A Study of Image Denoising Methods through U-net Based Machine Learning Design

Joseph Quan

Abstract

There have been significant advances in the field of image denoising using machine learning techniques. In state-of-the-art methods, system complexity and system performance have been steadily improving over the years. Still, there is additional work to simplify the networks while keeping high performance. In this paper, we introduce several image denoising techniques, including those state-of-the-art methods with system complexity and the simplified ones. We did experiments using NAFNet, a deep learning denoising model that made the simplification through avoiding nonlinear activation functions. We found that by refining the dataset and introducing new training images, the quality of the results could be substantially improved. Furthermore, we experimented with different model structures and found that we can reduce model complexity substantially, while model effectiveness does not diminish much.

Introduction and Related Work

Image denoising is a fundamental task in image processing and computer vision. The goal of denoising is to recover a clean image from a noisy observation. Reliable denoising is crucial not only for aesthetically pleasing photographs, but also for numerous applications, including scientific imaging, remote sensing, and surveillance [1].

In imaging, noise is uncertainty added to an image during capture and processing. It shows up as random fluctuations in brightness or color that are not part of the actual scene [2]. Noise originates from both the physics of light detection and the electronics of the camera. Many different forms of noise exist, including shot noise, fixed pattern noise, and photoresponse non-uniformity noise [3].

Before the rise of machine learning-based methods, a variety of techniques were developed for noise reduction. Early approaches include spatial filtering, such as Gaussian and mean filters [4]. These methods denoise by smoothing pixel intensities, but often blur important edges and fine details [5]. Later, more advanced techniques, such as frequency-based denoising, notably wavelet thresholding [6], which preserves important signal frequencies and removes noisy signal frequencies, were developed. Another influential method, non-local means [7], performs more adaptive denoising by comparing pixels to the entire image, rather than a local subset of the image. These classical approaches remain useful for their simplicity and efficiency, but often struggle with complex noise patterns or retaining important details.

The transition from classical models to machine learning models in denoising marks a major shift in the field. With machine learning methods, especially deep learning methods, being developed, image denoising methods have advanced considerably. Convolutional neural networks (CNNs), at first, became the dominant model, such as MemNet [8]. CNNs learn mappings from noisy to clean images directly from data.

UNet-based architectures [9] use an encoder-decoder structure with skip connections. The encoder downsamples the image to gather high-level information, while the decoder upsamples to capture spatial details. The skip connections link the encoder and decoder layers, allowing the network to combine the global context with local details. There is an improvement that better allows UNet-based architectures to preserve details over CNN-based approaches. More recently, transformer-based methods, such as Restormer [10], have extended denoising by using self-attention, which further boosts performance by using long-range dependencies.

Restormer [10] is an efficient transformer model that overcomes the challenges CNNs have, namely their limited receptive field not being able to capture broad, contextual information and inability to flexibly adapt to input content. Restormer employs a Multi-DConv Head Transposed Self-Attention (MDTA), which applies self-attention across channels rather than spatial dimension. This enables Restormer to model long-range dependencies while also preserving computational efficiency. In addition, Restormer's Gated-DConv Feed-Forward Network (GDFN) focuses on enriching features with contextual information. This allows useful information to propagate further. Restormer uses a UNet-style encoder-decoder framework with skip connections and effectively integrates global context with local details.

NAFNet [11] is a more recent contribution that removes nonlinear activation functions, aiming to reduce both inter and intra-block complexity. Inter-block complexity refers to complexity that arises based on how the blocks are connected and stacked, while intra-block complexity refers to complexity coming from within each block itself. Low inter-block complexity is achieved by adopting a classic single-stage U-shaped architecture with skip connections. Low intra-block complexity is achieved by modifying the overall structure of each block.

Many previous deep learning methods have relied extensively on nonlinear activation functions, but NAFNet replaces those with linear layers, gating mechanisms, and simplified residual connections. Each block from NAFNet combines the structure of Restormer's block [10] and PlainNet's block, which combines the most common components, while also using SimpleGate structures in place of nonlinear activation functions, like ReLU and GELU. SimpleGate structures are simple element-wise gate structures, or multiplication of feature maps. NAFNet also simplifies channel attention from their baseline model. This design reduces complexity, yet still retains state-of-the-art performance in image restoration tasks, including denoising.

