LIDIA: Lightweight Learned Image Denoising with Instance Adaptation

Gregory Vaksman
CS Department - The Technion
grishav@campus.technion.ac.il

Michael Elad
Google Research
melad@google.com

Peyman Milanfar
Google Research
milanfar@google.com

Abstract

Image denoising is a well studied problem with an extensive activity that has spread over several decades. Leading classical denoising methods are typically designed to exploit the inner structure in images by modeling local overlapping patches, and operating in an unsupervised fashion. In contrast, newcomers to this arena are supervised and universal neural-network-based methods that bypass this modeling altogether, targeting the inference goal directly and globally, tending to be deep and parameter heavy.

This work proposes a novel lightweight learnable architecture for image denoising, using a combination of supervised and unsupervised training of it, the first aiming for a universal denoiser and the second for an instance adaptation. Our architecture embeds in it concepts taken from classical methods, leveraging patch processing, non-local self-similarity, representation sparsity and a multiscale treatment. Our proposed universal denoiser achieves near state-of-the-art results, while using a small fraction of the typical number of parameters. In addition, we introduce and demonstrate two highly effective ways for further boosting the denoising performance, by adapting this universal network to the input image. The code reproducing the results of this paper is available at https://github.com/grishavak/LIDIA-denoiser.

1. Introduction

Image denoising is a well studied problem, and many successful algorithms have been developed for handling this task over the years, e.g. NLM [2], KSVD [11], BM3D [8], EPLL [52], WNMM [12] and others [36, 24, 47, 9, 32, 28, 15, 35, 43, 45, 40, 29]. These classically-oriented algorithms strongly rely on models that exploit properties of natural images, usually employed while operating on small fully overlapped patches. For example, both EPLL [52] and PLE [47] perform denoising using Gaussian Mixture Modeling (GMM) imposed on the image patches. The K-SVD algorithm [11] restores images using a sparse modeling of such patches. BM3D [8] exploits self-similarity by grouping similar patches to 3D blocks and filtering them jointly. The algorithms reported in [40, 29] harness a multi-scale analysis framework on top of the above-mentioned local models. Common to all these and many other classical denoising methods is the fact that they operate in an unsupervised fashion, adapting their treatment to each image.

Recently, supervised deep-learning based methods entered the denoising arena, showing state-of-the-art (SOTA) results in various contexts [3, 6, 44, 48, 25, 49, 34, 41, 23, 50, 22]. In contrast to the above-mentioned classical algorithms, deep-learning based methods tend to bypass the need for an explicit modeling of image redundancies, operating instead by directly learning the inference from the incoming images to their desired outputs. In order to obtain a non-local flavor in their treatment, as self-similarity or multi-scale methods would do, most of these algorithms ([22] being an exception) tend to increase their footprint by utilizing very deep and parameter heavy networks. These
reflect badly on their memory consumption, the required amount of training images and the time for training and inference. Note that most deep-learning based denoisers operate in an universal fashion, i.e., they apply the same trained network to all incoming images.

An interesting recent line of work by Lefkimmiatis proposes a denoising network with a significantly reduced number of parameters, while persisting on near SOTA performance \cite{18, 19}. This method leverages the non-local self-similarity property of images by jointly operating on groups of similar patches. The network’s architecture consists of several repeated stages, each resembling a single step of the proximal gradient descent method under sparse modeling \cite{30}. In comparison with DnCNN \cite{48}, the work reported in \cite{18, 19} shows a reduction by factor of 14 in the number of parameters, while achieving denoising PSNR that is only $\sim 0.2$ dB lower.

In this paper we propose two threads of novelty. Our first contribution aims at a better design of a denoising network, inspired by Lefkimmiatis’ work. This network is to be trained in a supervised fashion, creating an effective universal denoiser for all images. Our second contribution extends the above by introducing image adaptation: We offer ways for updating the above network for each incoming image, so as to accommodate better its content and inner structure, leading to better denoising.

