Modelling and Learning Approaches to Image Denoising

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Harold Christopher Burger
aus Hamburg

Tübingen
2012

Tag der mündlichen Qualifikation	18.03.2013
Dekan	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter	Prof. Dr. Bernhard Schölkopf
2. Berichterstatter	Prof. Dr. Hendrik Lensch

Summary

Currently, most photographs are taken with digital cameras. Improvements in chip technologies have made possible the integration of digital cameras into other devices, such as mobile phones. This in turn has caused an explosion in the number of digital photographs taken each day. Unfortunately, all digital photographs contain an undesirable component commonly referred to as noise. Noise arises for a number of reasons. For example, photon shot noise is due to the discrete nature of light, and dark-current noise is due to the thermal energy of a camera's sensor. Image denoising is the problem of finding a clean image, given a noisy one. Using a denoising method becomes necessary when modifying the image acquisition process in such a way as to reduce the noise is not an option. This thesis presents three novel contributions to the field of image denoising.

Improving existing approaches using a multi-scale meta-procedure. Most denoising algorithms focus on recovering high-frequencies. However, for high noise levels it is also important to recover low-frequencies. We present a multi-scale meta-procedure that applies existing denoising algorithms across different scales and combines the resulting images into a single denoised image. We show that our method can improve the results achieved by many denoising algorithms.

Astronomical image denoising with a pixel-specific noise model. For digital photographs of astronomical objects, where exposure times are long, the dark-current noise is a significant source of noise. Usually, denoising methods assume additive white Gaussian noise, with equal variance for each pixel. However, dark-current noise has different properties for every pixel. We use a pixel-specific noise model to handle dark-current noise, as well as an image prior adapted to astronomical images. Our method is shown to perform well in a laboratory environment, and produces visually appealing results in a real-world setting.

Image denoising using multi-layer perceptrons. Many of the best-performing denoising methods rely on a cleverly engineered algorithm. In contrast, we take a learning approach to denoising and train a multi-layer perceptron to denoise image patches. Using this approach, we outperform the previous state-of-the-art. Our approach also achieves results that are superior to one type of theoretical bound and goes a large way toward closing the gap with a second type of theoretical bound. Furthermore, we achieve outstanding results on other types of noise, including JPEG-artifacts and Poisson noise. Also, we show that multi-layer perceptrons can be used to combine the results of several denoising algorithms. This approach often yields better results than the best method in the combination. We discuss in detail which trade-offs have to be considered during the training procedure. We are also able to make observations regarding the functioning principle of multi-layer perceptrons for image denoising.

Keywords: Image denoising, multi-scale, meta-procedure, astronomical image denoising, multi-layer perceptrons, machine learning, JPEG-artifacts, Poisson noise.

Zusammenfassung

Heutzutage werden die meisten fotographischen Bilder mit Digitalkameras aufgenommen. Verfeinerungen der Chip Technologien haben ermöglicht, dass Digitalkameras in andere Geräte so wie Mobiltelefone integriert werden. Dies wiederum hat zu einer Explosion in der Anzahl der täglich aufgenommenen Digitalfotos geführt. Leider enthalten alle Digitalfotos eine unerwünschte Komponente, nämlich das Rauschen. Bildentrauschung ist das Problem, ein sauberes Bild zu finden, wenn ein rauschiges gegeben ist. Eine Bildentrauschungsmethode zu verwenden ist dann nötig, wenn es nicht möglich ist, das Bildaufnahmeverfahren so zu verändern, dass weniger Rauschen entsteht. Diese Dissertation präsentiert drei neue Beiträge zu dem Feld der Bildentrauschung.

Verbesserung existierender Methoden durch ein multiskalen Metaverfahren: Die meisten Entrauschungsverfahren setzen den Schwerpunkt auf das Wiederherstellen hoher Frequenzen. Allerdings ist es bei starkem Rauschen auch wichtig, niedrigere Frequenzen zu beachten. Wir präsentieren ein multiskalen Metaverfahren, welches existierende Entrauschungsverfahren auf mehreren Skalen anwendet und die jeweiligen Ergebnisse wieder in ein entrauschtes Bild kombiniert. Wir zeigen, dass unser Verfahren die Ergebnisse vieler Entrauschungsverfahren verbessern kann.

Entrauschen astronomischer Bilder durch ein pixel-spezifisches Modell des Rauschens: In Digitalbildern von astronomischen Objekten, in welchen die Belichtungszeiten lang sind, ist das Dunkelstromrauschen eine wichtige Quelle von Rauschen. Normalerweise nehmen Entrauschungsverfahren additives, weißes Rauschen, mit gleicher Varianz für jeden Pixel an. Allerdings hat Dunkelstromrauschen andere Eigenschaften für jeden Pixel. Wir benutzen ein pixel-spezifisches Modell des Rauschens sowie eine *a priori* Wahrscheinlichkeit für Bilder, welche an astronomische Bilder angepasst ist. Wir zeigen, dass unsere Methode in einem Laboraufbau gut funktioniert und mit echten Bildern astronomischer Objekte visuell ansprechende Ergebnisse liefert.

Entrauschen durch mehrlagige Perzeptronen: Viele der am besten funkionierenden Entrauschungsverfahren verlassen sich auf ausgeklügelt konstruierte Algorithmen. Im Gegensatz dazu benutzen wir einen auf Lernen basierten Ansatz und trainieren mehrlagige Perzeptronen darauf, kleine Bildstücke zu entrauschen. Mit diesem Ansatz übertreffen wir die Ergebnisse des neuesten Stand der Technik. Unser Ansatz erreicht Ergebnisse, die einer Klasse theoretischer Grenzen überlegen sind und macht große Schritte, um eine zweite Klasse Grenzen zu erreichen. Außerdem erzielen wir ausgezeichnete Ergebenisse auf anderen Arten von Rauschen, einschließlich JPEG Artefakte und Poisson Rauschen. Wir zeigen auch, dass mehrlagige Perzeptronen in der Lage sind, die Ergebenisse anderer Entrauschungsverfahren zu kombinieren. Dieser Ansatz liefert oft Ergebnisse, die besser als das beste Ergebnis in der Kombination sind. Wir diskutieren im Detail, welche Kompromisse in der Trainingsprozedur eingegangen werden müssen. Wir sind auch in der Lage, Beobachtungen bezüglich der Funkionsweise von mehrlagigen Perzeptronen für Bildentrauschung zu machen.

Acknowledgements

I am particularly grateful to Prof. Dr. Bernhard Schölkopf for giving me the opportunity to work in his lab and supporting me in many ways. Special thanks go to Dr. Stefan Harmeling for his supervision and his help, as well as many interesting discussions. Also, I am deeply indebted toward Prof. Dr. Hendrik Lensch, who immediately agreed to become my "Doktorvater" and provided invaluable help for my dissertation.

I would also like to thank my co-author and office mate Christian Schuler for working with me and for incredibly constructive discussions. Great thanks to Prof. Dr. Bernhard Schölkopf, Dr. Stefan Harmeling and Christian Schuler for letting me base chapters of my thesis on joint publications. I would also like to thank my other office mates Paul Joubert, Alexander Loktyushin, Rolf Köhler and on occasion Dr. Michael Hirsch, for great discussions and for not complaining while I eat in the office. I would also like to thank Martin Kiefel, Dr. Philipp Hennig and Dr. Peter Gehler for many interesting discussions.

Dr. Ronan Collobert and Dr. Eric Cosatto deserve special thanks for supervising me so well while I was a research assistant and NEC Labs America in Princeton, New Jersey, for encouraging me to pursue a Ph.D., and for recommending that I apply at the Max Planck Institute in Tübingen. They also deserve special thanks for the fun we had outside of work. I am deeply indebted to all my former colleagues at NEC for teaching me so much, but Dr. Ronan Collobert should be particularly commended for teaching me almost everything I know about neural networks.

This dissertation would have been impossible without Sebastian Stark, our IT administrator, who is apparently able to easily solve any IT-related problem. Our secretary Sabrina Rehbaum cannot be praised enough for her help regarding many different issues.

I would like to thank Steffi, for helping me in so many ways and for not noticing my flaws.

Finally, I would like to thank my parents for giving me life, mind, and books about astronomy and dinosaurs when I was only five. These books gave me an appetite for more knowledge.

Nomenclature

AWG	Additive white Gaussian
AWGN	Additive white Gaussian noise
BLS-GSM	Bayesian least squares, Gaussian scale mixture (a denoising algorithm)
BM3D	Block-matching and 3D-filtering (a denoising algorithm)
CT	Computed tomography
dB	decibel
EPLL	Expected patch log-likelihood (a denoising algorithm)
FoE	Fields of Experts
GSM	Gaussian scale mixture
MLP	Multi-layer perceptron
MRI	Magnetic resonance imaging
MS-	Multi-scale
NL-means	Non-local means (a denoising algorithm)
NLSC	Non-local sparse coding (a denoising algorithm)
PET	Positron-emission tomography
PSNR	Peak signal-to-noise ratio
RMSE	Root mean squared error
SAR	Synthetic aperture radar
SNR	Signal-to-noise ratio
SSIM	Structural similarity index
TV	Total variation

Papers included in this thesis

[11] H.C. Burger, and S. Harmeling. Improving denoising algorithms via a multi-scale meta-procedure. Proceedings of the 33rd international conference on Pattern recognition (DAGM). 2011.
This paper was awarded with a prize at DAGM 2011.

[14] H.C. Burger, B. Schölkopf, and S. Harmeling. Removing noise from astronomical images using a pixel-specific noise model. International Conference on Computational Photography (ICCP). 2011.

[15] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? Conference on Computer Vision and Pattern Recognition (CVPR). 2012.

[12] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds. **Submitted** to a journal, **available** at `http://arxiv.org/abs/1211.1544`

[13] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of hidden activation patterns. **Submitted** to a journal, **available** at `http://arxiv.org/abs/1211.1552`

Papers partially included in this thesis:

[16] H.C. Burger, C.J. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. **Submitted** to a conference.

Papers related to, but not included in this thesis:

[107] C.J. Schuler, H.C. Burger, S. Harmeling, and B. Schölkopf. A machine learning approach for non-blind image deconvolution. Conference on Computer Vision and Pattern Recognition (CVPR). 2013.

I would like to thank my co-authors for their permission to base chapters of my thesis on joint publications.

Contents

1

Introduction

Chapter abstract This chapter defines the problem of image denoising and describes settings in which image denoising is important. Also discussed are various measures to evaluate image denoising results. Finally, this chapter summarizes the contributions made by this thesis.

Figure 1.1: A noisy image is assumed to be the sum of an underlying clean image and noise.

1.1 Problem definition: What is image denoising?

Image denoising is the problem of finding a clean image, given a noisy one. In most cases, it is assumed that the noisy image is the sum of an underlying clean image and a noise component, see Figure 1.1. Hence image denoising is a decomposition problem: The task is to decompose a noisy image into a clean image and a noise component. Since an infinite number of such decompositions exist, one is interested in finding a *plausible* clean image, given a noisy one. The notion of plausibility is not clearly defined, but the idea is that the denoised image should look like an image, whereas the noise component should look noisy. The notion of plausibility therefore involves *prior knowledge*: One knows something about images and about the noise. Without prior knowledge, image denoising would be impossible.

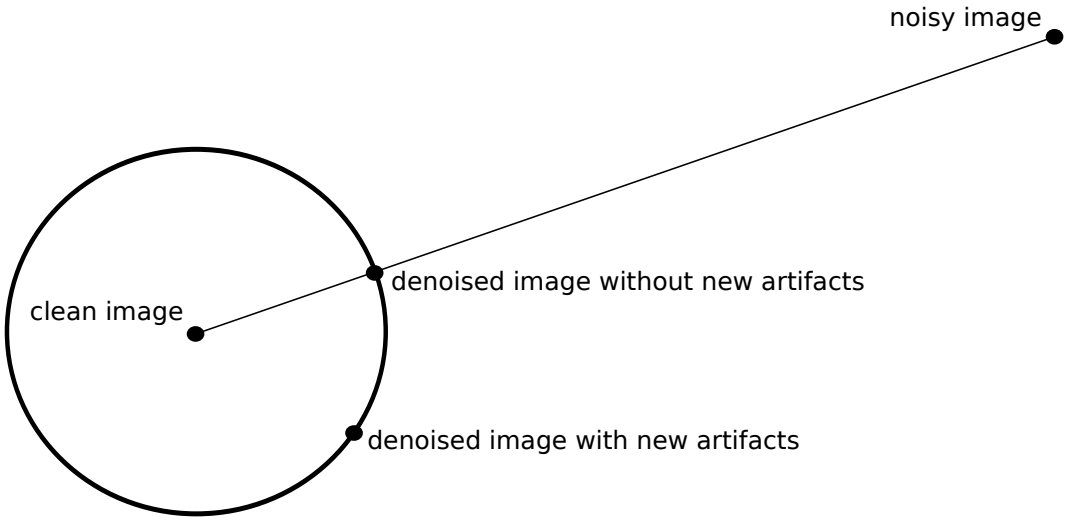


Figure 1.2: Two-dimensional illustration of the denoising problem. The two denoised images have the same ℓ_2-distance to the clean image, but only the denoised image lying on the path between the noisy image and the clean image contains no new artifacts.

One can think of an image as a point lying in a high-dimensional space. Hence, image denoising involves moving from one point in a high-dimensional space (the noisy image), to a different point in the same space (the clean image) which is unknown *a priori*. Usually, it is impossible to find the clean image exactly. One is therefore interested in finding an image that is *close* to the clean image. We discuss different measures of closeness in Section 1.3. In Figure 1.2, the denoising problem is illustrated using the ℓ_2-norm as a measure of closeness. In the figure, each point represents an image. All the images lying on the circle around the clean image have the same ℓ_2-distance to the clean image. However, some images on the circle are more desirable than others: The image lying on the straight line between the

noisy image and clean image is the most desirable because it contains no new artifacts (*i.e.* no artifacts that are not contained in the noisy image). This is due to the fact that the noise is assumed to be additive. All other points on the circle contain some new artifacts. Usually, it is impossible to find a point lying exactly on the line between the noisy image and clean image. Hence, denoised images almost invariably contain artifacts not contained in the noisy image. During denoising, one ideally seeks to introduce artifacts that are the least visually annoying. However, it is not clear how to define a measure or "visual annoyance", see Section 1.3.

1.2 Why is denoising important? Sources and types of noise

During any physical measurement, it is likely that the signal acquisition process is corrupted by some amount of noise. The sources and types of noise depend on the physical measurement. Noise often comes from a source that is different from the one to be measured (*e.g.* read-out noise in digital cameras), but sometimes is due to the measurement process itself (*e.g.* photon shot noise). Sometimes, noise might be due to the mathematical manipulation of a signal, as is the case in image deconvolution or image compression. Often, a measurement is corrupted by several sources of noise and it is usually difficult to fully characterize all of them. In all cases, noise is the undesirable part of the signal. Ideally, one seeks to reduce noise by manipulating the signal acquisition process, but when such a modification is impossible, denoising algorithms are required.

The characteristics of the noise depend on the signal acquisition process. Images can be acquired in a number of ways, including, but not limited to: Digital and analog cameras of various kinds (*e.g.* for visible or infra-red light), magnetic resonance imaging (MRI), computed tomography (CT), positron-emission tomography (PET), ultrasonography, electron microscopy and radar imagery such as synthetic aperture radar (SAR). The following is a list of possible types of noise.

Additive white Gaussian noise: In image denoising, the most common setting is to use black-and-white images corrupted with additive white Gaussian (AWG) noise, see *e.g.* [94, 1, 25]. For each pixel, a random value drawn from a normal distribution is added to the clean pixel value. The distribution is the same for every pixel (*i.e.* the mean and variance are the same) and the noise samples are drawn independently of each other. The read-out (or "amplifier") noise of digital cameras is often approximately AWG. An example of an image corrupted with AWG noise is shown in Figure 1.1.

Photon shot noise: Images are inevitably affected by photon shot noise. This is due to the discrete nature of light: During a given exposure time, only a finite number of photons reach the imaging sensor. The number of photons reaching the sensor follows a Poisson distribution. The standard deviation of a Poisson distribution is equal to the square root of its expected value. The signal-to-noise ratio (SNR) of a pixel x, defined as

$$\mathrm{SNR} = \frac{\mathbb{E}\{x\}}{\sqrt{\mathrm{Var}\{x\}}} \tag{1.1}$$

is therefore low when the expected number of photons is low:

$$\mathrm{SNR} = \frac{\mathbb{E}\{x\}}{\sqrt{\mathbb{E}\{x\}}} \tag{1.2}$$

$$= \sqrt{\mathbb{E}\{x\}}. \tag{1.3}$$

Photon shot noise if therefore especially noticeable in low light conditions (so-called photon-limited imaging [71, 76]), whereas it is barely noticeable in cases where many photons are

captured. When the mean is high (*e.g.* $\mathbb{E}\{x\} \geq 10$), the Poisson distribution looks similar to a Gaussian distribution with equal mean and variance. In that setting, removing Poisson noise is similar to removing Gaussian noise, where each pixel has a variance which depends on the pixel value of the underlying clean image. For lower mean values, the two distributions do not look similar.

The pixel size of image sensors has become smaller with progressing technology. The number of photons captured per pixel decreases as the area of a pixel decreases. Hence, photon shot noise is becoming more and more of a problem [106].

Thermal noise: Thermal noise arises due to the thermal energy of a chip. Thermally generated electrons accumulate in the chip's wells and are indistinguishable from photo-electrons. Thermal noise occurs even in the absence of light and is therefore sometimes referred to as dark-current noise. This type of noise is strongly dependent on the temperate of the sensor, but also on exposure time as well as the ISO-setting of the camera. Each pixel can be approximately modeled as a Gaussian.

Thermal noise is an example of noise which can be reduced by modifying the signal acquisition process: Cooling the camera's sensor reduces thermal noise. This type of noise is studied in more detail in Chapter 4.

Salt-and-pepper noise: Salt-and-pepper noise is a type of noise where the image contains a certain percentage of noisy pixels, where the noisy pixels are randomly either completely dark (pixel value zero) or saturated (highest possible pixel value). The value of the noisy pixels is therefore completely uncorrelated with the value of the same pixels in the clean image, which is different from *e.g.* AWG or Poisson noise. Salt-and-pepper noise can arise due to errors during transmission of an image.

Compression artifacts: Digital images are usually stored in a compressed format such as JPEG or JPEG-2000. The compression algorithm gives rise to artifacts, which can be considered a type of noise. The JPEG-algorithm is the most commonly used compression algorithm and causes "blocking" artifacts. In addition, information is lost due to compression. The noise can be said to be somehow non-linearly related to the clean image.

Rician noise: Magnetic resonance images are usually corrupted by Rician noise [46]. In MRI data, each pixel consists of a complex number. For viewing MRI data, the absolute value of each complex number is taken. If the real and imaginary parts of the complex number are Gaussian-distributed and independent (with the same variance), the absolute value is Rician-distributed. Similarly to the Poisson distribution, the Rician distribution can be well approximated with a Gaussian distribution, for higher mean values.

Colored noise: Colored noise refers to the situation where neighboring noise samples are correlated. Colored noise frequently occurs in electronics, where various shapes of the power spectral density are given names (*e.g.* pink and brown noise). In images, colored noise can arise during deconvolution, see Figure 1.3. Deconvolution is the process of finding a sharp image, given a blurry one. Colored noise can occur during deconvolution because a process called *direct deconvolution* [55] (a regularized inversion of the blur in Fourier domain) amplifies and colors the noise present in the input blurry image. Colored noise during deconvolution is an example of noise that occurs due to a mathematical manipulation of the signal: The noise is usually weak prior to direct deconvolution, but stronger thereafter.

