[RE]Variational Denoising Network: Toward Blind Noise Modeling and Removal

Abgeiba Isunza Navarro **Walid A Jalil** **Pratima Rao A.**

1 Introduction

Image denoising is a challenging research subject that has seen a lot of progress in recent years [9], especially with the advent of convolutional neural networks(CNNs) and their more recent improvements utilizing residual learning and batch normalization. In its essence, denoising is the process of removing distortions from an image and recovering a clean one. Historically, most denoising methods have been model-driven approaches coming from a Bayesian perspective. These models, however, may suffer from faulty prior assumptions, such as Additive White Gaussian Noise (AWGN) which is a term for i.i.d. Gaussian noise. In real world images, distortions can come from a variety of sources, such as cameras, data transmissions, etc. These can be, and often are, non-i.i.d.

With the advent of deep neural networks, denoising research has focused on data-driven approaches, such as the Denoising Convolutional Neural Network (DnCNN). These data-driven methods tend to perform well with large amounts of data but suffer from two drawbacks: They can easily overfit to specific noise types, leading to poor generalisation to different types of noises, and they suffer from lack interpretability.

To this end, (Yue, Yong, Zhao, Zhang, & Meng, 2019), propose a Variational Denoising Network (VDN). The VDN is a variational inference method that is capable of directly infering both the underlying clean image as well as the noise distribution by combining a UNet [10] and a Denoising CNN [13] within a Bayesian framework. Thus, the VDN is an attempt to combine the strengths of the model-driven and the data-driven approaches.

In this report we try to reproduce the contents of the paper by implementing the network architecture, running the experiments and analyzing the results. Furthermore, we also try to conduct some additional experiments with changes to the architecture as well as provide insight and help in how to reproduce the paper to the reader. The repository with the code used on this project can be found at: `https://github.com/AbIsuNav/DD2412_project`.

2 Background

2.1 DnCNN

A Denoising Convolutional Neural Network (DnCNN), first proposed in 2016 [13], is a model for learning image denoising by treating it as a completely discriminative problem. In this method, batch normalization and residual learning [6] are utilised for better denoising and for faster training. Unlike the classical *ResNet*, the one implemented for the DnCNN makes use of just one residual unit to predict the residual images, which are the corresponding noise values.

A key feature of this model is it's ability to perform well under blind Gaussian noise, and not just additive white Gaussian noise (AGWN).

2.2 UNet

UNet was first introduced in 2015 [10] as a fully convolutional network for medical image segmentation. The network is comprised of two phases: a contracting phase that extracts the context of the image, and an expansive phase that propagates the necessary information to higher resolutions, thereby allowing for precise localisation, which are connected by a "bottleneck". In its properties, the *U-Net* is similar to an autoencoder, but additionally makes use of skip connections between corresponding encoder and decoder units allowing for features representations to pass through the "bottleneck".

2.3 Variational Denoising Network - VDN

As stated in the introduction, the VDN [12] is a variational inference method aimed at directly inferring the underlying clean image and noise distribution within a Bayesian framework. More specifically, an approximate posterior is constructed by taking clean images and noise variances as latent variables conditioned on a noisy image:

$$q(\mathbf{z}, \boldsymbol{\sigma}^2|\mathbf{y}), \tag{1}$$

where $\boldsymbol{z}$ and $\boldsymbol{\sigma}^2$ are the latent variables of the clean images and noise variances respectively, and $\mathbf{y}$ is the noisy image.

2.3.1 Contructing the Bayesian Model

For an image pair $\{\mathbf{y}, \mathbf{x}\}$ in dataset $\mathcal{D}$, we construct the model of the noisy image generation as a pixel-wise Gaussian distribution:

$$y_i \sim \mathcal{N}(y_i|z_i, \sigma_i^2). \tag{2}$$

In order to complete the Bayesian model, conjugate priors are introduced over the parameters of (2). z_i is modelled as a Gaussian with the ground truth image x_i providing a strong prior. Sometimes the ground truth image x_i is a simulated clean image and therefore not always equal to the latent clean image z_i:

$$z_i \sim \mathcal{N}(z_i|x_i, \epsilon_0^2) \tag{3}$$

where ϵ_0^2 is a hyperparameter.

