DCT2net: an interpretable shallow CNN for image denoising

Sébastien Herbreteau and Charles Kervrann

***Abstract*—This work tackles the issue of noise removal from images, focusing on the well-known DCT image denoising algorithm. The latter, stemming from signal processing, has been well studied over the years. Though very simple, it is still used in crucial parts of state-of-the-art "traditional" denoising algorithms such as BM3D. Since a few years however, deep convolutional neural networks (CNN) have outperformed their traditional counterparts, making signal processing methods less attractive. In this paper, we demonstrate that a DCT denoiser can be seen as a shallow CNN and thereby its original linear transform can be tuned through gradient descent in a supervised manner, improving considerably its performance. This gives birth to a fully interpretable CNN called DCT2net. To deal with remaining artifacts induced by DCT2net, an original hybrid solution between DCT and DCT2net is proposed combining the best that these two methods can offer; DCT2net is selected to process non-stationary image patches while DCT is optimal for piecewise smooth patches. Experiments on artificially noisy images demonstrate that two-layer DCT2net provides comparable results to BM3D and is as fast as DnCNN algorithm composed of more than a dozen of layers.**

***Index Terms*—Convolutional Neural Network, image denoising, Canny edge detector, artifact removal.**

I. Introduction

IMAGE denoising is one of the most widely explored problems in computational imaging. In its most studied formulation, an image $\boldsymbol{x}$ is assumed to be corrupted by additive white Gaussian noise (AWGN), $\boldsymbol{\varepsilon}$, with variance σ^2. The observed noisy image $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{\varepsilon}$ has then to be processed to recover the original signal $\boldsymbol{x}$ while removing the noise component $\boldsymbol{\varepsilon}$.

Over the years, a rich variety of methods have been proposed to deal with this issue with inspiration coming from multiple fields. Frequency-based methods, aiming at decomposing the signal in a DCT or wavelet basis and then shrinking some transform coefficients, were considered quite early [1], [2]. This strategy, coming from compression algorithms [3], has the advantage to be both simple and fast but also the drawback of suppressing fine details in the image. An improvement of such methods consists in considering an overcomplete dictionary instead, under an assumption of sparse representation: each patch of an image is supposed to be sufficiently represented by a few vectors of an overcomplete basis [4]. In the meantime, the N(on)L(ocal)-means algorithm [5] opened the door to a new category of denoising algorithms exploiting the self-similarity assumption. Indeed, it was observed that, within the same image, similar patches are repeated in the whole image [6]. By gathering together the most similar patches and denoising them all at once, considerable gains in performance can be obtained [7], [8], [9], [10], [11], [12], [13], [14]. Most of those methods achieved state-of-the-art results until recently.

In the last five years, the development of deep learning have revolutionized computer vision, through significant accuracy improvements, denoising task being no exception. A lot of convolutional neural networks have been proposed [15], [16], [17], [18], [19], [20], [21] and they all outperformed the traditional algorithms via image training sets. Though fast and efficient, they all suffer from their lack of interpretability. Acting as "black boxes", it can be very challenging to thoroughly understand how they produce a result, which can be prohibitive for critical applications such as medical imaging.

Our work contributes to the recent trend, which builds on traditional algorithms and revisits them with a dose of deep learning, while keeping the original intuition [22], [23]. We focus specifically on the DCT denoiser [1] and show that it can be seen as a shallow CNN with weights corresponding to the DCT projection kernel and a hard shrinkage function as activation function. By training this particular CNN given external dataset, we can refine the resulting transform and boost its performance. As the so-called DCT2net inherently may create unpleasant artifacts in flat regions of the image, we apply a two-class classification procedure based on the Canny egde detector [24] applied to the image denoised with the original DCT denoiser. The classification produces a binary map that separates homogeneous regions from textured regions and contours. DCT2net is then applied to the set of pixels with more complex geometries, while DCT is applied to stationary patches (i.e., with no significant spatial gradients). Surprisingly, this strategy does not alter performances in terms of Peak Signal-to-Noise Ratio (PSNR) while visually improving the results.

The remainder of the paper is organized as follows. In Section II, we present the principle of DCT denoiser and the properties of DCT2net which has the advantage to be invariant to the level of noise in image unlike other CNN-based denoisers [15], [17], [18]. In Section III, we interpret the "pseudo" basis learned by DCT2net and show that patch denoising and aggregation are jointly performed unlike traditional patch-based denoisers. In Section IV, we analyze the behavior of DCT2net which is further mixed with the usual DCT denoiser to reduce unpleasant visual artifacts. In Section V, experimental results on datasets demonstrate that DCT2net is very fast as DnCNN, improves significantly the DCT results and is comparable to BM3D in terms of performance while remaining very simple and interpretable as the DCT denoiser.

Sébastien Herbreteau and Charles Kervrann are with Inria Rennes - Bretagne Atlantique and UMR144-CNRS Institut Curie PSL, Paris, France (e-mail: sebastien.herbreteau@inria.fr and charles.kervrann@inria.fr)

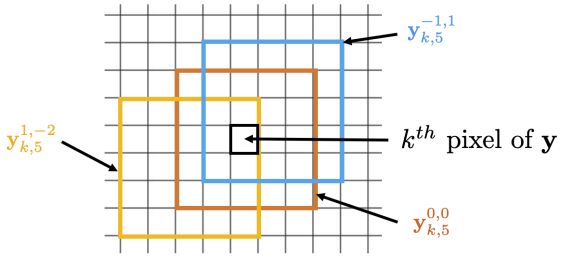


Fig. 1. Some examples of the notation $\boldsymbol{y}_{k,p}^{i,j}$.

II. From popular DCT denoising to DCT2net

In what follows, the vector representation of an image is adopted. A noisy 2D image $\boldsymbol{y}$ composed of n pixels is formally represented by a vector of $\mathbb{R}^n$.

A. Traditional DCT denoiser

In its most mature formulation for image denoising [1], the DCT denoiser proceeds on small overlapping patches accross the image. Each patch of size $p \times p$ is denoised independently, so that each pixel is in fact denoised p^2 times. For each pixel, the final denoised value is then obtained by averaging those p^2 estimators. Typical values for p are powers of 2 (generally 8 or 16) for practical reasons in the computation of the fast discrete cosine transform. However, we focus here on the case where p is an odd number, without loss of generality.

For the sake of representation, we denote by $\boldsymbol{y}_{k,p}^{i,j}$ the $p \times p$ patch, in the vector form, for which the central pixel, is located j pixels at the right of the k^{th} pixel of $\boldsymbol{y}$ and i pixels beneath it (see Fig. 1). Note that i and j can be negative numbers. Let us denote $q = \lfloor \frac{p}{2} \rfloor$ (i.e. closest integer less than or equal to $p/2$). The DCT denoiser denoted F can then be expressed as:

$$F(\boldsymbol{y})_k = \frac{1}{p^2} \sum_{i=-q}^{q} \sum_{j=-q}^{q} [\boldsymbol{P} \varphi_\lambda (\boldsymbol{P}^{-1} \boldsymbol{y}_{k,p}^{i,j})]_{s(i,j)} \qquad (1)$$

where $\boldsymbol{P}$ is a matrix of size $p^2 \times p^2$, φ_λ is the hard shrinkage function $\varphi_\lambda(x) = x \times \mathbf{1}_{\mathbb{R} \setminus [-\lambda, \lambda]}(x)$, and $s(i,j) = (q-i)p + q - j + 1$. According to [1], the most appropriate choice for λ is 3σ.

By definition of DCT, the mathematical expression of the coefficient of the matrix $\boldsymbol{P}$ at row $i = xp + y + 1$ and column $j = up + v + 1$ for $(x, y, u, v) \in [\![0, p-1]\!]^4$ is given by:

$$\boldsymbol{P}_{i,j} = \frac{2}{p} \alpha(u) \alpha(v) \cos\left[\frac{(2x+1)u\pi}{2p}\right] \cos\left[\frac{(2y+1)v\pi}{2p}\right] \qquad (2)$$

where $\alpha(u) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u = 0 \\ 1 & \text{otherwise} \end{cases}$ is a normalizing scale factor to make the transformation orthonormal. The columns of the matrix $\boldsymbol{P}$ are, in fact, the basis in which the signal is decomposed (see Fig. 4a). The matrix $\boldsymbol{P}$ is considered as a basis, in the sense that every signal (represented as a vector) can be decomposed in a unique way as a linear combination of the columns of $\boldsymbol{P}$. Alternatively, the term "dictionary" is also used in image processing. In the case of DCT, this basis has the particularity of being orthonormal, which implies that $\boldsymbol{P}^{-1} = \boldsymbol{P}^T$ where the superscript T denotes the transpose operator. The elements of this basis are generally ordered, in the zig-zag pattern, from the smoothest vector to the one containing the highest frequencies (see Fig. 4a). This ordering is very useful for applications in compression as frequencies higher than a certain threshold are typically canceled.

A small improvement of [1], called adaptive aggregation and inspired from [7], was proposed in [26]. The idea is to give higher weight to patches that have a sparser representation in the DCT domain, enabling to reduce the ringing effects near edges. The expression of the improved DCT denoiser then becomes:

$$F(\boldsymbol{y})_k = \frac{1}{W_k} \sum_{i=-q}^{q} \sum_{j=-q}^{q} w_{i,j,k} [\boldsymbol{P} \varphi_\lambda (\boldsymbol{P}^{-1} \boldsymbol{y}_{k,p}^{i,j})]_{s(i,j)} \qquad (3)$$

with $w_{i,j,k} = (1 + \|\varphi_\lambda(\boldsymbol{P}^{-1} \boldsymbol{y}_{k,p}^{i,j})\|_0)^{-1}$, where $\|.\|_0$ denotes the ℓ_0 pseudo-norm that counts the number of non-zero entries and $W_k = \sum_{i=-q}^{q} \sum_{j=-q}^{q} w_{i,j,k}$. We used this latter expression to derive DCT2net.