Other noise: The types of noise possibly corrupting images are too numerous to list in this work. We are mostly interested in digital images, but images taken with analog cameras are also affected by noise, such as film grain. Physical degradation of old photographs such as daguerreotypes also causes a variety of artifacts, such as scratches, see Figure 1.4.

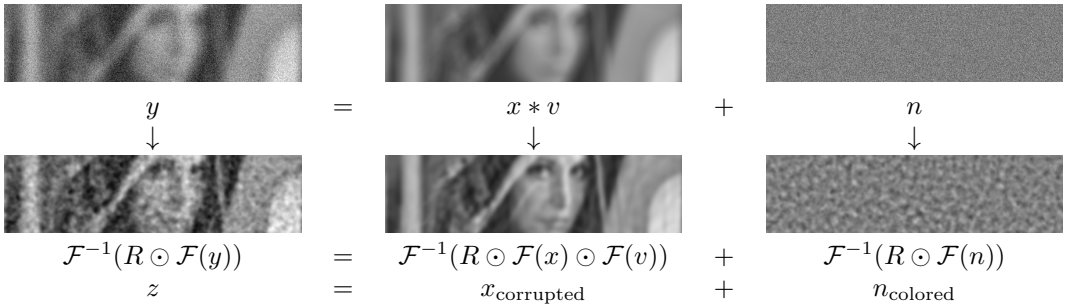


Figure 1.3: Image deconvolution can be addressed as a denoising problem (figure taken from [107], with permission of co-authors). The blurry image y contains a small amount of AWG noise. Performing a *direct deconvolution* [55] (multiplying with R, a regularized inverse of the blurring transformation, in Fourier domain) sharpens the image, but also (i) amplifies and colors the noise (see n_{colored}) and (ii) corrupts the image content (see $x_{\text{corrupted}}$). Therefore, image deconvolution can be performed by removing the noise in the image z. Methods such as [26, 27, 47] denoise z by attempting to remove n_{colored} (but ignore the noise in $x_{\text{corrupted}}$). In [107], both n_{colored} and $x_{\text{corrupted}}$ are treated.

Figure 1.4: Examples of noisy analog images, in this case of daguerreotypes of Andrew Jackson (left), taken in 1844 or 1845 and of Shimazu Nariakira (right), taken in 1857. Source: Wikimedia.

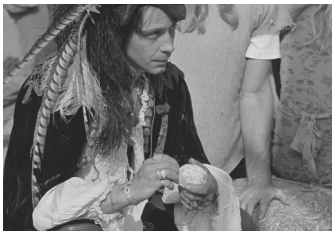
Such scratches are different from the types of noise we have discussed so far in that they have large-scale structure: A single scratch can potentially run across a whole image. Such large-scale structures can also occur in digital images: Some sensors are affected by row- or column-noise. In such cases, the noise samples within a row (or column) might be the same, but the noise samples between rows (or columns) different. In this thesis, we will only consider AWG noise, Poisson noise, thermal noise, salt-and-pepper noise, JPEG artifacts, as well as a "stripe" noise, resembling row or column noise.

Summary: Sources and types of noise are numerous and diverse and occur in almost all imaging settings. When designing a denoising algorithm, prior knowledge about the noise has to be adapted depending on the type of noise. The situation generally becomes more difficult when several types of noise affect the same image.

ground truth

noisy

PSNR: 14.16 dB
SSIM [123]: 0.37
MS-SSIM [125]: 0.61
IW-SSIM [124]: 0.62
IW-PSNR [124]: 21.18dB
VIF [109]: 0.05



denoised with "`replaceWithMan`"
PSNR: 10.69 dB
SSIM [123]: 0.18
MS-SSIM [125]: 0.13
IW-SSIM [124]: 0.05
IW-PSNR [124]: 9.50dB
VIF [109]: 0.01

Figure 1.5: Applying the hypothetical denoising algorithm "`replaceWithMan`" creates a denoised image that follows the statistics of natural images (and is therefore visually appealing), yet is unrelated to the underlying true image: The ℓ_2-distance between the images is high (and therefore the PSNR is small).

1.3 Evaluating denoising results: Measures of image quality

After denoising an image, we would like to know: How good is the denoising result? In asking this question, we are actually asking two questions: (i) How close (*e.g.* in terms of ℓ_2-norm) is the denoising result to the underlying true (clean) image? And (ii) How good does the denoised image look? One could imagine extremes in both scenarios. Let us a consider a hypothetical denoising algorithm called `replaceWithMan` that replaces the noisy input image with a different (non-noisy) image, see Figure 1.5. This algorithm produces an image that looks visually appealing, yet is unrelated to the true image underlying the noisy input image (*i.e.* the ℓ_2-distance to the true image is high). A different algorithm, called `denoiseWithBadFace` might produce a result that is close to the underlying true image (again in terms of ℓ_2-norm), but be bad at denoising faces, see Figure 1.6. The result produced by this algorithm is close to the underlying true image, yet is not visually appealing. Both scenarios are undesirable. But which trade-off is the most desirable?

Finding a good answer to this question is important in image denoising, because denoising almost inevitably introduces new artifacts. Hence it is important to know which artifacts are the most or least disturbing. A possible solution to this problem would be to rely on human (subjective) evaluation of the image quality. However, this solution is too inconvenient for many applications. Hence, one is interested in automatic image quality assessment and in particular in objective image quality metrics that correlate with subjective image quality.

Image quality metrics can be divided into three categories: (i) Full-reference, (ii) no-reference, and (iii) reduced reference metrics. Full-reference metrics assume that the true underlying image is available in order to compute a measure, whereas no-reference metrics perform a "blind" quality assessment: The true underlying image is not available. Reduced reference metrics lie somewhere in between the two previous scenarios and assume that the true image is partially known.

PSNR: The most commonly used metric for image quality assessment is the peak signal-to-noise ration (PSNR), which is a full-reference metric and calculated between two images $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\mathrm{PSNR} = 20 \log_{10} \left(\frac{\max}{\mathrm{RMSE}(\mathbf{x}, \mathbf{y})} \right), \tag{1.4}$$

ground truth	noisy	denoised with "`denoiseWithBadFace`"
	PSNR: 14.16 dB	PSNR: 26.09 dB
	SSIM [123]: 0.37	SSIM [123]: 0.94
	MS-SSIM [125]: 0.61	MS-SSIM [125]: 0.95
	IW-SSIM [124]: 0.62	IW-SSIM [124]: 0.90
	IW-PSNR [124]: 21.18dB	IW-PSNR [124]: 20.41dB
	VIF [109]: 0.05	VIF [109]: 0.84

Figure 1.6: Applying the hypothetical denoising algorithm "`denoiseWithBadFace`" creates a denoised image that is close to the underlying true image in terms of ℓ_2-distance (therefore the PSNR is high), but the result is visually not appealing: The artifacts in the middle of the denoised image do not look like a natural image.

where max refers to the maximum possible pixel value of the images (255 for 8-bit images). $\text{RMSE}(\mathbf{x}, \mathbf{y})$ refers to the root mean squared error: $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \mathbf{y}_i)^2}$, where the index i iterates over all pixels in the images. Hence, the PSNR is monotonically related to the ℓ_2-distance between two images. The unit used for the PSNR is decibel (dB), where a higher dB value indicates higher image quality in terms of RMSE (*i.e.* lower RMSE, or lower ℓ_2-distance). The PSNR is perhaps the simplest of allimage quality metrics. Still, higher dB values tend to correlate with higher visual similarity between the two images $\mathbf{x}$ and $\mathbf{y}$. However, higher dB values do not always indicate higher visual similarity, which is why extensive effort has been put into finding alternative metrics.

Other image quality metrics: Some image quality metrics attempt to exploit known characteristics of the human visual system. The structural similarity index (SSIM) [123] is a full-reference image quality metric which separates the task of similarity measurement into three components: (i) luminance, (ii) contrast, and (iii) structure. Among other things, the SSIM takes into account that the human visual system is sensitive to relative changes in luminance, rather than to absolute changes in luminance. The SSIM is a measure that is smaller or equal to 1. The measure is equal to 1 only in case the two images being compared are identical. Variants of the SSIM include a multi-scale extension (MS-SSIM [125]) and the information-content weighted SSIM (IW-SSIM [124]). Other full-reference image quality metrics include the information-content weighted PSNR (IW-PSNR [124]), the information fidelity criterion (IFC [110]) and the visual image information (VIF [109]). No-reference image quality metrics include DIIVINE [85], CBIQ [130], LBIQ [116], BLIINDS [104], BRISQUE [83], and BIQI [84]. These measures capture deviations from the expected statistics of natural images, where these deviations can be measured in different ways. For example, BLIINDS measures deviations from the expected histogram of certain features in DCT-domain. BIQI measures deviations from the expected distribution of wavelet coefficients in a multi-scale and multi-orientation decomposition.

Unfortunately, it is not clear which image quality assessment approach is the best, even in the full-reference setting. In the no-reference setting, this becomes even less clear. The PSNR is still the *de facto* standard in image denoising, though the SSIM is also sometimes used.

2

A brief summary of some existing denoising methods

Chapter abstract The following exposition is meant to be a brief overview of some existing denoising methods. Denoising is a long-standing problem, with techniques too numerous to list in this exposition. While not complete, the list of approaches we present here is intended to give the reader a feeling for the diversity of existing methods. A more complete, though also not exhaustive exposition of denoising techniques can be found in [62]. At the end of this chapter, we propose a hierarchical classification of the various approaches.

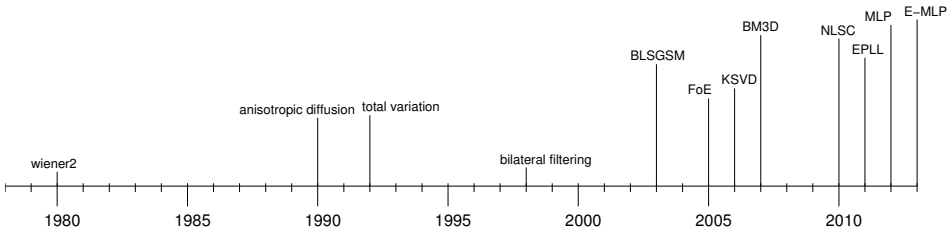


Figure 2.1: A timeline containing the date of introduction of a selection of denoising algorithms. The height of the vertical lines are proportional to the PSNR of the denoised image obtained by each algorithm on image "Lena", corrupted with AWG noise, $\sigma = 50$. The ranking of the different algorithms might be different on other images and noise levels. The methods MLP and E-MLP are contributions of this thesis and are described in Chapter 5.

Lena, clean noisy, PSNR: 14.16dB

Figure 2.2: Test image Lena. Ground truth (left) and noisy (right), corrupted with additive white Gaussian (AWG) noise with $\sigma = 50$.

2.1 Linear smoothing

A relatively simple approach to image denoising is to convolve a noisy image y with a Gaussian filter k:

$$\hat{x} = y * k. \tag{2.1}$$

This is a linear operation and can also be performed in Fourier domain:

$$\hat{X} = Y \odot K, \tag{2.2}$$

where capital letters denote the Fourier transform of their counterparts (*e.g.* $Y = \mathcal{F}(y)$, where $\mathcal{F}$ is the Fourier transform) and $\odot$ denotes the element-wise (Hadamard) product. The Gaussian filter k is still a Gaussian in the Fourier domain (*i.e.* K is also Gaussian). The Fourier representation makes it clear that the effect of Gaussian filtering is to keep low frequencies and to attenuate high frequencies (a Gaussian filter is a low-pass filter). This has a denoising effect because images contain mostly low-frequency information whereas AWG noise is spread evenly over the spectrum. Visually, images with attenuated high frequencies look smoother, which is why filtering with a Gaussian is a form of linear smoothing.

noisy, PSNR: 14.16dB denoised, PSNR: 25.42dB

Figure 2.3: Results using Gaussian filtering with filter width $\sigma = 1.8$.

Gaussian filtering can be implemented efficiently by exploiting the fact that the filter is *separable*: One first filters the image in the horizontal (or vertical) direction with a one-dimensional Gaussian filter. The resulting image is then filtered in the vertical (or horizontal) direction with a one-dimensional Gaussian filter. Doing so has the same effect as a two-dimensional convolution with a two-dimensional filter.

Prior to filtering with a Gaussian filter, one has to decide on the width σ of the filter. The optimal value depends both on the image at hand and on the strength of the noise, where higher noise levels require larger values of σ. An example of denoising with Gaussian filtering is shown in Figure 2.3. Linear filtering with filters other than the Gaussian are possible, though the Gaussian is the most commonly used filter.

Gaussian filtering attenuates high frequencies of a noisy image, which has the desirable effect of reducing the noise, but also has the undesirable effect of smoothing sharp image image features, such as edges. A possible solution to that problem is to adapt the shape of the filter to the image content: Different filters could be used for different parts of the image. We will discuss bilateral filtering [117] in Section 2.6, which is a method that exploits this idea.

noisy, PSNR: 14.16dB denoised, PSNR: 26.22dB

Figure 2.4: Results using two passes of median filtering with filter of size 5×5 and 7×7 pixels and symmetrically extended image borders.

2.2 Median filtering

An alternative to linear smoothing is median filtering. The idea underlying median filtering is to process an image pixel by pixel. Each pixel is replaced by the median of the value of a set of neighboring pixels. The method can therefore also be regarded as a filtering technique, though the filter is non-linear (and also non-separable in general).

It has often been said that median filtering is better at preserving edges than linear filtering, see *e.g.* [18]. However, more recent work indicates that this is not the case [4]: Under realistic assumptions, simple median filtering is no better at preserving edges than linear filtering. Another attribute of median filtering is that it is good at removing outliers [4]. Median filtering is therefore often used to remove salt-and-pepper noise.

The shape of the window in which the median is computed is a hyper-parameter, though a square window is the most common choice. The ideal window size depends on the image at hand as well as the noise level. Sometimes, median filtering is applied in several passes: The noisy image is first filtered with a median filter with a certain window size. The resulting image is then again median-filtered with a potentially different window size. It can be shown that this strategy can indeed better preserve edges than linear filtering [4]. We show an example of results achieved using this approach in Figure 2.4.

2.3 Denoising using local statistics

This section summarizes the method described by Jong-Sen Lee in 1980 [66]. This algorithm is an early approach to digital image denoising, invented at a time when the cost of computations was much higher than today. Still, the algorithm has become quite popular and can be executed in MATLAB using the `wiener2` function. The algorithm can handle both additive and multiplicative noise. We will here discuss the case of additive noise. A clean image x is corrupted with AWG noise n, giving us a noisy image y:

$$y = x + n. \tag{2.3}$$

We assume that the noise is independent of the underlying image, which gives us $\mathbb{E}\{n\} = 0$ and $\mathbb{E}\{xn\} = 0$. No assumption is made about the distribution other than that the variance σ^2 be finite and known and that the mean is 0. This gives us $\mathbb{E}\{n^2\} = \sigma^2 + \mathbb{E}\{n\}^2 = \sigma^2$.

The method described here estimates the mean and variance of the image locally. The assumption is that the *a priori* mean of a pixel is equal to the mean of the pixels surrounding it. A further assumption is that the same holds true for the variance of a pixel. It should be clear that these assumptions are often violated, and the paper [66] freely admits that the validity of the assumptions is debatable. Still, these assumptions lead to a denoising algorithm that is both simple and often effective in practice.

The algorithm therefore operates patch-wise: the mean of a pixel is estimated using the pixels in a surrounding patch of size $m \times n$, where m and n are hyper-parameters. Using a similar notation as [66], we denote with $\bar{x}$ the image of local mean estimates and Q will denote the image of local variance estimates. The derivations that follow hold true for each pixel in the image. For the mean, we have:

$$\bar{x} = \mathbb{E}\{x\} = \mathbb{E}\{y - n\} = \mathbb{E}\{y\} - \mathbb{E}\{n\} = \mathbb{E}\{y\} = \bar{y}. \tag{2.4}$$

The following useful derivation holds for the variance:

$$\begin{aligned}
\mathbb{E}\{(y - \bar{y})^2\} - \sigma^2 &= \mathbb{E}\{(y - \bar{y})^2 - n^2\} && (2.5)\\
&= \mathbb{E}\{y^2 - 2y\bar{y} + \bar{y}^2 - n^2\} && (2.6)\\
&= \mathbb{E}\{(x + n)^2 - 2(x + n)\bar{y} + \bar{y}^2 - n^2\} && (2.7)\\
&= \mathbb{E}\{x^2 + 2xn + n^2 - 2x\bar{y} - 2n\bar{y} + \bar{y}^2 - n^2\} && (2.8)\\
&= \mathbb{E}\{x^2 - 2x\bar{y} + \bar{y}^2\} && (2.9)\\
&= \mathbb{E}\{(x - \bar{x})^2\} && (2.10)\\
&\triangleq Q, && (2.11)
\end{aligned}$$

where we have exploited the independence of noise and image ($\mathbb{E}\{xn\} = 0$ and $\mathbb{E}\{n\bar{y}\} = 0$). We note that:

$$\begin{aligned}
\mathbb{E}\{(y - \bar{y})(x - \bar{y})\} &= \mathbb{E}\{(x + n - \bar{x})(x - \bar{x})\} && (2.12)\\
&= \mathbb{E}\{x^2 - x\bar{x} + xn - \bar{x}n - x\bar{x} + \bar{x}^2\} && (2.13)\\
&= \mathbb{E}\{x^2 - 2x\bar{x} + \bar{x}^2\} && (2.14)\\
&= \mathbb{E}\{(x - \bar{x})^2\}, && (2.15)
\end{aligned}$$

where we have again made use of the independence of the noise and the image ($\mathbb{E}\{xn\} = 0$ and $\mathbb{E}\{\bar{x}n\} = 0$). Our goal is to find an estimate of the true image using the following expression:

$$\hat{x} = \bar{x} + \beta(y - \bar{x}), \tag{2.16}$$

where β is a free parameter. In other words, the denoised image is given by the local mean estimate plus a weighted sum of the difference between the noisy image and the local mean

estimate. If β is close to 0, the denoised image will be close to $\bar{x}$, whereas if β is closer to 1, the denoised image will be closer to the noisy image y. This formulation is reminiscent of Wiener filtering. We will see that β will indeed take values between 0 and 1. We want to minimize the expectation of the squared error:

$$\min_{\beta} \mathcal{L} = \mathbb{E}\{(x - \bar{x} - \beta(y - \bar{x}))^2\}, \tag{2.17}$$

which can be done by taking the derivative over β:

$$\frac{\partial \mathcal{L}}{\partial \beta} = \mathbb{E}\{-2(y - \bar{x})(x - \bar{x} - \beta(y - \bar{x}))\} \tag{2.18}$$

$$= 2\mathbb{E}\{\beta(y - \bar{y})^2 - (y - \bar{y})(x - \bar{y})\}, \tag{2.19}$$

and setting to zero:

$$\hat{\beta} = \frac{\mathbb{E}\{(y - \bar{y})(x - \bar{y})\}}{\mathbb{E}\{(y - \bar{y})^2\}} \tag{2.20}$$

$$= \frac{\mathbb{E}\{(x - \bar{x})^2\}}{\mathbb{E}\{(y - \bar{y})^2\}}, \quad \text{using 2.15} \tag{2.21}$$

$$= \frac{\mathbb{E}\{(x - \bar{x})^2\}}{\mathbb{E}\{(x - \bar{x})^2\} + \sigma^2}, \quad \text{using 2.5} \tag{2.22}$$

$$= \frac{Q}{Q + \sigma^2}, \quad \text{using 2.11.} \tag{2.23}$$

We see that $\hat{\beta}$ is indeed guaranteed to lie between 0 and 1. If the local variance estimate is small compared to the noise level σ, $\hat{\beta}$ will be close to 0 and the denoised image will be close to $\bar{x}$. If the local variance estimate is large compared to the noise level σ, $\hat{\beta}$ will be close to 1 and the denoised image will be closer to y.

