RED-UCATION: A NOVEL CNN ARCHITECTURE BASED ON DENOISING NONLINEARITIES

Yaniv Romano[‡•] *Michael Elad*[⋆] *Peyman Milanfar*[⋆]

[‡]Department of Electrical Engineering, Technion, Haifa 32000, Israel.
[⋆]Google Research, Mountain View, California 94043, USA.

ABSTRACT

Image denoising is the most fundamental image enhancement task, and many algorithms have been proposed over the years for its solution. Interestingly, such an image denoising "engine" can be used to solve general inverse problems. Indeed, in our recent work we have presented the Regularization by Denoising (RED) framework: using a denoising engine in defining the regularization of any inverse problem. We have shown how this scheme leads to well-founded iterative algorithms in which the denoiser is applied in each iteration. In this work we describe how a learned version of RED defines a novel convolutional neural network architecture, where the commonly used point-wise nonlinearities are replaced by a denoising engine. We show how this network can be optimized end-to-end using a back-propagation that relies on guided denoising algorithms. As a case-study, we concentrate on the image deblurring problem and show the superiority of the trainable variant of RED over its analytic form.

Index Terms— Image Restoration, Deep Learning, Neural Networks, Regularization, RED - Regularization by Denoising

1. INTRODUCTION

Denoising is one of the most studied problems in image processing. It started back in the 70's with an L_2-based regularization and the Wiener filter, moved in the early 90's to robust statistics [1], PDE methods [2], and Wavelets [3], and continued in the 2000's with the introduction of spatially adaptive filtering such as the non-local means (NLM) algorithm [4]. More recent denoising algorithms include sparsity-inspired methods such as the K-SVD [5] and the BM3D [6]. The current throne holders are CNN-based methods [7, 8].

The recent progress made on the image denoising front brought researchers to believe that to a large extent, removal of additive noise from an image is a solved problem. This statement is supported by the fact that very different algorithms lead to quite similar output quality (e.g. [6–8]). This has been accompanied by a theoretical study which explored bounds on the best possible denoising [9, 10].

Armed with these insights, an exciting branch of research started, seeking ways to leverage the impressive progress in image denoising to solve other problems in image processing, and more specifically, to handle inverse problems [11, 13]. Before describing this line of work, we first define formally the restoration task that this paper aims to address. Suppose we are given a degraded image $\mathbf{y} = \mathbf{H}\mathbf{x}+\mathbf{v}$, where $\mathbf{y} \in \mathbb{R}^{N\times M}$ is the input, being a corrupted version of $\mathbf{x} \in \mathbb{R}^{N\times M}$. The matrix $\mathbf{H} \in \mathbb{R}^{NM\times NM}$ is a degradation operator (e.g. blur), and $\mathbf{v} \in \mathbb{R}^{N\times M}$ is white Gaussian noise of known standard-deviation σ. The restoration task is all about recovering the unknown $\mathbf{x}$ given $\mathbf{y}$. Notice that in the case of denoising $\mathbf{H}$ is the identity matrix, thereby considered as the simplest inverse problem. There is a long list of beautiful ideas for tackling this restoration problem, and many of these turn this task into a minimization of an energy function of the form:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda R(\mathbf{x}),$$

where $R(\mathbf{x})$ stands for a *regularizer* or *prior*.

We return to our question: How can we leverage an existing denoising algorithm to handle the above-described restoration task? The Plug-and-Play-Priors (P^3) approach [11] addresses this question by relying on the idea of variable splitting, being the core mechanism of the ADMM [12].

