Imperial College London

MEng Project

Imperial College London

Department of Electrical and Electronic Engineering

Invertible Neural Networks for Image Denoising

Author:
Brandon Cann (CID: 01724765)

Supervisor: Prof P.L. Dragotti

2nd Marker: Dr A. Bhandari

Date: June 20, 2023

Final Report Plagiarism Statement

I affirm that I have submitted, or will submit, an electronic copy of my final year project report to the provided EEE link.

I affirm that I have submitted, or will submit, an identical electronic copy of my final year project to the provided Blackboard module for Plagiarism checking.

I affirm that I have provided explicit references for all the material in my Final Report that is not authored by me, but is represented as my own work.

I have not used any LLM as an aid in the preparation of my report.

Acknowledgements

I would like to express gratitude to:

- My supervisor, Professor Pier Luigi Dragotti for the support, advice, and supervision throughout the project. And for giving me the freedom of choosing how to tailor this project to my interests.

- Post graduate students, for the initial support to get me started on the project and the continued support throughout the project.

- My family and friends for continuous help and support throughout my years at Imperial College London.

ABSTRACT

In imaging, noise is one of the biggest obstacles when it comes to extracting key pieces of information from an image. The conducted research aims to explore the properties of current invertible neural networks for image denoising, to potentially discover new solutions to improving these architectures.
The networks explored in this report were designed for only a single noise type being additive white Gaussian noise. The research conducted was to experiment with other noise types to test the robustness of the architectures and to see what kind of features are being preserved by each network.
Findings suggest there is compatibility with different types of Gaussian Noise and mixtures of Gaussian Noise but further research is needed for other noise types and for a more stable conclusion. The results regarding the features of the networks also currently supply contradicting arguments and hence this area too requires further digging.

Contents

1 Introduction

1.1 Motivation

The use of images has skyrocketed over the past ten years. During the process of acquisition, compression, and transmission, noise corrupts images. Noise corrupts images through channels like transmission, the environment, and others. Image noise is the random variation in a signal that affects the brightness or color of an image during observation and information extraction in image processing[1]. Image processing tasks like video processing, image analysis, and segmentation are negatively impacted by noise, which leads to incorrect diagnosis[2]. Consequently, image denoising is an essential component that enhances comprehension of the image processing task.

Image denoising techniques have emerged as an essential component of computer-aided analysis as a result of the growing number of digital images that have been taken in poor lighting conditions. In today's world, obtaining a clean image by restoring information from noisy images is a pressing issue. Procedures for image denoising eliminate noise and restore a clear image. Due to high-frequency components, it is difficult to tell the difference between noise, edges, and textures in image denoising.

Speckle Noise[3], Impulse Noise[4], Quantisation Noise[5], Poisson Noise[6], and Additive White Gaussian Noise (AWGN)[7] are just a few examples of noise. In analog circuitry, AWGN can be found, while impulse, speckle, Poisson, and quantisation noise can be caused by bit error, bad manufacturing, and too few photons[8].

Medical imaging, remote sensing, military surveillance, biometrics and forensics, industrial and agricultural automation, and individual recognition all make use of image denoising techniques. In contemporary medicine, imaging is crucial. Modern imaging methods, such as X-rays, ultrasonography, CT scans, and MRI, can provide very detailed images of the structures inside your body[9]. Denoising algorithms are fundamental pre-processing steps used to remove medical noise such as Speckle Noise, Rician Noise, Quantum Noise, and other noises in medical and biomedical imaging[10, 11]. Denoising algorithms are used to remove Additive White Gaussian Noise and Salt and Pepper Noise in remote sensing[12, 13]. In military surveillance[14], images from synthetic aperture radar (SAR) provide space and airborne operations. Speckle Noise in SAR images has been reduced by image denoising algorithms[15]. Additionally, forensic images can be tainted by any kind of noise, not just one particular kind. Image denoising techniques have contributed to the reduction of noise in forensic images[16] because noise can lower the quality of the evidence in the image. Image denoising was used to filter rice leaves and find disease in rice plants.

Due to the need of recovering the original image content for effective performance, image denoising is used in a variety of applications including image restoration, visual tracking, image registration, image segmentation, and image classification[17]. Image denoising is unquestionably a hot area of research that spans all academic disciplines. The focus of this project is to see how image denoising can be improved to better compliment these sectors by exploring existing architectures in this field.

1.2 Project Specification

As mentioned, there is a need for better image restoration for practical use. Given two different Invertible Neural Networks; Wavelet Inspired Invertible Neural Network (WINNET)[18] and Recursive Wavelet Neural Network (RWNN)[19], the research is to be conducted upon these two architectures. To evaluate and find possible improvements to existing architectures, there is a need to obtain the current properties such as robustness to different noise types to discover the adaptability of the networks and the types of features that are learnt by the networks to visualise and conclude what areas of an image are most valuable when it comes to image denoising.

1.3 Report Structure

The rest of this report is structered as follows: Chapter 2 covers the background information of this project;covering key networks and their functionality. Chapter 3 focuses on how the noise types to be tested have been implemented, likewise for the feature alteration, and how the models will be trained. Chapter 4 explores the testing process of the results and the initial testing conducted upon the two networks. Chapter 5 examines the results obtained by what has been implemented in Chapter 3 and tested in Chapter 4. Chapter 6 is an overview and evaluation of the project, where Chapter 7 concludes the report and states what work there is still to be done. Finally, Chapter 8 contains information regarding the original repositories and the updated repositories.

2 Background

The following section will start with the base level image restoration with Convolutional Neural Networks (CNNs) for Deep Learning and Recurrent Neural Networks (RNNs). Then, will proceed to more complex architectures that are Invertible Neural Networks (INNs) e.g. Lifting Inspired Invertible Neural Networks (LINNs) and the two architectures that this project is based upon, WINNET and RWNN, followed by other INNs.

2.1 Convolutional Neural Networks in Deep Learning

A CNN can have a number of layers, each of which learns to recognise the various characteristics of an input image. Each image is processed with a kernel/filter/mask to produce an output that gets better and more detailed with each layer. The filters can begin as straightforward features in the lower layers.

In order to check and identify features that uniquely represent the input object, the filters get more complex at each layer. As a result, the input for the subsequent layer is the output of each convolved image, which is the partially recognised image after each layer. The CNN recognises the image or object it represents in the final layer, a Fully Connected (FC) layer.

These filters are applied to the input image when convolution is used. Each filter performs its task and transmits its output to the subsequent layer's filter as it activates particular image features. The operations are repeated for multiple layers as each layer learns to recognise distinct features. Finally, the CNN is able to reconstruct an approximate representation of the ”clean” image by utilising all of the image data that moves through its various layers[20].

Some examples of CNN architectures used in image denoising are: AlexNet[21], VGG[22], and GoogleNet[23] which have been used for computer vision tasks, the initial CNN architecture utilised for image denoising tasks are referenced in[24, 25], the Denoising CNN (DnCNN) was utilised by Zhang et al[26] for super-resolution, JPEG image blocking, and image denoising. Convolutions, back normalisation[27], Rectified Linear Unit (ReLU)[28], and residual learning[29] make up the network.

In addition to general image denoising, CNN has excellent results for blind denoising[30], real noisy images[31], and numerous other applications. Despite the fact that a number of researchers have developed CNN methods for image denoising, only a small number have proposed a review to summarise the methods. CNN methods for image denoising were categorised according to the type of noise in the reference[32].

Reference[33] covers the two CNN image denoising approaches along with all the architectures that can be seen in Figure 1.

