Published in Image Processing On Line on 2016–01–19.
Submitted on 2015–09–16, accepted on 2016–01–14.
ISSN 2105–1232 © 2016 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2016.150

Small Neural Networks can Denoise Image Textures Well: a Useful Complement to BM3D

Yi-Qing Wang

CMLA, ENS Cachan, France (yiqing.wang@cmla.ens-cachan.fr)

Communicated by Marc Lebrun *Demo edited by* Yi-Qing Wang

Abstract

Recent years have seen a surge of interest in deep neural networks fueled by their successful applications in numerous image processing and computer vision tasks. However, such applications typically come with huge computational loads. In this article, we explore the possibility of using small neural networks to denoise images. In particular, we present SSaNN (Self-Similarity and Neural Networks), a denoising algorithm which combines the strength of BM3D on large-scale structured patterns with that of neural networks on small-scale texture content. This algorithm is able to produce a better overall recovery than both BM3D and small neural networks.

Source Code

The source code and an online demo are accessible at the IPOL web page of this article[1].

Keywords: feedforward neural networks; small-scale pattern denoising

1 Introduction

Multilayer neural networks have recently ignited much interest in the signal processing community thanks to their broad range of successful applications [1]. For example, it has been shown that a multilayer neural network, when endowed with sufficient capacity, could be trained to deliver state-of-the-art performance in denoising [2, 8] as well as in demosaicing [5, 9, 7].

This superior performance does come at a price. Because of its non-convex structure, no efficient algorithm exists to solve a typical empirical risk minimization program involving a multilayer neural network as the regression function. As a result, stochastic gradient descent becomes the *de facto* tool of choice although it implies an enormous computational cost. Moreover, as far as image denoising is concerned, a large observation window is required of a neural network to achieve an acceptable image denoising performance, because natural images typically have a great many smooth or highly structured areas. This again translates into a high time complexity when such a trained neural

[1] https://doi.org/10.5201/ipol.2016.150

network works on noisy images due to its wide hidden layers and deep architecture. Indeed, despite their performance, the neural networks presented in [2] need tens of millions of operations per pixel when denoising gray-scale images.

SSaNN (Self-Similarity and Neural Networks) [6] observes that while self-similarity is a fundamental property of natural signals in general, its range of application is affected by the signals' finite resolution inherent to their respective acquisition process. It is thus not surprising that self-similarity based algorithms like BM3D [3, 4] work less well on small-scale textures than on regular patterns of a scale larger than the algorithm's fixed patch size. On the other hand, it can be argued that approximating a Bayesian optimal filter, neural networks may be relatively insensitive to scale variation as long as the scale is small relative to the neural network's input size.

2 Image Denoising Neural Networks

We do not detail the training of our denoising neural networks as it has already been documented [7]. The only difference is that following [2], we collected the training and validation patches from Pascal VOC2012 [2] and processed them according to Algorithm 1 before feeding them to the neural networks.

Algorithm 1 Generating a Supervised Pair

1: **Input:** a clean patch $p \in \mathbb{R}^d$.
2: **Output:** a supervised pair.
3: **Parameter:** Gaussian noise level σ.
4: Create a noisy copy of the input patch $\tilde{p} = p + \sigma n$ with n distributed as $\mathcal{N}(0, I_d)$.
5: Normalize both p and $\tilde{p}$ pixel-wise with the linear transform $L(x) = 51^{-1}x - 2.5$.

Three neural networks of the same architecture have been trained under Gaussian noise levels set to 10, 25 and 35 respectively. Observing a 7-by-7 noisy patch, they attempt to predict the clean state of its central 3-by-3 block. These neural networks have 3 hidden layers composed of 147 non-linear encoding units each. During their 4×10^8 round training, the neural networks were tested every 10^4 rounds on a validation dataset comprising 10^4 examples from 100 randomly selected images of Pascal VOC2012 (see Figure 1). Its remaining 17025 images served as the training dataset.

Denoising with the trained neural networks is straightforward (see Algorithm 2 and Figure 3). Interestingly, we can also inspect the features that these neural networks acquired during the training and are thus deemed as important for the denoising task (see Figure 2). Regardless of input noise level, most of them turn out to be either oriented edges of varying lengths or rather localized blobs. But having many input features in common does not make one neural network similar to another [8].

