Wavelet/shearlet hybridized neural networks for biomedical image restoration

Bart Goossens, Hiêp Luong and Wilfried Philips

Ghent University, Dept. of Telecommunications and Information Processing, imec-IPI-UGent, Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

Keywords: Shearlets, neural networks, image restoration

ABSTRACT

Recently, new programming paradigms have emerged that combine parallelism and numerical computations with algorithmic differentiation. This approach allows for the hybridization of neural network techniques for inverse imaging problems with more traditional methods such as wavelet-based sparsity modelling techniques. The benefits are twofold: on the one hand traditional methods with well-known properties can be integrated in neural networks, either as separate layers or tightly integrated in the network, on the other hand, parameters in traditional methods can be trained end-to-end from datasets in a neural network "fashion" (e.g., using Adagrad or Adam optimizers). In this paper, we explore these hybrid neural networks in the context of shearlet-based regularization for the purpose of biomedical image restoration. Due to the reduced number of parameters, this approach seems a promising strategy especially when dealing with small training data sets.

1. INTRODUCTION

In the past decades, many image restoration techniques have been proposed, ranging from spatial domain filters (e.g., the bilateral filter[1], the guided filter[2]) to sparsity transform based methods (e.g., wavelets[3], dual-tree complex wavelets[4,5], steerable pyramids[6], curvelets[7] and shearlets[8]). Wavelet-based methods allow adapting the algorithm to the spatial content of the image (e.g., edge-adaptive methods or nonstationary noise suppression) while at the same time permit frequency (scale) selective processing. Many sparsity-based restoration techniques require models for the degradation (e.g., noise, blur) and for the underlying image (e.g., a statistical image model). Many of these models contain several parameters for which accurate estimation tends to be costly. In order to apply an existing restoration method to another problem, the underlying models need to be changed or tweaked, which can sometimes be a time-consuming task.

Recently, convolutional neural network (CNN)-based restoration architectures[9,10,11], have equalled or surpassed the performance of aforementioned techniques. Moreover, CNNs can be applied in a wide range of restoration problems, require limited user intervention, given that a training set can be obtained that is sufficiently large. Despite CNN-based restoration techniques being preferable over other techniques, it comes at the cost of a large number of parameters (typically millions), requiring large training data sets and relatively high computational demands for both training and inference, at the very minimum requiring a powerful high-end GPU to be employed.

To combine the best of both worlds, it is useful to integrate prior knowledge in the CNN-based methods (e.g., a noise power spectral density that has been measured beforehand, image statistics, prior knowledge on the characteristics of the degradation) in order to reduce the number of parameters, the subsequent training time and the inference computation time. Similarly, the "classical" restoration methods often have sets of parameters that are estimated empirically from the observed image; these methods can benefit from end-to-end training as is common in neural networks.

The incorporation of prior knowledge in existing CNNs is not always straightforward, as the prior knowledge cannot always be directly mapped onto "layers" that can be added to the neural networks. A conventional approach consists of including regularization that penalizes parameter values differing too much from the prior information. However, this approach does not solve the large parameter estimation problem. There exists different network architectures that encompass the prior knowledge and that have fewer parameters. However, it is not clear what the relationship is between prior knowledge and the network architecture. In this paper, we will demonstrate this relationship for the problem of joint denoising and deblurring under a shearlet prior.

Recently, the emerging paradigm of differentiable programming (DP)[12] allows classical techniques to be trained using neural network-based training methods. DP allows functions specified in a programming language to be differentiated, typically using the technique of algorithmic differentiation (AD). AD allows gradients of cost functions to be calculated very efficiently and accurately. The AD process is recursive: when a function calls another function, the other function will be differentiated as a consequence of applying the chain rule. DP can not only be applied to train neural networks parameters, but in principle to any cost function that is differentiable and that can be specified in a programming language. This allows easy implementation of new layers for which the backpropagation is derived algorithmically, and optimized to the target computation platform (e.g., a GPU). For our purposes, DP allows us to investigate novel network architectures with reduced number of parameters, without being restricted by the layers provided by deep learning frameworks.

In this paper, we present "hybridized wavelet/shearlet neural networks", which essentially apply a small neural network to every subband of a wavelet/shearlet decomposition. Each subband is subjected to the same neural network. At the same time, it is possible to deal with statistical dependencies between different subbands. This approach allows to reduce the number of parameters significantly, while allowing prior knowledge on the ideal image (e.g., its sparsity) and degradation to be integrated. By applying a convex optimization trick, we show that a hybridized wavelet/shearlet neural network with a L^2 cost criteration and under an additional wavelet/shearlet L^1 regularization is equivalent to the use of soft-thresholding as non-linear activation function in the transform domain.

The remainder of this paper is structured as follows: in Section 2 we explain the design of our 'hybridized' network architecture. In Section 3 give some implementation details of our method. The results are shown and discussed in Section 4. Finally, Section 5 concludes this paper.

2. DESIGN OF 'HYBRIDIZED' NEURAL NETWORK ARCHITECTURES

To solve a particular image restoration problem (e.g., denoising, deblurring, demosaicking, ...), we assume that a set of degraded images $\{\boldsymbol{y}_i\}_{i\in\{1,...,I\}}$ are available, together with the corresponding ground-truth images $\{\boldsymbol{x}_i\}_{i\in\{1,...,I\}}$. All images have size $M \times N$. In this notation, each patch $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$ is a vector of length MN (e.g., obtained through column stacking).

For real restoration applications in which no groundtruth images are available, there exist various techniques for obtaining "ideal" images that can be used as groundtruth. For example, in image denoising, an ideal image of a static scene can be estimated as an average over a sufficiently large number of exposures. Additionally, the camera lens point spread function and noise characteristics may be modeled to simulate degraded images based on a set of ideal images. This is the approach we will follow in the remainder of this paper.

We will put forward the following design goals for our technique: 1) the possibility to be used in a wide number of restoration tasks, 2) end-to-end training and 3) high computational efficiency with limited memory use. The third goal is to ensure that the method can be used on high-resolution video in real-time, e.g., in mobile devices.

2.1 Design of the restoration operator

We design an operator $\mathrm{H}_{\boldsymbol{\alpha}}$ with parameters $\boldsymbol{\alpha}$ to perform restoration of patches $\boldsymbol{y}_i$ (during training) and previously unseen images $\boldsymbol{y}$ (during inference):

$$\hat{\boldsymbol{x}}_i = \mathrm{H}_{\boldsymbol{\alpha}}\boldsymbol{y}_i \quad \forall i \in \{1,...,I\}. \tag{1}$$

To match the ground-truth patches $\boldsymbol{x}_i$, we want $\hat{\boldsymbol{x}}_i$ to be as close as possible to $\boldsymbol{x}_i$, in a certain norm, such as the L^2-norm. Additionally, we wish to integrate prior knowledge with respect to the ground-truth images in the training and inference procedures. This can be achieved by regularizing the cost function to maximize sparsity of the estimated image in a certain transform domain.

