Wavelets in the deep learning era

Zaccharie Ramzi, Kevin Michalewicz, Jean-Luc Starck, Thomas Moreau, Philippe Ciuciu

Noname manuscript No.
(will be inserted by the editor)

Wavelets in the Deep Learning Era

Zaccharie Ramzi · Kevin Michalewicz · Jean-Luc Starck · Thomas Moreau · Philippe Ciuciu

Received: date / Accepted: date

Abstract Sparsity based methods, such as wavelets, have been state-of-the-art for more than 20 years for inverse problems before being overtaken by neural networks. In particular, U-nets have proven to be extremely effective. Their main ingredients are a highly non-linear processing, a massive learning made possible by the flourishing of optimization algorithms with the power of computers (GPU) and the use of large available datasets for training. It is far from obvious to say which of these three ingredients has the biggest impact on the performance. While the many stages of non-linearity are intrinsic to deep learning, the usage of learning with training data could also be exploited by sparsity based approaches. The aim of our study is to push the limits of sparsity to use, similarly to U-nets, massive learning and large datasets, and then to compare the results with U-nets. We present a new network architecture, called learnlets, which conserves the properties of sparsity based methods such as exact reconstruction and good generalization properties, while fostering the power of neural networks for learning and fast calculation.

We evaluate the model on image denoising tasks. Our conclusion is that U-nets perform better than learnlets, while learnlets have better generalization properties.

Keywords Machine Learning · Deep Learning · Neural Networks · Wavelets · Denoising · Image restoration

Z. Ramzi
AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot, Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France
Inria Saclay Ile-de-France, Parietal team, Univ. Paris-Saclay, France
CEA/NeuroSpin, Bat 145, F-91191 Gif-sur-Yvette, France
E-mail: zaccharie.ramzi@inria.fr

K. Michalewicz
AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot, Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France

J-L. Starck
AIM, CEA, CNRS, Université Paris-Saclay, Université Paris Diderot, Sorbonne Paris Cité, F-91191 Gif-sur-Yvette, France

T. Moreau
Inria Saclay Ile-de-France, Parietal team, Univ. Paris-Saclay, France

P. Ciuciu
Inria Saclay Ile-de-France, Parietal team, Univ. Paris-Saclay, France
CEA/NeuroSpin, Bat 145, F-91191 Gif-sur-Yvette, France

1 Introduction

The U-net was introduced by [1] to perform biomedical image segmentation. It has since then been used in a wide variety of image-to-image problems, not just segmentation, either as a strong baseline or as the building block for a more complex model. In particular, the U-nets have had success in image-to-image translation [2], image reconstruction (in CT [3] or MRI [4, 5, 6]) and denoising [6].

However, like many other deep learning approaches, the reason for its success is not well understood. The ideas of the U-net come from [7] in part. In this work, the base choices that make a U-net are grounded with intuitive explanations. To be able to distinguish between critical and legacy parts in the U-net design, it is important to understand its mechanisms.

On the other hand, wavelets-based approaches are not state-of-the-art anymore for denoising but are theoretically grounded (see for example [8]). For applications where guarantees are needed – such as medical applications – this makes them ideal candidates.

Similarly to wavelets, U-nets present a multi-scale approach, which allows to analyze the signal at different resolutions. Their main difference is the non-linearity applications. Indeed, while wavelets apply only one non-linearity when applied to denoising – a method called wavelet shrinkage –, the U-net architecture relies on several ReLUs and max-poolings. These chained non-linearities make the analysis of the denoising in U-nets very complicated. In particular, it is difficult to see how a network trained on one type of noise can be applied to other types of noises. Some works [9] even show that classical neural networks can fail to recover elements that classical methods do, suggesting a trade-off between quality and stability.

In this paper, we investigate whether using massive learning and large available datasets in a sparsity framework could allow us to achieve U-nets performance or if the chained non-linearities are equally important. We propose a new network, called *Learnlets*, which makes use of one of the strongest advantages of neural networks, learning via gradient descent to enhance the expressive power of wavelets, while keeping some interesting wavelet properties such as exact reconstruction. We choose to test this network on a denoising problem, a task where wavelets have historically well-performed but are now overtaken by deep learning approaches. In parallel, we also propose a new U-net denoising scheme that guarantees an exact reconstruction when the noise tends to zero.

The full implementation of our method is open source in Python[1]. Furthermore, a previous short version of this work was presented in [10].

We review the main studies on wavelets and neural networks in Section 2, while the learnlets model is presented in Section 3. The implementation of the exact reconstruction property for learnlets and for a generic neural network is analyzed in Section 4. Finally, details regarding the datasets and experiments are given in Section 5, results are shown in Section 6, and conclusions in Section 7.

2 Related Work

Different studies have attempted to work at the intersection of wavelets and neural networks. In [11], the authors cast the wavelet transform as an auto-encoder where the latent representation has to be sparse and learn the filters. In this architecture only a simple high-pass and low-pass filter pair is learned. Similarly, but pushing further the idea, [12] developed a learning strategy to design new wavelet filters with certain properties imposed such as what they call perfect reconstruction (which we termed exact reconstruction) or vanishing moments. Their work was inspired by [13] where the authors chose to use data patches to learn their transform rather than noise like in [12].

Observing U-nets, two parts are very similar to synthesis and analysis concepts in wavelet decompositions, [6] proposed to use the wavelet transform to perform a better pooling/unpooling strategy than simply max-pooling/bilinear upsampling. [14] inspired themselves from the cascading wavelet shrinkage systems to enhance denoising autoencoders. In brief, they proved that using a soft-thresholding non-linearity provided more power to the denoising autoencoders than other non-linearities.

In these related papers, non-linearities (namely ReLU) are in majority applied to the low frequencies rather than the high frequencies, contrarily to what is common in the wavelet framework. In this work, we don't try to modify U-nets by importing wavelet ingredients, but rather try to push the limits of sparsity based approach by using learning while keeping sparsity concept unchanged. This allows us to recover the classical properties of wavelets i.e. decomposition with exact reconstruction, thresholding and reconstruction, while using a learning based approach.

