# Identifying Recurring Patterns with Deep Neural Networks for Natural Image Denoising

Zhihao Xia Ayan Chakrabarti
Washington University in St. Louis
{zhihao.xia,ayan}@wustl.edu

## Abstract

*Image denoising methods must effectively model, implicitly or explicitly, the vast diversity of patterns and textures that occur in natural images. This is challenging, even for modern methods that leverage deep neural networks trained to regress to clean images from noisy inputs. One recourse is to rely on "internal" image statistics, by searching for similar patterns within the input image itself. In this work, we propose a new method for natural image denoising that trains a deep neural network to determine whether patches in a noisy image input share common underlying patterns. Given a pair of noisy patches, our network predicts whether different sub-band coefficients of the original noise-free patches are similar. The denoising algorithm then aggregates matched coefficients to obtain an initial estimate of the clean image. Finally, this estimate is provided as input, along with the original noisy image, to a standard regression-based denoising network. Experiments show that our method achieves state-of-the-art color image denoising performance, including with a blind version that trains a common model for a range of noise levels, and does not require knowledge of level of noise in an input image. Our approach also has a distinct advantage when training with limited amounts of training data.*

## 1. Introduction

The sheer diversity of content, that can be present in photographs of natural scenes, makes them a challenge for algorithms that must model their statistics for various image restoration tasks, including the classical task of image denoising: recovering an estimate of a clean image from a noisy observation. A common approach is to rely on image models for local image regions—either explicitly as parametric priors or implicitly as estimators trained via regression—with parameters learned on databases of natural images [3, 5, 6, 9, 21, 22, 28, 29, 30, 32].

An important class of methods adopt a different modeling approach, to exploit self-similarity in images by relying on their "internal statistics" [2, 7, 8]. A particularly successful example from this class is the BM3D [7, 8] algorithm, which identifies sets of similar patches in noisy images using sum of squared distances (SSD) as the matching metric, and then uses the statistics of each set to denoise patches in that set. Applying this process twice, BM3D produced high-quality estimates that, until recently, represented the state-of-the-art in image denoising performance.

However, recent methods have been able to exceed this performance by using neural networks trained to regress to clean image patches from noisy ones [3, 6, 29]. With carefully chosen architectures, these methods are able to use the powerful expressive capacity of neural networks to better learn and encode image statistics from external databases, and thus exceed the capability of self-similarity based methods. In this work, we describe a denoising method that brings the expressive capacity of neural networks to the task of identifying and leveraging recurring patterns in the underlying images from their noisy observations.

Our contributions are as follows:

- We introduce a novel matching network that looks at pairs of noisy patches at a time, and makes fine-grained predictions of the similarity of their underlying clean versions. Specifically, our network outputs *separate* matching scores for different groups of wavelet coefficients, to exploit similarities that exist at some orientations and scales, but not others. These scores are used for averaging to form an initial denoised estimate.
- We propose a *two-step process* to train this matching network, with respect to denoising quality, that leads to convergence to a better network model.
- We combine the matching network with a standard regression step to yield a complete algorithm that achieves *state-of-the-art* denoising performance.
- We carry out extensive experiments on multiple datasets and show that our method consistently yields higher quality estimates than the state-of-the-art on a variety of metrics. Indeed, even a *blind* version of

our model—that does not have knowledge of the noise level—outperforms state-of-the-art methods that do.

- Finally, we show that our method has a distinct advantage over regression-based networks when learning from only a small amount of training data. In these cases, our method is able to generalize better due to its reliance on per-image internal statistics.

## 2. Related Work

Denoising is a classical problem in image restoration. In addition to its practical utility in improving the quality of photographs taken in low-light or by cheaper sensors, image denoisers can be used as generic "plug-and-play" priors within iterative approaches to solve a larger variety of generic image restoration tasks (e.g., [1, 4, 23, 29]).

Many classical approaches to image denoising are based on exploiting statistics of general natural images, using estimators or explicit statistical priors [9, 21, 22, 32], whose parameters are learned from datasets of clean images. A different category of approaches use patch-recurrence of self-similarity [2, 8], to address the fact that there is significant diversity in content across images, while the variation within a *specific* image is far more limited. There is a natural trade-off between these two approaches: methods based on external statistics have the opportunity to learn them from clean images, but these statistics may be too general for a specific image; while those based on self-similarity work with a more relevant model for each image, but must find a way to derive its parameters from the noisy observation itself. We refer the reader to the work of Mosseri et al. [19] for an insightful discussion.

BM3D [8] is a particularly successful method that is based on self-similarity. It works by organizing similar patches into groups (using SSD as the matching metric, and doing two rounds of matching), and denoising patches based on the statistics of its group through collaborative filtering. However, recent algorithms [3, 5, 6, 28, 29, 30] have been able to surpass BM3D's performance using estimators trained on external datasets, leveraging the powerful implicit modeling capacity of deep neural networks.

In this work, we propose a new method that uses neural networks to identify and exploit self-similarity in noisy images. This was also the goal of methods by Lefkimmiatis [15] and Yang and Sun [25], who proposed interesting approaches that are based on designing network architectures that "un-roll" and carry out the computations in BM3D and non-local means denoising, and then train the parameters of these steps discriminatively through back-propagation, resulting in performance gains over the baseline methods. More recently, [16] and [20] proposed using non-local aggregation steps on the intermediate activations of their network. In contrast, we employ a substantially different approach: denoising in our framework is achieved by weighted averaging of different sub-band coefficients of the noisy image patches themselves, while our network is tasked with predicting optimal values of these weights by matching. Also note that, unlike [20] which makes a hard selection of a small number of the best matches for every reference location, we average across a dense set of matches in a large neighborhood, with different continuous weights.