When using the L^2 error norm in combination with the L^1 sparsity norm, the above reconstruction problem can be mathematically expressed as:

$$\min_{\boldsymbol{\alpha}} \sum_{i=1}^{I} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 + \lambda \sum_{i=1}^{I} |\Psi\hat{\boldsymbol{x}}_i|_1 \quad \text{subject to} \quad \hat{\boldsymbol{x}}_i = \mathrm{H}_{\boldsymbol{\alpha}}\boldsymbol{y}_i \quad \forall i \in \{1,...,I\}, \tag{2}$$

where Ψ is a sparsity transform (e.g., wavelets, shearlets, curvelets, ...) and where λ is a regularization parameter that determines the importance of the L^1-norm of the sparsity transform coefficients. We use overcomplete sparsity transforms,

which generally offer better restoration performance due to lack of (or less) aliasing compared to their complete counterparts. Once the restoration operator parameters $\boldsymbol{\alpha}$ have been estimated, the restoration operator $\mathrm{H}_{\boldsymbol{\alpha}}$ can be used to restore images that are outside of the training set, under the premise that the same or similar degradation is present in these images. If the restoration operator $\mathrm{H}_{\boldsymbol{\alpha}}$ has sufficiently computationally efficient implementation, entire video sequences can be processed using the operator, in real-time.

When the restoration operator $\mathrm{H}_{\boldsymbol{\alpha}}$ is a CNN, (2) is in fact nothing more than a CNN restoration problem with an extra sparsity regularization term. In several existing CNN restoration methods a deep CNN is used with many layers (for an overview, see Tian et al.[13]). In other methods (e.g., Burger et al.[9]), a shallow neural network is used with fully connected layers. Often, the parameter count (i.e., the length of the vector $\boldsymbol{\alpha}$) is very large, with millions of parameters to be estimated. To reduce the number of parameters we "sparsify" the restoration operator $\mathrm{H}_{\boldsymbol{\alpha}}$ as well.

In general it is difficult to directly design $\mathrm{H}'_{\boldsymbol{\alpha}}$ such that $\Psi^{\mathrm{T}}\mathrm{H}'_{\boldsymbol{\alpha}}\Psi \approx \mathrm{H}_{\boldsymbol{\alpha}}$ where $\mathrm{H}_{\boldsymbol{\alpha}}$ is for example a neural network as in Burger et al.[9]. Inspired by structured receptive field networks,[14] we circumvent this problem by integrating the sparsity transform in the network architecture, keeping the sparsity transform *fixed* (non-learnable) and by applying small (learnable) CNN layers to the sparsity transform subbands. This results in a composite CNN with reduced number of parameters compared to a CNN without fixed basis, but with similar function approximation capabilities for the selected image sparsity prior. The overall restoration operator is then the composition of the adjoint sparsity transform, the transform domain restoration operator and the forward sparsity transform $\Psi^{\mathrm{T}}\mathrm{H}'_{\boldsymbol{\alpha}}\Psi$.

We have found that it is useful to incorporate cross-scale and cross-orientation convolutions to maximally exploit the statistical dependencies between the sparsity coefficients. Note that such dependencies occur because we are using overcomplete sparsity transforms. We will take special care in designing our sparsity transform for controlling such dependencies (see Section 3). One extra modification that we make, is the addition of an image domain affine operator $\mathrm{U}_{\boldsymbol{\beta}}$ with parameters $\boldsymbol{\beta}$, prior to the sparsity transform. This operator can be learnable or fixed. Certain types of degradation, like the presence of missing pixels, a convolution or contrast adjustment, cannot easily be undone by sparsity transform domain operations. By incorporating an additional transformation (from which the parameters are also to be trained), we can control adapt the restoration method to the problem at hand. For example,

- *Deblurring*: in this case $\mathrm{U}_{\boldsymbol{\beta}}$ functions as an inverse filter, which is trained from the input dataset. Such inverse filter causes the blurring to be undone, at the cost of amplifying the image noise. The image noise is then suppressed in the subsequent stage. This process is similar to the inverse filter factorization of the Wiener filter expression, where here the inverse filter is trained from the input data.
- *Demosaicking/missing pixels*: here $\mathrm{U}_{\boldsymbol{\beta}}$ provides an initial estimate for the missing pixels, that is later refined in the subsequent stage.
- *Interpolation/upsampling*: similar to the previous case, $\mathrm{U}_{\boldsymbol{\beta}}$ performs an initial interpolation (e.g., sinc with Blackman-Harris window).
- In addition, $\mathrm{U}_{\boldsymbol{\beta}}$ may incorporate other linear or affine corrections, such as brightness and contrast enhancement, color corrections, ...

We consider restoration operators $\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ and $\mathrm{H}'_{\boldsymbol{\alpha}}$ defined by the intertwining relationship

$$\Psi\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}. \tag{3}$$

which means that applying a sparsity operator to the restoration operator $\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ is equivalent to applying a preprocessing transform, a sparsity transform and restoration $\mathrm{H}'_{\boldsymbol{\alpha}}$ within the transform domain. Here, $\mathrm{H}'_{\boldsymbol{\alpha}}$ is a CNN with a small number of layers and a small number of parameters, which is applied to every subband of the sparsity transform. If the sparsity transform obeys Parsevals' property $\Psi^{\mathrm{T}}\Psi = \mathrm{I}$ (which is the case, e.g., for wavelets and shearlets), we have the composition

$$\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \underbrace{\Psi^{\mathrm{T}}}_{\substack{\text{non-}\\\text{learnable}}} \underbrace{\mathrm{H}'_{\boldsymbol{\alpha}}}_{\text{learnable}} \underbrace{\Psi}_{\substack{\text{non-}\\\text{learnable}}} \underbrace{\mathrm{U}_{\boldsymbol{\beta}}}_{\substack{\text{learnable or}\\\text{non-learnable}}}, \tag{4}$$

which describes well the structure of our restoration operator.

