calculated as:

$$\begin{aligned}
\nabla_x\mathbf{x} &= \begin{bmatrix}
\mathbf{x}_{2,1}-\mathbf{x}_{1,1} & \mathbf{x}_{2,2}-\mathbf{x}_{1,2} & \dots & \mathbf{x}_{2,n-1}-\mathbf{x}_{1,n-1} & \mathbf{x}_{2,n}-\mathbf{x}_{1,n} \\
\mathbf{x}_{3,1}-\mathbf{x}_{2,1} & \mathbf{x}_{3,2}-\mathbf{x}_{2,2} & \dots & \mathbf{x}_{3,n-1}-\mathbf{x}_{2,n-1} & \mathbf{x}_{3,n}-\mathbf{x}_{2,n} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{x}_{m,1}-\mathbf{x}_{m-1,1} & \mathbf{x}_{m,2}-\mathbf{x}_{m-1,2} & \dots & \mathbf{x}_{m,n-1}-\mathbf{x}_{m-1,n-1} & \mathbf{x}_{m,n}-\mathbf{x}_{m-1,n} \\
\mathbf{x}_{1,1}-\mathbf{x}_{m,1} & \mathbf{x}_{1,2}-\mathbf{x}_{m,2} & \dots & \mathbf{x}_{1,n-1}-\mathbf{x}_{m,n-1} & \mathbf{x}_{1,n}-\mathbf{x}_{m,n}
\end{bmatrix} \\
&= \begin{bmatrix}
-1 & 1 & \dots & 0 & 0 \\
0 & -1 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & -1 & 1 \\
1 & 0 & \dots & 0 & -1
\end{bmatrix} \mathbf{x}
\end{aligned} \tag{6}$$

and the horizontal gradient is calculated as:

$$\begin{aligned}
\nabla_y\mathbf{x} &= \begin{bmatrix}
\mathbf{x}_{1,2}-\mathbf{x}_{1,1} & \mathbf{x}_{1,3}-\mathbf{x}_{1,2} & \dots & \mathbf{x}_{1,n}-\mathbf{x}_{1,n-1} & \mathbf{x}_{1,1}-\mathbf{x}_{1,n} \\
\mathbf{x}_{2,2}-\mathbf{x}_{2,1} & \mathbf{x}_{2,3}-\mathbf{x}_{2,2} & \dots & \mathbf{x}_{2,n}-\mathbf{x}_{2,n-1} & \mathbf{x}_{2,1}-\mathbf{x}_{2,n} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{x}_{m-1,2}-\mathbf{x}_{m-1,1} & \mathbf{x}_{m-1,3}-\mathbf{x}_{m-1,2} & \dots & \mathbf{x}_{m-1,n}-\mathbf{x}_{m-1,n-1} & \mathbf{x}_{m-1,1}-\mathbf{x}_{m-1,n} \\
\mathbf{x}_{m,2}-\mathbf{x}_{m,1} & \mathbf{x}_{m,3}-\mathbf{x}_{m,2} & \dots & \mathbf{x}_{m,n}-\mathbf{x}_{m,n-1} & \mathbf{x}_{m,1}-\mathbf{x}_{m,n}
\end{bmatrix} \\
&= \mathbf{x} \begin{bmatrix}
-1 & 0 & \dots & 0 & 1 \\
1 & -1 & \dots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & -1 & 0 \\
0 & 0 & \dots & 1 & -1
\end{bmatrix}
\end{aligned} \tag{7}$$

We calculate the gradient in the horizontal and vertical directions separately, and then combine them to get the total variation of the image.

$$\|\nabla\mathbf{x}\|_1 = \|\nabla_x\mathbf{x}\|_1 + \|\nabla_y\mathbf{x}\|_1 \tag{8}$$

For the forward gradient, the norm-1 of the vertical gradient is calculated as:

$$\|\nabla_x\mathbf{x}\|_1 = \left(\sum_{i=1}^{m-1} \sum_{j=1}^{n} |\mathbf{x}_{i+1,j} - \mathbf{x}_{i,j}| \right) + \sum_{j=1}^{n} |\mathbf{x}_{1,j} - \mathbf{x}_{m,j}| \tag{9}$$

and norm-1 of the horizontal gradient is calculated as:

$$\|\nabla_y \mathbf{x}\|_1 = \left(\sum_{i=1}^{m} \sum_{j=1}^{n-1} |\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j}| \right) + \sum_{i=1}^{m} |\mathbf{x}_{i,1} - \mathbf{x}_{i,n}| \tag{10}$$

where $\mathbf{x}_{i,j}$ is the intensity value of pixel (i, j) in the image $\mathbf{x}$.

