WAVELETS IN THE DEEP LEARNING ERA

Zaccharie Ramzi*†‡, Jean-Luc Starck*, Thomas Moreau†, Philippe Ciuciu†‡
*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
Email: zaccharie.ramzi@inria.fr
‡CEA/NeuroSpin, Bat 145, F-91191 Gif-sur Yvette, France

***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 data sets for training. 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 with learning, and comparing the results with U-nets. We present a new network architecture, 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 and show it is competitive with learning-based models.**

***Index Terms*—Machine Learning, Deep Learning, Neural Networks, Wavelets, Denoising, Image restoration**

I. 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 analyse the signal at different resolutions. Their main difference is the application of non-linearity. Indeed, where 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.

The idea of this work was therefore to build a network which made use of one of the strongest advantages of neural networks, learning via gradient descent to enhance the expressive power of wavelets, while keeping some of the understanding we have of wavelets, in particular the denoising process. We term this network '*Learnlets*'. We chose to test this network on a denoising problem, a task where wavelets have historically well-performed but are now overtaken by deep learning approaches.

The full implementation of our method is open source in Python[1].

II. Related Work

Different studies have attempted to work at the intersection of wavelets and neural networks. In [10], 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 and only a simply high-pass and low-pass filter pair is learned. 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 than simply max-pooling/bilinear upsampling. [11] 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.

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

III. Learnlets, the model

Let $x \in \mathbb{R}^{n \times n}$ be an image. Let $\tilde{x} = x + \epsilon$ be the version of this image corrupted by an additive white Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ 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 $\theta = (\theta_S, \theta_t, \theta_A) \in \Theta_m$, we defined the learnlets as function f_θ from $(\mathbb{R}^{n \times n} \times \Sigma)$ to $\mathbb{R}^{n \times n}$:

$$f_\theta(\tilde{x}, \sigma) = S_{\theta_S}\left(T_{\theta_T}\left(A_{\theta_A}\left(\tilde{x}\right), \sigma\right)\right) \tag{1}$$

where we have:

1) A_{θ_A}, the analysis function defined in III-A.
2) T_{θ_T}, the thresholding function defined in III-B.
3) S_{θ_S}, the synthesis function defined in III-C.

An illustration of the learnlets is given in Figure 1.

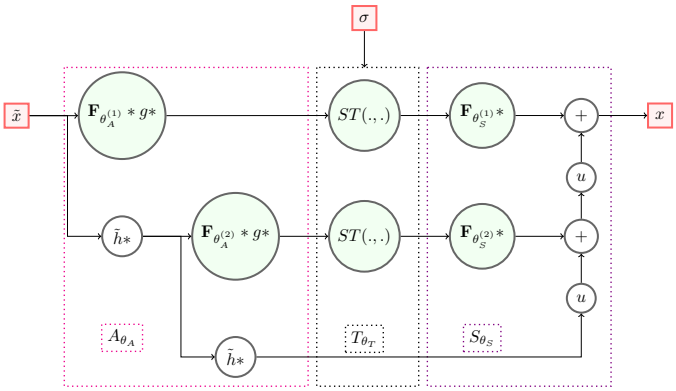

Figure 1. Schematic representation of the learnlets model, with $m = 2$ scales. The red nodes are inputs/outputs. The lightly green nodes correspond to functions whose parameters can be learned. Note that the standard deviation of the noise before thresholding is not learned but rather estimated, and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

A. Analysis

Intuitively, one can see the analysis linear function as the equivalent of the wavelet transform with some learned filters. It is important to note that the analysis function is linear. The analysis function is defined as:

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

where we have:

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

For ease of manipulation we rewrite $A_{\theta_A}(\tilde{x}) = ((\mathbf{d}_i)_{i=1}^m, c)$, 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 c the coarse coefficients.

B. 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:

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

where $t_{ij}(d, \sigma) = \hat{\sigma}_{ij} ST\left(\frac{1}{\hat{\sigma}_{ij}} d_{ij}, \theta_T^{(ij)} \sigma\right)$, 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.
- $ST(d, s)$ is the soft-thresholding function applied point-wise on d with threshold s: $ST(d, s) = \text{sign}(d) \max(|d| - s, 0)$.

