Robust and interpretable blind image denoising via bias-free convolutional neural networks

Sreyas Mohan*
Center for Data Science
New York University
sm7582@nyu.edu

Zahra Kadkhodaie*
Center for Data Science
New York University
zk388@nyu.edu

Eero P. Simoncelli
Center for Neural Science, and
Howard Hughes Medical Institute
New York University
eero.simoncelli@nyu.edu

Carlos Fernandez-Granda
Center for Data Science, and
Courant Inst. of Mathematical Sciences
New York University
cfgranda@cims.nyu.edu

Abstract

Deep convolutional networks often append additive constant ("bias") terms to their convolution operations, enabling a richer repertoire of functional mappings. Biases are also used to facilitate training, by subtracting mean response over batches of training images (a component of "batch normalization"). Recent state-of-the-art blind denoising methods (e.g., DnCNN) seem to require these terms for their success. Here, however, we show that these networks systematically overfit the noise levels for which they are trained: when deployed at noise levels outside the training range, performance degrades dramatically. In contrast, a bias-free architecture – obtained by removing the constant terms in every layer of the network, including those used for batch normalization– generalizes robustly across noise levels, while preserving state-of-the-art performance within the training range. Locally, the bias-free network acts linearly on the noisy image, enabling direct analysis of network behavior via standard linear-algebraic tools. These analyses provide interpretations of network functionality in terms of nonlinear adaptive filtering, and projection onto a union of low-dimensional subspaces, connecting the learning-based method to more traditional denoising methodology.

1 Introduction

The problem of denoising consists of estimating a signal from measurements corrupted by noise, and is a canonical application of statistical estimation that has been studied since the 1950's. Achieving high-quality denoising results requires (at least implicitly) quantifying and exploiting the differences between signals and noise. In the case of natural photographic images, the denoising problem is both an important application, as well as a useful test-bed for our understanding of natural images.

The classical solution to the denoising problem is the Wiener filter [24], which assumes a translation-invariant Gaussian signal model. Under this prior, the Wiener filter is the optimal estimator in terms of mean squared error. It operates by mapping the noisy image to the frequency domain, shrinking the amplitude of all components, and mapping back to the signal domain. In the case of natural images, the high-frequency components are shrunk more aggressively than the lower-frequency components because they tend to contain less energy. This is equivalent to convolution with a lowpass filter, implying that each pixel is replaced with a weighted average over a local neighborhood.

*Equal contribution.

Preprint. Under review.

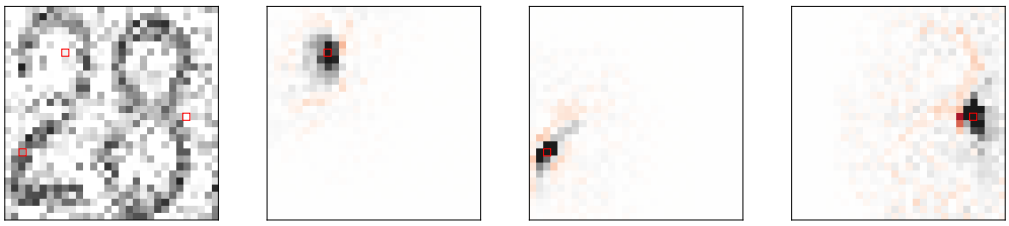


Figure 1: Visualization of the linear weighting functions (rows of A_y in equation (2)) of a BF-CNN for three example pixels of a noisy input image (left). The three images on the right show the linear weighting functions corresponding to each of the indicated pixels (red squares). All weighting functions sum to one, and thus compute a local average (although some weights are negative, indicated in red). Their shapes vary substantially, and are adapted to the underlying image content.

In the 1990's, more powerful solutions were developed based on multi-scale ("wavelet") transforms. These transforms map natural images to a domain where they have sparser representations. This makes it possible to perform denoising by applying nonlinear thresholding operations in order to discard components that are small relative to the noise level [5, 20, 1]. From a linear-algebraic perspective, these algorithms operate by projecting the noisy input onto a lower-dimensional subspace that contains plausible signal content. The projection eliminates the orthogonal complement of the subspace, which mostly contains noise. This general methodology laid the foundations for the state-of-the-art models in the 2000's (e.g. [4]), some of which added a data-driven perspective, learning sparsifying transforms [6], and nonlinear shrinkage functions directly from natural images [7, 17].

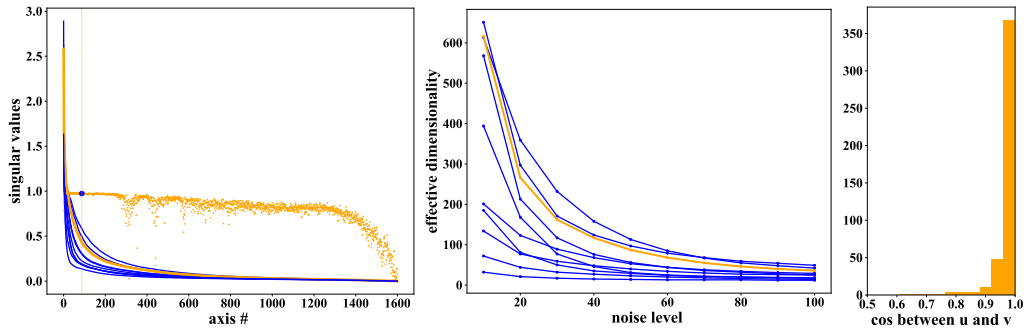
In the past decade, purely data-driven models based on convolutional neural networks [11] have come to dominate all previous methods in terms of performance. These models consist of cascades of convolutional filters, and rectifying nonlinearities, which are capable of representing a diverse and powerful set of functions. Training such architectures to minimize mean square error over large databases of noisy natural-image patches achieves current state-of-the-art results [25] (see also [2] for a related approach).