Algorithm 2 Neural network denoising

1: **Input:** a gray-scale image affected by Gaussian noise of standard deviation σ.
2: **Output:** a denoised image.
3: **Parameter:** a neural network trained to handle the same level of Gaussian noise.
4: Decompose the noisy image into overlapping 7-by-7 patches with spatial offset equal to 1.
5: Normalize these patches with the linear transform in Algorithm 1 and run them through the neural network to form estimated patches.
6: Apply the inverse linear transform pixel-wise $L^{-1}(x) = 51(x + 2.5)$ to the estimates.
7: Aggregate patches with equal weights to form the final image.

[2] http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2012/index.html

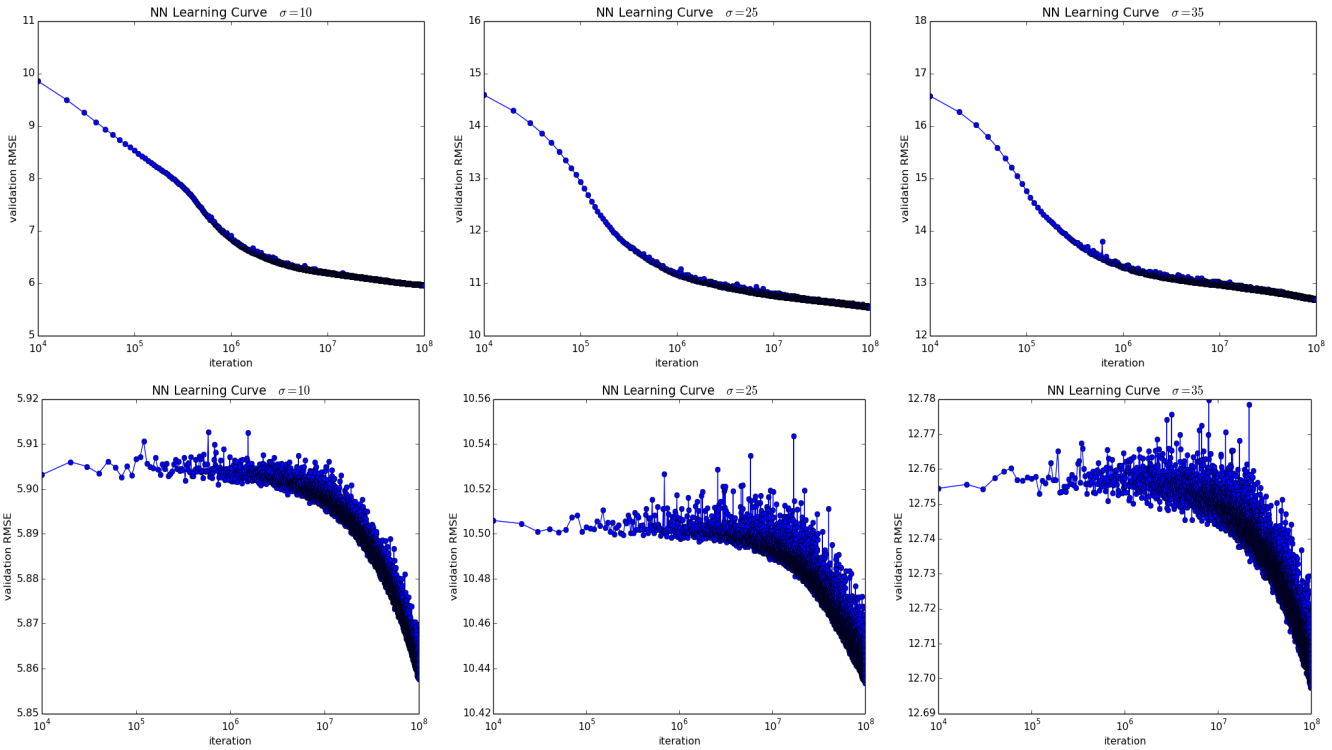

Figure 1: The validation RMSE (root mean square error) of the three neural networks trained under Gaussian noise level 10 (left), 25 (middle) and 35 (right) was recorded every 10^4 backpropagation iterations during their respective 4×10^8 round training. The top row corresponds to the first 10^8 round training and the bottom row the second 10^8 rounds. The horizontal axes are all plotted in the logarithmic scale. Two more training cycles had been carried out until their performances stabilized.