2.2 Training phase

Now that we have determined the structure of our restoration operator, we determine an efficient method to estimate the model parameters from a training set. For simplicity, we will focus on sparsity transforms obeying Parseval's property (also known as Parseval frames). These transforms have demonstrated excellent denoising performance in past work on image denoising. Parsevals' property has the consequence that: $\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|_2^2 = \|\Psi(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)\|_2^2$, so that equations (2) and (3) can be expressed entirely in the sparsity domain:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{i=1}^{I} \frac{1}{2} \|\Psi(\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)\|_2^2 + \lambda \sum_{i=1}^{I} |\Psi\hat{\boldsymbol{x}}_i|_1 \quad \text{subject to} \quad \Psi\hat{\boldsymbol{x}}_i = \mathrm{H}'_{\boldsymbol{\alpha}} \Psi \mathrm{U}_{\boldsymbol{\beta}} \boldsymbol{y}_i. \tag{5}$$

In case $\mathrm{U}_{\boldsymbol{\beta}}$ is an identity operator or a non-learnable transformation ($\mathrm{U}_{\boldsymbol{\beta}} = \mathrm{U}$), the solution can be entirely obtained in the sparsity domain by precomputing $\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i$. In this case, the optimization problem (5) can be decomposed into a number of independent optimization problems, and they can be learned in parallel (or even in a distributed fashion). Alternatively, for multiresolution sparsity transforms that perform decimation *across scale*, the number of network parameters can be reduced by tying the CNN parameters across scale. This means that the same parameter values are used on each scale of the sparsity transform. We will consider this further in Subsection 3.1. In the following, for generality, we assume that $\mathrm{U}_{\boldsymbol{\beta}}$ is learnable and/or that some parameters are tied across scale, therefore, the solution method must necessarily alternate between the image domain and the sparsity domain.

In essence, (5) can be minimized directly with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ using a gradient descent method. Because we are dealing with a convex optimization problem in $\hat{\boldsymbol{x}}_i$, we solve (5) first with respect to $\hat{\boldsymbol{x}}_i$ and we minimize over a new cost function with $\hat{\boldsymbol{x}}_i$ replaced by the corresponding minimum (alternating optimization). For this purpose, we introduce a splitting variable $\boldsymbol{z}_i = \Psi\hat{\boldsymbol{x}}_i = \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i$. Applying the augmented Lagrangian method, we obtain:

$$\min_{\{\boldsymbol{\alpha},\boldsymbol{\beta}\}\cup\{\boldsymbol{z}_i,\boldsymbol{p}_i\}_{i\in\{1,\ldots,I\}}} \frac{1}{2}\sum_{i=1}^{I} \|\Psi\boldsymbol{x}_i - \boldsymbol{z}_i\|_2^2 + \sum_{i=1}^{I} \boldsymbol{p}_i^{\mathrm{T}}(\boldsymbol{z}_i - \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i) + \frac{\mu}{2}\sum_{i=1}^{I} \left\|\boldsymbol{z}_i - \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i\right\|_2^2 + \lambda\sum_{i=1}^{I}|\boldsymbol{z}_i|_1 \tag{6}$$

where μ is a penalizer parameter and where $\{\boldsymbol{p}_i\}_{i\in\{1,\ldots,I\}}$ is a set of Lagrange multipliers. It can be shown that a closed-form minimizer for the splitting variable $\boldsymbol{z}_i$ in (6) is given by

$$\boldsymbol{z}_i^{\star} = \text{softthreshold}\left(\frac{\Psi\boldsymbol{x}_i + \mu\mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i - \boldsymbol{p}_i}{1+\mu}, \frac{\lambda}{1+\mu}\right), \tag{7}$$

so that (6) can solved using alternating optimizations:

$$\boldsymbol{z}_i^{(j+1)} = \text{softthreshold}\left(\frac{\Psi\boldsymbol{x}_i + \mu\mathrm{H}'^{(j)}_{\boldsymbol{\alpha}}\Psi\mathrm{U}^{(j)}_{\boldsymbol{\beta}}\boldsymbol{y}_i - \boldsymbol{p}_i^{(j)}}{1+\mu}, \frac{\lambda}{1+\mu}\right) \tag{8}$$

$$(\hat{\boldsymbol{\alpha}}^{(j+1)}, \hat{\boldsymbol{\beta}}^{(j+1)}, \hat{\boldsymbol{p}}_i^{(j+1)}) = \min_{\{\boldsymbol{\alpha},\boldsymbol{\beta}\}\cup\{\boldsymbol{p}_i\}_{i\in\{1,\ldots,I\}}} \frac{1}{2}\sum_{i=1}^{I}\left\|\Psi\boldsymbol{x}_i - \boldsymbol{z}_i^{(j+1)}\right\|_2^2 + \sum_{i=1}^{I}\boldsymbol{p}_i^{\mathrm{T}}(\boldsymbol{z}_i^{(j+1)} - \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i) + \frac{\mu}{2}\sum_{i=1}^{I}\left\|\boldsymbol{z}_i^{(j+1)} - \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}_i\right\|_2^2 \tag{9}$$

With this modification, the resulting technique bears a lot of similarity with the iterative soft-thresholding algorithm. Note that the objective function is (in general) not convex in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, therefore (9) is best solved using a stochastic gradient descent (SGD) method, such the Adam algorithm.[15]

2.3 Inference phase

Ideally, the inference can be carried out using (4), although in practice we have found that the resulting restored images are often noisy. The reason is that the optimization is carried out in the overcomplete sparsity transform domain, while for a given value of the cost function $\left\|\Psi\hat{\boldsymbol{x}} - \Psi\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}}\boldsymbol{y}\right\|_2^2$ multiple solutions exist in the image space. Of these solutions, some solutions are more sparse than others. To select a suitable candidate among these solutions, we use again sparsity regularization by minimizing the following cost function:

$$\min_{\hat{\boldsymbol{x}}} \frac{1}{2}\left\|\Psi\hat{\boldsymbol{x}} - \Psi\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}}\boldsymbol{y}\right\|_2^2 + \lambda|\Psi\hat{\boldsymbol{x}}|_1 = \min_{\hat{\boldsymbol{x}}} \frac{1}{2}\left\|\Psi\hat{\boldsymbol{x}} - \mathrm{H}'_{\boldsymbol{\alpha}}\Psi\mathrm{U}_{\boldsymbol{\beta}}\boldsymbol{y}\right\|_2^2 + \lambda|\Psi\hat{\boldsymbol{x}}|_1. \tag{10}$$