3 Learnlets, the model

Let $\mathbf{x} \in \mathbb{R}^{n \times n}$ be an image. Let $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$ be the version of this image corrupted by an additive white Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_{\mathbf{n}\times\mathbf{n}})$ whose variance σ^2 is assumed known. Let Σ be a compact set of possible values for σ, we chose to have $\sigma \sim \mathcal{U}(\Sigma)$. For a given number of scales m and a given set of parameters $\theta = (\boldsymbol{\theta_S}, \boldsymbol{\theta_T}, \boldsymbol{\theta_A}) \in \Theta_m$, we defined the learnlets as function $\mathbf{f}_\theta$ from $(\mathbb{R}^{n\times n} \times \Sigma)$ to $\mathbb{R}^{n\times n}$:

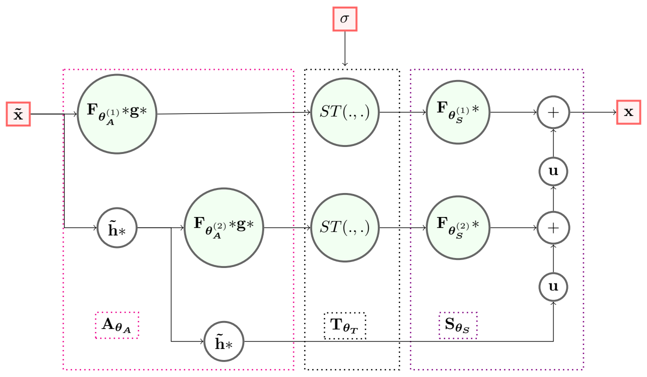
$$\mathbf{f}_\theta(\tilde{\mathbf{x}}, \sigma) = \mathbf{S}_{\boldsymbol{\theta_S}}\left(\mathbf{T}_{\boldsymbol{\theta_T}}\left(\mathbf{A}_{\boldsymbol{\theta_A}}\left(\tilde{\mathbf{x}}\right), \sigma\right)\right) \tag{1}$$

where we have:

1. $\mathbf{A}_{\boldsymbol{\theta_A}}$, the analysis function defined in 3.1.
2. $\mathbf{T}_{\boldsymbol{\theta_T}}$, the thresholding function defined in 3.2.
3. $\mathbf{S}_{\boldsymbol{\theta_S}}$, the synthesis function defined in 3.3.

An illustration of the learnlets is given in Figure 1.

[1] https://github.com/zaccharieramzi/understanding-unets



σ
$\tilde{\mathbf{x}}$
$\mathbf{F}_{\boldsymbol{\theta}_A^{(1)}} * \mathbf{g} *$
$\tilde{\mathbf{h}} *$
$\mathbf{F}_{\boldsymbol{\theta}_A^{(2)}} * \mathbf{g} *$
$\tilde{\mathbf{h}} *$
$ST(.,.)$
$ST(.,.)$
$\mathbf{F}_{\boldsymbol{\theta}_S^{(1)}} *$
$\mathbf{F}_{\boldsymbol{\theta}_S^{(2)}} *$
$+$
$\mathbf{u}$
$+$
$\mathbf{u}$
$\mathbf{x}$
$\mathbf{A}_{\boldsymbol{\theta}_A}$
$\mathbf{T}_{\boldsymbol{\theta}_T}$
$\mathbf{S}_{\boldsymbol{\theta}_S}$

Fig. 1 Schematic representation of the learnlets model, with $m = 2$ scales. The red nodes are inputs/outputs. The lightly green nodes correspond to functions whose parameters can be learned. Note that the standard deviation of the noise before thresholding is not learned but rather estimated, and is omitted in this diagram for clarity.

3.1 Analysis

Intuitively, one can see the analysis function as the equivalent of the wavelet transform with some learned filters. This linear function is defined as:

$$\mathbf{A}_{\boldsymbol{\theta}_A}(\tilde{\mathbf{x}}) = \left(\left(\mathbf{F}_{\boldsymbol{\theta}_A^{(i)}} * \mathbf{g}\left(\tilde{\mathbf{h}}^{i-1}(\tilde{\mathbf{x}})\right)\right)_{i=1}^{m}, \tilde{\mathbf{h}}^{m}(\tilde{\mathbf{x}})\right) \tag{2}$$

where we have:

- $\mathbf{F}_{\boldsymbol{\theta}_A^{(i)}}$, the filter bank at scale i. The convolutions are done without bias. $\boldsymbol{\theta}_{\boldsymbol{A}}^{(i)}$ are the J_i convolution kernels all of the same square size (k_A, k_A) (for now $J_i = J_m$).
- $\tilde{\mathbf{h}} = \bar{\mathbf{u}} \circ \mathbf{h}$, the low-pass filtering ($\mathbf{h}$) followed by a decimation ($\bar{\mathbf{u}}$). The decimation is performed by taking one line out of 2 and one row out of 2, in line with the way it is done in wavelet transforms.
- $\mathbf{g}$ the high-pass filtering defined as: $\mathbf{g}(\mathbf{y}) = \mathbf{y} - \mathbf{u}(\tilde{\mathbf{h}}(\mathbf{y}))$, with $\mathbf{u}$ the upsampling operation performed with a bicubic interpolator.

For ease of manipulation we rewrite $\mathbf{A}_{\boldsymbol{\theta}_A}(\tilde{\mathbf{x}}) = ((\mathbf{d}_i)_{i=1}^{m}, \mathbf{c})$, with $\mathbf{d}_i \in \mathbb{R}^{\frac{n}{2^{i-1}} \times \frac{n}{2^{i-1}} \times J_i}$ the detail coefficients and $\mathbf{c}$ the coarse coefficients.

Note that low and high pass filters $(\mathbf{h}, \mathbf{g})$ are fixed, and only $\mathbf{F}_{\boldsymbol{\theta}_A}^{(i)}$ filters are learned. As $\mathbf{g}$ has a zero mean, all coefficients $\mathbf{d}_i$ have by construction a zero mean. This wavelet property is fundamental to model the noise on the coefficients. Indeed, in the absence of signal, the coefficients follow a Gaussian distribution with a zero mean, and a standard $k\sigma$ thresholding can be applied, σ being the noise standard deviation. With wavelets, k would be chosen between 3 and 5, and would be a user parameter. In this setting, this k value can be learned, and can be different at each scale.

