A machine learning approach for non-blind image deconvolution

Christian J. Schuler, Harold Christopher Burger, Stefan Harmeling, and Bernhard Schölkopf
Max Planck Institute for Intelligent Systems, Tübingen, Germany
{cschuler,burger,harmeling,bs}@tuebingen.mpg.de
http://webdav.is.mpg.de/pixel/neural_deconvolution/

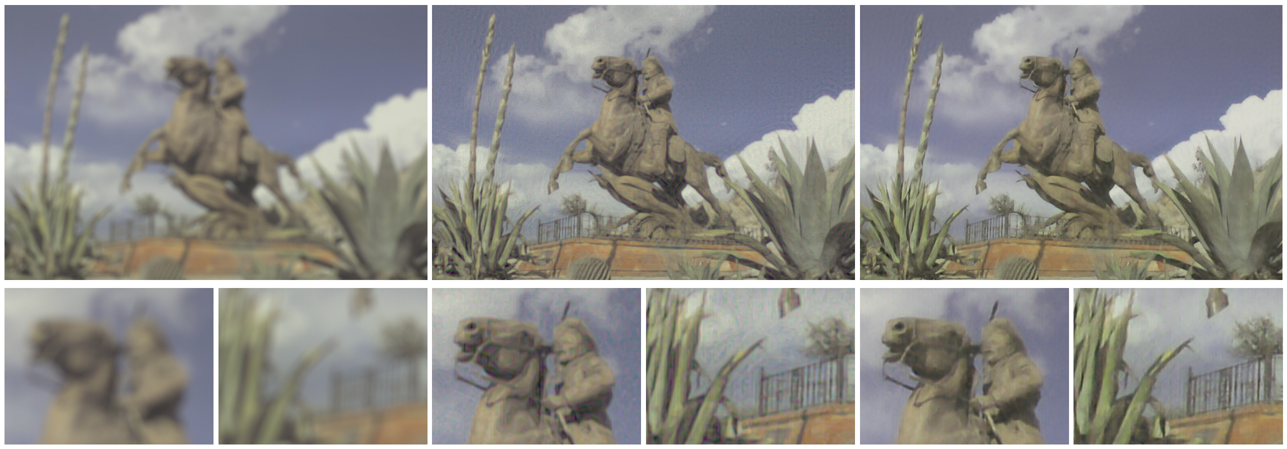


Figure 1. Removal of defocus blur in a photograph. The true PSF is approximated with a pillbox.

Abstract

Image deconvolution is the ill-posed problem of recovering a sharp image, given a blurry one generated by a convolution. In this work, we deal with space-invariant non-blind deconvolution. Currently, the most successful methods involve a regularized inversion of the blur in Fourier domain as a first step. This step amplifies and colors the noise, and corrupts the image information. In a second (and arguably more difficult) step, one then needs to remove the colored noise, typically using a cleverly engineered algorithm. However, the methods based on this two-step approach do not properly address the fact that the image information has been corrupted. In this work, we also rely on a two-step procedure, but learn the second step on a large dataset of natural images, using a neural network. We will show that this approach outperforms the current state-of-the-art on a large dataset of artificially blurred images. We demonstrate the practical applicability of our method in a real-world example with photographic out-of-focus blur.

1. Introduction

Images can be blurry for a number of reasons. For example, the camera might have moved during the time the image was captured, in which case the image is corrupted by motion blur. Another common source of blurriness is out-of-focus blur. Mathematically, the process corrupting the image is a convolution with a point-spread function (PSF). A blurry image y is given by $y = x * v + n$, where x is the true underlying (non-blurry) image, v is the point spread function (PSF) describing the blur and n is noise, usually assumed to be additive, white and Gaussian (AWG) noise. The inversion of the blurring process is called image deconvolution and is ill-posed in the presence of noise.

In this paper, we address space-invariant non-blind deconvolution, *i.e.* we want to recover x given y and v and assume v to be constant (space-invariant) over the image. Even though this is a long-standing problem, it turns out that there is room for improvement over the best existing methods. While most methods are well-engineered algorithms, we ask the question: Is it possible to automatically learn an image deconvolution procedure? We will show that this is indeed possible.

Contributions: We present an image deconvolution procedure that is *learned* on a large dataset of natural images with a multi-layer perceptron (MLP). We compare our approach to other methods on a large dataset of synthetically blurred images, and obtain state-of-the-art results for all tested blur kernels. Our method also achieves excellent results on a real photograph corrupted by out-of-focus blur. The execution time of our approach is reasonable (once trained for a specific blur) and scales linearly with the size of the image. We provide a toolbox on our website to test our method.

2. Related Work

Image deconvolution methods can be broadly separated into two classes. The first class of methods is based on probabilistic image priors, whereas the second class of methods relies on a pre-processing step followed by denoising.