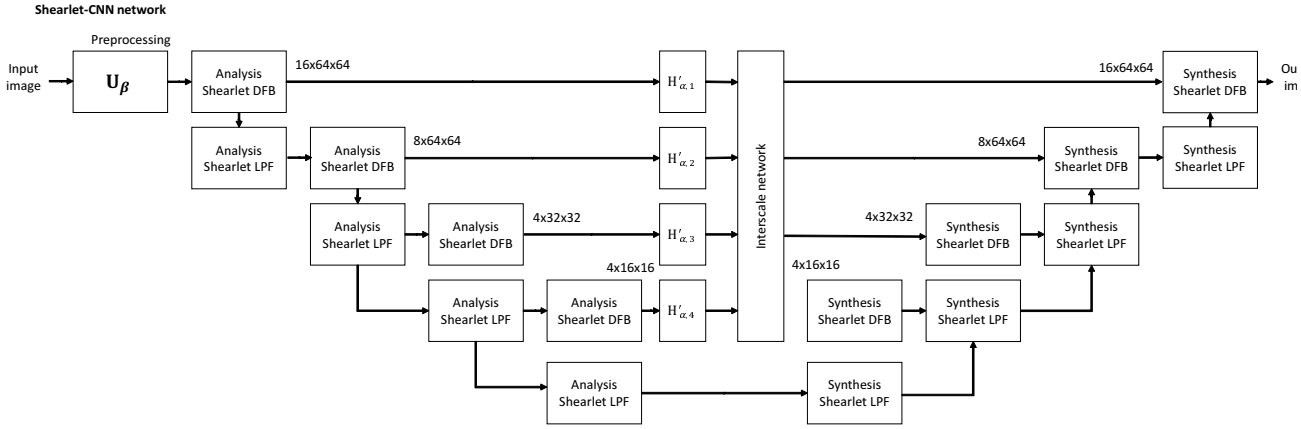


Figure 1. Hybrid shearlet-CNN network with number of orientation bands $K_1 = 16$, $K_2 = 8$, $K_3 = 4$, $K_4 = 4$. LPF=low pass filter, DFB=directional filter bank.

During inference, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are fixed, which simplifies the estimation of $\hat{\boldsymbol{x}}$ considerably. Similar as in the training phase, the solution can be determined using the augmented Lagrangian method, although this leads to an iterative algorithm to estimate $\hat{\boldsymbol{x}}$. Instead, aiming for a computationally simple approach, an approximate solution is given by:

$$\hat{\boldsymbol{x}} \approx \Psi^{\mathrm{T}} \mathrm{softthreshold}\left(\mathrm{H}'_{\boldsymbol{\alpha}} \Psi \mathrm{U}_{\boldsymbol{\beta}} \boldsymbol{y}, \lambda\right) \tag{11}$$

Note that for identity operators $\mathrm{H}'_{\boldsymbol{\alpha}} = \mathrm{I}$ and $\mathrm{U}_{\boldsymbol{\beta}} = \mathrm{I}$ we obtain classical sparsity transform denoising by soft-thresholding. Compared to the classical soft-thresholding method, our approach can be adapted to other restoration tasks, trained in an end-to-end fashion, while maintaining similar computational complexity.

2.4 Network architecture

An overview of the hybrid shearlet-CNN network performing the calculation $\Psi^{\mathrm{T}} \mathrm{H}'_{\boldsymbol{\alpha}} \Psi \mathrm{U}_{\boldsymbol{\beta}}$ is given in Figure 1. We use a four-level shearlet transform (see Section 3 for the details of the transform implementation). The input patch size is 64×64 and the patches are decimated after wavelet low-pass filtering starting from the second level. This leads to arrays of size $16 \times 64 \times 64$ (first level), $8 \times 64 \times 64$ (second level), $4 \times 32 \times 32$ (third level) and $4 \times 16 \times 64$ (fourth level). The restoration operator $\mathrm{H}_{\boldsymbol{\alpha},\boldsymbol{\beta}}$ is applied to every shearlet subband (i.e., band-pass or high-pass outputs). The scaling subband (low-pass output) is unprocessed, which is common in wavelet-based restoration methods. After processing, the transform subbands are merged again using the synthesis filters, leading eventually to the output image.

The convolutions are performed in a multi-channel way along the orientation subbands. This allows taking the inter-orientation dependencies between sparsity transform coefficients (see, e.g., Hammond and Simoncelli [16]) into account. For modeling the inter-scale dependencies between the coefficients, inspired by the zero-tree wavelet algorithm, [17] we use a recursive approach in which subbands are estimated recursively using convolutions, starting from the coarsest scale, up to the finest scale. This leads to the recurrent subnetwork depicted in Figure 2. Because subsequent scales have different subband sizes, an upscaling operation based on bilinear interpolation is performed before the summing operation takes place.

For the restoration operator $\mathrm{H}'_{\boldsymbol{\alpha}}$ we use a CNN with three convolutional layers, with a standard rectified linear unit (RELU) and one batch normalization step. The convolution (Conv) and biased convolution (Conv_Biased) layers do not include an activation function; instead the activation layers are depicted separately in Figure 3. The number of input channels is K_j (the number of orientation bands of scale j), which is multiplied by a factor C to increase the capacity of the network. In the second biased convolution step, the number of channels is again reduced to 1. Inspired by ResNet, We included one skip connection from the input to the second convolution step, to improve the training performance of the network. After the biased convolution with skip connection, a batch normalization layer is added. Similar to other work, [18] we have seen performance and training improvements by including this batch normalization layer.

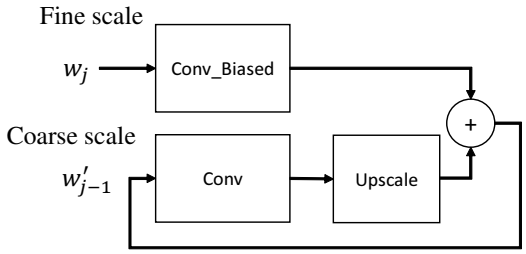


Figure 2. Shearlet interscale recurrent subnetwork.

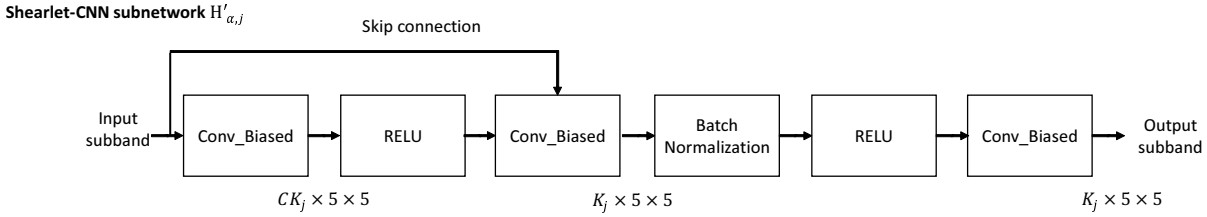


Figure 3. Hybrid shearlet-CNN subnetwork $H'_{\boldsymbol{\alpha}}$.

3. IMPLEMENTATION DETAILS

In this Section, we give some details on our implementation of the above network. Technically, the network can be implemented in existing deep learning frameworks such as TensorFlow and Pytorch, although there are a number of difficulties to overcome: 1) several layers are used which are not available by default in these frameworks and 2) the alternating optimization of Subsection 2.2 does not map in a straightforward manner onto a static computational graph. Nonstandard layers can be composed either from existing building blocks (leading possibly to a reduced execution performance) or implemented directly in C++ and/or CUDA (which requires a substantial development effort). To mitigate the second issue, when the number of iterations of (8)-(9) is fixed, the iteration loop may be unrolled entirely which in turn increases the complexity of the computational graph.