3.2 Thresholding

The non-linearity function used for wavelet shrinkage is typically either a hard-thresholding or a soft-thresholding [8]. The soft-thresholding offers more stability and therefore we made this choice for our architecture. The thresholding function, in the case of a white Gaussian noise of variance σ^2, is defined as:

$$\mathbf{T}_{\boldsymbol{\theta}_T}\left(((\mathbf{d}_i)_{i=1}^{m}, \boldsymbol{c}), \sigma\right) = \left(\left((t_{ij}(d_{ij}, \sigma))_{i=1}^{J_i}\right)_{i=1}^{m}, \boldsymbol{c}\right) \tag{3}$$

where $t_{ij}(\boldsymbol{d}, \sigma) = \hat{\sigma}_{ij} ST\left(\frac{1}{\hat{\sigma}_{ij}} d_{ij}, \theta_T^{(ij)} \sigma\right)$, with:

- $d_{ij} \in \mathbb{R}^{\frac{n}{2^{i-1}} \times \frac{n}{2^{i-1}}}$ the output of the j-th filter of i-th scale.
- $\hat{\sigma}_{ij}$ the estimated standard deviation of d_{ij} when the input of the transform is set to be a white Gaussian noise of variance 1. This ensures the noise coming just before the thresholding is of variance approximately σ. The threshold is therefore truly $\theta_T^{(ij)} \sigma$.
- $\theta_T^{(ij)}$ is the thresholding level applied at scale i on the j-th analysis filter.
- $ST(\boldsymbol{d}, \boldsymbol{s})$ is the soft-thresholding function applied point-wise on $\boldsymbol{d}$ with threshold $\boldsymbol{s}$: $ST(\boldsymbol{d}, \boldsymbol{s}) = \operatorname{sign}(\boldsymbol{d}) \max(|\boldsymbol{d}| - \boldsymbol{s}, 0)$.

It is important to notice that, thanks to linearity of the analysis operator, the thresholding strategy can be very easily adapted to non-stationary Gaussian noise or to any other kind of noise, such as Poisson noise or a mixture of Gaussian and Poisson noise.

3.3 Synthesis

Intuitively, one can see the synthesis function as the equivalent of the wavelet reconstruction operator, with learned filters. It is important to note that the synthesis function is linear. The synthesis function is defined recurrently as:

$$\mathbf{S}_{\boldsymbol{\theta}_S}\left((\mathbf{d}_i)_{i=1}^{m}, \boldsymbol{c}\right) = \mathbf{S}_{\boldsymbol{\theta}_S}^{(m-1)}\left((\mathbf{d}_i)_{i=1}^{m-1}, \boldsymbol{u}(\boldsymbol{c}) + \mathbf{F}_{\boldsymbol{\theta}_S^{(m)}} * \mathbf{d}_m\right) \tag{4}$$

where $\mathbf{S}_{\emptyset}(\emptyset, \boldsymbol{c}) = \boldsymbol{c}$ and:

- $\mathbf{F}_{\boldsymbol{\theta}_S^{(i)}}$, the filter bank at scale i, used for regrouping. The convolutions are done without bias and added all together. $\boldsymbol{\theta}_{\boldsymbol{S}}^{(i)}$ are the J_i convolution kernels all of the same square size (k_S, k_S).
- $\boldsymbol{u}$, the upsampling operation performed with a bicubic interpolator.

3.4 Constraints

Some constraints are used on the parameters of the learnlets to make them as close as possible to the wavelets and therefore make them understandable:

- The analysis filters are forced to have a unit norm.
- The thresholding levels are in $[0, 5]$.

3.5 Learning

The optimization problem is given as:

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \mathbb{E}_{\mathbf{x},\sigma} \left[L_f(\theta) \right] \tag{5}$$

where $L_f(\theta) = \|\mathbf{x} - \mathbf{f}_\theta(\tilde{\mathbf{x}}, \sigma)\|_2^2$ and the expected value is computed empirically, via the empirical mean over a batch.

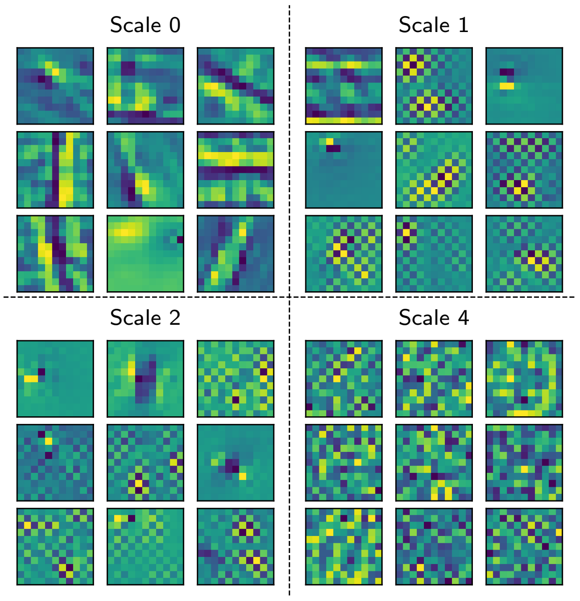


Fig. 2 Visualization of learnlet analysis filters for four different scales.

3.6 Learnlets as the bridge between Sparsity and U-nets

The learnlet transform is very similar in its spirit to the curvelet transform. Indeed, in both transforms the image is first decomposed into a set of wavelet scales, and filters are applied on each scale. In a curvelet decomposition, it would be directional and fixed filters, while filters are learned in our proposed scheme. Obtained coefficients can be manipulated exactly the same way as wavelets or curvelets coefficients.

It is interesting to notice that after training, the pixels of the learnlets' (with exact reconstruction) analysis filters constitute meaningful designs. This can be observed in Figure 2: lines with different slopes are displayed in scale-zero filters (details about the data and the training are given in the section 5).

On the other hand, learnlets share very similar properties with U-nets. For example they make use of gradient-based learning, but they also feature a multi-scale analysis along with the use of non-linearities.