Referring to the first part of this work (our universal denoiser), we continue with Lefkimmiatis’ line of lightweight networks and propose a novel, easy, and learnable architecture that harnesses several main concepts from classical methods: (i) Operating on small fully overlapping patches; (ii) Exploiting non-local self-similarity; (iii) Leveraging representation sparsity; and (iv) Employing a multi-scale treatment. Our network resembles the one proposed in \cite{18, 19}, with several important differences:

1. We introduce a multi-scale treatment that combats spatial artifacts, especially noticeable in smooth regions \cite{40}. While this change does not reflect strongly on the PSNR results, it has a clear visual contribution;

2. Our network is more effective by operating in the residual domain, similar to the approach taken by \cite{48};

3. Our patch fusion operator includes a spatial smoothing, adding an extra force to our overall filtering; and

4. Our architecture is trained end-to-end, whereas Lefkimmiatis’s scheme consists of a greedy training of its layers, followed by an end-to-end update.

Our proposed method operates on all the overlapping patches taken from the processed image by augmenting each with its nearest neighbors and filtering these jointly. The patch grouping stage is applied only once before any filtering, and it is not a part of the learnable architecture. Each patch group undergoes a series of trainable steps that aim to predict the noise in the candidate patch, thereby operating in the residual domain. As mentioned above, our scheme includes a multi-scale treatment, fusing the processing of corresponding patches from different scales.

Moving to the second novelty in this work (image adaptation), we present two ways for updating the universal network for any incoming image, so as to further improve its denoised result. This part relies on three key observations: (i) A universally trained denoiser is necessarily less effective when handling images falling outside the training-set statistics; (ii) While our universal denoiser does exploit self-similarity, this can be further boosted for images with pronounced repetitions; and (iii) As our universal denoiser is lightweight, even a single image example can be used for updating it without overfitting.

In accordance with these, we propose to retrain the universal network and better tune it for the image being served. One option, the external boosting, suggests taking the universally denoised result, and using it for searching the web for similar photos. Taking even one such related image and performing few epochs of update to the network may improve the overall performance. This approach is extremely successful for images that deviate from the training-set, as illustrated in Figure 1. The second adaptation technique, the internal one, re-trains the network on the denoised image itself (and a noisy version of it). This is found to be quite effective for images with marked inner-similarities.

To summarize, this paper has two key contributions:

1. We propose a novel architecture that is inspired by classical denoising algorithms. Employed as a universal denoiser and trained in a supervised fashion, our network gives near-SOTA results while using a very small number of parameters to be tuned \cite{2}.

2. We present an image-adaptation option, in which the above network is updated for better treating the incoming image. This adaptation becomes highly effective in cases of images deviating from the natural image statistics, or in situations in which the incoming image exhibits stronger inner-structure. In these cases, denoising results are boosted dramatically, surpassing known supervised deep-denoisers.

1A recently published work \cite{4, 5} proposes an interesting special architecture constraint that allows for training the network on the corrupted image itself in a fashion similar to our adaptation. We shall return to these papers in the results’ section and explain more on the relation to our work.

2Reducing the number of network parameters is important for low GPU memory consumption and for reducing the number of images required for training. Indeed, our proposed adaptation relies on this feature for avoiding overfitting. We should note, however, that while the number of parameters in our proposed network is relatively small, our inference computational load is similar to DnCNNs.
2. The Proposed Universal Network

2.1. Overall Algorithm Overview

Our proposed method extracts all possible overlapping patches of size $\sqrt{n} \times \sqrt{n}$ from the processed image, and cleans each in a similar way. The final reconstructed image is obtained by combining these restored patches via averaging. The algorithm is shown schematically in Figure 2.

In order to formulate the patch extraction, combination and filtering operations, we introduce some notations. Assume that the processed image is of size $M \times N$. We denote by $Y, \hat{Y} \in \mathbb{R}^{MN}$ the noisy and denoised images respectively, both reshaped to a 1D vector. Similarly, the corrupted and restored patches in location $i$ are denoted by $z_i, \hat{z}_i \in \mathbb{R}^n$ respectively, where $i = 1, \ldots, MN$. Patch extraction from the noisy image is denoted by $z_i = R_i Y$, and the denoised image is obtained by combining the denoised patches $\{\hat{z}_i\}$ using weighted averaging,

$$\hat{Y} = \left( \sum_i w_i R_i^T R_i \right)^{-1} \sum_i w_i R_i^T \hat{z}_i,$$

where these weights are $w_i = \exp\{ -\beta \cdot \text{var}(z_i) \}$, (\text{var} (z) is a sample variance of z, and $\beta$ is learned).