Putting everything together, we have the following expression for the denoised image:

$$\hat{x} = \bar{x} + \frac{(\mathbb{E}\{(y - \bar{y})^2\} - \sigma^2)(y - \bar{x})}{\mathbb{E}\{(y - \bar{y})^2\}}. \tag{2.24}$$

Denoising therefore involves the computation of $\bar{x}$ and $\mathbb{E}\{(y - \bar{y})^2\}$. The computation of $\bar{x}$ can be done efficiently by filtering the noisy image with a filter of size $m \times n$ consisting of only 1s. $\mathbb{E}\{(y - \bar{y})^2\}$ can be computed similarly by filtering the pixel-wise square of the noisy image. A big advantage of the method is therefore its computational efficiency. Using square windows, we found the optimal window size to be 5×5 for the image Lena on noise level $\sigma = 50$, see Figure 2.5. The method indeed achieves a big gain in PSNR, but the denoised image still looks very noisy.

noisy, PSNR: 14.16dB

denoised, PSNR: 25.25dB

Figure 2.5: Results using MATLAB's `wiener2` filter [66].

2.4 Total variation

The intuition behind image denoising based on total variation [101] is that noisy images have a larger discrete image gradient than noise-free images. In other words, noisy images look grainy, whereas clean images tend to be smooth. Hence finding an image that is smooth according to some measure but is close to the original noisy image should yield good denoising results.

We assume that we are given a noisy image which is the sum of an underlying true image corrupted by AWG noise $u = v + n$. We are looking for a cleaner image $\hat{v}$. The problem can be formalized as a maximum a posteriori (MAP) estimation problem:

$$\hat{v} = \arg\max_v p(v|u) \tag{2.25}$$

$$= \arg\max_v p(v)p(u|v) \tag{2.26}$$

$$= \arg\min_v -\log p(v) - \log p(u|v), \tag{2.27}$$

where for Gaussian noise, we have the likelihood term

$$\log p(u|v) = \frac{1}{2\sigma^2} \sum_{i,j} (u_{i,j} - v_{i,j})^2, \tag{2.28}$$

where the indices i and j run over all positions in the image. The image prior is given by $-\log p(v) = \mu ||v||_{\mathrm{TV}}$, where μ controls the regularization strength. Different choices for $||v||_{\mathrm{TV}}$ are possible, but most commonly, the following is used

$$||v||_{\mathrm{TV}} = \sum_{i,j} \sqrt{(\nabla_x v)_{i,j}^2 + (\nabla_y v)_{i,j}^2 + \epsilon}, \tag{2.29}$$

where $\nabla_x v$ and $\nabla_y v$ refer to the discrete horizontal and vertical image derivatives, respectively. One should use a value for ϵ that is slightly larger than zero to ensure differentiability at 0. The problem is convex and can be solved using gradient descent steps:

$$v^{(t+1)} = v^{(t)} + \eta \left[\frac{\partial ||v||_{\mathrm{TV}}}{\partial v} - \lambda(u - v) \right], \tag{2.30}$$

where $v^{(t=0)} = u$. The parameter λ controls the trade-off between the prior and the likelihood terms and should be chosen according to the level of noise. One usually iterates until the change between successive updates is deemed small enough. The discrete derivatives in $||v||_{\mathrm{TV}}$ can be computed in a number of ways, where the choice of how they are computed affects the optimization procedure through the computation of $\frac{\partial ||v||_{\mathrm{TV}}}{\partial v}$. Two possible choices are the one-sided difference $(\nabla_x v)^2 = (\nabla_x^+ v)^2$ and the central difference: $(\nabla_x v)^2 = (\frac{\nabla_x^+ v + \nabla_x^- v}{2})^2$, where ∇_x^+ and ∇_x^- refer to the difference in pixel value between reference pixels and pixels to the right or to the left, respectively.

A disadvantage of the optimization procedure through gradient descent steps is that it is rather slow, which is why considerable effort has been put into finding faster alternatives. An overview of different optimization procedures is given in [41]. To list just a few: [86] proposes a Newton-based method, [19] proposes a duality-based method. Other methods are based on graph-cuts [20] or on operator splitting [122].

noisy, PSNR: 14.16dB denoised, PSNR: 26.82dB

Figure 2.6: Result using total variation denoising [101]. The "cartoon"-like appearance of the denoised image is typical for total variance denoising.

Another disadvantage of total variation denoising is its tendency to produce areas that are overly smooth, particularly in textured regions. An example image is shown in Figure 2.6, showing the typical cartoon-like appearance of the denoised image due to overly smoothed regions that should contain texture. A modification of total variation denoising [43] addresses this issue by effectively making the trade-off parameter λ space-varying. Total variation denoising can also be modified to handle other kinds of noise, such as Rician noise [42].

2.5 Anisotropic diffusion

Anisotropic diffusion [90, 40] is an iterative procedure based on smoothing that can be used for image denoising. The method attempts to fulfill the following requirements: (i) Object boundaries should be preserved, and (ii) noise should be efficiently removed in homogeneous (flat) regions. Images can be considered to consist of regions (*e.g.* one region per object), in which case the goal of anisotropic diffusion is to preferentially perform smoothing within regions rather than between regions. The name of the procedure comes from the fact that it bears mathematical similarities to heat diffusion equations and from the fact that the diffusion or smoothing process is not performed uniformly over the whole image: Smoothing adapts to the image content. Anisotropic diffusion can therefore be seen as an improvement over isotropic Gaussian smoothing or blurring described in Section 2.1.

The procedure begins with a noisy image I. At each iteration t of the procedure, the current denoised estimate is updated using the previous estimate:

$$I^{t+1} = I^t + \lambda G, \tag{2.31}$$

where $I^{t=0}$ is the original noisy image. The scalar λ defines the step-size and G is of the same size as I and defines the updates at the current iteration. The procedure has to be stopped after a certain number of iterations: After a large number of iterations, the image becomes too smooth. The update G relies on the discrete gradients of the image:

$$G = \Phi_N(\nabla_N I) + \Phi_S(\nabla_S I) + \Phi_E(\nabla_E I) + \Phi_W(\nabla_W I), \tag{2.32}$$

where the flow functions $\Phi_\bullet$ are defined as follows

$$\Phi_N(\nabla_N I) = c(\nabla_N I) \odot \nabla_N I \tag{2.33}$$
$$\Phi_S(\nabla_S I) = c(\nabla_S I) \odot \nabla_S I \tag{2.34}$$
$$\Phi_E(\nabla_E I) = c(\nabla_E I) \odot \nabla_E I \tag{2.35}$$
$$\Phi_W(\nabla_W I) = c(\nabla_W I) \odot \nabla_W I. \tag{2.36}$$

The symbol $\odot$ refers to the Hadamard (element-wise) product. $\nabla_\bullet I$ refers to the discrete gradients along four directions:

$$\nabla_N I = I_{i-1,j} - I_{i,j} \tag{2.37}$$
$$\nabla_S I = I_{i+1,j} - I_{i,j} \tag{2.38}$$
$$\nabla_E I = I_{i,j+1} - I_{i,j} \tag{2.39}$$
$$\nabla_W I = I_{i,j-1} - I_{i,j}, \tag{2.40}$$

where we have assumed a grid size of 1. $\nabla_\bullet I$ can be computed quickly, through filtering with a 3×3 filter. Sometimes the diagonals are also included, giving eight filtered images instead of four.

The diffusion function (sometimes also called conduction coefficient) $c(\nabla I)$ depends on the magnitude of the gradient of the image intensity. It is monotonically decreasing. Two choices for $c(\nabla I)$ are proposed:

$$c(\nabla I) = (1 + \left(\frac{\nabla I}{\kappa}\right)^2)^{-1}, \tag{2.41}$$

or:

$$c(\nabla I) = e^{-(\frac{\nabla I}{\kappa})^2}, \tag{2.42}$$

see Figure 2.7. The effect of both diffusion functions is the following: When the discrete image gradient ∇I is large, the response of $c(\nabla I)$ is small. However, when the discrete

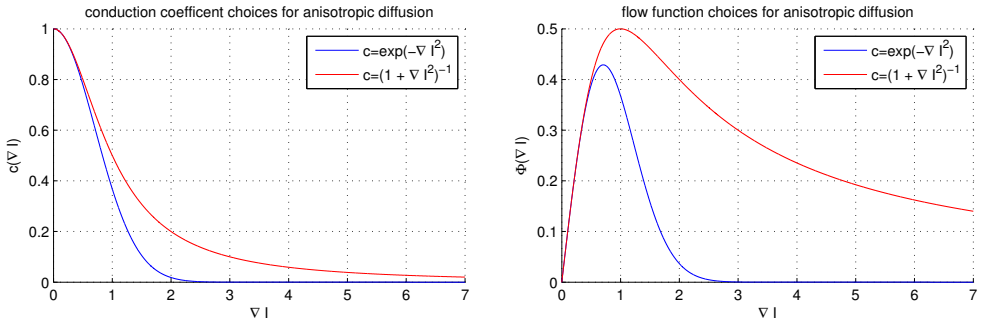


Figure 2.7: Two common choices for $c(\nabla I)$, for $\kappa = 1$. $\Phi(\nabla I)$ has a maximum at either $\nabla I = \sqrt{\frac{\kappa}{2}}$ or $\nabla I = \kappa$, depending on the choice of the diffusion function c.

noisy, PSNR: 14.16dB

denoised, PSNR: 26.75dB

Figure 2.8: Results using anisotropic diffusion. We used $\kappa = 40$, 12 iterations and $c(\nabla I) = (1 + (\frac{\nabla I}{\kappa})^2)^{-1}$

image gradient is small, $c(\nabla I)$ is large. This has the effect that the flow function is small for very small image gradients and for large image gradients, with a maximum flow lying somewhere in between. It can be shown [126] that this has the desirable effect of blurring small fluctuations and sharpening edges.

The following hyper-parameters affect the results of anisotropic diffusion: The value of λ, the choice of the diffusion function $c(\nabla I)$ as well as the value of its hyper-parameter κ and the number of iterations. The optimal choice of these hyper-parameters depends both on the image to be denoised and on the strength of the noise contained in the image. The results shown in Figure 2.8 were achieved by tuning the hyper-parameters on the image itself (the hyper-parameters are therefore optimal for that image).

One can see that anisotropic diffusion is straightforward to implement and is computationally not very expensive. Still, the results are quite satisfactory. In addition, there exist directed versions of anisotropic diffusion with additional direction information that might further improve on these results.

2.6 Bilateral Filtering

Bilateral filtering [5, 117], like anisotropic diffusion, attempts to smooth an image while preserving edges. A difference between the two approaches is that bilateral filtering is non-iterative. The idea underlying bilateral filtering is to non-linearly combine nearby image values. The pixels to be combined are chosen not only based on their geometric proximity, as is usual for filtering methods, but also based on their photometric similarity. Bilateral filtering can therefore be seen as a blend of two approaches: Domain-filtering and range-filtering.

noisy, PSNR: 14.16dB denoised, PSNR: 25.37dB

Figure 2.9: Results using bilateral filtering. We used $\sigma_d = 3.91$ and $\sigma_r = 0.69$.

Domain-filtering refers to classic filtering where pixels that are geometrically close-by are filtered together. Filtering an image with a Gaussian blur is an example of domain-filtering. Domain-filtering can be intuitively justified by the fact that images usually vary slowly over space (*i.e.* low frequencies dominate), whereas the noise is not correlated over space. Domain-filtering then has the effect of attenuating the noise while preserving the signal. However, the assumption that images vary slowly fails at image edges.

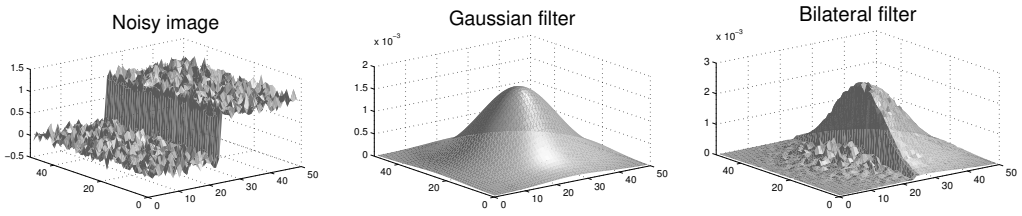
Range-filtering refers to filtering all pixels within an image that are similar in appearance. It can be shown [117] that range-filtering merely changes the gray map of the image (*i.e.* it changes the color histogram). More precisely, it compresses unimodal histograms (*i.e.* the resulting image has fewer different gray values), making images appear more "cartoon-like".

However, combining range filtering with domain filtering can result in more interesting effects, including denoising. Figure 2.10 shows the difference between domain-filtering and bilateral filtering. Bilateral filtering ensures that edges are preserved, which is not the case for pure domain-filtering.

In keeping with the notation used in [117], we denote with f the original (noisy) image. The pixel-value of an image f at position x is given by $f(x)$. The output image is obtained by filtering:

$$h(x) = k^{-1}(x) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\xi) c(\xi, x) s(f(\xi), f(x)) d\xi. \tag{2.43}$$

Both x and ξ are 2D-coordinates, explaining the double integral. The normalization $k(x)$ is



(a) (b) (c)

Figure 2.10: The noisy image (a) is to be denoised. Using a Gaussian filter (b) would not preserve the edge in the image. However, combining the Gaussian filter with a measure of pixel similarity can alleviate the problem. The filter in (c) is obtained using the Gaussian in image (b) for $c(\xi, x)$ and a Gaussian with $\sigma_r = 0.5$ for $s(f(\xi), f(x))$. The position x of the center pixel is two pixels to the right of the edge in (a).

given by:

$$k(x) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} c(\xi, x) s(f(\xi), f(x)) d\xi. \tag{2.44}$$

The function $c(\xi, x)$ defines the domain-filtering component and is therefore a measure of pixel closeness. A typical choice is the Gaussian:

$$c(\xi, x) = e^{-\frac{1}{2}\left(\frac{d(\xi,x)}{\sigma_d}\right)^2}, \tag{2.45}$$

where $d(\xi, x) = ||\xi - x||_2$ is the Euclidean distance between ξ and x. The function $s(f(\xi), f(x))$ is a measure of pixel similarity and therefore defined the range-filtering component of bilateral filtering. A typical choice for $s(f(\xi), f(x))$ is to also use a Gaussian:

$$s(f(\xi), x) = e^{-\frac{1}{2}\left(\frac{\delta(f(\xi), f(x))}{\sigma_r}\right)^2}, \tag{2.46}$$

where $\delta(f(\xi), f(x)) = ||f(\xi) - f(x)||_2$

Setting $c(\xi, x) = 1$ gives pure range-filtering whereas setting $s(f(\xi), f(x)) = 1$ gives pure domain-filtering. It can therefore be seen that bilateral filtering is a combination of domain-filtering and range-filtering.

Bilateral filtering has two hyper-parameters σ_d (the *geometric spread*) and σ_r (the *photometric spread*). The optimal value of the hyper-parameters is image-dependent and furthermore depends on the level of noise. However, it is not clear what the relation between the strength of the noise and the optimal hyper-parameter values is. The results shown in Figure 2.9 were obtained by optimizing the hyper-parameters on the image itself. The denoising effect is not very impressive, but the method is conceptually quite simple and is also computationally not too intensive.

2.7 Fields of Experts

The Fields of Experts framework [99, 100] is a model for image priors based on Markov random fields (MRFs). The prior probability of an image is modeled using a random field with overlapping cliques, whose potentials are represented as Products of Experts [50]. The framework is applicable to multiple low-level vision tasks, such as denoising and inpainting. Usually, extended cliques are used (3×3 or 5×5), so that image statistics beyond pairwise neighborhoods are captured. All parameters of the model are learned, where learning is performed with contrastive divergence [51] using a dataset of natural images. Denoising is achieved using a simple iterative gradient-descent approach on a negative log-likelihood term. Results are competitive with some other denoising methods even though the framework is versatile and can be applied to other problems.

Using a Products of Experts to model the prior probability of an image patch $\mathbf{x}$ can be done as follows

$$p(\mathbf{x}) = \frac{1}{Z(\Theta)} \prod_{i=1}^{N} \phi_i(\mathbf{J}_i^{\mathbf{T}}\mathbf{x}; \alpha_i), \quad \Theta = \{\theta_1, \ldots, \theta_N\}, \tag{2.47}$$

where $\theta_i = \{\alpha_i, \mathbf{J}_i\}$, N is the number of filters, and $Z(\Theta)$ is a partition function. The term $\mathbf{J}_i^{\mathbf{T}}\mathbf{x}$ is the response of a patch $\mathbf{x}$ to a filter $\mathbf{J}_i$. The functions ϕ_i are referred to as experts. Using

$$\phi_i(\mathbf{J}_i^{\mathbf{T}}\mathbf{x}; \alpha_i) = \left(1 + \frac{1}{2}(\mathbf{J}_i^{\mathbf{T}}\mathbf{x})^2\right)^{-\alpha_i} \tag{2.48}$$

gives us a product of t-distribution (PoT) model. The use of filters resembling the t-distribution is justified by the observation that the responses of linear filters to natural image patches resemble the t-distribution. The probability density $p(\mathbf{x})$ is often written in so-called Gibbs form:

$$p(\mathbf{x}) = \frac{1}{Z(\Theta)} \exp(-E_{\text{PoE}}(\mathbf{x}, \Theta)), \tag{2.49}$$

with

$$E_{\text{PoE}}(\mathbf{x}, \Theta) = -\sum_{i=1}^{N} \log \phi_i(\mathbf{J}_i^{\mathbf{T}}\mathbf{x}; \alpha_i). \tag{2.50}$$

Learning involves finding the parameter values for both the α_is and the filters $\mathbf{J}_i$. This model learns a prior distribution for small image patches. However, we are interested in modeling the distribution of entire images. The probability density of the full image is expressed using a Fields of Experts model as

$$E_{\text{FoE}}(\mathbf{x}, \Theta) = -\sum_{k} \sum_{i=1}^{N} \log \phi_i(\mathbf{J}_i^{\mathbf{T}}\mathbf{x}_{(k)}; \alpha_i), \tag{2.51}$$

where $\mathbf{x}_{(k)}$ refers to an image patch (or clique) centered at position k. It is assumed that the potentials are the same for all image patches: The MRF is therefore homogeneous. The difference between the Products of Experts model and the Fields of Experts model is that in the Fields of Experts model, one takes the product over all overlapping neighborhoods k (or equivalently: The sum over all neighborhoods is taken, using the Gibbs notation).

Training: Training is performed by minimizing the Kullback-Leibler divergence between the model and the data distribution. This makes sure that the model distribution is as close as possible to the data distribution and is equivalent to maximizing the likelihood of the FoE model. Training therefore involves repeatedly drawing samples from $p(x)$, which is time-consuming even with efficient sampling techniques. Training is made more efficient using contrastive divergence [51], where the sampling procedure is stopped after a small number of steps. The size of the training images has to be chosen carefully: Choosing very large

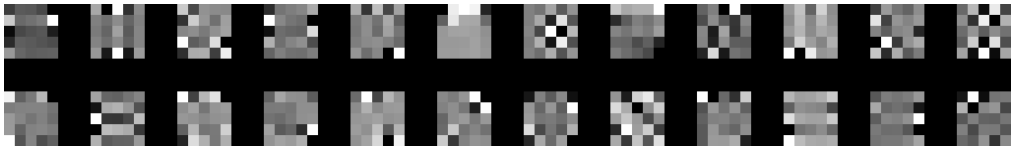


Figure 2.11: The FoE filters learned by contrastive divergence, as in [99]. The filters lack a clear interpretation.

images would make the training procedure very time-consuming, whereas choosing images that are no larger than the clique size would not model spatial dependencies of neighboring cliques (which is equivalent to the PoE model). A typical trade-off is to use training images that are three times as large as the clique size.