Levin *et al.* [20], Krishnan *et al.* [18], EPLL [31], and FoE [26] belong to the first category. Levin *et al.*, Krishnan *et al.*, and EPLL seek a maximum a posteriori (MAP) estimate of the clean image x, given a blurry (and noisy) version y and the PSF v. In other words, one seeks to find the x maximizing $p(x|y, v) \propto p(y|x, v)p(x)$. The first term is a Gaussian likelihood, but modeling the marginal distribution of images $p(x)$ is a long-standing research problem and can be handled in a number of ways. Levin *et al.* and Krishnan *et al.* assume that the image gradients follow a hyper-Laplacian distribution (this is a common and well-founded assumption, see *e.g.* [28]). EPLL [31] models $p(x)$ using a Gaussian mixture model (GMM). FoE [26] uses a Bayesian minimum mean squared error estimate (MMSE) instead of a MAP estimate and uses the Fields of Experts [24] framework to model $p(x)$.

The second category of methods apply a regularized inversion of the blur, followed by a denoising procedure. In Fourier domain, the inversion of the blur can be seen as a pointwise division by the blur kernel. This makes the image sharper, but also has the effect of amplifying the noise, as well as creating correlations in the noise, see Figure 2. Hence, these methods address deconvolution as a *denoising* problem. Unfortunately, most denoising methods are designed to remove AWG noise [23, 12, 9]. Deconvolution via denoising requires the denoising algorithm to be able to remove colored noise (non-flat power spectrum of the noise, not to be confused with color noise of RGB images). Methods that are able to remove colored noise, such as DEB-BM3D [10], IDD-BM3D [11] and others (*e.g.* [14]) have been shown to achieve good deconvolution results.

Image denoising is itself a well-studied problem, with methods too numerous to list in this paper. Some approaches to denoising rely on learning, where learning can involve learning a probabilistic model of natural images [24], or of smaller natural image patches [31]. In that case, denoising can be achieved using a maximum a posteriori method. In other cases, learning involves learning a discriminative model for denoising, for example using convolutional neural networks [19]. In [16], it is shown that convolutional neural networks can achieve good image denoising results for AWG noise.

More recently, it was shown that a type of neural network based on stacked denoising auto-encoders [29] can achieve good results in image denoising for AWG noise as well as for “blind” image inpainting (when the positions of the pixels to be inpainted are unknown) [30].

Also recently, plain neural networks achieved state-of-the-art results in image denoising for AWG noise, provided the neural nets have enough capacity and that sufficient training data is provided [3, 4]. It was also shown that plain neural networks can achieve good results on other types of noise, such as noise resembling stripes, salt-and-pepper noise, JPEG-artifacts and mixed Poisson-Gaussian noise.

Differences and similarities to our work: We address the deconvolution problem as a denoising problem and therefore take an approach that is in line with [10, 11, 14], but different from [18]. However, as opposed to *engineered* algorithms [10, 11, 14], ours is *learned*. In that respect, we are similar to [24, 31]. However, our method is a discriminative method, and therefore more in line with [16, 30, 3]. We make no effort to use specialized learning architectures [16, 30] but use multi-layer perceptrons, similar to [3].

3. Method

The most direct way to deconvolve images with neural networks is to train them directly on blurry/clean patch pairs. However, as we will see in Section 4, this does not lead to good results. Instead, our method relies on two steps: (i) a regularized inversion of the blur in Fourier domain and (ii) a denoising step using a neural network. In this section, we describe these two steps in detail.

3.1. Direct deconvolution

The goal of this step is to make the blurry image sharper. This has the positive effect of localizing information, but it has the negative side-effect of introducing new artifacts. In our model, the underlying true (sharp) image x is blurred with a PSF v and corrupted with AWG noise n with standard deviation σ:

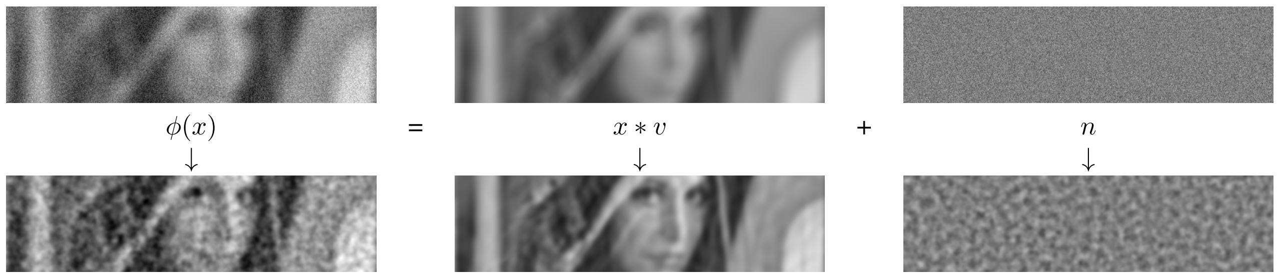
$$y = v * x + n. \tag{1}$$

The uncorrupted image can be estimated by minimizing $\|y - v * x\|^2$ with respect to x. A Gaussian prior on the gradient of x adds a regularization term $\alpha\sigma^2\|\nabla x\|^2$ to the objective. Furthermore, if we assume that our measurement of the blur kernel is corrupted by AWG, a further term $\beta\|x\|^2$ is obtained (see Sec. 6.4.1 in [2]), yielding