3 SSaNN

As stated in the introduction, SSaNN seeks to improve BM3D on small-scale textures where self-similarity cannot be meaningfully exploited for denoising purposes (see Figure 3). Some of these areas can be detected by Algorithm 3. They tend to be bright and thus having a high signal-to-noise ratio. Based on the output of this texture detector, SSaNN (Algorithm 4) assembles a new image by taking values from the neural network restored image only if the pixels in question are part of small scale texture patterns. The results as shown in Table 1 and Figure 3 and 4 indeed indicate that for images rich in texture content, a better recovery can be obtained by complementing BM3D with small neural networks. A visual comparison is given in Figure 5.

As to the parameters used in the texture detector, we set h=7, κ=12, n=3 and

$$\beta = \begin{cases} 1.7, & \sigma = 10 \\ 1.2, & \sigma = 25 \\ 1.1, & \sigma = 35 \end{cases},$$

to reflect a decreasing signal to noise ratio.

Finally, note that one might be tempted to apply the same technique on noisy color images. However, a naive approach with the same patch size would triple the input data size. As a result, the number of computing units at each layer, if not the number of hidden layers itself, needs to be adjusted accordingly. It results in a polynomial growth in the neural network's runtime complexity. Compounding the difficulty is that it also becomes more difficult to train a larger neural network.

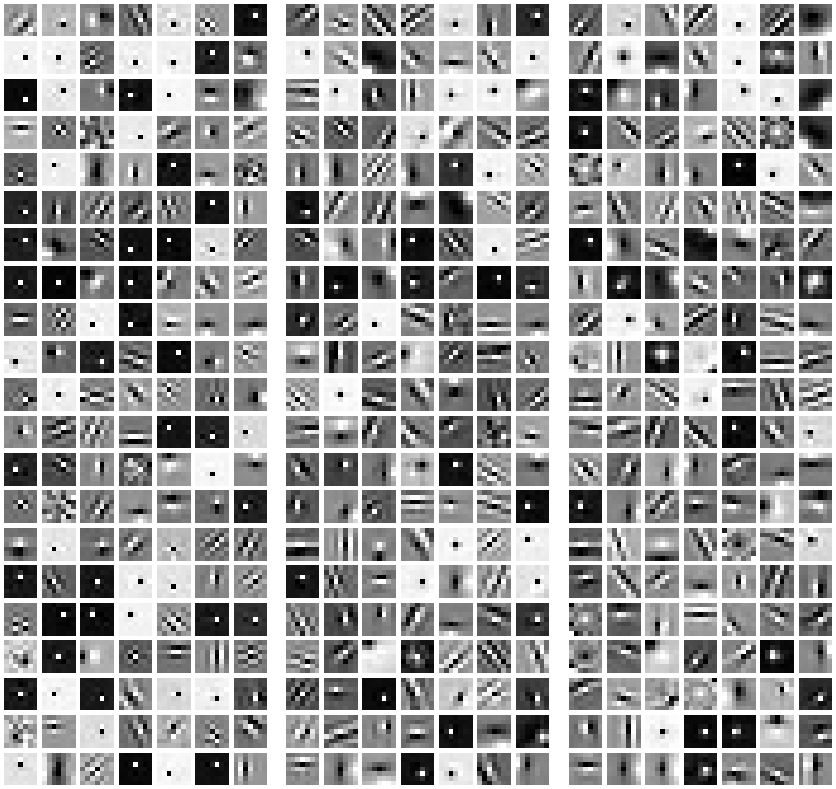

Figure 2: The learned input features of three image denoising neural networks trained under Gaussian noise level 10 (left), 25 (middle), 35 (right) respectively. Note that regardless of input noise level, most of them depict oriented edges and localized blobs. Having features in common, however, does not mean that these neural networks trained under different noise levels are interchangeable.