The primary component of our method is a network that must learn to match patches through noise. Several neural network-based methods have been proposed to solve matching problems [14, 26, 27], with the goal of finding correspondences across images for tasks like stereo. Our method is motivated by their success, and we borrow several design principles of their architectures. However, our matching network has a completely different task: denoising. Our network is thus trained with a loss optimized for denoising (as opposed to classification or triplet losses), and instead of predicting a single matching score for a pair of patches, produces a richer description of their commonality with distinct scores for different sub-bands.

## 3. Proposed Denoising Algorithm

Our goal is to produce an estimate of an image $X$ given observation $Y$ that is degraded by i.i.d. Gaussian noise, i.e.,

$$Y = X + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_z^2 I). \tag{1}$$

Our algorithm leverages the notion that many patterns will occur repeatedly in different regions in the underlying clean image $X$, while the noise in those regions in $Y$ will be uncorrelated and can be attenuated by averaging. In this section, we describe our approach to training and using a deep neural network to identify these recurring patterns from the noisy image, and forming an initial estimate of $X$ by averaging matched patterns. We then use a second network to regress to the final denoised output from a combination of these initial estimates and the original noisy observation.

### 3.1. Denoising by Averaging Recurring Patterns

Our initial estimate is formed by denoising individual patches in the image, by computing a weighted average over neighboring noisy patches with weights provided by a matching network. Formally, given the noisy observation $Y$ of an image $X$, we consider sets of overlapping patches $\{y_i = P_i Y\}$ (corresponding to clean versions $\{x_i = P_i X\}$), where each $P_i$ is a linear operator that crops out intensities of a different square patch (of size $8 \times 8$ in our implementation) from the image. We then use a decorrelating color space followed by a Harr wavelet transform to obtain coefficients $s_i = T y_i$ (corresponding to clean versions $r_i = T x_i$), where the orthonormal matrix $T$ represents the color and wavelet transforms.

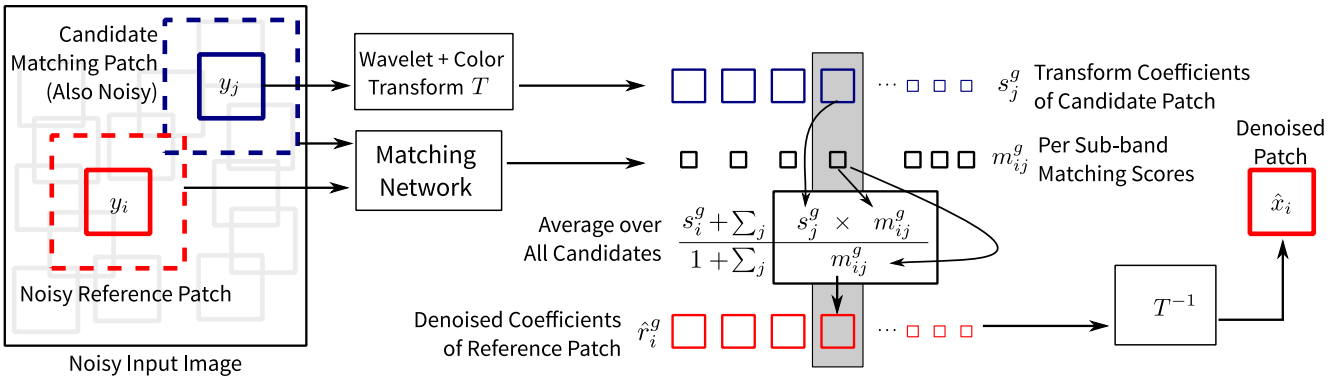


Figure 1. Overview of Our Approach to Patch Denoising. We produce estimates of clean patches by weighted averaging across a candidate set of nearby patches in the observed noisy input. We decompose every patch using a wavelet and de-correlating color transform into sets of sub-band coefficients, with coefficients at the same scale, orientation, and color channel grouped together in each set. We then train a neural network that, given a pair of patches, computes a vector of matching scores—one for each group of coefficients. For every patch, we compute these score vectors with respect to all candidate patches. The denoised patch is obtained by averaging, across all candidates, of each group of coefficients using its corresponding matching scores.

We group these coefficients into sets $\{s_i^g\}$ where each set includes all coefficients with the same orientation (horizontal, vertical, or diagonal derivative), scale or pyramid level, and color channel[1]. Then, for every patch $y_i$ we consider a set of candidate matches composed of other noisy patches in the image $y_j, j \in \mathcal{N}_i$, from a large neighborhood around $i$. As illustrated in Fig. 1, our method produce an estimate of the denoised coefficients $\hat{r}_i$ as a weighted average of the corresponding coefficients of the candidate patches:

$$\hat{r}_i^g = \left(1 + \sum_{j \in \mathcal{N}_i} m_{ij}^g\right)^{-1} \left(s_i^g + \sum_{j \in \mathcal{N}_i} m_{ij}^g s_j^g\right), \quad (2)$$

where $m_{ij}^g \geq 0$ are scalar matching weights that are a prediction of the similarity between the $g^{th}$ set of coefficients in patches $i$ and $j$ respectively.