2.2. Our Scheme: A Closer Look

Zooming in on the local treatment, it starts by augmenting the patch $z_i$ with a group of its $k-1$ nearest neighbors, forming a matrix $Z_i$ of size $n \times k$. The nearest neighbor search is done using a Euclidean metric, limited to a search window of size $b \times b$ around the center of the patch.

The matrix $Z_i$ undergoes a series of trainable operations that aim to recover a clean candidate patch $\hat{z}_i$. Our filtering network consists of several blocks, each consisting of (i) a forward 2D linear transform; (ii) a non-negative thresholding (ReLU) on the obtained features; and (iii) a transform back to the image domain. All transforms operate separately in the spatial and the similarity domains. In contrast to BM3D and other methods, our filtering predicts the residual rather than the clean patches, just as done in [48].

Our scheme includes a multi-scale treatment and a fusion of corresponding patches from the different scales. The adopted strategy borrows its rationale from [46], which studied the single image super resolution task. In their algorithm, high-resolution patches and their corresponding low-resolution versions are jointly treated by assuming that they share the same sparse representation. The two resolution patches are handled by learning a pair of coupled dictionaries. In a similar fashion, we augment the corresponding patches from the two scales, and learn a joint transform for their fusion. In our experiments, the multi-scale scheme includes only two scales, but the same concept can be applied to a higher pyramid. In our notations, the 1st scale is the original noisy image $Y$, and the 2nd scale images are created by convolving $Y$ with the low-pass filter $f_{LP} = \frac{1}{16} [1 \ 2 \ 1]^T \cdot [1 \ 2 \ 1]$ and down-sampling the result by a factor of two. In order to synchronize between patch locations in the two scales, we create four downsampled images by sampling the convolved image at either even or odd locations: $Y_{oe(2)}, Y_{eo(2)}, Y_{oc(2)}, Y_{ec(2)}$ (even/odd columns & even/odd rows). For each 1st scale patch, the corresponding 2nd scale patch (of the same size $\sqrt{n} \times \sqrt{n}$) is extracted from the appropriate down-sampled image, such that both patches are centred at the same pixel in the original image.

We denote the 2nd scale patch that corresponds to $z_i$ by $z_i^{(2)} \in \mathbb{R}^n$. This patch is augmented with a group of its $k-1$ nearest neighbors, forming a matrix $Z_i^{(2)}$ of size $n \times k$. The nearest neighbor search is performed in the same down-scaled image from which $z_i^{(2)}$ is taken. Both matrices, $\{Z_i\}$ and $\{Z_i^{(2)}\}$, are fed to the filtering network, which fuses the scales using a joint transform, as described next.

2.3. The Filtering Network

A basic component within our network is the TRT (Transform–ReLU–Transform) block, which follows the classic Thresholding algorithm in sparse approximation theory [10]. The same conceptual structure is employed by the well-known BM3D [8]. The TRT block applies a learned transform, non-negative thresholding (ReLU) and another transform on the resulting matrix. Both transforms are separable, denoted by the operator $T$ and implemented using a Separable Linear (SL) layer, $T (Z) = SL (Z) = W_1 Z W_2 + B$, where $W_1$ and $W_2$ operate in the spatial and the similarity domains. Separability of SL allows a substantial reduction in the number of parameters. As concatenation of two SL layers can be replaced by a single effective SL, we remove one SL layer in such concatenations. The TRT component without the second transform is denoted by $\text{TR}$, and when concatenating $k$ TRT-s, the first $k-1$ blocks should be replaced by $\text{TR}$-s. Another variant we use in our network is TBR, which is a version of TR with batch normalization added before the ReLU.