Methodology

NAFNet was first trained on the SIDD dataset [14]. NAFNet comes with two settings: width 64 and width 32, which indicates the base number of feature channels used throughout the network. From experiments, we found that a higher width will have higher complexity but could obtain better quality. A high width will increase the training time and memory consumed. Our machine had memory limitations in running complexities higher than width 64, such as width 96 or width 128, so we used width 64 for our following experiments.

In our experiments, the parameters were tuned such that the total number of iterations was set as 400,000, the initial learning rate was 0.001, with the optimizer used as AdamW [12]. AdamW is similar to the standard Adam optimizer [13], but has different weight decay methods. The

Adam optimizer [13] adapts learning rates by combining momentum, which comes from past gradients, with adaptive learning rates. The loss function was a combination of the pixel loss and perceptual loss. Pixel loss refers to the difference between the restored image and ground truth image, while perceptual loss compares high-level feature representations of the restored image and ground truth, rather than raw pixels.

After obtaining the results for the base model, we found that the model could process the general noisy images very well. But when we tested some even darker and noisier images, we found that the model could not clean the noise thoroughly, with still a lot of noise left. Because of that, we added additional data from images taken from a Raspberry Pi to the training dataset [15]. We chose this dataset because the images that came from that dataset were noisier than that of the SIDD dataset.

Based on that, we compared the results between the base model and the model with additional training data. We could see that the results from the model trained with additional dataset could improve the denoising quality through removing more noise and making the image cleaner.

While the trained model achieves good performance on noisy images, we want to see if there is room to further simplify the model. The model that we tested above has 4 encoder layers, 4 decoder layers and 1 middle layer, and a total of 36 blocks. Each of the layers has a different number of blocks. We tried training a smaller size model with 18 total blocks and another even smaller size model with only 9 total blocks. We also increased the model size to have 54 total blocks to see whether increasing model size enhances the performance. The comparison of the results from different sizes of models will be shown in Experiments.

Experiments

In our experiment, we tried both less noisy and more noisy images to test the models. The original images were dark and were boosted through applying digital gains on the images to make them brighter. The example images are shown in Figure 1.

Figure 1: Sample noisy images.

First, we used the NAFNet model trained with the SIDD dataset. As shown in the examples in Figure 2, the model reduces noise effectively while still recovering the details in the noisy input images. This means the model has good performance on general noisy images.