Note that, we can move λ inside the norm, so the regularisation term can be written as:

$$\mathcal{R}(\mathbf{x}) = \lambda\|\nabla \mathbf{x}\|_1 = \|\lambda \nabla \mathbf{x}\|_1 = \|\lambda \nabla_x \mathbf{x}\|_1 + \|\lambda \nabla_y \mathbf{x}\|_1 \tag{11}$$

The regularisation parameter does not need to be a scalar. Since regularisation is multiplied by the gradient, higher λ values will penalize areas with many changes (typically areas with more details), while lower λ values prioritize detail preservation over noise reduction. Using a single λ value penalises all regions of the image equally, which is suboptimal. Instead, it would be better to assign higher penalties to smooth regions and lower penalties to detailed regions to preserve image details. We can swap the scalar λ with a regularisation parameter map $\mathbf{\Lambda}$,

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}^{(x)} \\ \mathbf{\Lambda}^{(y)} \end{bmatrix} \tag{12}$$

where $\mathbf{\Lambda}^{(x)}, \mathbf{\Lambda}^{(y)} \in \mathbb{R}^{m \times n}$ are matrices that contain the regularisation parameters for the vertical and horizontal gradients respectively. Here we will assume $\mathbf{\Lambda}^{(x)} = \mathbf{\Lambda}^{(y)}$ and denote them as $\mathbf{\Lambda}^{(xy)}$, i.e.

$$\mathbf{\Lambda} = \begin{bmatrix} \mathbf{\Lambda}^{(xy)} \\ \mathbf{\Lambda}^{(xy)} \end{bmatrix} \tag{13}$$

This gives us a new regularisation term that can be written as

$$\mathcal{R}(\mathbf{x}) = \|\mathbf{\Lambda} \circ \nabla \mathbf{x}\|_1 = \|\mathbf{\Lambda}^{(xy)} \circ \nabla_x \mathbf{x}\|_1 + \|\mathbf{\Lambda}^{(xy)} \circ \nabla_y \mathbf{x}\|_1 \tag{14}$$

where $\circ$ denotes the Hadamard or element-wise product.

Putting it all together, the solution to the total variational denoising problem can be defined using the following formula:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + \lambda\|\nabla \mathbf{x}\|_1 \tag{15}$$

if we use a scalar λ regularisation parameter, or

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{x} - \mathbf{z}\|_2^2 + \|\mathbf{\Lambda} \circ \nabla \mathbf{x}\|_1 \tag{16}$$

if we use a regularisation parameter map $\mathbf{\Lambda}$ of regularisation parameters.

In this dissertation, we will use U-Net to help us find the regularisation parameter map $\mathbf{\Lambda}$ given any (noisy) image. We will compare the results to the best-case scenario where we use a scalar regularisation parameter λ.

1.2 U-Net

The U-Net is a type of neural network that is commonly used for image segmentation [5]. The U-Net is an encoder-decoder network that is designed to take an image as input and produce a segmentation mask as output. The U-Net is made up of a series of convolutional layers that downsample the image and a series of transposed convolutional layers that upsample the image. The U-Net is able to learn to segment images by training on a large dataset of images and their corresponding segmentation masks.

In our case, we will optimise a U-Net model to find the regularisation parameter map Λ that produces the best denoised image for any particular noisy image.

1.3 Related Work

1.4 Contribution

- Adapt the existing code of the dynamic image denoising project in [2] to static image denoising. Train and evaluate the model using a new dataset - Chest X-ray dataset.

- Present the theory of the total variational denoising method and the experimental set-up in detail so that a fellow student can understand the method and implement it themselves.

2 Method and Implementation

2.1 Primal Dual Hybrid Gradient (PDHG)

To solve equation (14) we can use an iterative method called the Primal Dual Hybrid Gradient (PDHG).

The algorithm is

Algorithm 1 PDHG algorithm for image denoising with fixed regularization parameter-map $\mathbf{\Lambda}$ (adapted from ...)

Input: $L = \|[\mathbf{I}, \nabla]^{\mathrm{T}}\|$, $\tau = 1/L$, $\sigma = 1/L$, $\theta = 1$, noisy image $\mathbf{x}_0$
Output: reconstructed image $\hat{\mathbf{x}}$
$\bar{\mathbf{x}}_0 = \mathbf{x}_0$
$\mathbf{p}_0 = \mathbf{0}$
$\mathbf{q}_0 = \mathbf{0}$
for $k < T$ **do**
  $\mathbf{p}_{k+1} = \left(\mathbf{p}_k + \sigma(\bar{\mathbf{x}}_k - \mathbf{x}_0)\right)/(1+\sigma)$
  $\mathbf{q}_{k+1} = \mathrm{clip}_{\mathbf{\Lambda}}\left(\mathbf{q}_k + \sigma\nabla\bar{\mathbf{x}}_k\right)$
  $\mathbf{x}_{k+1} = \mathbf{x}_k - \tau\mathbf{p}_{k+1} - \tau\nabla^{\mathrm{T}}\mathbf{q}_{k+1}$
  $\bar{\mathbf{x}}_{k+1} = \mathbf{x}_{k+1} + \theta(\mathbf{x}_{k+1} - \mathbf{x}_k)$
end for
$\hat{\mathbf{x}} = \mathbf{x}_T$

where $\mathbf{I}$ is the identity matrix, $\mathrm{clip}_{\mathbf{\Lambda}}$ is a function that clips the values of $\mathbf{q}_{k+1}$ to the range $[\mathbf{0}, \mathbf{\Lambda}]$.