One also needs to decide on the number of filters N. In principle, a N can be chosen to be much smaller or much larger than the number of pixels in a clique. In [99], $N = 24$ filters is chosen for cliques of size 5×5. Usually, the filters learned in this way lack a clear interpretation, see Figure 2.11.

noisy, PSNR: 14.16dB denoised, PSNR: 27.29dB

Figure 2.12: Results using Fields of Experts [100] with early stopping.

Denoising: Denoising can be performed using a maximum a posteriori (MAP) estimate. Given an observed noisy image $\mathbf{y}$ assumed to be corrupted with AWG noise $\mathbf{n}$: $\mathbf{y} = \mathbf{x} + \mathbf{n}$, we want to maximize the posterior probability $p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x}) \cdot p(\mathbf{x})$. The likelihood term is written as

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_j \left(-\frac{1}{2\sigma^2}(y_j - x_j)^2 \right), \tag{2.52}$$

where j goes over all pixels in an image and σ refers to the standard deviation of the AWG noise. Denoising is performed using gradient ascent on the logarithm of the posterior probability distribution, giving rise to the following update rule:

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + \eta \left[\sum_{i=1}^{N} \mathbf{J}_i^- * \psi_i(\mathbf{J}_i * \mathbf{x}^{(t)}) + \frac{\lambda}{\sigma^2}(\mathbf{y} - \mathbf{x}^{(t)}) \right], \tag{2.53}$$

where $\psi_i(y) = \frac{\partial}{\partial y} \log \phi_i(y; \alpha_i)$, $\mathbf{J}_i^-$ refers to the filter $\mathbf{J}_i$ mirrored around its center pixel and $*$ refers to a convolution. The parameter η defines a step size and λ controls the trade-off between the prior and the likelihood. The optimal value of λ depends on the noise level σ. Good values can be found using cross-validation. In [99], 2500 iterations of gradient descent are performed. We found that we could obtain slightly better results by adapting the number of iterations to the noise level σ. For $\sigma = 50$, we found the best number of iterations to be 1900, see Figure 2.15. For lower noise levels, fewer iterations are necessary.

The FoE framework is considered to be a very successful MRF-based approach to modeling images and has been studied in later work, see *e.g.* [127, 105]. A disadvantage of the FoE model is that the partition function is intractable. Hence, calculating the probability of an image under the model is virtually impossible. In addition, the training procedure is very slow. Weiss and Freeman [127] propose solutions to both these problems. High likelihood filters are learned without the use of sampling, employing a procedure involving iterated PCA computations. This makes the learning process relatively fast. A Gaussian scale mixture (GSM) is used to model the potential function, which makes it possible to compute a lower and upper bound on the partition function (together with the assumptions that the filters have unit norm and are orthogonal to each other). In addition, it is shown that the non-intuitive filters learned in [99] indeed capture properties of natural images (and are not artifacts of the learning process).

In [105], an FoE model with GSM potentials is learned where the shape of the potentials is not fixed beforehand, but is determined by the training procedure. It is shown that the new model has heavier tails than other models and that its generative properties are superior. In addition, it is shown that MAP is not the best solution to the denoising problem. Instead, denoising is performed by estimating the Bayesian minimum mean squared error estimate (MMSE) using sampling. Denoising is therefore performed in a purely generative setting. This has the additional benefit of eliminating the need for ad-hoc modifications such as the trade-off parameter λ or deciding on a specific number of iterations.

2.8 BLS-GSM and other wavelet-based methods

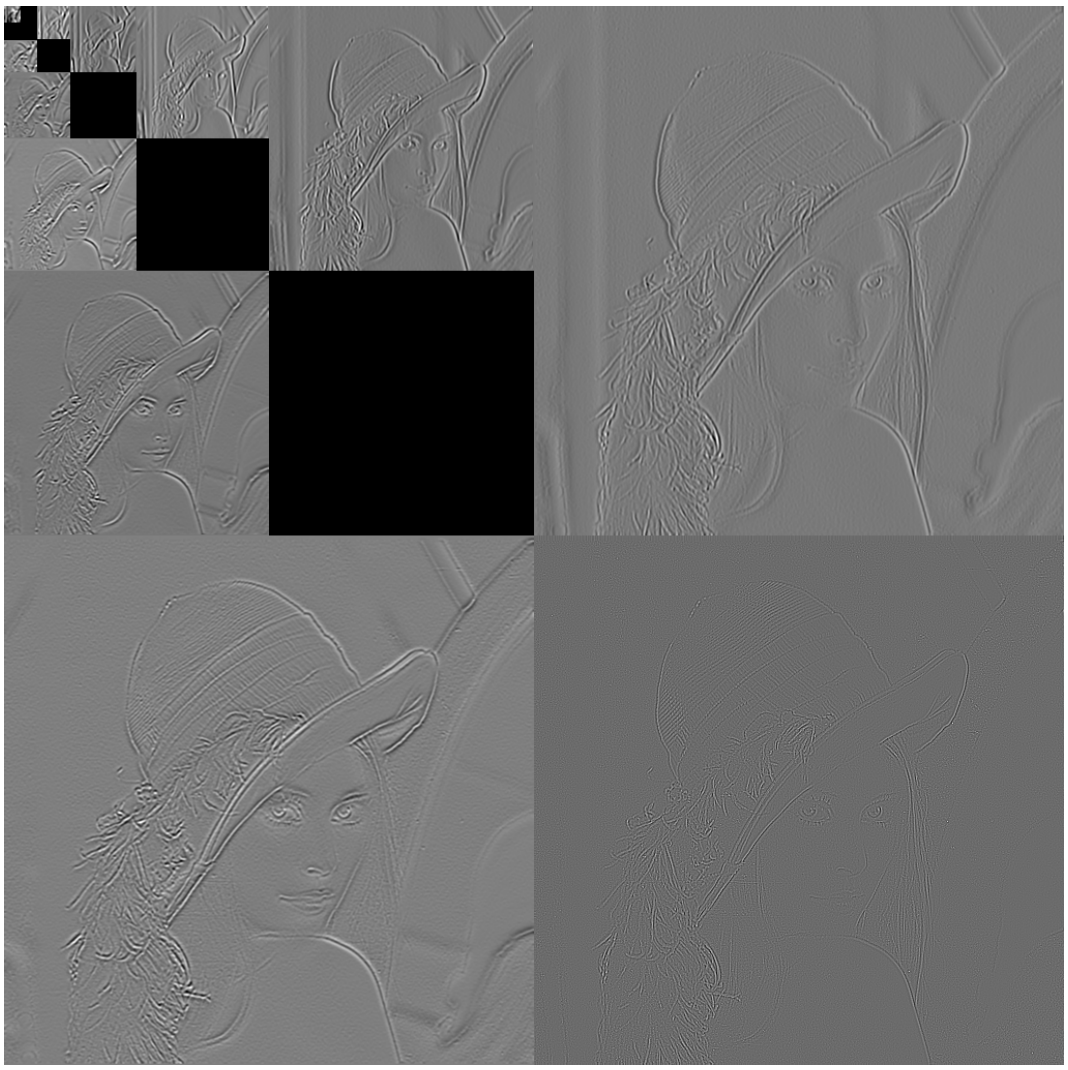
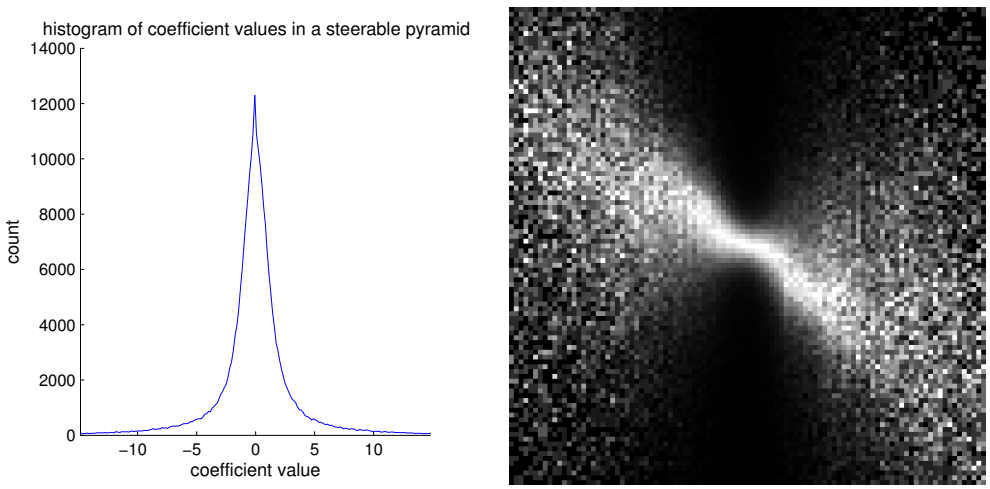


Figure 2.13: Steerable pyramid decomposition of image "Lena", using five scales. There are two orientations per scale (images of the same size belong to the same scale). The coarsest scale and the finest scales are special cases: The coarsest scale is not decomposed into orientation sub-bands and the finest scale contains an additional "residual" image.

Many image denoising approaches perform denoising on a wavelet decomposition of the noisy image. Wavelet decompositions have the desirable property of locality both in space and in frequency, which is not the case for other transforms, such as the Fourier transform. Wavelet-based denoising algorithms are based on the following steps. First, an image is transformed into a wavelet domain. Next, denoising is effected on the wavelet coefficients, and finally the denoised image is obtained by applying the inverse wavelet transform on the denoised wavelet coefficients. Methods relying on this strategy and too numerous to list in this work. We will focus on methods based on "steerable pyramid" [112] representation, though methods using other forms of decompositions also exist, *e.g.* curvelets [113], contourlets [82], chirplets [2] and wedgelets [28].

Figure 2.13 shows a steerable pyramid decomposition of the image "Lena". Decomposing

an image into a steerable pyramid is a linear operation. A steerable pyramid is conceptually similar to a Laplacian pyramid [17]: In both cases an image is represented using several scales. The coarsest scale contains the lowest frequencies of the image and the finer scales contain subsequently higher frequencies. A difference between the two representations is that in the steerable pyramid, each scale contains several orientation sub-bands. Each orientation sub-band corresponds to the response of a directional derivative operator. In the example shown in Figure 2.13, there are two orientation sub-bands per scale, but any number of orientation sub-bands can be used in principle. The steerable pyramid transform is over-complete (by a factor of $\frac{4k}{3}$, where k is the number of orientation sub-bands per scale): The decomposition is higher-dimensional than the original image. Over-completeness has both advantages and disadvantages over orthogonal transforms: Performing an over-complete transform is more computationally demanding than performing an orthogonal transform. Another advantage of orthogonal transforms is that they preserve the inner-product (also called the isometry property), so white noise in the input image remains white after the transform. Over-complete transforms such as the steerable pyramid do not share this convenient property: White noise in the image becomes correlated in the transform domain. However, over-complete transforms have the advantage of introducing fewer aliasing (or "ringing") artifacts. Ringing artifacts occur due to the Gibbs phenomenon when coefficients of an orthogonal transform are set to zero. A similarity between the steerable pyramid and orthogonal transforms is that the steerable pyramid decomposition is "self-inverting": The inverse transformation can be performed by applying the transpose of the matrix performing the forward transform. The same holds true for orthogonal transforms. The steerable pyramid has frequently been used in image denoising [111, 93, 94, 39] as well as for other applications, such as texture generation [92].



(a) marginal histogram (b) conditional histogram

Figure 2.14: The marginal histogram of coefficients in a steerable pyramid decomposition has more kurtosis than a Gaussian distribution (*i.e.* it is heavier-tailed, or more "peaked"). The conditional histogram of two spatially adjacent coefficients has the shape of a bow-tie. A brighter color corresponds to higher probability. Each column in the image has been independently normalized. When a coefficient is close to zero, a neighboring coefficient is likely to be close to zero as well. The standard deviation of a coefficient scales approximately linearly with the value of a neighboring coefficient.

Two properties of steerable pyramid decompositions are particularly useful for image denoising. The first property is that the distribution of coefficients in a steerable pyramid decomposition is highly peaked, see Figure 2.14a. The other property is that edges in the input image cause a clustering in wavelet activity. This has the effect that the absolute

values of neighboring coefficients are mutually dependent. This effect resembles a bow-tie in conditional histograms, see Figure 2.14b. This property holds for spatially adjacent coefficients, but also for coefficients corresponding to the same image location in different scales or orientations.

Peaked marginal histograms: The property that marginal histograms of steerable pyramid coefficients are highly peaked was the first property to have been exploited: In [111], the shape of the marginal histogram is exploited, but the shape of the conditional histogram is ignored. Assuming a noisy observation $y = x + n$ of a pyramid coefficient x, an unbiased least squares estimate of x is given by the mean of the posterior distribution:

$$\hat{x}(y) = \int p(x|y)x\mathrm{d}x \tag{2.54}$$

$$= \frac{\int p(y|x)p(x)x\mathrm{d}x}{\int p(y|x)p(x)\mathrm{d}x}, \tag{2.55}$$

where $p(y|x)$ is given by the distribution of the noise and $p(x)$ is the prior of the signal. In [111], a generalized Laplacian is used to model the signal prior: $p(x) \propto e^{-|\frac{x}{s}|^p}$, where s and p are parameters which can be estimated directly from the noisy data. The fact that the noise is correlated in the transform domain is ignored, and assumed to be AWG. The pyramid coefficients can then be denoised by numerically integrating equation 2.55. The denoised image is obtained by inverting the steerable pyramid transform. It is shown that using this approach to denoising has a "soft" thresholding effect: Small wavelet coefficients are attenuated, while larger coefficients are preserved. Intuitively, this result makes sense: Given the highly peaked shape of the prior distribution, small coefficients are assumed to arise due to coefficients with value zero. Denoising by attenuating small wavelet coefficients is referred to as "coring" and pre-dates [111], see *e.g.* [98].

Mutually dependent neighboring coefficients: The mutual dependence of neighboring coefficients can be exploited for denoising in a number of ways [121, 94, 39]. A well-known approach is based on Gaussian scale mixtures (GSMs) [3] and is referred to as BLS-GSM [94]. The idea is to model several pyramid coefficients together, using a GSM. In [94], the coefficients in a 3×3 neighborhood in a given scale and orientation, together with the "parent" coefficient at the same location but coarser scale are modeled as a vector $\mathbf{x}$. The vector $\mathbf{x}$ is modeled as a GSM: $\mathbf{x} = \sqrt{z}\mathbf{u}$, where z is a hidden random positive scalar multiplier, and $\mathbf{u}$ is a zero-mean Gaussian random vector. z modulates the variance of the coefficients in the neighborhood. The density of $\mathbf{x}$ is then given by

$$p(\mathbf{x}) = \int p(\mathbf{x}|z)p(z)\,\mathrm{d}z \tag{2.56}$$

$$= \int \frac{\exp\left(\frac{-\mathbf{x}^T(z\mathbf{C}_u)^{-1}\mathbf{x}}{2}\right)}{(2\pi)^{N/2}|z\mathbf{C}_u|^{\frac{1}{2}}} p(z)\,\mathrm{d}z, \tag{2.57}$$

where $\mathbf{C}_u$ is the covariance matrix of vector $\mathbf{u}$. BLS-GSM seeks the Bayesian least squares estimate for the coefficient in the center of each neighborhood.

BLS-GSM begins by decomposing a noisy image using a steerable pyramid. In the transform domain, a noisy neighborhood $\mathbf{y}$ is given by:

$$\mathbf{y} = \mathbf{x} + \mathbf{n} = \sqrt{z}\mathbf{u} + \mathbf{n}, \tag{2.58}$$

where all three variables z, $\mathbf{u}$ and $\mathbf{n}$ are independent. Next, the covariance of the noise $\mathbf{C}_n$ is estimated, which can be done in different ways, the simplest of which involves computing the pyramid coefficients of delta functions in the image domain. Then, the noisy neighborhood covariance $\mathbf{C}_y$ is estimated. The density of the observed neighborhood vector $\mathbf{y}$ conditioned

on z is a zero-mean Gaussian, with covariance $\mathbf{C}_{\mathbf{y}|z} = z\mathbf{C}_u + \mathbf{C}_n$. Taking expectations over z gives us:

$$\mathbf{C}_y = \mathbb{E}\{z\}\mathbf{C}_u + \mathbf{C}_n. \tag{2.59}$$

Since we have already estimated $\mathbf{C}_n$, we can also get an estimate for the signal covariance:

$$\mathbf{C}_u = \mathbf{C}_y - \mathbf{C}_n, \tag{2.60}$$

where it is assumed that $\mathbb{E}\{z\} = 1$.

Then, the goal is to compute the Bayesian least squares estimator $\mathbb{E}\{x_c|\mathbf{y}\}$ for the central coefficient x_c of each neighborhood. $\mathbb{E}\{x_c|\mathbf{y}\}$ therefore constitutes the denoising result for each neighborhood. The Bayesian least squares estimator is given by the conditional mean:

$$\begin{aligned}
\mathbb{E}\{x_c|\mathbf{y}\} &= \int x_c p(x_c|\mathbf{y})\,\mathrm{d}x_c && (2.61)\\
&= \int\int_0^\infty x_c p(x_c, z|\mathbf{y})\,\mathrm{d}z\,\mathrm{d}x_c && (2.62)\\
&= \int\int_0^\infty x_c p(x_c|\mathbf{y}, z) p(z|\mathbf{y})\,\mathrm{d}z\,\mathrm{d}x_c && (2.63)\\
&= \int_0^\infty p(z|\mathbf{y})\mathbb{E}\{x_c|\mathbf{y}, z\}\,\mathrm{d}z. && (2.64)
\end{aligned}$$

One can therefore compute $\mathbb{E}\{x_c|\mathbf{y}\}$ by numerically integrating over z. The distribution of the multiplier, conditioned on the neighborhood $\mathbf{y}$ is computed using Bayes' rule:

$$p(z|\mathbf{y}) = \frac{p(\mathbf{y}|z)p_z(z)}{\int_0^\alpha p(\mathbf{y}|\alpha)p_z(\alpha)\,\mathrm{d}\alpha}, \tag{2.65}$$

where Jeffrey's prior is used for z: $p_z(z) \propto \frac{1}{z}$. The conditional density $p(\mathbf{y}|z)$ is a zero-mean Gaussian and is given by

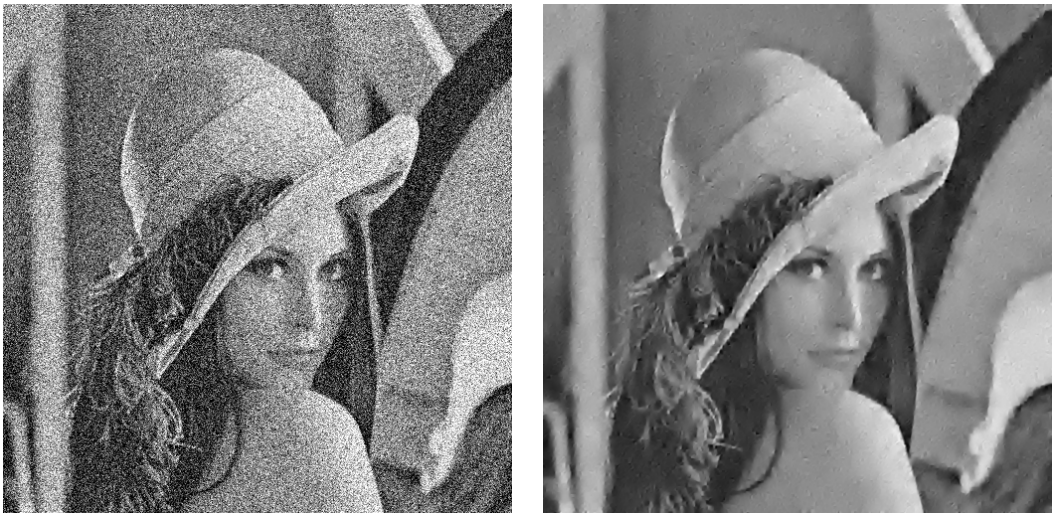
$$p(\mathbf{y}|z) = \frac{\exp\left(\frac{-\mathbf{y}^T(z\mathbf{C}_u+\mathbf{C}_n)^{-1}\mathbf{y}}{2}\right)}{\sqrt{(2\pi)^N|z\mathbf{C}_u + \mathbf{C}_n|}}, \tag{2.66}$$

where we used the fact that $\mathbf{C}_{\mathbf{y}|z} = z\mathbf{C}_u+\mathbf{C}_n$. $\mathbb{E}\{\mathbf{x}|\mathbf{y}, z\}$ can be computed using a Wiener estimate, because $\mathbf{x}$ is Gaussian when conditioned on z and the noise is additive and Gaussian:

$$\mathbb{E}\{\mathbf{x}|\mathbf{y}, z\} = z\mathbf{C}_u(z\mathbf{C}_u + \mathbf{C}_n)^{-1}\mathbf{y}, \tag{2.67}$$

where $\mathbf{C}_u$ and $\mathbf{C}_n$ have already been computed. $\mathbb{E}\{x_c|\mathbf{y}, z\}$ can be computed by using a variation of this formulation, but the exact derivation of $\mathbb{E}\{x_c|\mathbf{y}, z\}$ is quite involved and therefore left out in this exposition. Once the Bayesian least squares estimator $\mathbb{E}\{x_c|\mathbf{y}\}$ has been computed for the central coefficient of each neighborhood, the pyramid transform is inverted to obtain the denoised image.