4 Exact reconstruction

4.1 Learnlets

Exact reconstruction guarantees that if no noise is present, the signal will be perfectly reconstructed, without any error. This can be achieved using the analysis filter previously fixed as identity. In particular, let's consider a single scale i, after the application of the $\mathbf{g}$ filter. The operation carried out by the network, without thresholding can be written as:

$$\mathbf{x_{out}}^{(i)} = \sum_{j=1}^{N} \mathbf{F}_{\boldsymbol{\theta}_S^{(i,j)}} * \mathbf{F}_{\boldsymbol{\theta}_A^{(i,j)}} * \mathbf{x_{in}} \tag{6}$$

where N is the number of filters at that scale. Since we have $\mathbf{F}_{\boldsymbol{\theta}_A^{(i,1)}} = \mathbf{Id}$, we can also fix the corresponding synthesis filter $\mathbf{F}_{\boldsymbol{\theta}_S^{(i,1)}} = \mathbf{Id} - \sum_{j=2}^{N} \mathbf{F}_{\boldsymbol{\theta}_S^{(i,j)}} * \mathbf{F}_{\boldsymbol{\theta}_A^{(i,j)}}$. This trivially gives without thresholding, $\mathbf{x_{out}} = \mathbf{x_{in}}$. We implemented this constraint in the network, allowing to learn a different thresholding level for this filter.

4.2 The general case

In order to better understand the properties of exact reconstruction in the learnlets, we can study whether it is possible to enforce it as well for black-box residual neural networks. A simple solution, given a known noise level σ is to use the following general expression:

$$\mathbf{g}_\theta(\tilde{\mathbf{x}}, \sigma) = \tilde{\mathbf{x}} - \sigma \mathbf{f}_\theta(\tilde{\mathbf{x}}) \tag{7}$$

where $\mathbf{f}_\theta$ is the output of the network without exact reconstruction. It can be noted that when σ tends to zero, then $\mathbf{g}_\theta(\tilde{\mathbf{x}}, \sigma) \to \tilde{\mathbf{x}}$ and we can assure that the output will retrieve the input signal. It should be noted that this formulation might be unstable as it can amplify errors at high noise levels. This aspect will be analyzed in the next section.

5 Data and Experiments

The implementation was done in Python 3.6, using the TensorFlow 2.1 framework [15] for model design. The training was done on the Jean Zay public supercomputer, using for each job a single GPU Nvidia Tesla V100 SXM2 with 32GB of RAM.

5.1 Data

The data used was the BSD500 dataset [16]. This data consists of natural images of sizes 481×321 and 321×481. The train and tests subsets of BSD500 were used as the training dataset. The validation subset of BSD500, containing the BSD68 [17] images was left out. We used BSD68 as the test dataset. This choice is motivated by the fact that many other denoising studies [18], [19] use this dataset for comparison.

5.2 Pre-processing

For training, patches of size 256×256 were randomly extracted on-the-fly. The images were then linearly mapped from $[0, 255]$ to the $[-0.5, 0.5]$ interval and converted from RGB to grayscale using the function provided by TensorFlow[2]. In addition, data augmentation techniques such as random flipping and random θ-degree ($\theta = 90°, 180°, 270°$) rotations were applied. Noise was then added by first drawing uniformly at random in the specified interval Σ a noise level σ, then generating a 256×256 white Gaussian noise patch ϵ with this standard deviation. It is to note that during training, a single batch can feature different noise standard deviations.

At test time, the images were mirror-padded to a 352×512 size (or 512×352), in order to avoid shape mismatches when downsampling and upsampling, and the image quality metric was computed only on the original image shape. The test images were also corrupted by an additive white Gaussian noise for various standard deviations σ: $\{0.0001, 5, 15, 20, 25, 30, 50, 55, 60, 75, 85, 95, 100\}$. This allowed us to test the performance of our method in different noise level settings.

[2] `https://www.tensorflow.org/api_docs/python/tf/image/rgb_to_grayscale`; TensorFlow Documentation for RGB to grayscale

5.3 Model and training

5.3.1 Models design

We compare the learnets with the U-net for the task of denoising. For the U-net, we used the architecture described in [6, Fig.10.(a)] which contains 124 million parameters for the case of a network with 128 base filters.

Unless specified otherwise, the learnlets parameters were chosen as:

- $m = 5$ scales.
- 256 learnable analysis filters + 1 fixed analysis filters being just the identity, $\mathbf{F}_{\boldsymbol{\theta}_A^{(i)}}$, of size 11×11.
- 257 learnable synthesis filters, $\mathbf{F}_{\boldsymbol{\theta}_S^{(i)}}$, of size 13×13.
- the thresholding levels only depend on the scale, $\theta_T^{(ij)} = \theta_T^{(i)}$.

This amounts to 372k trainable parameters, only three hundredths of the size of the U-net.

5.3.2 Training parameters

The networks were both trained on the mean squared error in line with (5). Each epoch consisted of 200 batches of 8 extracted patches, and their respective noise level in the case of the learnlets. The training noise standard deviation range was chosen as $\Sigma = [0; 55]$. The networks were trained with an Adam optimizer [20]. The learning rate was set at 10^{-3}, then decreased by half every 25 epochs, until it reached a minimum of 10^{-5}. The trainings took about 8 hours for 500 epochs each.

5.4 Evaluation

5.4.1 Evaluation metric

For the evaluation of the performance of the different models we used the Peak Signal to Noise Ratio (PSNR) metric. It is defined image-wise as the following (with images taken in the $[-0.5; 0.5]$ range):

$$PSNR(\mathbf{x}, \hat{\mathbf{x}}) = -10 \log_{10} \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{8}$$

For each test noise standard deviation σ, we compute the mean of the PSNR of the denoised images, for all BSD68 images.

5.4.2 Testing

In addition, the networks were compared to wavelets denoising [21], which was implemented by using the code of PySAP [22]. The wavelets family was the Biorthogonal 7.9, 5 scales were used, a hard-thresholding was used with a thresholding level of 3 (except for the first scale where it was 4).

6 Results

6.1 Quantitative results

6.1.1 Comparison with other methods

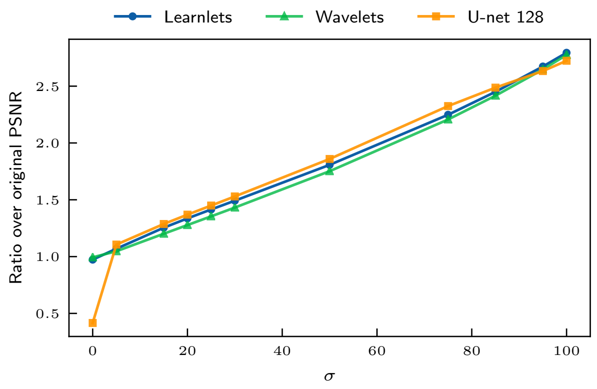


Fig. 3 Ratio of the denoised image PSNR compared to the original noisy image PSNR for different standard deviations of the noise added to the test images for all considered models. The train noise standard deviation range was [0; 55].