The Regularization by Denoising (RED) framework [13] tackles the same task much more broadly by using denoisers for handling general inverse problems. RED adopts a different and theoretically better founded approach than the P^3. In essence, RED uses a denoiser to define the regularizer $R(\mathbf{x})$, as follows:

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\mathbf{x}^T(\mathbf{x} - f(\mathbf{x})). \tag{1}$$

The function $f(\cdot)$ is an arbitrary denoiser, and the prior $R(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T(\mathbf{x} - f(\mathbf{x}))$ is an image-adaptive Laplacian regularizer [14–16]. This choice favors either a small inner product between $\mathbf{x}$ and the residual $(\mathbf{x} - f(\mathbf{x}))$, or a small residual

• The research leading to these results has received funding in part from the European Research Council under EUs 7th Framework Program, ERC under Grant 320649, and in part by Israel Science Foundation (ISF) grant no. 1770/14.

image. How can we minimize RED's objective? Under mild assumptions it was shown in [13] that $\nabla_{\mathbf{x}} R(\mathbf{x}) = \mathbf{x} - f(\mathbf{x})$. Furthermore, if the spectral radius of $\nabla_{\mathbf{x}} f(\mathbf{x})$ is smaller or equal to 1 then the objective of Eq. (1) is *convex*. Thus, since we have access to the gradient of this convex objective, it can be minimized via any convex optimization technique [17], while leading to the globally optimal solution.

Embarking from Eq. (1), the work reported in [13] suggested three iterative optimization schemes to minimize it: the Steepest Descent, an ADMM-based method, and a Fixed-Point (FP)-based algorithm. In this paper we concentrate on the latter, and show that it puts forward a novel deep convolutional neural-network (CNN) architecture [18, 19], in which the simple point-wise nonlinearity (e.g. ReLU [20]) is replaced by an arbitrary image denoising algorithm. Using this new architecture, we derive a trainable version of RED – an end-to-end learning paradigm for solving any restoration task, where the training is done by a back-propagation algorithm [18]. A key ingredient in this process is propagating reversely through the denoising algorithm, which is handled using the concept of guided filtering. This process puts forward a trained Fixed-Point-like scheme that achieves better reconstruction error with significantly fewer iterations compared to its origin – the analytic RED-based scheme.

The flavor of our work is reminiscent of the migration from the Iterative Soft Thresholing Algorithm (ISTA) to its learned version Learned-ISTA (LISTA). In [21] the authors showed that a (recurrent) neural-network architecture can be designed and trained in order to imitate and improve ISTA. Interestingly, they reported that the learning approach significantly reduces the number of iterations required to achieve a comparable solution to ISTA – a similar phenomenon to the one presented in our work.

2. RED: THE FIXED-POINT STRATEGY

We consider the problem defined by the energy function

$$E(\mathbf{x}) = \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\mathbf{x}^T(\mathbf{x} - f(\mathbf{x})),$$

and we use the FP strategy to minimize the above penalty. Under RED's assumptions, the gradient of $E(\mathbf{x})$ is given by

$$\nabla_{\mathbf{x}} E(\mathbf{x}) = \mathbf{H}^T(\mathbf{H}\mathbf{x} - \mathbf{y}) + \lambda(\mathbf{x} - f(\mathbf{x})).$$

The FP strategy puts indices to the appearances of $\mathbf{x}$ in the equation $\nabla_{\mathbf{x}} E(\mathbf{x}) = 0$, leading to the update rule of $\hat{\mathbf{x}}_{k+1}$:

$$\begin{aligned} 0 &= \mathbf{H}^T(\mathbf{H}\hat{\mathbf{x}}_{k+1} - \mathbf{y}) + \lambda(\hat{\mathbf{x}}_{k+1} - f(\hat{\mathbf{x}}_k)) \\ \rightarrow \hat{\mathbf{x}}_{k+1} &= \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1}\left(\mathbf{H}^T\mathbf{y} + \lambda f(\hat{\mathbf{x}}_k)\right). \end{aligned} \quad (2)$$

This algorithm is typically initialized by $\hat{\mathbf{x}}_0 = \mathbf{y}$.