B. DCT2net: a CNN representation of a DCT denoiser

Interestingly, one of the easiest implementation of a DCT denoiser, as formulated in (3), can be done with a neural network. Indeed, all operations involved can be interpreted in terms of convolutions with a hard shrinkage function as activation function.

Our shallow CNN is first composed of a convolutional layer with kernel size $p \times p$ leading to p^2 output channels. Figure 2 shows such an architecture when $p = 5$. Note that the weights involved in kernels are to be found in the matrix $\boldsymbol{P}^{-1}$: each row is associated with one convolutional kernel. The action of this layer can be summed up as follows: each pixel is replaced by the patch formed by its neighborhood, after transformation by $\boldsymbol{P}^{-1}$. Then, the hard shrinkage function is applied element-wise. Afterwards, a 1×1 convolution layer operates on those patches where the weights correspond to the elements of the matrix $\boldsymbol{P}$. In order to compute the adaptive aggregation, we have to generate a weight map from the first layer computing the values $w_{i,j,k}$ in (3). This latter is used to balance the features resulting from the second layer by channel-wise multiplication. The weighted pixels are then repositioned at their corresponding locations and then aggregated by summation. This can be implemented with the help of a last convolutional layer where the values of weights are either 0 or 1. Note that, for computational efficiency, a 2D transposed convolution is recommended. Finally, a normalization by W_k (see (3)) is performed by dividing the last layer by the weight map, beforehand convolved by a kernel composed of ones.

We want to stress that our DCT2net is the strict implementation of the formulation in (3). Equipped with the correct weights given by the definition of the DCT in (2), it exactly produces the same results as those obtained with the traditional implementation.

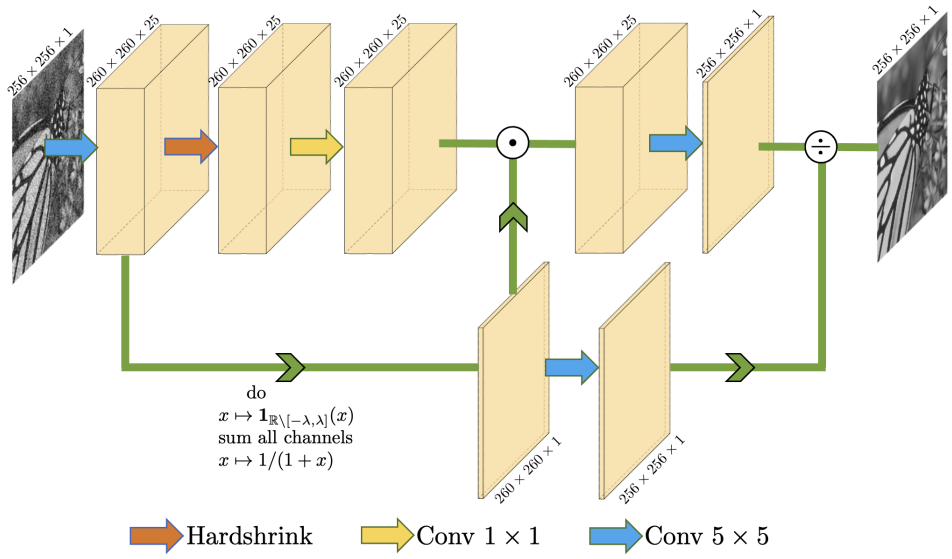


Fig. 2. Architecture of DCT2net for a patch size $p = 5$.

C. Improvement of the transform

As it is usually done with neural networks, we can train our DCT2net on an external dataset composed of N pairs of noise-free and noisy images $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$ to improve the underlying transform. More precisely, our objective is to solve the following optimization problem:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{N} \|F_{\boldsymbol{P}}(\boldsymbol{y}_i, \sigma_i) - \boldsymbol{x}_i\|_2^2 \tag{4}$$

where $F_{\boldsymbol{P}}$ denotes the network (Fig. 2) and $\boldsymbol{P}$ gathers the unknown parameters.

To that extent, we need to restrict the model to the original transform where only one matrix is involved (the other one being its inverse) and where the other convolutions of the network composed of 0 and 1 are frozen. This can be achieved with the help of modern machine learning libraries such as Pytorch for which automatic differentiation can be kept on for complex operations such as matrix inversion but also deactivated for some layers. Nevertheless, the thresholding operation must be slightly adapted in a context of gradient descent where differentiation is needed. Thus, we replace the function $\mathbf{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}$ by $\zeta_{m,\lambda}(x) = \frac{x^{2m}}{x^{2m}+\lambda^{2m}}$ with $m \in \mathbb{N}^*$. This choice is legitimate as the sequence of functions $(\zeta_{m,\lambda})_{m\in\mathbb{N}^*}$ converges pointwise to $\mathbf{1}_{\mathbb{R}\setminus[-\lambda,\lambda]}$. The hard shrinkage function φ_λ then becomes $\varphi_{m,\lambda}(x) = \frac{x^{2m+1}}{x^{2m}+\lambda^{2m}}$. Figure 3 shows how close this approximation is from the original hard shrinkage function with ever growing values of m. By the way, this approximation is adopted only during the training phase for facilitating the optimization process. To stick with the original DCT denoiser, we use the original hard shrinkage function for the testing phase that gives the same results in terms of PSNR with no noticeable visual differences for the denoised images. It is worth noting that the use of the original hard shrinkage function instead of our differentiable approximation for the training phase does not work in our case, leading to a poor suboptimal local minimum, even though this activation function is available in most modern machine learning libraries. It is likely that this disappointing behavior is due to the discontinuity of the original function.

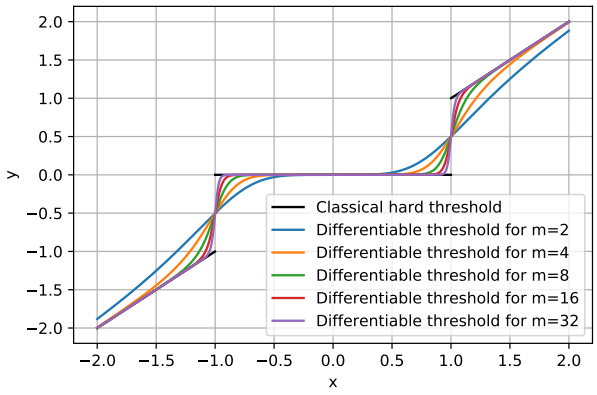
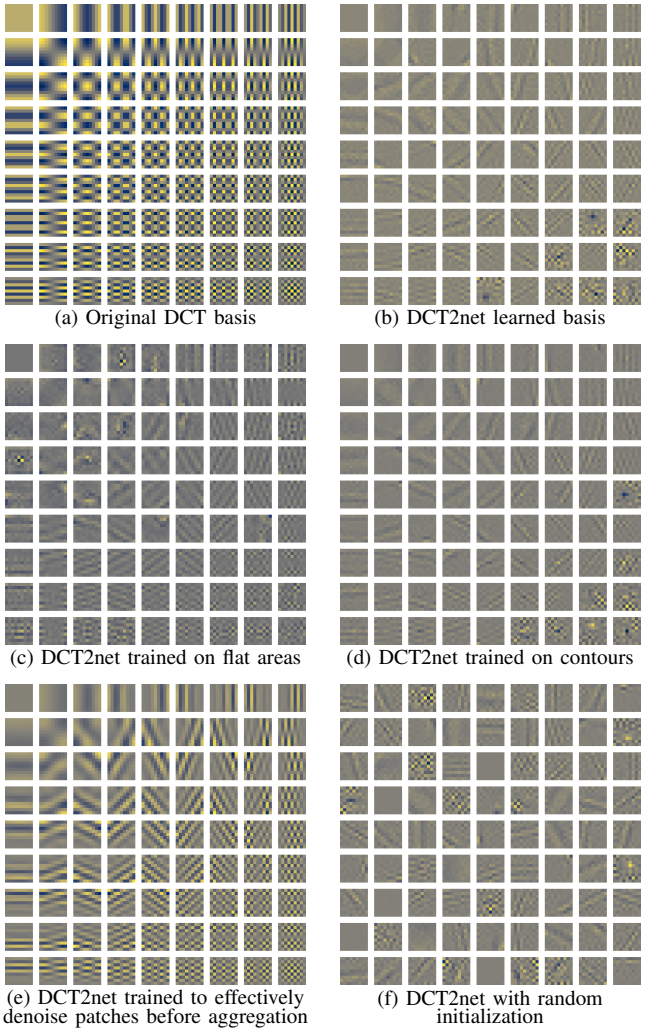


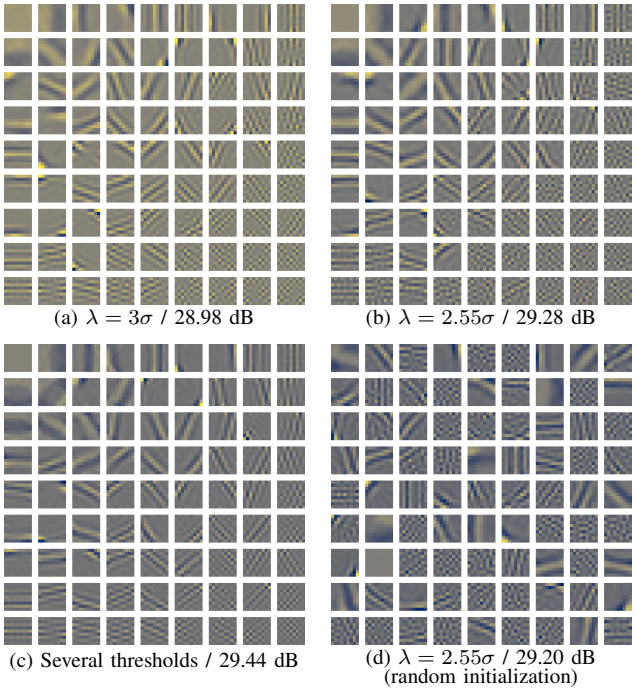
Fig. 3. Approximation of the hard thresholing function φ_λ by the sequence of differentiable functions $\varphi_{m,\lambda}$ for $\lambda = 1$ and $m \in \{2, 4, 8, 16, 32\}$.