The choice of L is taken from [6]. The number of iterations T is a hyperparameter that we can choose for the training process.

(TODO: Understand what L is. Do we need to include $\theta = 1$? Can we write $\bar{\mathbf{x}}_{k+1} = 2\mathbf{x}_{k+1} - \mathbf{x}_k$? In the code, initially $\mathbf{p}_0 = \mathbf{x}_0$?)

The challenge lies in finding the regularisation parameter map $\mathbf{\Lambda}$. Narrowing down the search space is challenging, as we potentially have d^{mn} different maps to consider, where d is the number of λ values (typically around 100, ranging from 0.01 to 1) and n is the image dimension. This is computationally infeasible. The U-Net can be used to find the regularisation parameter map $\mathbf{\Lambda}$ more efficiently.

2.2 Full Architecture

For this project, our model comprises two primary components:

1. A U-Net, denoted as NET_Θ, which is responsible for learning the regularisation parameters $\mathbf{\Lambda}_\Theta$ from an input image $\mathbf{x}_{\text{noisy}}$
2. A Primal Dual Hybrid Gradient (PDHG) algorithm solver

We refer to the set of trainable parameters in the U-Net as Θ, and the output of the U-Net as $\mathbf{\Lambda}_\Theta$.

The general architecture is depicted in shown in Figure 1,

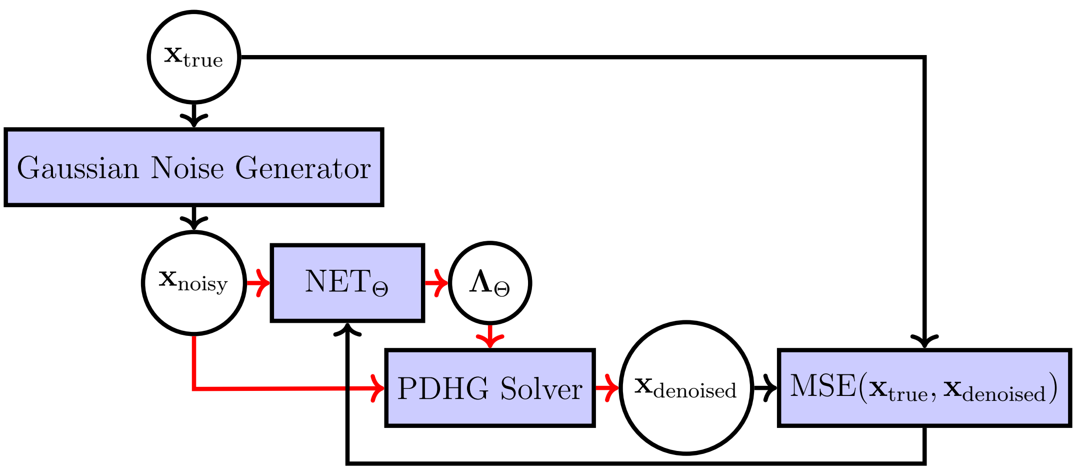


Figure 1: The general architecture

where

- $\mathbf{x}_{\text{true}}$ represents the clean image, serving as the ground truth.
- $\mathbf{x}_{\text{noisy}}$ denotes the noisy image inputted to NET_Θ.
- $\mathbf{\Lambda}_\Theta$ is the regularisation parameters map which is the final output from the U-Net.

We can treat the PDHG solver as the final hidden layer in our network. The result of the loss function $\text{MSE}(\mathbf{x}_{\text{true}}, \mathbf{x}_{\text{denoised}})$ is then used for backpropagation to train the parameters Θ.

As previously mentioned, the specific architecture for NET_Θ is the U-Net, a special type of convolutional neural network architecture.

2.3 Convolutional Neural Network (CNN)

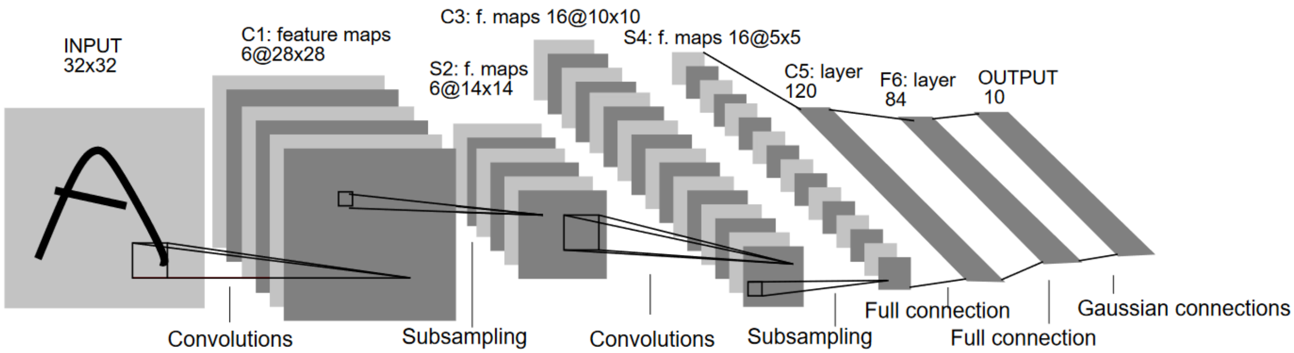


Figure 2: Architecture of LeNet-5, one of the earliest Convolutional Neural Networks [3]

Convolutional Neural Networks (CNNs) are a specific type of artificial neural network particularly well-suited for analyzing image data and other grid-like patterns. Here we assume a 2D CNN architecture, where each layer is represented as one or more 2-dimensional matrices, rather than a vector as in a normal "vanilla" network.