3. MOVING TO A LEARNING PARADIGM

We start by proposing a generalization of the above FP iterative formula, incorporating the over-relaxation idea [12] as follows:

$$0 = \mathbf{H}^T(\mathbf{H}\hat{\mathbf{x}}_{k+1} - \mathbf{y}) + \lambda(\alpha\hat{\mathbf{x}}_{k+1} + (1-\alpha)\hat{\mathbf{x}}_k - f(\hat{\mathbf{x}}_k)),$$

where the choice of $\alpha = 1$ reduces the above to the original FP method. By rearranging this equation we get

$$\hat{\mathbf{x}}_{k+1} = \left(\mathbf{H}^T\mathbf{H} + \lambda\alpha\mathbf{I}\right)^{-1}\left(\mathbf{H}^T\mathbf{y} - \lambda(1-\alpha)\hat{\mathbf{x}}_k - \lambda f(\hat{\mathbf{x}}_k)\right).$$

The above can be separated into three terms, as follows:

$$\hat{\mathbf{x}}_{k+1} = \mathbf{Q}\mathbf{H}^T\mathbf{y} - \lambda(1-\alpha)\mathbf{Q}\hat{\mathbf{x}}_k - \lambda\mathbf{Q}f(\hat{\mathbf{x}}_k),$$

where $\mathbf{Q} = \left(\mathbf{H}^T\mathbf{H} + \lambda\alpha\mathbf{I}\right)^{-1}$. The lesson we take is that updating the solution $\hat{\mathbf{x}}_{k+1}$ could be done more generally by a system of the form

$$\hat{\mathbf{x}}_{k+1} = \mathbf{W}_1\mathbf{y} + \mathbf{W}_2\hat{\mathbf{x}}_k + \mathbf{W}_3 f(\hat{\mathbf{x}}_k),$$

where $\mathbf{W}_1, \mathbf{W}_2$, and $\mathbf{W}_3$ are convolution filters. An appealing extension of this equation can be to train different such filters $\mathbf{W}_j^k$ per each layer, leading to

$$\hat{\mathbf{x}}_k = \mathbf{W}_1^k\mathbf{y} + \mathbf{W}_2^k\hat{\mathbf{x}}_{k-1} + \mathbf{W}_3^k f(\hat{\mathbf{x}}_{k-1}). \quad (3)$$

The chain of these iterations offers a feed-forward architecture for recovering $\mathbf{x}$, in which each layer applies 3 convolutions and a very special nonlinearity – a denoising algorithm.

Given a set of training images $\{\mathbf{x}^i\}_{i=1}^P$ and their corresponding degraded versions $\{\mathbf{y}^i\}_{i=1}^P$, we would like to design filters $\mathbf{W}_j^k$ so as to extract the best overall performance. For a *single-layer* algorithm that is initialized by $\hat{\mathbf{x}}_0^i$ (e.g. $\hat{\mathbf{x}}_0^i = \mathbf{y}^i$), this implies that we minimize the L_2 recovery error

$$\frac{1}{2}\sum_{i=1}^{P}\|(\mathbf{W}_1^1\mathbf{y}^i + \mathbf{W}_2^1\hat{\mathbf{x}}_0^i + \mathbf{W}_3^1 f(\hat{\mathbf{x}}_0^i)) - \mathbf{x}^i\|_2^2,$$

and this is a Least-Squares problem over the three unknown filters. A loss for a multi-layer architecture of depth K can be also defined; it is initialized similarly and reads as follows

$$\ell = \frac{1}{2}\sum_{i=1}^{P}\|\hat{\mathbf{x}}_K^i - \mathbf{x}^i\|_2^2, \quad (4)$$

where $\hat{\mathbf{x}}_K$ is the output image. This obtained architecture is of interest because it is originated from a clear and well-understood optimization scheme, being born from the fixed-point algorithm referring to the RED formulation [13]. As mentioned above, this scheme resembles a CNN with skip connections [22] if we choose to train different filters in each layer, or to recurrent neural network (RNN) [19] if the

weights are shared. The core difference from standard networks is the non-linearity – while the common choice in CNN is the ReLU element-wise function [20], our architecture uses an image denoiser, which is believed to be tuned much better to image content. Due to this change, a shallow network of our algorithm could replace a deeper version of regular CNN.

Comparing this approach to a direct use of the iterations in Eq. (2), the hope is that the learned method would be much more efficient, as it optimizes the filters (possibly chosen to be different per layer), and targets the reconstruction error with respect to the ground-truth images. Whereas the original FP algorithm is applicable to any content in $\mathbf{y}$ (and $\mathbf{x}$), our scheme is specifically tailored to serve the family of images we are training on. Thus, this approach is likely to lead to much better performance if this family of images is compact.