$$\|y - v * x\|^2 + \alpha\sigma^2\|\nabla x\|^2 + \beta\|x\|^2. \tag{2}$$

In Fourier domain, this can be solved in a single step [6]. Denoting the Fourier representations with capital letters (*e.g.* Fourier transform of x is X), the regularized inverse of the blurring transformation is

$$R = \frac{\bar{V}}{|V|^2 + \alpha\sigma^2 G + \beta}, \tag{3}$$


$$\phi(x) = x * v + n$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\mathcal{F}^{-1}(R \odot \mathcal{F}(\phi(x))) = \mathcal{F}^{-1}(R \odot \mathcal{F}(x) \odot \mathcal{F}(v)) + \mathcal{F}^{-1}(R \odot \mathcal{F}(n))$$

$$z = x_{\text{corrupted}} + n_{\text{colored}}$$

Figure 2. Illustration of the effect of the regularized blur inversion. The goal of image deconvolution is to deblur y. The result z of the regularized inversion is the sum of a corrupted image $x_{\text{corrupted}}$ and colored noise n_{colored}. Other methods [10, 11, 14] attempt to remove n_{colored} but ignore the noise in $x_{\text{corrupted}}$, whereas our method learns to denoise z and therefore addresses both problems.

where the division refers to element-wise division, $\bar{V}$ is the complex conjugate of V and $G = |\mathcal{F}(g_x)|^2 + |\mathcal{F}(g_y)|^2$. $\mathcal{F}(g_x)$ and $\mathcal{F}(g_y)$ refer to the Fourier transforms of the discrete gradient operators horizontally and vertically, respectively. The hyper-parameters α and β are responsible for the regularization: If both $\alpha = 0$ and $\beta = 0$, there is no regularization. Using the regularized inverse R, we can estimate the Fourier transform of the true image by the so-called *direct deconvolution* (following [15])

$$Z = R \odot Y = R \odot (X \odot V + N) \tag{4}$$
$$= R \odot X \odot V + R \odot N, \tag{5}$$

where $\odot$ is element-wise multiplication. Hence, the image recovered through the regularized inverse is given by the sum of the colored noise image $R \odot N$ and an image $R \odot X \odot V$ (as illustrated in Figure 2). The latter image is exactly equivalent to X if $\alpha = \beta = 0$ and the blur kernel doesn't have zeroes in its frequency spectrum, but otherwise generally not. We therefore see that methods trying to remove the colored noise component $R \odot N$ ignore the fact that the image itself is corrupted. We propose as step (ii) a procedure that removes the colored noise and additional image artifacts.

After direct deconvolution, the inverse Fourier transform of Z is taken. The resulting image usually contains a special form of distortions, which are removed in the second step of our method.

3.2. Artifact removal by MLPs

A *multi-layer perceptron* (MLP) is a neural network that processes multivariate input via several hidden layers and outputs multivariate output. For instance, the function expressed by an MLP with two hidden layers is defined as

$$f(x) = b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)), \tag{6}$$

where the weight matrices W_1, W_2, W_3 and vector-valued biases b_1, b_2, b_3 parameterize the MLP, and the function $\tanh$ operates component-wise. We denote the *architecture* of an MLP by a tuple of integers, *e.g.* $(39^2, 2047, 2047, 2047, 2047, 13^2)$ describes an MLP with four hidden layers (each having 2047 nodes) and patches of size 39×39 as input, and of size 13×13 as output. Such an MLP has approximately 1.6×10^7 parameters to learn, which is similar in scale to other large networks reported in literature [8, 27]. MLPs are also called *feed-forward neural networks*.

Training procedure: Our goal is to learn an MLP that maps corrupted input patches to clean output patches. How do we generate training examples? Starting with a clean image x from an image database, we transform it by a function ϕ that implements our knowledge of the image formation process. For instance, in the simulated experiment in Section 4.2, the clean image x is blurred by the PSF v and additionally corrupted by noise n. In this case ϕ is equivalent to the linear blur model in Equation (1).

The real-world photograph deblurred in Section 4.3 requires a more complicated ϕ as described in that section. We apply the direct deconvolution to $\phi(x)$ to obtain the image

$$z = \mathcal{F}^{-1}(R \odot \mathcal{F}(\phi(x))), \tag{7}$$

which is an image containing artifacts introduced by the direct deconvolution. Input-output pairs for training of the MLP are obtained by chopping z and x into patches. Using a large image database we can generate an abundance of training pairs.

The free parameters of the MLP are learned on such pairs of corrupted and clean image patches from z and x, using stochastic gradient descent [19]. The parameters of the MLP are then updated using the *backpropagation* algorithm [25], minimizing the pixel-wise squared error between the prediction of the MLP and the clean patch. The use of the squared error is motivated by the fact that we are interested in optimizing the peak signal-to-noise ratio (PSNR), which is monotonically related to the PSNR. We follow the setup described in [3] for data normalization, weight initialization and choice of the learning rate. We perform the training procedure on a modern GPU, resulting in a speedup factor of approximately an order of magnitude compared to a CPU implementation.