C. 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:

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

where $S_\emptyset(\emptyset, c) = c$ and:

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

D. 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 $[0, 5]$.

E. Learning

The optimization problem is given as:

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

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

F. Learnlets with exact reconstruction

Exact reconstruction guarantees that if no noise is present, the signal will be reconstructed as is. 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 g filter. The operation carried out by the network, without thresholding can be written as:

$$x_{out}^{(i)} = \sum_{j=1}^{N} \mathbf{F}_{\theta_S^{(i,j)}} * \mathbf{F}_{\theta_A^{(i,j)}} * x_{in} \tag{6}$$

where N is the number of filters at that scale. Since we have $\mathbf{F}_{\theta_A^{(i,1)}} = Id$, we can also fix the corresponding synthesis filter $\mathbf{F}_{\theta_S^{(i,1)}} = Id - \sum_{j=2}^{N} \mathbf{F}_{\theta_S^{(i,j)}} * \mathbf{F}_{\theta_A^{(i,j)}}$. This trivially gives without thresholding, $x_{out} = x_{in}$. We implemented this constraint in the network, allowing to learn a different thresholding level for this filter. We compare in the next section the performance against a network which didn't feature this forced exact reconstruction.

IV. Data and Experiments

The implementation was done in Python 3.6, using the TensorFlow 2.1 framework [12] for model design. The training was done on a computer equipped with a single GPU Quadro P5000 with 16GB of RAM.

A. Data

The data used was the BSD500 dataset [13]. 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 [14] images was left out. We used BSD68 as the test dataset. This choice is motivated by the fact that many other denoising studies [15], [16] use this dataset for comparison.

B. 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]. oise 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.

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

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: $\{0.0001, 5, 15, 20, 25, 30, 50, 55, 60, 75\}$. This allowed us to test the performance of our method in different noise level settings.

C. Model and training

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 31 million parameters.

Unless specified otherwise, the learnlets parameters were chosen as:

- $m = 5$ scales.
- 256 learnable analysis filters + 1 fixed analysis filters being just the identity, $\mathbf{F}_{\theta_A^{(i)}}$, of size 11×11.
- 257 learnable synthesis filters, $\mathbf{F}_{\theta_S^{(i)}}$, 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 one hundredth of the size of the U-net.

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 [17]. 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 1 day for 500 epochs each.

D. Evaluation

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

$$PSNR(x, \hat{x}) = -10 \log_{10} \|x - \hat{x}\|_2^2 \tag{7}$$

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

2) Testing: In addition, the networks were compared to wavelets denoising [18] and BM3D [19]. We used the code of PySAP [20] to implement the wavelets denoising. 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). The results for BM3D were taken from various sources [15, 21, 16] while the qualitative results were generated using the code provided by the authors of [19][3].

[3]http://www.cs.tut.fi/~foi/GCF-BM3D/

V. Results

A. Quantitative results

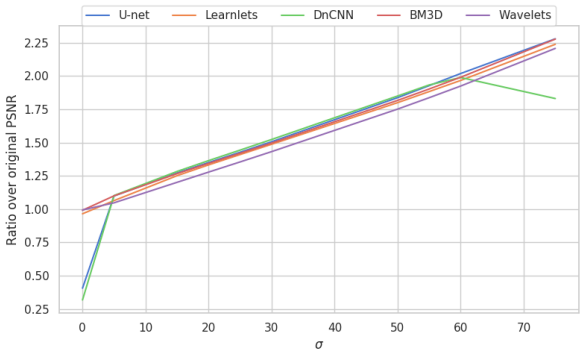

Figure 2. Ratio of the denoised image PSNR compared to the original noisy image PSNR for different standard deviations of the noise added to the test images for all considered models. The train noise standard deviation range was $[0; 55]$.

Model name	BM3D	Wavelets	Learnlets	U-net
Denoising runtime in ms (std)	10800 (223)	274 (21)	106 (12)	64 (1)

Table I
Runtimes of the different models for the denoising of one image. Parameters used are the same as Figure 2.

1) Comparison with other methods: We compared the U-net and the learnlets against algorithms not involving learning, namely BM3D and wavelets shrinkage, and another neural network that was for long state of the art in image denoising, DnCNN [15]. Figure 2 shows that for a large part of the band, $[5; 55]$ where they have been trained, the learnlets are competitive compared to BM3D PSNR-wise. Keeping the same formula as the wavelets, we manage to reach the performance of state-of-the-art classical methods that are relatively complex. In this same band, the wavelets have performance that are 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, both learning methods gets degraded performances compared with wavelet and BM3D. In this setting, the denoiser must act as the identity. For the learnlets this can be fixed by forcing exact reconstruction as can be seen in the next section. Finally, we can see that for unseen test noise levels (i.e. 70), the performances of DnCNN drop significantly while the learnlets keep relatively good performances. This suggests that the learnlets generalise much better on unseen noise levels than U-nets.

In addition, we can see in Table V-A1 that the learnlets benefit from their GPU implementation and run much faster than the wavelets and BM3D.