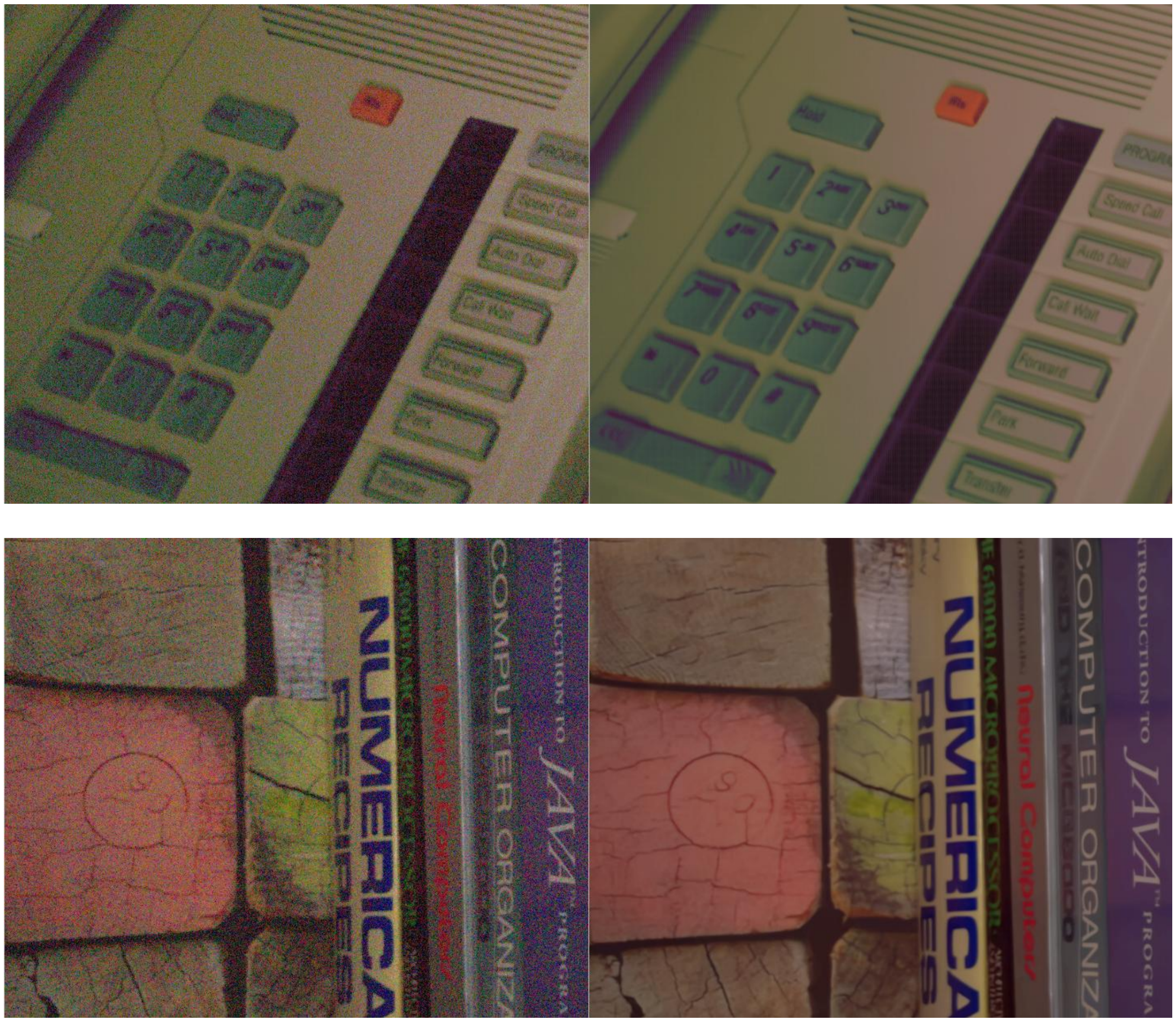