For our own implementation, we adapted the differentiable programming (DP) paradigm, in which the forward calculation of the neural network is specified entirely in a procedural programming language (i.e., with functions containing loops, calling other functions on their turn). A DP compiler then provides an algorithmic differentiation routine which derives the backpropagation function automatically at compile-time. Recently, we have extended our own programming language Quasar to implement neural networks using DP. The underlying hardware acceleration libraries are NVIDIA's cuDNN, cuFFT and cuBLAS. Whenever possible, the layer implementation is mapped onto primitive functions from these libraries, which is beneficial because most of these functions are significantly tuned to the GPU hardware to maximize performance. Through "expression rewriting", the programming language provides a mechanism to specify an implementation for a high level operation.

Additionally, for maximal performance and because the patches are relatively too small to keep the multi-processors of the GPU busy, the processing is entirely done using batches of patches. Here, a batch of patches (typically 8, 16 or 32) are processed in parallel within the same operation. For this purpose, we adopted the NCHW tensor representation, which is also used by cuDNN. In the NCHW tensor representation, four-dimensional data is processed, with dimensions batch_size $\times$ features $\times$ patch_height $\times$ patch_width. The second dimension either corresponds to feature maps (e.g., for object recognition), color channels (e.g., for color image processing), or in our case for orientation subbands.

For the inter-orientation convolution, the separate orientation bands are considered as features, so that for the finest scale and 16 orientation subbands, the convolution is performed on an array of size batch_size $\times 16 \times 64 \times 64$. Along the second dimension, essentially a 16×16 matrix multiplication is performed, leading a large number of parameters to be estimated (namely, 16^2 times the filter support size). We believe that a parameter reduction can be obtained while preserving restoration performance by imposing a certain structure on this 16×16. For example, we may assume that the matrix is band-circulant, so that only dependencies between neighboring orientations are taken into account. When

using three diagonals for example, the number of parameters per element of the filter support drops from $16^2 = 256$ to $16 + 15 + 15 = 45$. On the other hand, such structure may disregard negative correlations between a subband of angle ϑ and the subband in the orthogonal direction with angle $\vartheta + \pi/2$, which typically occur in the presence of edges. For this reason, we decided to keep the dense matrix structure.

To increase the network capacity, the second dimension of the NCHW representation is extended with C feature maps per orientation subband. This way, the network can use several filters per orientation subband. This gives patches of size batch_size $\times\, 16C \times 64 \times 64$. Alternatively, the batch processing may be performed in five dimensions (as in the NCDHW tensor format) but we opted not to do this to keep a dense interconnection between the different feature maps.

The recurrent inter-scale subnetwork from Figure 2 is traditionally implemented using unrolling (which means that the subnetwork is replicated for each scale and connected with the subnetwork of the previous iteration). This is feasible because the number of scales (four) is relatively low. In the DP approach, unrolling is not required because the compiler can deal correctly with for loops. Unrolling the subnetwork is then equivalent to loop unrolling, an optimization technique that is standard in modern compilers. The DP approach however offers the possibility to adapt the number of scales to the input data, a direction we would like to investigate in our future research.

3.1 Discrete Shearlet Transform

The discrete shearlet transform (DST) implementation was adapted from our previous work [19] to perform batch processing. Our implementation uses an FFT-based implementation of shearlets, with a specific radial decimation scheme.

Our DST implementation uses FFTs along the third and fourth dimension of the NCHW representation. This is achieved using batched FFTs (available in FFTW for CPU and cuFFT for GPU). The shearlet filters are defined in pseudo-polar frequency coordinates (ω_r, ϑ); so that each directional shearlet filter analyses a wedge-shaped region of the 2D frequency plane (see ref [19]).

The shearlet transform is implemented with the following principles in mind:

- *Lack of aliasing during decimation*: Aliasing in wavelets/shearlets causes the transform to be shift-variant, which is to be avoided when training a CNN-based network. Additionally, for many image restoration problems, aliasing-free transformations have yielded a signficantly increased restoration performance compared to their aliased counterparts. We will achieve this by using bandlimited filters.
- *Controlling the dependencies between neighboring scales and orientations* by controlling the overlap in the frequency responses of the subband filters. In particular, the inter-scale recurrent network is based on a "Markovian" assumption of the different transform scales. Therefore, we ensure that there is only frequency overlap between directly neighboring transform subbands.
- *Possibility to tie parameters across scales*: when tying parameters, the same parameters are learned for every scale, rather than allowing different scales to have different parameter values. This reduces the parameter count of the model, but requires the transform subbands of different scales to be compatible. More concretely, every scale needs to have a constant number of orientation subbands.
- *Compatibility of different orientation subbands*: the shearlet transform allows the shearlet coefficients to be decimated after a digital shearing operation, resulting in a reduction of the redundancy factor of the transform. This trick cannot be used in this work because the resulting shearlet coefficients in different orientation subbands then correspond to different positions in the image, leading again to (a form of) shift variance of the transform.

Similar to the frequency domain implementation of wavelets (or the related Steerable Pyramid transform [20]), the image is decomposed in the frequency domain into a low-pass subband and a K_j orientational high-pass subband, where K_j is an even number. Starting from the third finest scale ($j = 3$), after every low-pass filter, a decimation of a factor 2 (both horizontally and vertically) is performed. We consider two configurations:

- *No parameter tying*: the number of orientations of the shearlet transform as function of the scale by construction depends on the dilation matrix being used. To ensure sufficient frequency resolution, we deviate from the default construction and we use an isotropic scaling matrix $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ and we fix the number of orientation subbands per scale: $K_j = \max(4, 2^{1+J-j})$, where J is the number of scales of the DST.

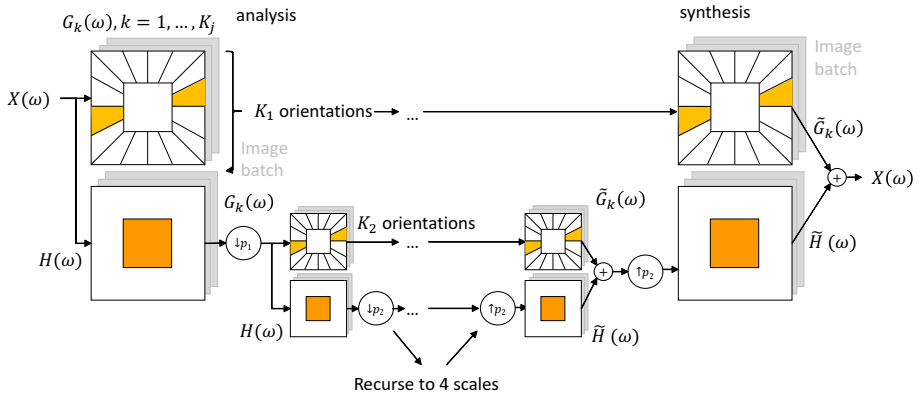


Figure 4. Illustration of the FFT-based implementation of the analysis and synthesis filter bank, for image batches.