Model name	Wavelets	U-net 128	Learnlets	U-net 64
Denoising runtime in ms (std)	274 (21)	272 (18)	106 (12)	64 (1)

Table 1 Runtimes of the different models for the denoising of one image. Parameters used are the same as Figure 3.

We compared the U-net and learnlets with exact reconstruction against algorithms not involving learning, namely wavelets shrinkage. Figure 3 shows that for a large part of the band [5; 55], where they have been trained, the wavelets have a performance that is degraded compared to the learnlets. Using learning, the learnlets enhance their decomposition power compare to the original wavelet model with no learning. For small noise level, the U-net gets degraded performances compared with learnlets with exact reconstruction and wavelets. In this setting, the denoiser must act as the identity. Finally, we can see that for unseen test noise levels (i.e. 95), the performance of U-net drops slightly while the learnlets keep relatively good performances. This suggests that the learnlets generalize better than U-nets on unseen noise levels.

In addition, we can see in Table 6.1.1 that the learnlets benefit from their GPU implementation and run faster than both the wavelets and U-net 128.

6.1.2 Learnlets with exact reconstruction

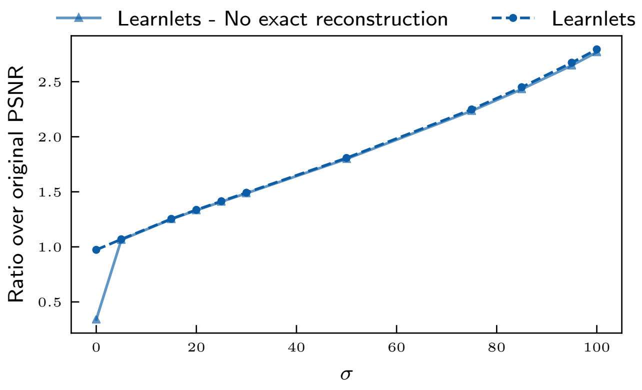


Fig. 4 Ratio of the denoised image PSNR compared to the original noisy image PSNR for different standard deviations of the noise added to the test images for learnlets with and without forcing exact reconstruction. The train noise standard deviation range was [0; 55]. The number of filters used was 64.

We saw in Figure 3 that learnlets with exact reconstruction compete with classical methods for a wide range of noise standard deviations. Figure 4 shows that the performance of the network with forced exact reconstruction is almost the same as the one without forced exact reconstruction (we only lose 0.1dB at $\sigma = 30$ for example) on the majority of the test noise standard deviations. However, for low noise standard deviations, the network with forced exact reconstruction completely overpowers the other one. This is due to the fact that, at low noise standard deviations, for the i-th scale, the term $x_{out}^{(i)}$ is practically the same as its thresholded version, because the thresholds $\theta_T^{(ij)}\sigma$ are going to be low. Therefore, it is compensated in the corresponding synthesis filter used for exact reconstruction at that scale, $\mathbf{F}_{\boldsymbol{\theta}_A^{(i,1)}}$. This allows to guarantee, in this case, no loss of information in the signal if it is clearly present.

6.1.3 U-nets of different sizes

Due to their reduced number of parameters, *overfitting* is *a priori* less likely to occur in small neural networks than in larger ones. Therefore, we studied whether the generalization to high noise levels would be better with small-sized U-nets. To do this, the number of base filters (see Appendix A) was modified with respect to the original case, obtaining Figure 5. We can see that deeper networks perform better for seen and unseen noise levels.

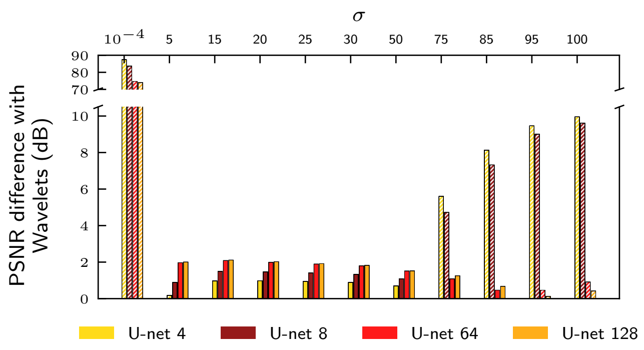


Fig. 5 PSNR difference (in dB) with respect to wavelets denoising for different standard deviations of the noise added to the test images for U-nets of various sizes. The striped bars correspond to negative differences. The train noise standard deviation range was [0; 55].

In terms of generalization, the PSNR difference between U-nets with a low quantity of filters (4 or 8) and those with a larger amount (64 or 128) is amplified for the range [55; 100].

6.1.4 U-net with exact reconstruction

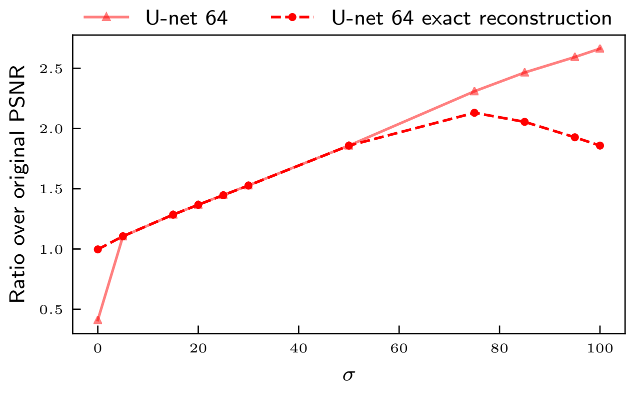


Fig. 6 Ratio of the denoised image PSNR compared to the original noisy image PSNR for different standard deviations of the noise added to the test images for U-net with and without exact reconstruction. The train noise standard deviation range was [0; 55].

It was of interest to know if the exact reconstruction could be implemented in U-nets so as to avoid a large drop in performance for low noise levels. Figure 6 shows that the application of the general case equation (7) yields a PSNR ratio of approximately 1 when $\sigma \to 0$. Apart from that, the PSNR remains similar for higher, but seen, noise standard deviations values (for instance, there is a loss of only 0.03dB at $\sigma = 30$). However, the exact reconstruction is incompatible with generalization at high noise levels in the case of U-nets, as can be observed from the low performance in the interval [55; 100].