Inspiration and Function

CNNs are inspired by the visual cortex in the human brain. The visual cortex processes visual information, extracting features one-by-one, from simple edges and shapes to complex objects. CNNs try to mimic this through convolutional layers and pooling layers.

Convolutional Layers

Similar to a normal "vanilla" neural network layer, each element in a convolutional layer's output comes from a linear combination followed by an activation function. The difference is the use of filters (also called kernels) instead of a single weight matrix. Each filter slides across the input, computing the dot product between its weights and the corresponding elements, then passes the output through an activation function. This is a convolution operation. For each input matrix, one filter produces one output matrix. We can have multiple filters in one layer. Each output matrix is called a feature map, since one filter can be thought of as a feature-extractor that is designed to highlight a specific feature. The number of feature maps is also called the number of channels. Implementation-wise, using filters instead of weight matrices reduces the ratio between the number of weights and the number of output values, hence cutting down on the number of trainable parameters for a image task.

Pooling Layers

An additional type of layer is a pooling layer. Usually we use a max-pooling layer which outputs the maximum instead. This project also utilises max-pooling. The goal is to keep only the most signification values.

These convolutional and pooling layers will form the building block for the U-Net architecture.

2.4 U-Net

A U-Net, as introduced by Ronneberger et al. [5], extends the conventional CNN architecture by incorporating skip connections, similar to the addition connection in a residual network. Each skip connection basically concatenates the outputs of two different convolution layers into a bigger output.

The U-Net architecture is divided into two principal components: an Encoder and a Decoder.

Encoder

The Encoder functions similarly to a standard CNN, utilizing a combination of convolutional layers and max-pooling layers organized into distinct "blocks." In this project, each Encoder block comprises a pair of convolutional layers followed by a max-pooling layer, designed to successively double the number of channels (or feature maps) while reducing the feature map size due to the pooling.

Decoder

The Decoder is also a CNN with a series of blocks. Opposite to an encoding block which doubles the number of channels, each successive decoding block cuts the number of channels (feature maps) in half while increasing the size of the output feature map (hence the "unrolling"). Instead of a max-pooling layer, each decoding block ends with a so-called up-convolutional layer and a skip connection. The up-convolutional layer is just a normal convolutional layer whose output is concatenated with the output of another convolutional layer in an Encoder block.