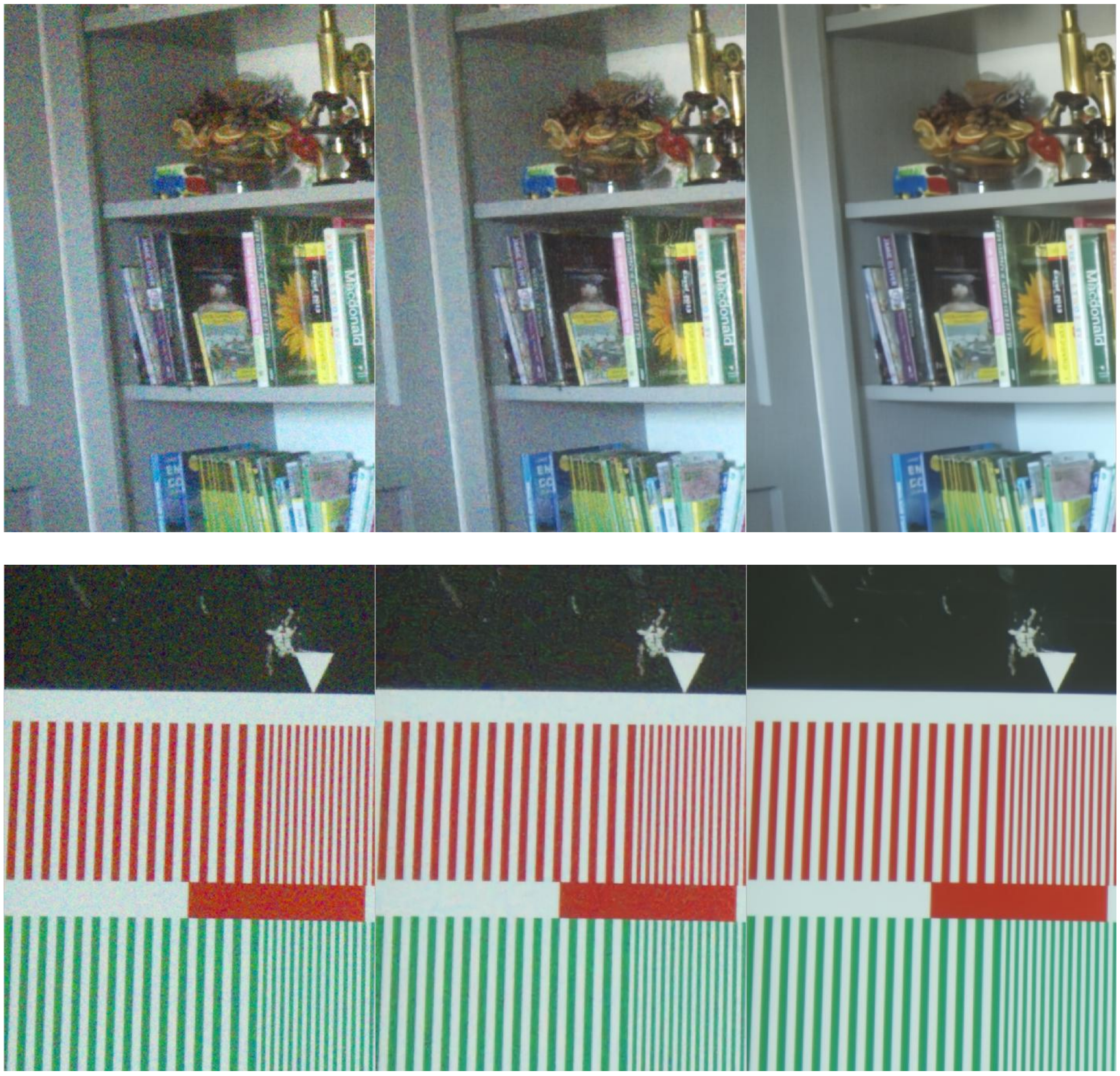