Application to images: To deblur an image, we first apply the direct deconvolution. The resulting image (showing characteristic artifacts) is then chopped into overlapping patches and each patch is processed separately by the trained MLP. The resulting reconstructed patches are placed at the locations over their corrupted counterparts and averaged in regions where they overlap. As described in [3], instead of choosing every sliding-window patch, we use a stride size of 3 (we pick every third patch) to achieve a speed-up factor of 9, while still achieving excellent results. This way, we can remove artifacts from an image of size 512×512 in approximately one minute on a modern computer (on CPU in MATLAB).

4. Results

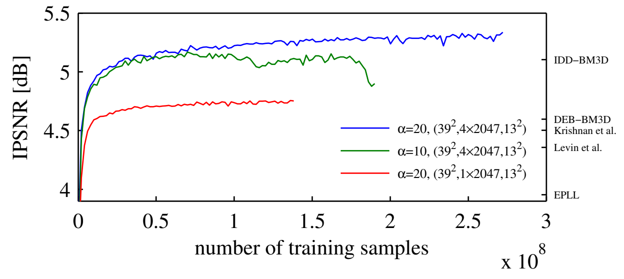


Figure 4. MLPs with more capacity lead to better results. If the regularization in the direct deconvolution is weak, strong artifacts are created, leading to bad results. IPSNR refers to the mean improvement in PSNR over 11 test images over their blurry counterparts. A square blur was used to produce this figure. The labels on the right indicate the results achieved with competing methods.

4.1. Choice of parameter values

Which experimental setups lead to good results? To answer this question, we monitor the results achieved with different setups at different times during the training procedure. Figure 4 shows that the results tend to improve with longer training times, but that the choice of the MLP's architecture as well as of the regularization strength α during direct deconvolution is important. Using four hidden layers instead of one leads to better results, given the same setting for direct deconvolution. If four hidden layers are used, better results are achieved with $\alpha = 20$ than with $\alpha = 10$. This is explained by the fact that too weak a regularization produces stronger artifacts, making artifact removal more difficult. In our experiments, we use $\alpha = 20$ for the direct deconvolution and $(39^2, 4 \times 2047, 13^2)$ for the architecture.

As mentioned above, it is also conceivable to train directly on blurry/clean patch pairs (*i.e.* on pairs $\phi(x)$ and x, instead of on pairs z and x), but this leads to results that are approximately 1.5dB worse after convergence (given the same architecture).

4.2. Comparison to other methods

To compare our approach to existing methods (described in Section 2), we first perform controlled experiments on a large set of images, where both the underlying true image and the PSF are known. Since the PSF is known exactly, we set β to zero. We train five MLPs, one for each of the following scenarios.

(a) Gaussian blur with standard deviation 1.6 (size 25×25) and AWG noise with $\sigma = 0.04$.

(b) Gaussian blur with standard deviation 1.6 (size 25×25) and AWG noise with $\sigma = 2/255$ (≈ 0.008).

(c) Gaussian blur with standard deviation 3.0 (size 25×25) and AWG noise with $\sigma = 0.04$.

(d) Square blur (box blur) with size 19×19 and AWG noise with $\sigma = 0.01$.

(e) Motion blur from [21] and AWG noise with $\sigma = 0.01$.

Scenarios (a) and (b) use a small PSF and (c) and (d) use a large PSF, whereas (b) and (d) use weak noise and (a) and (c) use strong noise. Scenarios (a), (b) and (c) have been used elsewhere, *e.g.* [11]. All of these blurs are particularly destructive to high frequencies and therefore especially challenging to deblur. Scenario (e) uses a motion blur recorded in [21], which is easier to deblur. Each MLP is trained on randomly selected patches from about $1.8 \cdot 10^8$ photos from the ImageNet dataset. Results seem to converge after approximately $2 \cdot 10^8$ training samples, corresponding to two weeks of GPU time. However, most competing methods are surpassed within the first day of training.

We evaluate our method as well as all competitors on black-and-white versions of the 500 images of the Berkeley segmentation dataset. The exponent of the sparseness prior in Krishnan *et al.* [18] was set to 0.8. Krishnan *et al.* and Levin *et al.* require a regularization parameter and IDD- BM3D [11] has two hyper-parameters. We optimized unknown values of these parameters on 20 randomly chosen images from ImageNet. Since only the methods using an image prior would be able to treat the boundary conditions correctly, we use circular convolution in all methods but exclude the borders of the images in the evaluation (we cropped by half the size of the blur kernel).