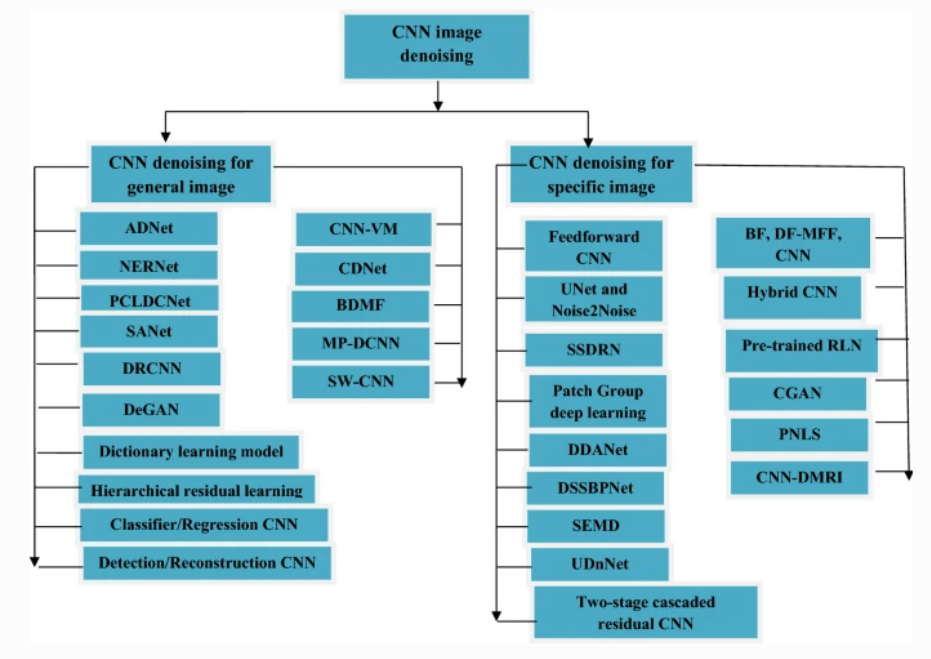

Figure 1: This shows a CNN denoising scheme where mentioned that different architectures can be categorised into two groups[33].

2.2 Recurrent Neural Networks

A special kind of artificial neural network known as a Recurrent Neural Network is designed to work with time series data or data that involves sequences. Only data points that are distinct from one another are suitable for normal feed forward neural networks. However, in the event that we have data in a sequence in which one data point is dependent on the previous data point, we will need to adjust the neural network to take these dependencies into account. RNNs use a concept called ”memory” to store the states or information of previous inputs in order to produce the subsequent output in the sequence.[34] The previously mentioned CNNs and these RNNs can be used as part of Invertible Neural Networks such as in the forward-feeding pass.

2.3 Invertible Neural Networks

Architectures of neural networks that are designed to be invertible are known as Invertible Neural Networks (INNs). They frequently have explicit formulas and tractable algorithms for calculating the Jacobian determinant and inverse map. The flow layers, which are special invertible layers, are carefully designed to create INNs. Reference[35] goes into further detail of how INNs work and uses other than image restoration.

2.3.1 Basic Convolutional Invertible Neural Network

This is a basic INN that makes use of CNNs. The ResNet-34[36] image classification network serves as the foundation for this CINN architecture. However, there are numerous significant differences. The first is that this architecture does not use max pooling or any other dimension reduction method because the output needs to be the same resolution as the input. To account for the larger feature maps, the number of channels per layer are kept constant throughout the network at 64. Additionally, this network is smaller, with 25 hidden layers. After that, the BReLU[37] activation function is applied and a convolution with three 7x7x64 kernels is used to take the place of the FC layer. It uses 224x224 patches to train the network, but at inference time, larger inputs can be used, the size of which is only limited by how much memory is on the device that is running the model. Figure 2 provides an overview of this architecture[38].

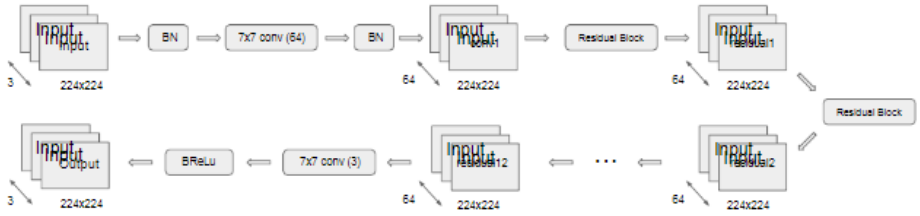

Figure 2: A visual representation of the CINN mentioned above[38].

2.3.2 Lifting Inspired Invertible Neural Networks

A non-linear transform can effectively distinguish image content from noise using a LINN. A Lifting-Inspired Invertible Neural Network's forward pass produces a redundant representation of the noisy input image. The simple denoising network removes features that correspond to noise in order to denoise the transform coefficients. The Lifting-Inspired Invertible Neural Networks' backward pass is then used to construct the denoised image[39].

The LINN is capable of learning these features by first splitting the input image into odd and even parts using a split operator. The next function is to then use a predictor on one of the parts to "predict" the other part and is then followed by an updater to tune this prediction. After this process is complete, you are left with a detailed part of the input image and a coarse part. After undergoing a similar process in the backward pass, the merge operator is used to complete the output[40].

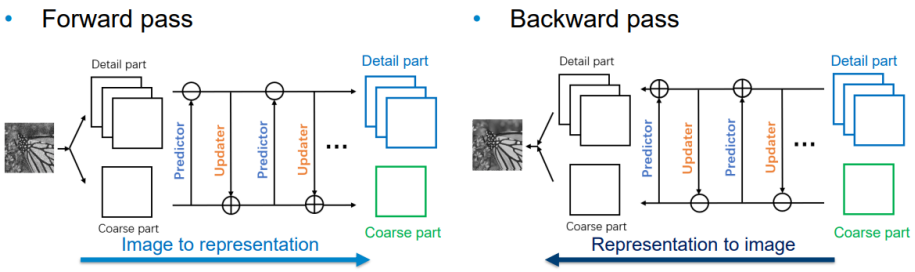

Figure 3: A visual representation of a LINN[40].

2.3.3 Wavelet Inspired Invertible Neural Network

This network makes use of the previously mentioned LINN. There are two main aspects of the WINNET's convolutional network architecture that draw inspiration from wavelets. To begin, the overall structure adheres to the wavelet denoising principles of a multi-scale forward transform, denoising of the coefficients of the detail transform, and an inverse transform. Second, the forward and inverse transforms utilised are the forward and backward passes of an INN whose architecture is influenced by wavelet theory's lifting scheme. The network acquires the wavelet transform's multi-scale property, perfect reconstruction property, and sparsifying capability as a result. Figure 4 depicts a WINNET with a K-level LINN, in which Convolutional Neural Networks with soft-thresholding as a non-linear activation function take the place of conventional predictor and updater operations and iterate only on the coarse version of the image. A redundant decomposition is produced when we substitute a redundant linear operator for the split operator. The merge operator is the inverse of the split operator in the backward pass of LINNs. In addition, the filters are upsmapled in the convolutional layers at larger scales to imitate the wavelet theory implementation of the algorithm à trous[41]. A non-invertible denoising operation on the detail coefficients is used to achieve sparsity. The denoising operation forces the network to sparsify the detail coefficients. Denoising also makes use of a sparsity-driven network[18].

2.3.4 Recursive Wavelet Neural Network

This network is similar to the previously mentioned WINNET and also incorporates aspects of a RNN. A blind Denoising Neural Network (DNN) designed to remove spatially variable Gaussian noise is the proposed architecture. It consists of an inverse transform, a processing step in the transformed domain, and a forward transform operator. A LINN and a Noise Estimation Network (NENet) are the components of RWNN that predict noise level maps. It is in charge of separating the clean and coarse signal from the noisy residuals/details. A denoising sub-network is used to denoise the aforementioned residuals in the transformed domain. The result, along with the coarse, is back-projected via RWNN-1 to the original domain. If the noise variance is large, RWNN-F can be repeated on the coarse parts to get better results, where F is the network's fine-tuning.

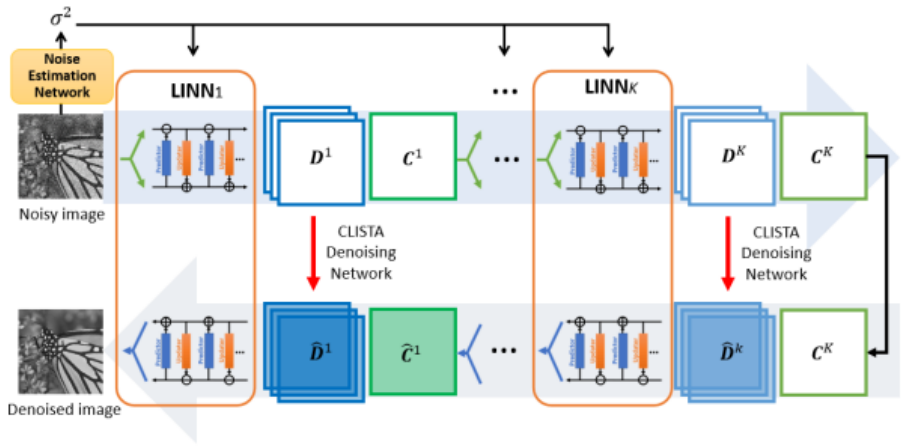

Figure 4: This shows the architecture of WINNET[18].