6.1.5 Generalization test: Denoising astrophysical images

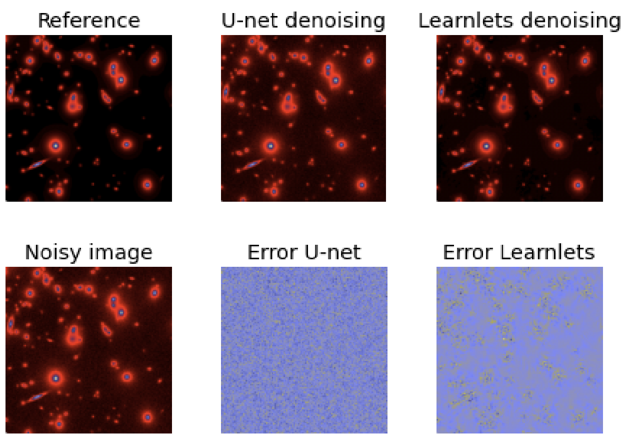


Fig. 7 Denoising results for an astrophysical image contaminated with a noise of $\sigma = 50$. The last two images correspond to the subtraction of the original image to its denoised version.

Another interesting test to evaluate how a network generalizes consists in denoising images that are different from the training dataset. This is for instance similar to what was done in [9], where letters were added in the test image, while no image contains letters in the training data set. Here we applied our trained neural networks on a simulated astronomical image, contaminated with noise of a standard deviation $\sigma = 50$. This experience complements the previous generalization ones where the test images belong to the same class of natural images, but some noise levels where higher than those in the training data.

The astronomical image was firstly normalized such that every pixel had a value in the $[-0.5; 0.5]$ interval and it was fed to pre-trained U-net and learnlet models. The noisy, original and denoised versions, as well as the subtraction of the latter to the original image are presented in Figure 7. Using the Mean Squared Error (MSE) metric, learnlets (MSE of 20.83) perform almost twice as well as U-nets (MSE of 41.25).

Hence, similarly to the previous experiment, the generalization is much better for learnlets than for U-nets.

6.1.6 Influence of the number of samples

In a lot of Computer Vision problems, training data is scarce. It is reasonable to think that a small network

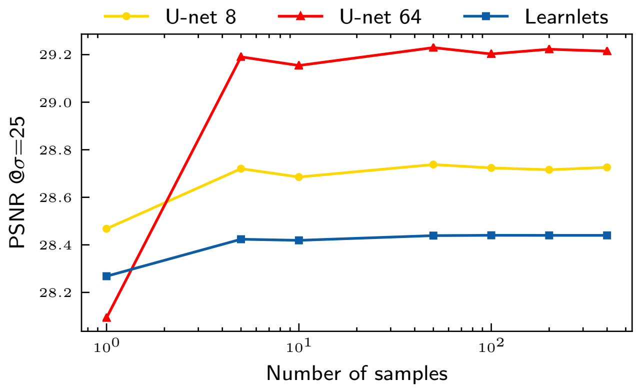


Fig. 8 The PSNR of the denoised image at $\sigma = 25$ added to the test images as a function of the number of samples used during training. The train noise standard deviation range was $[0; 55]$.

(i.e. low quantity of parameters) would start performing better than a deeper one as fewer samples become available. To test this, three models were examined: two U-nets of different sizes (8 and 64) and learnlets without exact reconstruction. The first aspect that can be mentioned about Figure 8 is that for the three networks the PSNR does not vary significantly when reducing the original number of samples all the way down to 50. Despite learnlets overcoming U-nets with 64 base filters for the lowest number of samples considered, they fail to outperform a U-net model with 8 base filters. It can be inferred that a reduced number of parameters tends to improve the robustness of a given neural network to the number of samples. In other words, relatively few samples are required to obtain top performance.

6.2 Qualitative results

6.2.1 Comparison with other methods

The Figure 9 shows that the learnlets suffer from some of the drawbacks of the wavelets like the creation of artifacts in the high frequency parts of the image. However the results are less blurred in comparison. Compared to the U-net, the learnlets are clearly suffering visually from a loss of contrast. This is a known effect of the soft thresholding which inherently biases the results. This could be improved by the use of reweighting [23] to further approach the hard-thresholding, which does not bias the results.

7 Conclusions

Pushing the limits of sparsity using massive learning and training data, we have proposed a novel neural net-

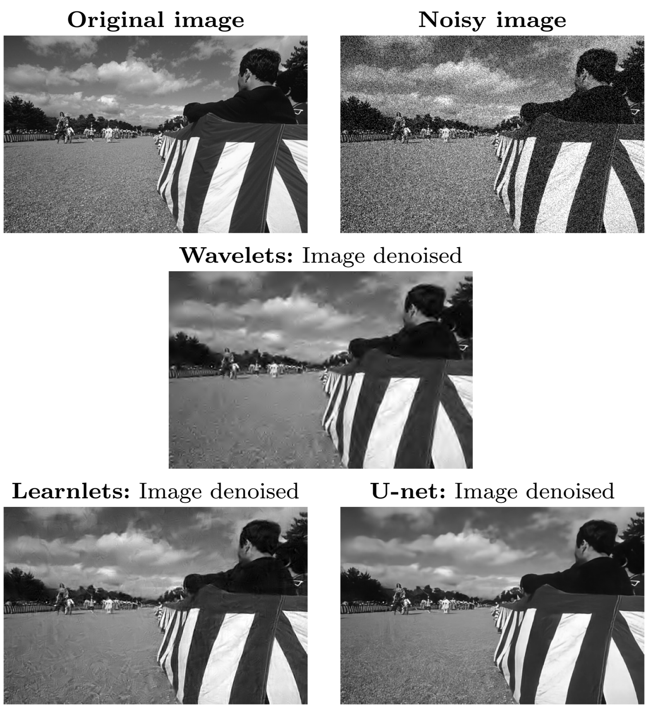

Original image
Noisy image
Wavelets: Image denoised
Learnlets: Image denoised
U-net: Image denoised

Fig. 9 Denoising results for a specific image in the BSD68 dataset. The noise standard deviation used was of 30. Parameters used for the methods are the same as for Figure 3.

work architecture – named *Learnlets* – with the following properties:

- Although their performances are inferior to U-nets, learnlets generalize better on noise levels that were not present in the training data and in the exact reconstruction domain. They also do better on the astronomical image which is different than the images present in the training dataset. In this case, the U-net's result is very poor.
- Learnlets can be forced to guarantee exact reconstruction when no thresholding is applied. This allows an embedding of the learnlets in applications where there is a need for guarantees of retrieval like in medical imaging. By contrast, U-nets suffer from a loss of performance at high noise levels.