Neural networks have achieved particularly impressive results on the *blind* denoising problem, in which the noise amplitude is unknown [25, 26, 12]. Despite their success, these solutions are not well understood. We lack intuition about the denoising mechanisms they implement. Network architecture and functional units are often borrowed from the image-recognition literature, and it is unclear which of these aspects contribute positively, or limit, the denoising performance. Many authors claim critical importance of specific aspects of architecture (e.g., skip connections, batch normalization, recurrence), but the benefits of these attributes are difficult to isolate and evaluate in the context of the many other elements of the system.

In this work, we show that bias-free CNNs (BF-CNNs), in which the bias terms (additive constants) throughout the network have been eliminated, yield performance matching the state-of-the-art in image denoising, while offering two important advantages. First, BF-CNNs are locally linear, and hence amenable to direct analysis with linear-algebraic tools. In Section 2 we leverage such tools to visualize locally adaptive properties of the denoising map, and to show that the network approximates a projection onto an adaptively-selected low-dimensional subspace. The analysis uncovers direct connections between the denoising mechanism of the neural networks and classical methods based on filtering and projection operations. Second, we show that BF-CNNs generalize robustly to noise levels well beyond the range at which they have been trained, in stark contrast to existing architectures. Section 3.1 provides a theoretical explanation of this phenomenon, based on the insight that removing the bias results in invariance to rescaling. Section 3.2 demonstrates the overall performance and generalization capabilities of BF-CNNs through comprehensive numerical experiments, showing that the advantages of bias removal hold for several popular architectures.

2 Analysis of bias-free neural networks for denoising

We assume a measurement model in which images are corrupted by additive noise: $y = x + n$, where $x \in \mathbb{R}^N$ is the original image, containing N pixels, n is an image of i.i.d. samples of Gaussian



(a) (b) (c)

Figure 2: Analysis of the SVD of the Jacobian of a BF-CNN for ten natural images, corrupted by noise of standard deviation $\sigma = 50$. **(a)** Singular value distributions. For all images, a large proportion of the values are near zero, indicating (approximately) a projection onto a subspace (the *signal subspace*). The orange points show the cosine of the angle between the left and right singular vectors, for a single image (orange curve). Larger blue point (and vertical line) indicates the effective dimensionality (sum of squared singular values) for that image. Note that orange points at dimensionalities less than this are nearly 1, implying that the corresponding left and right singular vectors are nearly identical. **(b)** Effective dimensionality of the signal subspaces as a function of noise level (orange curve for same image highlighted inf panel (a)). For comparison, the total dimensionality of the space is 1600 (40×40 pixels). **(c)** Histogram of dot products (cosine of angle) between the left and right singular vectors that lie within the signal subspaces.

noise with variance σ^2, and y is the noisy observation. The denoising problem consists of finding a function $f : \mathbb{R}^N \to \mathbb{R}^N$ that provides a good estimate of the original image, x. Commonly, one minimizes the mean squared error : $f(y) = \arg\min_f E||x - f(y)||^2$, where the expectation is taken over some distribution over images, x, as well as over the distribution of noise realizations. Finally, if the noise standard deviation, σ, is unknown, the expectation must also be taken over a distribution of this variable. This problem is often called *blind denoising* in the literature.

Feedforward neural networks with rectified linear units (ReLUs) are piecewise affine: for a given activation pattern of the ReLUs, the effect of the network on the input is a cascade of linear transformations (convolutional or fully connected layers), additive constants, and pointwise multiplications by a binary mask corresponding to the fixed activation pattern. Since each of these is affine, the entire cascade implements a single affine transformation. For a fixed noisy input image $y \in \mathbb{R}^N$ with N pixels, the function $f : \mathbb{R}^N \to \mathbb{R}^N$ computed by a denoising neural network may be written

$$f(y) = A_y y + b_y, \tag{1}$$

where $A_y \in \mathbb{R}^{N \times N}$ is the Jacobian of $f(\cdot)$ evaluated at input $y \in \mathbb{R}^N$, and $b_y \in \mathbb{R}^N$ represents the net bias. The subscripts on A_y and b_y serve as a reminder that the corresponding matrix and vector, respectively, depend on the ReLU activation patterns, which in turn depend on the input vector y.