Likewise, the noise variance is modelled as Inverse-Gamma distributed. with

$$\sigma_i^2 \sim \mathrm{IG}\left(\sigma_i^2|\frac{p^2}{2} - 1, \frac{p^2\xi_i}{2}\right), i = 1, 2, \cdots, d \tag{4}$$

Where ξ_i is the filtering output of a Gaussian filter with a p x p window. The combination of these gives us a parametric form of the posterior $p(\mathbf{z}, \boldsymbol{\sigma}^2|\mathbf{y})$

A variational distribution $q(\mathbf{z}, \boldsymbol{\sigma}^2|\mathbf{y})$, can be approximated to the true distribution $p(\mathbf{z}, \boldsymbol{\sigma}^2|\mathbf{y})$ by minimizing the KL-Divergence between the two distributions:

$$\log p\left(\boldsymbol{y}; \boldsymbol{z}, \boldsymbol{\sigma}^2\right) = \mathcal{L}\left(\boldsymbol{z}, \boldsymbol{\sigma}^2; \boldsymbol{y}\right) + D_{KL}\left(q\left(\boldsymbol{z}, \boldsymbol{\sigma}^2|\boldsymbol{y}\right) \| p\left(\boldsymbol{z}, \boldsymbol{\sigma}^2|\boldsymbol{y}\right)\right) \tag{5}$$

Instead of minimizing the D_{KL} the Evidence Lower Bound (ELBO) can be maximized. By assuming conditional independence between $\boldsymbol{z}$ and $\boldsymbol{\sigma}^2$, we get the ELBO term in eq (6):

$$\mathcal{L}\left(z, \sigma^2; y\right) = E_{q(z,\sigma^2|y)}\left[\log p\left(y|z, \sigma^2\right)\right] - D_{KL}(q(z|y)\|p(z)) - D_{KL}\left(q\left(\sigma^2|y\right) \| p\left(\sigma^2\right)\right) \tag{6}$$

Since the distributions for all three terms in eq (6) are known, the expectations can be calculated analytically eqs (7)-(9) and no reparametrizaton trick [7] needed. Instead, we combine the two neural networks mentioned in sections 2.1 and 2.2, the UNet and the DnCNN, and have them output the parameters $\boldsymbol{\mu}$, $\boldsymbol{m}^2$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of the conditionally independent variational posteriors $q(\mathbf{z}|\mathbf{y})$ and $q(\boldsymbol{\sigma}^2|\mathbf{y})$ respectively. A more intuitive explanation can be seen in Figure 1.

In Figure 1, the UNet is the network doing the denoising, and is therefore called D-Net. The DnCNN is the network that estimates the noise distribution and is called S-Net. By minimizing the objective

in eq (10) with respect to the network parameters W_D, W_S, the S-Net and the D-Net are guided to learn from each other, as can be seen in the log likelihood term. During backpropagation, the information from the likelihood gradient is used to update both networks. Whilst the loglikelihood term will try to fit the data, the other terms will prevent the network from straying too far from the priors, thus they constrain the parameter space and regularize the network. When ϵ_0^2 is small, it can be approximated as a dirac delta $\delta(x)$, and thus the objective $\mathcal{L}$ degenerates into an MSE loss function.

The denoising result is obtained from $\boldsymbol{\mu}$ and the noise variance is obtained from the mode of the Inverse-Gamma distribution.

$$E_{q(z,\sigma^2|y)}\left[\log p\left(y|z,\sigma^2\right)\right] = \sum_{i=1}^{d}\left\{-\frac{1}{2}\log 2\pi - \frac{1}{2}\left(\log\beta_i - \psi\left(\alpha_i\right)\right) - \frac{\alpha_i}{2\beta_i}\left[\left(y_i-\mu_i\right)^2 + m_i^2\right]\right\} \tag{7}$$

$$D_{KL}(q(z|y)\|p(z)) = \sum_{i=1}^{d}\left\{\frac{\left(\mu_i - x_i\right)^2}{2\varepsilon_0^2} + \frac{1}{2}\left[\frac{m_i^2}{\varepsilon_0^2} - \log\frac{m_i^2}{\varepsilon_0^2} - 1\right]\right\} \tag{8}$$

$$\begin{aligned} D_{KL}\left(q\left(\sigma^2|y\right)\|p\left(\sigma^2\right)\right) = \sum_{i=1}^{d}\Bigg\{&\left(\alpha_i - \frac{p^2}{2} + 1\right)\psi\left(\alpha_i\right) + \left[\log\Gamma\left(\frac{p^2}{2} - 1\right) - \log\Gamma\left(\alpha_i\right)\right] \\ &+ \left(\frac{p^2}{2} - 1\right)\left(\log\beta_i - \log\frac{p^2\xi_i}{2}\right) + \alpha_i\left(\frac{p^2\xi_i}{2\beta_i} - 1\right)\Bigg\} \end{aligned} \tag{9}$$

$$\min_{W_D, W_S} -\sum_{j=1}^{n}\mathcal{L}\left(z_j, \sigma_j^2; y_j\right) \tag{10}$$