Figure 2: (a) noisy input images. (b) denoised image with NAFNet model trained with SIDD dataset.

When we tried the model on the even noisier images, the model trained with the SIDD dataset did not denoise effectively enough. In the examples in Figure 3, column (a) is the noisier input images and column (b) is the results from NAFNet trained with SIDD. We see that much of the noise still remains. When we use the model trained with the additional Raspberry Pi dataset, the results are in column (c). We see that the noise is much more reduced and the details are also very well kept.



(a) (b) (c)

Figure 3: (a) Noisier image (b) denoised image from NAFNet trained with SIDD dataset. (c) denoised image from NAFNet trained with SIDD and PI dataset.

In addition, we tried to reduce the model size through modifying the number of blocks inside the encoder, decoder and middle layers to see whether the model could be further simplified without significantly influencing the quality. The original model that we tested above is shown in Figure 4(a) with 36 total blocks. We tried smaller models as shown in Figure 4(b) with 18 total blocks and Figure 4(c) with only 9 total blocks. We also expanded the model to 5 layers encoder and decoder with 54 total blocks, as shown Figure 4(d).

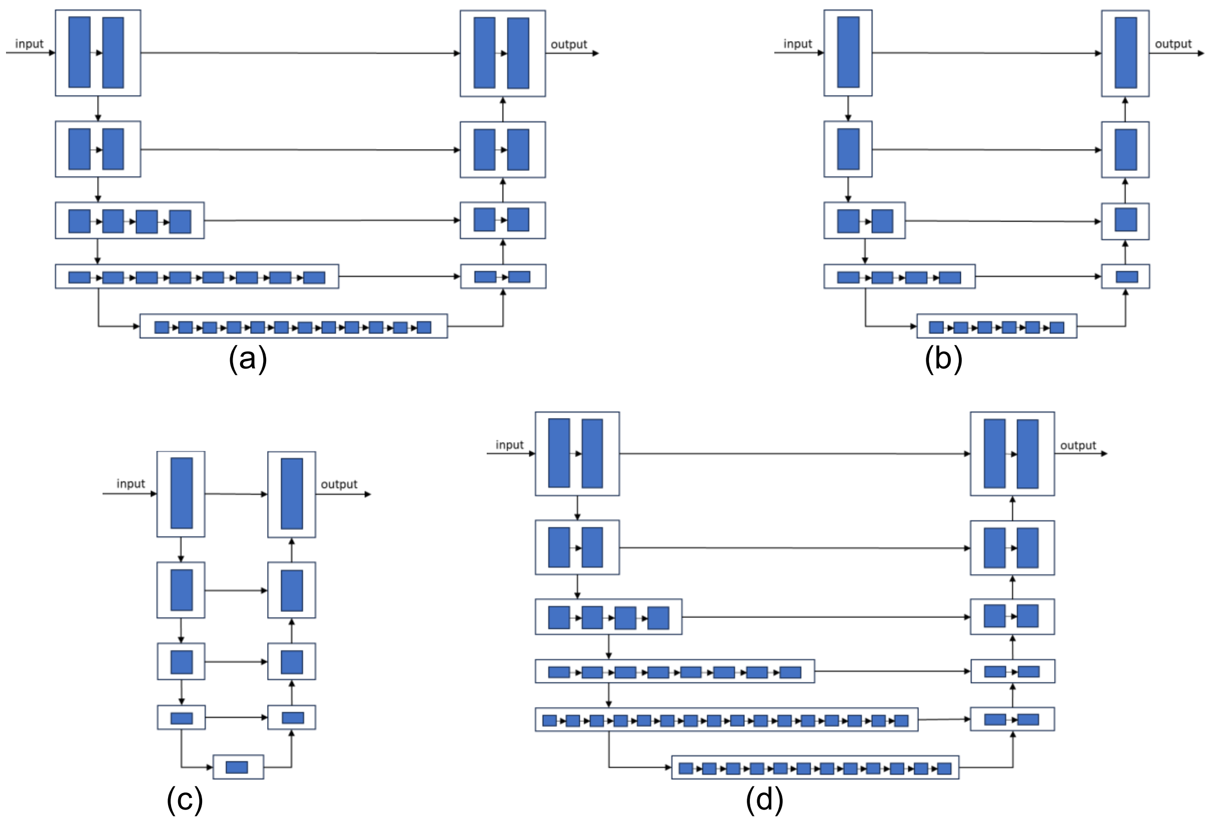


Figure 4: Model diagrams with different block combinations. (a) original model that was used for the above experiments. (b) reduced model with half size of original. (c) further reduced model with only 9 blocks. (d) expanded model from original with 5 layers encoder and decoder and 54 blocks.

We tried different models of different sizes, their corresponding Number of parameters and the GMAC values are shown in Table 1. Table 1 also includes the performance measurement of each model using PSNR and SSIM matrices.

Table 1: Model size and performance of different models as shown in Figure 4.

Model	Diagram	Number of Blocks	No. of parameters (Million)	MACs (G)	PSNR	SSIM
1	Fig. 4(c)	9	17.8	19.47	39.6104	0.9578
2	Fig. 4(b)	18	60.7	34.09	39.7226	0.9584
3	Fig. 4(a)	36	115.9	63.64	39.8987	0.9593
4	Fig. 4(d)	54	530.1	93.85	39.7632	0.9587

From the PSNR and SSIM results shown in Table 1, we can see that smaller models also achieve good performances. Model 1 and Model 2 have a mild PSNR and SSIM drop from Model 3, but small amounts. Also, from the PSNR and SSIM values of Model 4, we can see that increasing the model size may not help the performance while the number of parameters and MACs number increase.

In order to check the visual quality of the results, we used the different models to test the real noisy images to evaluate the visual quality differences between the models of different sizes. Figure 5 and 6 show the comparison of the denoising results. In Figure 5, (a) is the input image. From (b) to (e) are the results from different models with the size from small to big. We can see that all the models were effective at denoising, removing noise and keeping details. The differences are not obvious.

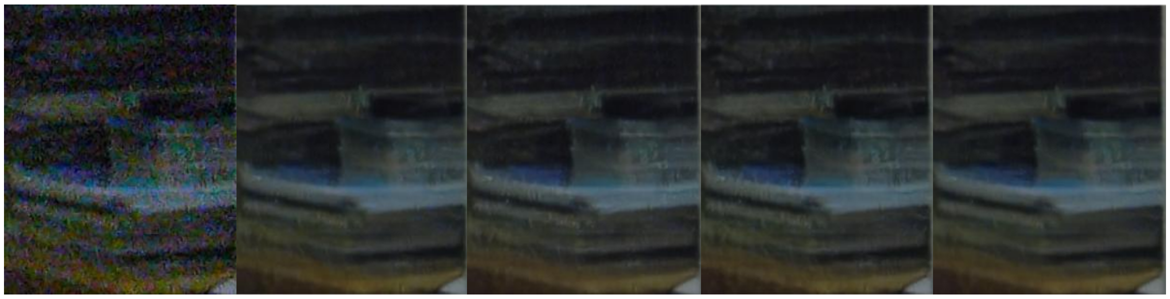


Figure 5: Comparison of denoising results from 4 different models. (a) input noisy image. (b) result from Model 1. (c) result from Model 2. (d) result from Model 3. (e) result from Model 4.