If we remove all the additive ("bias") terms from every stage of a CNN, the resulting bias-free CNN (BF-CNN) is strictly linear, and its net action may be expressed as

$$f_{\mathrm{BF}}(y) = A_y y, \tag{2}$$

where A_y is again the Jacobian of $f_{\mathrm{BF}}(\cdot)$ evaluated at y. In the following two sections, we analyze this local representation to reveal and visualize the noise-removal mechanisms implemented by BF-CNNs. We illustrate our analysis using a BF-CNN based on the architecture of the Denoising CNN (DnCNN, [25]). A detailed description of the architecture is provided in Section 3.2.

Visualization approaches based on differentiating neural-network functions with respect to their input have been previously proposed in the context of image classification (see e.g. [21, 15]). However, architectures used for classification include non-ReLU activation functions, and thus the Jacobian serves only to provide a 1st-order Taylor approximation of the mapping. A similar complication arises when considering denoising architectures that are not bias-free: The resulting filters do not fully represent the denoising map due to the presence of the net bias b_y, which confounds the analysis.

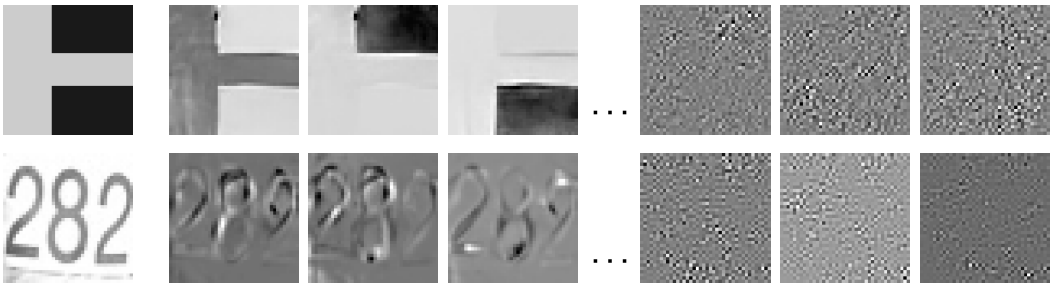


Figure 3: Visualization of left singular vectors of the Jacobian of a BF-CNN, evaluated on two different images (top and bottom rows), corrupted by noise with standard deviation $\sigma = 50$. The left column shows original (clean) images. The next three columns show singular vectors corresponding to non-negligible singular values. The vectors capture features from the clean image. The last three columns on the right show singular vectors corresponding to singular values that are almost equal to zero. These vectors are noisy and unstructured.

2.1 Visualization of equivalent filters

The linear representation of the denoising map given by equation (2) implies that the ith pixel of the output image is computed as an inner product between the ith row of A_y, denoted $a_y(i)$, and the input image:

$$f_{\text{BF}}(y)(i) = \sum_{j=1}^{N} A_y(i,j)y(j) = a_y(i)^T y. \tag{3}$$

The vectors $a_y(i)$ can be interpreted as *adaptive filters* that produce an estimate of the denoised pixel via a weighted average of noisy pixels. Examination of these filters reveals their diversity, and their relationship to the underlying image content (Figure 1): They are adapted to the local features of the noisy image, averaging over homogeneous regions of the image without blurring across edges.

Analyzing BF-CNNs in terms of equivalent filters facilitates a comparison with existing denoising methods. As mentioned in the introduction, classical Wiener filtering denoises images by convolving with a lowpass filter, preserving low-frequency information, while suppressing fine-scale details. Many modern denoising techniques can be interpreted as implementing nonlinear spatially-varying filters designed to preserve fine-scale details such as edges (e.g. [22], see also [14] for a comprehensive review, and [3] for a recent learning-based approach). Our analysis shows that BF-CNNs can be interpreted as a data-driven method to learn adaptive filters implicitly.

2.2 Analysis via the singular-value decomposition

The local linear structure of a BF-CNN facilitates analysis of its functional capabilities via the singular value decomposition (SVD). For a given input y, we compute the SVD of the Jacobian matrix: $A_y = USV^T$, with U and V orthogonal matrices, and S a diagonal matrix. We can decompose the effect of the network on its input in terms of the left singular vectors $\{U_1, U_2 \ldots, U_N\}$ (columns of U), the singular values $\{s_1, s_2 \ldots, s_N\}$ (diagonal elements of S), and the right singular vectors $\{V_1, V_2, \ldots V_N\}$ (columns of V):

$$f_{\text{BF}}(y) = A_y y = USV^T y = \sum_{i=1}^{N} s_i (V_i^T y) U_i. \tag{4}$$

The output is a linear combination of the left singular vectors, each weighted by the projection of the input onto the corresponding right singular vector, and scaled by the corresponding singular value.

Analyzing the SVD of a BF-CNN on a set of ten natural images reveals that most singular values are very close to zero (Figure 2a). The network is thus discarding all but a very low-dimensional portion of the input image. We can measure an "effective dimensionality" of this preserved subspace by computing the total noise variance remaining in the denoised image, $f_{\text{BF}}(y)$, which corresponds to the sum of the squared singular values.