Figure 3: Noisy *valldemossa* (σ=25) recovered by the neural network (left) and BM3D (right). The RMSEs of both recoveries are close and around 12.32. But the sky and building facade restored by BM3D is much cleaner, indicating a better recovery. This fact also implies that the neural network does a better job on other parts of the image.

In response, we have tried some data reduction techniques using color transforms and PCA, though without success.

Algorithm 3 Texture detector

1: **Input:** a noisy image and a reference patch.
2: **Parameter:** Gaussian noise standard deviation σ, patch size h, search window size κ, similarity threshold β, required number of similar neighbors n.
3: Compute the difference between the reference patch and its surrounding patches in the search window. Declare a neighboring patch similar if its ℓ_2-distance is smaller than $\sqrt{2}\beta\sigma h$.
4: Label the reference patch as structure if more than n similar patches have been found.

Algorithm 4 SSaNN

1: **Input:** noisy image $\tilde{I}$.
2: **Output:** denoised image.
3: **Parameter:** Gaussian noise standard deviation σ.
4: Denoise $\tilde{I}$ with BM3D and the neural network to get I_B and I_N.
5: Extend $\tilde{I}$ to $\bar{I}$ so that $\bar{I}$'s 7-by-7 patches form a one-to-one mapping with $\tilde{I}$'s pixels.
6: Use Algorithm 3 to classify $\tilde{I}$'s patches (hence $\bar{I}$'s pixels) into either structure or texture. Form a matrix T_S of the same size as $\bar{I}$ with $T_S(x, y)$ set to 1 if $\bar{I}(x, y)$ is labeled structure and 0 otherwise.
7: Return $I_N \circ (1 - T_S) + I_B \circ T_S$ where $\circ$ denotes the element-wise product.

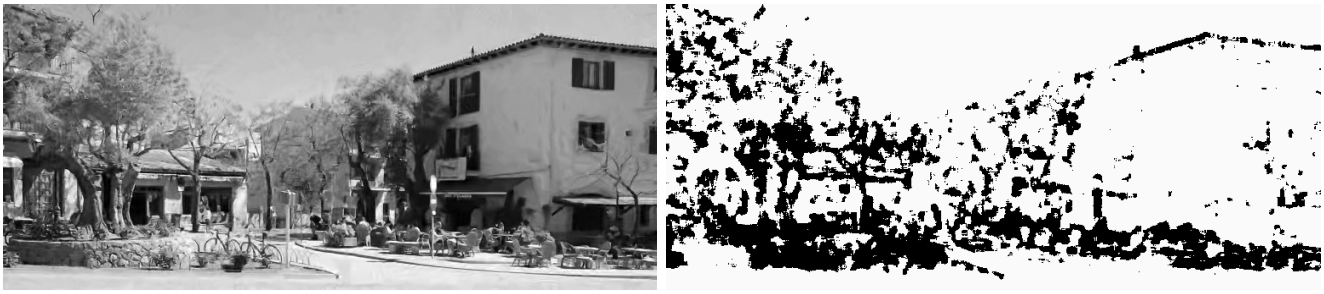

Figure 4: SSaNN forms a new image (left) by replacing texture pixels in BM3D's restored image with those estimated by the neural network. Pixels identified as part of the texture are marked in black in the right image. This combination enables a RMSE reduction from 12.34 to 12.15.

