Exploiting the Redundancy of Deep Neural Networks for Image Denoising

Yi-Qing Wang, and Jean-Michel Morel

***Abstract*—Deep neural networks have been shown to hold great promise for various computer vision and image processing tasks. However, they are usually computationally expensive to train and deploy because of their enormous sizes. In this note, using certain invariances of the natural patch distribution, we show that state-of-the-art image denoising networks are redundant in both their architectures and their domain of definition, which suggests that a much smaller network trained on a much smaller dataset might do the task just as well. Two crucial numerical experiments lend full support to our invariance arguments.**

***Index Terms*—deep neural network, natural patch space, distribution invariance, redundancy analysis**

I. Introduction

RECENTLY, a deep neural network based approach [1] to image denoising has been shown to outperform BM3D [2], widely accepted as the state-of-the-art until now. This dramatic turn of event clearly demonstrates the supremacy of a pure learning strategy over careful handcrafting, a prevailing practice in image processing. A philosophical difference sets these two patch-based methods apart: BM3D, a major spin-off of the original non-local means [3], seeks information exclusively inside the noisy image while the neural network derives all its power by looking at noisy and clean patch pairs gathered from other images. Other algorithms exist [4], [5], [6] which fall between these two ends of the spectrum.

It is well known [7] that neural networks form a class of universal approximators. Recent studies [8] further suggest that multi-layer, or deep, neural networks tend to be more efficient in signal representation than their traditional one-hidden-layer counterpart. Whatever their architectures, neural networks in a regression framework seek to approximate the conditional expectation under some input distribution. In our setting, let x, $\tilde{x}$, and y denote the clean, noisy and denoised patch. Note that y does not necessarily have the same dimension as $\tilde{x}$. All the neural networks [1] under this study, for instance, produce a $y \in \mathbb{R}^{17\times 17}$ based on a noisy observation $\tilde{x} \in \mathbb{R}^{39\times 39}$ which includes not only y's corresponding noisy pixels but also those surrounding ones. Also note that with the *true* patch distribution beyond reach, a huge number of patches are drawn from a large natural image dataset to provide the conditional expectation's underlying distribution

$$\begin{aligned}\theta^* &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\|f(\tilde{x},\theta) - x\|_2^2 \\ &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}\|f(\tilde{x},\theta) - \mathbb{E}[x|\tilde{x}]\|_2^2\end{aligned} \tag{1}$$

The authors are with CMLA, Ecole Normale Supérieure de Cachan, Cachan, 94230, France (e-mail: yqwang9@gmail.com; morel@cmla.ens-cachan.fr).

where $f(\cdot,\theta) : \tilde{x} \mapsto y$ is a neural network and θ its connection weights. Due to this problem's non-convex nature in general, instead of the potentially intractable θ^*, a good θ, judged on the basis of the resulting network's generalization error, is usually accepted as a solution.

Despite their impressive performance, the proposed deep networks [1] are impractical for the majority of industrial applications because they impose a daunting computational cost, besides being inflexible in response to different noise levels. In this work, by analyzing the interplay between the neural networks and the underlying patch distribution they seek to learn, we construct a generic neural network able to handle all levels of Gaussian noise and suggest two ways to scale down these networks without compromising their power.

II. Towards one generic network

A standard algorithm for Gaussian noise removal needs to know the noise's variance, which seems to require, however unrealistically, a series of networks, one for each noise level. Fortunately, this is not the case. To show it, we started by investigating the statistical regularity of the *natural patch space*: we drew 10^6 random patches of dimension 39×39 from the *Berkeley Segmentation Dataset* (BSD500) rendered to grayscale with *Matlab*'s `rgb2gray` function. These patches were then normalized using the formula