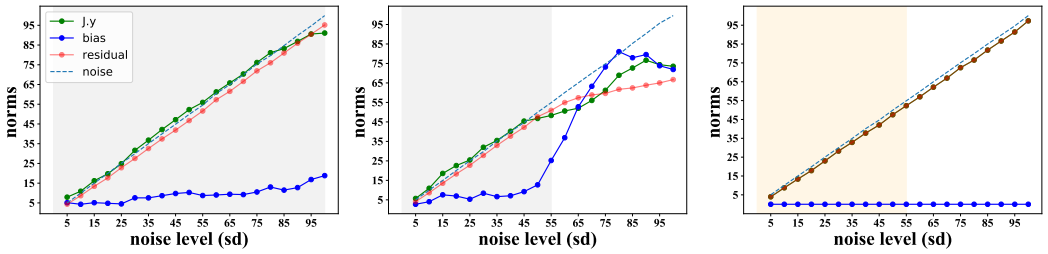


Figure 4: Analysis of bias and linear terms as a function of noise level. For a noisy image y and a denoising neural network f, we write the residual as $y - f(y) = (I - A_y)y - b_y$ (see equation (1)). The plots show the magnitudes of the residual, the linear term $(I - A_y)y$ and the net bias b_y averaged over 100 20×20 natural-image patches for networks calibrated over different training ranges. **(a)** DnCNN network, trained over the full range of noise levels ($\sigma \in [0, 100]$). The net bias is small, and grows gradually as the noise increases. **(b)** DnCNN network, trained over the range $\sigma \in [0, 55]$ (gray band). The net bias grows explosively for noise levels outside the training range. **(c)** BF-CNN network, trained over range $\sigma \in [0, 55]$ (beige band). The net bias is zero by design.

We also observe that the left and right singular vectors corresponding to the singular values with non-negligible amplitudes are approximately the same (Figures 2a and 2c). This means that the Jacobian is (approximately) symmetric, and we can interpret the action of the network as projecting the noisy signal onto a low-dimensional subspace, as is done in wavelet thresholding schemes.

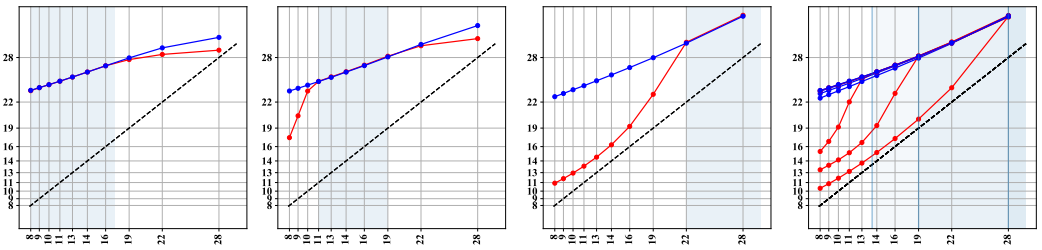
For inputs of the form $y := x + n$ (where x is the clean image and n the noise), the subspace spanned by the singular vectors corresponding to the non-negligible singular values contains x almost entirely, in the sense that projecting x onto the subspace preserves most of its energy. This holds for the whole range of noise levels over which the network is trained. Specifically, averaged over 50 example images, the fraction of the energy (squared ℓ_2 norm) preserved by the projection falls from 0.999 to 0.986 as the standard deviation of the noise n grows from 10 to 100 (relative to the image pixels, which lie in the range $[0, 255]$). The low-dimensional subspace encoded by the Jacobian is therefore tailored to the input image. This is confirmed by visualizing the singular vectors as images. The singular vectors corresponding to non-negligible singular values capture features of the input image; the ones corresponding to near-zero singular values are unstructured (Figure 3). The BF-CNN therefore implements an approximate projection onto an adaptive *signal subspace* that preserves image structure, while suppressing the noise.

The signal subspace depends on the noise level. We find that for a given clean image corrupted by noise, the effective dimensionality of the signal subspace decreases as the noise level increases (Figure 2b). In addition, the signal subspaces are nested: the subspaces corresponding to lower noise levels contain at least 95% of the subspaces corresponding to higher noise levels. At lower noise levels the network detects a richer set of image features, and constructs a larger signal subspace.

In conclusion, the SVD analysis of the BF-CNN reveals that the network learns a prior on natural images that is structured as a union of subspaces. This is reminiscent of sparsity-based techniques, in which images are assumed to lie in unions of subspaces spanned by sparse linear combinations of basis functions, which may be handcrafted (e.g., [1, 16]) or learned (e.g., [6]). The BF-CNN implements the prior implicitly: the subspaces are obtained through a concatenation of weight matrices governed by the activation patterns corresponding to specific inputs.

3 Generalization across noise levels

In order to apply learning-based denoising methods on real data, it is crucial to understand their generalization properties. In particular, methods should be robust to deviations between the data used for training and the data encountered at test time. Here, we take a step in this direction by studying generalization across noise levels, i.e. denoising performance for noise levels not included in the training set. We provide theoretical analysis and computational experiments showing that BF-CNNs generalize robustly across noise levels, in stark contrast to architectures that have nonzero net bias.



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

Figure 5: Comparisons of the performance of DnCNN (red curves) and BF-CNN (blue curves) for the experimental design described in Section 3.2. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR. The black dashed line depicts the identity function, for comparison. **(a-c)** The two networks are trained over a fixed ranges of noise levels indicated by a gray background. In all cases, the performance of DnCNN degrades significantly beyond the training range. In contrast, BF-CNN generalizes robustly. **(d)** Superposition of results from networks trained on three different low-noise ranges. All training ranges share the same maximum PSNR (corresponding to $\sigma = 10$). The vertical gray lines mark the different values of the minimum PSNR. In all cases, BF-CNN generalizes well, in marked contrast to DnCNN.