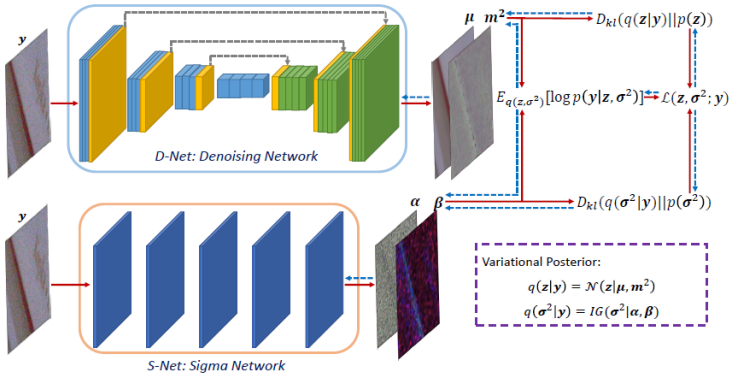


Figure 1: VDN architecture [12]

3 Target objectives

To test the reproducibility of the paper and to check whether their conclusions are reliable, we seek to accomplish the following:

1. Assess if the methods mentioned in the paper are sufficient to obtain the required results
2. Assess the reproducebility cost
3. To provide supplementary information in cases where we find the paper lacking in details

4. To try a few extra experiments by tweaking some properties such as changing network depth and rectifiers.

4 Implementation

After reading the papers on UNet and DnCNN, we built them separately (from scratch) using PyTorch library in Python and then combined them through the loss function. Additionally, we used other popular libraries for computer vision such as NumPy, OpenCV.

4.1 Datasets and Data Pre-processing / Augmentation

The experiments are divided into 2 cases : Synthetic Non-I.I.D. Gaussian Noise case and Real-World Noise case. For the first, we made use of 432 images from BSD [3], 400 images from the validation set of ImageNet [4] and 4744 images from the Waterloo Exploration Database [8] for training, and Set5, LIVE1 and BSD68 [8] for testing. For the second case, we used SIDD Medium Dataset [1] for training, and SIDD validation for testing. Before being used as input for the network, the data required a significant amount of pre-processing:

1. The images were first divided into patches of size 128×128 (consecutive patches for case 1 and patches with a stride for case 2)
2. To each patch, a Gaussian noise was added as follows:

$$\mathbf{n} = \mathbf{n}^1 \odot \mathbf{M}, n_{ij}^1 \sim \mathcal{N}(0, 1)$$

   where $\mathbf{M}$ is the Gaussain map with same size as the patch.
3. For each patch, the peakiness and the center of the Gaussian was varied randomly within a range of $\sigma = [0, 75]$. The reason this range was set was that the other methods such as FDnCNN[13] and FFDNet [14] were also trained using the same range, allowing for a fair comparison of the results.
4. Following this, the patches were randomly augmented in what is called an (8x) flip/rotation ensemble [2]. This means that each patch was either rotated $0°, 90°, 180°, 270°$ with equal probability, or flipped upside down and then rotated in the same manner. Thus creating a total of 8 different augmentations with equal probability. This process was repeated for the ground-truth patches, the generated noise as well as the resulting noisy patches for improved generalisation.

In each epoch, 64×5000 randomly selected patches with size 128×128 were used. For testing, 3 different noise maps were generated : $sin(x)cos(y)$, $ysin(x) - xcos(y)$ and $sin\sqrt{x^2 + y^2}$ ((a), (b) and (c) in figure 2) and an additional Gaussain noise with $\sigma = 15$, 25 and 50 for the AWGN removal task. These were used to generate noisy test data in a process similar to the one used to generate noisy training data, with the only difference being that instead of being split into patches, the entire image was augmented.

It must be noted that the aforementioned pre-processing steps are not explicitly stated in the paper, but were told to us by the authors through our interactions. These steps appear to be standard practice in many of the methods employed in denoising tasks, and are not specific to this one.