This gives us a denoised estimate for each patch $\hat{x}_i$ in the image as $T^{-1}\hat{r}_i$. We then obtain an estimate $\hat{X}$ of the full clean image simply by averaging the denoised patches $\hat{x}_i$, i.e., the denoised estimate of each pixel is computed as the average of its estimate from all patches that contain it.

### 3.2. Predicting Matches from Noisy Observations

The success of our match-averaging strategy in (2) depends on obtaining optimal values for the matching scores $m_{ij}^g$. Intuitively, we want $m_{ij}^g$ to be high when the *clean* coefficients $r_i^g$ and $r_j^g$ are close, so that the averaging in (2) will attenuate noise and yield $\hat{r}_i^g$ close to $r_i^g$. Conversely, we want $m_{ij}^g$ to be low where the two sets of underlying clean coefficients are not similar, because averaging them would yield poor results, potentially worse than the noisy observation itself. However, note that while the optimal values of these matching scores depend on the characteristics of the clean coefficients $\{r_i^g\}$, we only have access to their noisy counterparts $\{s_j^g\}$.

Therefore, we train a neural network $\mathcal{M}$ to predict the matching scores given a pair of larger noisy patches ($16 \times 16$ in our implementation) $y_i^+$ and $y_j^+$ centered around $y_i$ and $y_j$ respectively: $m_{ij} = \mathcal{M}(y_i^+, y_j^+)$, where $m_{ij} = [\ldots, m_{ij}^g, \ldots]$ is a vector of matching scores for all sets of coefficients. We don't require the output of the network $\mathcal{M}$ to be symmetric ($m_{ij}$ need not be the same as $m_{ji}$), and we use the same network model for evaluating all patch pairs, being agnostic to their absolute or relative locations.

The matching network $\mathcal{M}$ has a Siamese-like architecture as illustrated in Fig. 2. It begins with a common feature extraction sub-network applied to both input patches to produce a feature-vector for each. This sub-network has a receptive field of $16 \times 16$, and includes a total of fourteen convolutional layers with skip connections [11] including at the final output (see Fig. 2). The computed feature-vectors for each of the two inputs are then concatenated and passed through a comparison sub-network, which comprises of a set of five fully-connected layers. All layers have ReLU activations, except for the last which uses a sigmoid to produce the match-scores $m_{ij}^g$. These scores are thus constrained to lie in $[0, 1]$. Note that during inference, the feature extraction sub-network needs to be applied only once to compute feature-vectors for all patches in a fully-convolutional way. Only the final five fully connected layers need to be repeatedly applied for different patch pairs.

Observe that our matching network takes the noisy patches directly as input, while using the wavelet and color transforms to parameterize its outputs, as a means of pro-

[1]For $8 \times 8$ patches, this gives us 30 coefficient groups: 27 corresponding to 3 color channels, 3 scales, and 3 derivative orientations; and an additional 3 coefficients for the scaling coefficients of the 3 color channels.

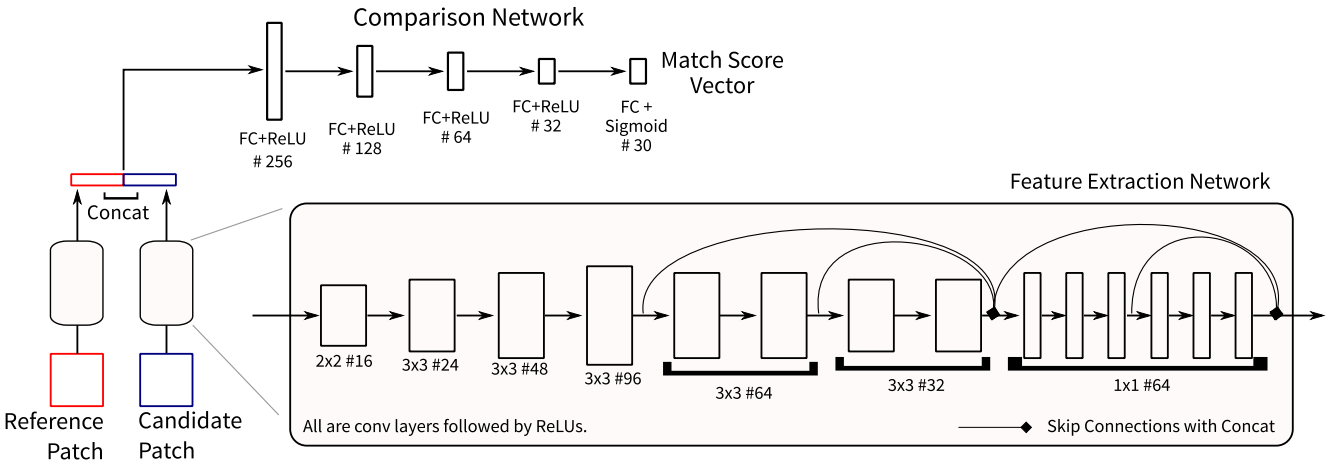


Figure 2. Matching Network Architecture. To produce the matching scores $m_{ij}^g$, we first extract a feature vector for all patches by passing the image through a feature extraction network, comprised of a set of convolutional layers with skip connections (where the join is performed by a concatenate operation. Then, for any pair of patches, we take the corresponding pair of feature vectors, and pass them after concatenation through a series of fully-connected layers. The final layer has a sigmoid activation, yielding scores that lie between 0 and 1.

viding a more fine-grained characterization of similarity between patches than a single score. Moreover, although the inputs to the network itself consist only of a pair of relatively small patches, it enables aggregation from a dense set of patches in a large neighborhood, through repeated application on multiple patch pairs and averaging as per (2).