RWNN employs a divide and conquer strategy from an algorithmic perspective. There are now two sub-problems to the denoising task: denoising the general and the specific. It stands to reason that the coarse will be significantly less noisy. As a result, denoising it is a simpler problem that can be solved recursively, and FusionNet can use the coarse solutions as a precondition to make denoising the details easier[19].

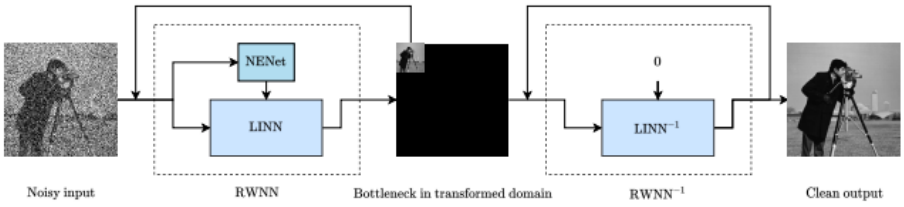

Figure 5: This shows the architecture of RWNN[19].

2.3.5 Denoising Invertible Neural Network

A sparsity-driven denoising network and a LINN are part of the DnINN approach. The denoising network employs sparse coding, while LINN employs a non-linear transform with perfect reconstruction property. A model that is easy to understand is produced by the clear objectives and functionalities of both the LINN and the denoising networks.

The three main steps of the DnINN method are based on the transform-based denoising principle[42]. First, the noisy image undergoes a non-linear transformation during the forward pass of the LINN, resulting in three detail channels and one

coarse channel. The coarse channel ought to be less affected by noise, whereas the detail channels ought to contain noise and high-frequency content. A multi-scale architecture can be created by iterating the decomposition on the coarse channel. The coarse channel is then moved to the reconstruction step while the sparsity-inspired denoising network denoises the detail channels. By using the LINN's backward pass on the denoised representations, the denoised image is finally reconstructed[39].

2.3.6 Other Invertible Networks

Quasi-Invertible Network: The main invertible branch and a non-invertible encoder-decoder branch are combined in the quasi-invertible module. The forward approximation and reverse restoration passes are denoted by the blue and red arrows, respectively, in Figure 6's overview architecture of the proposed method. The model takes an image as the input during the forward approximation phase and produces the processed image as the output. The quantised, compressed output image is saved. The original image can be restored along the reverse pass during the restoration[43].

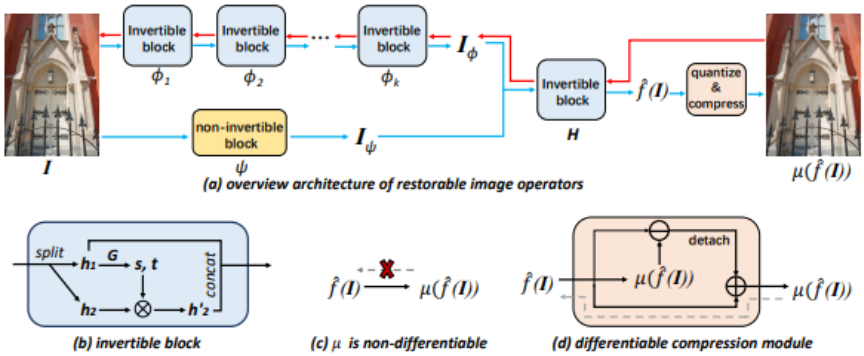

Figure 6: This shows the architecture of the quasi-invertible module[43].

Invertible Denoising Network: A number of Downscale Blocks make up the Invertible Denoising Network, or InvDN. Following a number of Invertible Blocks, each Downscale Block has a Wavelet Transformation layer that reduces the spatial resolution by two and increases the number of channels by four. It learns a latent variable and a low-resolution image from the noisy input in the forward pass. In the backward procedure, the clean image is restored by combining the low resolution portion of the image with the latent variable, which is randomly sampled using a normal distribution[44].

Invertible UNet: This is a type of conditional INN (cINN). The UNet architecture[45] serves as the foundation for this model, which begins by substituting invertible layers for all common layers. A conditioning network with a similar UNet structure is also

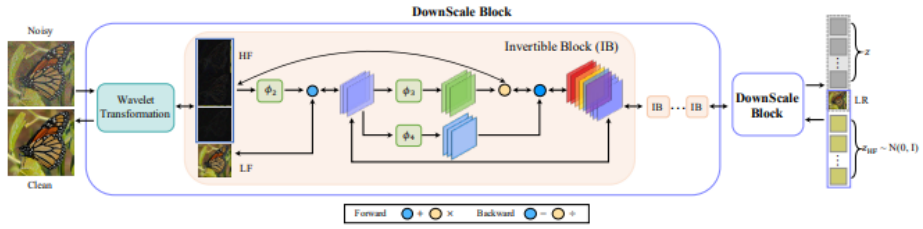

Figure 7: This shows the architecture of the Invertible Denoising Network[44].

introduced. The layers need not necessarily be invertible in this instance. Similar to iUNet[46], the conditioning network employs the same spatial downsampling and upsampling scales. Figure 8 depicts the entire network as a representation. In the upsampling path, iUNet concatenates the splits one at a time, rather than all at once in the final layer. A conditioning network and end-to-end invertible UNet make up this model. The conditioning input is processed by the conditioning network, and it is sent to the appropriate conditional coupling layer as an output[47].

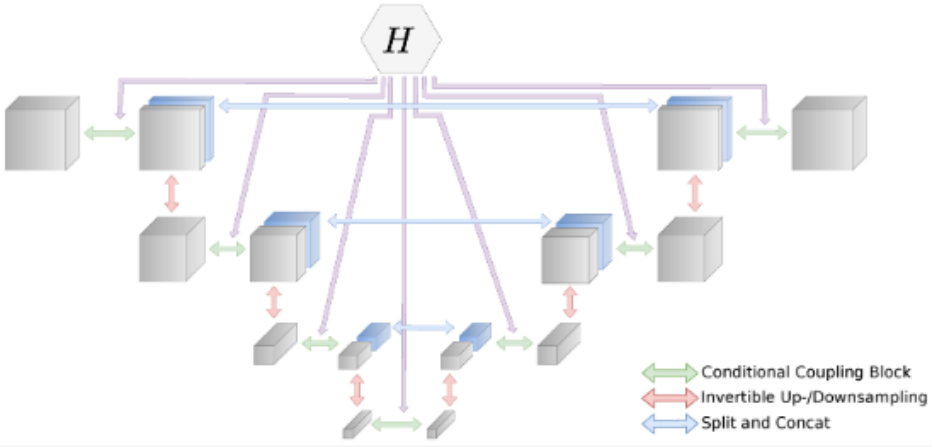

Figure 8: This shows the architecture of the Invertible Denoising Network[47].

There are also other INNs that are to do with imaging but not with image restoration in particular: Dual Latent Variable INN for enhancing image rescaling[48], Auto-Invertible NN Q'tron for image half toning[49], Conditional INN for image translation/generation[50, 51].

3 Implementation

3.1 Noise types

This section will explore the different noise types that have been implemented as well as the proposed training models to be constructed from them.

Figure 9: Here is the original image as a reference to compare with the noisy images that will follow in this section.

3.1.1 Additive Gaussian Noise

$$X \sim \mathcal{N}(\mu, \sigma^2).$$

Additive Gaussian Noise is a normal distribution where the mean μ is zero and the variance σ^2 is the noise level to be changed. The default noise levels that have been used in WINNET and RWNN are of the form $\frac{NL}{255}$ where NL is the noise level to be adjusted and the pretrained networks have been trained with $NL = 15, 25, 50$.

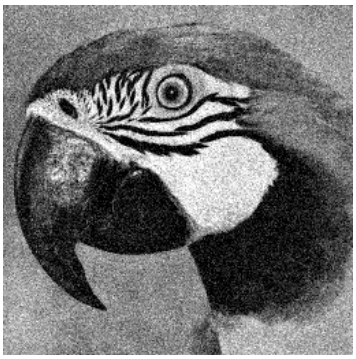

Figure 10: Noisy image implemented using Additive Gaussian Noise.