Later we tested a noisy image with weak textures as shown in Figure 6. Figure 6 (a) is the input noisy image. (b) to (e) shows the results of the models with the size from small to large. We can see that all the models did reasonable denoising. However, Model 1 and Model 2 have less denoising effects compared to Model 3 and 4. The Model 1 result has less details compared to Model 2. Model 3 and 4 have better denoising compared to Model 1 and 2, and Model 4 could keep even more details compared to Model 3. From the experiment, we can see that models with larger size do have better performance compared to models with smaller size, although all of them have reasonable qualities. Thus, we can infer that there is a trade off between the model size and model quality.

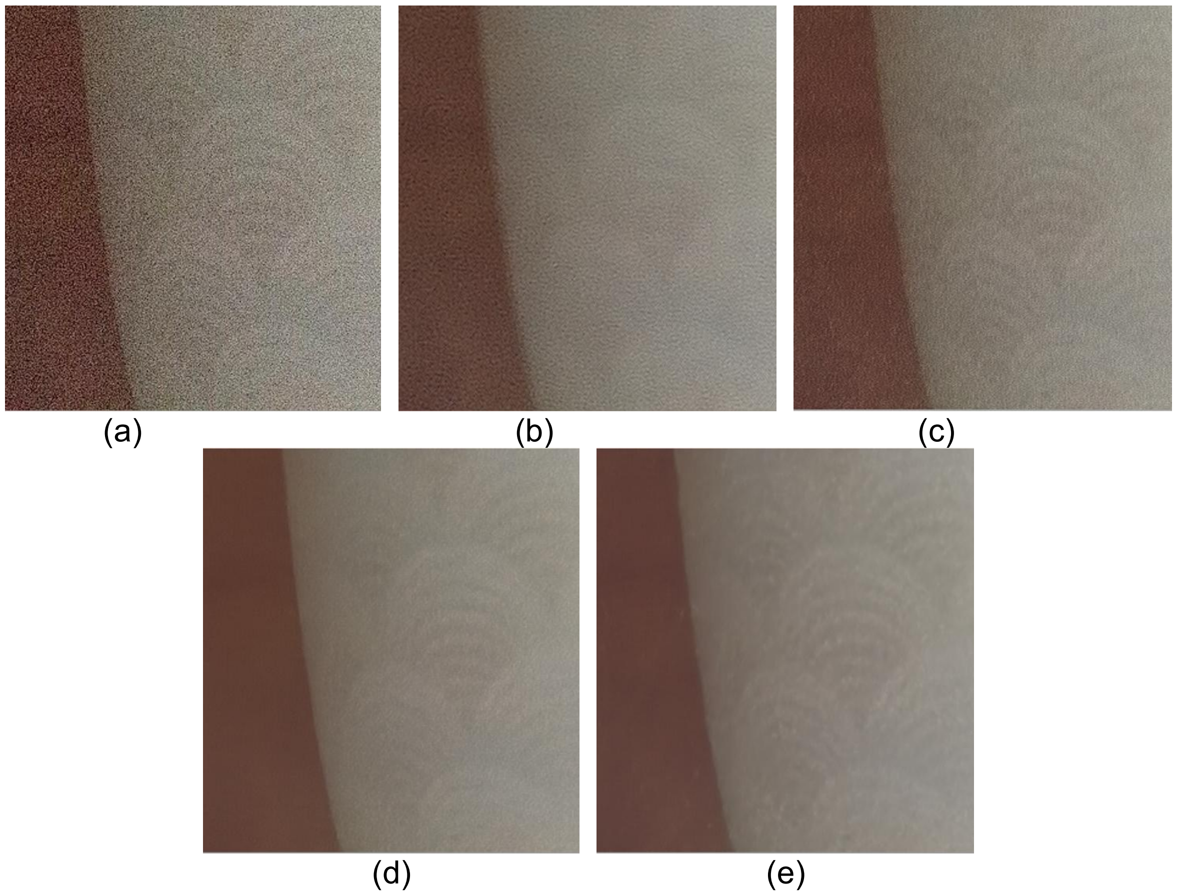


Figure 6: Comparison of denoising results of image with weak texture. (a) input noisy image. (b) result from Model 1. (c) result from Model 2. (d) result from Model 3. (e) result from Model 4.