Finally, as recommended in [1], the threshold parameter λ is set to 3σ to significantly remove noise. But the choice of the multiplicative constant in front of σ is actually of little importance and any other constant would produce same results as long as $\boldsymbol{P}$ can adapt itself. Indeed, for two levels of threshold λ and λ', we have $\varphi_\lambda(x) = \frac{\lambda}{\lambda'}\varphi_{\lambda'}(\frac{\lambda'}{\lambda}x)$. In particular, for two choices of multiplicative constant c and c', $\varphi_{c\sigma}(x) = \frac{c}{c'}\varphi_{c'\sigma}(\frac{c'}{c}x)$, hence $\boldsymbol{P}\varphi_{c\sigma}(\boldsymbol{P}^{-1}\boldsymbol{y}) = \boldsymbol{Q}\varphi_{c'\sigma}(\boldsymbol{Q}^{-1}\boldsymbol{y})$ with $\boldsymbol{Q} = \frac{c}{c'}\boldsymbol{P}$ This means that, in theory, choosing a value other than 3 would result in estimating the same transform, up to a multiplicative constant, and with exactly the same denoising performance. Appendix A gives more details about the choice of threshold, studying more particularly the case where multiple thresholds are used.



(a) Original DCT basis (b) DCT2net learned basis (c) DCT2net trained on flat areas (d) DCT2net trained on contours (e) DCT2net trained to effectively denoise patches before aggregation (f) DCT2net with random initialization

Fig. 4. Different bases in which patches are decomposed and thresholded for image denoising.



(a) $\lambda = 3\sigma$ / 28.98 dB (b) $\lambda = 2.55\sigma$ / 29.28 dB (c) Several thresholds / 29.44 dB (d) $\lambda = 2.55\sigma$ / 29.20 dB (random initialization)

Fig. 5. Orthonormal bases learned by DCT2net by addition of a regularization term. The threshold used is indicated (learned by optimization process when different from 3σ) as well as the average PSNR on Set12 for $\sigma = 25$ with each of these orthonormal bases. By way of comparison, the average PSNR on Set12 for the unconstrained DCT2net with patch size 9×9 is 29.57 dB. The basis exposed with several thresholds corresponds actually to an orthogonal basis with only one threshold as shown in appendix C.

Note that, in practice, optimizing over the set of invertible matrices $\mathcal{GL}_{p^2}(\mathbb{R})$ in (4) is not an issue and the problem can be treated through stochastic gradient descent without specific precaution. It is attributable to the fact that $\mathcal{GL}_{p^2}(\mathbb{R})$ is dense in $\mathcal{M}_{p^2}(\mathbb{R})$ but $\mathcal{M}_{p^2}(\mathbb{R})\backslash\mathcal{GL}_{p^2}(\mathbb{R})$ is not.

III. A NON-INTUITIVE LEARNED TRANSFORM

What is particularly attractive in our model is that we can easily display the learned transform and thus have a direct visual intuition of what the network has learned. Once DCT2net has been trained on an external dataset, a "pseudo" basis is derived, which is not orthonormal, but presumably more appropriate to encode non-stationary signals than the conventional DCT. Figure 4 shows a visual comparison of the learned bases in different contexts for patches of size 9×9.

A. On the orthonormality of the learned transform

In the definition of DCT2net (3), we impose no orthonormality constraint to learn the basis. Therefore, it is no wonder that, during the building of the matrix $\boldsymbol{P}$ through stochastic gradient descent, the property of orthonormality gets lost. One way [1] to address this issue is to add a regularization term that encourages orthonormality in the optimization process. The problem amounts to solving:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{N} \|F_{\boldsymbol{P}}(\boldsymbol{y}_i, \sigma_i) - \boldsymbol{x}_i\|_2^2 + \beta \|I - \boldsymbol{P}^T\boldsymbol{P}\|_1 \quad (5)$$

where $\beta \geq 0$ is the regularization parameter. We consider here the ℓ_1 norm, defined for a matrix $\boldsymbol{A}$ by $\|\boldsymbol{A}\|_1 = \sum |a_{i,j}|$, but the ℓ_2 norm can be used instead. One can prove that this optimization problem with a penalty term corresponds to an underlying constraint problem of the form:

$$\begin{aligned} \boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} & \sum_{i=1}^{N} \|F_{\boldsymbol{P}}(\boldsymbol{y}_i, \sigma_i) - \boldsymbol{x}_i\|_2^2 \\ \text{s.t.} \quad & \|I - \boldsymbol{P}^T\boldsymbol{P}\|_1 \leq t \end{aligned} \quad (6)$$

which makes explicit the constraint of "close-orthonormality". Note that the parameter t depends both on β and on the data $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$.

It is worth noting that the resulting matrix $\boldsymbol{P}^*$ is not guaranteed to be orthonormal whatever the parameter β is. To derive an orthonormal matrix from $\boldsymbol{P}^*$, we can select its nearest orthonormal matrix $\boldsymbol{P}_{\text{ortho}}$ in the Frobenius norm sense. The unique solution is given by $\boldsymbol{P}_{\text{ortho}} = \boldsymbol{U}\boldsymbol{V}^T$ where $\boldsymbol{U}$ and

[1] For a direct technique to derive an orthonormal matrix, with similar results compared to the regularization form, see appendix B.

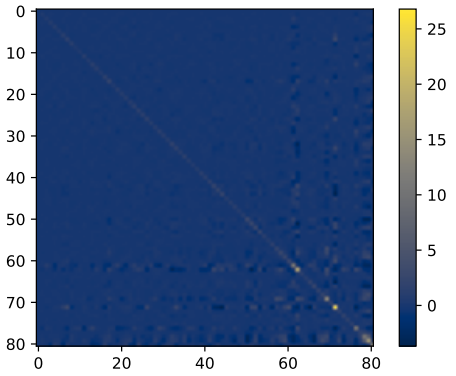


Fig. 6. Representation of the matrix $\boldsymbol{P}^T\boldsymbol{P}$ where $\boldsymbol{P}$ denotes the transform learned by DCT2net for patches of size 9×9. If $\boldsymbol{P}$ were orthogonal, $\boldsymbol{P}^T\boldsymbol{P}$ would be equal to an invertible diagonal matrix. This is not strictly the case here, but we can notice that the elements outside the diagonal are very close to 0. Moreover, the diagonal represents an important weight of this matrix as $\sum_i q_{i,i}^2 / \sum_{i,j} q_{i,j}^2 \approx 60\%$ where the $q_{i,j}$ designate the coefficients of $\boldsymbol{P}^T\boldsymbol{P}$.

$\boldsymbol{V}$ are the matrices from the singular value decomposition of $\boldsymbol{P}^* = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.

However, adding a constraint of orthonormality limits the expressivity of the network and we observed that the denoising performance was not as good as the regularization-free solution (see Fig. 5). Moreover, the choice of the regularization parameter β can be challenging as it needs to be adapted for each patch size. For all those reasons, we decided not to retain a solution with an orthonormal matrix. In spite of this, the matrix $\boldsymbol{P}$ learned by DCT2net is quite close to be orthogonal, even if no constraint has been imposed. This is illustrated in Fig. 6 which displays the matrix $\boldsymbol{P}^T\boldsymbol{P}$ for patches of size 9×9. We can notice that the non-diagonal elements are close to zero, which is expected for a matrix close to be orthogonal. In appendix C, we show how orthogonal and orthonormal matrices are linked, stating that if $\boldsymbol{P}$ is orthogonal, there exists an orthonormal matrix $\boldsymbol{Q}$ that would give exactly the same results in DCT2net as long as we set a different threshold by element of the basis.

Finally, as regards the initialization for the stochastic gradient descent, the original DCT basis (see (2)) is considered by default. Considering random initializations such as Xavier initialization which is common in deep neural networks, lead to similar bases, even though the time for convergence is slightly more important. The convergence to the same solution whatever the initialization is a very good news, suggesting that optimizing the underlying non-convex problem is tractable.

B. DCT2net does not denoise patches

When displaying the learned basis on a popular image dataset such as BSD400 [27], one may be surprised. Interestingly, this basis, in which each patch is decomposed, is much more disorganized than the original DCT basis. One may doubt that the DCT2net basis denoises better patches as it contains no clear pattern. As a matter of fact, applying this basis does not denoise patches but rather degrades them even more as shown in Fig. 7. The main reason is that the network actually denoise image patches and performs aggregation at once, making it difficult to understand why such a basis improves the PSNR value of the restored image.