### 3.3. Training

We train the matching network $\mathcal{M}$ to produce matching scores that are optimal with respect to the quality of the denoised patches $\hat{x}_i$. Specifically, we use an $L_2$ loss between the true and estimated clean patches:

$$L = \|x_i - \hat{x}_i\|^2 = \sum_g \|\hat{r}_i^g - r_i^g\|^2, \tag{3}$$

where the denoised coefficients $\hat{r}_i^g$ are computed using (2) based on matching-scores predicted by the network. Note that the loss for a single patch $x_i$ will depend on matching scores produced by the network for $x_i$ paired with all candidate patches in its neighborhood $\mathcal{N}_i$.

While it is desirable to train the network in this end-to-end manner with our actual denoising approach, we empirically find that training the network with this loss from a random initialization often converges to a sub-optimal local minima. We hypothesize that this is because we compute gradients corresponding to a large number of matching scores (all candidates in $\mathcal{N}_i$) with respect to supervision only from a single denoised patch.

Thus, we adopt a pre-training strategy using a loss defined on pairs of patches at a time, using a simplified loss for denoising patch $i$ by averaging it with patch $j$ as:

$$\hat{L}_{ij} = \sum_g \frac{\|s_i^g - r_i^g\|^2 + {m_{ij}^g}^2 \|s_j^g - r_i^g\|^2}{\left(1 + m_{ij}^g\right)^2}. \tag{4}$$

This is equivalent to the actual loss in (3) with performing the averaging in (2) with only one candidate patch $j$, by dropping the cross term between $(x_i - y_i)$ and $(x_i - y_j)$, i.e., by assuming the noise is un-correlated with the difference between the two patches. It is interesting to note here if we assume that the deviations between reference and candidate patches are un-correlated, for different candidates $j \in \mathcal{N}_i$, then the optimal averaging weight for a given candidate is the same whether averaging with one or multiple candidates. The modified loss in (4) serves as a good initial proxy for pre-training, but since the un-correlated deviation assumption does not hold in practice, we follow this with training with the actual loss in (3).

In particular, we pre-train the network for a number of iterations using the modified loss in (4)—constructing the training set by considering all non-overlapping patches $i$ in an image, with random shuffling to select candidate $j$ for each patch $i$, and train with respect to the loss of both matching $i$ to $j$ and vice-versa. This allows us to compute updates with respect to a much more diverse set of patches into a training batch, and to make maximal use of the feature extraction computation during training. The pre-training step is followed by training the network with the true loss in (3) till convergence—here, we extract a smaller number of training reference patches from each image, along with *all* their neighboring candidates.

### 3.4. Final Estimates via Regression

While the initial estimates produced by our method as described above are of reasonable quality, they are limited by (2) restricting every denoised patch to be a weighted average of observed noisy patches. To overcome this and achieve further improvements in quality, we use a second

network trained via traditional regression to derive our final denoised estimate. Specifically, we adopt the architecture of IRCNN [29] with has seven dilated convolutional layers. In our case, this network takes a six-channel input formed by concatenating the original noisy input and our initial denoised estimate from match-based averaging. The output of the last layer is interpreted as a residual, and added to the initial estimates to yield the final denoised image output.

After the matching network has been trained, we generate sets of clean, noisy, and initial denoised estimates. This serves as the training set for this second network which is trained using an $L_2$ regression loss. We find that this step leads to further improvement over our initial estimates, while also outperforming state-of-the-art denoising networks (including IRCNN [29] itself).

# 4. Experiments

## 4.1. Preliminaries

We train our algorithm on a set of 1600 color images from the Waterloo exploration dataset [17], and 168 images from the BSD-300 [18] train set, using the remaining 32 images for validation and parameter setting. We train our network using noisy observations generated by adding Gaussian noise to clean images in the training set. Unless otherwise specified, we construct the candidate set $\mathcal{N}_i$ of patches by considering all the overlapping patches in a $31 \times 31$ search window around patch $i$.

We use Adam [13] to train both the matching and regression networks, with an initial learning rate of $10^{-3}$. We pre-train the matching network for a 100k iterations based on the modified loss (4), with batches of 16 images and pairing all non-overlapping patches with randomly shuffled counterparts. This leads to a large number of ordered matching pairs for pre-training in each batch. We then continue training the matching network with the true loss in (3), in this case forming a batch with 256 unique reference patches from various images, and computing matching scores for each with respect to all $31^2$ candidates. We train with this loss till saturation, with two learning rate drops of $10^{0.5}$. Once the matching network is trained, we store a set of noisy and denoised version of our training set, and use these to train the regression network (with the same training schedule, but without pre-training). Our code and trained models are available at `https://projects.ayanc.org/rpcnn/`.

## 4.2. Denoising with Known Noise Level

We evaluate our method for the task of color image denoising at five different noise levels, corresponding to additive white Gaussian noise with standard deviations of $\sigma$ =25, 35, 50, and 75 gray levels. We train a separate network model for each level, and report their performance in Table 1 on four datasets: Urban-100 [12], Kodak-24 [10], CBSD-68 [21], and McMaster [31]. For comparison, we also show results from a number of other state-of-the-art color denoising methods [7, 15, 25, 28, 29, 30]. We evaluate performance in terms of the standard PSNR and SSIM [24] metrics, and to measure robustness, report worst-case errors as the $25^{th}$%-ile among PSNR values of all individual $8 \times 8$ patches in all images in each dataset. We find that our results are consistently more accurate across all datasets, with significant improvements over state-of-the-art methods at higher noise levels (with an improvement of 0.63 dB at noise-level 75 on the Urban-100 images). Moreover, not only are our denoised estimates more accurate on average in terms of PSNR and SSIM, our worst-case performance is also better—highlighting the robustness of our approach.