4.2 Network Architecture

The two networks *S-Net* and *D-Net* were built with the architectures described in section 2.3. The *D-Net* was built with a U-Net architecture[10] with 3 encoder blocks (each followed by average pooling), 1 bottleneck and 3 decoder blocks with symmetric skip connections. For the *S-Net*, a DnCNN architecture[13] was used with 5 layers and 64 feature channels of each layer. After conferring with the authors, we confirmed that there were slight variations in the hyper-parameter details between those mentioned in the paper, the repository on GitHub and a training log provided to us by the authors. In order to accurately assess the validity of their findings, we implemented all configurations based on all 3. They are described as Config 1, 2 and 3, respectively in Figure 1.

Additionally, a fourth configuration was tested on real world noise that had additional DnCNN layers, this is referred to as Config 4.

Table 1: Hyper-parameter values and various model configurations

	ϵ_0^2	Grad. norm clipping	Batchnorm	ReLu Type
Config. 1 (Paper)	5e-5	N	Y	ReLU
Config 2 (Github)	1e-6	Y(1k, 10k) (grad S, grad D)	N	Leaky ReLU (slope = 0.2)
Config 3 (Authors' Log)	5e-5	Y(50,1000) (grad S, grad D)	N	Leaky ReLU (slope = 0.2)
Config 4 3 Additional DnCNN Layers (extension)	5e-5	N	Y	ReLU

All the models were trained over 60 epochs, with a batch size of 64. A window size $p = 7$ was used. The initial learning rate was set to $2e^{-4}$, with a decay rate of $\gamma = 0.5$. The weights were initialized using the method proposed by *Kaiming et al.* [5].

4.3 Experiments & Additional Extensions

We used all three trained models for the Synthetic Non-I.I.D. Gaussian noise case, and tested them on the 3 test datasets augmented with the 6 different noise maps each. We used PSNR (Peak signal-to-noise ratio) and SSIM (Structural Similarity Index) as measures of fidelity of the predicted denoised "clean images". We also compared our learnt noise with the 3 noise maps to check similarity.

Additionally, as an extension to the methods in the paper, we tried adding two modifications to the VDN - one with a deeper *DnCNN* network (8 layers instead of 5), motivated by the fact that the original *DnCNN* had the best results when built with 20 layers [13]. This is basically the same as configuration 1 but with added layers in the DnCNN. We call it configuration 4.

The other modification is trained only on the real world noise case, In that modification we use an additional DnCNN layer but also using Randomized ReLU (RReLU), which is a randomized version of Leaky ReLU, in both networks. According to (B, Xu et al.,2015), RReLU randomizes the negative slope every time it is used in training, uniformly between (1/8 and 1/3) and achieves better results on test data. Although, the reasons of its superior performance still lacks rigorous justification from a theoretical standpoint [11]. This last configuration is referred to in Table 5.

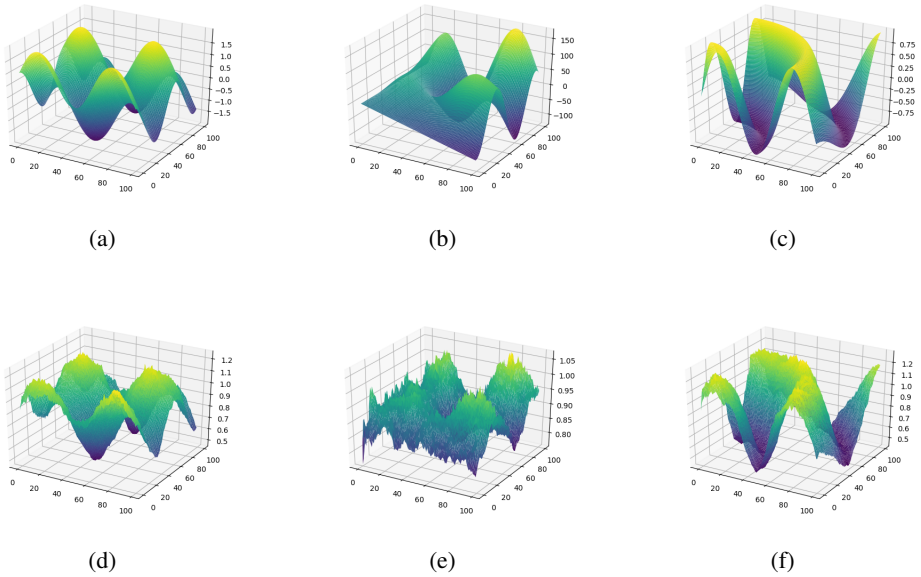
For the Real-World noise case, we trained our models with varying window sizes (p = 5, 7, 11, 15 and 19) and ϵ_0^2 values (1e-4, 1e-5, 1e-6 and 1e-7). In this case too, we used PSNR and SSIM as metric for measuring performance.