Figure 2.15 shows the result obtained using BLS-GSM on a noisy version of image "Lena". The result compares favorably to many other algorithms.



noisy, PSNR: 14.16dB denoised, PSNR: 28.24dB

Figure 2.15: Results using BLS-GSM [94].

2.9 Dictionary-based methods

Several denoising algorithms rely on "dictionaries" for denoising image patches [75, 74, 1, 31]. It is usually assumed that an image x is corrupted with AWG noise:

$$y = x + n. \tag{2.68}$$

Denoising is performed patch-wise: Each patch of size $k \times k$ (with k usually between 8 and 12) is denoised separately and inserted into the denoised image. Usually, averaging is performed in areas of overlapping patches.

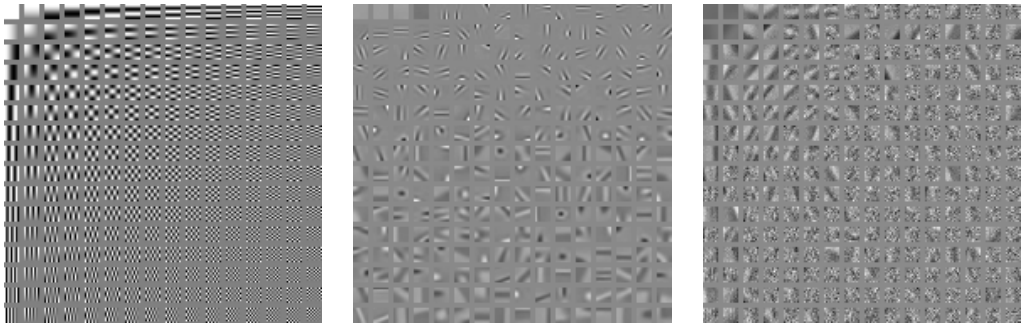
Denoising relies on an over-complete *dictionary* D of size $k^2 \times m$, where $m > k^2$. The dictionary contains a set of *atoms*, which can be thought of as basis functions. The idea underlying dictionary-based denoising methods is to denoise by approximating the noisy patch using a sparse linear combination of atoms. The problem to be solved is usually the following:

$$\min ||\alpha||_0 \text{ s.t. } ||D\alpha - y||_2 \leq \epsilon, \tag{2.69}$$

where the ℓ_0-pseudo norm enforces sparsity of the solution. The problem is combinatorial due to the ℓ_0 pseudo-norm and therefore usually impossible to solve exactly. Good approximations to the true solution can usually be found using orthogonal matching pursuit (OMP) [89], which is a greedy procedure and therefore quite fast. It is possible to replace the ℓ_0 norm with the ℓ_1 norm, in which case the problem is convex, though better solutions are usually obtained using the ℓ_0 norm. The parameter ϵ is usually chosen in such a way that the norm of the approximation error $||D\alpha - y||_2$ can be explained by the variance of the noise. Typically, $\epsilon = k^2((C\sigma)^2)$, where σ^2 is the variance of the noise and C is a hyper-parameter (typically $C = 1.15$).

The quality of the denoising result is highly dependent on the choice of the dictionary D. Three possibilities exist:

1. The dictionary can be designed,
2. the dictionary can be learned globally on a dataset of noise-free images, or
3. the dictionary can be learned adaptively from the noisy image itself .



designed learned globally learned adaptively

Figure 2.16: Various dictionaries. The designed dictionary is an over-complete DCT basis. The learned dictionary was learned using stochastic gradient descent on the Berkeley segmentation dataset [81]. The adapted dictionary was learned on the noisy Lena image using KSVD [1].

One usually observes that learned dictionaries provide better results than designed dictionaries but that the best results are achieved with dictionaries that are adapted to the noisy image at hand. An example of a designed dictionary is an over-complete DCT [31], see Figure 2.16.

Learning a dictionary from a dataset: Learning a dictionary on a dataset of noise-free images can be done by employing a gradient descent technique, such as stochastic gradient descent [72]. One attempts to approximate an image patch x using the current version of the dictionary:

$$\min ||x - D\alpha||_2 \text{ s.t. } ||\alpha||_p < L, \tag{2.70}$$

where L is a hyper-parameter and p is either 0 or 1. It is reported in [74] that better results can be achieved by learning dictionaries with the ℓ_1-norm but using the ℓ_0-pseudo-norm when denoising. One then updates the dictionary by following the gradient of the approximation error over the dictionary:

$$D \leftarrow \eta \frac{\partial ||x - D\alpha||_2}{\partial D}, \tag{2.71}$$

where η is a step size. Dictionaries learned in such a way typically contain features resembling Gabor filters, see Figure 2.16.

KSVD: Learning from the noisy image: KSVD [1] is an iterative algorithm that learns a dictionary on the noisy image at hand. One iteration of the algorithm consists of the following two steps:

1. Find the coefficients α for each patch in the image (for example using OMP)

2. Update the dictionary, one column at a time.

Usually 10 iterations are sufficient to achieve good results. The step updating the dictionary relies on an SVD-decomposition, hence the name of the algorithm. Dictionaries learned in such a way often contain features also present in the image on which the dictionary was learned, see Figure 2.16. Figure 2.17 shows a result obtained with KSVD.

noisy, PSNR: 14.16dB denoised, PSNR: 27.57dB

Figure 2.17: Results using KSVD [1].

NLSC: NLSC [74] is a dictionary-based method that is similar to K-SVD in that the dictionary is adapted to the noisy image at hand. A difference between the two methods is that NLSC employs *simultaneous sparse coding* [118], which encourages similar-looking noisy patches to be approximated using the same sparse decomposition. The idea underlying the approach is that images contain self-similarities: Similar-looking patches are expected to be found at several locations in an image. This idea is exploited by a number of other algorithms and is discussed in more detail in Section 2.11. Figure 2.18 shows a result obtained using NLSC. NLSC is one of the best currently available denoising algorithms in terms of quality of the results, but requires long computation times.

noisy, PSNR: 14.16dB denoised, PSNR: 28.95dB

Figure 2.18: Results using NLSC [74].

2.10 EPLL

Many denoising methods denoise image patches independently and apply averaging or other similar techniques in areas of overlapping patches. Dictionary denoising method such as KSVD [1] are examples of such methods. The problem with this approach is that the averaging process can create patches in the denoised images that do not look good.

EPLL [132] is an acronym from expected patch log likelihood. The method contrasts itself from methods that denoise patches independently by aiming at creating a denoised image in which each patch is likely under a given patch prior, while staying close to the noisy image. EPLL takes a maximum a posteriori (MAP) approach to denoising: Given an image corrupted with AWG noise $\mathbf{y} = \mathbf{x} + \mathbf{n}$, we want to find $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\mathbf{y}) \tag{2.72}$$

$$= \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) \tag{2.73}$$

$$= \arg\min_{\mathbf{x}} -\log(p(\mathbf{y}|\mathbf{x})) - \log(p(\mathbf{x})) \tag{2.74}$$

$$= \arg\min_{\mathbf{x}} -\log \frac{\lambda}{2}||\mathbf{x} - \mathbf{y}||^2 - \text{EPLL}(\mathbf{x}), \tag{2.75}$$

where λ controls the trade-off between the prior and the data-fidelity term, as usual in MAP estimation. The expected patch log-likelihood (EPLL) is defined as:

$$\text{EPLL}(\mathbf{x}) = \sum_i \log p(\mathbf{P}_i\mathbf{x}), \tag{2.76}$$

where $\mathbf{P}_i$ extracts patch i out of an image. The EPLL is therefore the sum over the expected patch log-likelihoods of all sliding window patches in an image. The EPLL is not the expected log-likelihood of a full image.

Optimization is performed using half-quadratic splitting, which introduces auxiliary variables and alternates between two steps: (i) updating the auxiliary variables while keeping the image patches fixed, and (ii) updating the image patches while keeping the auxiliary variables fixed. This procedure is repeated for a small number of iterations (four or five in the paper).

We see that the method does not depend on a specific image prior: In principle, any probabilistic patch prior could be used. An advantage of the method is that one need not learn a prior on entire natural images, as other methods such as Fields of Experts attempt to do. Instead, one need only learn a prior on natural image patches, which is considerably easier.

Though the method can theoretically use any probabilistic patch prior, the best results achieved in the paper are obtained using a Gaussian mixture model (GMM):

$$\log p(\mathbf{x}) = \log \left(\sum_{k=1}^{K} \pi_k N(\mathbf{x}|\mu_k, \Sigma_k) \right), \tag{2.77}$$

which was trained using expectation maximization (EM) on 2×10^6 natural image patches using $K = 200$ Gaussians and patches of size 8×8 pixels. All parameters are learned, with full covariance matrices.

noisy, PSNR: 14.16dB

denoised, PSNR: 28.42dB

Figure 2.19: Results using EPLL [132].

The results described in the paper are comparable to those achieved with other state-of-the-art methods such as BM3D and NLSC. The result shown in Figure 2.19 demonstrates that good results are indeed achievable with this method.

2.11 BM3D

(a) (b)

Figure 2.20: Block matching applied on a noisy version of image Lena. Image (a) shows the whole image, image (b) shows only a smaller region. The goal of the procedure is to find the patches most similar to the reddish (reference) patch. The neighbors (blueish patches) have to be found within a search region (represented by the larger black bounding box). Patches can overlap. In BM3D, the red and blue patches would be denoised together.

BM3D [25] is often considered the state-of-the-art in image denoising. The name stands for "block-matching and 3D-filtering". The method assumes that a clean image x is corrupted with AWG noise:

$$y = x + n. \tag{2.78}$$

The procedure does not explicitly rely on an image prior, but assumes the images to contain similar-looking patches. In other words, given a "reference" patch in an image, it is likely that similar patches exist elsewhere in the same image. Such similar-looking patches are grouped in a procedure called block-matching, see Figure 2.20. The group of patches can be regarded as a three-dimensional block. The patches in a block are denoised together and collaboratively: Each patch helps to denoise the other patches. The algorithm proceeds patch-wise: For each patch in the image, neighbors are found and grouped into a block, denoised together, and re-inserted into their original locations. Each image patch can belong to several blocks, which is different from clustering, where each patch can only belong to one cluster.

Two types of transforms are applied on the block. The resulting coefficients are shrunk, followed by the inverse of the transforms. This has the effect of denoising all patches in the block. The denoised patches are inserted into the denoised image at the location at which their noisy counterparts were found.

The main idea of the algorithm is that within a block, the image information is correlated both within each patch and between patches (*intra*-patch correlations and *inter*-patch correlations), whereas the noise is white and therefore not correlated. The first transform is a 2D-transform such as a DCT-transform, which has the effect of exploiting intra-patch correlations: A few coefficients of each patch will have large value. The second transform is a 1D-transform such as the Haar-transform. This transform is applied along the third dimension of the block: The same DCT-coefficient of all patches are transformed. This has the effect of exploiting inter-patch correlations: A few Haar coefficients will have large value. The crucial point is that the noise is white both before and after application of the

transforms. The image information however has been concentrated into a few coefficients with high values. This allows for effective denoising via shrinkage of coefficients.

The algorithm relies on two steps. The difference between the two steps lies in how similar-looking patches are found and how coefficients are shrunk. In the first step, similar-looking patches are found in the noisy image itself, as in Figure 2.20. Hard-thresholding is used as a shrinkage procedure: All values below a threshold are set to 0. This first step already provides a denoised image. In the second step, the denoised image obtained using the first step is exploited. Similar looking-patches are found using the denoised image from the first step, which is more reliable than looking for similar-looking patches in the noisy image. Two blocks are formed: A block containing the noisy patches and a block containing the corresponding denoised patches from the first step. The same transforms are applied on both blocks. Shrinkage is then achieved using Wiener filtering: One attempts to approximate the block of denoised patches by weighting the block of noisy patches. The second step of the procedure is important and often leads to significant gains over the first step. The method is

noisy, PSNR: 14.16dB denoised, PSNR: 29.05dB

Figure 2.21: Results using BM3D [25].

relatively computationally intensive: For each patch, matching neighbors have to be found and denoised. Both operations are time-consuming. The algorithm uses a few practical tricks to reduce the computational burden. For example, not every image patch is used as a reference patch; the algorithm rather skips a few pixels between reference patches. Also, rather than searching for neighboring patches in the whole image, search is restricted to a small window (see Figure 2.20). In practice, the algorithm is quite fast: Denoising an image of size 512×512 takes on the order of 5 seconds on a modern computer. The method often achieves outstanding results (see Figure 2.21) and is considered state-of-the-art.

2.12 Method taxonomy and discussion of the various approaches

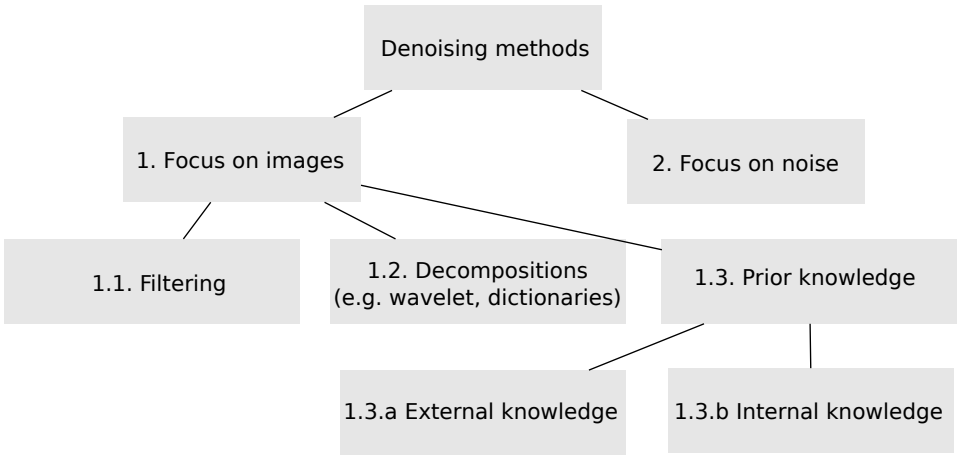


Figure 2.22: A possible taxonomy of denoising methods.

We propose to classify denoising methods according to the hierarchy depicted in Figure 2.22. Denoising methods follow one of the following two paradigms:

1. **Focus on images:** Methods making simple assumptions about the noise, and focusing instead on the properties of images.

2. **Focus on noise:** Methods making simple assumptions about images, and focusing instead on the properties of the noise.

In this section, we have described a number of denoising approaches following the first paradigm. The noise is usually assumed to be AWG, whereas the images to be denoised are assumed to contain more structure. This has become the standard setting in image denoising, where the images to be denoised are so-called "natural" images, or images of every-day scenes. However, certain situations might require a different approach. In Section 4 we will discuss denoising of astronomical images, which follow statistics that are different from natural images. An additional difference is that astronomical images are usually taken with long exposure times. The noise arising in such settings is different from AWG noise and contains more structure. In such situations, it often makes sense to focus on the properties of the noise rather than on the properties of the images. The first denoising paradigm is the approach taken by most algorithms. Few approaches follow the second paradigm (an example of such an approach can be found in [45]).

The denoising algorithms belonging to the first paradigm can be further sub-divided into the following categories:

1.1. Filtering: Linear and non-linear filtering-based methods, such as linear smoothing, median filtering, bilateral filtering.

1.2. Decompositions: Methods based on wavelet or dictionary decompositions of the image.

1.3. Prior knowledge: Methods based on global image statistics, or other image properties, such as self-similarities.

Algorithms cannot always be clearly assigned to one category and might sometimes belong to several categories (*e.g.* Fields of Experts can be regarded as a filtering-based method, but

also uses prior knowledge about images). Linear smoothing, median filtering, MATLAB's `wiener2` function, anisotropic diffusion and bilateral filtering can all be considered to be filtering-based methods. BLS-GSM and KSVD are based on a decomposition of the image (though KSVD can also be said to rely on prior knowledge). Fields of Experts, EPLL, BM3D and NLSC rely on prior knowledge about images.

Methods based on prior knowledge about images can be further divided into:

1.3.a External knowledge Methods based on knowledge about all images.

1.3.b Internal knowledge Methods based on knowledge about the image to be denoised.

By external knowledge, we mean knowledge regarding the all natural images, whereas by internal knowledge we mean knowledge gained from the noisy image itself. Fields of Experts and EPLL belong to the category exploiting external knowledge, whereas BM3D and NLSC belong to the category exploiting internal knowledge.

Methods based on internal knowledge are a more recent development than methods based on external knowledge. The idea underlying methods exploiting internal knowledge is to look for regions within an image that are similar in appearance. A simple example of how to exploit this idea is to average pixels with a similar grey value within a given spatial neighborhood. This is an idea resembling bilateral filtering [117], but was originally proposed in [129]. More recently, this idea was extended to image patches: Given an image patch, one looks for similar patches within the same image. This idea was first used for texture synthesis [30]. The same idea can be used for image denoising: The NL-means approach [9] looks for similar patches within a given noisy image and performs a weighted average of the center pixels for denoising. BM3D [25] also exploits the idea of grouping patches that are similar in appearance, but performs a more effective denoising step on the group of patches than NL-means [9]. NLSC [74] is a further example of a method exploiting this idea: Similar-looking patches are grouped and denoised together using simultaneous sparse coding, where a dictionary is used that is adapted to the noisy image itself.

Dictionary-based methods can belong to either the category exploiting external or internal knowledge, depending on what kind of dictionary is used for sparse representation. If the dictionary is learned on a large dataset of images (such as the *global* setting in [31]), the method belongs to the category of methods exploiting external knowledge. If the method learns a dictionary on the noisy image at hand, such as KSVD [1], the method belongs to the category of methods exploiting internal knowledge. NLSC employs a dictionary that is learned on a dataset of noise-free images, but adapts this dictionary to the noisy image at hand, similarly to KSVD. NLSC can therefore be said to belong both to the category employing internal and to the category employing external knowledge. In case the dictionary is hand-crafted (as opposed to learned on either a larger dataset or a single noisy image), the method belongs to the same category as methods based on wavelet-decompositions: They rely on a decomposition of the image that is useful, but do not rely on learning.