Learnlets therefore bridge the gap between parsimony and neural networks, by combining massive learning and the computing power of GPUs as in neural networks, but keeping a perfect understanding of how results are obtained, with all the theoretical guarantees existing in the area of parsimony. Learnlets do clearly not outperform U-nets in denoising images compatible with the training dataset, which would indicate that massive learning and large datasets are not enough to explain the difference between sparse techniques and neural networks. The highly non-linear processing in

U-nets brings certainly a critical aspect in achieving high quality results.

Our main message is that we have clearly identified in this study a trade-off to be made in any application between performance and generalization. For performance, standard U-nets should clearly be chosen, while if the generalization is important, learnlets give the security of sparse techniques, with however using as well massive learning and GPU tools.

The future directions of this work are to try to adapt what has been successful in the sparse domain to this network. For example, reweighting [23] could help us to get rid of the loss of contrast. Curvelet filters [24] could also be used as a good initialisation or as complementary filters for the analysis. Apart from that, just like with the wavelets, many different types of noise – such as Poisson or spatially non-uniform white Gaussian noise – could be taken into account with a single model when implemented in an undecimated way, by adapting the thresholding function to the noise. Finally, it would be interesting to study the impact of incorporating learnlets as building blocks of the DIDN architecture [25].

Acknowledgements This work was granted access to the HPC resources of IDRIS under the allocation 2021-AD011011554 made by GENCI.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[2] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[3] Jonas Adler and Ozan Oktem. "Learned primal-dual reconstruction". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1322–1332.

[4] Jure Zbontar et al. "fastMRI: An open dataset and benchmarks for accelerated MRI". In: *arXiv preprint arXiv:1811.08839* (2018).

[5] Tran Minh Quan, Thanh Nguyen-Duc, and Won-Ki Jeong. "Compressed sensing MRI reconstruction using a generative adversarial network with a cyclic loss". In: *IEEE transactions on medical imaging* 37.6 (2018), pp. 1488–1497.

[6] Jong Chul Ye, Yoseob Han, and Eunju Cha. "Deep convolutional framelets: A general deep learning framework for inverse problems". In: *SIAM Journal on Imaging Sciences* 11.2 (2018), pp. 991–1048.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[8] David L Donoho. "De-noising by soft-thresholding". In: *IEEE transactions on information theory* 41.3 (1995), pp. 613–627.

[9] Nina M Gottschling et al. "The troublesome kernel: why deep learning for inverse problems is typically unstable". In: *arXiv preprint arXiv:2001.01258* (2020).

[10] Ramzi Zaccharie et al. "Wavelets in the Deep Learning Era". In: *2020 28th European Signal Processing Conference (EUSIPCO)* (2021). ISSN: 2076-1465. DOI: `10.23919/Eusipco47968.2020.9287317`.

[11] Daniel Recoskie and Richard Mann. "Learning filters for the 2D wavelet transform". In: *Proceedings - 2018 15th Conference on Computer and Robot Vision, CRV 2018* (2018), pp. 198–205. DOI: `10.1109/CRV.2018.00036`.

[12] Dhruv Jawali, Abhishek Kumar, and See. "A Learning Approach for Wavelet Design". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2019, pp. 5018–5022. ISBN: 9781538646588.

[13] Luke Pfister and Yoram Bresler. "Learning Filter Bank Sparsifying Transforms". In: *IEEE Transactions on Signal Processing* 67.2 (2019), pp. 504–519. ISSN: 1053587X. DOI: `10.1109/TSP.2018.2883021`. arXiv: `1803.01980`.

[14] Fenglei Fan et al. "Soft-Autoencoder and Its Wavelet Shrinkage Interpretation". In: *arXiv preprint arXiv:1812.11675* (2018).

[15] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `http://tensorflow.org/`.

[16] Pablo Arbelaez et al. "Contour Detection and Hierarchical Image Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.5 (May 2011), pp. 898–916. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2010.161`. URL: `http://dx.doi.org/10.1109/TPAMI.2010.161`.

[17] David Martin et al. "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics". In: *Proc. 8th Int'l Conf. Computer Vision*. Vol. 2. 2001, pp. 416–423.

[18] Kai Zhang et al. "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising". In: *IEEE Transactions on Image Processing* 26.7 (2017), pp. 3142–3155. ISSN: 10577149. DOI: `10.1109/TIP.2017.2662206`. arXiv: `arXiv:1608.03981v1`.

[19] Stamatios Lefkimmiatis. "Universal denoising networks: a novel CNN architecture for image denoising". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 3204–3213.

[20] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. arXiv: `1412.6980 [cs.LG]`.

[21] Stephane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[22] Samuel Farrens et al. "PySAP: Python Sparse Data Analysis Package for Multidisciplinary Image Processing". In: *arXiv preprint arXiv:1910.08465* (2019).

[23] Emmanuel J Candes, Michael B Wakin, and Stephen P Boyd. "Enhancing sparsity by reweighted l1 minimization". In: *Journal of Fourier analysis and applications* 14.5-6 (2008), pp. 877–905.

[24] Jean-Luc Starck, Emmanuel J Candes, and David L Donoho. "The curvelet transform for image denois-

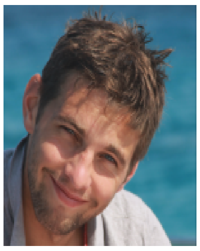
ing". In: *IEEE Transactions on image processing* 11.6 (2002), pp. 670–684.

[25] Songhyun Yu, Bumjun Park, and Jechang Jeong. "Deep iterative down-up CNN for image denoising". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

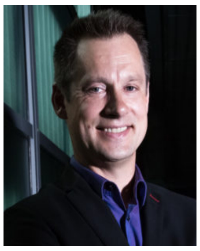
Zaccharie Ramzi is a PhD student working on designing Deep Learning models for MRI reconstruction as part of 3 teams: Parietal (Inria), NeuroSpin and CosmoStat (CEA-Saclay). He graduated from Telecom Paris and received an MSc in machine learning from ENS Paris-Saclay (MVA). He is interested in Artificial Intelligence, Computer Vision, Deep Learning and AI for Healthcare.