- *Parameter tying*: when parameters are tied across scale, each scale must have the same number of orientation subbands. Therefore, we use the same isotropic scaling matrix as in the previous case, with $K_J = 8$, resulting in a directional wavelet transform with 8 orientations.

A schematic illustration of the transform is depicted in Figure 4. The discrete Fourier transform of the input image ($X(\boldsymbol{\omega})$) is subjected to a low-pass analysis filter $H(\boldsymbol{\omega})$ as well as a set of shearlet analysis filters as $G_k(\boldsymbol{\omega})$, with $k = 1,...,K_j$ the index of the orientation band. Next, the low-pass output is decimated horizontally and vertically by a factor p_j (using the Fourier domain equivalent of decimation). This process is recursive in nature, where the transform is reapplied to the decimated output. The synthesis is entirely analogous, with upsampling instead of decimation and summing of the low-pass and high-pass inputs. The scaling synthesis filters and shearlet synthesis filters are $\tilde{H}(\boldsymbol{\omega})$ and $\tilde{G}_k(\boldsymbol{\omega})$, respectively.

Following ref,[19] the analysis and scaling filters are defined in pseudo-polar coordinates:

$$H(\omega_r,\vartheta) = H_0(\omega_r), G_k(\omega_r,\vartheta) = G_0(\omega_r)\sum_{i=-\infty}^{+\infty} R\left(\frac{(\vartheta+i\pi)K_j}{\pi} - k + 1\right),$$
$$\tilde{H}(\omega_r,\vartheta) = \tilde{H}_0(\omega_r), \tilde{G}_k(\omega_r,\vartheta) = \tilde{G}_0(\omega_r)\sum_{i=-\infty}^{\infty} \tilde{R}\left(\frac{(\vartheta+i\pi)K_j}{\pi} - k + 1\right) \tag{12}$$

where $H_0(\omega_r)$ is the frequency response of a 1D scaling filter, $G_0(\omega_r)$ the frequency response of a 1D wavelet filter and $R(\vartheta)$ a real-valued compactly supported bump function. In (12), the bump function is periodized to enable the conjugate symmetry property of the discrete Fourier transform, so that the filters are real-valued in spatial domain. The bump function $R(x)$ is given by:

$$R(x) = \tilde{R}(x) = \begin{cases} 0 & x < -\frac{1+\tilde{\alpha}}{2} \\ \sin\left(\frac{\pi}{2}v\left(\frac{\tilde{\alpha}+2x+1}{2\tilde{\alpha}}\right)\right) & \left|x+\frac{1}{2}\right| \le \frac{\tilde{\alpha}}{2} \\ 1 & |x| < \frac{1-\tilde{\alpha}}{2} \\ \cos\left(\frac{\pi}{2}v\left(\frac{\tilde{\alpha}+2x-1}{2\tilde{\alpha}}\right)\right) & \left|x-\frac{1}{2}\right| \le \frac{\tilde{\alpha}}{2} \\ 0 & \text{else} \end{cases} \tag{13}$$

with $\tilde{\alpha} \in \left[0, \frac{1}{2}\right]$ a constant parameter that determines the bandwidth of the angular filters. Here, we use the interpolation function $v(x) = 3\mathrm{cl}(x)^2 - 2\mathrm{cl}(x)^3$ with $\mathrm{cl}(x) = \max(0, \min(1, x))$. To enable compactly supported (bandlimited) filters in frequency domain, we use the following modified Meyer wavelet filters for $H_0(\omega_r)$ and $G_k(\boldsymbol{\omega})$:

$$\tilde{H}_0(\omega_r) = H_0(\omega_r) = \begin{cases} 1 & |\omega_r| < \frac{\pi}{2} \\ \cos\left(\frac{\pi}{2}v\left(\frac{2|\omega|}{\pi} - 1\right)\right) & \frac{\pi}{2} \le |\omega_r| \le \pi \\ 0 & \text{else} \end{cases}, \tilde{G}_0(\omega_r) = G_0(\omega_r) = \begin{cases} 0 & |\omega_r| < \frac{\pi}{2} \\ \sin\left(\frac{\pi}{2}v\left(\frac{2|\omega|}{\pi} - 1\right)\right) & \frac{\pi}{2} \le |\omega_r| \le \pi \\ 1 & \text{else} \end{cases} \tag{14}$$

Table 1. Denoising results in PSNR [dB] for white additive Gaussian noise with variance σ^2.

σ	10				25			
Method	**Prop.**	[19]+[6]	[9]	[21]	**Prop.**	[19]+[6]	[9]	[21]
Lena	35.38	35.25	35.88	**35.89**	31.55	31.40	**32.28**	32.06
Barbara	33.49	33.85	34.07	**35.39**	28.91	29.22	29.52	**30.98**
Hill	33.15	32.87	33.59	**33.60**	29.25	29.11	**29.84**	29.79
Boats	33.30	33.08	33.85	**33.91**	29.34	29.02	**29.95**	29.85
Peppers	34.41	34.13	-	**34.97**	31.42	30.93	-	**31.94**
Man	33.57	32.96	**34.10**	33.93	29.25	28.81	**29.85**	29.59
Couple	33.34	32.94	33.89	**34.00**	29.04	28.79	**29.75**	29.71
σ	35				50			
Method	**Prop.**	[19]+[6]	[9]	[21]	**Prop.**	[19]+[6]	[9]	[21]
Lena	29.95	29.91	-	**30.48**	28.25	28.44	**29.34**	28.83
Barbara	26.74	27.50	-	**29.09**	25.05	25.86	25.37	**27.28**
Hill	28.00	27.89	-	**28.48**	26.67	26.56	**27.32**	26.96
Boats	27.83	27.61	-	**28.32**	26.24	26.13	**27.02**	26.53
Peppers	30.02	29.55	-	**30.54**	28.17	27.96	-	**28.73**
Man	27.79	27.43	-	**28.15**	26.28	26.39	**27.08**	26.56
Couple	27.43	27.33	-	**28.09**	25.91	25.86	**26.71**	26.36

Because the filters are bandlimited, we can use the following decimation factors: $p_1 = 1$, $p_j = 2$ for $j > 1$. The particular choice of the filters (13) and (14) causes only subbands which are neighbor in scale and/or orientation to have partially overlapping frequency content.