Another component of the filtering network is an Aggregation block (AGG). This block imposes consistency between overlapping patches $\{z_i\}$ by combining them to a temporary image $x_{tmp}$ using plain averaging and extracting them back from the obtained image by $R_i x_{tmp}$. The complete architecture of the filtering network is presented in Figure 3. The network receives as input two sets of matrices, $\{Z_i\}$ and $\{Z_i^{(2)}\}$, and its output is an array of filtered overlapping patches $\{\hat{z}_i\}$. At first, each of these matrices is multiplied by a diagonal weight matrix $\text{diag}(w_i) Z_i$. Recall that the columns of $Z_i$ (or $\{Z_i^{(2)}\}$) are image patches, where

---

3 We handle boundaries by padding using mirror reflection. Thus, the number of extracted patches is equal to the number of pixels in the image.
the first is the processed one and the rest are its neighbors. The weights \( w_i \) express the network’s belief regarding the relevance of each of the neighbors to the denoising process. These weights are calculated using an auxiliary network denoted as “weight net”, consisting of seven FC (Fully Connected) layers of size \( k \times k \) with batch normalization and ReLU between each two FC’s. The network gets as input the sample variance of the processed patch and \( k - 1 \) squared distances between the patch and its neighbors.

After multiplication by \( \text{diag}(w_i) \) the \( Z_i \) matrix undergoes a series of operations that include transforms, ReLUs and AGG, until it gets to the TBR\(_2\) block, as shown in Figure 3. The aggregation block imposes consistency of the \( \{ Z_i \} \) matrices, which represent overlapping patches, but also causes loss of some information. Thus, we split the flow into two branches: with and without AGG. Since the output of any TR or TBR component is in the feature domain, we wrap the AGG block with T\(_{pre}\) and TR\(_{post}\), where T\(_{pre}\) transforms the features to the image domain, and TR\(_{post}\) transforms the AGG output back to the feature space, while imposing sparsity. The 2\(^{nd}\) scale matrices, \( \{ Z_i^{(2)} \} \), undergo very similar operations as the 1\(^{st}\) scale ones, but with different learned parameters.

TBR\(_2\) applies a joint transform that fuses the features coming from four origins: the 1\(^{st}\) and 2\(^{nd}\) scales with and without aggregation. The columns of all these matrices are concatenated together such that the same spatial transformation \( W_i \) is applied on all. Note that the network size can be reduced at the cost of a slight degradation in performance by removing the TBR\(_1\), TBR\(_1^{(2)}\) and TBR\(_3\) components. We discuss this option in the result section.

3. Image Adaptation

The above designed network, once trained, offers a universal denoising machine, capable of removing white additive Gaussian noise from all images by applying the same set of computations to all images. As such, this machine is not adapted to the incoming image, which might deviate from the general statistics of the training corpus, or could have an inner structure that is not fully exploited. This lack of adaptation may imply reduced denoising performance. In this section we address this weakness by discussing an augmentation of our denoising algorithm that leads to such an adaptation and improved denoising results.

Several recent papers have already proposed techniques for training networks while only using corrupted examples [20, 42, 16, 27, 1, 17, 21, 39, 38]. An interesting special case is where the network is trained on the corrupted image itself. Indeed, this single-image unsupervised learning has been successfully employed by classical algorithms. For example, The KSVD Denoising algorithm [11] trains a dictionary using patches extracted from the corrupted image. Another example is PLE [47], in which a GMM is fitted to the given image. Recent deep learning based methods [42, 27] adopted this approach, training on the corrupted image. However, their obtained performance tends to be non-competitive with fully supervised schemes.