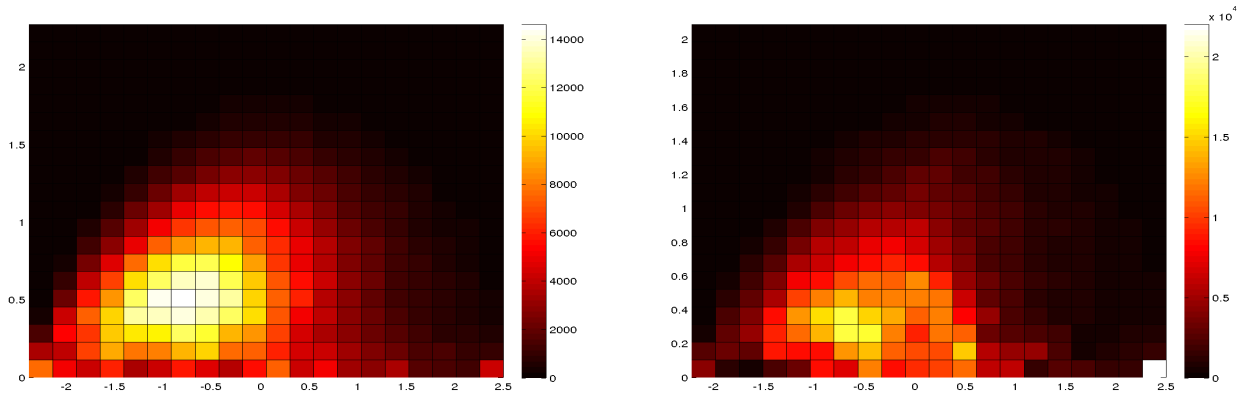
$$\bar{p} = (p/255 - 0.5) \cdot 5$$

provided in [1]. For each normalized patch, we computed the mean and standard deviation of its 1521 pixels and then plotted their population distribution along these two dimensions. For comparison, we did the same with the *Kodak* image dataset (http://r0k.us/graphics/kodak). The results (Fig.1) show that most normalized natural patches are distributed around $(-0.5, 0.5)$. Even without access to the supervised pairs used in [1], we can therefore expect their neural networks trained according to the criterion (1) to do well with patches from this neighborhood because of the sheer number of examples available. Moreover, the patch-wide variance peaking at the patch-wide means around -0.5 implies a higher tolerance for linear transforms there. That is, it is more probable for a natural patch p transformed by

$$q = a \cdot (p - s) \text{ with } s \text{ some constant patch and } a \geq 0$$

to remain *natural* if q's mean is close to -0.5.

To verify this hypothesis, we shifted the mean of individual patches before denoising them. Running Algorithm 1 on 10^5 39×39 random patches drawn from the grayscale BSD500, we



(a) (b)

Fig. 1. 2D histogram of random patches from (1a) the grayscale BSD500 and (1b) the grayscale *Kodak*. The horizontal (resp. vertical) axis represents the normalized patch's mean (resp. standard variation). In both cases, patches concentrate around $(-0.5, 0.5)$. Also note that at the two ends of the horizontal axis, there are two important flat patch clusters.

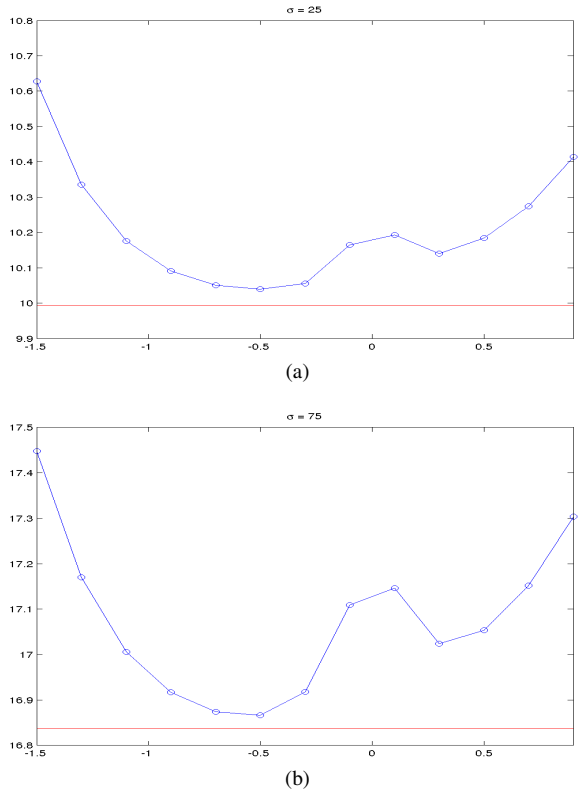
computed the resulting root mean square error (RMSE). Fig.2 shows that regardless of the applied noise strength σ, the best empirical performance is attained with the patch mean shift set to -0.5, thereby confirming our conjecture. In addition, it is important to note that thanks to a high ratio between the patch size and σ, estimating the clean patch mean from its noisy version (Line 7 in Algorithm 1) is fairly reliable. The fact that the patch mean shift yields a slightly worse RMSE may be attributed to our indiscriminate treatment of all patches.