4. EXPERIMENTAL RESULTS AND DISCUSSION

In the following sections, we give experimental results for both white Gaussian noise removal (on a standard image dataset) and correlated Gaussian noise removal on electron microscopy (EM) images.

4.1 White Gaussian noise removal

First, we perform some experiments with images corrupted with white Gaussian noise, which allows our proposed technique to be compared with existing denoising methods. In paricular, we compare with the Bayes' Least Square Gaussian Scale Mixture (BLS-GSM) method from Portilla et al.[6], here applied in the shearlet transform domain, with shearlet transform calculated using the method described in ref.[19] This method is still one of the state-of-the-art wavelet-based denoising methods. We apply BLS-GSM in the shearlet domain to compare the effects of the transform domain processing rather than effects of the transform itself. We also compare with the state-of-the-art non-local denoising method BM3D from Dabov et al.[21], as well as the multi-layer neural network perception image denoising method from Burger.[9]

For the training of our hybridized neural network, we use the Dataset Berkeley Segmentation Data Set and Benchmarks (BSDS500) which consists of 500 natural images. For testing we use the SIPI USC database such that the training set distinct from the test dataset. We further made sure that we are using exactly the same images as in refs[6,9,21], e.g., by investigating the images in the software toolboxes provided by these authors. Artificial white Gaussian noise was added with standard deviations 10, 25, 35 and 50. We use 20,000 stochastic gradient iterations with batch size 32 and patch size 64×64, resulting in 640,000 patches used for training. Patches were extracted at random positions and uniformly randomly rotated 0°, 90°, 180°, 270° (a data augmentation technique).

For implementing $\mathrm{H}'_{\boldsymbol{\alpha}}$, we use 5×5 convolutions in the sparsity domain. The preprocessing operator $\mathrm{U}_{\boldsymbol{\beta}}$ is also a 5×5 convolution, learned from the images. All convolution coefficients are initialized with Dirac impulses with a relatively large amount of Gaussian noise added (with standard deviation 10). We selected the penalizer parameter value $\mu = 1/4$. During training, we used $\lambda = 2.5$ for each noise level. The inference regularization parameter λ was manually tuned for a particular input noise level σ; the default value to start the optimization was again $\lambda = 2.5$.

In Table 1, the results of the comparison are depicted. The results for the proposed method were obtained *without* parameter tying (see Section 2), our hybridized network then has 97,605 parameters. In general, the proposed method is about 0.4dB-0.5 dB behind BM3D[21] (with an exception for the Barbara image, in which the difference is often more than

Input image | Noisy image: PSNR=20.17 dB | (c) Our method: PSNR=29.35dB

[19]+[6]: PSNR=29.25 dB | [21]: PSNR=29.85dB | [9]: PSNR=29.95 dB

Figure 5. Visual denoising results for Boats with additive Gaussian noise with $\sigma = 25$.

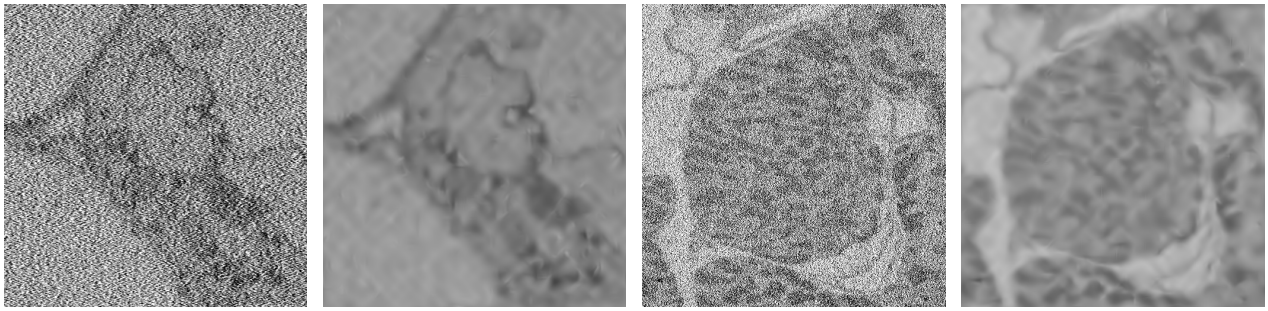
1 dB) and 0.5dB-1.0dB behind MLP[9]. In general, such result is not bad given the simplicity of the approach, the small number of CNN layers and additionally, the small number of learnable parameters (compare to over 20,000,000 parameters in MLP). The inference stage of our method is also quite fast: on a MSI GE63 laptop with NVIDIA Geforce RTX 2080 GPU, the execution time for processing 512×512 grayscale images is about 31ms on average.

Visual inspection reveals that our method does not always reconstruct the fine structures and textures, as demonstrated in Figure 5. One cause may be that the shearlet transform does not offer optimal performance for reconstructing textures (as the transform is in fact better suited for reconstructing edges and curve-like structures). Another cause may be that such structures are not sufficiently present in the BSDS500 dataset used during training. To improve the results, larger training datasets in which such details are more prominent, may be required.

We have also investigated the effect of parameter tying on the results. With parameter tying, the number of parameters is reduced to 18,329. We have found that parameter tying degrades the performance, but not severly: the PSNR is -0.18 dB worse for $\sigma = 25$ while -0.11 dB worse for $\sigma = 35$ on the images used for testing.

4.2 Correlated noise removal

In this section, we apply our method to remove correlated noise from EM images. Because no noise-free ground-truth data was available, we estimated the noise power spectral density (PSD) from a single image and used the estimated PSD to generate artificially correlated noise. The method was then trained on the BSDS500 dataset with artifically noise added, allowing us to create an indefinite number of training images. Because our method effectively learns how to reconstruct edges and textures in the sparsity domain, we believe that it does not matter too much that photographs (rather than EM images) are used for training. In Figure 6 two visual results are shown for this approach. We see that our method manages to remove the noise in the image rather well, while preserving most of the image details.



(a) (b) (c) (d)

Figure 6. Denoising results for correlated noise in electron microscopy (EM) images (a) Noisy EM image (hmg mut), (b) Denoised version of (a) using the proposed method, (c) Noisy EM image (heart), (d) Denoised version of (c).

4.3 Discussion

The proposed neural network architectures consists of two components: a shearlet transform and a relatively shallow CNN that acts in the shearlet domain. Due to the local nature of both the shearlet transform and CNN, the method will essentially learn how to estimate edges of images *locally* entirely on the shearlet subbands. The experimental in the previous subsections results demonstrate that - despite the simplicity of the approach - already high quality results can be obtained. Two factors contribute to these good results: 1) the use of a multi-channel convolution to combine information about edges and textures in multiple *orientations* of the shearlet transform and 2) the interscale recurrent network to propagate information from coarse scales to finer scales. Once the neural network parameters are learned, the inference stage is computationally very efficient. However, we note that one of the limiting factors compared to other methods (such as BM3D) is possibly the *local nature* of the estimation.