This raises an intriguing question: Could deep regression machines benefit from both supervised and single-image unsupervised training techniques? In this paper we provide a positive answer to this question, while leveraging the fact that our denoising network is lightweight. Our proposed denoiser is able to combine knowledge learned from an external dataset with knowledge that lies in the currently processed image, all while avoiding overfitting. We introduce two novel types of adaptation techniques, external and internal, which should remind the reader of transfer learning. Both adaptation types start by denoising the input image regularly. Then the network is re-trained and updated for few epochs. In the external adaptation case, we seek (e.g., using Google image search) one or few other images closely related to the processed one, and re-train the network on this small set of clean images (and their noisy versions). In the internal adaptation mode, the network is re-trained on the denoised image itself using a loss function of the form

\[
\| f_\theta( \hat{Y} + n) - \hat{Y} \|_2^2,
\]

where \( \hat{Y} = f_\theta(Y) \) is the universally denoised image, and \( n \) is a synthetic noise. For both ex-
ternal and internal adaptations the procedure concludes by denoising the input image by the updated network. 4

While the two adaptation methods seem similar, they serve different needs. External adaptation should be chosen when handling images with special content that is not well represented in the training set. For example, this mode could be applied on non-natural images such as scanned documents. Processing images with special content could be handled by training class-aware denoisers, however this might require a large amount of images for training each class, and holding many networks for covering the variety of classes to handle. For example, [33, 34] train their networks on 900 images per class. In contrast, our external adaptation requires a single network for all images, while updating it for each incoming image. In Section 5 we present denoising experiments, in which applying external adaptation gains substantial improvement of more than 1dB in terms of PSNR. In contrast to external adaptation, the internal one becomes effective when the incoming image is characterized by a high level of self-similarity. For example, as shown in Section 5, applying internal adaptation on images from Urban 100 [13] gains a notable improvement of almost 0.3dB in PSNR on average. Note that these adaptation procedures are not always successful. However, failures usually do not cause performance degradation, indicating that these procedures, in the context of being deployed on a lightweight network, do not overfit. Indeed, while we got a negligible decrease in performance of up to 0.02dB for few images with the internal adaptation, most unsuccessful tests led to a marginal increment of at least 0.02 − 0.05dB.

4. Experimental Results: Universal Denoising

This section reports the performance of the proposed scheme, with a comprehensive comparison to recent SOTA denoising algorithms. In particular, we include in these comparisons the classical BM3D [8] due to its resemblance to our network architecture, the TNRD [6], DnCNN [48], and FFDNet [49] networks, the non-local and high performance NLRN [22] architecture, and the recently published Learned K-SVD (LKSVD) [37] method. We also include comparisons to Lefkimiatis’ networks, NLNet [18] and UNLNet [19], which inspired our work. Our algorithm is denoted as Lightweight Learned Image Denoising with Instance Adaptation (LIDIA), and we present two versions of it, LIDIA and LIDIA-S. The second is a simplified network with slightly weaker performance (see more below).

We start with denoising experiments in which the noise is Gaussian white of known variance. This is the common case covered by all the above mentioned methods. Our network is trained on 432 images from the BSD500 set [26], and the evaluation uses the remaining 68 images (BSD68). The network is trained end-to-end using decreasing learning rate over batches of 4 images using the MSE loss. We start training with the Adam optimizer (learning rate 10−2) and switch to SGD with an initial learning rate of 10−3.

Figure 4 presents a comparison between our algorithm and leading alternative ones by presenting their PSNR versus their number of trained parameters. This figure exposes the fact that the performance of denoising networks is heavily influenced by their complexity. As can be seen, the various algorithms can be split into two categories: lightweight architectures with a number of parameters below 100K (TNRD [6], LKSVD [37], NLNet [18] and UNLNet, and much larger and slightly better performing networks (DnCNN [48], FFDNet [49], and NLRN [22]) that use hundreds of thousands of parameters. Other SOTA denoisers published in the past year are FOCNet [14], RDN+ [51], and N4Net [31]. However, all of these achieve denoising performance similar to NLRN [22], while belonging to the heavier networks category. Therefore, we show comparisons with NLRN as a representative of this class of high performing large architectures. As we proceed in this section, we emphasize lightweight architectures in our comparisons, a category to which our network belongs. Figure 4 shows that our networks (both LIDIA and LIDIA-S) achieve the best results within this lightweight category. Detailed quantitative denoising results per noise level are...
Figure 4: Comparing denoising networks: PSNR performance vs. the number of trained parameters ($\sigma = 25$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise $\sigma$</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>TNRD[6]</td>
<td>31.42</td>
<td>29.57</td>
</tr>
<tr>
<td>DnCNN[48]</td>
<td>31.73</td>
<td>29.23</td>
</tr>
<tr>
<td>BM3D[8]</td>
<td>31.07</td>
<td>28.57</td>
</tr>
<tr>
<td>NLRN[22]</td>
<td>31.88</td>
<td>29.41</td>
</tr>
<tr>
<td>NLNet[18]</td>
<td>31.50</td>
<td>28.98</td>
</tr>
<tr>
<td>NLMS(our)</td>
<td>31.62</td>
<td>29.11</td>
</tr>
</tbody>
</table>