Algorithm 1 Patch Mean Normalization Test

1: **Input:** n clean patches p_j of dimension 39×39
2: **Output:** n denoised (resp. clean) 17×17 patches $\widehat{\mathfrak{p}}_j$ (resp. $\mathfrak{p}_j$)
3: **Parameter:** noise variance σ^2 and desired patch mean value $\mathfrak{m}$
4: **for** $j = 1$ to n **do**
5: retrieve the 17×17 clean patch $\mathfrak{p}_j$ in the center of p_j
6: generate the normalized noisy patch

$$\tilde{x}_j \leftarrow ((p_j + n_j)/255 - 0.5) \cdot 5$$

with n_j having 1521 i.i.d. Gaussian random variables $\mathcal{N}(0, \sigma^2)$ and n_j independent of $n_{j'}$ for $j \neq j'$
7: compute the patch mean shift $s_j = \frac{1}{1521} \sum_{k=1}^{1521} \tilde{x}_{jk} - \mathfrak{m}$ where $\tilde{x}_{jk}$ is the k-th pixel in $\tilde{x}_j$
8: shift the network's input $\forall k$, $\tilde{x}_{jk} \leftarrow \tilde{x}_{jk} - s_j$
9: denoise $y_j = f_\sigma(\tilde{x}_j, \theta)$ where $f_\sigma(\cdot, \theta)$ is the deep neural network [1] trained with Gaussian noise standard deviation σ
10: shift the network's output by the same amount $\forall k$, $y_{jk} \leftarrow y_{jk} + s_j$ in the opposite direction
11: reverse the normalization $\widehat{\mathfrak{p}}_j \leftarrow (y_j/5 + 0.5) \cdot 255$
12: **end for**



(a) (b)

Fig. 2. (2a) (resp. (2b)) plots the RMSEs resulting from Algorithm 1 with $\sigma = 25$ (resp. 75). The blue curve records the dedicated network's performance with the patch mean values ranging from -1.5 to 0.9. And the red line is the RMSE achieved on the same test data without patch mean shift. Other available neural networks have also been tested with very similar results.

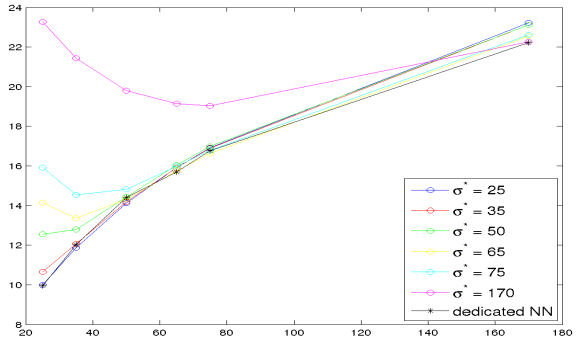
The previous analysis paves the way for a generic network able to handle all levels of noise. Let us use the neural network trained with the noise level σ^* for instance. To restore a noisy patch $\tilde{p}$ with noise standard deviation at σ, one multiplies $\tilde{p}$ by $\sigma^*\sigma^{-1}$, apply the patch mean shift and let the neural network operate on the normalized patch (Algorithm 2). Henceforth σ^* itself becomes a parameter for further optimization, too.