A performance profile of our method against all others on the full dataset is shown in Figure 3 and two example images are shown in Figure 5. Our method outperforms all competitors on most images, sometimes by a large margin (several dB). The average improvement over all competitors is significant. In Figure 5 we see that in smooth areas, IDD-BM3D [11] and DEB-BM3D [10] produce artifacts resembling the PSF (square blur), whereas our method does

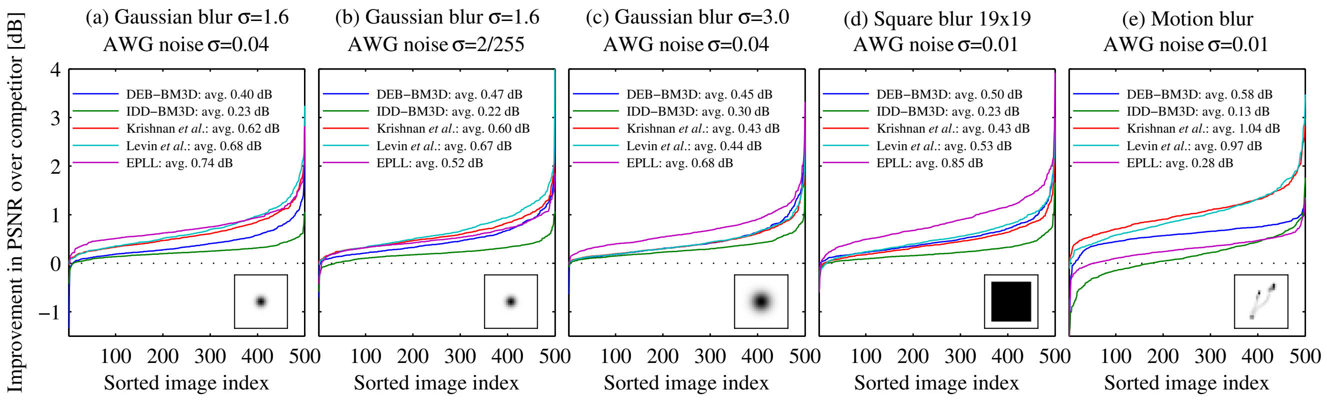


Figure 3. Comparison of performance over competitors. Values above zero indicate that our method outperforms the competitor.

not. The results achieved by Levin *et al*. and Krishnan *et al*. look “grainy” and the results achieved by EPLL [31] look more blurry than those achieved by our method. However, IDD-BM3D yields better results than our method in areas with repeating structures.

	(a)	(b)	(c)	(d)	(e)
EPLL [31]	24.04	26.64	21.36	21.04	29.25
Levin *et al.* [20]	24.09	26.51	21.72	21.91	28.33
Krishnan *et al.* [18]	24.17	26.60	21.73	22.07	28.17
DEB-BM3D [10]	24.19	26.30	21.48	22.20	28.26
IDD-BM3D [11]	24.68	27.13	21.99	22.69	**29.41**
FoE [26]	24.07	26.56	21.61	22.04	28.83
MLP	**24.76**	**27.23**	**22.20**	**22.75**	**29.42**

Table 1. Comparison on 11 standard test images. Values in dB.

A comparison against the Fields of Experts based method [26] was infeasible on the Berkeley dataset, due to long running times. Table 1 summarizes the results achieved on 11 standard test images for denoising [9], downsampled to 128×128 pixels.

For our scenarios IDD-BM3D is consistently the runner-up to our method. The other methods rank differently depending on noise and blur strength. For example, DEB-BM3D performs well for the small PSFs.

In the supplementary material we demonstrate that the MLP is optimal only for the noise level it was trained on, but still achieves good results if used at the wrong noise level.

Poisson noise For scenario (c) we also consider Poisson noise with equivalent average variance. Poisson noise is approximately equivalent to additive Gaussian noise, where the variance of the noise depends on the intensity of the underlying pixel. We compare against DEB-BM3D, for which we set the input parameter (the estimated variance of the noise) in such a way as to achieve the best results. Averaged over the 500 images in the Berkeley dataset, the results achieved with an MLP trained on this type of noise are slightly better (0.015dB) than with equivalent AWG noise, whereas the results achieved with DEB-BM3D are slightly worse (0.022dB) than on AWG noise. The fact that our results become somewhat better is consistent with the finding that equivalent Poisson noise is slightly easier to remove [22]. We note that even though the improvement is slight, this result shows that MLPs are able to automatically adapt to a new noise type, whereas methods that are not based on learning would ideally have to be engineered to cope with a new noise type (*e.g.* [22] describes adaptations to BM3D [9] for mixed Poisson-Gaussian noise, [7] handles outliers in the imaging process).

4.3. Qualitative results on a real photograph

To test the performance of our method in a real-world setting, we remove defocus blur from a photograph. We use a Canon 5D Mark II with a Canon EF 85mm f/1.2 L II USM lens to take an out-of-focus image of a poster, see Figure 1. In order to make the defocus blur approximately constant over the image plane, the lens is stopped down to f/5.6, which minimizes lens aberrations.