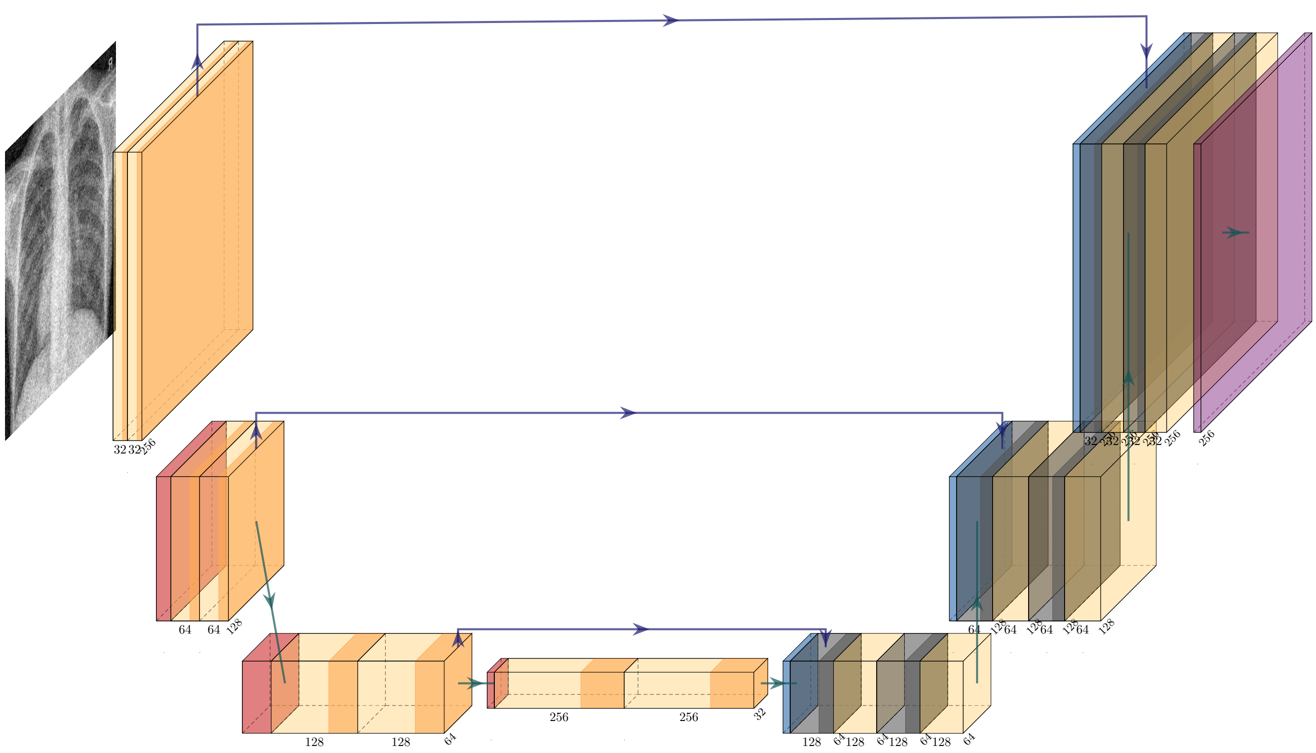


Figure 3: Our NET_Θ Architecture

3 Experimental Set-Up

We used the Chest X-Ray Images (Pneumonia) dataset which was downloaded from [1]. This is a dataset for binary classification of normal and pneumonia chest X-ray images. The dataset consists of 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). All images are black-and-white and are in the JPEG format. The X-rays images have various resolutions.

At the time of download, the original dataset had already been split into a training set, a validation set, and a test set. We picked 200 images for the training dataset for training and 100 images from the test set for testing. The validation set consists of only 8 images which were all used for validation. We used only images which were classified as normal.

For consitent input, we cropped the images to squares and then resized them to 256×256. For each image we generated a random value of σ uniformly in the range from 0.1 to 0.5, and noised the image with Gaussian noise with mean 0 and the corresponding standard deviation σ.

For learning the regularisation parameter map $\boldsymbol{\Lambda}_\Theta$, we used a U-Net structure with 1 initial double convolutional block, followed by 3 encoding blocks, 3 decoding blocks, and 1 final fully connected layer. The number of initial filters, or output channels of the first convolution layer, is set to 32. As commonly done, each encoding/decoding block contains a(n) downsampling/upsampling step, followed by 2 (fully) convolutional layers, in which the first convolutional layer doubles/halves the number channels which the second maintains the number of channels. In other words, the number of output channels of each subsequent encoding block is doubled, while the number of output channels of each subsequent decoding block is halved. The U-Net has 1 input channel and 2 final output channels. This leads to a 1-32-64-128-256-128-64-32-2 structure.

All convolutional layers have a kernel size of 3×3 and a stride of 1. Each side of a feature map has zero-padding of size 1 to maintain the size of the feature map after the convolution. Keeping the number of size of the feature map constant after each convolution has the advantage of making the implementation of the skip connections simpler by not having to crop the output of the encoding blocks to match the size of the input of the decoding blocks.

Each encoding block begins with a max pooling layer with 2×2 kernels and a step size of 2. Prior to the max pooling, a zero-padding of size 1 is added in order to exactly halve the length of each side of the output. On the other hand, all upsampling steps are done with linear interpolation with a scale factor of 2 which doubles the length of each side of the feature map. In the end, the total number of trainable parameters of the set Θ is 1,796,034.

For the activation function, we used the Leaky ReLU with a negative slope of 0.01. For the primal dual solver, we set the up-bound parameter to 0 and T_{train} to 256. During testing we also set T_{test} to 256.

For reproducibility, we manually set a seed value to 42 for the random number generator at the beginning of the training process. We use the Adam optimiser with an initial learning rate of 1e-4, a batch size of 1, and the Mean Square Error (MSE) loss function. We trained for a total of 500 epochs. The total training time was 30 hours, and the GPU memory footprint was around 3 GB.

It is worth noting that, for the implementation of the U-Net, we used the 3D implementations of the convolutional layers as well as the downsampling and upsampling steps from the PyTorch library [4]. The main reasons we went with 3D instead of the 2D implementations are convenience and generality. The 3D U-Net implementation is adapted from the dynamic image denoising application in [2], and can therefore be extended to 3D dynamic imaging applications in the future.