Fig.3 shows that the generic network built with $\sigma^* = 25$ is the best, which is hardly surprising because a neural network can learn the most about the underlying patch space when noise is weak. Moreover, except for the extremely noisy case, one sees no tangible difference in Fig.3 between a dedicated network and the best generic network. Also observe that the higher the σ^*, the worse the RMSEs at low noise levels, an expected phenomenon since a high $\sigma^*\sigma^{-1}$ exaggerates the patch-wide variation so much that resultant patches no longer remain in the natural patch space.

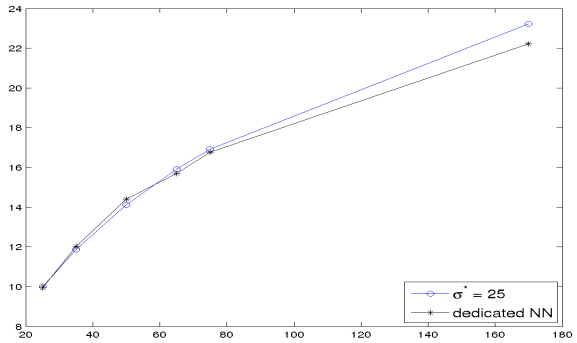
The implication of this analysis is straightforward: training a network with the same architecture but at an even lower noise level may be rewarding. But it should also be said that too low a σ^* is not likely to work well in a strong noise environment, as already observed in Fig.3. Because the factor $\sigma^*\sigma^{-1}$ will

Algorithm 2 Generic Neural Network Denoising

1: **Input:** noisy patch $\tilde{p}$ of dimension 39×39 and its noise level σ
2: **Output:** denoised 17×17 patch $\widehat{\mathfrak{p}}$
3: **Parameter:** noise level σ^* of the trained neural network $f_{\sigma^*}(\cdot,\theta)$ and optimal shift $\mathfrak{m} = -0.5$
4: Scale the noise $\tilde{p} \leftarrow \sigma^* \sigma^{-1} \tilde{p}$
5: Normalize the patch $\tilde{x} \leftarrow (\tilde{p}/255 - 0.5) \cdot 5$
6: Compute the patch mean shift $s = \frac{1}{1521} \sum_{k=1}^{1521} \tilde{x}_k - \mathfrak{m}$ where $\tilde{x}_k$ is the k-th pixel in $\tilde{x}$
7: Shift the network's input mean $\forall k,\ \tilde{x}_k \leftarrow \tilde{x}_k - s$
8: Run the generic neural network denoising $y = f_{\sigma^*}(\tilde{x},\theta)$
9: Shift the network's output by the same amount $\forall k,\ y_k \leftarrow y_k + s$ in the opposite direction
10: Reverse the normalization $\widehat{\mathfrak{p}} \leftarrow (y/5 + 0.5) \cdot 255 \cdot (\sigma^* \sigma^{-1})^{-1}$



(a)



(b)

Fig. 3. (3a) Performance discrepancy between dedicated neural networks and σ^*-indexed generic neural networks (using Algorithm 2) at handling different noise levels. The horizontal axis marks various test Gaussian noise levels and the vertical axis the achieved RMSEs on 10^5 random patches from BSD500. (3b) singles out the best generic network with $\sigma^* = 25$.

then flatten all the meaningful patch-wide variations, leaving the network unable to tell one patch from another.

Put differently, what we have shown so far is that the trained deep network is not very different from its most informative section. Hence, if it had been trained on mean normalized patches, its performance would have been better thanks to a denser data distribution. Even with the current network, it may still be possible to improve Algorithm 2 with a variance dependent patch mean shift, resulting in an undulating section on the mean-variance plane. However, this would require additional knowledge of the training patch distribution.

To conclude this section, we tested our simple algorithm with $\sigma^* = 25$ on some real images. The test images (Fig.4) come from the *Image Processing On Line* (IPOL) website and the results are compiled in Tab.I. Recall that even for $\sigma = 25$, our generic network is different from the dedicated one, in that ours involves one additional step of patch mean shift. The obtained results are thus consistent with Fig.2a.