The function ϕ mimicking the image formation for this setup performs the following steps. First, an image from the training dataset is gamma-decompressed and transformed to the color-space of the camera (coefficients can be obtained from DCRAW). Then the image is blurred with a pillbox PSF with radius randomly chosen between 18.2 and 18.6. The radius of the actual PSF can be estimated by looking at the position of the first zero-frequency in Fourier domain. The randomness in the size of the pillbox PSF expresses that we don’t know the exact blur and a pillbox is only an approximation. This is especially true for our lens stopped down by eight shutter blades. Then the color image is converted to four half-size gray-scale images to model the Bayer pattern. Next, noise is added to the image. The variance of readout noise is independent of the expected illumination, but photon shot noise scales linearly with the mean, and pixel non-uniformity causes a quadratic increase in variance [1]. Our noise measurements on light frames are in agreement with this and can therefore be modeled by

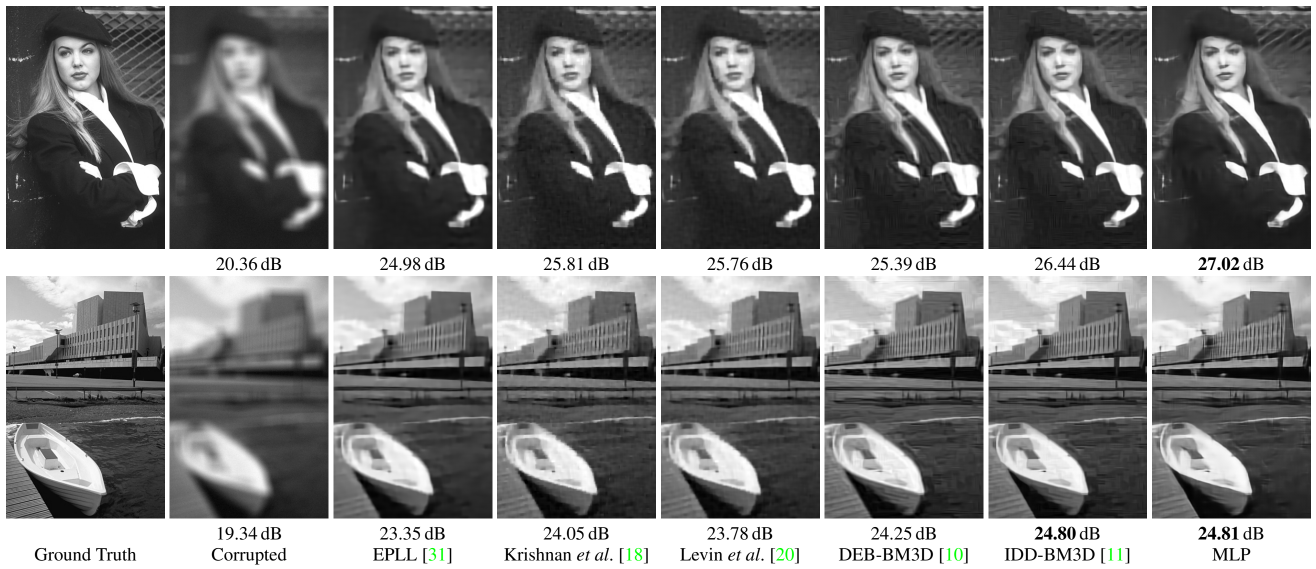


Figure 5. Images from the best (top) and worst (bottom) 5% results of scenario (d) as compared to IDD-BM3D [11]

a second-order polynomial. We have shown in Section 4.2 that our method is able to handle intensity-dependent noise.

To generate the input to the MLP we pre-process each of the four channels generated by the Bayer pattern via direct deconvolution using a pillbox of the corresponding size at this resolution (radius 9.2). Because of the uncertainty of the true kernel we set $\beta = 10^{-3}$. With this input, we learn the mapping to the original full resolution images with three color channels. The problem is higher-dimensional than in previous experiments, which is why we also increase the number of units in the hidden layers to 3071 (the architecture is therefore $(4 \times 39^2, 4 \times 3071, 3 \times 9^2)$). In Figure 1 we compare to the best visual results we could achieve with DEB-BM3D, the top algorithm with only one tunable parameter. The results were obtained by first de-mosaicking and then deconvolving every color channel separately (see supplementary material for other results).

In summary, we achieve a visually pleasing result by simply modeling the image formation process. By training on the full pipeline, we even avoid the need for a separate de-mosaicking step. It is not clear how this can be optimally incorporated in an engineered approach.

5. Understanding

Our MLPs achieve state-of-the-art results in image deblurring. But how do they work? In this section, we provide some answers to this question.

Following [5], we call weights connecting the input to the first hidden layer *feature detectors* and weights connecting the last layer to the output *feature generators*, both of which can be represented as patches. Assigning an input to an MLP and performing a forward pass assigns values to the hidden units, called *activations*. Finding an *input pattern* maximizing the activation of a specific hidden unit can be performed using *activation maximization* [13].

We will analyze two MLPs trained on the square PSF from scenario (d), both with the architecture $(39^2, 4 \times 2047, 13^2)$. The first MLP is trained on patches that are pre-processed with direct deconvolution, whereas the second MLP is trained on the blurry image patches themselves (*i.e.* no pre-processing is performed).

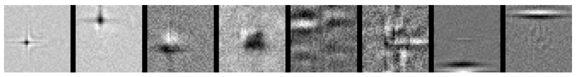


Figure 6. Eight feature detectors of an MLP trained to remove a square blur. The MLP was trained on patches pre-processed with direct deconvolution. The two rightmost features detect edges that are outside the area covered by the output patch, presumably detecting artifacts.