This noise has been implemented by using the `torch.normal(mean, variance)` function to calculate the noise distribution. As this is an additive noise, this noise can be added to the original image to produce the noisy image.

3.1.2 Conflated Gaussian Noise

Conflated Gaussian Noise is a combination of Gaussian Noise and Conflation Noise. Conflation Noise are noises independent of the Gaussian Noise. For the purpose of this project, it was decided to use a random sum of two other Gaussian Noises with different means and variances. As another variant of Conflated Noise, there was some implementation of Salt and Pepper Noise with the Gaussian Noise and also with the previously defined Conflated Gaussian Noise.

Figure 11: Noisy image implemented using Conflated Gaussian Noise.

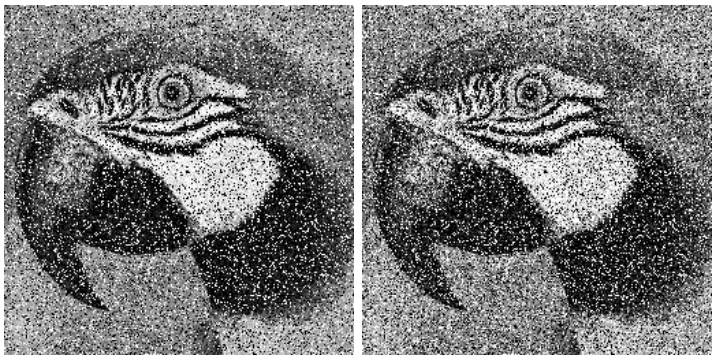

Figure 12: The above images are the previously mentioned variants with the left image using Gaussian Noise and the right Conflated Gaussian Noise.

The noisy image for this noise type has been produced by the sum of the original image, the same noise mentioned before hand in the Additive Gaussian Noise section, and Conflation Noise. The implementation of Conflation Noise is as follows: using the `np.concatenate()` function on two `np.random.normal(mean, variance)` functions to combine the two normal distributions, followed by a `np.random.shuffle()` function call to shuffle the two distributions together. And then converting the result to a float tensor to allow compatible addition between the image and noise.

3.1.3 Salt and Pepper Noise

Salt and Pepper Noise is where random pixels of the image are set to pixel values of either 0 or 255 as each pixel is represented by 8bits, where 0 represents black and 255 represents white.

Figure 13: Noisy image implemented using Salt and Pepper Noise.

Implementation wise, an array of zeros with size equal to the original image is constructed using `np.zeros()`. Using to random number generators of the form `cv2.randu(array,0,255)` followed by a thresholding function `cv2.threshold()`, two noise arrays are made; one for white pixels and one for black pixels. As before, these two arrays are converted to float tensors and summed to the original image.

3.1.4 Poisson Noise

$$X \sim \mathcal{P}(\lambda).$$

Poisson Noise is calculated by the poisson distribution $P(X) = \frac{\lambda e^{-\lambda}}{X!}$ where λ is the parameter to be changed to adjust the degree of noise applied.

Unlike the other noise types that are mentioned here, Poisson is not an additive noise but is an applied noise. For the previous noise types, the general form has been `Noisy_Image = Input_Image + Noise`. For Poisson it is simply `Noisy_Image = Poisson(Input_Image)`. This has been implemented using the `torch.poisson()` function, where the λ can be adjusted by scaling the image. For this project, there has been no scaling of the image and is set as the default state.

Figure 14: Noisy image implemented using Poisson Noise.

3.2 Features

In WINNET, modifications were made to the `foward()` function so that the values of the 15 detail layers and coarse layer feature maps could be modified. This was part of the initial testing for this project which will be further discussed in the next section.
Similarly, in RWNN changes were made to the `dae()` function where this function was already designed to set the detail layers feature maps to zero but not the coarse layer. Due to the recursive nature of this architecture, it was more intriguing on how the recurrsiveness plays a role when it comes to producing images that have had their feature space altered and hence the focus for feature alteration was mainly conducted on this architecture.

3.3 Training

In WINNET for each noise type mentioned above a level 1 and level 2 model have been trained for 10 epochs at batch size 32 and 16 respectively. The process of training takes approximately within a day and within 2 days for the level 1 and level 2 models respectively. Other models that have been trained include the other type of Conflated Gaussian Noise mentioned previously at level 1.

The training process for RWNN is slightly different in terms of training a DAE first followed by finetuning. Each stage is trained for 10 epochs each at batch size 32. Similarly to WINNET, each noise type has a model trained from it but at a recursion depth of 1 only due to time constraints. Each model can be trained within a day. Further models that have been trained include Gaussian Noise models where the variance is proportional to the image which has undergone edge detection; areas with no edges will have near zero variance whereas edges will have a large variance.

The parameters for each noise type is as follows: NL25 and 0 mean for Additive Gaussian Noise, NL15 and 50 for the two normal distributions for the Conflation Noise with 0 mean, and the thresholds for Salt and Pepper Noise are set to 220 -

meaning the numbers produced by the random number generator exceeding 220 are converted to 255 with the rest set to 0, this process is inverted for the black pixels to produce pixel values of 255. For the edge detection dependant noise, a normal distribution with 0 mean and NL between ϵ and 100 is used where $\epsilon = 1e-6$ depending on the edge detection image. Edge detection is performed by convolving a padded version of the original image with a Prewitt/Sobel mask in the x and y direction which are of the form:

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 15: The above left image is the noisy image generated with Prewitt edge detection and the right with Sobel edge detection.

An extra level of noise for the Prewitt edge detection noise was implemented with the same parameters but with double scaling on the variance which can be visually seen in Figure 16.

Figure 16: Prewitt edge detection noisy image with double variance scaling.

4 Testing

4.1 Noise

With respects to the initial goal, the evaluation method will be a comprehensive discussion about the findings and whether any of the findings have lead to a better approximation of the output. For the given two architectures, the methods of evaluation are the same. They can be visually evaluated and they can be evaluated using a noise estimation network, NENet.[52, 18]

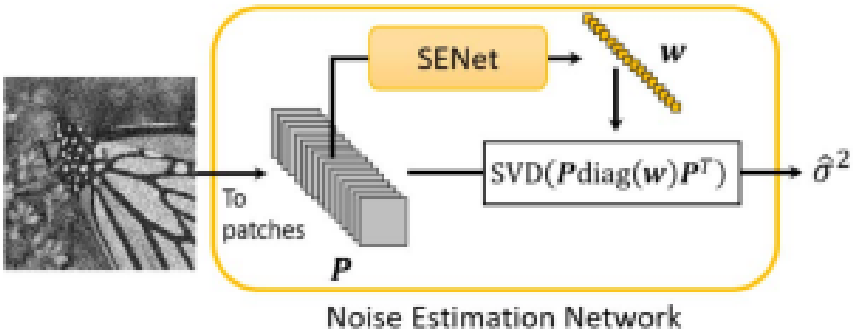

Figure 17: This is the noise estimation network previously mentioned[18].

In simple terms, NENet produces an estimation for the variance of the noise for the output images via performing singular value decomposition on the weights acquired by the Selection Network, SENet. SENet produces an estimate of the scalar weights from the input image which has been converted to patches. This estimation of the noise variance can be used to produce peak to signal noise ratio (PSNR) and structural index similarity (SSIM) values so that the different models can be compared to each other in a quantitative manner. PSNR is a comparison method which compares pixel-by-pixel and SSIM compares based on three factors: loss of correlation, luminance distortion, and contrast distortion[53]. Running the testingg algorithm using the trained models produces approximations of the images so the different models can be qualitatively compared too.

4.2 Initial Testing

4.2.1 WINNET

The initial testing consisted of reproducing some similar results to that of Figure 8 in the WINNET paper[18]. This included the manipulation of the feature maps and setting the detail parts to zero and keep the coarse part, and then the opposite of setting the coarse parts to zero and keep only the detail parts for a single layer. The pure purpose of this testing was to make sure the environment for the repository

had been correct setup so that meaningful results could be obtained in the following sections. It can be seen from Figure 18 that similar features for image 3 in set12 have been learnt by the level 1 and level 2 architectures for the detail layers. As for the coarse results, as expected the level 1 map is compact and the level 2 map is spread.