TABLE I
COMPARISON BETWEEN DEDICATED NEURAL NETWORKS AND OUR ALGORITHM (GENERIC $\sigma^* = 25$)

$\sigma = 25$	dedicated	generic
computer	**7.99**	8.10
dice	2.77	2.77
flower	**4.59**	4.63
girl	**3.38**	3.41
traffic	**9.16**	9.22
valldemossa	**11.85**	11.99
avg.	**6.62**	6.68

$\sigma = 35$	dedicated	generic
computer	**9.63**	9.82
dice	**3.29**	3.45
flower	**5.53**	5.64
girl	**3.88**	4.07
traffic	**10.81**	10.94
valldemossa	**14.21**	14.28
avg.	**7.89**	8.03

$\sigma = 50$	dedicated	generic
computer	**11.59**	11.92
dice	**4.17**	4.50
flower	**6.85**	7.06
girl	**4.60**	4.92
traffic	**12.83**	13.07
valldemossa	**16.98**	17.06
avg.	**9.50**	9.75

$\sigma = 65$	dedicated	generic
computer	**13.16**	13.82
dice	**4.83**	5.39
flower	**7.89**	8.20
girl	**5.28**	5.74
traffic	**14.32**	14.78
valldemossa	**18.67**	18.99
avg.	**10.69**	11.15

$\sigma = 75$	dedicated	generic
computer	**14.10**	14.78
dice	**5.48**	6.14
flower	**8.57**	8.96
girl	**5.66**	6.20
traffic	**15.08**	15.56
valldemossa	**19.97**	20.34
avg.	**11.47**	11.99

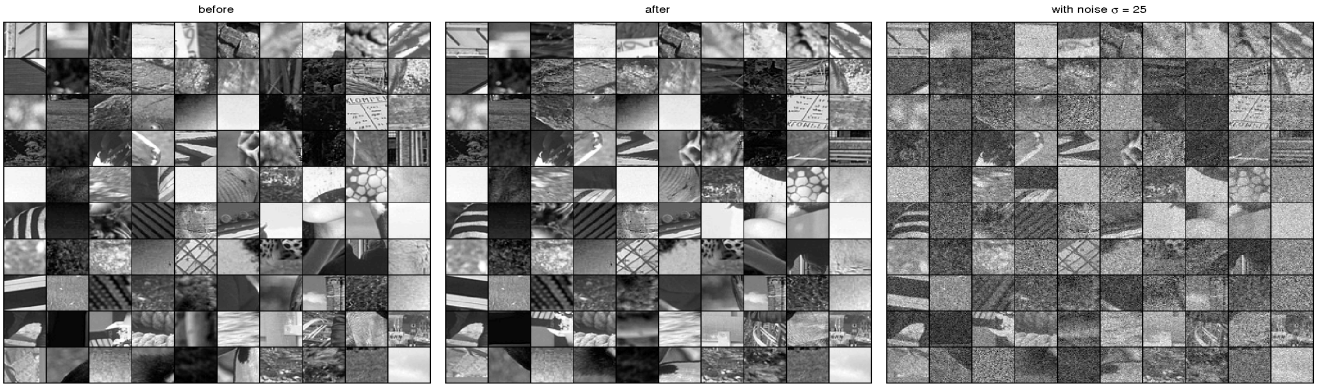
$\sigma = 170$	dedicated	generic
computer	**20.51**	21.81
dice	**9.23**	10.92
flower	**12.84**	13.64
girl	**8.79**	10.54
traffic	**20.71**	21.72
valldemossa	**26.42**	27.26
avg.	**16.41**	17.64

(a) (b) (c) (d) (e) (f)

Fig. 4. Test images for algorithm comparison (4a) computer (4b) dice (4c) flower (4d) girl (4e) traffic (4f) valldemossa. All images are of dimension 704×469 except for valldemossa (769×338).