Strengths and weaknesses of the approaches: Denoising methods vary in terms of denoising results and in terms of the required computation times to denoise a given image. The computationally most intensive methods learn or adapt a dictionary: KSVD [1] and NLSC [74] require long computation times (up to an hour on a modern machine for images of size 512×512). Some filtering-based methods such as `wiener2`, but also BLS-GSM and BM3D are particularly fast and require a few seconds of computation for images of size 512×512. Most methods lie somewhere in the middle of these two extremes *e.g.* EPLL requires approximately five minutes for images of size 512×512.

BM3D, EPLL and NLSC perform better than other methods (*e.g.* BLS-GSM) on average. However, the methods have complementary strengths and weaknesses: Methods relying on internal knowledge are particularly effective on images with repeating structure, whereas methods relying on external knowledge are usually more effective on images with more complex structures which are not identical within the same image, see Figure 2.23. It

Image “Man”

Image “Barbara”

Figure 2.23: Methods based on external knowledge such as EPLL [132] tend to perform well on images with complex structures (*e.g.* image “Man”) , whereas methods based on internal knowledge such as BM3D [25] and NLSC [74] tend to perform well on images with regular structure (*e.g.* image “Barbara”).

has been noted elsewhere [58] that denoising methods have complementary strengths and weaknesses.

2.13 Contributions of this thesis

Image denoising is a long-standing problem and has been thoroughly studied. In this section, we describe three areas in which we were able to contribute to the field of image denoising and give an overview of the remaining chapters of this thesis.

Figure 2.24: At higher noise levels, denoised images often contain cloudy-looking artifacts. These are due to improper handling of lower frequencies. Result obtained with KSVD, noisy image contained AWG noise with $\sigma = 50$.

High noise settings: Applying denoising methods on images contaminated with higher levels of noise often lead to "cloudy" artifacts, see Figures 2.17, and 2.24. Such artifacts occur due to improper handling of lower frequencies. Can this situation be avoided?

We will see in Chapter 3 that many denoising methods are based on small patches or small filters. Thus, they naturally focus only on high frequencies. This leads to a deterioration of performance at higher noise levels, because AWG noise corrupts all frequencies. We will demonstrate that methods that improperly handle low frequencies can be combined with a meta-procedure in such a way as to also properly handle low frequencies. No modification of the existing methods is required for the meta-procedure to work. A limitation of the approach is that not all methods can be improved on: Some methods already properly handle low frequencies.

Astronomical images: In Section 2.12 above, we mentioned that most denoising methods belong to category focussing on images instead of noise. We ask the question: When is it useful to consider the paradigm focussing on noise? In Chapter 4 we consider astronomical images, where expossure times are typically long and ISO settings high. The noise under such settings is different for every pixel of the camera's sensor. Hence, it becomes interesting to study and model the noise. We will see that doing so indeed leads to better denoising results for astronomical images.

Learning vs. engineering, internal vs. external knowledge: Most denoising methods rely on cleverly designed algorithms. This is also the case for methods relying on a learning component: Fields of Experts [99] and EPLL [132] are two examples of methods that rely on learning to capture statistics about natural images (or natural image patches for EPLL) but require sophisticated methods to exploit this knowledge. Thus a question arises: Is it possible to rely more on learning and less on engineering?

A further observation is that some of the best-performing denoising methods heavily rely on what we refer to as internal knowledge. BM3D [25] is a prime example: The method is considered state-of-the-art, and relies entirely on internal knowledge. We therefore ask the question: Is it also possible to achieve excellent results with external knowledge?

noisy, PSNR: 14.16dB

denoised, PSNR: 29.34dB

Figure 2.25: Results using an MLP, see Chapter 5. This result is better than the result achieved with all previously described methods, including BM3D and NLSC. In Chapter 5, we will see that MLPs outperform all competitors on most images.

In Chapter 5, we answer both these questions with yes. We demonstrate that it is possible to outperform the previous state-of-the-art in image denoising with multi-layer perceptrons (MLPs, a type of neural network) trained to denoise image patches. This method requires little engineering and relies fully on external knowledge. Figure 2.25 shows a result obtained with such an MLP.

An additional question begs to be asked: Is it possible to combine internal and external knowledge in order to achieve even better results? In Section 5.9 we propose to combine the results of several denoising algorithms, using an MLP. By combining the results of BM3D and an MLP with a second MLP we combine internal and external knowledge and indeed achieve better results. The output of this method is usually superior to the best of the inputs. Figure 2.26 shows an example of results achieved using this method.

Training MLPs is sometimes considered more of an art than a science: The choice of the architecture as well as other parameters may seem arbitrary. In Chapter 6, we describe the training procedure of our MLPs in detail and discuss which trade-offs have to be considered in order to achieve good results. An additional criticism regarding MLPs is that they are often seen as "black boxes": One usually does not understand their inner workings. We show that it is possible to gain some understanding about how MLPs achieve denoising. Put simply, the MLPs work as feature detectors and generators: The MLPs "look for" certain features in the noisy input patch and, if found, copy these same features into the output patch. The noise is removed through saturation of the sigmoid layers within the MLPs.

noisy, PSNR: 14.16dB denoised, PSNR: 29.48dB

Figure 2.26: Results using an ensembling MLP (E-MLP), see Section 5.9. The E-MLP efficiently combines the result obtained with BM3D and another MLP.

3

Improving existing denoising methods using a multi-scale meta-procedure

Chapter abstract Many state-of-the-art denoising algorithms focus on recovering high-frequency details in noisy images. However, images corrupted by large amounts of noise are also degraded in the lower frequencies. Thus properly handling all frequency bands allows us to better denoise in such regimes. To improve existing denoising algorithms we propose a meta-procedure that applies existing denoising algorithms across different scales and combines the resulting images into a single denoised image. With a comprehensive evaluation we show that the performance of many state-of-the-art denoising algorithms can be improved.

The material of this chapter is based on the following publication:

[11] H.C. Burger, and S. Harmeling. Improving denoising algorithms via a multi-scale meta-procedure. Proceedings of the 33rd international conference on Pattern recognition (DAGM). 2011.

This paper was awarded with a prize at DAGM 2011.

noisy
$\sigma = 200$, PSNR: 7.59dB

BM3D [25]
PSNR: 18.88dB

MS-BM3D (our approach)
PSNR: 20.96dB

Figure 3.1: In high noise settings, our approach improves the results achieved with BM3D.

3.1 Introduction

The problem of removing noise from natural images has been extensively studied, so methods to denoise natural images are numerous and diverse. [34] classifies denoising algorithms into three categories: The first class of algorithms rely on smoothing parts of the noisy image [101, 126, 117] with the aim of "smoothing out" the noise while preserving image details. The second class of denoising algorithms exploit learned image statistics. A natural image model is typically learned on a noise-free training set (such as the Berkeley segmentation dataset) and then exploited to denoise images [100, 127, 57]. In some cases, denoising might involve the careful shrinkage of coefficients. For example [111, 21, 91, 94] involve shrinkage of wavelet coefficients. Other methods denoise small images patches by representing them as sparse linear combinations of elements of a learned dictionary [31, 75, 74]. The third class of algorithms exploits the fact that different patches in the same image are often similar in appearance [25, 34].

Denoising algorithms are usually evaluated on their ability to remove additive white Gaussian noise (AWGN). Standard test images exist for this purpose. The most popular performance measure is arguably the peak signal to noise ratio (PSNR), which is related to the mean squared error (MSE).

Hypothesis: We speculate that most denoising algorithms focus on removing noise on the higher frequencies and thus are often best suited for recovering fine-scale information. Wiener filtering, bilateral filtering [117], but also the fields of experts approach [100] rely on relatively small filters to denoise images. The small size of these filters causes these approaches to ignore larger-scale information. Denoising approaches based on dictionaries such as [31] typically decompose the image into small patches and then denoise the patches separately and independently. Larger-scale structure is lost when the image is decomposed into small patches. So we hypothesize that many denoising algorithms can be improved by employing a multi-scale approach.

Assumption: Our approach assumes that the statistics of natural images are invariant to changes in spatial scale. An intuitive justification for this assumption is that scenes are about equally likely to be viewed from different distances. This assumption has been successfully exploited by others [94].

Contributions: We present a meta-procedure than can be used in combination with existing denoising methods, yet often improves the results. We choose algorithms from all three categories to show that our procedure is versatile. We evaluate the PSNR on a set of 13 standard test images with varying amounts of added noise. In most cases, we use commonly available implementations of these algorithms.

Related work: Besides the denoising method mentioned above, there is a procedure that is relatively similar to ours. In [34], the authors introduce the "stochastic denoising" procedure and propose an extension (called "multi-pass denoising") in order to handle "larger-scale" noise (*i.e.* noise that is not uncorrelated across neighboring pixels). The extension is similar to our meta-procedure in that in addition to the original image, a single down-scaled version is denoised. The down-scaled denoised image is up-scaled and combined with the denoised image of the original size. Different from our method is that the authors combine the images using a pixel-specific linear blend between the two images. The ratio of the blend is controlled by the gradient of the image at that pixel. No quantitative evaluation was provided in [34], but we include it in our evaluation. [95] also considers a multiscale approach for image denoising by thresholding coefficients in different frequency bands.

3.2 Down-scaling has a denoising effect

When an image that has been corrupted with AWGN is down-scaled (low-pass filtered followed by down-sampling), the image becomes more recognisable. The effect is illustrated in Fig. 3.2: adding a large amount of Gaussian noise leaves the "Lena" image barely recognisable (upper left). Nonetheless, the down-scaled version (upper middle) seems to contain much less noise.

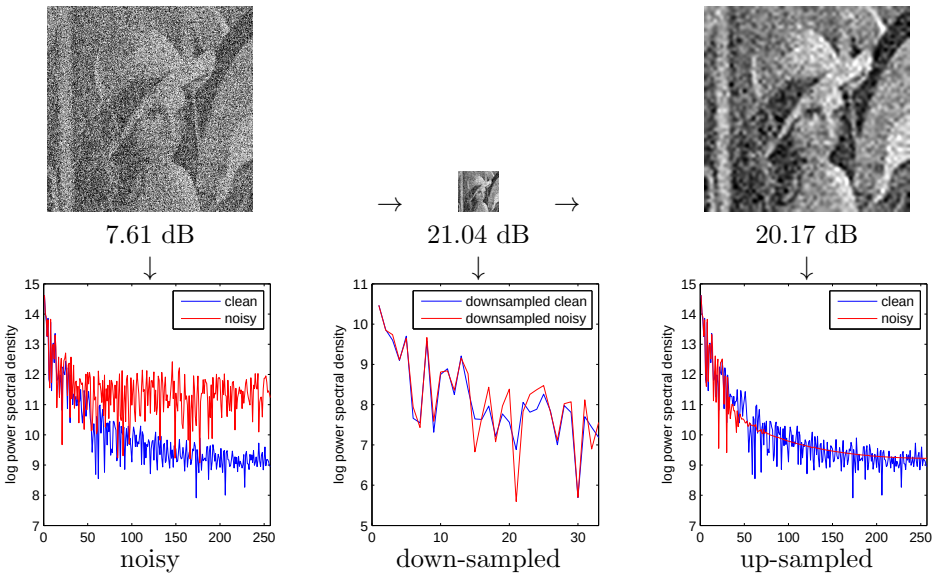


Figure 3.2: Noisy Lena becomes less noisy by down- and up-sampling (top row) with power spectra (bottom row).

Down-scaling an image effectively averages neighboring pixel values, causing the uncorrelated values of the noise to become smaller. Since neighboring pixels in natural images are often highly correlated, the down-scaling process is not that damaging to the image information. Another explanation is that natural images have the most energy in the low frequencies whereas AWGN is uniformly spread over the whole spectrum. Down-scaling an image keeps mainly the low frequencies, which are precisely the frequencies where the image information is strongest (bottom row in Fig. 3.2). Nonetheless, if the amount of noise is very large, frequencies in the middle of the spectrum are also affected, so the image information in lower frequencies should also be denoised.

3.3 How to denoise lower frequencies

We imagine a hypothetical scenario in which we wish to recover the low frequencies of a noisy image as best as possible. To evaluate how well we recovered the low frequencies, we compare the resulting image to a down-scaled version of the ground truth image. We compare two approaches:

1. First denoise, then down-scale the result.
2. First down-scale, then denoise.

Which approach is better? In the first approach, the denoising algorithm has more information available, while in the second approach the denoising algorithm is applied to the down-sampled version. Denoising a down-scaled image should be an easier task, which would suggest that the second approach is better. If the second approach achieves better results, we could conclude that denoising algorithms are not good at recovering large-scale (*i.e.* low frequency) information, confirming the hypothesis we advanced in the introduction.

Fig. 3.3 compares the two approaches using KSVD as the denoising procedure. Comparing the achieved PSNRs we see that the second approach is preferable to the first. This effect also holds for other denoising algorithms for a variety of different noise settings, see Figure 3.4.

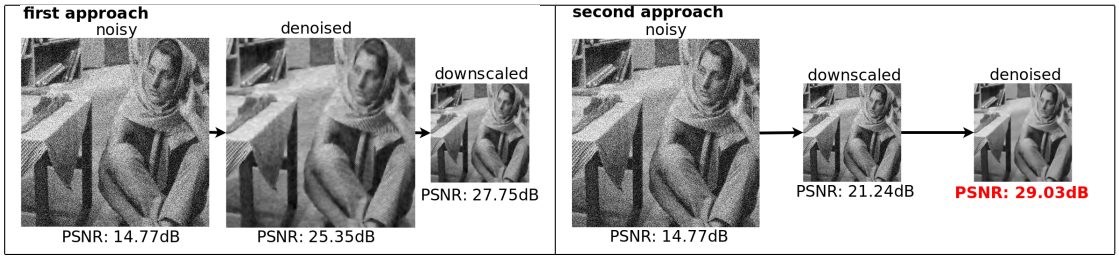


Figure 3.3: Which approach better recovers the low frequencies? First down-scaling, then denoising is better than the other way around when the noise is strong.

Thus we can conclude that if we wish to recover low-frequency information with a denoising algorithm that is not designed to recover low frequencies, down-scaling the image might help. Effectively the down-scaling transforms the low-frequency information into high-frequency information which can be accessed by the denoising algorithm. In the following we show how this insight can be exploited with a multi-scale procedure such that the high-frequencies are recovered from the given noisy image, while we get the low frequencies from a down-scaled version of it.

3.4 Multi-scale denoising

We propose a meta-procedure that relies on denoising not only the original noisy image, but also down-scaled versions of that image. This meta-procedure is formulated such that it can be combined with any existing denoising algorithm. The last step of our procedure consists in combining the denoised images at the different scales. The combination is motivated by Laplacian pyramids. Fig. 3.5 summarizes our method graphically.

We will denote by $d_\alpha(x)$ a procedure that down-scales the image x by the factor α. Similarly, we denote by $u_\alpha(x)$ the procedure that up-scales the image x by the factor α. Re-scaling an image involves a filtering step, followed by re-sampling. In practice, we applied Matlab's `imresize` function with the Lanczos-3 kernel. Other kernels do not lead to significantly different results.

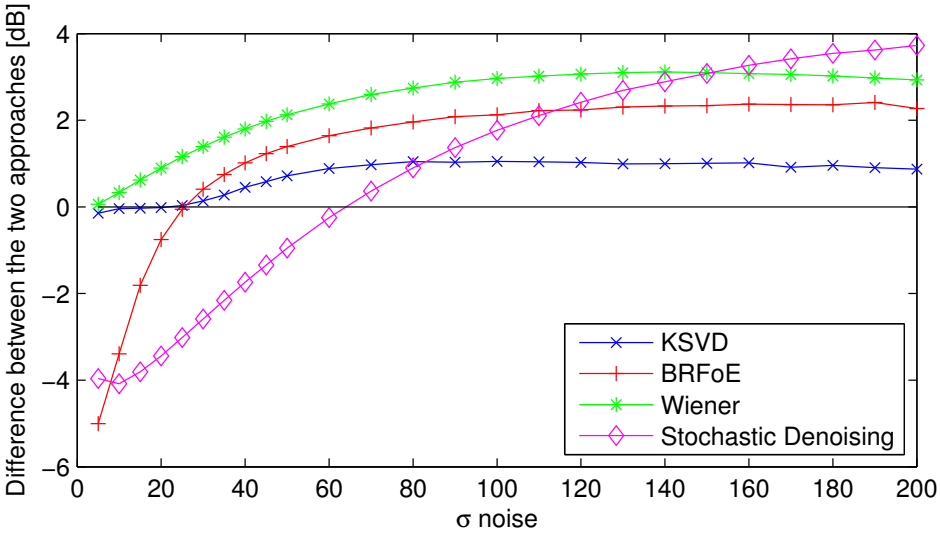


Figure 3.4: The effect observed in Figure 3.3 holds for a number of denoising algorithms. A value larger than 0 indicates that it is better to first down-scale, then denoise than vice-versa. So we see that first down-scaling, then denoising is better than the other way around when the noise is strong.

Note that resizing is a linear operator which can be represented as a matrix D. The covariance matrix of downsampled Gaussian noise is proportional to DD^T which is approximately the identity matrix for most resampling kernels (*e.g.* Lanczos). This fact implies that the AWGN assumption also holds for downsampled images.

Denoising at different scales. As parameters to our procedure we initially choose a denoising algorithm and scaling factors $\alpha_1, \ldots, \alpha_n$ (sorted in ascending order). Given a noisy image x_0, we create n down-scaled versions $x_1, \ldots, x_n$,

$$x_1 = d_{\alpha_1}(x_0); \quad \cdots \quad x_n = d_{\alpha_n}(x_0). \tag{3.1}$$

The images $x_0, \ldots, x_n$ are subsequently denoised using the same denoising procedure:

$$y_0 = \text{denoise}(x_0); \quad \cdots \quad y_n = \text{denoise}(x_n). \tag{3.2}$$

Next we combine the $n+1$ denoised images $y_0, \ldots, y_n$ in a Laplacian-pyramid fashion to obtain the best possible denoised image z_0 (which will have the same size as the input image x_0).

Recombining the images on the different scales. For this we decompose the image y_i into low and high frequency components l_i and h_i:

$$l_i = d_{\alpha_i/\alpha_j}(y_i) \qquad\qquad h_i = y_i - u_{\alpha_j/\alpha_i}(l_i). \tag{3.3}$$

Next, the low frequency information l_i is discarded and replaced by y_{i+1}, which has the same size as l_i. We do so because y_{i+1} contains more accurate low-frequency information. Combining y_{i+1} and h_i we obtain a reconstruction z_i at level i:

$$z_i = h_i + u_{\alpha_j/\alpha_i}(y_{i+1}) \tag{3.4}$$

which combines the best of y_i and y_{i+1}, *i.e.* the high frequencies of y_i and the low frequencies from y_{i+1}.

As common for Laplacian pyramids, we start the multi-scale reconstruction with the two smallest images y_{n-1} and y_n and proceed through all scales until we reconstruct the image z_0 which is the denoising result of our method.