Analysis of the feature detectors: We start with the feature detectors of the MLP trained with pre-processed patches, see Figure 6. The feature detectors are of size 39×39 pixels. The area covered by the output patch lies in the middle of the patches and is of size 13×13 pixels. Some feature detectors seem to focus on small features resembling a cross. Others detect larger features in the area covered by the output patch (the middle 13×13 pixels). Still other feature detectors are more difficult to describe. Finally, some feature detectors detect edges that are completely outside the area covered by the output patch. A potential explanation for this surprising observation is that these feature detectors focus on artifacts created by the regularized inversion of the blur.

We perform the same analysis on the MLP trained on blurry patches, see Figure 7. The shape of the blur is evident

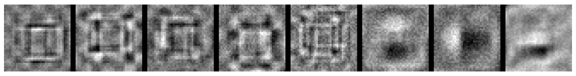


Figure 7. Eight feature detectors of an MLP trained to remove a square blur. The MLP was trained on the blurry patches themselves (*i.e.* no pre-processing). The features are large compared to the output patches because the information in the input is very spread out, due to the blur.

in most feature detectors: They resemble squares. In some feature detectors, the shape of the blur is not evident (the three rightmost). We also observe that all features are large compared to the size of the output patch (the output patches are three times smaller than the input patches). This was not the case for the MLP trained with pre-processing (Figure 6) and is explained by the fact that in the blurry inputs, information is very spread out. We clearly see that the direct deconvolution has the effect of making the information more local.

Analysis of the feature generators: We now analyze the feature generators learned by the MLPs. We will compare the feature generators to the input patterns maximizing the activation of their corresponding unit. We want to answer the question: What input feature causes the generation of a specific feature in the output?

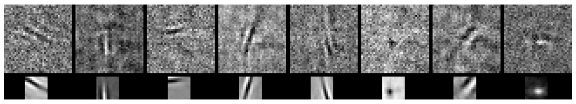


Figure 8. Input patterns found via activation maximization [13] (top row) vs. feature generators (bottom row) in an MLP trained on pre-processed patches. We see a clear correspondence between the input patterns and the feature generators. The MLP works by generating the same features it detects.

We start with the MLP trained on pre-processed patches. Figure 8 shows eight feature generators (bottom row) along with their corresponding input features (top row) maximizing the activation of the same hidden unit. The input patterns were found using activation maximization [13]. Surprisingly, the input patterns look similar to the feature generators. We can interpret the behavior of this MLP as follows: If the MLP detects a certain feature in the corrupted input, it copies the same feature into the output.

We repeat the analysis for the MLP trained on blurry patches (*i.e.* without pre-processing). Figure 9 shows eight feature generators (middle row) along with their corresponding input features (top row). This time, the features found with activation maximization look different from their corresponding feature generators. However, the feature detectors look remarkably similar to the feature generators convolved with the PSF (bottom row). We interpret this observation as follows: If the MLP detects a blurry version of a certain feature in the input, it copies the (non-blurry) feature into the output.

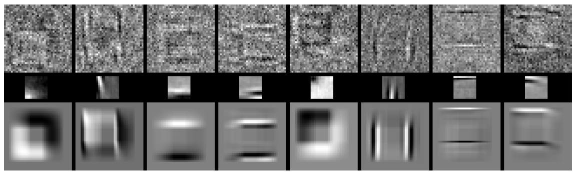


Figure 9. Input patterns found via activation maximization [13] (top row) vs. feature generators (middle row) in an MLP trained on blurry patches (*i.e.* no pre-processing). The input patterns look like the feature generators convolved with the PSF (bottom row). The MLP works by detecting blurry features and generating sharp ones.

Summary: Our MLPs are non-linear functions with millions of parameters. Nonetheless, we were able to make a number of observations regarding how the MLPs achieve their results. This was possible by looking at the weights connecting the input to the first hidden layer and the weights connecting the last hidden layer to the output, as well as through the use of activation maximization [13].

We have seen that the MLP trained on blurry patches has to learn large feature detectors, because the information in the input is very spread-out. The MLP trained on pre-processed patches is able to learn finer feature detectors. For both MLPs, the feature generators look similar: Many resemble Gabor filters or blobs. Similar features are learned by a variety of methods and seem to be useful for a number of tasks [12, 29]. We were also able to answer the question: Which inputs cause the individual feature generators to activate? Roughly speaking, in the case of the MLP trained on pre-processed patches, the inputs have to look like the feature generators themselves, whereas in the case of the MLP trained on blurry patches, the inputs have to look like the feature generators convolved with the PSF. Additionally, some feature detectors seem to focus on typical pre-processing artifacts.