3.1 Scaling invariance

A key property of bias-free neural networks with ReLU activations is *invariance to scaling*: rescaling the input by a constant value simply rescales the output by the same amount.

Lemma 1. *Let $f_{\mathrm{BF}} : \mathbb{R}^N \to \mathbb{R}^N$ be a feedforward neural network with ReLU activation functions and no constant terms in any layer. For any input $y \in \mathbb{R}$ and any nonnegative constant α,*

$$f_{\mathrm{BF}}(\alpha y) = \alpha f_{\mathrm{BF}}(y). \tag{5}$$

Proof. We can write the action of a bias-free neural network with L layers in terms of the weight matrix W_i, $1 \leq i \leq L$, of each layer and a rectifying operator $\mathcal{R}$, which sets to zero any negative entries in its input. Multiplying by a nonnegative constant does not change the sign of the entries of a vector, so for any z with the right dimension and any $\alpha > 0$ $\mathcal{R}(\alpha z) = \alpha \mathcal{R}(z)$, which implies

$$f_{\mathrm{BF}}(\alpha y) = W_L \mathcal{R}(W_{L-1} \cdots \mathcal{R}(W_1 \alpha y)) = \alpha W_L \mathcal{R}(W_{L-1} \cdots \mathcal{R}(W_1 y)) = \alpha f_{\mathrm{BF}}(y). \tag{6}$$

□

The proof breaks down if any of the layers in the network contains additive constant parameters because scaling the input may change the activation pattern of the ReLUs. Networks with a nonzero net bias are not scaling invariant.

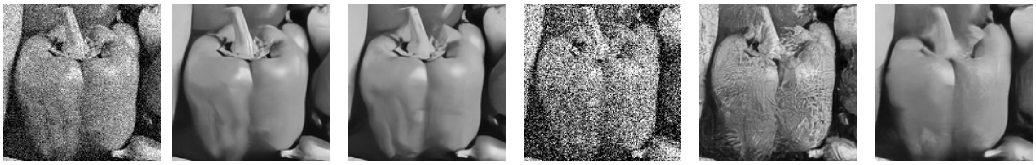
Scaling invariance renders BF-CNNs robust to varying noise levels. Consider a clean image x and a noise realization n corresponding to a noise level within the training range. Let *err* denote the error of a bias-free network when estimating x from $x + n$,

$$f_{\mathrm{BF}}(x + n) = x + \text{err}. \tag{7}$$

By Lemma 1, the network is automatically able to denoise clean images at a different noise level, as long as they are scaled accordingly. For any $\alpha > 0$, the error of the network when denoising an image $x_2 := \alpha x$ corrupted by a noise realization αn scales linearly with α,

$$f(x_2 + \alpha n) = x_2 + \alpha \,\text{err}. \tag{8}$$

A bias-free network therefore generalizes across new noise levels, as long as examples with the same signal-to-noise ratio are present in the training data. Note that this argument applies to image patches of size equal to the field-of-view of the neural network. This theoretical analysis shows that bias-free networks should provide robust generalization across noise levels when trained on a collection of image patches with different intensities, even if the range of noise levels is limited. This is confirmed by the numerical experiments reported in the following section.



(a) (b) (c) (d) (e) (f)

Figure 6: Denoising of an example natural image from the test set described in Section 3.2. The image is denoised by applying a DnCNN and a BF-CNN trained over the range $\sigma \in [0, 55]$. **(a)** Noisy image, with intensities in range $[0, 255]$, and noise standard deviation $\sigma = 30$. **(b)** Result of applying the DnCNN to (a). **(c)** Result of applying the BF-CNN to (a). **(d)** Noisy image for $\sigma = 70$, which lies outside the training range. **(e)** Result of applying the DnCNN to the image in (d). **(f)** Result of applying the BF-CNN to the image in (d).

3.2 Computational experiments

In this section, we investigate generalization across noise levels computationally, comparing networks with and without net bias. In order to do this, we train the networks using images corrupted by i.i.d. Gaussian noise with a range of standard deviations. This range is the *training range* of the network. We then evaluate the network for noise levels that are both within and beyond the training range. Our experiments are carried out on 180×180 natural images from the Berkeley Segmentation Dataset [13]. To make our experiments consistent with previous results including [19, 2, 25], we use a subset of 400 images as a training set. The training set is augmented via downsampling, random flips, and random rotations of patches in these images to produce a total of 541,600 patches [25]. We use a separate set of 68 images as the test set.