4 Results

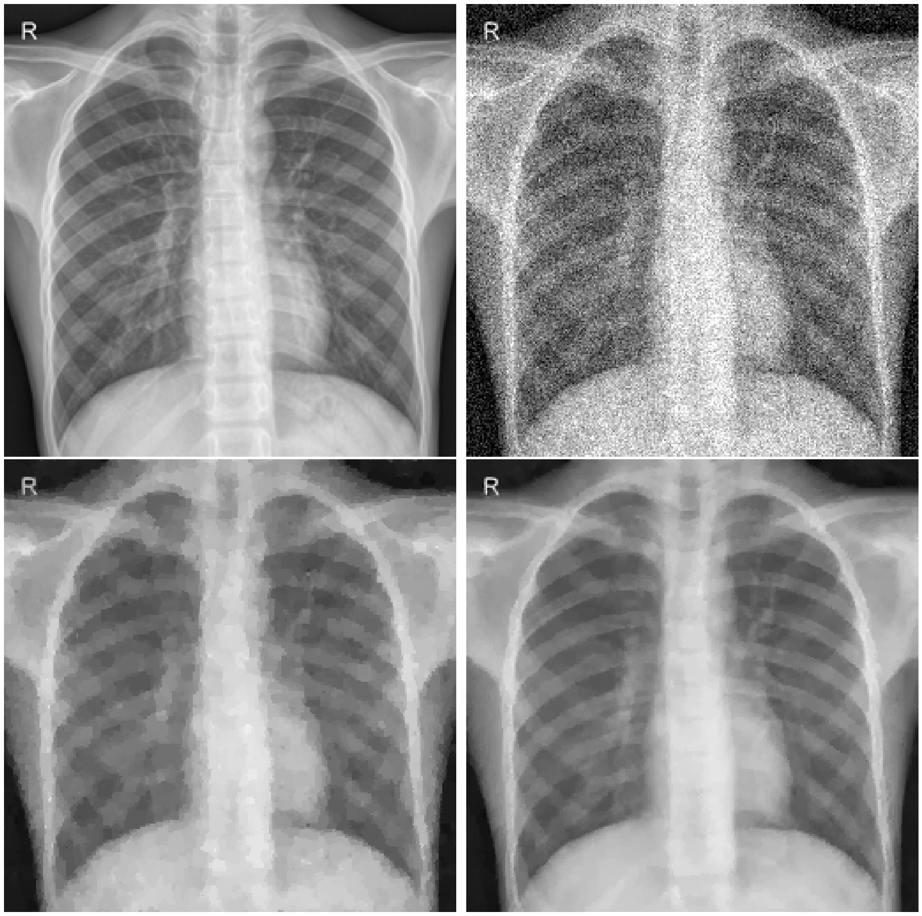
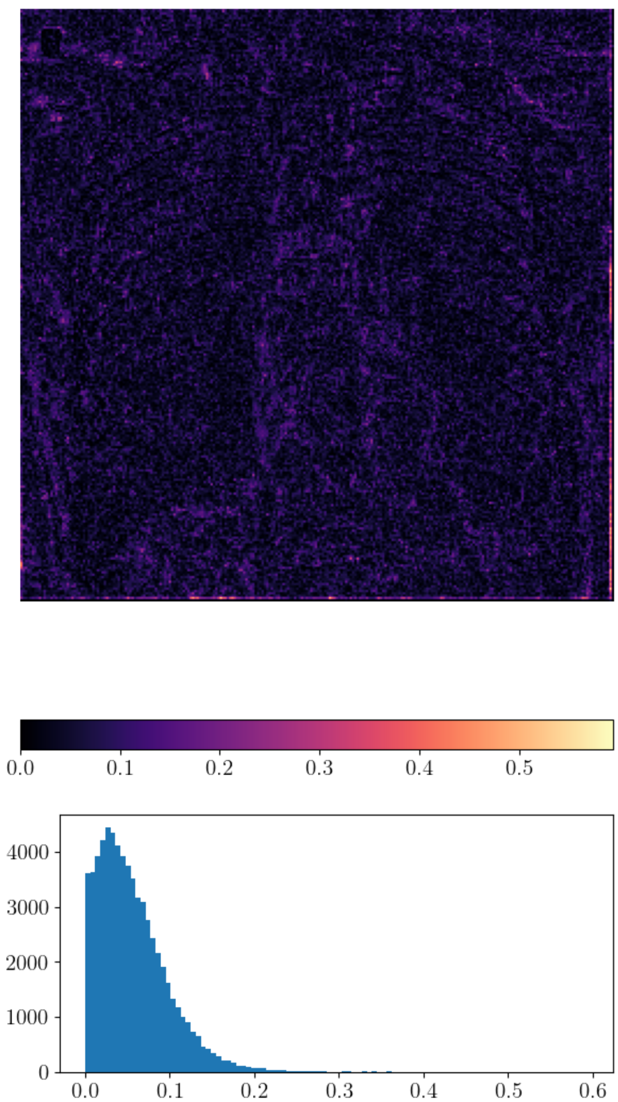


Figure 4: Chest X-ray example. The top row shows the original image on the left and the noisy image, generated with Gaussian noise with $\sigma = 0.5$. The bottom row shows the denoised images using the TV method; the left image with PSNR = 29.945 was produced using the best scalar regularisation parameter $\lambda = 0.07$, while the right image with PSNR = **31.248** was denoised using the regularisation parameter map $\mathbf{\Lambda}_\Theta$ found using NET_Θ.