6. Conclusion

We have shown that neural networks achieve a new state-of-the-art in image deconvolution. This is true for all scenarios we tested. Our method presents a clear benefit in that it is based on learning: We do not need to design or select features or even decide on a useful transform domain, the neural network automatically takes care of these tasks. An additional benefit related to learning is that we can handle different types of noise, whereas it is not clear if this is always possible for other methods. Finally, by directly learning the mapping from corrupted patches to clean patches, we handle both types of artifacts introduced by the direct deconvolution, instead of being limited to removing colored noise. We were able to gain insight into how our MLPs operate: They detect features in the input and generate corresponding features in the output. Our MLPs have to be trained on GPU to achieve good results in a reasonable amount of time, but once learned, deblurring on CPU is practically feasible. A limitation of our approach is that each MLP has to be trained on only one blur kernel: Results achieved with MLPs trained on several blur kernels are inferior to those achieved with MLPs trained on a single blur kernel. This makes our approach less useful for motion blurs, which are different for every image. However, in this case the deblurring quality is currently more limited by errors in the blur estimation than in the non-blind deconvolution step. Possibly our method could be further improved with a meta-procedure, such as [17].

References

[1] Noise, dynamic range and bit depth in digital slrs. http://theory.uchicago.edu/~ejm/pix/20d/tests/noise/. By Emil Martinec, updated May 2008. 5

[2] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 2

[3] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with bm3d? *IEEE Conf. Comput. Vision and Pattern Recognition*, pages 2392–2399, 2012. 2, 3, 4

[4] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds. *arXiv:1211.1544*, 2012. 2

[5] H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of their mechanisms. *arXiv:1211.1552*, 2012. 6

[6] S. Cho and S. Lee. Fast motion deblurring. In *ACM Trans. Graphics*, volume 28, page 145. ACM, 2009. 2

[7] S. Cho, J. Wang, and S. Lee. Handling outliers in non-blind image deconvolution. In *IEEE Int. Conf. Comput. Vision*, 2011. 5

[8] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010. 3

[9] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Trans. Image Process.*, 16(8):2080–2095, 2007. 2, 5

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3d transform-domain collaborative filtering. In *Soc. Photo-Optical Instrumentation Engineers*, volume 6812, page 6, 2008. 1, 2, 3, 4, 5, 6

[11] A. Danielyan, V. Katkovnik, and K. Egiazarian. Bm3d frames and variational image deblurring. *IEEE Trans. Image Process.*, 21(4):1715–1728, 2012. 2, 3, 4, 5, 6

[12] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. on Image Process.*, 15(12):3736–3745, 2006. 2, 7

[13] D. Erhan, A. Courville, and Y. Bengio. Understanding representations learned in deep architectures. Technical report, 1355, Université de Montréal/DIRO., 2010. 6, 7

[14] J. Guerrero-Colón, L. Mancera, and J. Portilla. Image restoration using space-variant gaussian scale mixtures in overcomplete pyramids. *IEEE Trans. Image Process.*, 17(1):27–41, 2008. 2, 3

[15] M. Hirsch, C. Schuler, S. Harmeling, and B. Scholkopf. Fast removal of non-uniform camera shake. In *IEEE Int. Conf. Comput. Vision*, pages 463–470. IEEE, 2011. 3

[16] V. Jain and H. Seung. Natural image denoising with convolutional networks. *Advances Neural Inform. Process. Syst.*, 21:769–776, 2008. 2

[17] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *Europ. Conf. Comput. Vision*. IEEE, 2012. 8

[18] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-Laplacian priors. In *Advances Neural Inform. Process. Syst.*, 2009. 2, 4, 5, 6

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998. 2, 3

[20] A. Levin, R. Fergus, F. Durand, and W. Freeman. Deconvolution using natural image priors. 26(3), 2007. 2, 5, 6

[21] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman. Understanding and evaluating blind deconvolution algorithms. In *IEEE Conf. Comput. Vision and Pattern Recognition*, pages 1964–1971. IEEE, 2009. 4

[22] M. Mäkitalo and A. Foi. Optimal inversion of the anscombe transformation in low-count poisson image denoising. *IEEE Trans. Image Process.*, 20(1):99–109, 2011. 5

[23] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Process.*, 12(11):1338–1351, 2003. 2

[24] S. Roth and M. Black. Fields of experts. *Int. J. Comput. Vision*, 82(2):205–229, 2009. 2

[25] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 3

[26] U. Schmidt, K. Schelten, and S. Roth. Bayesian deblurring with integrated noise estimation. In *IEEE Conf. Comput. Vision and Pattern Recognition*, pages 2625–2632. IEEE, 2011. 2, 5

[27] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *IEEE Int. Joint Conf. Neural Networks*, pages 2809–2813. IEEE, 2011. 3

[28] E. Simoncelli and E. Adelson. Noise removal via bayesian wavelet coring. In *IEEE Int. Conf. Image Process.*, volume 1, pages 379–382. IEEE, 1996. 2

[29] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learning Research*, 11:3371–3408, 2010. 2, 7

[30] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. *Advances Neural Inform. Process. Syst.*, 26:1–8, 2012. 2

[31] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *IEEE Int. Conf. Comput. Vision*, pages 479–486. IEEE, 2011. 2, 5, 6