Discussion and Conclusion

In this paper, we studied machine learning methods, especially UNet-based image denoising methods. We selected a state-of-the-art method, NAFNet, to evaluate the denoising quality. The original NAFNet model has some quality limitations on images with high noise levels. Through adding more noisy image dataset into the training, we see great improvement on denoising quality as shown in Figure 3. The original NAFNet model shows good denoising efficiency, but residual noise still exists throughout the image. The model when trained with additional Raspberry Pi data gets rid of most of the noise present in the images while also preserving edges and retaining clarity. The primary reason is that the original model dataset did not have many sufficiently noisy images. As we hypothesized, including additional noisier images in the training data, as well as a greater variety of images, improves denoising performance.

Furthermore, we did experiments on different modifications of the model, from a very small size to an expanded size. We found that by simplifying the NAFNet model, the overall effectiveness, in most cases, stays intact. This is evident when the PSNR and SSIM values of the simplified models did not significantly decrease, as well as when looking at the example images qualitatively. Nevertheless, there are still instances, such as in Figure 6, where residual noise still exists in the smaller models, and the models with bigger size have better denoising and detail keeping capabilities.

Through the study of different models, we found that even a very small size model could obtain good denoising quality. This is useful when potential future implementation of smaller models into smaller devices such as mobile phones, where latency and power consumption could be limiting factors.

In the future, we will study more about how to further improve the quality through small size models through more optimizations in order to achieve both high quality and low power consumption and latency. We will also study other image restoration areas, such as defogging and debluring; those have important applications in subjects such as autonomous driving, human-machine interaction, and artificial intelligence.

Bibliography

[1] L. Fan, F. Zhang, H. Fan, and C. Zhang, “Brief review of image denoising techniques,” *Visual Computing for Industry, Biomedicine, and Art*, vol. 2, no. 1, Jul. 2019, doi: 10.1186/s42492-019-0016-7.

[2] N. Wani and K. Raza, “Multiple kernel-learning approach for medical image analysis,” in *Soft Computing Based Medical Image Analysis*, pp. 31–47, 2018, doi: 10.1016/b978-0-12-813087-2.00002-6.

[3] R. Verma and J. Ali, “A comparative study of various types of image noise and efficient noise removal techniques,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, pp. 617–622, 2013.

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2006.

[5] Z. Al-Ameen, S. Al Ameen, and G. Sulong, “Latest methods of image enhancement and restoration for computed tomography: a concise review,” *Appl. Med. Inf.*, vol. 36, no. 1, pp. 1–12, 2015.

[6] J. H. Hou, “Research on image denoising approach based on wavelet and its statistical characteristics,” Ph.D. dissertation, Huazhong Univ. Sci. Technol., Wuhan, China, 2007.

[7] A. Buades, B. Coll, and J. M. Morel, “A non-local algorithm for image denoising,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, San Diego, CA, USA, 2005, pp. 60–65, doi: 10.1109/CVPR.2005.38.

[8] Y. Tai, J. Yang, X. Liu, and C. Xu, “MemNet: A persistent memory network for image restoration,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4549–4557, doi: 10.1109/ICCV.2017.486.

[9] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI)*, 2015, pp. 234–241.

[10] S. W. Zamir, A. Arora, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, “Restormer: Efficient transformer for high-resolution image restoration,” *arXiv preprint arXiv:2111.09881*, 2021.

[11] L. Chen, X. Chu, X. Zhang, and J. Sun, “Simple baselines for image restoration,” in *Lecture Notes in Computer Science*, 2022, pp. 17–33, doi: 10.1007/978-3-031-20071-7_2.

[12] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 6–9, 2019.

[13] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 7–9, 2015.

[14] A. Abdelhamed, S. Lin, and M. S. Brown, “A high-quality denoising dataset for smartphone cameras,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018.

[15] D. Plowman, “AI denoise training for RGB images.” Accessed: Sep. 14, 2025. [Online]. Available: https://github.com/davidplowman/denoise-rgb