III. NEURAL NETWORK DOMAIN REDUCTION

This second issue is less task specific. Viewed as an ordinary function, one major source of a neural network's redundancy comes from its domain of definition [9]. If we can both restrict its domain and keep all the training examples at the same time, a neural network shall be able to learn better for two reasons. First, the image patches' intrinsic geometry makes it essential for a good but rudimentary neural network to acquire a certain symmetry, although this property does not seem to follow naturally from the way a network is parametrized, thereby leading to a larger than necessary architecture and training pool. It is thus desirable to remove this degree of freedom from the network's inputs. Second, if we can somehow convert and bring training examples to a smaller area that would otherwise be scattered around in the domain, in a spirit akin to importance sampling in Monte Carlo, the network should learn better with more data (see the previous section).



(a) (b) (c)

Fig. 5. (5a) Some random patches of dimension 39×39 from the grayscale BSD500 (5b) their geometrically normalized versions (5c) the geometrically normalized noisy versions of (5a) with $\sigma = 25$ (using Algorithm 3). There is one noticeable noise induced error (the fourth from left in the second row).

Fortunately, this is feasible, again due to the natural patch distribution being invariant to many linear operations. The two *geometrically normalizing* operations proposed in Algorithm 3 are such examples. They are reversible, non patch size specific, and thus can be applied without difficulty to a possibly smaller denoised patch as in our case. In addition, invariant to linear transforms by design, they are also insensitive to noise, provided that the patch they operate on is large enough. As shown in Fig.5, Algorithm 3 is indeed robust. The inconsistency that occurs at times due to noise, however, is unlikely to change much the result in RMSE because it affects only those patches with insignificant intrinsic orientation. For an empirical validation of this hypothesis, we ran 10^5 geometrically normalized random noisy patches ($\sigma = 25$) through the neural network trained with $\sigma^* = 25$ and applied the reversed operators T_p^{-1} on the outputs. The resulting RMSE (9.96), compared with that (9.95) from a direct application of the neural network using the same noisy patches without geometrical normalization, fully confirmed our approach.

Algorithm 3 Patch Geometrical Normalization

Input: a possibly noisy patch p of dimension 39×39
Output: its geometrically normalized version p_g and the associated geometrically normalizing operator $T_p : p \mapsto p_g$
Let $\mathfrak{1}$ be a column vector of 39 ones and $\mathrm{var}(\cdot)$ be the variance of the argument vector's entries.
if $\mathrm{var}(p\mathfrak{1}) > \mathrm{var}(p'\mathfrak{1})$ **then**
    $p_g \leftarrow p'$ and $T_p \leftarrow \mathfrak{t}$ with $\mathfrak{t}$ the standard matrix transposition
else
    $p_g \leftarrow p$ and $T_p \leftarrow \mathfrak{I}$ with $\mathfrak{I}$ the identity
end if
if the sum of the top half of p_g is bigger than that of its bottom half **then**
    Flip p_g upside down and $T_p \leftarrow \mathfrak{f} \circ T_p$ with $\mathfrak{f}$ the flipping operation
else
    $T_p \leftarrow \mathfrak{I} \circ T_p$
end if

ACKNOWLEDGMENT

This work was supported in part by CNES, ERC(AG Twelve Labours), and ONR N00014-97-1-0839.

REFERENCES

[1] H. Burger, C. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?" in *CVPR*, 2012, pp. 2392–2399.
[2] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image restoration by sparse 3D transform-domain collaborative filtering," in *Electronic Imaging*, 2008.
[3] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *MMS*, vol. 4, no. 2, pp. 490–530, 2005.
[4] D. Zoran and Y. Weiss, "From learning models of natural image patches to whole image restoration," in *ICCV*, 2011, pp. 479–486.
[5] Y. Q. Wang and J. M. Morel, "SURE guided Gaussian mixture image denoising," *SIIMS*, vol. 6, no. 2, pp. 999–1034, 2013.
[6] H. Burger, C. Schuler, and S. Harmeling, "Learning how to combine internal and external denoising methods," in *Pattern Recognition*. Springer, 2013, pp. 121–130.
[7] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine Learning*, vol. 14, no. 1, pp. 115–133, 1994.
[8] Y. Bengio, "Learning deep architectures for AI," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[9] J. L. Lisani, A. Buades, and J. M. Morel, "How to explore the patch space." *Inverse Problems & Imaging*, vol. 7, no. 3, 2013.