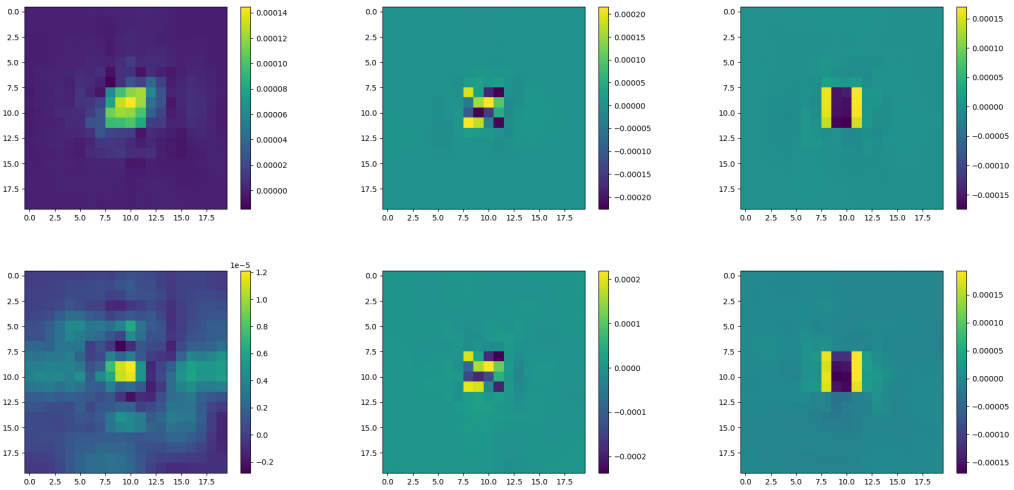

Figure 18: The above are the feature maps of the level 1 (top row) coarse layer, detail layers 4 and 5 respectively. The row below is the same as the top but for the level 2 network.

As a baseline for comparison of the results to be produced, there was a need to test the PSNR values obtained from the pretrained networks. Table 1 shows the results obtained that will later be used to check the robustness of the architecture to other noise types.

The general pattern is as expected where the higher the noise level, the lower the PSNR score. We can also see that the model general tests better on the smaller test set and the level 2 models perform slightly better than their level 1 counterparts. A visual representation of these PSNR scores can be seen in Figure 19. As can be seen there is not much visual difference between a PSNR score of 27.3 and 31.5.

4.2.2 RWNN

Similar to the previous section, the same process has been undertaken for the RWNN architecture. From Table 2, comparing to the WINNET noise level 25 model, WINNET appears to have better overall performance but not by much. In this case the finetuned model (epoch73) performs slightly better than the dae model (epoch41).

Model	PSNR
Lv1 NL15 Set12	31.466885
Lv1 NL15 Set68	29.920182
Lv2 NL15 Set12	31.596462
Lv2 NL15 Set68	30.065497
Lv1 NL25 Set12	30.476462
Lv1 NL25 Set68	29.231154
Lv2 NL25 Set12	30.548351
Lv2 NL25 Set68	29.272308
Lv1 NL50 Set12	27.318809
Lv1 NL50 Set68	26.304159
Lv2 NL50 Set12	27.407495
Lv2 NL50 Set68	26.362032

Table 1: Pretrained models of WINNET tested with Additive Gaussian Noise at their corresponding noise levels.

Model	PSNR
epoch41	28.04408
epoch73	29.14376

Table 2: Pretrained models of RWNN tested with Additive Gaussian Noise noise level 25 on set12.

Figure 19: A visual representation taken from image 7 of set12 for each of the models with the level 1 models in the top row with noise levels 15, 25, and 50. Similarly for the level 2 models below.

5 Results

5.1 Noise Robustness

This section will discuss the results obtained from testing the two architectures and their models with the implemented noise types to see their compatibility with these noise types.

5.1.1 WINNET

Table 3 shows the results obtained by testing the pretrained networks with the new noise types. As the Conflation Noise is made up of Gaussian Noises it was expected to a degree that the results for this noise type would produce decent results. For Salt and Pepper Noise and Poisson Noise, it can be established that the Gaussian Noise-trained models do not do a good job of reconstructing the images. Also differing from before, the level 2 models cannot be said to be performing better than the level 1 models like for the Gaussian Noise.

Model	Conflated Gaussian	Salt and Pepper	Poisson
Lv1 NL15 Set12	19.045894	10.54486	6.896231
Lv1 NL15 Set68	19.055036	10.465966	7.223439
Lv2 NL15 Set12	19.458272	9.279176	6.861679
Lv2 NL15 Set68	19.443823	9.261608	7.229291
Lv1 NL25 Set12	18.610532	8.753754	7.898438
Lv1 NL25 Set68	18.648312	8.720106	8.214223
Lv2 NL25 Set12	18.566898	7.704486	7.870728
Lv2 NL25 Set68	18.638102	7.677025	8.281721
Lv1 NL50 Set12	20.854734	5.573508	9.88541
Lv1 NL50 Set68	20.725473	5.414897	10.18487
Lv2 NL50 Set12	21.005955	5.380957	9.231499
Lv2 NL50 Set68	20.917268	5.225139	9.909037

Table 3: PSNR scores of the pretrained WINNET networks with the newly implemented noise types.

After training the models for each noise type and testing these models, the results can be seen in Table 4.

The Conflated Gaussian Noise model has produced similar quality results to the pretrained models for Gaussian Noise and better for Conflated Gaussian Noise. As expected, the results for the other two noise types are similar to that of the pre-trained models. Although the model has been trained on Conflated Gaussian Noise, the results for Additive Gaussian Noise are slightly better. This suggests that the model needs a bit more epochs to learn the features of this noise type or that the architecture as a whole is not flexible for other noise types/combinations.

Overall, the Salt and Pepper Noise models have produced very poor results with the exception of the level 2 models against Salt and Pepper Noise. This is rather interesting as from previous results, the level 1 and level 2 models have produced fairly similar results. The level 2 models seem to have learnt the features at a faster rate which implies that the level 1 models too can produce similar PSNR scores if trained with more epochs. This could also simply be a case of lower level models not being able to learn the features. It would also be very beneficial to obtain results for the level 3 model for this noise type to grasp a deeper understanding for Salt and Pepper Noise.

The Poisson Noise models have all around produced decent results for all noise types except for Salt and Pepper Noise. The models do not seem to have learnt features specific to Poisson Noise and have learnt more of general features that seem to apply to three of the four noise types. This could imply that either more epochs are required to learn the Poisson Noise features or that the architecture is unable to learn these features.

From these results, a reasonable conclusion can be made that the WINNET architecture is capable of producing good restorations for the noise types that involve some sort of Gaussian Noise and seems to struggle in particular with Salt and Pepper Noise. Looking at Figure 20, a PSNR score of around 20 (3rd image) does not look much like a clean image and nor does a PSNR score of around 24 (4th image). Some further tuning of parameters may lead to situations where more reliable images can be outputted for Salt and Pepper Noise and Poisson Noise.

Model	Additive Gaussian	Conflated Gaussian	Salt and Pepper	Poisson
Lv1 Conflated Gaussian Set12	29.64028	27.810166	6.958313	7.960184
Lv1 Conflated Gaussian Set68	28.269686	26.827844	6.869752	8.426562
Lv2 Conflated Gaussian Set12	29.804067	28.061167	5.887913	9.245602
Lv2 Conflated Gaussian Set68	28.450479	26.98329	5.752571	9.615401
Lv1 Salt and Pepper Set12	5.734496	5.75299	5.958696	5.771442
Lv1 Salt and Pepper Set68	6.802599	6.82443	7.05366	6.840518
Lv2 Salt and Pepper Set12	9.257773	9.528797	20.398139	11.548959
Lv2 Salt and Pepper Set68	10.29845	10.692598	20.686105	12.782442
Lv1 Poisson Set12	24.475205	23.487115	6.224758	21.657514
Lv1 Poisson Set68	24.281904	23.321314	6.119352	21.967513
Lv2 Poisson Set12	23.028978	22.776281	5.312347	21.91561
Lv2 Poisson Set68	22.891768	22.730147	5.144205	22.16953

Table 4: PSNR scores of the newly trained WINNET networks.