We include examples of denoised images in Fig. 3 for a qualitative evaluation, and see that denoised results from our method often contain better reconstructions of texture and image detail than state-of-the-art denoising methods. In general, we find that our method has an advantage when a scene contains many repeating textures as expected, and also when it contains unique patterns—that are rare in training data and which regression-based methods are thus unable to reliably estimate. For images with limited repeating patterns, the burden of denoising then falls more to our second regression network, which then is able to still achieve results of acceptable quality at the level of standard regression-based methods.

## 4.3. Blind Denoising

Next, we consider the task of blind denoising, when the level of Gaussian noise in an observed image is unknown. For this, we follow the approach of [28] in training a common model for a range of noise levels $\sigma \in [0, 55]$, by adding Gaussian noise with $\sigma$ chosen randomly for each image during training. Note that unlike for FFDNet [30], the noise level for a specific input image is *not* provided to our model. Table 1 also includes an evaluation of this version of our method (as Ours-Blind). We find that its performance is only slightly lower than that of our noise-specific networks, and still better in almost all cases than that of state-of-the-art methods that are aware of the level of noise in their inputs. This represents an attractive and practically useful variant of our method—which does not require maintaining multiple models for each noise level, and can be applied even when the noise level is unknown.

## 4.4. Training with Limited Data

For many forms of image data and measurement models (*e.g.*, medical images), a large amount of training data is difficult to acquire. Here, our method presents an advantage because of its focus on leveraging common patterns and textures in the input image itself, rather than those it

| | Method | $\sigma$=75 | | | $\sigma$=50 | | | $\sigma$=35 | | | $\sigma$=25 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | 25%-ile | PSNR | SSIM | 25%-ile | PSNR | SSIM | 25%-ile | PSNR | SSIM | 25%-ile |
| Urban-100 | *CBM3D [7] | 25.97 | 0.784 | 24.09 | 27.94 | 0.843 | 25.96 | 29.27 | 0.875 | 27.07 | 31.38 | 0.912 | 29.20 |
| | IRCNN [29] | - | - | - | 27.69 | 0.842 | 25.63 | 29.50 | 0.881 | 27.39 | 31.20 | 0.911 | 29.06 |
| | FFDNet [30] | 26.05 | 0.793 | 23.98 | 28.05 | 0.850 | 25.93 | 29.78 | 0.887 | 27.61 | 31.40 | 0.914 | 29.17 |
| | Ours-Blind | - | - | - | 28.57 | 0.859 | 26.47 | 30.21 | 0.893 | 28.04 | 31.70 | 0.917 | 29.49 |
| | Ours | **26.68** | **0.811** | **24.64** | **28.62** | **0.862** | **26.52** | **30.26** | **0.895** | **28.09** | **31.81** | **0.919** | **29.59** |
| Kodak-24 | *CBM3D [7] | 26.82 | 0.714 | 25.07 | 28.45 | 0.775 | 26.51 | 29.90 | 0.821 | 27.77 | 31.67 | 0.868 | 29.51 |
| | IRCNN [29] | - | - | - | 28.81 | 0.792 | 26.76 | 30.43 | 0.838 | 28.32 | 32.03 | 0.878 | 29.91 |
| | CDnCNN [28] | 25.04 | - | - | 28.85 | - | - | 30.46 | - | - | 32.03 | - | - |
| | FFDNet [30] | 27.27 | 0.733 | 25.30 | 28.98 | 0.793 | 26.89 | 30.57 | 0.841 | 28.41 | 32.13 | 0.879 | 29.95 |
| | Ours-Blind | - | - | - | 29.21 | 0.803 | 27.11 | 30.78 | 0.849 | 28.63 | 32.31 | **0.884** | 30.14 |
| | Ours | **27.56** | **0.748** | **25.59** | **29.25** | **0.805** | **27.15** | **30.81** | **0.849** | **28.66** | **32.34** | **0.884** | **30.19** |
| CBSD-68 | *CBM3D [7] | 25.75 | 0.698 | 23.60 | 27.38 | 0.767 | 25.07 | 28.89 | 0.821 | 26.46 | 30.71 | 0.872 | 28.25 |
| | *CBM3D-Net [25] | - | - | - | 27.48 | - | - | - | - | - | 30.91 | - | - |
| | *CNL-Net [15] | - | - | - | 27.64 | - | - | - | - | - | 30.96 | - | - |
| | IRCNN [29] | - | - | - | 27.86 | 0.792 | 25.54 | 29.50 | 0.844 | 27.14 | 31.16 | 0.886 | 28.81 |
| | CDnCNN [28] | 24.47 | - | - | 27.92 | - | - | 29.58 | - | - | 31.23 | - | - |
| | FFDNet [30] | 26.24 | 0.723 | 23.92 | 27.96 | 0.792 | 25.56 | 29.58 | 0.844 | 27.14 | 31.21 | 0.886 | 28.78 |
| | Ours-Blind | - | - | - | 28.03 | 0.797 | 25.65 | 29.62 | 0.848 | 27.21 | 31.22 | **0.888** | 28.82 |
| | Ours | **26.39** | **0.734** | **24.08** | **28.06** | **0.799** | **25.69** | **29.64** | **0.849** | **27.24** | **31.24** | **0.888** | **28.85** |
| McMaster | *CBM3D [7] | 26.80 | 0.735 | 24.78 | 28.52 | 0.794 | 26.41 | 29.92 | 0.833 | 27.73 | 31.66 | 0.874 | 29.49 |
| | IRCNN [29] | - | - | - | 28.91 | 0.807 | 26.78 | 30.59 | 0.851 | 28.48 | 32.18 | 0.885 | 30.11 |
| | CDnCNN [28] | 25.10 | - | - | 28.61 | - | - | 30.14 | - | - | 31.51 | - | - |
| | FFDNet [30] | 27.33 | 0.760 | 25.19 | 29.18 | 0.816 | 26.99 | 30.81 | 0.857 | 28.62 | **32.35** | 0.889 | **30.20** |
| | Ours-Blind | - | - | - | 29.31 | 0.824 | 27.14 | 30.85 | 0.861 | 28.69 | 32.31 | **0.890** | 30.14 |
| | Ours | **27.59** | **0.775** | **25.47** | **29.35** | **0.826** | **27.16** | **30.90** | **0.863** | **28.70** | 32.33 | **0.890** | 30.17 |