Table 1: B/W denoising performance: Best PSNR in red, and best PSNR within the low-weight category in blue.

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise $\sigma$</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFDNet[49]</td>
<td>33.87</td>
<td>31.21</td>
</tr>
<tr>
<td>CDnCNN[48]</td>
<td>33.99</td>
<td>31.31</td>
</tr>
<tr>
<td>CBM3D[7]</td>
<td>33.50</td>
<td>30.68</td>
</tr>
<tr>
<td>CNLNet[18]</td>
<td>33.81</td>
<td>31.08</td>
</tr>
<tr>
<td>C-LIDIA(our)</td>
<td>34.03</td>
<td>31.31</td>
</tr>
</tbody>
</table>

Table 3: Color denoising performance: Best PSNR in red, and best PSNR within the low-weight category in blue.

Figure 5: Color image denoising example with $\sigma = 50$. Differences between the results are better seen when zooming in on the region marked with a red rectangle. For more examples see the Supplementary Material.

Blind denoising, i.e., denoising with unknown noise level, is a useful feature when it comes to neural networks. This allows using a fixed network for performing image denoising, while serving a range of noise levels. This is a more practical solution, when compared to the one discussed above, in which we have designed a series of networks, each trained for a particular $\sigma$. We report blind denoising performance of our architecture and compare to...
Table 4: Blind denoising performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise $\sigma$</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>DnCNN-b[48]</td>
<td>31.61</td>
<td>29.16</td>
</tr>
<tr>
<td>LIDIA-b (ours)</td>
<td>31.54</td>
<td>29.06</td>
</tr>
</tbody>
</table>

Table 5: LIDIA vs. its smaller version LIDIA-S.

<table>
<thead>
<tr>
<th>Method</th>
<th>Noise $\sigma$</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>LIDIA</td>
<td>31.62</td>
<td>29.11</td>
</tr>
<tr>
<td>LIDIA-S</td>
<td>31.57</td>
<td>29.08</td>
</tr>
<tr>
<td>LIDIA-b</td>
<td>31.54</td>
<td>29.06</td>
</tr>
<tr>
<td>LIDIA-S-b</td>
<td>31.49</td>
<td>29.01</td>
</tr>
</tbody>
</table>

similar results by DnCNN-b [48] (a version of DnCNN that has been trained for a range of $\sigma$ values) and UNLNet [19]. Our blind denoising network (denoted LIDIA-b) preserves all its structure, but simply trained by mixing noise level examples in the range $10 \leq \sigma \leq 30$. The evaluation of all three networks is performed on images with $\sigma = [15, 25]$. The results of this experiment are brought in Table 4. As can be seen, our method obtains a higher PSNR than UNLNet, while being slightly weaker than DnCNN-b. Considering again the fact that our network has nearly 10% of the parameters of DnCNN-b, we can say that our approach leads to SOTA results in the lightweight category.

Our LIDIA denoising network can be further simplified by removing the TBR$_1$, TBR$_2$, and TBR$_3$ components. The resulting smaller network, denoted by LIDIA-S, contains 30% less parameters than the original LIDIA architecture (see Table 2), while achieving slightly weaker performance. Table 5 shows that for both regular and blind denoising scenarios, LIDIA-S achieves an average PSNR that is only 0.05dB lower than the full-size LIDIA network. For a visual comparison between NLMS and NLMS-S results see Section 4 of the Supplementary Material.