We implement a BF-CNN based on the architecture of the Denoising CNN (DnCNN) [25], a neural network with 20 convolutional layers, each containing 3×3 filters and 64 channels. Each intermediate layer in a DnCNN applies a convolution followed by batch normalization [9] and a ReLU. There is a skip connection from the initial layer to the final layer, which has no nonlinear units. To construct a BF-CNN from the DnCNN, we remove all sources of additive bias, including the mean parameter of the batch-normalization in every layer (note however that the rescaling parameters are preserved). We train the BF-CNN, and a DnCNN, on natural image patches corrupted by i.i.d. Gaussian noise with varying standard deviation σ, sampled uniformly at random from a training range specified by minimum and maximum standard deviations, $[\sigma_{\text{min}}, \sigma_{\text{max}}]$. The architectures are trained by minimizing the mean squared denoising error $\sum_i ||f(y_i) - x_i||_2^2$ over the entire training set. Here (x_i, y_i) denotes the ith training pair of clean and noisy images. The loss is minimized using the Adam Optimizer [10] over 70 epochs with an initial learning rate of 10^{-3} and a decay factor of 0.5 at the 50^{th} and 60^{th} epochs.

We begin by analyzing the affine local model of a DnCNN trained over different noise levels. In particular, for a neural network f we write the residual corresponding to a fixed noisy image y as $y - f(y) = (I - A_y)y - b_y$ (see equation (1)). Figure 4 shows the magnitudes of the residual, the linear term, $(I - A_y)y$, and the net bias, b_y, as a function of noise level. Over the training range, the linear term dominates, implying that it is responsible for most of the denoising effort. The net bias is significantly smaller, although it does grow systematically with the noise level. However, when the network is evaluated out of the training range, the norm of the bias increases dramatically. The strange behavior of the bias term out of the training range coincides with a substantial drop in denoising performance. Figure 5 compares DnCNN and BF-CNN for different noise levels, inside and outside of the training range. Performance is quantified in peak signal-to-noise-ratio (PSNR). In all cases, DnCNN generalizes very poorly to noise levels outside the training range. In contrast, BF-CNN generalizes robustly, as predicted by the theoretical argument in Section 3.1. Section A shows that the same holds for the more perceptually-meaningful Structural Similarity Index (SSIM) [23]. Figure 6 shows an example that demonstrates visually the striking difference in generalization performance. The presence of a net bias in the architecture results in severe overfitting to the training range.

The theoretical analysis in Section 3.1 holds for any neural network architecture that combines convolutional layers and ReLUs. This suggests that the overfitting phenomenon observed for DnCNN is likely to be true of other networks as well. In order to investigate this, we trained several different

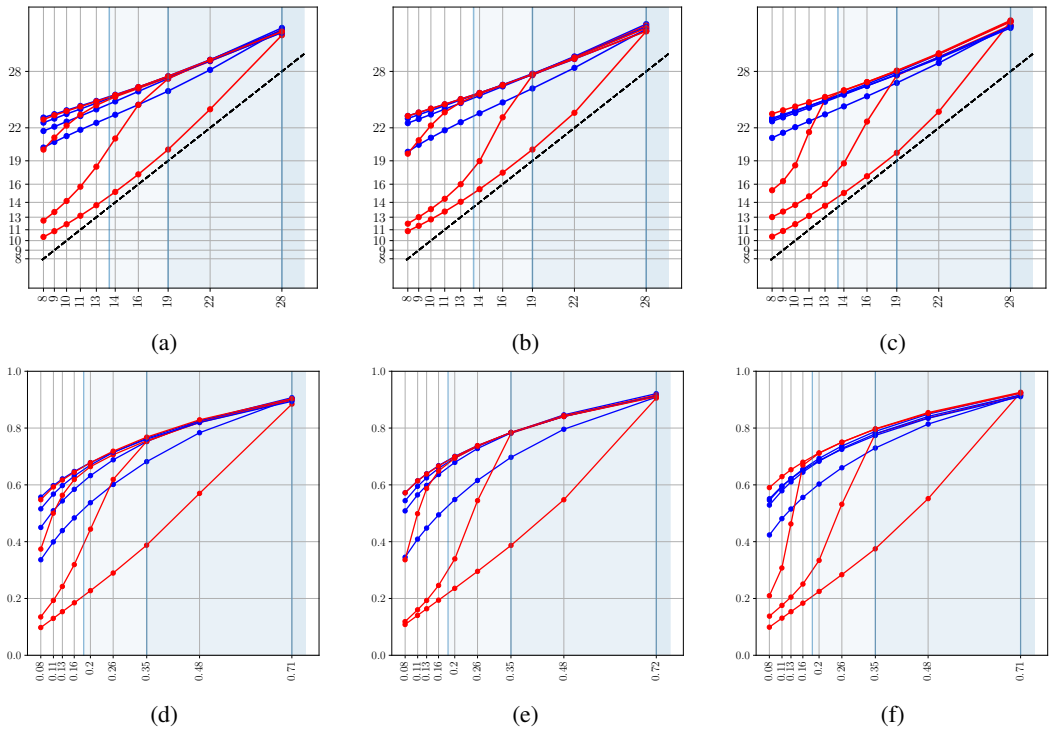


Figure 7: Comparisons of architectures with (red curves) and without (blue curves) a net bias for the experimental design described in Section 3.2. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR **(a, b, c)** and SSIM of the denoised image as a function of the input SSIM **(d, e, f)**. All the architectures with bias perform poorly out of their training range, whereas the bias-free versions all achieve excellent generalization across noise levels. **(a, d)** Recurrent architecture inspired by DURR [26]. **(b, e)** Multiscale architecture inspired by the UNet [18]. **(c, f)** Architecture with multiple skip connections inspired by the DenseNet [8].

