

COMP6248 REPRODUCIBILITY CHALLENGE

HIGH-QUALITY SELF-SUPERVISED DEEP IMAGE DENOISING

David Jones, Richard Crosland & Jason Barrett

Department of Electronics & Computer Science

University of Southampton

{dsj1n15, rtc1g16, jbl7g16}@soton.ac.uk

1 INTRODUCTION

The aim of denoising is to reduce noise in a corrupted image while preserving features. Traditional techniques include median filtering and non-local means. Recently a focus has been placed on training autoencoder neural networks to learn this typically non-linear mapping. Often these models will be trained via supervised methods, requiring clean and noisy image pairs for training; when a clean target is not provided the training is considered to be self-supervised. This report details the reimplementation of the NeurIPS 2019 conference paper ‘High-Quality Self-Supervised Deep Image Denoising’ (Laine et al., 2019). It claims significant improvements to both the image quality and training efficiency over other self-supervised methods. An overview of the reimplementation process and a comparison of results against a subset of those seen in the original paper follow.

2 BACKGROUND

2.1 SELF-SUPERVISED DEEP IMAGE DENOISING (SSDN)

This paper introduces a network architecture that incorporates a blindspot into the receptive field of the convolution and down-sampling layers. This has a resultant effect of the central pixel not being considered as part of the loss function so that the autoencoder does not learn the identity when the target is the same as the input; thus permitting self-supervised learning. This follows on from the concepts of Noise2Void (Krull et al., 2018), which uses a mask to replace the central pixel with a different value. At evaluation time it is deemed that the central pixel likely carries relevant information, and is therefore incorporated back into the cleaned image via Bayes’ rule, as in the following equation:

$$p(x|y, \Omega_y) \propto p(y|x)p(x|\Omega_y)$$

where x is the clean value of pixel being analysed, y the noisy value, and Ω_y the noisy context. The model learns the value of $p(x|\Omega_y)$, and the noise model is either given as a parameter to training, or learnt via a second, internal model. The mean value of this inference can then be used, as it minimises the MSE loss, and as such maximises the PSNR.

2.2 BASELINES

To supplement results, implementations of baseline models were required for comparison. The following gives a brief overview of the baselines used in the original paper which were also reimplemented:

- **CBM3D** (Dabov et al., 2007) is the colour variant of the block-matching and 3D filtering algorithm used for noise reduction. This is a state-of-the-art non-neural network approach which does not require training.
- **Noise2Clean** (N2C), as named in the original paper, is an autoencoder that uses clean reference images in training alongside standard MSE; this can generally be considered the best-case scenario for training-based approaches.
- **Noise2Noise** (N2N) (Lehtinen et al., 2018) is an approach to image denoising that trains using two different noisy versions of the same image. This removes the need for clean references but still has limited applications.

- **Noise2Void** (N2V) (Krull et al., 2018) is a self-supervised training method that introduces a blindspot via a masking scheme. This involves selecting a certain percentage of pixels in an image patch as centre pixels for sub-patches. From each sub-patch, a random pixel is selected to replace the centre pixel. The training loss is calculated using this mask rather than the whole image.

3 IMPLEMENTATION OVERVIEW

The original paper, for which the source code was available¹, was implemented in Tensorflow. The reimplementations² instead used PyTorch. The original paper’s supplement details the network structure, defining a network based on U-Net (Ronneberger et al., 2015) with additional layers for handling creation and collation of multiple rotated views of the input, as well as modification to layers to handle upward shifts. These modifications define the blindspot in the network. The network was implemented as per the specification. Where details were not specified, such as the upsampling method, the original source was referred to (it using nearest-neighbour). All other baseline networks used the same U-Net architecture without blindspot additions. The differences between methods, other than the network used, were the inputs/targets and loss calculation pipeline. Data preparation involved adding synthetic noise and padding to shapes accepted by the networks.

The paper itself details the loss function and integration of the prior. For SSDN this involved prior calculation using either the known σ , learning of a constant σ using a single learnable parameter, or a variable σ using a separate U-Net. When the prior is not incorporated this is referred to as SSDN- μ . Other features implemented and used in testing, but not discussed in this report, include support for Poisson noise, diagonal covariance matrices, and single channel inputs. Impulse noise was not implemented.