*Other methods that are based on internal image statistics. See supplementary for comparisons of these to denoising with only our matching network.

Table 1. Denoising Performance at Various Noise Levels on Different Datasets. We report performance in terms of Average PSNR (dB) and SSIM. To gauge robustness, we also report the $25^{th}$%-ile worst-case PSNR (dB), computed across $8 \times 8$ patches across each dataset. Ours-Blind refers to results from a common model of our method that is trained for a range of noise levels $\sigma \in [0, 55]$ (and does not have knowledge of the specific noise level of its input).

has observed previously in a training set. We demonstrate this advantage by comparing our method to IRCNN [29] in Fig. 4, when both methods are trained with only a few training images (selected from our complete training set).

We assume a fixed known noise level of $\sigma = 50$, and train versions of both models with different training set sizes—ranging from 10 to 50 images. Since overfitting is an issue with so little data, we track the performance of both methods on a validation set of 32 images through training, and choose the version with highest validation accuracy. We do not drop the learning rate for either method in this setting. Figure 4 shows the average PSNR for both methods on the Urban-100 dataset [12], for different training set sizes. While our method outperforms IRCNN [29] in all cases, the performance gap is notably larger for smaller training sets—at 1.5 dB when training with only 10 images.

### 4.5. Matching Network Analysis

Our matching network is a key component of our denoising algorithm. We end by analyzing its performance when applied to neighborhoods of different sizes, and the role that pre-training plays in convergence to a good solution. We also visualize the matching scores it generates, and how these differ across different groups of sub-bands. Furthermore, the supplementary includes evaluations of denoising using just the matching network (i.e., without the second regression network) on all datasets.

**Effect of Window Size and Pre-training.** In Table 2, we characterize the trade-off between quality and computational cost when choosing different search window sizes over which to match and average patches. For different window sizes, we report average PSNR (over our validation set) for our initial match-averaged estimates when training with a known noise level of $\sigma = 25$. We also report the corresponding running time required to compute matching scores and perform the averaging for different window sizes—for a $256 \times 256$ input image on an NVIDIA 1080Ti GPU. Note that computing the initial estimates takes a majority of the time in our denoising method—the following regression step takes only an additional 0.01 seconds, and is independent of window size.

As expected, running time goes up roughly linearly with

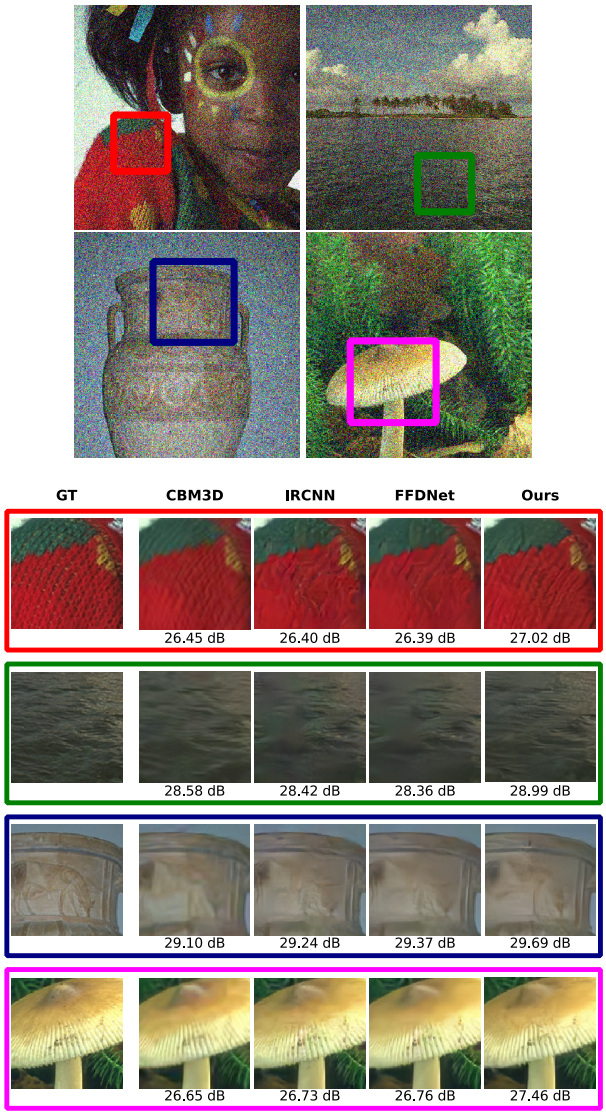


Figure 3. Example Crops of Denoised Images ($\sigma = 50$). Compared to state-of-the-art denoising algorithms (IRCNN [29], DnCNN [28], and FFDNet [30]), we see that our overall method is often able to recover texture and detail with higher fidelity, by exploiting similar patterns in the input image itself.

the number of candidate matches (i.e., as square of the search window size), but we find that the drop in PSNR is a modest 0.06 dB when going down to a $23 \times 23$ window. Table 2 also demonstrates the importance of pre-training, and reports performance (again, of our initial estimates) achieved by a network that is initialized with random weights instead of with pre-training. We find that this leads to a PSNR drop of about 0.1 dB, highlighting that pre-training is important for convergence to a good model.