CNN architecture with and without net biases. These architectures include popular features of existing neural-network techniques in image processing: recurrence, multiscale filters, and skip connections.

The first architecture is a recurrent neural network, where the basic module is a CNN with 5 layers, 3×3 filters and 64 channels in the intermediate layers (see Section B.1). The experiment is inspired by Ref. [26], in which the authors argue that recurrence is responsible for improved generalization across noise levels. Our results are inconsistent with this hypothesis (Figure 7b). The recurrent architecture fails to generalize if it has a net bias, but generalizes robustly when this bias is removed, both in terms of PSNR and SSIM. The second network is an example of a popular multiscale architecture known as a UNet [18] (see Section B.2). Figure 7b shows that generalization is again governed by the presence or absence of a net bias. Finally, Figure 7c shows that the same phenomenon occurs for a network inspired by the DenseNet architecture [8, 27]. The network consists of 4 consecutive CNN modules with 5 layers, 3×3 filters and 64 channels, where there is a skip connection from the input to the end of each module (see Section B.3). These experiments suggest that the presence of a net bias is the primary impediment to denoising generalization for CNN architectures.

4 Discussion

In this work, we show that removing constant terms from CNN architectures boosts generalization performance across noise levels, and also provides interpretability of the denoising method via linear-algebra techniques, such as the SVD. Our linear-algebraic analysis uncovers interesting aspects of the denoising map, but these interpretations are very local: small changes in the input image change the activation patterns of the network, resulting in a change in the corresponding linear mapping. Extending the analysis to reveal global characteristics of the neural-network functionality

is a challenging direction for future research. It is also of interest to examine whether bias removal can facilitate generalization in other image-processing tasks, such as single-image superresolution.

References

[1] CHANG, S. G., YU, B., AND VETTERLI, M. Adaptive wavelet thresholding for image denoising and compression. *IEEE Trans. Image Processing 9*, 9 (2000), 1532–1546.

[2] CHEN, Y., AND POCK, T. Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration. *IEEE Trans. Patt. Analysis and Machine Intelligence 39*, 6 (2017), 1256–1272.

[3] CHOI, S., ISIDORO, J., GETREUER, P., AND MILANFAR, P. Fast, trainable, multiscale denoising. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018), IEEE, pp. 963–967.

[4] DABOV, K., FOI, A., KATKOVNIK, V., AND EGIAZARIAN, K. Image denoising with block-matching and 3d filtering. In *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning* (2006), vol. 6064, International Society for Optics and Photonics, p. 606414.

[5] DONOHO, D., AND JOHNSTONE, I. Adapting to unknown smoothness via wavelet shrinkage. *J American Stat Assoc 90*, 432 (December 1995).

[6] ELAD, M., AND AHARON, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. on Image processing 15*, 12 (2006), 3736–3745.

[7] HEL-OR, Y., AND SHAKED, D. A discriminative approach for wavelet denoising. *IEEE Trans. Image Processing* (2008).

[8] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition* (2017), pp. 4700–4708.

[9] IOFFE, S., AND SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[10] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] LECUN, Y., BENGIO, Y., AND HINTON, G. Deep learning. *nature 521*, 7553 (2015), 436.

[12] LEFKIMMIATIS, S. Universal denoising networks: a novel cnn architecture for image denoising. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 3204–3213.

[13] MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision* (July 2001), vol. 2, pp. 416–423.

[14] MILANFAR, P. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE signal processing magazine 30*, 1 (2012), 106–128.

[15] MONTAVON, G., LAPUSCHKIN, S., BINDER, A., SAMEK, W., AND MÜLLER, K.-R. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Rec. 65* (2017), 211–222.

[16] PORTILLA, J., STRELA, V., WAINWRIGHT, M. J., AND SIMONCELLI, E. P. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Image Processing 12*, 11 (2003).

[17] RAPHAN, M., AND SIMONCELLI, E. P. Optimal denoising in redundant representations. *IEEE Trans Image Processing 17*, 8 (Aug 2008), 1342–1352.

[18] RONNEBERGER, O., FISCHER, P., AND BROX, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[19] SCHMIDT, U., AND ROTH, S. Shrinkage fields for effective image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2774–2781.

[20] SIMONCELLI, E. P., AND ADELSON, E. H. Noise removal via Bayesian wavelet coring. In *Proc 3rd IEEE Int'l Conf on Image Proc* (Lausanne, Sep 16-19 1996), vol. I, IEEE Sig Proc Society, pp. 379–382.

[21] SIMONYAN, K., VEDALDI, A., AND ZISSERMAN, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[22] TOMASI, C., AND MANDUCHI, R. Bilateral filtering for gray and color images. In *ICCV*, vol. 98.

[23] WANG, Z., BOVIK, A. C., SHEIKH, H. R., SIMONCELLI, E. P., ET AL. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Processing 13*, 4 (2004), 600–612.

[24] WIENER, N. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Technology Press, 1950.

[25] ZHANG, K., ZUO, W., CHEN, Y., MENG, D., AND ZHANG, L. Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Trans. Image Processing 26*, 7 (2017), 3142–3155.

