COMP6248 REPRODUCIBILITY CHALLENGE NOISE2SCORE

William Broster, George Everdell, Benjamin Roberts
Department of Electronics & Computer Science
University of Southampton
{wb1e21,gbae1g18,bdr1g18}@soton.ac.uk

ABSTRACT

Image denoising is the act of removing noise from a noisy image, so as to preserve maximum features and ideally restore the true image. There are already existing algorithms available which make use of traditional techniques such as spatial filtering. There is ongoing research in training deep networks in an attempt to outperform existing algorithms. Often, these models are configured for supervised learning, requiring both a noisy image and its clean variant for training. However recently there has been more focus in training self-supervised deep networks which attempt to denoise images without the clean reference. This report explores the NeurIPS 2021 paper 'Noise2Score: Tweedie's Approach to Self-Supervised Image Denoising without Clean Images' (Kim & Ye (2021)) and details its reimplementation to ultimately clarify if the results published in the paper are accurate and reproducible.

1 INTRODUCTION

1.1 IMAGE DENOISING

$$l_{AR-DAE}(\Theta) = \mathop{\mathbb{E}}_{y\sim P_Y} ||u + \sigma_a R_\Theta(y + \sigma_a u)||^2$$

For Gaussian noise they state that Tweedie's formula defines the expected denoised input image to be the following:

$$\mathbb{E}[x|y] = y + \sigma^2 l'(y)$$

1.2 BASELINE MODELS

The original paper uses the results obtained from previous denoising methods to perform quantitative and qualitative comparisons of the performance of their own model. They discuss the usage of four datasets (two greyscale and two RGB) with Gaussian, Poisson and Gamma noise applied to them, run through seven different methods of denoising. Here we discuss the difference in implementation of these methods, and describe Noise2Void (Krull et al. (2019)) as a method that we used to draw qualitative results from.

- **BM3D** Block matching and 3D filteringDabov et al. (2006) uses sliding windows over the input images and estimates a denoised approximation of the sample of the image underneath the window, then combining all of the estimates to produce a final image. This method does not use deep learning.
- **Noise2Void** Using self supervised learning without clean targets, noise2void uses patches of images to learn a mapping between input patches and the center pixel in the patch.
- **Noise2Noise** Noise2Noise is a self supervised method that uses multiple noisy versions of the same image to train a deep learning model.

2 IMPLEMENTATION OVERVIEW

Due to the complex nature of the implementation, we decided to use the existing code to try and reproduce the results in the original paper.

The first thing to do was install all the dependencies for the project. Thankfully, the authors included a file which allowed us to directly download and install all the requirements to a Conda environment. First, we tried to do this on a Windows machine, but soon learned that some packages were only available for Linux. After switching to Linux, the environment was built and all the dependencies were successfully installed.

2.1 Pre-trained Models

In the download for the code, there was some test data already included so this was run through the pre-trained model. It was now that we discovered the first issue with the code. In the test.py script there was a call to model.test() - however this function simply did not exist at all throughout the entire project. We had to write it in ourselves. However, after this the code ran perfectly fine on the Set12 dataset. This came out with some promising results. However, to confirm that the authors hadn't chosen specific noisy images to maximise the results, we generated our own noisy images from the same dataset (Set12) and ran these through the model.

Set12 is a dataset of 12 black and white images, so after this we decided to put the BSD68 dataset through the model which is also black and white. Here we found another error. It seems the model did not like the fact that the images in BSD68 had odd length dimensions. The dimensions were originally 481x321, but when debugging the code it was expecting 480x320 - so we manually had to remove the first row and column. After this the model worked fine and denoised the images in the BSD68 dataset.

The other two test datasets that were referenced in the paper were CBSD68 and Kodak, which are both colour datasets. From the limited README file there was no information on how to run a colour dataset through the model. There were many different possible options, all of which threw different errors. The main error seemed to be with input channels with the existing model expecting 1 for greyscale, but being given 3 for RGB. There was no obvious indication in the paper that models can only work on either black and white or RGB, not both.

2.2 Training Attempts

Whilst it is still good to collect data using the pre-trained models, we wanted to see if we could train the network ourselves to learn the weights of a model that would perform just as well. We were hoping to train a model that would be able to denoise colour images since a pre-trained model wasn't provided. However, when we tried to run the commands specified in the README file, errors were constantly being thrown. Functions that were trying to be called but never even defined, and even after these were commented out - there was another error about failing to start a Visdom Server for live visualisation.

We did try and work past all these errors, but it soon became apparent that there was an issue with the code provided. Some things just simply would not work. Whether this is an error in the setup that proper documentation would have solved, or whether the code was simply incomplete with the authors uploading broken code - no one would have been able to get it working in a considerable time frame without prior knowledge of the project.

3 Results

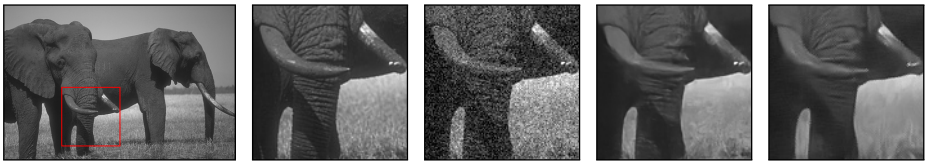
Table 1 shows a comparison of the PSNR values we calculated versus the values reported in the Noise2Score paper, with one other denoising algorithm for reference. For our reference model, Noise2Void (Krull et al. (2019)), we trained the network ourselves using code made available the authors. The model was trained on the same data sets in the Noise2Score paper for fair comparison.