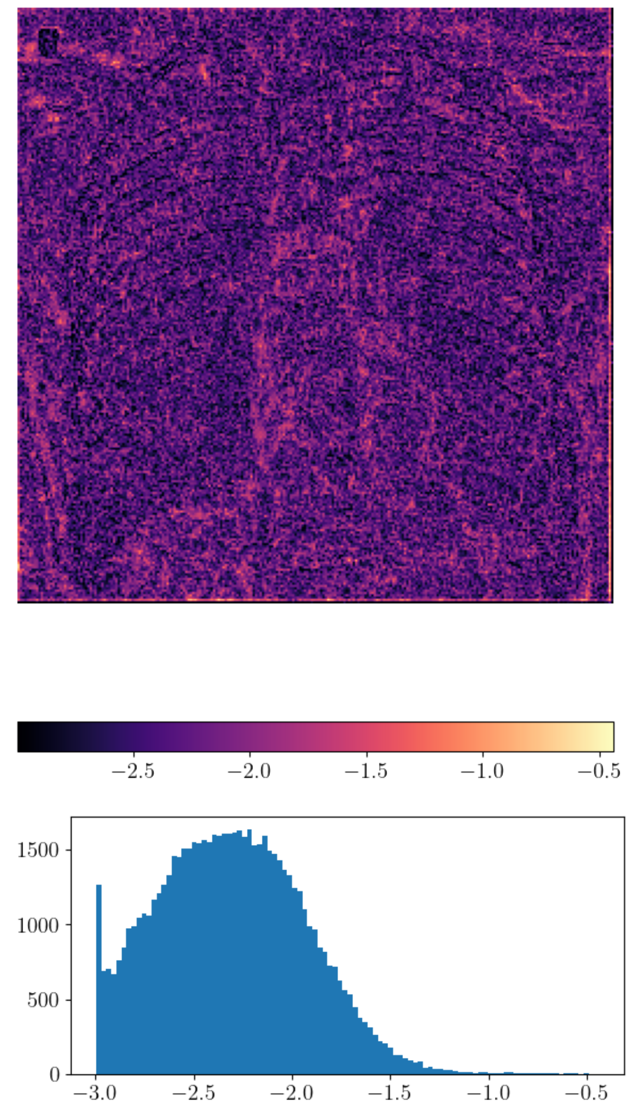
As previously mentioned, we used 100 images with label "normal" from the test set provided in the Chest X-ray dataset for testing. Each test image was noised with a Gaussian noise of mean 0 and standard deviation $\sigma = 0.5$. For each test image, we also did a grid search with 81 equally spaced values between 0 and 0.8 in order to find the best scalar λ for the total variational denoising method, which we used as a baseline to show the effectiveness of our method.

As an example, we present the results for one of the test images. The results of the grid search is shown in the line plots in figure 5. For this particular test image, the highest value of PSNR of 29.945 dB was achieved by $\lambda = 0.07$. The regularisation parameter map $\mathbf{\Lambda}_\Theta$ found using NET_Θ resulted in a PSNR of 31.248 dB. The clean and orignal versions as well as the resulting denoised images are shown in Figure 4.

Qualitatively, we can see that there is a good reduction in the stair-case effect in the denoised image using the regularisation parameter map $\mathbf{\Lambda}_\Theta$ compared to the denoised image using the scalar λ.



(a) $\mathbf{\Lambda}_\Theta$ map and histogram of the values



(b) $\log(\mathbf{\Lambda}_\Theta + 0.05)$ and histogram of the log values

Figure 5: The produced $\mathbf{\Lambda}_\Theta$ map for the chest x-ray dataset example. Since the values are too close to one another in the lower spectrum, the bottom row shows the log of the map and the histogram of the log values for better separation.

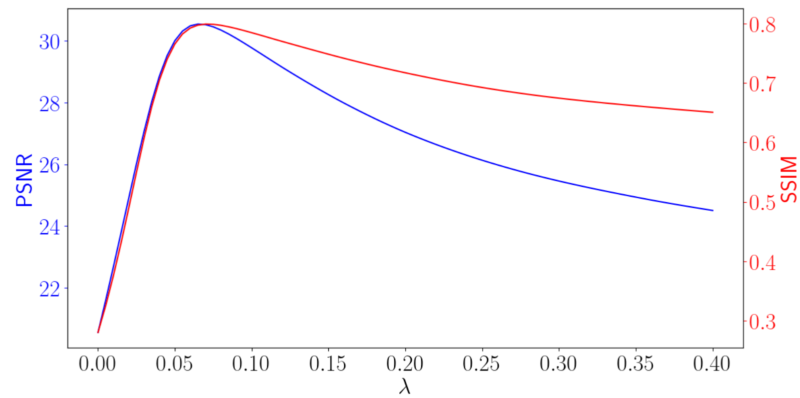


Figure 6: Grid search for the best scalar regularisation parameter λ for the example test image. The highest PSNR of 29.945 dB was achieved by $\lambda = 0.07$.