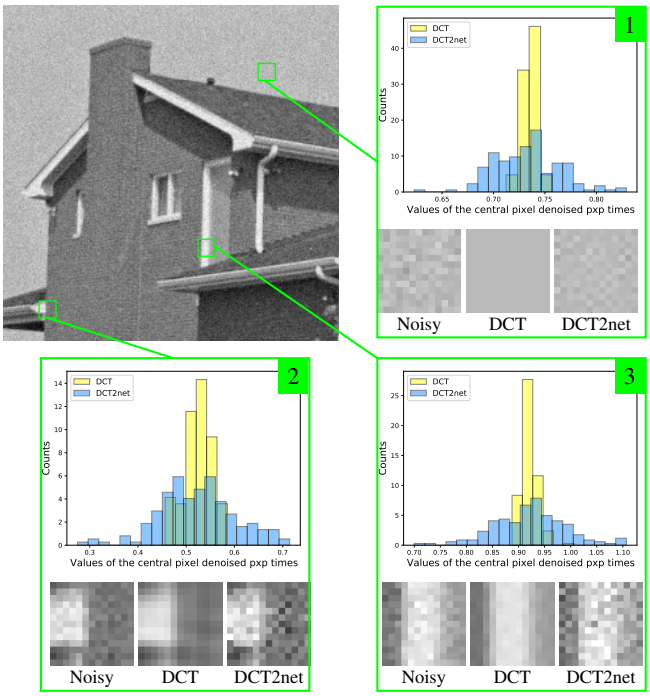


Fig. 7. For each noisy patch of size 13×13 extracted from *House* image corrupted by AWGN with $\sigma = 10$, its denoised version is displayed when processed by the original DCT and by the transform learned by DCT2net. The patches produced by DCT2net are very noisy compared to the original patches and patches denoised with DCT. Histograms show a comparison of the p^2 denoised values for the central pixel after transformation in each of the p^2 patches it belongs to ($p = 13$). The variance of pixels intensities is higher with DCT2net.

We observed that, for a given noisy pixel k belonging to p^2 patches, the p^2 "denoised" versions of pixel k with DCT2net have a very high variance when compared to the p^2 denoised values obtained with the DCT denoiser, for which all p^2 denoised values are generally almost all the same, as illustrated in Fig. 7. Nevertheless, after the adaptive aggregation step, the pixels denoised by applying the learned transform are closer, in average, to the ground truth ones. This is a counter-intuitive result that questions our preconceptions on denoising. This suggests that the final aggregation step is not a basic post-processing step but plays an important role in denoising, as confirmed below.

C. Constraining DCT2net to effectively denoise patches is an unsuccessful strategy

As the strategy followed by DCT2net is counter-intuitive and hardly comprehensible for a human brain, we tried to constrain the learned transform to effectively denoise patches. In this study, aggregation is performed in a second step with conventional weighted averages. This can be done in practice by cutting our neural network represented in Fig. 2 after the two first convolutional layers, so that the output $F_{\boldsymbol{P}}(\boldsymbol{y}_i)$ is of

Noisy / 20.17 dB | Binary classification map | DCT / 31.18 dB | DCT2net / 32.20 dB | DCT/DCT2net / 32.26 dB

Fig. 8. Application of the proposed procedure to get a partition of the noisy image *House*. Pixels belonging to white areas after classification are denoised with DCT2net and the others with a traditional DCT denoiser. Results (in PSNR) are given for each method for a noise level $\sigma = 25$.

TABLE I
THE AVERAGE PSNR (DB) RESULTS OF TWO DIFFERENT TRANSFORMS ON PATCHES OF SIZE 15×15 OF SET12 CORRUPTED WITH WHITE GAUSSIAN NOISE AND $\sigma = 15$, 25 AND 50.

Methods	$\sigma = 15$	$\sigma = 25$	$\sigma = 50$
Before aggregation			
DCT	27.74	25.19	21.92
DCT2net trained on patches	28.18	25.98	22.97
After adaptive aggregation			
DCT	31.08	28.53	25.37
DCT2net trained on patches	30.90	28.66	25.65

size $(H - p + 1) \times (W - p + 1) \times p^2$. For a clean image $\boldsymbol{x}_i$, we can compute its patch representation $\Pi(\boldsymbol{x}_i)$ of size $(H-p+1)\times(W-p+1)\times p^2$ as well and solve the following optimization problem:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{N} \|F_{\boldsymbol{P}}(\boldsymbol{y}_i, \sigma_i) - \Pi(\boldsymbol{x}_i)\|_2^2. \qquad (7)$$

Figure 4e shows the learned transform $\boldsymbol{P}^*$ that effectively denoise patches. The resulting matrix $\boldsymbol{P}^*$ is much more natural and interestingly very close the original DCT basis. The gain in PSNR on patches of this new transform was evaluated on the Set 12 dataset. The results reported in Table I show that the learned transform produces systematically a higher PSNR in average for the patches than the traditional DCT. One could expect that this transform would outperform DCT after the aggregation step. Unfortunately, this is not the case. Once the transform has been re-used in our DCT2net shown in Fig. 2 with the adaptive aggregation integrated, the expected boost compared to traditional DCT in terms of PSNR significantly decreases. The difference of PSNR is insignificant, with a visually less attractive result. We can notice an interesting phenomenon for $\sigma = 15$: from better denoised patches, our learned transform fails to outperform the traditional DCT after any classical aggregation technique. This counter-intuitive result confirms, once again, that the aggregation step is equally important as denoising patches.

To go beyond DCT, the key issue is to follow a non-intuitive path that consists in degrading the patches and performing aggregation step to rearrange everything and produce a high-quality denoised image.

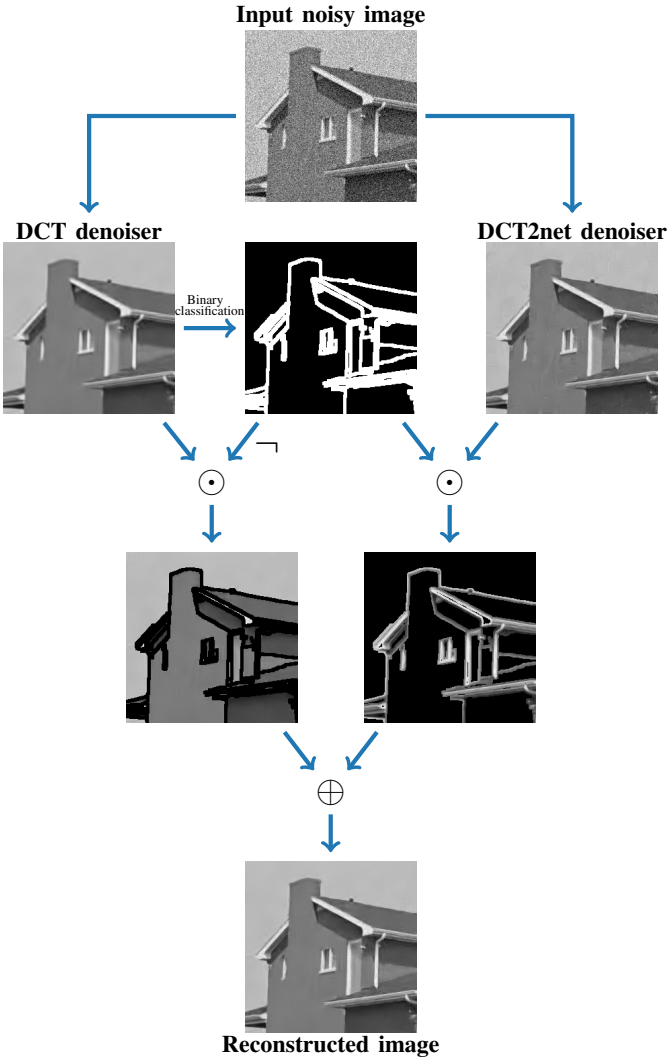


Fig. 9. Hybrid denoising scheme. A first rough denoising is performed by DCT to improve the classification procedure. Then the denoised image is recycled on the flat areas while DCT2net denoises the contours.

IV. DCT2NET MIXED WITH DCT TO REDUCE UNPLEASANT VISUAL ARTIFACTS

Even though DCT2net produces high PSNR values as BM3D [7], the visual results can be surprisingly not as good as expected, especially in flat regions in images. This is due to the emergence of structured unpleasant artifacts in those regions

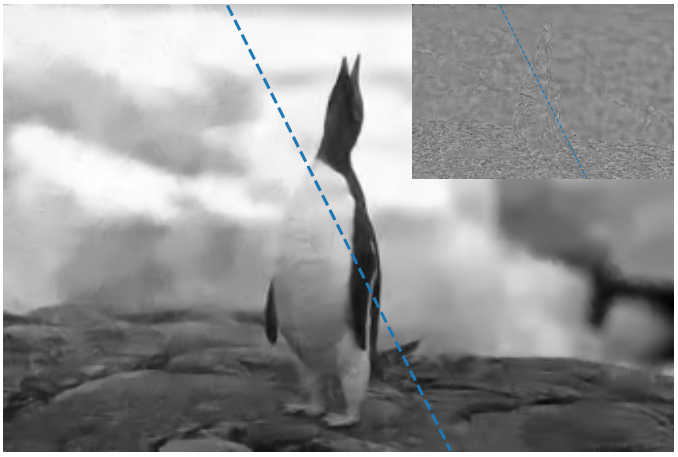


Fig. 10. Image from BSD68 denoised with DCT2net producing unpleasant visual artifacts (left) and denoised using both DCT and DCT2net in collaboration (right) for $\sigma = 25$. The difference between the noise-free image and the denoised images is also provided, highlighting these artifacts.

that are extremely eye-catcher. Figure 10 shows an example of those artifacts that are inherent to our method. They are difficult to characterize and very different from what we would get with a non-adaptive DCT denoiser. The visual impression is as if the recovered image had been scratched in several locations. These undesirable artifacts were probably promoted by blind stochastic gradient descent, to produce the best PSNR value in average. It is likely that this blind choice originates from a trade-off between denoising flat regions and textures. The problem was not solved by considering an adequate loss function [25] or by adopting a multi-scale scheme [26].