In related work, Yin et al.[22] investigated patch based CNN methods that combine local and non-local bases. The non-local bases are constructed based on non-linear dimension reductions (through spectral analysis) on the patch data, resulting in excellent results for image approximation and inpainting. Also, Labate et al.[23] presented a structured receptive field neural network with non-local component for hyperspectral image classification. An interesting avenue for future work therefore consists in incorporating a non-local model in our approach.

In general, improvements to our method can focus on two directions: 1) further reducing the number of parameters (e.g., by tying parameters also across orientations exploiting rotational symmetry) and 2) improving the performance of the method. When reducing the number of parameters even further, the method may become practical when small datasets of both corrupted images and groundtruth data are available (as in practice is often the case) and it is no longer necessary to artificially simulate degradations as we did in Subsection 4.2. On the other hand, the performance can be improved by incorporating a non-local model, but also by the use of a more sophisticated interscale recurrent network or by using mixtures of local transforms (e.g., shearlets + wavelet packet bases), to optimally reconstruct point-like structures, curved structures and textures). Moreover, using information from shearlet/wavelet scaling coefficients at each scale, in addition to the shearlet coefficients, may help the network to perform better in particular situations (e.g., textures).

5. CONCLUSION

Our hybridized neural network achieves a high denoising performance despite the relatively low computational complexity and the smaller number of learnable parameters (compared to the MLP neural network denoising method). Our approach is highly flexible in nature and allows incorporating domain-specific knowledge on the restoration task at hand, in the form of a specific image prior, or by specifying a suitable preprocessing transformation. After training, the inference method is simple and computationally very efficient. To match state-of-the-art methods in performance, we believe that several improvements can be made to our method, such as using mixtures of sparsity transforms, including non-locality in the modelling and taking advantage of the scaling coefficients at each scale.

6. ACKNOWLEDGEMENTS

We would like to thank Saskia Lippens and Christopher J. Guérin (VIB - Bio Imaging Core) for providing the electron microscopy data.

References

[1] Tomasi, C. and Manduchi, R., "Bilateral filtering for gray and color images," *International Conference on Computer Vision (ICCV)* , 839–846 (1998).

[2] He, K., Sun, J., and Tang, X., "Guided image filtering," in [*European conference on computer vision*], 1–14, Springer (2010).

[3] Donoho, D. L., "De-noising by soft-thresholding," *IEEE Trans. Inform. Theory* **41**, 613–627 (May 1995).

[4] Kingsbury, N. G., "Complex wavelets for shift invariant analysis and filtering of signals," *Journal of Applied and Computational Harmonic Analysis* **10**, 234–253 (may 2001).

[5] Şendur, L. and Selesnick, I., "Bivariate shrinkage with local variance estimation," *IEEE Signal Processing Letters* **9**, 438–441 (Dec 2002).

[6] Portilla, J., Strela, V., Wainwright, M., and Simoncelli, E., "Image denoising using scale mixtures of Gaussians in the wavelet domain," *IEEE Transactions on image processing* **12**, 1338–1351 (Nov. 2003).

[7] Starck, J. L., Candès, E. J., and Donoho, D. L., "The curvelet transform for image denoising," *IEEE Trans. on Image Process.* **11**, 670–684 (2000).

[8] Labate, D., Lim, W.-Q., Kutyniok, G., and Weiss, G., "Sparse multidimensional representation using shearlets.," in [*Proc. of SPIE Wavelets XI*], 2254–262, SPIE (2005).

[9] Burger, H. C., Schuler, C. J., and Harmeling, S., "Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds," *arXiv preprint arXiv:1211.1544* (2012).

[10] Mao, X., Shen, C., and Yang, Y.-B., "Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections," in [*Advances in Neural Information Processing Systems 29*], Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., eds., 2802–2810, Curran Associates, Inc. (2016).

[11] Jin, K. H., McCann, M. T., Froustey, E., and Unser, M., "Deep convolutional neural network for inverse problems in imaging," *IEEE Transactions on Image Processing* **26**(9), 4509–4522 (2017).

[12] Gaunt, A. L., Brockschmidt, M., Kushman, N., and Tarlow, D., "Differentiable programs with neural libraries," in [*Proceedings of the 34th International Conference on Machine Learning-Volume 70*], 1213–1222, JMLR (2017).

[13] Tian, C., Xu, Y., Fei, L., and Yan, K., "Deep learning for image denoising: a survey," in [*International Conference on Genetic and Evolutionary Computing*], 563–572, Springer (2018).

[14] Jacobsen, J.-H., van Gemert, J., Lou, Z., and Smeulders, A. W., "Structured receptive fields in CNNs," in [*Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*], 2610–2619 (2016).

[15] Kingma, D. P. and Ba, J., "Adam: A method for stochastic optimization," (2014). arxiv:1412.6980 Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[16] Hammond, D. K. and Simoncelli, E. P., "Image modeling and denoising with orientation-adapted Gaussian scale mixtures," *IEEE Transactions on Image Processing* **17**(11), 2089–2101 (2008).

[17] Shapiro, J. M., "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Proc.* **41**, 3445–3462 (dec 1993).

[18] Ioffe, S. and Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167* (2015).

[19] Goossens, B., Aelterman, J., Luong, H. Q., Pižurica, A., and Philips, W., "Efficient Design of a Low Redundant Discrete Shearlet Transform," in [*2009 International Workshop on Local and Non-Local Approximation in Image Processing (LNLA2009)*], 112–124, IEEE, Tuusula, Finland (Aug. 2009).

[20] Simoncelli, E., Freeman, W. T., Adelson, E. H., and Heeger, D. J., "Shiftable Multi-scale Transforms," *IEEE Trans. Information Theory* **38**(2), 587–607 (1992).

[21] Dabov, K., Foi, A., Katkovnik, V., and Egiazarian, K., "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Transactions on image processing* **16**(8), 2080–2095 (2007).

[22] Yin, R., Gao, T., Lu, Y. M., and Daubechies, I., "A tale of two bases: Local-nonlocal regularization on image patches with convolution framelets," *SIAM Journal on Imaging Sciences* **10**(2), 711–750 (2017).

[23] Labate, D., Safaripoorfatide, M., Karantzas, N., and Prasad, S., "Structured receptive field networks and applications to hyperspectral image classification," in [*SPIE Optics and Photonics (Wavelets and Sparsity XVIII)*], SPIE (2019).