Peak signal to noise ratio (PSNR) is a standard metric for evaluating image reconstructions with denoising algorithms, it a logarithmic comparison of the reconstructed image with the original clean one. Higher values of PSNR represent better reconstructions which are closer to their original images.

We parsed the BSD68 and Set 12 images through the trained models and calculated the PSNR of the reconstructed images, and the results can be seen in table 1. The average PSNR values we calculated for Noise2Score were very similar to the values reported in their paper, which is to be expected given that we used their pre-trained model to generate the reconstructed images. There were larger discrepancies with the reported PSNR values for Noise2Void, but the differences were still quite small and can be explained by random variance in training.

Noise type	Data set	Our generated values Noise2Score	Our generated values Noise2Void	Values reported in paper Noise2Score	Values reported in paper Noise2Void
Gaussian	Set 12	30.03	28.47	30.13	27.56
$\sigma = 25$	BSD68	28.65	27.21	29.12	26.77
Poisson	Set 12	31.51	28.98	31.58	30.06
$\zeta = 0.01$	BSD68	30.50	27.66	30.81	28.73
Gamma	Set 12	33.03	28.99	33.01	30.54
$\kappa = 100$	BSD68	32.34	27.65	32.67	29.32

Table 1: Comparison for various noise models using various methods in terms of PSNR (dB) with known noise parameters.



Clean image from BSD68 dataset | Clean image | Noise added to image | Noise2Score 30.30 dB | Noise2Void 26.14 dB

Figure 1: Qualitative comparison using BSD68 dataset - Gaussian ($\sigma = 25$) - Values below images represent PSNR values

4 Discussion

From the table in the previous section we can see that the biggest difference in PSNR values between our generated images, and the results in the original paper is 0.47. This is small enough to conclude that the results on the black and white datasets, for the 3 noise types, in the original paper, are reproducible.

So, the model clearly works well on black and white datasets. However, this took a while to get working and the code was not well documented. The difficulties faced obtaining our results, and the attempt to get further results are detailed below.

4.1 Possible Reimplementation

When trying to reproduce a paper's results there are two possible routes to take. One is to reimplement the project from scratch using the information in the paper. The other is to take existing code that was uploaded alongside the paper, and use that. In an ideal world, we would have liked to have reimplemented the Noise2Score project, however this would have been impossible.

Whilst the paper details the loss functions, and how the output is used with Tweedie's formula to create a denoised image - there is absolutely no indication as to the structure of the neural network used. From the code, we can see that there are multiple 2D convolutional layers, with varying configurations of channels, padding, and stride. There were also other techniques used such as dropout layers, alongside ReLU layers to bring it all together. None of this was mentioned in the paper or supplementary material.

Although, in the supplementary material there is one section of pseudo-code which gives a brief top-level description of the Noise2Score algorithm. However, this alone is not nearly enough to reimplement the code from scratch yourself. Their actual code structure is far vaster and more complex than this small section of pseudo code implies. There is no documentation on how the different python files fit together, or which of the dozens of different operation modes need to be selected for a given task. For example, training on colourised imaged requires you to call different functions in the top-level code, but this is not mentioned anywhere.

If there was considerably more time, the existing code could be fully analysed and perhaps a better picture of understanding would be gained. But the information given in terms of reimplementation is simply not enough. The paper chooses to focus more on the types of noise and the loss functions.

4.2 Poorly Documented Code

Since reimplementation wasn't going to be possible, we had to get the existing code to run. This was by no means easy either. There was one very short README file that was included with the code. This had no real guidance or instructions apart from example commands to run. To start off, we couldn't run the training commands because there were so many different errors being produced that we couldn't figure out. And then all the other commands were examples on how to use their pre-trained models. The commands that executed the model to run on the Set12 dataset all worked fine after some missing code was implemented - but running the model on colour images never worked. Other than this there was no useful information or guidance in the README file.

As mentioned in the implementation overview, there also seemed to be missing and broken code. This isn't something we expected to see from a project such as this. To get the results in the paper they must have been using slightly different code to the one they chose to upload. This has certainly reduced the amount that could be reproduced in a shorter time frame.

We also had to work out ourselves that the models did not take images in as direct input. We had to supply the NumPy arrays directly in the form of .npy files. This wasn't too difficult to work out, it's just the fact that this was never mentioned anywhere in the paper or supplementary material.

One strange issue we ran into when attempting to train the network was a missing function named 'GANLoss'. When attempting to train the model, the code would throw the error: 'object has no attribute GANLoss' (amongst others). This is very peculiar as there is no mention of any form of GAN in the paper or any of the surrounding documentation, and the function cannot be found in any of their given code.

5 Conclusion

The paper chooses to focus solely on Tweedie's formula of the posterior mean for image denoising and denoising techniques used by other similar papers. However, the equations detailed in the paper only relate to post processing which is done to the output of the network. There is no formal description of the neural network architecture used, and there is disturbingly little documentation of the code in both the paper and surrounding material. In conclusion, there is not enough information to produce a full reimplementation of this paper and the results are only partially reproducible using their pre-trained model.

6 GitHub Repository Link

https://github.com/COMP6248-Reproducability-Challenge/COMP6248-ImageDenoising

References

Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising with block-matching and 3D filtering. In Nasser M. Nasrabadi, Syed A. Rizvi, Edward R. Dougherty, Jaakko T. Astola, and Karen O. Egiazarian (eds.), *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, volume 6064, pp. 354 – 365. International Society for Optics and Photonics, SPIE, 2006. doi: 10.1117/12.643267. URL https://doi.org/10.1117/12.643267.

Kwanyoung Kim and Jong Chul Ye. Noise2score: Tweedie's approach to self-supervised image denoising without clean images. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=ZqEUs3sTRU0.

Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2129–2137, 2019.