To tackle this problem, we combine below the performance of both transforms, that is the original DCT and the transform learned by DCT2net. While the original DCT performs very well in flat regions, DCT2net recovers details more efficiently in the vicinity of contours and in fine textured regions. Our idea is then to classify the pixels into two classes and to apply the most appropriate denoiser at each pixel. In what follows, we show how a binary map can be obtained, telling us which pixels are to be denoised with DCT2net or DCT. Figure 8 illustrates an example of such a procedure applied to the *House* image.

The noisy image must be roughly denoised beforehand in order to robustly detect the flat regions, textured regions and contours. The DCT denoiser (3) is appropriate in our case, as illustrated in Figure 9. It has the advantage of being particularly cost-efficient and nearly parameter-free. The resulting denoised image is also re-used to produce the final image, as illustrated in Fig. 9, saving time and resources. Interestingly, letting the traditional DCT denoiser operate on a large majority of pixels of the noisy image does not alter the PSNR values in our experiments. As DCT produces smooth images in homogeneous regions, the artifacts are removed and the visual result is enhanced considerably.

a) Classification based on Canny edge detector: Multiple choices of classification techniques are possible to separate the flat regions from the contours and textured areas. A high precision in the classification is not required as our goal is to isolate parts of the image that are susceptible to contain artifacts after denoising with DCT2net.

TABLE II
THE AVERAGE PSNR (DB) RESULTS OF OUR DCT/DCT2NET METHOD ON SET12 CORRUPTED WITH WHITE GAUSSIAN NOISE AND $\sigma = 20$ FOR DIFFERENT SIZES OF DILATION KERNEL.

Size of dilation	3	5	7	9	11	∞
DCT/DCT2net	30.58	30.67	30.69	30.70	30.71	30.75

The classification problem into two classes can be achieved with an efficient traditional technique: the Canny edge detector [24]. This method uses Sobel filters in both horizontal and vertical direction at its core to compute the gradient for each pixel. The direction of edges is then analyzed to remove any unwanted pixels which may not constitute contours. Finally, an hysterisis thresholding is used to decide which pixels, detected positively in the first instance, are actually edges. This last step is based on the spatial analysis of connectivity, ensuring some coherence in the final classification map. To enlarge the support of edges found by this Canny edge detector, we apply a simple dilation operation in the end.

In practice, the image is preliminary slightly smoothed with a unit standard deviation Gaussian filter. The lower and upper thresholds involved in the Canny edge detector are set respectively to 0.1 and 0.2 (values of pixels being in the interval $[0, 1]$). These thresholds are set once and for all and are not changed in all experiments. Finally, the dilation operation is performed using a kernel of size 5×5. This choice of size of dilation kernel was motivated by its good performance in terms of PSNR, without scarifying the visual quality depending on the amount of artifacts (which is the case for larger sizes). Table II reports the influence of the size of the dilation operation on the PSNR values for $\sigma = 20$ on the Set 12 dataset composed of 12 widely used images for denoising. Unsurprisingly, the larger the dilation filter (that is, the more pixels are processed with DCT2net), the higher the PSNR value is. It can be noticed that considering small or large kernels does not affect the PSNR values significantly. If the dilation is very large, it amounts to applying DCT2net to all pixels in the image.

b) Classification based on Total Variation: Alternatively, classification can be performed by applying the local Total Variation (TV). This amounts to computing the sum of gradients on small windows, which is expected to be low in flat regions and high on edges. After computing the value of the TV for every pixel, a local-TV map is derived where values are all the more important as the original noisy pixel belongs to a complex geometry, that is edges or texture. By defining an arbitrary threshold, it is possible to partition the image into two distinct components: *high gradient* and *low gradient* pixels where DCT2net and DCT are applied, respectively.

c) Comparisons of classification methods: Figure 11 shows the computed binary masks on the image *Lena* for the two aforementioned classification techniques as well as the recomposed denoised image using both DCT2net/DCT denoiser.

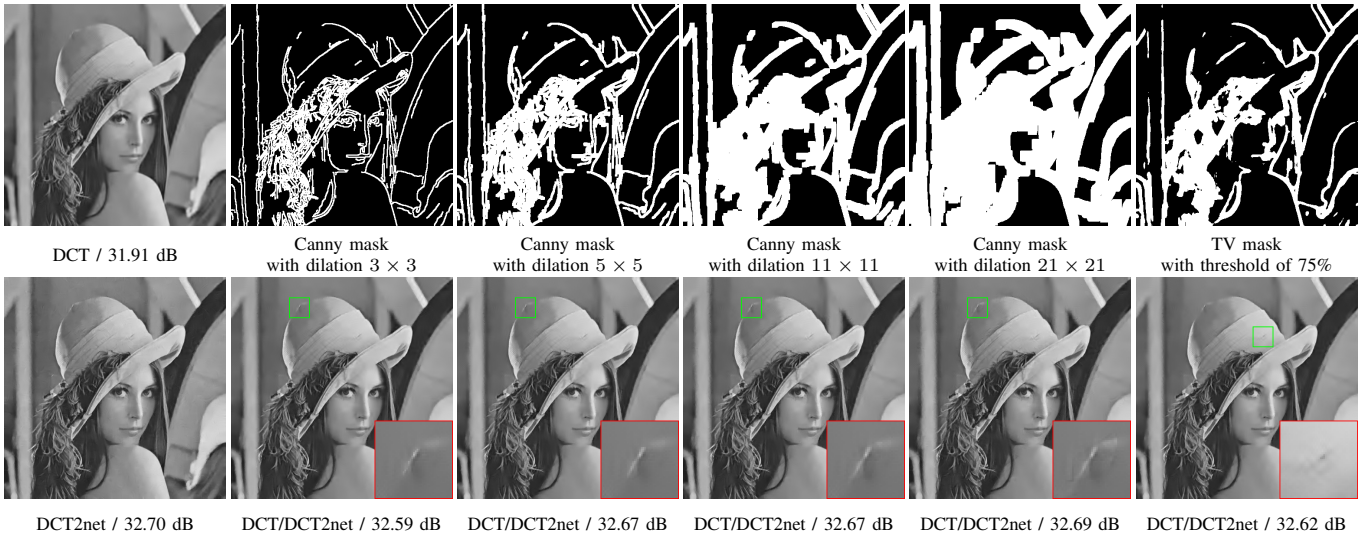


Fig. 11. Some examples of the classifications based on Canny edge detector and Total Variation on noisy *Lena* for $\sigma = 20$.

We can notice that the technique based on the Canny edge detector gives a coherent classification where almost all contours are detected. When dilating more and more those contours with larger and larger dilation kernels, the PSNR improves as more pixels are processed with DCT2net. However, there is a risk of generating unpleasant artifacts near contours as it is the case in our example with a dilation kernel of size 21×21 (see Fig. 11). That is why, we set the size of this kernel to 5×5 once and for all which is a good balance between performance based on PSNR value and subjective visual perception.

Compared to the classification based on the Canny edge detector, the TV-based one is more limited. Indeed, some edges are missing. Worse still, some isolated white blocks appear in the background. It is very troublesome as those isolated zones will create further unpleasant artifacts at the end, as shown on Figure 11. It appears that the *denoising styles* of those two denoisers are not compatible on similar zones. The human eye quickly notices a lack of coherence in the *denoising tone* which is prejudicial to the visual quality. We observed similar issues for all size of windows for the TV computation and for all thresholds. Those critical drawbacks are prohibitive in the application of a such a method and we decided to focus on the Canny edge detector in our experiments.

V. Experiments

In this section, we describe the experiments conducted to train our model DCT2net. Moreover, we provide comparisons with traditional and deep-learning-based state-of-the-art algorithms. The code and pre-trained models can be downloaded here: https://github.com/sherbret/DCT2net/.

A. Training Settings

We trained our DCT2net on 400 gray-scale images from the Berkeley segmentation dataset (BSDS) [27] where synthetic Gaussian noise with zero mean and a random standard deviation σ taken in $\in [1, 55]$ was added in order to create our pairs $(\boldsymbol{x}_i, \boldsymbol{y}_i)_{i\in\{1,\dots,N\}}$ with $N \gg 400$ (the $\boldsymbol{x}_i$ are redundant). Contrary to numerous deep learning models, our network can adapt to the level of noise as the differentiable hard shrinkage function depends on σ, so that we train our model only once for all levels of noise at the same time.

During training, we randomly sample cropped images from the training set of size 128×128 with a mini-batch size of 32. We use horizontal and vertical flipping as well as random rotations $\in \{0°, 90°, 180°, 270°\}$ as further data augmentation. In total, 400×665 overlapping patches from 400 clean images are used for training. The mean squared error was used as loss function and we used Adam optimizer [28]. The learning rate was set to 10^{-3} and decreased exponentially to 10^{-5} during the 15 epochs required for convergence. Note that we initialized the weights of our networks according to the original discrete cosine transform given by (2). Initializing the weights randomly, for example with a Xavier initialization as it is usually done with deep neural networks, only slows down the time for convergence. As for the parameter m specifying the degree of approximation of the hard shrinkage function φ_λ, we took $m = 32$. Training a model took approximately 8 hours with a GeForce RTX 2080 Ti.

Note that a small improvement of our hybrid solution DCT/DCT2net can be obtained by training DCT2net only on parts of images where the learned transform will be applied. In practice, this is done by pre-computing binary masks b_i according to the classification proposed for every image of the external dataset and solving:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{N} \|b_i \odot (F_{\boldsymbol{P}}(\boldsymbol{y}_i, \sigma_i) - \boldsymbol{x}_i)\|_2^2 \qquad (8)$$

where $F_{\boldsymbol{P}}$ designates the network and $\odot$ is the Hadamard product.

We recall the parameters chosen for the Canny edge detector: lower and upper thresholds are set once and for all respectively to 0.1 and 0.2 and a supplementary dilation operation is performed using a kernel of size 5×5.