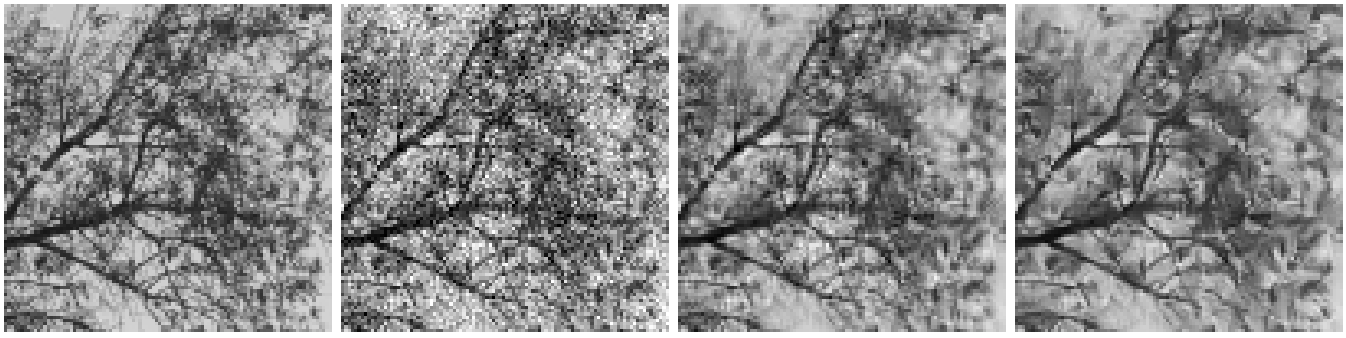

Figure 5: A visual comparison between BM3D and SSaNN: from left to right, we have respectively the original image, its noisy version affected by Gaussian noise with level 25, and its restoration by BM3D (RMSE = 19.23) and SSaNN (RMSE = 19.16).

σ=10	NN	BM3D	SSaNN
computer	4.85	4.66	**4.59**
dice	2.60	**1.82**	**1.82**
flower	3.22	2.86	**2.83**
girl	2.95	**2.36**	**2.35**
traffic	5.68	5.68	**5.60**
valldemossa	6.56	6.63	**6.51**
average	4.31	4.00	**3.95**

σ=25	NN	BM3D	SSaNN
computer	8.85	8.19	**8.13**
dice	4.89	**3.12**	**3.12**
flower	5.68	5.13	**5.11**
girl	5.14	**3.67**	**3.67**
traffic	10.24	10.03	**9.92**
valldemossa	12.30	12.34	**12.15**
average	7.85	7.08	**7.02**

σ=35	NN	BM3D	SSaNN
computer	10.89	**9.92**	**9.91**
dice	6.35	**3.79**	**3.79**
flower	7.04	6.37	**6.35**
girl	6.53	**4.47**	**4.47**
traffic	12.29	11.88	**11.78**
valldemossa	14.82	14.76	**14.56**
average	9.65	8.53	**8.48**

Table 1: Algorithm comparison in RMSE on six IPOL images. The best results are marked in bold.

Acknowledgment

Work partly founded by DxO Labs, the European Research Council (advanced grant Twelve Labours nº 246961) and the Office of Naval research (ONR grant N00014-14-1-0023).

Image Credits



CC-BY A. Buades.

References

[1] Y. Bengio, *Learning deep architectures for AI*, Foundations and trends® in Machine Learning, 2 (2009), pp. 1–127. http://dx.doi.org/10.1561/2200000006.

[2] H. Burger, C. Schuler, and S. Harmeling, *Image denoising: Can plain neural networks compete with BM3D?*, in IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2392–2399. http://dx.doi.org/10.1109/CVPR.2012.6247952.

[3] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, *Image denoising by sparse 3-D transform-domain collaborative filtering*, IEEE Transactions on Image Processing, 16 (2007), pp. 2080–2095. http://dx.doi.org/10.1109/TIP.2007.901238.

[4] M. Lebrun, *An Analysis and Implementation of the BM3D Image Denoising Method*, Image Processing On Line, 2 (2012), pp. 175–213. http://dx.doi.org/10.5201/ipol.2012.l-bm3d.

[5] Y. Q. Wang, *A multilayer neural network for image demosaicking*, in IEEE International Conference on Image Processing, Oct 2014, pp. 1852–1856. http://dx.doi.org/10.1109/ICIP.2014.7025371.

[6] Y. Q. Wang, *A note on the size of denoising neural networks*, SIAM Journal on Imaging Sciences, (2015). Accepted.

[7] Y. Q. Wang and N. Limare, *A Fast C++ Implementation of Neural Network Backpropagation Training Algorithm: Application to Bayesian Optimal Image Demosaicking*, Image Processing On Line, (2015). http://dx.doi.org/10.5201/ipol.2015.137.

[8] Y. Q. Wang and J. M. Morel, *Can a Single Image Denoising Neural Network Handle All Levels of Gaussian Noise?*, IEEE Signal Processing Letters, (2014). http://dx.doi.org/10.1109/LSP.2014.2314613.

[9] ——, *A dilemma of demosaicing methods and its Bayesian solution*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2015). Submitted.