A challenge that we found when defining this architecture touches on the special structure of the filters involved. Specifically, one can assume that $\mathbf{W}_k^j$ are of convolutional form, and they have a relatively small support which facilitates learning. However, this puts a limit on the achievable effect of the proposed algorithm in Eq. (3), since the original iteration in FP scheme uses much wider filters due to the inversion $\left(\mathbf{H}^T\mathbf{H}+\lambda\alpha\mathbf{I}\right)^{-1}$. How could we overcome this gap?

The answer we propose is working in the transform domain. Recall that each filtering operation of the form $\mathbf{Wy}$ could be written alternatively as $\mathbf{T}^H\mathbf{TWT}^H\mathbf{Ty}$, where $\mathbf{T}$ is any unitary transform. Choosing $\mathbf{T}$ to diagonalize $\mathbf{W}$, i.e. $\mathbf{TWT}^H = \Lambda$ where Λ is a diagonal matrix, the above can be written as $\mathbf{Wy} = \mathbf{T}^H\mathbf{TWT}^H\mathbf{Ty} = \mathbf{T}^H\Lambda\mathbf{Ty}$. In words, this expression applies a transform on the incoming image ($\mathbf{z} \rightarrow \mathbf{Tz}$), multiplies each 'frequency' bin by the proper scaler weight $\Lambda_{i,i}$, and then transforms the result back to the pixel-domain. Using this rationale, we revisit Eq. (3) and get

$$\hat{\mathbf{x}}_K = \mathbf{T}^H\Lambda_1^K\mathbf{Ty} + \mathbf{T}^H\Lambda_2^K\mathbf{T}\hat{\mathbf{x}}_{K-1} + \mathbf{T}^H\Lambda_3^K\mathbf{T}f(\hat{\mathbf{x}}_{K-1}).$$

Leveraging the diagonal structure of Λ_j^K, the above equation can be written as follows:

$$\hat{\mathbf{x}}_K = \mathbf{M}_1^K\mathbf{z}_1^K + \mathbf{M}_2^K\mathbf{z}_2^K + \mathbf{M}_3^K\mathbf{z}_3^K, \tag{5}$$

where the column vectors $\mathbf{z}_1^K$, $\mathbf{z}_2^K$ and $\mathbf{z}_3^K$ are composed of the diagonal entries of Λ_1^K, Λ_2^K and Λ_3^K, respectively. The matrices $\mathbf{M}_j^K$ are defined as $\mathbf{M}_1^K = \mathbf{T}^H diag\{\mathbf{Ty}\}$, $\mathbf{M}_2^K = \mathbf{T}^H diag\{\mathbf{T}\hat{\mathbf{x}}_{K-1}\}$, and $\mathbf{M}_3^K = \mathbf{T}^H diag\{\mathbf{T}f(\hat{\mathbf{x}}_{K-1})\}$. Armed Eq. (5) and by relying on the unitary property of $\mathbf{T}$, we rewrite the training error in Eq. (4) and get

$$\ell = \frac{1}{2}\sum_{i=1}^{P}\|\mathbf{T}\hat{\mathbf{x}}_K^i - \mathbf{Tx}^i\|_2^2. \tag{6}$$

As such, the implication of the transition to the transform domain is that we take the images $\mathbf{x}^i, \mathbf{y}^i, \hat{\mathbf{x}}_{K-1}^i, f(\hat{\mathbf{x}}_{K-1}^i)$ and apply a transform on them, and then work on each bin separately, optimizing 3 scalars (or 6 in case of a complex transform) per each such frequency bin. We choose the Fourier transform as it diagonalizes the linear space invariant convolutions, with the cost of dealing with complex numbers.