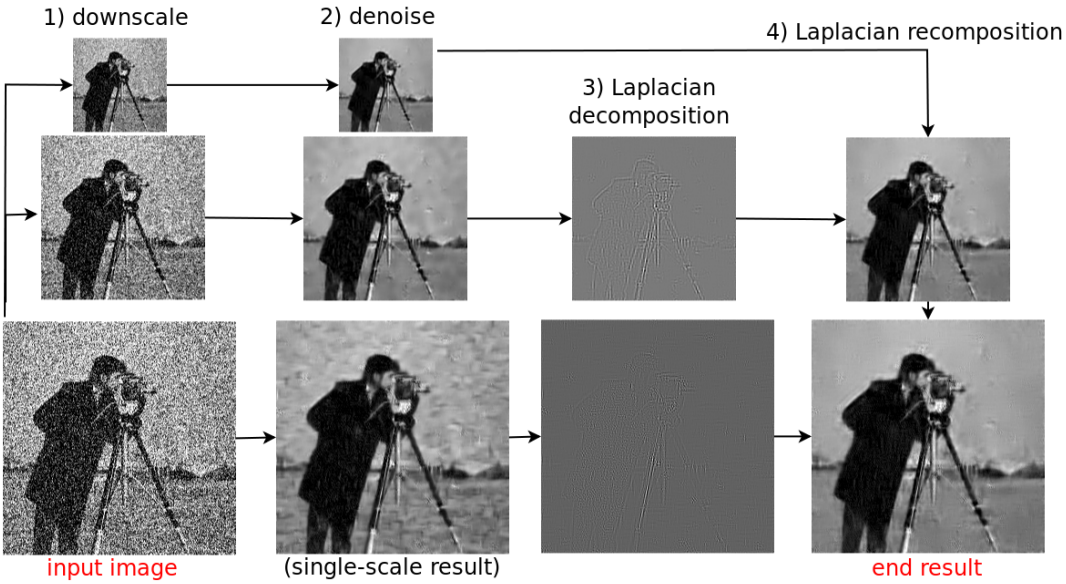


Figure 3.5: Our procedure denoises a noisy image at different scales and then combines these images similarly to Laplacian pyramids.

Shrinking high frequency coefficients. The right panel in Fig. 3.6 shows the benefit of using the proposed multi-scale meta-procedure with two scales in combination with the KSVD denoising algorithm. At noise levels above $\sigma = 25$, the meta-procedure (MS-KSVD, no thresholding, line '—x—') improves the results over the plain denoising algorithm (solid line). At first, the improvement grows with growing noisiness. However, when the noise becomes very strong, this effect is reversed: The multi-scale meta-procedure helps less and less. This effect is due to the fact that the high-frequency components z_i are beneficial in lower noise settings, but detrimental at higher noise levels. At very high noise levels, the denoising algorithm becomes incompetent at recovering high-frequencies. A possible solution to the problem is to attenuate the values in the high-frequency image z_i in such a way as to keep only the strongest components. We replace Eq. (3.4) by:

$$z_i = \mathcal{T}(h_i, \lambda) + u_{\alpha_j/\alpha_i}(y_j) \;, \tag{3.5}$$

where $\mathcal{T}(h_i, \lambda)$ is the hard-thresholding operator with threshold λ. Other attenuation methods lead to similar results.

The three images on the left of Fig. 3.6 show the effect of the hard-thresholding operator on a high-frequency image: The smaller values in the high-frequency are mostly due to errors in the denoising procedure and are successfully removed by the thresholding operation. The larger values however are unlikely to be due to errors in the denoising procedure and are therefore kept.

3.5 Experimental evaluation and results

Our meta-procedure is sensitive to the threshold parameter λ as well as to the sizes and numbers of scales used in the Laplacian pyramid. We tuned those hyperparameters for each considered denoising algorithm and for each noise level σ on a training set of 20 images from the Berkeley segmentation training dataset, see Figure 3.7. The smallest number of scales is 1 (no multi-scale approach) and the largest is 4. The scale sizes we chose are $(1/2)^k$ with $0 \leq k \leq 3$. This corresponds to repeatedly down-scaling by a factor of two.

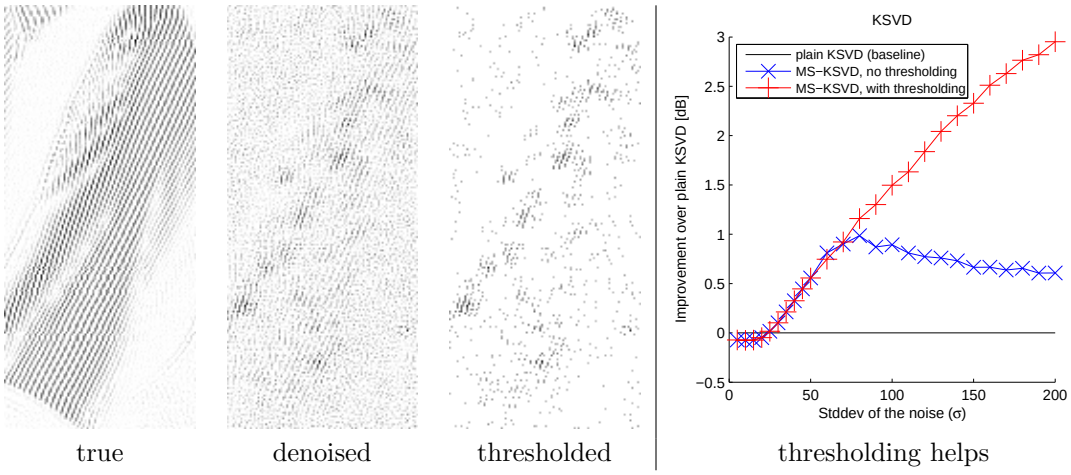


Figure 3.6: Left: high frequencies of the clean image (lower-right corner of "Barbara"). Center: high frequency image of the denoised image (recovered from noisy image with $\sigma = 100$). The image contains mostly noise, but Barbara's pants are discernible. Right: thresholded high frequency image. Structure from the pants is kept. Panel on the right shows that thresholding helps.

As the test set, we used the 13 standard gray-scale images commonly known as: "Barbara", "Boat", "Cameraman", "Couple", "Fingerprint", "Flintstones", "Hill", "House", "Baboon", "F16", "Lena", "Man" and "Peppers" (see Figure 3.7).

We applied our meta-procedure to nine state-of-the-art denoising algorithms whose implementations are commonly available. (1) Wiener filtering using Matlab's `wiener2` function with the default neighborhood size of 3. (2) Bilateral filtering [117][1] with three hyper-parameters that need to be set. Empirically, we found 10 to be a good value for the half-size of the Gaussian bilateral filter window. We chose $\sigma_1 = 3$ and set σ_2 between 10^{-4} and 2.2 depending on the noisiness of the image. (3) Bayesian least-squares Gaussian scale mixtures (BLS-GSM) [94][2], (4) Stochastic denoising [34][3], (5) Block-matching 3D (BM3D) [25][4], (6) Fields of Experts (FoE) [100][5], (7) Basis roation fields of experts (BRFoE) [127][6], and (8) Total variation denoising (TV) [101][7] all have implementations publicly available online. We used the default parameters for all methods except for FoE, where we were able to improve results over the publicly available implementation by adapting the number of iterations to the amount of noise in the image. We used our own implementation for (9) KSVD [31]. We found 10 iterations for training the dictionary to be sufficient.

Improvements for varying noise levels. Fig. 3.8 reports for various noise levels σ the difference between the results obtained in the single scale setting (denoted "baseline ...") compared to our multi-scale meta-procedure (denoted "MS-..."). We also included results obtained with the "multi-pass" procedure proposed in [34] (denoted "Estrada-..."). The integer values from one to four along the line of our multi-scale procedure ("—x—") indicate the number of scales applied.

When the noise level is low, in most cases our multi-scale meta-procedure does not improve the results of the baseline algorithm. In fact, the results are in those cases identical

[1] `http://www.mathworks.com/matlabcentral/fileexchange/12191`
[2] `http://decsai.ugr.es/~javier/denoise/software/`
[3] `http://www.cs.utoronto.ca/~strider/Denoise/`
[4] `http://www.cs.tut.fi/~foi/GCF-BM3D/`
[5] `http://www.gris.informatik.tu-darmstadt.de/~sroth/research/foe/index.html`
[6] `http://www.cs.huji.ac.il/~yweiss/BRFOE.zip`
[7] `http://visl.technion.ac.il/~gilboa/PDE-filt/tv_denoising.html`

training images test images

Figure 3.7: Training images (Berkeley segmentation dataset) for setting the hyper parameters, and test images.

to the baseline algorithm. This happens when our multi-scale approach employs only the original scale (leading to the original denoising algorithm), indicating that in those noise regimes, it was not beneficial to use more scales in the training set.

The largest improvement achieved with our multi-scale meta-procedure occurs when the noise becomes stronger, which corrupts the low frequencies more and more. The improvement is particularly dramatic for Wiener and BRFoE (more than 8dB), which are patch-based methods that ignore the lower frequencies. Also KSVD is a method that is based on small patches, which also makes it blind to low frequencies, explaining the improvements obtained. However, some algorithms cannot be improved, such as BLS-GSM and Total Variation. This can be explained by the fact that BLS-GSM is a wavelet method and therefore already a multi-scale algorithm. So we see that a limitation of our meta-procedure is that it is only useful to apply it to denoising methods which are not already considering lower frequencies.

Note that our proposed meta-procedure outperforms the procedure by Estrada et al. [34] in almost all cases. Furthermore, our approach almost never deteriorates the denoising results, which sometimes happens for Estrada's method, especially when the noise is low. The improvements are reported in terms of PSNR, but we observed similar improvements in the structural similarity index [123].

In Figures 3.10, 3.11, and 3.12 we show the results obtained with all nine denoising algorithms on images Lena, Barbara and Peppers with $\sigma = 130$. For KSVD it is especially clear that low frequencies are not properly handled: Areas that should be smooth have a "cloudy" appearance. Our multi-scale approach strongly reduces these artifacts. In images Lena and Peppers, FoE produces blurry results. Our multi-scale approach produces sharper results (see *e.g.* the eyes in image Lena). A similar effect can be observed with KSVD on image Barbara: The pattern of the tablecloth is not visible without our multi-scale extension, but is apparent with our multi-scale extension. Wiener filtering produces results that look noisy, but the multi-scale extension greatly improves the results. In fact, MS-Wiener achieves better results than KSVD on these three images.

KSVD vs. BLS-GSM revisited In [31], the KSVD denoising algorithm is compared

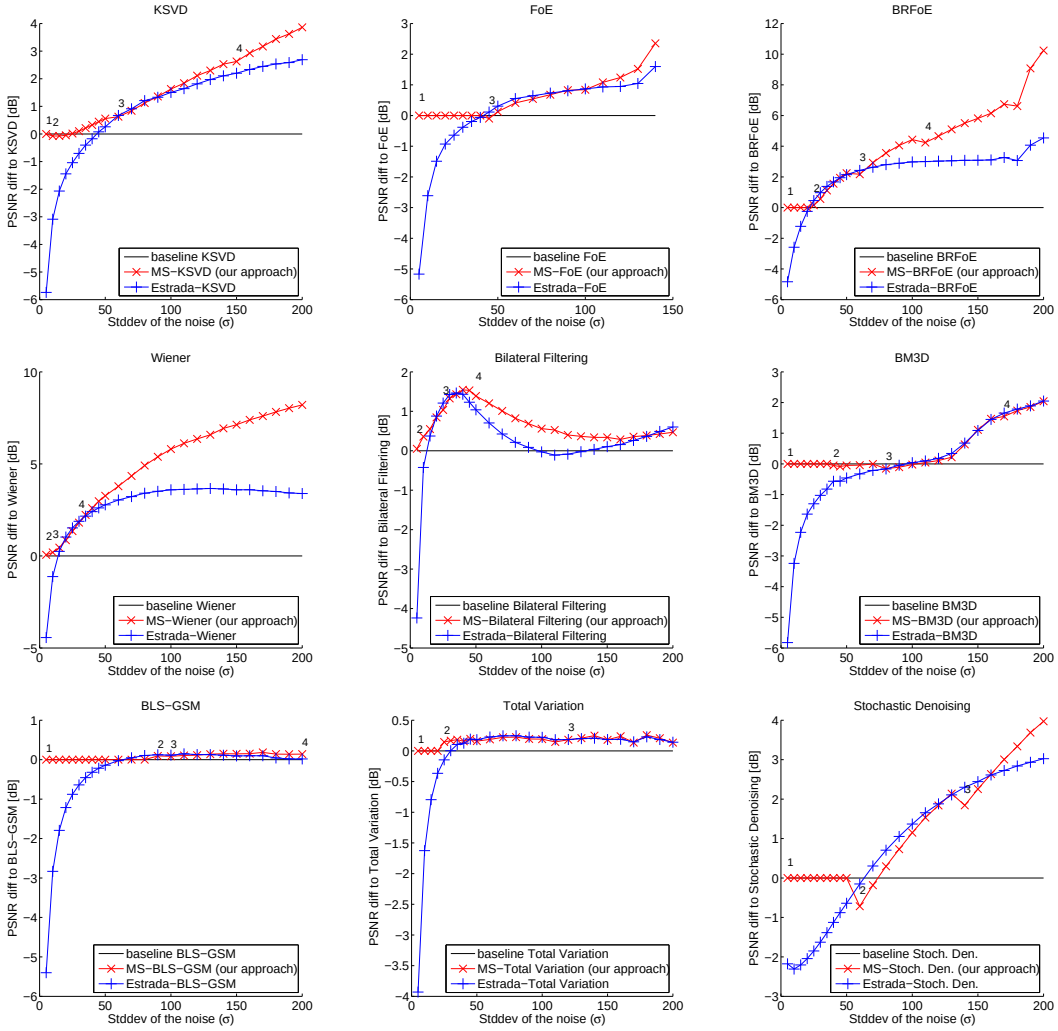


Figure 3.8: Improvements achieved by combining our meta-procedure with nine different denoising algorithms. Results are averaged over 13 test images.

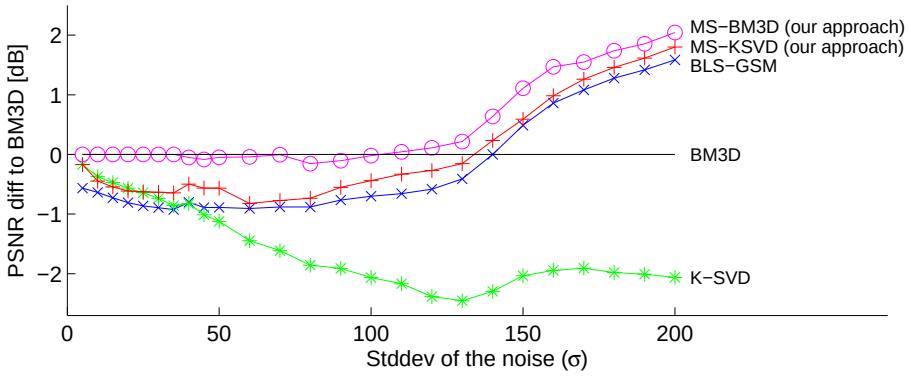


Figure 3.9: The overall best methods compared to the baseline BM3D. For high noise setting our multi-scale approach applied to BM3D leads to the best results.

to BLS-GSM, described in [94]. It was noted that on the images "Peppers", "House" and "Barbara", KSVD outperforms BLS-GSM as long as the noise is below $\sigma = 50$. When the noise level is increased, BLS-GSM outperforms KSVD. We repeat the experiment on our images, but this time also report the results achieved with the multi-scale extension applied to KSVD (Fig. 3.9). We indeed observe that baseline KSVD outperforms BLS-GSM when the noise is low. However, the multi-scale version of KSVD outperforms BLS-GSM on all noise settings, see Fig. 3.9.

Multi-scale KSVD vs. BM3D BM3D is often considered to be the best denoising algorithm currently available, even though Fig. 3.9 shows that for high noise levels BLS-GSM is superior. Also the multi-scale extensions of KSVD is better when the noise is very high.

Multi-scale BM3D vs. all others Our multi-scale extension combined with BM3D delivers results that outperform all other denoising algorithms especially on the high noise levels, see Fig. 3.9.

3.6 Conclusion

For high noise levels, not only the high frequencies but also the low frequencies are corrupted. However, most image denoising algorithms are not always good at recovering low-frequency information. To improve such algorithms we devised a strategy to improve the denoising results using a multi-scale approach.

In comprehensive experiments we have shown that several state-of-the-art image denoising algorithms can be improved using this approach. Even though BM3D is arguably one of the best currently existing denoising algorithms, our method was able to improve its results on images that have been corrupted by high noise levels.

Figure 3.10: Results obtained with our method on image "Lena", corrupted with AWG noise with $\sigma = 130$.

Figure 3.11: Results obtained with our method on image "Barbara", corrupted with AWG noise with $\sigma = 130$.

Figure 3.12: Results obtained with our method on image "Peppers", corrupted with AWG noise with $\sigma = 130$.

4

Astronomical image denoising using a pixel-specific noise model

Chapter abstract For digital photographs of astronomical objects, where exposure times are usually long and ISO settings high, the so-called dark-current is a significant source of noise. Dark-current refers to thermally generated electrons and is therefore present even in the absence of light. In this chapter, we present a novel approach for denoising astronomical images that have been corrupted by dark-current noise. Our method relies on a probabilistic description of the dark-current of each pixel of a given camera. The noise model is then combined with an image prior which is adapted to astronomical images. In a laboratory environment, we use a black and white CCD camera containing a cooling unit and show that our method is superior to existing methods in terms of root mean squared error. Furthermore, we show that our method is practically relevant by providing visually more appealing results on astronomical photographs taken with a single lens reflex CMOS camera.

The material of this chapter is based on the following publication:

[14] H.C. Burger, B. Schölkopf, and S. Harmeling. Removing noise from astronomical images using a pixel-specific noise model. International Conference on Computational Photography (ICCP). 2011.

4.1 Introduction

The problem of removing noise from images has been extensively studied. Methods to denoise images are numerous and diverse, see Chapter 2. However, most work focuses on the denoising of natural images or images of everyday scenes. Denoising algorithms are usually evaluated on their ability to remove additive white Gaussian noise with zero mean and uniform variance across the image. Often it is assumed that the variance of the noise is known.

The state-of-the-art image denoising methods try to leverage properties that are inherent to natural images. *E.g.* denoising with Fields of Experts [100] exploits the statistics of natural images: the algorithm relies on a set of small filters that have been trained on a dataset of natural images. Another successful denoising method, Bayesian least squares - Gaussian scale mixtures (BLS-GSM) [94] applies a wavelet transform on a noisy image and then exploits correlations between neighboring coefficients that are observed on natural images. A further recent method, called BM3D [25] uses the fact that in natural images, different patches are often similar in appearance.

The problem of attenuating noise in astronomical images is quite different, however. Astronomical images have statistics that are completely different from natural images. Often, such images contain little structure, such as a few stars against a black background. In addition, the characteristics on the dark-current noise that corrupts these images differs from the uniform additive white Gaussian noise most methods have been tuned to remove. We will show that pixels of a sensor behave differently. Therefore, we emphasize understanding and exploiting the statistics of the noise, rather than the image.

Assumption: The statistics of sensor noise is not adequately described by uniform additive white Gaussian noise (AWGN), see *e.g.* [49].

Hypothesis: Exploiting the noise statistics of each individual pixel of a camera's sensor leads to better denoising results.

Contribution: We present a novel method to remove noise from astronomical images that have been corrupted by dark-current noise. Our method relies on the combination of a statistical description of the noise of each pixel of a camera's sensor and an image prior. We show that the results achieved in that way are better than those obtained with state-of-the-art denoising methods.

Related work: Besides the state-of-the-art works on image denoising mentioned above, various authors have tried to create an artificial dark frame to decrease dark current noise. Such dark-frames are created in such a way as to optimize some image quality measure once the dark-frame has been subtracted from the noisy image. *E.g.*, Goesele *et al.* [44] have proposed to create an artificial dark-frame by scaling a given dark-frame in such a way as to minimize the entropy of the image. The method assumes that dark-current increases with increasing temperature, but does not take into account the random fluctuations for the dark-current. Gomez-Rodriguez *et al.* [45] have proposed to create a convex combination of previously recorded dark-frames in such a way as to minimize the discrete gradient of the image at certain locations.