5. Experimental Results: Network Adaptation

We turn to present results related to external and internal image adaptation. We compare the performance of our network (before/after adaptation) with DnCNN [48]. Unless said otherwise, all adaptation results are obtained via 5 epochs, requiring few minutes (depending on the image size) on Nvidia GeForce GTX 1080 Ti GPU. The adaptation does not require early stopping – over-training the network leads to similar and sometimes better results.

External adaptation is useful when the input image deviates from the statistics of the training images. We demonstrate this adaptation on two non-natural images: an astronomical photograph and a scanned document. Applying a universal denoising on these images create pronounced artifacts, as can be seen in Figures 1 and 6. Adapting LIDIA (our network) by training on a single similar image significantly improves the PSNR and the visual quality of the results. The training images used in the above experiments are shown in the Supplementary Material, along with a graph showing the PSNR vs. the number of adaptation batches.

Table 6 summarises quantitative denoising results of applying the proposed adaptation procedure on Urban100 and BSD68 image sets. This table shows that the internal adaptation can improve the denoising capability of the network, but the benefit varies significantly from image to another, depending on its content. Figure 7 presents histograms of improvement per image. Clearly, the proposed adaptation has a minor effect on most of images in BSD68, but it succeeds well on Urban100, leading to a significant boost. This difference can be attributed to the pronounced self similarity that is inherent in the Urban100 images. Figure 8 presents...
<table>
<thead>
<tr>
<th>Image set</th>
<th>CDnCNN [48]</th>
<th>C-LIDIA (ours) with adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban100</td>
<td>28.16</td>
<td>28.23</td>
</tr>
<tr>
<td>BSD68</td>
<td>28.01</td>
<td>27.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28.04</td>
</tr>
</tbody>
</table>

Table 6: Internal adaptation for color images.

<table>
<thead>
<tr>
<th>DnCNN</th>
<th>FC-AIDE (adapt.)</th>
<th>LIDIA (adapt.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>26.28</td>
<td>25.55</td>
<td>26.42</td>
</tr>
<tr>
<td></td>
<td>26.51</td>
<td>26.71</td>
</tr>
</tbody>
</table>

Table 7: Internal adaptation on Urban100 grayscale images.

Figure 7: Improvements obtained by the internal adaptation.

Visual example of the internal adaptation. The image presented in this figure is characterised by a strong self-similarity, which our adaptation is able to exploit. As evident from Figure 7a, internal adaptation is not always successful, and it may lead to slightly decreased PSNR. However, we note that among the 168 test images (BSD68 and Urban100) only two such failures were encountered, both leading to a degradation of 0.02dB. We conclude that internal adaptation is robust and cannot do much harm.

N-AIDE [4] and FC-AIDE [5] fine-tune their networks for each input image in a fashion similar to our adaptation, by constraining each output pixel to be a polynomial function of the corresponding input one. As FC-AIDE outperforms N-AIDE, we focus on comparing LIDIA to it. Our network is much lighter – FC-AIDE contains 820K parameters while LIDIA uses 62K. Also, FC-AIDE is designed for grayscale denoising, while LIDIA can handle color as well. For a quantitative comparison, we run LIDIA with an internal adaptation on the grayscale Urban100 with $\sigma = 50$. The results of this experiment are brought in Table 7. As can be seen, LIDIA obtains a higher PSNR in both the universal and the adapted versions.

6. Conclusion

This work presents a lightweight universal network for supervised image denoising, demonstrating competitive performance with SOTA. Our patch-based architecture exploits non-local self-similarity and representation sparsity, augmented by a multiscale treatment. Separable linear layers, combined with non-local $k$ neighbor search, allow capturing non-local interrelations between pixels. In addition, this work offers two image-adaption techniques, aiming for improved denoising performance by better tuning the above universal network to the incoming noisy image. We demonstrate the effectiveness of these methods on images with unique content or having significant self-similarity.
References