The results for the mentioned other types of Conflation Noise can be found in Table 5. From the previous table we have seen that the level 1 model for Salt and Pepper Noise has performed poorly and thus makes sense as to why these results are also

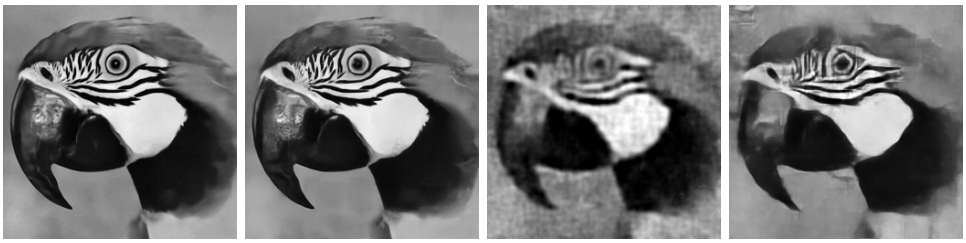

Figure 20: WINNET best results for each noise type: Additive Gaussian Noise, Conflated Gaussian Noise, Salt and Pepper Noise, and Poisson Noise respectively.

very poor.

Model	Additive Gaussian	Conflated Gaussian	Salt and Pepper	Poisson
Salt and Pepper with Additive Gaussian Set12	6.265811	6.264364	6.000292	5.94566
Salt and Pepper with Additive Gaussian Set68	7.360372	7.352221	7.096638	7.019549
Salt and Pepper with Conflated Gaussian Set12	5.793858	5.798342	5.986681	5.768698
Salt and Pepper with Conflated Gaussian Set68	6.862009	6.868357	7.084494	6.83607

Table 5: PSNR scores of the alternative Conflated Gaussian Noise level 1 networks.

5.1.2 RWNN

The same process has been undergone for the RWNN repository, where the results are in Table 6. Similar to WINNET though not as good, the results for the Additive Gaussian and Conflated Gaussian models perform well for Additive Gaussian and Conflated Gaussian Noise.

Given three of the eight models have returned NaN values for testing against Salt and Pepper Noise suggests that this architecture is not compatible with Salt and Pepper Noise and the rest of the results are fairly poor for that column. However, the Salt and Pepper models themselves do outperform the ones of WINNET. This models seems to have learnt a more general features of the image that are applicable to all noises looking at how each PSNR is quite similar, but these scores are still too low to be fit for use. Perhaps like WINNET, the level 2 model may lead to better results.

Surprisingly, the Additive Gaussian model outperforms the Poisson model in every aspect. This model too does not seem to have learnt the features of Poisson Noisy images and as before requires more epochs to fully conclude the compatibility.

Model	Additive Gaussian	Conflated Gaussian	Salt and Pepper	Poisson
Additive Gaussian dae	27.31649	25.8008	5.326977	18.99414
Additive Gaussian finetuned	29.00531	24.82932	5.297812	15.14422
Conflated Gaussian dae	27.65586	25.99552	NaN	10.26103
Conflated Gaussian finetuned	28.77703	25.72963	5.312058	10.07102
Salt and Pepper dae	13.41835	13.3882	13.12375	13.38209
Salt and Pepper finetuned	12.72312	14.47709	11.70614	14.78912
Poisson dae	25.02724	22.40338	NaN	16.33674
Poisson finetuned	24.3887	21.12914	NaN	13.40956

Table 6: PSNR scores of the newly trained RWNN networks.

The testing results for the edge detection noise models are in Table 7 where the finetuned model with Prewitt edge detection gave rise to the best result the RWNN architecture has produced. For the other noise types, the finetuned models seem to perform worse in general compared to their dae counterparts. Once again, there is decent performance for the Gaussian-based noise types.

Model	Additive Gaussian	Conflated Gaussian	Salt and Pepper	Poisson	Prewitt	Sobel
Prewitt dae	26.66658	24.17316	5.288819	13.84015	27.83065	27.88905
Prewitt finetuned	22.73676	17.76723	5.34452	8.039541	31.13298	31.09282
Sobel dae	26.74135	24.22693	5.304328	13.32073	27.82178	27.8264
Sobel finetuned	22.72854	17.84353	5.333769	8.1984	30.21367	30.26819

Table 7: PSNR scores of the trained edge detection-based noise networks.

5.2 Features

From Figure 21, it can be seen that altering the number of recursion levels only seems to have a significant effect on the Additive Gaussian model's images. The visual performance of the two edge detection noise models outperform the one of the Additive Gaussian model and hence suggest that edges are not as much of significance when it comes to the reconstruction of an image. This is further supported by the fact that there is virtually no difference between the images in the bottom two rows.

Looking at the first two rows of Figure 22, the Additive Gaussian model seems to produce a much better image from just the coarse layer and thus would imply that the edges of an image are actually very key to the reconstruction of an image. However, when increasing the noise at these edges it has now produced results better than when there was less noise on the edges. Another model with more of an extreme amount of noise at the edges is required to conclude the importance for edges for

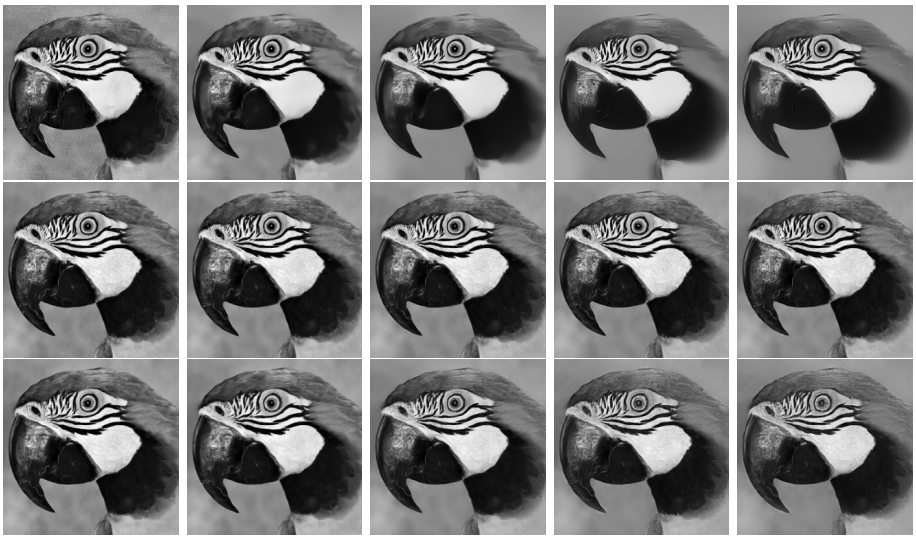

Figure 21: Output image for recursion levels 1 to 5 (left to right) for the Additive Gaussian model (row1), Prewitt model (row2), and Prewitt model with doubled variance (row3).

setting all the detail layers to zero as the results for the second row may be caused by too little noise. This means that the model with an extreme amount of noise is likely to go of two ways: producing results somewhat similar to the second row - proving that the reconstruction is in fact sensitive the amount of noise at the edges, or an alternatively producing images that are better than of what is present in both the first and third row - supporting the theory that edges are indeed not vital to the reconstruction of an image.

The results for setting the coarse layer to zero in Figure 23 is a similar scenario where the higher noise on the edges produces a better output image. In this case, it is clear that the edge detection-based noise starts to produce a better image with less recursions. Conducting the testing mentioned in the previous section will also yield additional results to be interpreted for this part too, thus most likely being able to conclude the importance of edges.

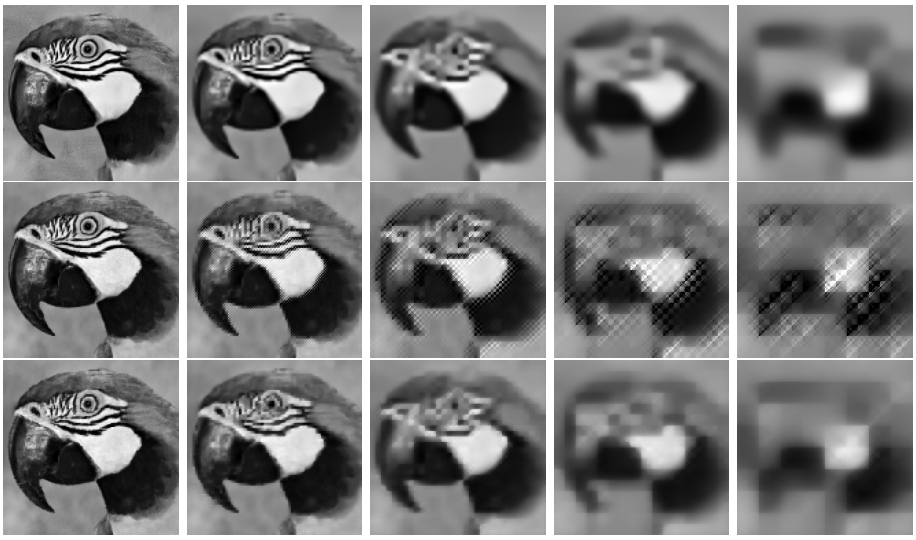

Figure 22: Output image for recursion levels 1 to 5 (left to right) with the detail layers' feature map set to zero for the Additive Gaussian model (row1), Prewitt model (row2), and Prewitt model with doubled variance (row3).