3.1. Back-Propagation Rules for RED

In the process of optimizing the multi-layered filters of the proposed architecture, we employ the back-propagation (BP) algorithm. This use of BP is conventional apart from one major difference: we encounter the need to pass through the gradients of the denoisers in each layer. If the denoiser is given as an independent CNN (e.g. [7,8]), one can simply consider the entire path as a large CNN and optimize it end-to-end, while keeping the parameters of the denoiser intact. However, we would like to use available general purpose denoisers, and the question is how to handle the back-propagation in this case.

In order to answer this question, we present a much simpler system which exposes both the difficulty we describe and the remedy to it. Consider the following problem

$$\ell^s = \frac{1}{2}\|\mathbf{M}_b f(\mathbf{M}_a\mathbf{z}) - \boldsymbol{\gamma}\|_2^2 \tag{7}$$

where, just as in Eq. (5), $\mathbf{z}$ is a set of filter coefficients in the transform domain, $\mathbf{M}_a$ represents the combination of the image to be filtered along with the inverse transform to the pixel-domain. $\mathbf{M}_b$ is yet another linear operator of general form. The gradient of the above loss w.r.t. $\mathbf{z}$ is given by

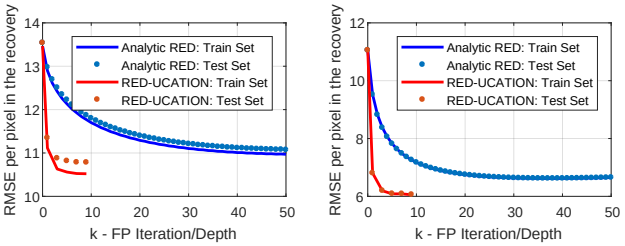
$$\nabla_{\mathbf{z}}\ell^s = \mathbf{M}_a^H\nabla_{\mathbf{x}}^T f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{M}_a\mathbf{z}}\mathbf{M}_b^H(\mathbf{M}_b f(\mathbf{M}_a\mathbf{z}) - \boldsymbol{\gamma}) \tag{8}$$

We now rely on two critical observations:

1. In many cases, the Jacobian of the denoiser $f(\cdot)$ is symmetric or can be made so [23]. This is the case with the K-SVD [5] and the NLM filter version that is used in this work [24]. Thus, the above transpose operation can be discarded.

2. Many of the denoisers we operate with admit a pseudo-linear form $f(\mathbf{x}) = \mathbf{A}\{\mathbf{x}\}\mathbf{x}$, in which the matrix $\mathbf{A}\{\mathbf{x}\}$ absorbs all the non-linear decisions, and then applies a linear filtering $\mathbf{A}\{\mathbf{x}\}$ on $\mathbf{x}$. Thus, in the above equation, the term $\nabla_{\mathbf{x}}f(\mathbf{x})\Big|_{\mathbf{x}=\mathbf{M}_a\mathbf{z}}$ is essentially $\mathbf{A}\{\mathbf{M}_a\mathbf{z}\}$ [13]. Its multiplication by the vector $\mathbf{M}_b^H(\mathbf{M}_b f(\mathbf{M}_a\mathbf{z}) - \boldsymbol{\gamma})$ is nothing but a denoiser activation on this image, *but this denoiser draws its non-linearities from the image $\boldsymbol{M}_a\boldsymbol{z}$ on which it has been built*. This is known as "guided filtering" and it is the strategy we employ in back-propagating the derivatives through the architecture.

As said above, the overall optimization of our chain of filters follows the usual back-propagation, and the above is plugged to manage the activation of the denoiser within this chain rule. This way, one can compute $\partial\ell/\partial\Lambda_j^k$ (or $\partial\ell/\partial\mathbf{z}_j^k$), for all j and k. These gradients allow us to update the parameters of the network, which can be done via the stochastic gradient descent (SGD) algorithm.



(a) Image-Net (b) Facial Images

Fig. 1. Comparison between the average RMSE of the analytic algorithm (blue curves) and the trained one (red curves).

4. EXPERIMENTAL RESULTS

We tested the RED-UCATION paradigm on two different datasets: Image-Net [25] and Chinese Facial images [26]. In both cases we tackled the image deblurring problem as a way to compare the analytic FP algorithm to its trainable variant.