[26] ZHANG, X., LU, Y., LIU, J., AND DONG, B. Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration. *arXiv preprint arXiv:1805.07709* (2018).

[27] ZHANG, Y., TIAN, Y., KONG, Y., ZHONG, B., AND FU, Y. Residual dense network for image restoration. *CoRR abs/1812.10477* (2018).

Appendix A Generalization performance measured by SSIM

Figure 8 compares the performance of DnCNN (red curves) and BF-CNN (blue curves) for the experimental design described in Section 3.2 of the main paper. The performance is quantified by the Structural Similarity Index (SSIM) of the denoised image as a function of the input SSIM.

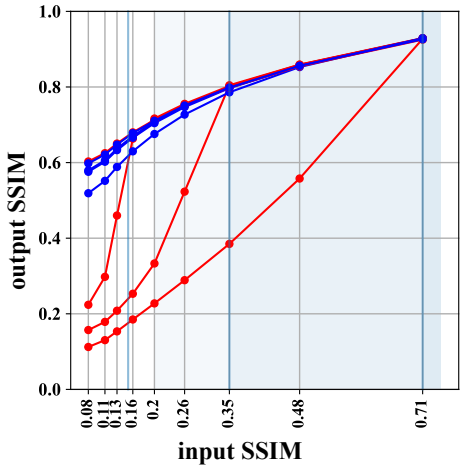


Figure 8: Comparisons of the performance of DnCNN (red curves) and BF-CNN (blue curves) for the experimental design described in Section 3.2. The performance is quantified by the SSIM of the denoised image as a function of the input SSIM. The graph shows a superposition of results from networks trained on three different ranges (indicated by the shaded regions). The training ranges share the same maximum SSIM (corresponding to the noise level $\sigma = 10$), the minimum SSIM in the training range is indicated by a blue line. In all cases, BF-CNN generalizes well, in marked contrast to DnCNN.

Appendix B Experiments with different architectures

In this section we describe the computational experiments used to evaluate the performance of the recursive architecture, the UNet, and DenseNet-style architecture in Section 3.2. We train these models using a smaller version of the dataset described in Section 3.2 of the main paper. In more detail, we use patches of size 128×128 instead of 50×50, which yields a total of 22,400 training patches. We train the models using the Adam optimizer [10] with an initial learning rate of 10^{-3} and train for 50 epochs with a learning rate schedule which decreases by a factor of 0.25 if the validation PSNR decreases from one epoch to the next. We use early stopping and select the model with the best validation PSNR. The following subsections provide details regarding the different architectures.

B.1 Recursive Framework

Inspired by Ref. [26], we consider a recurrent framework that produces a denoised image estimate of the form $\hat{x}_t = f(\hat{x}_{t-1}, y_{\text{noisy}})$, at time t where f is a neural network. We use a 5-layer fully convolutional network with 3×3 filters in all layers and 64 channels in each intermediate layer to implement f. We initialize the denoised estimate as the noisy image, i.e $\hat{x}_0 := y_{\text{noisy}}$. For the version of the network with net bias, we add trainable additive constants to every filter in all but the last layer. During training, we run the recurrence for a maximum of T times, sampling T uniformly at random from $\{1, 2, 3, 4\}$ for each mini-batch. At test time we fix $T = 4$.

B.2 UNet

Our UNet model [18] has the following layers:

1. *conv1* - Takes in input image and maps to 32 channels with 5×5 convolutional kernels.
2. *conv2* - Input: 32 channels. Output: 32 channels. 3×3 convolutional kernels.
3. *conv3* - Input: 32 channels. Output: 64 channels. 3×3 convolutional kernels with stride **2**.
4. *conv4*- Input: 64 channels. Output: 64 channels. 3×3 convolutional kernels.
5. *conv5*- Input: 64 channels. Output: 64 channels. 3×3 convolutional kernels with dilation factor of 2.
6. *conv6*- Input: 64 channels. Output: 64 channels. 3×3 convolutional kernels with dilation factor of 4.
7. *conv7*- Transpose Convolution layer. Input: 64 channels. Output: 64 channels. 4×4 filters with stride 2.
8. *conv8*- Input: 96 channels. Output: 64 channels. 3×3 convolutional kernels. The input to this layer is the concatenation of the outputs of layer *conv7* and *conv2*.
9. *conv9*- Input: 32 channels. Output: 1 channels. 5×5 convolutional kernels.

The structure is the same as in Ref. [26], but without recurrence. For the version with bias, we add trainable additive constants to all the layers other than *conv9*. This configuration of UNet assumes even width and height, so we remove one row or column from images in with odd height or width.

B.3 Simple DenseNet

Our simplified version of the DenseNet architecture [8] has 4 blocks in total. Each block is a fully convolutional 5-layer CNN with 3×3 filters and 64 channels in the intermediate layers with ReLU nonlinearity. The first three blocks have an output layer with 64 channels while the last block has an output layer with only one channel. The output of the i^{th} block is concatenated with the input noisy image and then fed to the $(i+1)^{th}$ block, so the last three blocks have 65 input channels. In the version of the network with bias, we add trainable additive parameters to all the layers except for the last layer in the final block.