**Visualizing Matching Scores.** For a number of reference patches cropped from different training images, and their corresponding search windows, we visualize the matching

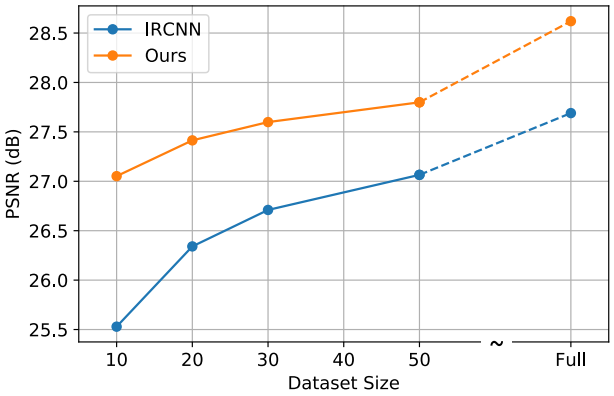


Figure 4. Effect of training set size. We report average PSNR on Urban-100 [12] for denoising at $\sigma = 50$ with our method and IRCNN [29], when both are trained with a limited number of training images ("Full" represents using the entire training set for our method, and the official model for IRCNN). While our estimates are always more accurate than those from IRCNN, the gap is especially higher when the number of training images is small.

| Window Size | 15 | 23 | 31 | 31 (No Pre-training) |
|---|---|---|---|---|
| PSNR (dB) | 31.31 | 31.40 | 31.46 | 31.35 |
| Run Time | 1.07s | 2.47s | 4.42s | 4.42s |

Table 2. Window Size and Pre-Training Ablation. We report average PSNR (db) of the initial match-averaged estimates from our method on a validation set for $\sigma = 25$. Run-times are for $256 \times 256$ images on a 1080Ti GPU.

scores predicted by our network in Fig. 5. We show the average matching score across all sub-bands, as well as average weights corresponding to combinations of sets at the same wavelet scale (averaging over color channels and orientation), and at the same orientation (averaging over scale and color). We see that the matching network produces very different averaging weights for different sub-bands.

We find that that the weights tend to be generally higher at the finest scale (indicating more averaging), and lowest for the scaling coefficients. This is likely because the highest-frequencies are close to zero in most patches, and thus to each other. For lower-frequencies and DC values, the network selects only those patches that are close to the reference patch (in the clean image). For different orientations, the high matches are sometimes concentrated at different locations for the same reference, especially when there are strong edges and repeating textures. Thus, free from the contraint of matching patches as a whole with a single score, our network finds different sets of matches for different sub-bands in order to achieve optimal denoising.

## 5. Conclusion

In this work, we presented a denoising method that employed a neural network to identify and exploit recurring