The SIDD Benchmark ground truth is withheld. In order to get the PSNR/SSIM values for the SIDD Benchmark, the results have to be uploaded to the data set owners and tested at their site. The denoised images have been sent to the owners and we are still waiting for answers. The DND benchmark ground-truth is also withheld and requires an official submission and waiting time.

4.4 Reproducibility cost

Given that each epoch consisted of $N = 5000 \times 64 = 320.000$ patches, the experiments were computationally expensive. Nvidia Tesla V100 and Nvidia Tesla P100 GPUs were both used and tested. Each epoch took roughly 70 minutes and 120 minutes on the V100 and P100 GPU respectively. It is worth mention that the code has not been tested for optimal performance, so a lower processing time per epoch might be achievable. During our experiments, the training roughly consumed 8-8,5 GBs of VRAM (FP32). For that reason, we used for later experiments the V100, where every full experiment took around 60-70 hours to conduct.

5 Results



(a) (b) (c)

(d) (e) (f)

Figure 2: Figure showing noise maps [a], [b] and [c], and the maps predicted by the model [d], [e] and [f]. The noise maps are generated by the functions $sin(x)cos(y)$, $ysin(x) - xcos(y)$ and $sin\sqrt{x^2 + y^2}$

Table 2: PSNR(dB) and SSIM values obtained by testing on the 3 datasets with AWGN (Config. 1, Config. 2, Config. 3 and Config. 4). Best results are marked in bold.

Noise	Dataset	Conf. 1		Conf. 2		Conf. 3		Conf. 4		VDN Paper	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
σ = 15	LIVE1	29.27	0.892	33.04	0.916	32.95	0.913	33.02	0.917	**34.34**	-
	CBSD68	30.10	0.895	32.81	0.914	32.76	0.912	32.81	0.915	**33.94**	-
	Set5	30.30	0.904	33.40	0.908	33.39	0.908	33.43	0.909	**33.90**	-
σ = 25	LIVE1	29.39	0.891	33.00	0.914	32.94	0.913	**33.04**	0.918	32.34	-
	CBSD68	30.10	0.895	32.82	0.914	32.73	0.911	**32.86**	0.917	31.50	-
	Set5	30.05	0.901	33.44	0.908	**33.47**	0.910	33.34	0.907	31.35	-
σ = 50	LIVE1	29.27	0.892	**33.03**	0.915	32.92	0.912	33.01	0.917	29.47	-
	CBSD68	30.09	0.895	32.80	0.914	32.75	0.912	**32.83**	0.916	28.36	-
	Set5	29.96	0.901	33.37	0.908	**33.43**	0.909	33.37	0.908	28.19	-

Table 3: PSNR(dB) values obtained by testing on the 3 datasets 3 different noise types(Config. 1, Config. 2, Config. 3 and Config. 4). Best results are marked in bold.

Noise	Dataset	Conf. 1		Conf. 2		Conf. 3		Conf. 4	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
Case 1	LIVE1	28.17	0.830	28.41	0.812	27.90	0.791	**28.93**	**0.835**
	CBSD68	27.98	0.823	28.25	0.808	27.81	0.790	**28.77**	**0.827**
	Set5	29.73	0.845	29.50	0.836	28.96	0.817	**29.90**	**0.848**
Case 2	LIVE1	28.88	0.830	**28.98**	**0.834**	28.91	0.832	28.94	**0.834**
	CBSD68	28.60	0.824	28.78	0.826	28.37	0.824	**28.85**	**0.827**
	Set5	29.93	0.848	29.96	0.851	29.83	0.849	**30.00**	0.852
Case 3	LIVE1	27.91	0.829	24.56	0.677	24.14	0.683	**28.79**	**0.835**
	CBSD68	27.51	0.825	24.84	0.693	24.29	0.691	**28.67**	**0.830**
	Set5	29.73	**0.850**	25.60	0.709	24.66	0.704	**29.81**	0.849