Initialisation of weights was handled as directed using He et al. (2015). Initial testing showed when using parameter estimation, the model failed to converge; this was traced in the original source to the last layer using zeroed weights in the parameter estimation network; this was not mentioned in either the paper or supplement. Training conditions were mimicked using the Adam optimiser with default parameters except the learning rate, which was set every iteration following a cosine ramp-down, ramp-up. It should be noted that training used an iteration based approach (not epoch based), where each iteration is a random cropped patch of a randomly sampled image from the training dataset (without replacement). The dataset would then reset once all images are used. The suggested minibatch-size of 4 was used with this batch split across GPUs if available. Tensorboard and checkpointing were implemented alongside the implementation to aid tracking and allow pausing/resuming of long training runs.

The main implementation issue that occurred was a misunderstanding of how to treat noise addition. The paper expected that synthetic noise is not clipped after addition; this leads to values outside the standard `uint8` boundary (< 0 and > 255). Initial reimplementations clipped these values to closer represent real-world scenarios, this caused performance drops with a known parameter and with parameter estimation to fail completely. These results are attributed to clipping causing truncated noise addition, which would require different prior calculations – performance for this is not known. Establishing this issue required fine inspection of the original source and was not clear in either the supplement or paper.

4 RESULTS

Due to limited compute infrastructure it was decided to limit result reproduction to a single noise type and level – Gaussian ($\sigma = 25$); this is one of the main benchmarks provided in the original paper. The original paper typically used 2,000,000 iterations per training, however, due to slower hardware and a slightly slower implementation this would take approximately 7 days per training (versus 16 hours). Therefore the number of iterations used for each model was 200,000 unless stated. Outputs from the models are evaluated by their peak signal-to-noise ratio (PSNR), calculated between the clean (pre-noised) image and the cleaned noisy image. This metric is commonly used

¹Tensorflow Source, <https://github.com/NVlabs/selfsupervised-denoising>

²PyTorch Source, <https://github.com/COMP6248-Reproducibility-Challenge/selfsupervised-denoising>

to quantify the extent to which an image has been restored after removing noise with higher values suggesting less noise. The training curves for a subset of the models are shown in Figure 1. Table 1 show calculated metrics for SSDN and all neural-net baselines, with data for CBM3D sourced from Laine et al. (2019). Additional verification of the parameter estimation aspects of the network was performed using a variable σ network configuration trained on images with σ between 25 and 50. This was trained for 844k iterations and achieved a PSNR of 28.89dB on the BSD300 dataset.

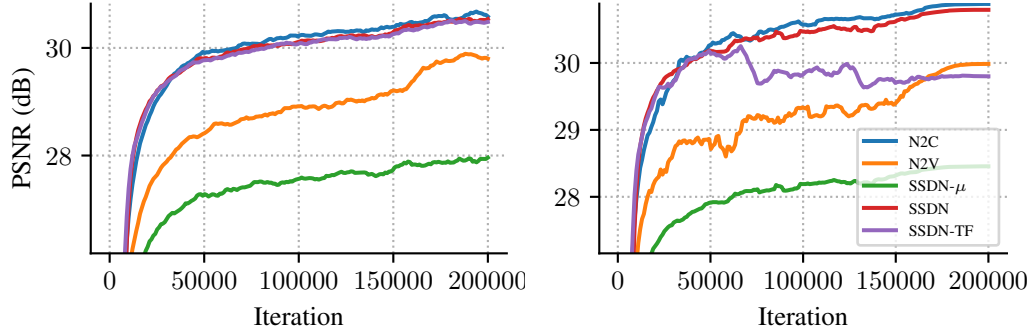


Figure 1: Training and validation curves comparing PSNRs [5th-100th percentile] of implementation versus baselines. Gaussian noise ($\sigma = 25$). Training (ImageNet-Validation) [left], Validation (BSD300-Test) [right]. N.B. Variable σ estimation used for SSDN.

Table 1: Validation PSNRs for denoising images from both the Kodak and BSD300 dataset with Gaussian noise ($\sigma = 25$). See Figure 2 for example image comparison.