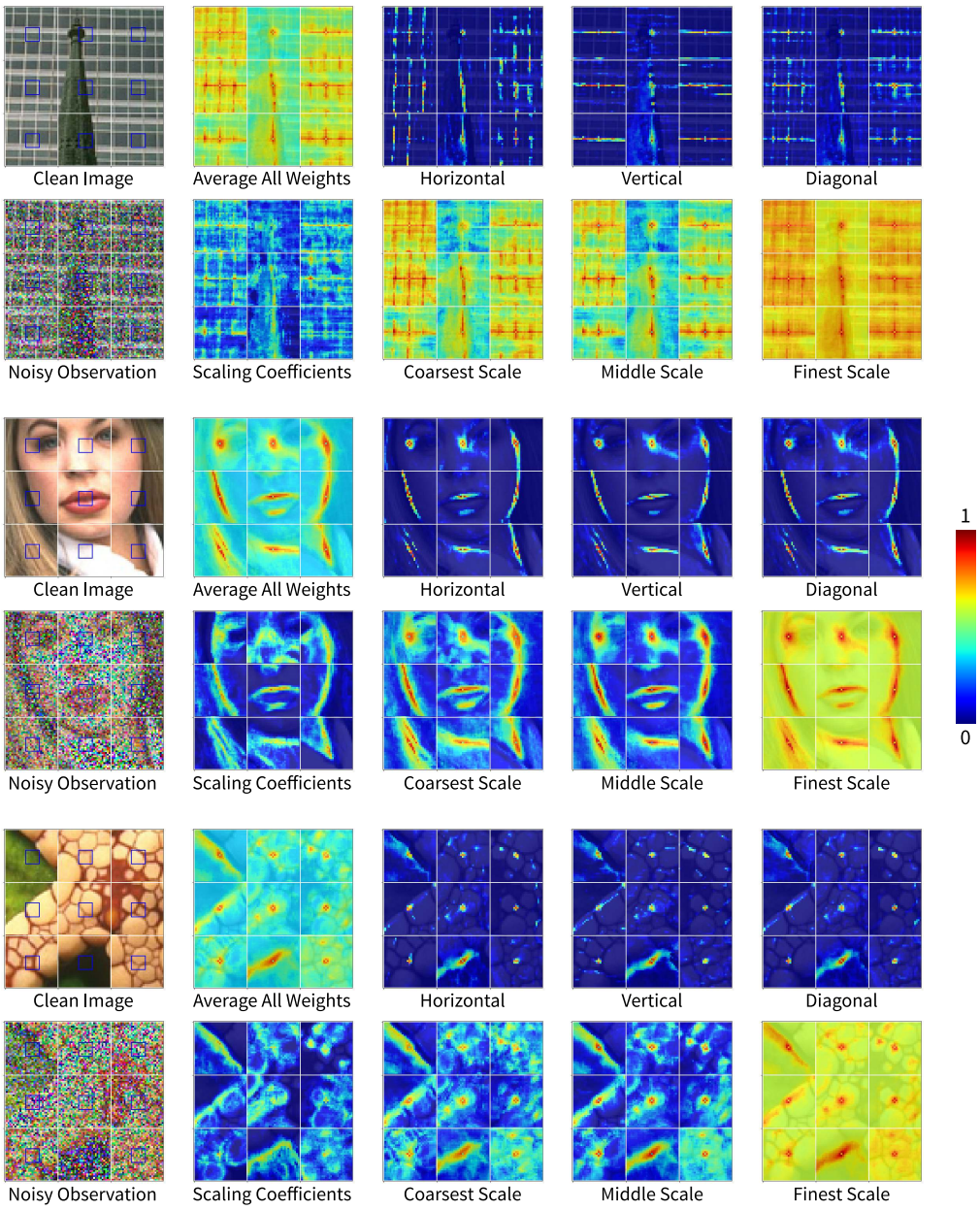


Figure 5. Visualization of Matching Score Distributions in Different Sub-bands. We show reference patches (indicated by blue squares) along with their local search windows from various images of the training set (9 windows per image), and visualize the matching scores predicted by our network. We show the predicted weights averaged across all sub-bands, as well as specific to different scales (averaging over color and orientation at each scale), and orientations (averaging over color and scale).

patterns in an observed noisy image. Our network provided a fine-grained characterization of similarity, in terms of separate scores for different corresponding sub-band components, and thus enabled the recovery of high-quality denoised estimates. We also showed that our network is especially useful in regimes where training data is scarce, being able to achieve relatively higher performance from training on a small number of examples than standard regression-based methods. A natural direction of future work lies in exploring applications of our approach, of characterizing sub-band level self-similarity, to other image-like signals such as depth maps and motion-fields.

**Acknowledgments.** This work was supported by the National Science Foundation under award no. IIS-1820693.

## References

[1] S. A. Bigdeli, M. Zwicker, P. Favaro, and M. Jin. Deep mean-shift priors for image restoration. In *Advances in Neural In-*

*formation Processing Systems*, 2017.

[2] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Proc. CVPR*, 2005.

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? In *Proc. CVPR*, 2012.

[4] J.-H. R. Chang, C.-L. Li, B. Poczos, B. V. Kumar, and A. C. Sankaranarayanan. One network to solve them all-solving linear inverse problems using deep projection models. In *Proc. ICCV*, 2017.

[5] C. Chen, Z. Xiong, X. Tian, and F. Wu. Deep boosting for image denoising. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[6] Y. Chen and T. Pock. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *PAMI*, 2017.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. In *Proc. ICIP*, 2007.

[8] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 2007.

[9] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 2006.

[10] R. Franzen. Kodak lossless true color image suite. *source: http://r0k. us/graphics/kodak*, 4, 1999.

[11] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proc. CVPR*, 2017.

[12] J.-B. Huang, A. Singh, and N. Ahuja. Single image super-resolution from transformed self-exemplars. In *Proc. CVPR*, 2015.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] B. Kumar, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *Proc. CVPR*, 2016.

[15] S. Lefkimmiatis. Non-local color image denoising with convolutional neural networks. *Proc. CVPR*, 2017.

[16] D. Liu, B. Wen, Y. Fan, C. C. Loy, and T. S. Huang. Non-local recurrent network for image restoration. In *Advances in Neural Information Processing Systems*, 2018.

[17] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang. Waterloo exploration database: New challenges for image quality assessment models. *IEEE Transactions on Image Processing*, 2017.

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. ICCV*, 2001.

[19] I. Mosseri, M. Zontak, and M. Irani. Combining the power of internal and external denoising. In *Proc. ICCP*, 2013.

[20] T. Plötz and S. Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems*, 2018.

[21] S. Roth and M. J. Black. Fields of experts. *IJCV*, 2009.

[22] U. Schmidt and S. Roth. Shrinkage fields for effective image restoration. In *Proc. CVPR*, 2014.

[23] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013.

[24] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Orocessing*, 2004.

[25] D. Yang and J. Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 2018.

[26] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proc. CVPR*, 2015.

[27] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 2016.

[28] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[29] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *Proc. CVPR*, 2017.

[30] K. Zhang, W. Zuo, and L. Zhang. Ffdnet: Toward a fast and flexible solution for cnn based image denoising. *IEEE Transactions on Image Processing*, 2018.

[31] L. Zhang, X. Wu, A. Buades, and X. Li. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic imaging*, 20(2):023016, 2011.

[32] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proc. ICCV*, 2011.