2) Learnlet with exact reconstruction: We saw in Figure 2 that the learnlets without exact reconstruction compete with classical methods on small noise standard deviations. Figure 3 shows that the performance of the network with forced exact reconstruction is almost the same as the one without forced

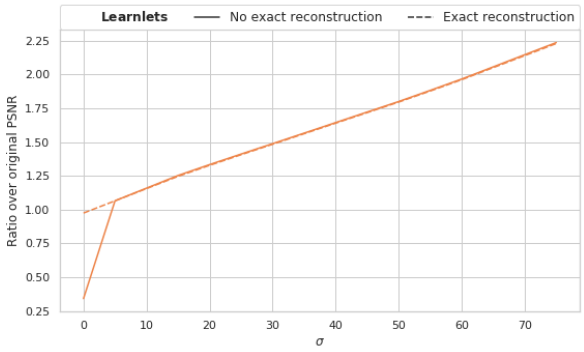

Figure 3. Ratio of the denoised image PSNR compared to the original noisy image PSNR for different standard deviations of the noise added to the test images for learnlets with and without forcing exact reconstruction. The train noise standard deviation range was $[0; 55]$. The number of filters used was 64.

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, $\mathbf{F}_{\theta_A^{(i,1)}}$. This allows to guarantee, in this case, no loss of information in the signal if it is clearly present.

B. Qualitative results

The Figure 4 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 or BM3D, 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 to further approach the hard thresholding, which doesn't bias the results.

VI. Conclusions

Pushing the limits of sparsity using massive learning and training data, we have proposed a novel neural network architecture – named learnlet – with the following properties:

- Despite very simple used concepts (linear filters and soft thresholding), it allows performances similar to relatively complex sparsity based methods like BM3D.
- Although its performances are inferior to DnCNN, learnlets generalize better than DnCNN on noise levels that were not present in the training data and in the exact reconstruction domain.
- Learnlets benefit from a fast implementation lacking for both wavelets and BM3D.

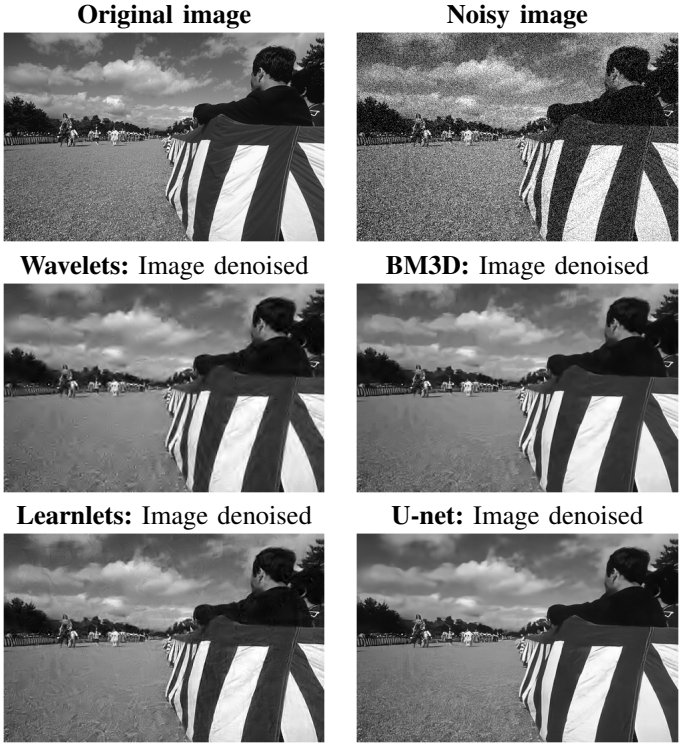

Figure 4. Denoising results for a specific image in the BSD68 dataset. The noise standard deviation used was of 30. Parameters used for the methods are the same as for Figure 2.

- 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.

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.

The future directions of this work are to try to adapt what has been successful in the wavelets domain to this network For example, reweighting [22] could help us to get rid of the loss of contrast. Curvelet filters [23] could also be used as a good initialisation or as complementary filters for the analysis. Finally, 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.

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] 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.

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

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

[13] 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.

[14] D. 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.

[15] 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.

[16] 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.

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

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

[19] Kostadin Dabov et al. "Image denoising with block-matching and 3D filtering". In: *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*. Vol. 6064. International Society for Optics and Photonics. 2006, p. 606414.

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

[21] Kai Zhang, Wangmeng Zuo, and Lei Zhang. "FFDNet: Toward a fast and flexible solution for CNN-Based image denoising". In: *IEEE Transactions on Image Processing* 27.9 (2018), pp. 4608–4622. ISSN: 10577149. DOI: 10.1109/TIP.2018.2839891.

[22] 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.

[23] Jean-Luc Starck, Emmanuel J Candes, and David L Donoho. "The curvelet transform for image denoising". In: *IEEE Transactions on image processing* 11.6 (2002), pp. 670–684.