Method	σ known?	KODAK	BSD300	Average
CBM3D (Untrained Baseline)	no	31.81	30.40	31.11
N2C (Baseline)	no	32.19	31.22	31.71
N2N (Baseline)	no	32.16	31.20	31.68
N2V (Baseline)	no	31.03	30.24	30.64
SSDN (μ only)	no	30.00	28.56	29.28
SSDN	no	31.61	30.55	31.08
SSDN	yes	32.12	31.13	31.63

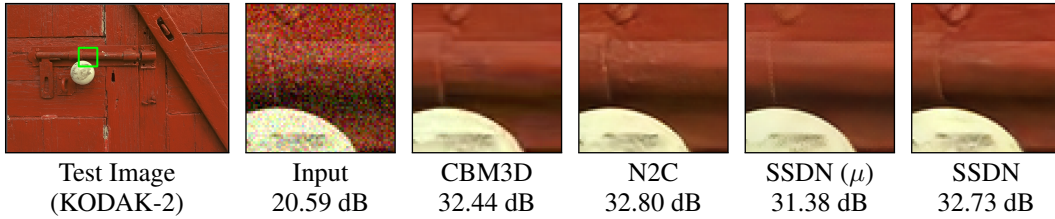


Figure 2: Results of applying SSDN and Baselines to Gaussian ($\sigma = 25$ noise [σ known])

5 DISCUSSION

Despite constraints in breadth of models trained and training durations, a range of results were reproduced from the original paper with relative comparisons against the baselines showing the expected trends. Results in Table 1 correspond to the results shown in Table 1 of the original paper, covering validation performance of: CBM3D, N2C, N2N and different configurations of SSDN. A major claim is that SSDN's training performance matches that of N2C; this was reproducible even when σ was not known as can be seen in Figure 1. Likewise, visual checks using Figure 2

show very little detail difference between N2C and SSDN. Validation performance with σ known also manages to exceed that of CBM3D as expected. In validation, SSDN is expected to achieve an average PSNR of 31.73dB (on BSD300 and KODAK); with a known sigma across the same datasets; the reproduced results show a similar 31.63dB average PSNR, only 0.1dB less with one tenth of the training iterations. Since the number of training iterations in these experiments is lower than that used in the original paper, it is not possible to verify the performance of the fully trained network, however the trend of all plots indicate that further training would keep increasing PSNR for all models. One discrepancy noted between the paper and its results, as well as the reproduced results, is a claim made in the abstract that the methods introduced improve images quality, however they are only ever seen to at best match the state-of-the-art results, rather than exceed them.

One key claim of this paper is that it achieves improved training efficiency compared to state-of-the-art self-supervised denoising methods. The original paper suggests training the N2V baseline by maintaining a smoothed network created from an exponential moving average of previous weights; this is not suggested by Krull et al. (2018). The reproduced baseline results (Figure 1), which do not do this, indicate that the implementation of SSDN does indeed learn at a higher rate (PSNR achieved in same iterations) than N2V. However, it should be noted that the training time per iteration for SSDN was $4\times$ that of N2V due to loss calculation and the $4\times$ larger network, it is therefore unclear if real-world training performance is improved.

To verify the implementation, training performance was captured in the same configuration using the original Tensorflow implementation (SSDN-TF in Figure 1). Although training PSNRs tracked almost exactly, the validation PSNRs from the Tensorflow implementation decreased significantly after approximately 65,000 iterations (this was repeatable for default seed setups). It is not clear why this occurred but may be a quirk in training or highlight a potential issue in their original implementation that was not carried to the PyTorch implementation.

6 CONCLUSION

When reading the paper, the reader is provided with a detailed account of the theory behind the workings of the paper, as well as justifications for the changes made to the previous solutions to the problem. The provision of the supplement also gave more specific details about the model itself that would not be crucial to the theory, namely details about testing, and the architecture of the network used. The combination of paper and supplement alone was not enough to fully reproduce the results. The authors' code was also required, shedding light on the clipping issue discussed at the end of Section 3. In conclusion, fully utilising a combination of sources allowed for a successful reimplement and reproduction of results.

REFERENCES

- Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. volume 1, pp. I – 313, 09 2007. ISBN 978-1-4244-1437-6. doi: 10.1109/ICIP.2007.4378954.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.
- Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void - learning denoising from single noisy images. *CoRR*, abs/1811.10980, 2018. URL <http://arxiv.org/abs/1811.10980>.
- Samuli Laine, Jaakko Lehtinen, and Timo Aila. Self-supervised deep image denoising. *CoRR*, abs/1901.10277, 2019. URL <http://arxiv.org/abs/1901.10277>.
- Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *CoRR*, abs/1803.04189, 2018. URL <http://arxiv.org/abs/1803.04189>.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL <http://arxiv.org/abs/1505.04597>.