The learning of the network's parameters was done using SGD, and common deep-learning ideas were borrowed to improve the optimization. Specifically, we ran the training for several epochs, divided the training set into batches (of size 10 images), and used ADAM [27] method (with default parameters) to improve the stochastic optimization. We chose the symmetric NLM [24] to be our non-linear function, enabling to compute efficiently its derivative. Notice that this denoiser has an inherent hyper-parameter – the input-noise level, which was simply set to be 20 in all the experiments.

The parameters of the analytic FP algorithm were tuned to lead to the best performance, where we used the symmetric NLM [24] as the denoiser, just as described above.

4.1. Image-Net

We constructed a database of natural images by randomly picking 6000 images from the validation set of Image-Net [25], converting these to gray-scale (1 channel) and cropped the resulting images to a fixed dimensions of 64×64. Next, we divided the obtained images into a training set of 5000 examples and the remaining 1000 images served as our test set. We then degraded the above $\{\mathbf{x}^i\}_i$ images by convolving these with a 2D Gaussian blur kernel with a standard-deviation of 1, and further contaminated the outcomes with a white Gaussian noise of $\sigma = 4$. This process results in the corrupted versions $\{\mathbf{y}^i\}_i$ of the original images $\{\mathbf{x}^i\}_i$.

Fig. 1(a) presents the root mean squared error[1] (RMSE), averaged over the train or test images, of the analytic FP algorithm (in blue) as a function of the iterations. As can be seen, the error decreases as the algorithm evolves until it tends to a plateau, getting closer to the convergence of the FP. Clearly, 50 iterations are not enough, and more iterations are required to achieve this convergence. We should note that in this part of our experiment, there is no conceptual difference between the errors for the train and the test - the same method applies to both, simply changing the corpus we evaluate the error on.

[1]This measure is defined as $\text{RMSE}(\mathbf{z}, \mathbf{x}) = \frac{1}{\sqrt{MN}} \|\mathbf{z} - \mathbf{x}\|_2$.

(a) Original (b) Input (12.07) (c) FP (6.76)

(d) $K = 1$ (7.11) (e) $K = 3$ (6.41) (f) $K = 9$ (6.15)

Fig. 2. Comparison between the restoration of a face image (a,b) via the analytic algorithm (c) and its trainable variants (d,e,f). The numbers in the parenthesis refer to the RMSE.

Turning to the red-colored curves in the same figure, these correspond to the trainable version of the FP algorithm, using a different trained network of varying depth (1, 3, 5, 7, and 9). One can see the superiority of the proposed approach. We achieved a gain in performance, both in terms of the number of iterations (which is analogous to the depth of the network) and the resulting average RMSE. The generalization results on the test data are also very encouraging.

4.2. Facial Images

Differently from the above experiment, here we tested our method on a dataset of facial images [26] of size 64×64. We randomly chose 4000 images for training and 900 for testing.

Similarly to the previous experiment, Fig. 1(b) compares the average RMSE of the analytic FP algorithm and its trainable version. As can be seen, the learning approach outperforms the analytic one. A visual demonstration of the resulting images is provided in Fig. 2. Notice how the depth improves the quality of the restored image, corroborating the numerical RMSE score, reported in Fig. 1.

5. CONCLUSIONS

In this paper we presented RED-UCATION: A trainable version of the Regularization by Denoising framework. This formulates a novel CNN architecture that originates from the RED's fixed-point optimization scheme, suggesting to replace the simple point-wise nonlinearity used in CNN (e.g. ReLU) with a sophisticated denoiser. We formulated the back-propagation rules for the training of this new architecture, leading to an end-to-end learning scheme. Lastly, we demonstrated the effectiveness of RED-UCATION by tackling the image deblurring problem, where the obtained results clearly show that the learned variant of RED is indeed superior to its analytic origin.

6. REFERENCES

[1] Huber, P.J., Robust Statistical Procedures, *SIAM*, 1996.

[2] T. F. Chan, and J. Shen, Image processing and analysis: variational, PDE, wavelet, and stochastic methods, *SIAM*, 2005.