Kevin Michalewicz is an engineering student at IMT Atlantique, France and Facultad de Ingeniería de la UBA, Argentina. He is doing an internship at the CosmoStat laboratory (CEA-Saclay) about neural networks. Some of his main interests include Machine Learning, Image and Signal Processing and Physics in general.

Jean-Luc Starck is Director of Research and head of the CosmoStat laboratory at the Institute of Research into the Fundamental Laws of the Universe, Département d'Astrophysique, CEA-Saclay, France. He has a PhD from Nice Observatory and an Habilitation from University Paris XI. He is strongly involved in the Euclid ESA space mission.



Thomas Moreau is a researcher in the Parietal team at Inria Saclay, working on unsupervised learning for time series and on deep learning methods applied to solving inverse problems. He did his PhD in the CMLA at ENS Paris-Saclay. His research interests touch several areas of Machine Learning, Signal Processing and High-Dimensional Statistics.



Philippe Ciuciu is Research Director at NeuroSpin, the largest high field MRI center dedicated to cognitive and clinical neuroscience in France. His research interests are inter-disciplininary ranging from signal and image processing to functional brain imaging for applications to cognitive neuroscience and clinical trials in Alzheimer's disease and neurological disorders.

σ	**Original**	**Learnlets**	**U-net 128**	**Wavelets**
0.0001	128.13	124.69	53.29	127.30
5.0	34.15	36.51	37.76	35.76
15.0	24.61	30.87	31.67	29.56
20.0	22.11	29.55	30.27	28.25
25.0	20.17	28.54	29.24	27.32
30.0	18.59	27.74	28.43	26.61
50.0	14.15	25.58	26.31	24.79
75.0	10.63	23.90	24.71	23.46
85.0	9.54	23.38	23.74	23.06
95.0	8.58	22.93	22.59	22.71
100.0	8.13	22.71	22.14	22.56

Table 2 PSNR for different standard deviations of the noise added to the test images for every model in Figure 3 and the original noisy images.

A U-net architecture

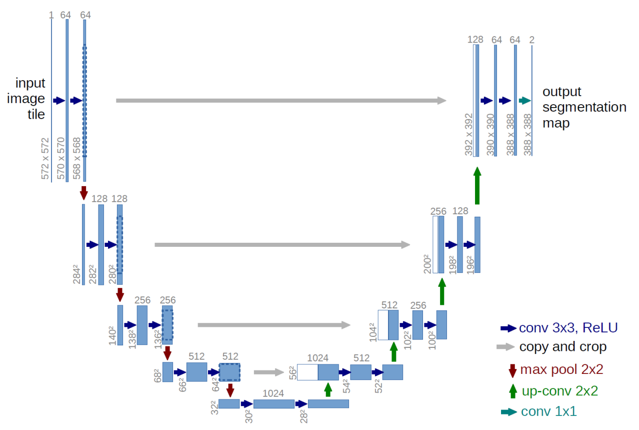


Fig. 10 The U-net architecture. The amount of channels of each feature map is indicated on the top of the blue rectangles. In this case, the number of base filters is 64. Figure from [1].

The Figure 10 shows the U-net architecture. It consists of:

– A contracting path that allows to capture context by applying convolutions, non-linearities (ReLU) and max pooling for downsampling.
– An expanding path that contains upsampling and convolution operations.

B Values of figures

In Tables 2, 3, 4 and 5 there can be found all the PSNR values as a function of σ for each model in Figures 3, 4, 5 and 6 respectively. Conversely, Table 6 corresponds to the information presented in Figure 8.

σ	**Original**	**Learnlets**	**Learnlets No exact recon.**
0.0001	128.13	124.69	43.56
5.0	34.15	36.51	36.32
15.0	24.61	30.87	30.81
20.0	22.11	29.55	29.46
25.0	20.17	28.54	28.44
30.0	18.59	27.74	27.63
50.0	14.15	25.58	25.45
75.0	10.63	23.90	23.76
85.0	9.54	23.38	23.21
95.0	8.58	22.93	22.71
100.0	8.13	22.71	22.50

Table 3 PSNR for different standard deviations of the noise added to the test images for both models in Figure 4 and the original noisy images.

σ	**Original**	**U-net 4**	**U-net 8**	**U-net 64**	**U-net 128**	**Wavelets**
0.0001	128.13	39.89	43.67	52.60	53.29	127.30
5.0	34.15	35.58	36.65	37.73	37.76	35.76
15.0	24.61	30.53	31.06	31.65	31.67	29.56
20.0	22.11	29.23	29.72	30.24	30.27	28.25
25.0	20.17	28.26	28.73	29.21	29.24	27.32
30.0	18.59	27.50	27.94	28.41	28.43	26.61
50.0	14.15	25.49	25.88	26.30	26.31	24.79
75.0	10.63	17.86	18.74	24.55	24.71	23.46
85.0	9.54	14.93	15.74	23.52	23.74	23.06
95.0	8.58	13.25	13.70	22.25	22.59	22.71
100.0	8.13	12.60	12.94	21.65	22.14	22.56

Table 4 PSNR for different standard deviations of the noise added to the test images for all the models in Figure 5 and the original noisy images.

σ	**Original**	**U-net 64**	**U-net 64 Exact recon.**
0.0001	128.13	52.60	127.73
5.0	34.15	37.73	37.77
15.0	24.61	31.65	31.63
20.0	22.11	30.24	30.22
25.0	20.17	29.21	29.18
30.0	18.59	28.41	28.38
50.0	14.15	26.30	26.30
75.0	10.63	24.55	22.64
85.0	9.54	23.52	19.61
95.0	8.58	22.25	16.53
100.0	8.13	21.65	15.11

Table 5 PSNR for different standard deviations of the noise added to the test images for both models in Figure 6 and the original noisy images.

Samples	**U-net 8**	**U-net 64**	**Learnlets No exact recon.**
1	28.47	28.09	28.27
5	28.72	29.19	28.42
10	28.69	29.15	28.42
50	28.74	29.23	28.44
100	28.72	29.20	28.44
200	28.71	29.22	28.44
400	28.73	29.21	28.44

Table 6 PSNR at $\sigma = 25$ added to the test images as a function of the number of samples used during training for the three models in Figure 8.