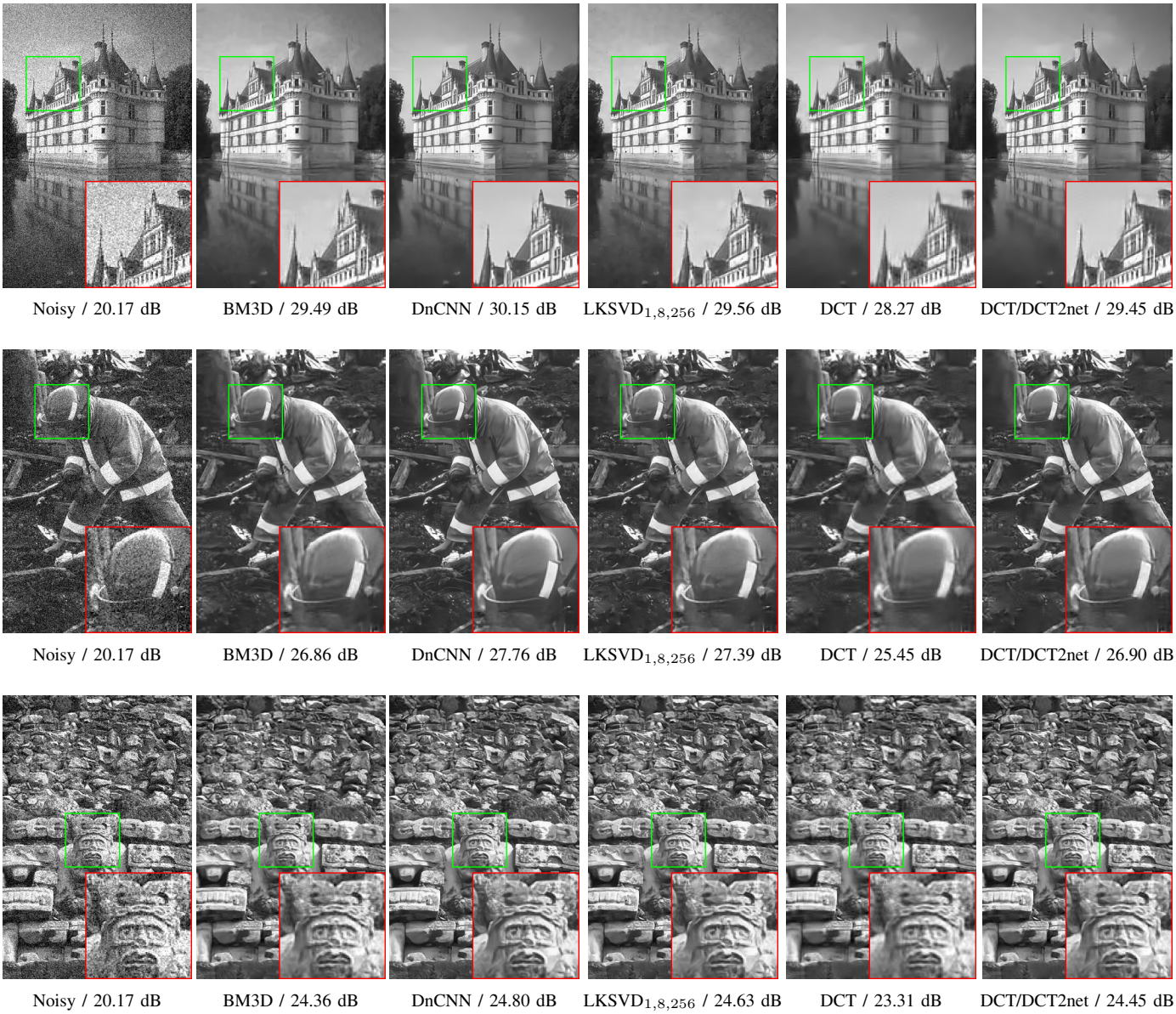


Fig. 12. Denoising results (in PSNR) of some images from BSD68 corrupted with white Gaussian noise and $\sigma = 20$.

B. Results on test datasets

We tested the denoising performance of our architecture on two well-known datasets: Set 12 and BSD68. According to our experiments, the best model for DCT/DCT2net in terms of performance (i.e., PSNR) and subjective visual quality is obtained with a patch size of 13. Larger sizes of patch only bring negligible enhancement for a complexity much more important.

Tables III and IV compare the performance of traditional and deep-learning-based state-of-the-art algorithms with our model. We compare our DCT2net with BM3D [7] and PEWA [11], both state-of-the-art traditional methods that exploit self-similarity and DCT decomposition. We also compare DCT2net to related algorithm Deep K-SVD [22]. Scetbon et al. [22] proposed multiple models that depend on the patch size, the dictionary size and the number of denoising steps. In what follows, we consider the smallest model for which an implementation is given by the authors and which is denoted LKSVD$_{1,8,256}$. We performed the training by ourselves for each noise level, as no pretrained models were supplied by the authors.

We can notice that DCT2net achieves comparable performances with state-of-the-art traditional algorithms on both Set12 and BSD68 datasets, outperforming its original counterpart DCT[2] [1]. Unlike BM3D and PEWA and other algorithms such as NL-Bayes [9], DCT2net is a very simple one-pass algorithm, able to produce similar performances to Deep K-SVD for high noise level while it is not trained specifically to address such challenging situations.

Beyond the performance assessed with the PSNR criterion, nothing can replace the subjective assessment of a human eye. On this criterion, our DCT/DCT2net can hold its own against

[2]The non-adaptive version of DCT denoiser was considered as it produced a slightly higher PSNR, despite its poor subjective visual quality.

TABLE III
THE PSNR (DB) RESULTS OF DIFFERENT METHODS ON SET12 CORRUPTED WITH WHITE GAUSSIAN NOISE AND $\sigma = 15$, 25 AND 50. THE BEST TOW RESULTS ARE HIGHLIGHTED IN RED AND BLUE COLORS, RESPECTIVELY.

Images	*C.man*	*House*	*Peppers*	*Starfish*	*Monarch*	*Airplane*	*Parrot*	*Lena*	*Barbara*	*Boat*	*Man*	*Couple*	*Average*
Noise Level	$\sigma = 15$												
BM3D [7]	31.91	34.93	32.69	31.14	31.85	31.07	31.37	34.26	33.10	32.13	31.92	32.10	32.37
PEWA [11]	31.88	34.72	32.64	30.85	31.83	31.04	31.29	34.09	32.73	31.90	31.81	31.88	32.22
DnCNN [15]	32.61	34.97	33.30	32.20	33.09	31.70	31.83	34.62	32.64	32.42	32.46	32.47	32.86
$\text{LKSVD}_{1,8,256}$ [22]	32.07	34.26	32.79	31.62	32.49	31.37	31.62	34.03	31.84	31.97	32.07	31.85	32.33
DCT [1]	30.77	33.56	31.65	30.09	30.62	30.17	30.64	33.44	31.63	31.36	31.04	31.20	31.35
DCT2net	31.60	34.31	32.57	31.04	31.69	30.99	31.36	33.96	31.81	31.95	31.97	31.89	32.10
DCT/DCT2net	31.49	34.30	32.52	30.88	31.60	30.93	31.27	33.93	31.90	31.82	31.78	31.78	32.02
Noise Level	$\sigma = 25$												
BM3D [7]	29.45	32.85	30.16	28.56	29.25	28.42	28.93	32.07	30.71	29.90	29.61	29.71	29.97
PEWA [11]	29.48	32.77	30.30	28.13	29.13	28.41	28.90	31.89	30.28	29.65	29.50	29.48	29.83
DnCNN [15]	30.18	33.06	30.87	29.41	30.28	29.13	29.43	32.44	30.00	30.21	30.10	30.12	30.43
$\text{LKSVD}_{1,8,256}$ [22]	29.49	31.99	30.19	28.76	29.73	28.75	29.09	31.67	28.86	29.66	29.65	29.33	29.76
DCT [1]	28.09	31.18	29.02	27.30	27.71	27.50	28.10	31.05	28.69	28.94	28.72	28.70	28.75
DCT2net	29.29	32.20	30.15	28.45	29.16	28.48	28.96	31.76	29.16	29.71	29.64	29.51	29.71
DCT/DCT2net	29.16	32.26	30.08	28.32	29.08	28.42	28.88	31.75	29.29	29.56	29.41	29.41	29.64
Noise Level	$\sigma = 50$												
BM3D [7]	26.13	29.69	26.68	25.04	25.82	25.10	25.90	29.05	27.22	26.78	26.81	26.46	26.72
PEWA [11]	26.25	29.29	26.69	24.53	25.46	25.07	25.82	28.83	26.58	26.64	26.67	26.02	26.49
DnCNN [15]	27.03	30.00	27.32	25.70	26.78	25.87	26.48	29.39	26.22	27.20	27.24	26.90	27.18
$\text{LKSVD}_{1,8,256}$ [22]	26.26	28.53	26.52	25.12	26.00	25.31	25.93	28.32	24.75	26.55	26.68	26.07	26.34
DCT [1]	24.67	27.73	25.48	23.93	24.10	24.05	24.78	27.71	24.98	25.81	26.01	25.55	25.40
DCT2net	26.20	28.78	26.59	24.86	25.54	25.15	25.91	28.55	25.53	26.62	26.70	26.27	26.39
DCT/DCT2net	26.20	29.05	26.48	24.74	25.41	25.15	25.89	28.63	25.73	26.47	26.56	26.20	26.38

TABLE IV
THE AVERAGE PSNR (DB) RESULTS OF DIFFERENT METHODS ON BSD68 CORRUPTED WITH WHITE GAUSSIAN NOISE AND $\sigma = 15$, 25 AND 50. THE BEST TWO RESULTS ARE HIGHLIGHTED IN RED AND BLUE COLORS, RESPECTIVELY.