[3] S Mallat, A Wavelet Tour of Signal Processing, *Academic Press*, 1999.

[4] A. Buades, B. Coll, and J.M. Morel, A Review of Image Denoising Algorithms, With a New One. *Multiscale Modeling & Simulation*, Vol. 4, No. 2, pp.490–530, 2005.

[5] M. Elad and M. Aharon, Image Denoising via Sparse and Redundant Representations Over Learned Dictionaries, *IEEE Trans. on Image Processing*, Vol. 15, No. 12, pp. 3736–3745, 2006.

[6] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering, *IEEE Trans. on Image Processing*, Vol. 16, No. 8, pp. 2080–2095, 2007.

[7] Y. Chen and T. Pock, Trainable Nonlinear Reaction Diffusion: A Flexible Framework for Fast and Effective Image Restoration, *IEEE PAMI*, Vol. 39, No. 6, 2017.

[8] T. Remez, O. Litany, R. Giryes, and A.M. Bronstein, Deep Class Aware Denoising, *arXiv*, 1701.01698, 2017.

[9] P. Chatterjee and P. Milanfar, Is Denoising Dead? *IEEE Trans. on Image Processing*, Vol. 19, No. 4, pp. 895–911, 2010.

[10] A. Levin, B. Nadler, F. Durand, and W.T. Freeman, Patch Complexity, Finite Pixel Correlations and Optimal Denoising, *ECCV*, 2012.

[11] S.V. Venkatakrishnan, C.A. Bouman, and B. Wohlberg, Plug-and-Play Priors for Model Based Reconstruction, *GlobalSIP*, 2013.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers, *Foundations and Trends in Machine Learning*, Vol. 3, No. 1, pp. 1–122, July 2011.

[13] Y. Romano, M. Elad, and P. Milanfar, The Little Engine that Could: Regularization by Denoising (RED), *SIAM J. on Imaging Sciences*, Vol. 10, No. 4, pp. 1804-1844, 2017.

[14] P. Milanfar, A Tour of Modern Image Filtering, *IEEE Signal Processing Magazine*, Vol. 30, No. 1, pp. 106–128, 2013.

[15] Y. Romano and M. Elad, Boosting of Image Denoising Algorithms, *SIAM Journal on Imaging Sciences*, Vol. 8, No. 2, pp. 1187–1219, June 2015.

[16] A Kheradmand and P. Milanfar, A General Framework for Regularized, Similarity-based Image Restoration, *IEEE Trans. on Image Processing*, Vol. 23, No. 12, pp. 5136–5151, Dec. 2014.

[17] S. Boyd and L. Vandenberghe, Convex Optimization, *Cambridge University Press*, 2004.

[18] Y. LeCun, Y. Bengio, and G. Hinton, Deep Learning, *Nature*, Vol. 521, No. 7553, pp.436–444, 2015.

[19] Y. Bengio, I.J. Goodfellow, and A. Courville, Deep learning, *MIT press*, 2016.

[20] X. Glorot, A. Bordes, and Y. Bengio, Deep Sparse Rectifier Neural Networks, in *Aistats*, vol. 15, p. 275, 2011.

[21] K. Gregor, and Y. LeCun, Learning fast approximations of sparse coding, In *ICML*, pp. 399–406, 2010.

[22] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, *IEEE CVPR*, pp. 770-778, 2016.

[23] P. Milanfar, Symmetrizing Smoothing Filters, SIAM Journal on Imaging Sciences, Vol. 6, No. 1, pp. 263284, 2013.

[24] P. Milanfar, and H. Talebi, A New Class of Image Filters Without Normalization. *IEEE ICIP*, pp. 3294–3298, 2016.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, *IJCV*, Vol. 115, No. 3, pp. 211–252, 2015.

[26] O. Bryt, and M. Elad, Compression of facial images using the K-SVD algorithm, *Journal of Visual Communication and Image Representation*, Vol. 19, No. 4, pp. 270-282, 2008.

[27] D. Kingma, and J. Ba, Adam: A Method For Stochastic Optimization, *arXiv*, 1412.6980, 2014.