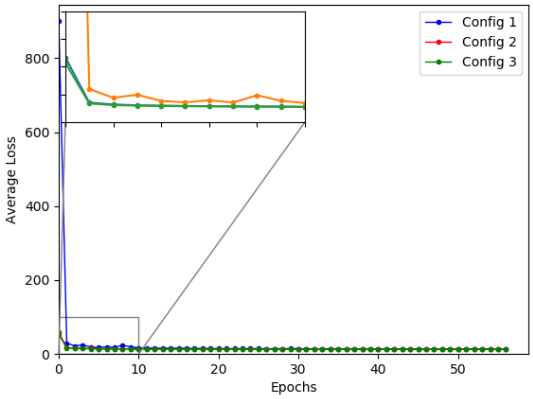
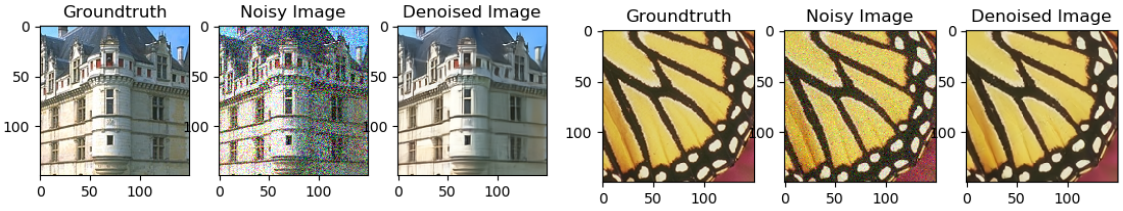


Figure 3: Plot showing the average loss across the epochs for the 3 network configurations. Config 2 is hidden behind the line for Config 3, indicating that their performance is very similiar, despite having different clipping values.



(a) (b)

Figure 4: Comparison between ground truth, noisy and denoised image using the model trained with simulated noise. Figure [a] shows an image from CBSD68 dataset where simulated noise from map (b) [2] was applied on the noisy image. Figure [b] shows an image from Set5 dataset where AWGN with $\sigma = 25$ has been applied.

Table 4: Results of the Real World Noise Case on SIDD validation dataset

(a) Performance with different ϵ_0^2 values and $p = 7$.

	Our Results		**Paper**	
ϵ_0^2	PSNR	SSIM	PSNR	SSIM
1e-4	39.02	0.9127	38.89	0.9046
1e-6	**39.31**	**0.9160**	39.28	0.9086
1e-7	39.29	0.9158	39.05	0.9064

(b) Performance with different p values and $\epsilon_0^2 = 1\mathrm{e} - 6$.

	Our Results		**Paper**	
p	PSNR	SSIM	PSNR	SSIM
5	39.22	0.9153	39.26	0.9089
7	**39.31**	**0.9160**	39.28	0.9086
11	39.27	0.9155	29.26	0.9086

Table 5: Results of our improved model compared with the best results of the original paper.

Extension Study		**Best Results VDN Paper**	
PSNR	SSIM	PSNR	SSIM
39.37	**0.9165**	39.28	0.9086

6 Analysis & Discussion

As can be seen in Figure 2, the network performs very well in estimating the noise. It appears to capture complex noise distributions despite being trained using a Gaussian noise distribution. This is the result of randomizing the peakiness and centre of the Gaussian kernel everytime a patch is randomly cropped. This is a crucial aspect of the training process, as it prevents the network from overfitting to a particular noise type. Instead, the network can be seen as learning a number of Gaussian mixtures, since in each batch, a variety of different Gaussian kernels have been used. Despite being of importance, this detail was omitted from the paper.

As much as the augmentation trick helps, it is the generative approach that, within a solid Bayesian framework, is responsible for its robustness and generalization capability of the model. Table 4 shows similar results to [12] on the SIDD-validation set, despite only training for 30 epochs instead of 60.

Tables 2 and 3 show the PSNR and SSIM values obtained on the test cases using the four model configurations. Although the model parameters of the 4 configurations vary slightly, this did not seem to affect the test performance by a large margin. In all cases, we seem to achieve results close to the ones presented in the paper. The configurations that involve gradient norm clipping have an overall better performance than the one that does not. This, we believe, is because the gradient norm clipping prevents the gradient values from fluctuating too much. Interestingly, the longer DnCNN network (config 4) seem to be making the VDN perform better on non-i.i.d. noise cases but not necessarily on the AWGN case. Adding additional layers may not help with the i.d.d. case as there may not be any more complex patterns to capture.