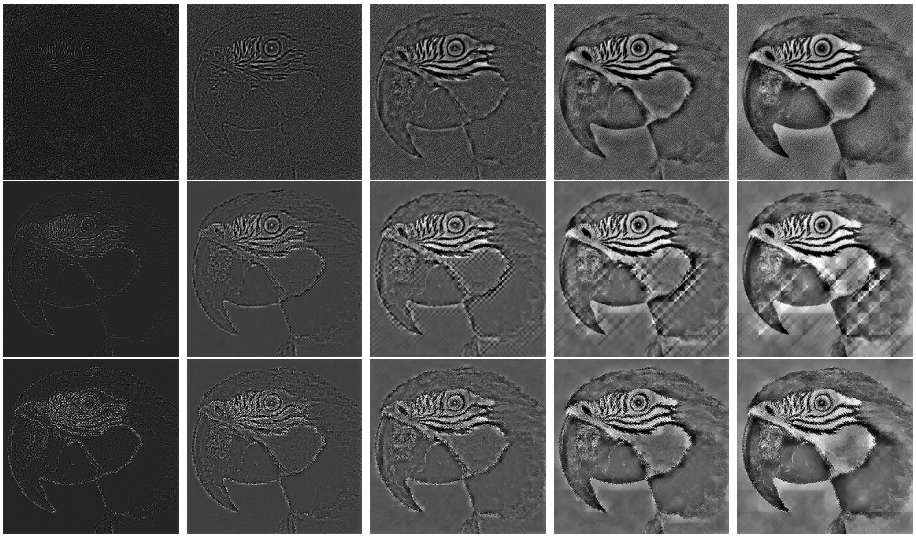

Figure 23: Output image for recursion levels 1 to 5 (left to right) with the coarse layer's feature map set to zero for the Additive Gaussian model (row1), Prewitt model (row2), and Prewitt model with doubled variance (row3).

6 Evaluation

The objective of this research project was to identify areas of improvement for these networks. In terms of measuring the extent of completion, the milestones that are based off of what has been trained and tested is probably the best method of evaluating this project seeing as you cannot expect what findings will be made. This objective that has been set does not really have a point of completion and thus the number of milestones reached are a better metric for seeing the extent of completion. As each milestone was reached and the results were analysed, a new milestone was created to test a new aspect of the networks and this process would be repeated, therefore there is still plenty of research to be done on these architectures including the ones that have been mentioned in the background of this report.

The two networks explored in this paper have been developed based off of only a single noise type being Additive Gaussian Noise and their respective papers explore the models based off of this. However, the research that has been conducted in this paper has not been conducted elsewhere.
The very obvious advantage of the research conducted is that in the real world, Additive Gaussian Noise will not be the only noise type that affects images and may even be combinations of multiple noise types. Ensuring the compatibility of different types of noise is very key in the sense that if compatible, then there are models available for denoising these other noise types. Altering of the feature maps is a sector that has been explored slightly in both papers, and this paper extends on these areas by exploring the alterations with new noise types which can help identify areas of importance when it comes to preserving key features.
The main disadvantage of the methodology was the fact that the training process takes an extremely long time. Training with a GTX 1080Ti GPU, level 1 models take roughly a day to train whilst the level 2 models would take under 2 days to train. The only solution to this problem is to use a more powerful GPU, a more reccent GPU such as the RTX 4090 could cut down on the time taken to train significantly. The training process was by far the most time consuming part of this project and also the most difficult when it came to debugging the code as errors would sometimes appear just as the model has nearly completed training and the training process has to be restarted after trying to debug the issue. Particularly with the RWNN architecture, there were many issues to do with ill-conditioned matrices/too many repeated singular values and calculations returning NaN values that made training very troublesome and time consuming. The RWNN network is a lot more sensitive to how new models are implemented and is likely to return errors if the noise level is too low/high.

Due to time constraints, higher level models could not be trained where it would have been interesting to see how these would perform as we had seen with the level 1 and 2 Salt and Pepper Noise models. As testing trained models is not as heavy on the GPU, it may have been more time efficient to implement an algorithm to simultaneously test all the models and store the results in their respective folders.

7 Conclusion & Future Work

In terms of testing compatibility of the architectures with different noise types this was fairly successful for level 1 and 2 networks as all the newly implemented noise types had their respective level 1 and 2 models trained and tested against all the types of noise. However, there is more research that could go into this area where similar to what has been done with the Additive Gaussian Noise models in the original repositories of testing different noise levels for each noise type. There is also the grey area of how higher level models would perform but this would require a lot more time or a much more powerful GPU. This can also be extended to testing other noise types that have been mentioned at the very beginning of the report and also not just different Conflation Noises but also just combinations of different noise types.

Although results had been obtained for altering the feature space of the models, it was first thought that edges were a key part of images for reconstruction but the results for increasing the noise at edges then contradicted this initial theory and thus a conclusion cannot be made with what has been tested in this report. This area would require more research into how greater noise levels on edges impact the results and also how other edge detection algorithms perform against each other. Having more data to work with on this sector could help to conclude what areas of an image are key to restoring the image.

As mentioned before, the main difficulties that came with this project was the training process errors and the time required to train these models.
Due to this project being a research project, there is still plenty to be done. Other than what has already been mentioned beforehand, some examples of what can still be done are: training/testing with different datasets, tuning of hyperparameters of the networks, extending what has been conducted in this report to other networks, and research into feature invariancies such as what learnt features remain unchanged when the image has been rotated/scaled etc.

8 User Guide

The links to the original repositories of WINNET and RWNN are as follows respectively:

https://github.com/jjhuangcs/WINNet
https://github.com/andreasfloros/RWNN

All credit goes to the original authors of the repositories. The updated repositories can be found here:

https://github.com/bc319IC/WINNET
https://github.com/bc319IC/RWNN

Changes include: updated compatibility with python version 3.9.13, implementation of the mentioned noise types, implementation of modifying the coarse and detail feature maps, the trained models used as part of this project.

References

[1] Geoff Harris. **Image Noise Explained**. 2013. URL: https://www.learningwithexperts.com/photography/blog/image-noise-why-it-occurs-and-how-to-avoid-it#:~:text=Noise%5C%20is%5C%20caused%5C%20by%5C%20random,digital%5C%20noise%5C%3A%5C%20luminance%5C%20and%5C%20chroma..

[2] Diwakar M and Kumar M. "A review on CT image noise and its denoising". In: (2018).

[3] Ilesanmi AE et al. "Multiscale hybrid algorithm for pre-processing of ultrasound images". In: (2021).

[4] Awad A. "Denoising images corrupted with impulse, Gaussian, or a mixture of impulse and Gaussian noise". In: (2019).

[5] Bingo W-KL et al. "Reduction of quantization noise via periodic code for oversampled input signals and the corresponding optimal code design". In: (2014).

[6] Rajagopal A, Hamilton RB, and Scalzo F. "Noise reduction in intracranial pressure signal using causal shape manifolds". In: (2016).

[7] Buades A, Coll B, and Morel JM. "A review of image denoising algorithms, with a new one". In: (2005).

[8] Goyal B et al. "Image denoising review: from classical to state-of-the-art approaches". In: (2020).

[9] NPS MedicineWise. **Imaging Explained**. URL: https://www.nps.org.au/consumers/imaging-explained#:~:text=Imaging%5C%20plays%5C%20an%5C%20important%5C%20role,your%5C%20body%5C%20in%5C%20great%5C%20detail..