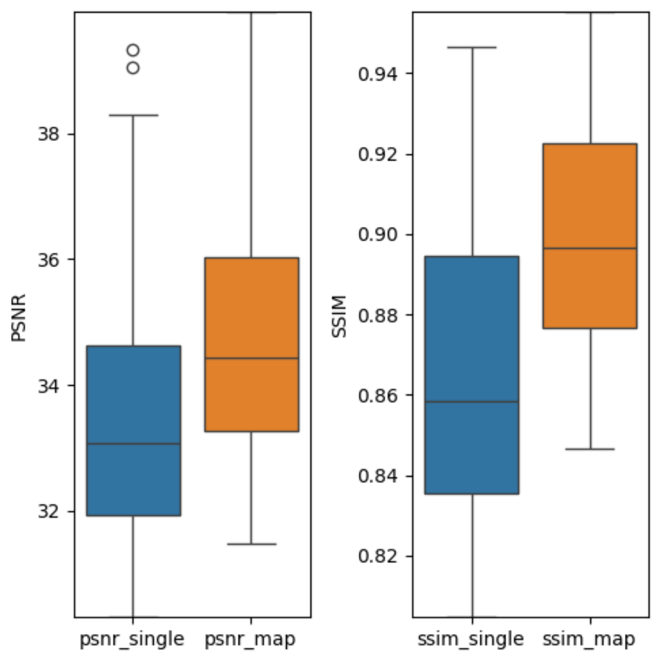


Figure 7: Test results summary

	PDHG - λ	**PDHG -** $\mathbf{\Lambda}_\Theta$
SSIM	0.87 ± 0.04	$\mathbf{0.90} \pm 0.03$
PSNR	33.43 ± 2.16	$\mathbf{34.73} \pm 2.02$

Table 1: Mean and standard deviation of the measures SSIM and PSNR obtained over the test set for the chest x-ray dataset example. The TV-reconstruction using the parameter-map $\mathbf{\Lambda}_\Theta$ improves both measures.

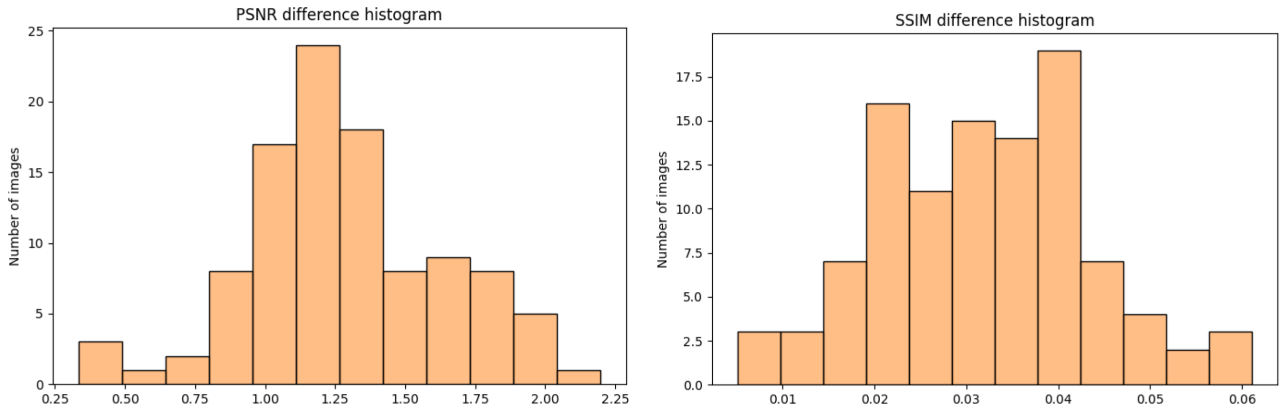


Figure 8: Histograms of the PSNR and SSIM differences between using λ scalar and using $\mathbf{\Lambda}_\Theta$ map for reconstructions of noisy versions of the test images. The $\mathbf{\Lambda}_\Theta$ map method consistently outperforms the scalar λ method.

Plotting the values for all the test images, we can see that the regularisation parameter map consistently outperforms the scalar λ. The box plots in Figure **??** show the distribution of PSNR and SSIM values for the test images using the two methods. The regularisation parameter map consistently produces higher PSNR and SSIM values than the scalar λ.

5 Conclusion

6 Future Work

References

[1] Daniel Kermany, Kang Zhang, and Michael Goldbaum. *Large Dataset of Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images*. Version V3. 2018. DOI: 10.17632/rscbjbr9sj.3.

[2] Andreas Kofler et al. *Learning Regularization Parameter-Maps for Variational Image Reconstruction using Deep Neural Networks and Algorithm Unrolling*. 2023. arXiv: 2301.05888 [math.OC]. URL: https://github.com/koflera/LearningRegularizationParameterMaps.

[3] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791. URL: https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition.

[4] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[6] Emil Y Sidky, Jakob H Jørgensen, and Xiaochuan Pan. "Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle–Pock algorithm". In: *Physics in Medicine Biology* 57.10 (Apr. 2012), p. 3065. DOI: 10.1088/0031-9155/57/10/3065. URL: https://dx.doi.org/10.1088/0031-9155/57/10/3065.

The following list contain the primary Python libraries used in the implementation of this project.

- `NumPy`: Utilised for numerical representations and operations, including vector and matrix multiplication.
- `PyTorch`: Employed for constructing and training neural network models due to its robust, flexible, and efficient computational dynamics.
- `matplotlib` and `seaborn`: Employed for data visualisation tasks, including the creation of histograms, box plots, and other graphical representations.
- `scikit-learn`: Used for benchmarking our model against traditional machine learning algorithms, providing tools for data preprocessing, cross-validation, and more.
- `random`: Critical for generating random numbers, used extensively in stochastic processes like K-fold cross-validation and bootstrap sampling.
- `wandb`: Integrated for experiment tracking and logging, facilitating the monitoring of model training metrics and performance in real-time.