Figure 3 shows the average loss at different stages of training for the first 3 different model configurations. As can be seen, all models show a sharp drop in the loss value during the first epoch, after which they drop very slowly. We continue to train them for 60 epochs, as mentioned in the paper. In hindsight, it would have been interesting to compare these values against a validation set to observe the trends. The main reason, we believe, that the configurations 2 and 3 perform so similarly, despite being having their gradient norms clipped at different thresholds, is that a moving average is actually used, where the smallest gradient norm in each epoch is set as the new maximum allowed norm in the next.

Figure 4 shows the groundtruth, noisy and denoised images obtained from our trained model. As can be seen, the model manages to recover sharp edges and fine details without blurring or over-smoothening edges, which is a notable feature in denoising models.

Table 4 show the performance of VDN under different values of p and ϵ^2. The best results are indeed achieved at the hyperparameter parameter values desribed by the authors. Namely,

$\epsilon_0^2 = 1e - 6$ and p = 7. The original results from the VDN paper are also included for comparison. These tests lead us to believe that the VND is a powerful model capable of denoising varied noise types. Moreover, comparisons with previously obtained results of other models show that at the moment VDN is one of the best for this task.

Finally, our model of using both one extra layer on DnCNN and with RReLU seems to be performing not only better, but also better than what the authors achieved. This reflects that fact that the authors themselves have yet to try many modifications. In our exchanges, they state that they are still assessing modifications of their own. The results are seen in Table 5.

7 Conclusions & future suggestions

In this paper we have tried to analyse the reproducibility and reliability of the paper *Variational Denoising Network: Toward Blind Noise Modeling and Removal* [12]. We found the paper to be reproducible to a good extent. Although the paper skips the details necessary for the data augmentation and gradient clipping, we observed that omitting them does not adversely affect the results. Some other network parameters such as *stride*, *kernel size* etc. were not mentioned, but in most cases using the default values or using the values in the cited papers seemed enough to reproduce the results. Though we managed to reproduce most parts of the paper, we found guessing and estimating these missing details rather challenging. As a final note, adding RReLU and a deeper DnCNN seemed to have helped with performance. This indicates there are more potential improvements as the UNet and DnCNN can be replaced with any other denoising network.

References

[1] Abdelrahman Abdelhamed, Stephen Lin, and Michael S Brown. A high-quality denoising dataset for smartphone cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1692–1700, 2018.

[2] Abdelrahman Abdelhamed, Radu Timofte, and Michael S Brown. Ntire 2019 challenge on real image denoising: Methods and results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2010.

[4] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S Bernstein, Alex Berg, and Li Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102. ACM, 2014.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[8] Kede Ma, Zhengfang Duanmu, Qingbo Wu, Zhou Wang, Hongwei Yong, Hongliang Li, and Lei Zhang. Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 26(2):1004–1016, 2016.

[9] Michael Moeller and Daniel Cremers. *Image Denoising—Old and New: Fundamentals, Open Challenges and New Trends*, pages 63–91. 09 2018.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[11] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[12] Zongsheng Yue, Hongwei Yong, Qian Zhao, Lei Zhang, and Deyu Meng. Variational denoising network: Toward blind noise modeling and removal. *arXiv preprint arXiv:1908.11314*, 2019.

[13] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[14] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

8 Appendix

Figure 5: Original, noisy and denoised image from dataset Set5 where AWGN with $\sigma = 50$ has been applied

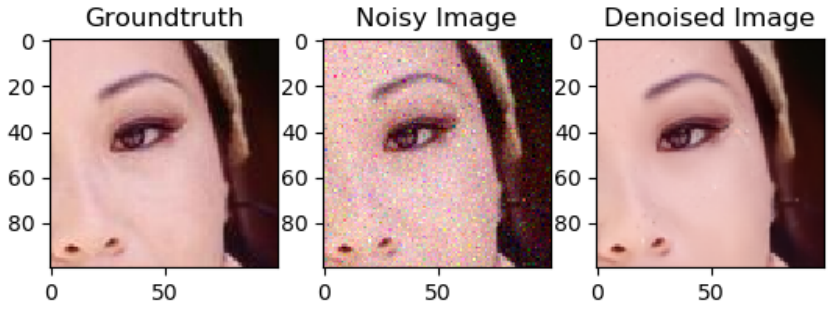


Figure 6: Original, noisy and denoised image from dataset CBSD68 where noise b form Figure 2 has been applied