[10] Gai S et al. "Speckle noise reduction in medical ultrasound image using monogenic wavelet and Laplace mixture distribution". In: (2018).

[11] Baselice F et al. "Denoising of MR images using Kolmogorov–Smirnov distance in a non local framework". In: (2019).

[12] Vijay M and Devi LS. "Speckle noise reduction in satellite images using spatially adaptive wavelet thresholding". In: (2012).

[13] Bhosale NP and Manza RR. "Analysis of effect of noise removal filters on noisy remote sensing images". In: (2013).

[14] Berens P. "Introduction to synthetic aperture radar (SAR)". In: ().

[15] Sivaranjani R, Mohamed-Mansoor-Roomi S, and Senthilarasi M. "Speckle noise removal in SAR images using multi-objective PSO (MOPSO) algorithm". In: (2019).

[16] Aljarf A and Amin S. "Filtering and reconstruction system for gray forensic images". In: (2015).

[17] Alexander Wong, David A. Clausi, and Paul Fieguth. **Image Denoising**. URL: https://uwaterloo.ca/vision-image-processing-lab/research-demos/image-denoising#:~:text=Therefore%5C%2C%5C%20image%5C%20denoising%5C%20plays%5C%20an,is%5C%20crucial%5C%20for%5C%20strong%5C%20performance..

[18] Jun-Jie Huang and Pier Luigi Dragotti. "WINNet: Wavelet-Inspired Invertible Network for Image Denoising". In: **IEEE Transactions on Image Processing** 31 (2022), pp. 4377–4392. DOI: 10.1109/TIP.2022.3184845.

[19] Andreas Floros. "Beyond Wavelets: Wavelet-Inspired Invertible Neural Networks for Image Modelling and Approximation". MA thesis. Imperial College London, 2022.

[20] Rahul Awati. **convolutional neural network (CNN)**. URL: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network#:~:text=How%5C%20convolutional%5C%20neural%5C%20networks%5C%20work,more%5C%20detailed%5C%20after%5C%20each%5C%20layer..

[21] Krizhevsky A, Sutskever I, and Hinton GE. "Imagenet classification with deep convolutional neural networks". In: (2012).

[22] Ha I et al. "Image retrieval using BIM and features from pretrained VGG network for indoor localization". In: (2018).

[23] Tang P, Wang H, and Kwong S. "G-MS2F: GoogLeNet based multi-stage feature fusion of deep CNN for scene recognition". In: (2017).

[24] Liang J and Liu R. "Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network". In: (2015).

[25] Xu J et al. "Patch group based nonlocal self-similarity prior learning for image denoising". In: (2015).

[26] Zhang K et al. "Beyond a Gaussian denoiser: Residual learning of deep cnn for image denoising". In: (2017).

[27] Li Y et al. "Adaptive batch normalization for practical domain adaptation". In: (2018).

[28] Nair V and Hinton GE. "Rectified linear units improve restricted boltzmann machines". In: (2010).

[29] Zhang M et al. "Synthetic aperture radar image despeckling with a residual learning of convolutional neural network". In: (2021).

[30] Zhang F et al. "Random noise attenuation method for seismic data based on deep residual networks". In: (2018).

[31] Guo Z et al. "Deep residual network with sparse feedback for image restoration". In: (2018).

[32] ChunweiTian LunkeFei, WenxianZheng YX, and WangmengZuo C-W. "Deep learning on image denoising: an overview". In: (2020).

[33] Ademola E. Ilesanmi and Taiwo O. Ilesanmi. "Methods for image denoising using convolutional neural network: a review". In: 7 (2021), pp. 2179–2198.

[34] Mehreen Saeed. **An Introduction to Recurrent Neural Networks and the Math That Powers Them**. 2022. URL: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

[35] Isao Ishikawa et al. "Universal approximation property of invertible neural networks". In: (2022).

[36] K. He et al. "Deep residual learning for imagerecognition". In: (2016), pp. 770–778.

[37] B. Cai et al. "Dehazenet: An end-to-endsystem for single image haze removal". In: 26 (2016), pp. 5187–5198.

[38] Eglen Protas et al. "Understading Image Restoration Convolutional Neural Networks with Network Inversion". In: (2017).

[39] Jun-Jie Huang and Pier Luigi Dragotti. "LINN: Lifting Inspired Invertible Neural Networks for Image Denoising". In: **2021 29th European Signal Processing Conference (EUSIPCO)**. EURASIP. 2021, pp. 23–27.

[40] Junjie Huang and Pier Luigi Dragotti. **LINN: Lifting Inspired Invertible Neural Network for Image Denoising**. URL: https://www.commsp.ee.ic.ac.uk/~jhuang/Slides/LINN_EUSIPCO2021.pdf.

[41] M. Shensa. "The discrete wavelet transform: Wedding the a trous and Mallat algorithms". In: 40 (1992), pp. 2464–2482.

[42] D. L. Donoho and J. M. Johnstone. "Ideal spatial adaptation by wavelet shrinkage". In: 81 (1994), pp. 425–455.

[43] Hao Ouyang, Tengfei Wang, and Qifeng Chen. "Restorable Image Operators with Quasi-Invertible Networks". In: 36 (2022).

[44] Yang Liu et al. "Invertible Denoising Network: A Light Solution for Real Noise Removal". In: (2021).

[45] Ronneberger O., Fischer P., and Brox T. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: (2015), pp. 234–241.

[46] Etmann C., Ke R., and Schönlieb C.B. "iUNets: Learnable invertible up-and downsampling for large-scale inverse problems". In: (2020), pp. 1–6.

[47] Alexander Denker et al. "Conditional Invertible Neural Networks for Medical Imaging". In: (2021).

[48] Min Zhang et al. "Enhancing Image Rescaling Using Dual Latent Variables in Invertible Neural Network". In: (2022).

[49] Tai- Wen Yue and Guo-Tai Chen. "An Auto-Invertible Neural Network For Image Halftoning and Restoration". In: (1995).

[50] Lynton Ardizzone et al. "Conditional Invertible Neural Networks for Diverse Image-to-Image Translation". In: (2021).

[51] Lynton Ardizzone et al. "Guided Image Generation with Conditional Invertible Neural Networks". In: (2019).

[52] X. Liu, M. Tanaka, and M. Okutomi. “Single-image noise level estimation for blind denoising”. In: 22 (2013), pp. 5226–5237.

[53] Data Monsters. “A Quick Overview of Methods to Measure the Similarity Between Images”. In: (2020). URL: https://medium.com/@datamonsters/a-quick-overview-of-methods-to-measure-the-similarity-between-images-f907166694ee.

[54] Jun-Jie Huang and Pier Luigi Dragotti. “LINN: Lifting Inspired Invertible Neural Networks for Image Denoising”. In: **2021 29th European Signal Processing Conference (EUSIPCO)**. EURASIP. 2021, pp. 636–640. DOI: 10.23919/EUSIPCO54536.2021.9615931.

[55] Imperial College London. **Ethics approval overview**. URL: https://www.imperial.ac.uk/research-ethics-committee/ethics-approval-overview/.

[56] J. Xue et al. “Multilayer sparsity-based tensor decomposition for low-rank tensor completion”. In: (2021).

[57] J. Xue et al. “Enhanced sparsity prior model for low-rank tensor completion”. In: 31 (2020), pp. 4567–4581.

[58] K. Zhang et al. “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising”. In: 26 (2017), pp. 3142–3155.

[59] K. Zhang, W. Zuo, and L. Zhang. “FFDNet: Toward a fast and flexible solution for CNN-based image denoising”. In: 27 (2018), pp. 4608–4622.

[60] S. Guo et al. “Toward convolutional blind denoising of real photographs”. In: (2019), pp. 1712–1722.

[61] M. E. Helou and S. Susstrunk. “Blind universal Bayesian image denoising with Gaussian noise level learning”. In: 29 (2020), pp. 4885–4897.

[62] S. Mohan et al. “Robust and interpretable blind image denoising via bias-free convolutional neural networks”. In: (2020).

[63] S. G. Bahnemiri, M. Ponomarenko, and K. Egiazarian. “Learning-based Noise Component Map Estimation for Image Denoising”. In: (2021). URL: https://arxiv.org/abs/2109.11877.

[64] I. Daubechies and W. Sweldens. “Factoring wavelet transforms into lifting steps”. In: 4 (1998), pp. 247–269.