Methods	BM3D	PEWA	DnCNN	$\text{LKSVD}_{1,8,256}$	DCT	DCT2net	DCT/DCT2net
$\sigma = 15$	31.07	31.04	31.72	31.33	30.32	31.09	30.97
$\sigma = 25$	28.57	28.52	29.23	28.76	27.76	28.64	28.53
$\sigma = 50$	25.62	25.53	26.23	25.68	24.86	25.68	25.59

TABLE V
RUNNING TIME (IN SECONDS) OF DIFFERENT METHODS FOR DENOISING IMAGES WITH SIZE 256×256, 512×512 AND 1,024×1,024. RUN TIMES ARE GIVEN ON CPU (LEFT) AND GPU (RIGHT) WHEN POSSIBLE.

Image size	BM3D[7]	PEWA[11]	DnCNN[15]	$\text{LKSVD}_{1,8,256}$ [22]	DCT 16×16[1]	DCT2net	DCT/DCT2net
256×256	1.73	38.85	0.87 / 0.010	1.15 / 0.020	0.49 / 0.005	0.39 / 0.005	1.05
512×512	6.65	190.82	3.47 / 0.037	5.78 / 0.082	2.02 / 0.037	1.56 / 0.027	4.08
1,024×1,024	26.90	803.76	18.35 / 0.145	25.78 / 0.332	8.70 / 0.161	5.88 / 0.112	16.87

TABLE VI
MODEL COMPLEXITIES COMPARISON OF OUR PROPOSED METHOD WITH TWO POPULAR NETWORKS

Methods	DnCNN	$\text{LKSVD}_{1,8,256}$	DCT2net
Number of layers	17	5	2
Number of parameters	556,096	35,138	28,561

established methods such as BM3D as shown in Figure 12. The use of the traditional DCT on flat areas produces, for example, a better-looking sky than Deep K-SVD or BM3D when applied to the *Castle* image. Unsurprisingly, DCT/DCT2net (based on two layers) cannot compete with very deep neural networks such that DnCNN [15] but it is faster (see below and Table V).

Nevertheless, the performance has to be put in perspective with the complexity of the model which is studied in the next subsection.

C. Complexity and low-cost training

We want to emphasize that DCT2net is very light and fast compared to its traditional and deep-learning-based counterparts. In Table VI, we reported the complexity in terms of layers numbers and parameters. The number of parameters of DCT2net represents only 5% of the total of parameters of DnCNN. Moreover, the underlying parameters are the same whatever the noise level is, which is not the case for DnCNN or Deep K-SVD where the models have to be trained from scratch for every noise level[3]

In addition to the number of parameters, the executing time is a crucial feature of denoising algorithms. Table V is provided for information purposes only, as the implementation, the language used and the machine on which the code is run,

[3]Although solutions for handling multiple noise levels within the same network were proposed, including a noisemap at the network entry by the authors of [16].

highly influence the execution time. The CPU used is a 2,3 GHz Intel Core i7 and the GPU is a GeForce RTX 2080 Ti. We used the IPOL implementation [29] for BM3D [7] and the implementation provided by the authors for the others algorithms, except for DCT [1] that we re-implemented with Pytorch on our own, leveraging the network unfolding scheme already used in [22]. By the way, DCT2net can be easily adapted to this specific unfolding implementation, as there is no difference between DCT2net and DCT denoiser, apart from the underlying bases.

We also tried to train our network on fewer images than the original BSD400 dataset. As our network relies on a two-layers architecture, it is less prone to overfitting and the learning can be performed only from several dozens of images. With 10 and 40 images, corresponding to 10×665 and 40×665 overlapping patches of size 128×128 respectively, our DCT/DCT2net achieves almost the same performance with no visual difference. By way of comparison, training a model with 40 images takes less than one hour with a GeForce RTX 2080 Ti and the average PSNR values on Set12 are 32.07dB, 29.69dB and 26.39dB, for $\sigma=15$, 25 and 50 respectively.

VI. Discussion and conclusion

DCT2net is one of the first attempts to create an interpretable shallow CNN for image denoising. Sure enough, the performance of DCT2net still falls short when compared to state-of-the-art deep-learning-based methods such as DnCNN [15]. However, manipulating shallow networks has much to offer. Beyond the fact that they are extremely fast as the number of hidden layers is limited, these networks challenge us to think differently our approach to neural networks and encourage us to be more creative than the traditional "Transform-BatchNorm-ReLU" repeated dozens of times. With shallow networks, the activation function must be carefully designed to best match its purpose. Thus, during the training phase, DCT2net uses an approximation of a hard shrinkage function as activation function that depends on the noise level. This is, to the best of our knowledge, the first time such a function is used in a CNN.

Moreover, DCT2net is fully interpretable, unlike Deep K-SVD [22] that uses a multilayer perceptron (MLP) ahead of its sparsity-based network. This interpretability is an important advantage, making the method more robust. At the end of the optimization process, it is possible to check what the network has just learned and, in the case of DCT2net, to directly display the learned basis. By easily exploring the different steps of the process, some usages, usually taken for granted, are disproved by the machine. Thus, we were surprised to realize that the aggregation step that is common in denoising methods based on patches, is not a basic post-processing step but can be fully integrated in the denoising process to considerably improve the performance.

This study shows that signal processing methods such as the popular DCT denoising algorithm can have a comeback by improving the transform involved through deep learning framework. We showed that fully interpretable CNNs can be designed, for which denoising performances compare favorably with state-of-the-art traditional algorithms. We hope that our work will open the door to new architectures, more reliable and understandable for the human brain.

Appendix A
Why is taking multiple thresholds useless ?

In the definition of DCT2net (and traditional DCT), a unique threshold λ, dependent on the level of noise σ, is applied to all the coefficients of the vector $\boldsymbol{P}^{-1}\boldsymbol{y}$, corresponding to the frequency representation of the signal $\boldsymbol{y}$. One may wonder what would bring a different threshold for every coefficient, replacing the function φ_λ by $\varphi_{\lambda_1,\dots,\lambda_n}$ defined by:

$$\forall \boldsymbol{x}\in\mathbb{R}^n,\ \varphi_{\lambda_1,\dots,\lambda_n}(\boldsymbol{x})=(\varphi_{\lambda_1}(x_1),\dots,\varphi_{\lambda_n}(x_n))$$

As a matter of fact, defining multiple thresholds is useless as the matrix $\boldsymbol{P}$ and the threshold values $\lambda_1,\dots,\lambda_n$ can be "encoded" in a single matrix as explained by the following result.

Proposition 1. *Let $\lambda_1,\dots,\lambda_n>0$ be n values of threshold and $\boldsymbol{\Lambda}=diag(\lambda_1,\dots,\lambda_n)$.*
$\forall \boldsymbol{P}\in\mathcal{GL}_n(\mathbb{R}),\forall \boldsymbol{y}\in\mathbb{R}^n,\forall\sigma>0,$

$$\boldsymbol{P}\varphi_{\lambda_1\sigma,\dots,\lambda_n\sigma}(\boldsymbol{P}^{-1}\boldsymbol{y})=(\boldsymbol{P}\boldsymbol{\Lambda})\varphi_\sigma((\boldsymbol{P}\boldsymbol{\Lambda})^{-1}\boldsymbol{y})$$

Proof. The result can be easily derived thanks to the property on hard shrinkage functions, stating that for two levels of threshold λ and λ', we have $\varphi_\lambda(x)=\frac{\lambda}{\lambda'}\varphi_{\lambda'}(\frac{\lambda'}{\lambda}x)$. □

Appendix B
Direct technique to derive an orthonormal matrix for DCT2net

In addition to the technique relying on the introduction of a regularization term, we expose here a direct technique that is based on the following lemma.

Lemma 1. *Let $\mathcal{O}_n(\mathbb{R})$ be the set of orthonormal matrices, $\mathcal{GL}_n(\mathbb{R})$ the set of invertible matrices and $\mathcal{S}_n^{++}$ the set of symmetric positive definite matrices of size $n\times n$. Then,*

$$\mathcal{O}_n(\mathbb{R})=\left\{\boldsymbol{M}\left(\sqrt{\boldsymbol{M}^T\boldsymbol{M}}\right)^{-1}\mid \boldsymbol{M}\in\mathcal{GL}_n(\mathbb{R})\right\}.$$

where $\sqrt{\boldsymbol{A}}$ designates the only matrix of $\mathcal{S}_n^{++}$ such that $\boldsymbol{A}=\sqrt{\boldsymbol{A}}\times\sqrt{\boldsymbol{A}}$ (exists and is unique if $\boldsymbol{A}\in\mathcal{S}_n^{++}$).

Proof. First of all, $\forall \boldsymbol{M}\in\mathcal{GL}_n(\mathbb{R})$, $\boldsymbol{M}^T\boldsymbol{M}\in\mathcal{S}_n^{++}$. Moreover, $\forall \boldsymbol{A}\in\mathcal{S}_n^{++}$, $\boldsymbol{A}$ is invertible (with $\boldsymbol{A}^{-1}\in\mathcal{S}_n^{++}$). Therefore, for all $\boldsymbol{M}\in\mathcal{GL}_n(\mathbb{R})$, $\boldsymbol{M}(\sqrt{\boldsymbol{M}^T\boldsymbol{M}})^{-1}$ is well defined.

Now, by double inclusion:
($\subset$): Let $\boldsymbol{Q}\in\mathcal{O}_n(\mathbb{R})$. We set $\boldsymbol{M}=\boldsymbol{Q}\in\mathcal{GL}_n(\mathbb{R})$.
$\sqrt{\boldsymbol{M}^T\boldsymbol{M}}=\boldsymbol{I}$ is invertible and $\boldsymbol{Q}=\boldsymbol{M}(\sqrt{\boldsymbol{M}^T\boldsymbol{M}})^{-1}$.
($\supset$): Let $\boldsymbol{M}\in\mathcal{GL}_n(\mathbb{R})$ and $\boldsymbol{Q}=\boldsymbol{M}(\sqrt{\boldsymbol{M}^T\boldsymbol{M}})^{-1}$. Using that for all $\boldsymbol{A}\in\mathcal{S}_n^{++}$, $(\sqrt{\boldsymbol{A}})^{-1}=\sqrt{\boldsymbol{A}^{-1}}$, we have:

$$\begin{aligned}\boldsymbol{Q}\boldsymbol{Q}^T &= \boldsymbol{M}(\sqrt{\boldsymbol{M}^T\boldsymbol{M}})^{-1}(\sqrt{\boldsymbol{M}^T\boldsymbol{M}})^{-1}\boldsymbol{M}^T\\ &= \boldsymbol{M}\sqrt{(\boldsymbol{M}^T\boldsymbol{M})^{-1}}\sqrt{(\boldsymbol{M}^T\boldsymbol{M})^{-1}}\boldsymbol{M}^T\\ &= \boldsymbol{M}(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\\ &= \boldsymbol{M}\boldsymbol{M}^{-1}(\boldsymbol{M}^T)^{-1}\boldsymbol{M}^T\\ &= \boldsymbol{I}\end{aligned}$$

hence, $\boldsymbol{Q} \in \mathcal{O}_n(\mathbb{R})$. □

Let $F_{\boldsymbol{P}}$ denote the network DCT2net where $\boldsymbol{P}$ is the learned transform. The direct technique consists in solving the following optimization problem:

$$\boldsymbol{M}^* = \arg \min_{\boldsymbol{M} \in \mathcal{GL}_{p^2}(\mathbb{R})} \sum_{i=1}^{N} \| F_{\boldsymbol{M}\left(\sqrt{\boldsymbol{M}^T\boldsymbol{M}}\right)^{-1}}(\mathbf{y}_i, \sigma_i) - \mathbf{x}_i \|_2^2 \tag{9}$$

Similarly to the unconstrained formulation of DCT2net (4), the optimization problem is solved by stochastic gradient descent, leveraging the power of automatic differentiation in modern machine learning libraries such as Pytorch. The learned transform $\boldsymbol{P}^*$ is reconstructed at the end and is guaranteed to be orthonormal thanks to Lemma 1:

$$\boldsymbol{P}^* = \boldsymbol{M}^* \left(\sqrt{\boldsymbol{M}^{*T}\boldsymbol{M}^*}\right)^{-1}$$

APPENDIX C
LINK BETWEEN ORTHONORMAL MATRICES AND ORTHOGONAL ONES IN DCT2NET

Although often used as synonyms in the literature, a clear distinction between orthonormal matrices and orthogonal ones is made in this paper.

Definition 1. *Let $\boldsymbol{P}$ be a matrix of size $n \times n$.*

- *$\boldsymbol{P}$ is an orthonormal matrix, and we note $\boldsymbol{P} \in \mathcal{O}_n(\mathbb{R})$, if $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}_n$.*
- *$\boldsymbol{P}$ is an orthogonal matrix, and we note $\boldsymbol{P} \in \mathcal{O}_n^g(\mathbb{R})$, if $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{D}$, with $\boldsymbol{D}$ an invertible diagonal matrix.*

In other words, a matrix $\boldsymbol{P}$ is said to be *orthonormal* if its columns $c_1, \ldots, c_n$ have the property: $\forall i, j \in \{1, \ldots, n\}$, $\langle c_i, c_j \rangle = \delta_{i,j}$ where $\delta_{i,j}$ is the Kronecker delta. The *orthogonality* property is less restrictive as its columns must satisfy $\forall i, j \in \{1, \ldots, n\}$, $\langle c_i, c_j \rangle = 0 \Leftrightarrow i \neq j$.

Taking $\boldsymbol{P} \in \mathcal{O}_n(\mathbb{R})$ with multiple values of threshold amounts to considering only one value of threshold with $\boldsymbol{P} \in \mathcal{O}_n^g(\mathbb{R})$ and conversely. Indeed, let $\boldsymbol{P} \in \mathcal{O}_n^g(\mathbb{R})$. There exists $\boldsymbol{D}$ an invertible diagonal matrix such that $\boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{D}$. We can write $\boldsymbol{P} = \boldsymbol{Q}\sqrt{\boldsymbol{D}}$ with $\boldsymbol{Q} = \boldsymbol{P}(\sqrt{\boldsymbol{D}})^{-1} \in \mathcal{O}_n(\mathbb{R})$. Now applying Prop. 1 for $\lambda_i = \sqrt{\boldsymbol{D}}_{i,i} > 0$ and $\boldsymbol{Q}$ gives that $\forall \boldsymbol{y} \in \mathbb{R}^n, \forall \sigma > 0$,

$$\boldsymbol{P}\varphi_\sigma(\boldsymbol{P}^{-1}\boldsymbol{y}) = \boldsymbol{Q}\varphi_{\lambda_1\sigma,\ldots,\lambda_n\sigma}(\boldsymbol{Q}^{-1}\boldsymbol{y})$$

ACKNOWLEDGMENT

This work was supported by Bpifrance agency (funding) through the LiChIE contract. Computations were performed on the Inria Rennes computing grid facilities partly funded by France-BioImaging infrastructure (French National Research Agency - ANR-10-INBS-04-07, "Investments for the future").

We would like to thank R. Fraisse (Airbus) for fruitful discussions.

REFERENCES

[1] G. Yu and G. Sapiro, "DCT image denoising: a simple and effective image denoising algorithm," in *Image Processing On Line*, vol. 1, pp. 292–296, 2011.

[2] H. Chipman, E. Kolaczyk, and R. McCulloch, "Adaptive Bayesian Wavelet Shrinkage," in *Journal of the American Statistical Association*, vol. 92, no. 440, pp. 1413-1421, 1997.

[3] W. B. Pennebaker and J. L. Mitchell, "JPEG: still image data compression standard," Van Nostrand Reinhold, New York, 1992.

[4] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," in *IEEE Transactions on Image Processing*, vol. 15, no. 12, pp. 3736-3745, 2006.

[5] A. Buades, B. Coll, and J.-M. Morel, "A review of image denoising algorithms, with a new one," in *SIAM Journal on Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490-530, 2005.

[6] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, CO, USA, 2011, pp. 977-984.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3D transform-domain collaborative filtering," in *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, 2007.

[8] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Non-local sparse models for image restoration," in *IEEE 12th International Conference on Computer Vision*, Tokyo, Japan, 2009, pp. 2272-2279.

[9] A. Buades, M. Lebrun, and J.-M. Morel. "A Non-local Bayesian image denoising algorithm," in *SIAM Journal on Imaging Science*, vol. 6, no. 3, pp. 1665-1688, 2013.

[10] S. Gu, L. Zhang, W. Zuo, and X. Feng, "Weighted Nuclear Norm Minimization with Application to Image Denoising," in *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014, pp. 2862-2869.

[11] C. Kervrann, "PEWA : Patch-based exponentially weighted aggregation for image denoising," in *Neural Information Processing Systems*, Montreal, Canada, 2014, pp. 2150–2158.

[12] Q. Jin, I. Grama, C. Kervrann, and Q. Liu, "Non-local means and optimal weights for noise removal," in *SIAM Journal on Imaging Sciences*, vol. 10, no. 4, pp. 1878-1920, 2017.

[13] C. Kervrann and J. Boulanger, "Optimal spatial adaptation for patch-based image denoising," in *IEEE Transactions on Image Processing*, vol. 15, no. 10, pp. 2866-2878, 2006.

[14] C. Kervrann and J. Boulanger, "Local adaptivity to variable smoothness for exemplar-based image denoising and representation," in *International Journal of Computer Vision*, vol. 79, no. 1, pp. 45-69, 2008.

[15] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: residual learning of deep CNN for image denoising," in *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, 2017.

[16] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising," in *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4608–4622, 2018.

[17] X. Mao, C. Shen, and Y. Yang. "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in *Advances in Neural Information Processing Systems*, pp. 2802–2810, 2016.

[18] Y. Chen and T. Pock, "Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1256–1272, 2017.

[19] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang, "Non-local recurrent network for image restoration," in *Advances in Neural Information Processing Systems*, pp. 1673–1682, 2018.

[20] P. Tobias and R. Stefan, "Neural Nearest Neighbors Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[21] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with BM3D?," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2392–2399, 2012.

[22] M. Scetbon, M. Elad, and P. Milanfar, "Deep K-SVD Denoising," in *IEEE Transactions on Image Processing*, vol. 30, pp. 5944 - 5955, 2021.

[23] D. Yang and J. Sun, "BM3D-Net: A Convolutional Neural Network for Transform-Domain Collaborative Filtering," in *IEEE Signal Processing Letters*, vol. 25, no. 1, pp. 55-59, Jan. 2018.

[24] J. Canny, "A computational approach to edge detection," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[25] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss Functions for Image Restoration With Neural Networks," in *IEEE Transactions on Computational Imaging*, vol. 3, no. 1, pp. 47-57, 2017.

[26] N. Pierazzo, J.-M. Morel, and G. Facciolo, "Multi-Scale DCT Denoising," in *Image Processing On Line*, vol. 7, pp. 288–308, 2017.
[27] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *International Conference on Computer Vision*, vol. 2, pp. 416–423, 2001.
[28] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations*, 2015.
[29] M. Lebrun, "An Analysis and Implementation of the BM3D Image Denoising Method," in *Image Processing On Line*, vol. 2, pp. 175–213, 2012.