4.2 Dark-current noise

There exist many sources of noise in the imaging process. The different units of a CCD chip produce different voltages for the same amount of input light, a phenomenon sometimes called "fixed pattern noise" (FPN). This effect is due to the fact that the wells in a CCD chip slightly vary in size. An imaging sensor also produces "dark current", which is due to thermal energy [119] and therefore also present in the absence of light. The analog to digital conversion process also introduces noise.

In the following we are not considering all possible sources of noise, but focus our attention on the image sensor's dark current. The thermal energy of the imaging sensor frees electrons, which then accumulate in the chip's wells. When an image is read out, the thermal electrons are indistinguishable from photoelectrons. We therefore assume the dark current to be additive and independent of the image signal. This assumption has been successfully exploited by others to denoise astronomical images [45]. Random samples of dark-current can be recorded with a so-called "dark-frame", which is a photograph taken with closed lens and non-zero exposure time.

Since dark-current noise is due to thermal electrons, it is possible to reduce the amount of dark-current by cooling the camera's sensor. For cameras on which cooling the image sensor is not possible, a simple approach for denoising photographs is to subtract a dark-frame with matching camera settings from the image. An improvement over this method would be to subtract the average of many dark-frames [49], thereby reducing the random components of the dark current.

Dark-current depends on the exposure time of the image, as well as the ISO-setting of the camera and the temperature of the image sensor [128]. More generally speaking, the problem of removing dark-current noise can be seen as a decomposition problem: Given a noisy observation y, what is the most likely true image x and the most likely noise sample d such that $y = x + d$, see Fig. 4.1. The problem is solvable if we exploit knowledge about the statistics of the image and about the distribution of the noise.

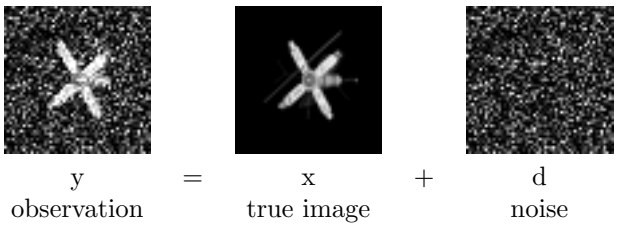


Figure 4.1: The noisy image is modelled as a sum of the true image and the dark current.

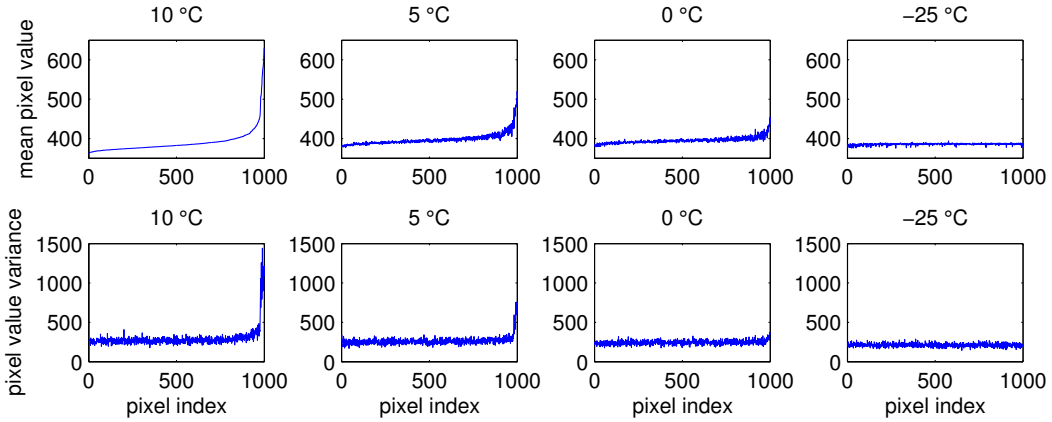


Figure 4.2: Mean (top) and variance (bottom) of 1000 randomly chosen pixels for decreasing temperatures (left to right).

To study the statistics of dark current, we use a camera with a CCD chip (pco.2000, image sensor KAI-4021, cooled) that has a fixed conversion factor of 2.2 electrons per pixel count. Each dark-frame contains 2048×2048 pixels. Each pixel value is encoded as a 16 bit value. The camera has a built-in cooling unit, allowing us to study the property of the

noise at different temperature settings. We record 200 dark-frames with 10 second exposure time at chip temperatures 10°C, 5°C, 0°C, and −25°C.

We randomly select 1000 pixels which we analyse in the following. For each temperature setting we calculate the mean and variances of these pixels. The top left panel of Fig. 4.2 shows the sorted mean values of those 1000 pixels over the 200 frames recorded at 20°C. Using this sorting of the pixels we show in the top row panel the corresponding mean values for the other temperatures and in the bottom panel (again with the same ordering as the top left panel) the variances. First of all we observe that the variances increase with the mean pixel values. Also we see that for different temperatures the same pixels exhibit a large mean and variance. Furthermore, we observe that for lower temperatures the means and variances decrease.

4.3 Theory

For notational simplicity, the recorded image y, the dark frame d, and the true image x are column vectors of the same lengths. The entries are denoted by subscripts, *i.e.* y_i, d_i, and x_i.

Maximum likelihood estimator. Theoretically, the pixel values of a dark frame d should be modelled with a Poisson distribution, *e.g.* [115]. This would imply that for the correct (ISO-dependent) scaling of the dark frames their pixel-wise mean and variance should coincide. However, such a scaling factor did not exist for our library of dark frames. For that reason we model the pixel values of the dark frames with Gaussian distributions for which we can adjust pixel-wise the mean and variance independently. For simplicity we consider in this part only dark frames of a fixed temperature. For each pixel d_i in a dark frame d, we can estimate its mean vector μ and variance vector σ^2 from some large set of dark frames that have been recorded with a fixed temperature. Modelling the recorded noisy image as $y = x + d$, the negative log likelihood of observing some image y is:

$$-\log p(y|x) = \frac{1}{2}\|y - x - \mu\|_D^2 + c \tag{4.1}$$

$$= \frac{1}{2}(y - x - \mu)^\mathsf{T} D^{-1}(y - x - \mu) + c \tag{4.2}$$

$$= \sum_i \frac{(y_i - x_i - \mu_i)^2}{2\sigma_i^2} + c \tag{4.3}$$

where D is the diagonal matrix with the variances σ^2 along its diagonal. The constant c depends only on D but not on y or x. Note that this likelihood models the amount of noise individually for each pixel location. This is different from most other methods, *e.g.* [31], who assume a global value for the variance. Methods that allow different amounts of noise in different locations of the image, usually estimate that amount from the noisy image [70]. Instead, we determine the noise distribution from a library of dark frames.

The maximum likelihood estimate of x is equal to the difference of y and μ, because setting $x = y - \mu$ minimizes $-\log p(y|x)$. We will denote this approach by DF-ML.

Maximum a posteriori estimator. The state of the art denoising methods have successfully introduced priors for natural images, *e.g.* [127]. Unfortunately, these priors do not apply to astronomical images as we will see in the experimental section. Instead we employ a generalization of a simple image prior used by Gomez-Rodriguez *et al.* [45] which is based on the idea that an astronomical image should be smooth and not grainy, *i.e.* the prior assumes that neighboring pixels should have similar values,

$$-\log p(x) = \lambda \frac{1}{|N_i|} \sum_{j \in N_i} |x_i - x_j|^p + c, \tag{4.4}$$

where c is a constant independent of x, and N_i are the indices of the eight neighbors of pixel i. In case the camera has a color filter array (CFA), the set of neighbors refers to the closest pixels on the same color channel. Gomez-Rodriguez *et al.* [45] applied this prior for $p = 2$. However, in our experiments it turned out that we get the best results for $p = 1.4$. The factor λ depends on the inverse variance of the image prior, *i.e.* it controls to which degree a pixel can be different from its neighbors. For the maximum a posteriori estimator, λ is a hyper parameter which controls the trade-off between the image prior and the likelihood. Throughout all reported experiments it was fixed to $\lambda = 100$. This value was determined on artificial training images.

Together with the likelihood we can write down the posterior distribution for x,

$$-\log p(x|y) = -\log p(y|x) - \log p(x) + c, \tag{4.5}$$

where c is again a constant independent of x and y. To determine the maximum a posteriori estimator, we have to minimize the negative log-likelihood $-\log p(x|y)$ in x. For this, we initialize x with its maximum likelihood estimate and proceed with gradient descent steps minimizing the negative log posterior $-\log p(x|y)$ in x. We employ an early stopping strategy to avoid over-smoothing. In the following, we denote this method by DF-MAP$_p$.

4.4 Experiments

4.4.1 Artificial stars with ground truth

To evaluate our method, we create artificial stars in a dark sky by employing a black surface containing small holes through which dim light shines. We obtain a low-noise ground truth image by the following procedure: we take 200 photos with a camera (pco.2000, image sensor KAI-4021, cooled down to $-25°$C) and average the resulting images to reduce the noise. From the resulting image, we subtract the average of 200 dark-frames recorded with the same chip temperature and with the same exposure times. The exposure time of all images is 10 seconds. Besides the ground truth image, we take also 100 noisy test images at 10°C chip temperature. The images are shot in very low light conditions, resulting in noticeable noise, see Fig. 4.3. Our goal is to recover the clean image as well as possible, given a single noisy image.

To compare a reconstructed image with the ground truth image, we calculate the root mean squared error (RMSE) between the ground truth image x^* and the reconstructed image x, defined as $\mathrm{RMSE}(x^*, x) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^* - x_i)^2}$, where n is the number of pixels in the image.

As competitors to our method we apply state-of-the-art methods for image denoising: BM3D [25], BLS-GSM [94], Bilateral filtering [117], Fields of Experts (FoE) [100] and dictionary-based denoising with an overcomplete DCT dictionary (DCT) [31]. All those methods are initialized with the maximum likelihood solution, *i.e.* with the difference between the noisy frame and the mean dark-frame. We then apply the various denoising algorithms. Usually, these denoising algorithms require a parameter describing the standard deviation of the noise present in the noisy image and it is assumed that this standard deviation is the same for each pixel in the image. We set this parameter value to be the average of all σ_is, which yields good results in practice.

A method that is tailored for astronomical images is the method proposed by Gomez-Rodriguez *et al.* [45], which creates an artificial dark-frame as a convex combination of available dark-frames. The method—called QP in the following—attempts to minimize the same image penalty function as we do (with $p = 2$) on a small selection of pixels (so-called "evaluation points"), which leads to a tractable quadratic programming problem. Following [45], we apply their method using 1000 evaluation points. We use two different settings regarding the library of dark-frames: once we use 160 dark-frames, all recorded with

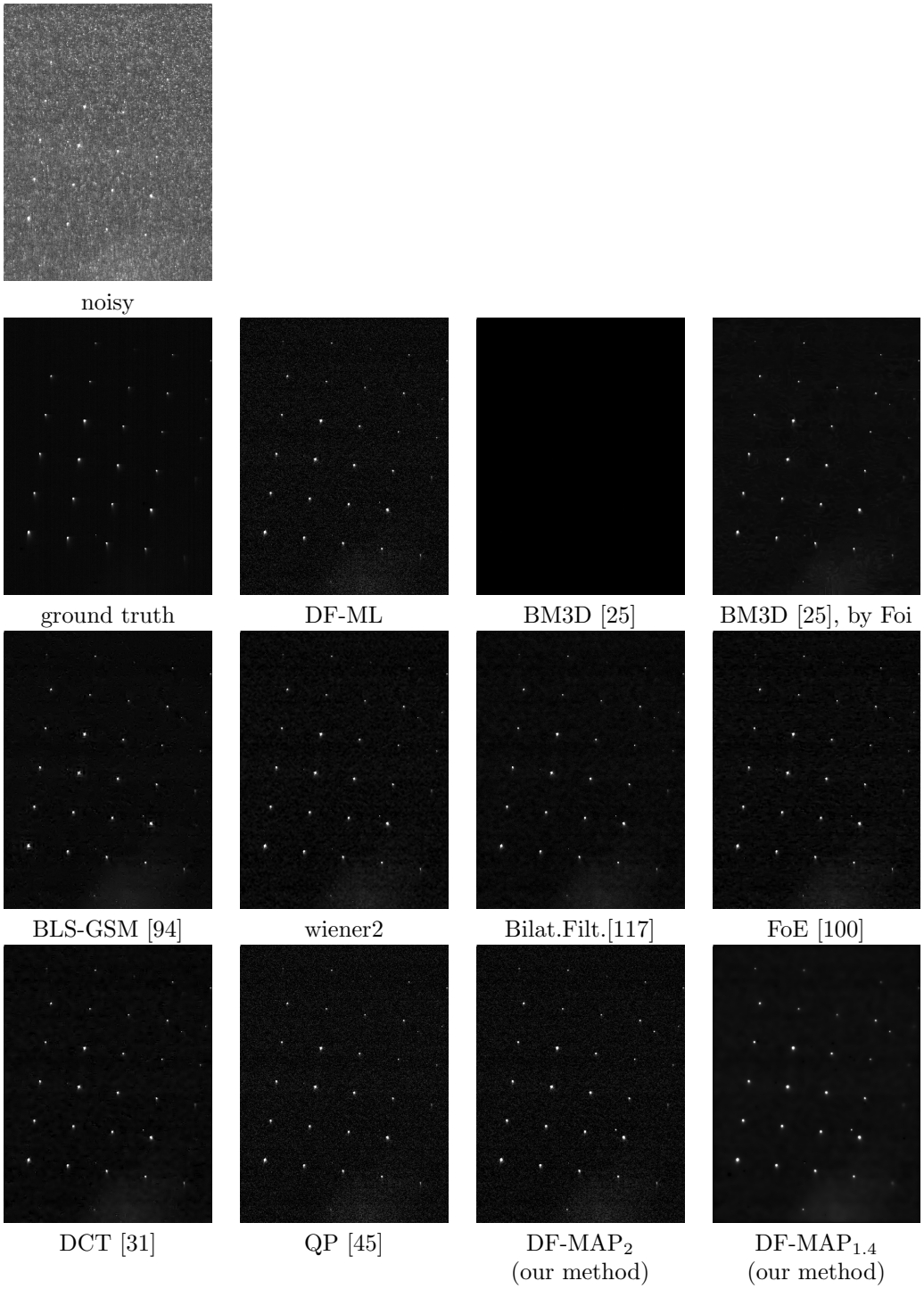


Figure 4.3: The results obtained by denoising using various methods.

10 second exposure time but with temperatures ranging from −25°C to 10°C (20 frames per temperature, using 5°C increments). In a second setting, we use 200 dark-frames recorded with 10 second exposure time and the temperature matching the temperature with which the image was taken.

The method that only subtracts the mean dark-frame, corresponding to the maximum likelihood estimate, is denoted by DF-ML. The results of our proposed maximum a posteriori methods are denoted by DF-MAP$_p$.

denoising method	mean RMSE
no denoising	416.3
BM3D [25]	103.7
BLS-GSM [94]	36.7
QP (all temperatures) [45]	35.2
DF-ML (mean dark frame)	32.1
QP (only one temperature) [45]	29.6
DF-MAP$_2$ (our method)	27.8
Matlab's wiener2	27.5
Bilateral Filtering [117]	27.4
Fields of Experts [100]	27.2
DCT [31]	26.9
BM3D [25], by Foi	21.6
DF-MAP$_{1.4}$ (our method)	**20.0**

Table 4.1: Average results obtained by denoising 100 noisy test images with different methods

The results in Tab. 4.1 show that the state-of-the-art denoising methods are not suitable for astronomical images. *E.g.*, BM3D and BLS-GSM yield results that are worse than those obtained by removing the mean dark-frame from the noisy image. Since the mean dark-frame subtracted image is provided as input, one can say that these methods deteriorate the results. For BM3D, we were informed by one of the authors of the BM3D paper (Alessandro Foi) that the poor results might be due to a software bug and were provided with a result achieved with a corrected toolbox ("BM3D, by Foi" in Tab. 4.1). We thank Alessandro Foi for helping us with the BM3D toolbox. Fig. 4.3 compares the results obtained by denoising strategies we have tried. The result of BLS-GSM contains artifacts that resemble the "ringing" phenomenon. We presume that BLS-GSM makes assumptions about images which are violated by the images we are trying to denoise. Further evidence is that simpler denoising methods, such as Matlab's `wiener2` function and bilateral filtering [117] work relatively well: these methods make fewer assumptions about the statistics of images. Another method that works well is the dictionary-based denoising method with an overcomplete DCT Dictionary [31]. The method assumes that patches in an image can be well represented using a sparse combination of predefined patches, which appears to work well for astronomical images.

If we set the parameter p of our image prior to 2, we use the same image penalty function as the method proposed by Gomez-Rodriguez *et al.* [45]. Yet, our results are better. This is presumably due to the fact that our model is more flexible to repair individual pixels, whereas Gomez-Rodriguez *et al.* are forced to subtract a convex combination of dark frames.

Modifying the image prior by setting $p = 1.4$, the results obtained with our method are even better (last line in Tab. 4.1). It is not clear if the method proposed by Gomez-Rodriguez *et al.* could be modified to use a different image prior.

4.4.2 Real astronomical images

Orion constellation. To test our approach under real-world conditions we applied it to a noisy image of part of the constellation Orion. The image was taken by a Canon EOS 5D

with 60 seconds exposure time and ISO 1600 and Canon lens 35/1.4 at aperture f/2.8. To minimize motion blur due to celestial movements a tracking mount was used. Demosaicing and gamma correction was performed using dcraw [23].

In addition to the noisy astronomical image, we had a library of dark-frames recorded with the same camera. We had no control over the temperature of the image sensor. The library of dark-frames was composed of 16 darkframes at exposure time 10 seconds, 32 at exposure time 60 seconds, 32 at exposure time 120 seconds as well as 16 bias-frames (dark-frames with the shortest possible exposure time).

Fig. 4.4 shows the results of different denoising approaches for an enlarged cropped version (800×1000 pixels). The presented images were reconstructed by first denoising the raw images, then demosaicing with dcraw [23] and finally gamma correction. Because BLS-GSM is not meant to be applied to raw images, we apply it separately to the four color channels. The approach proposed by Gomez-Rodriguez *et al.* [45] and our method are also able to treat raw images: the image prior is not calculated by considering a pixel's immediate neighbors, but rather the closest neighbors on the same color channel.

Subtracting the mean dark-frame, *i.e.* the maximum likelihood estimate, does not significantly improve the visual quality of the image. In fact, artifacts are introduced: some pixels seem to be too dark. A possible explanation for this phenomenon would be if the camera's sensor was warmer at the time the image was recorded than at the time the dark-frames were recorded. Dark-current increases with increasing temperature, so the dark-current in the image would be weaker than in the dark-frames.

Applying BLS-GSM to the dark-frame subtracted image provides little improvement over the dark-frame subtracted image. We found the results obtained by BLS-GSM to be representative of results obtained with the other state-of-the-art image denoising methods.

Using our method with $p = 2$ also did not create satisfactory results. Visually, it is not clear whether using our method with $p = 2$ provides better results than the mean dark-frame subtracted result or using BLS-GSM.

We apply the approach proposed by Gomez-Rodriguez *et al.* [45] using all dark-frames contained in our library. The result obtained in this way is visually better than both the original noisy image and the mean dark-frame subtracted image. The dark pixels that are present in the mean dark-frame subtracted image do not exist in the image denoised by the method proposed by Gomez-Rodriguez *et al.* It is possible that the method was able to select dark-frames that were recorded at a matching sensor temperature. However, noise is still present in the image. The background looks grainy.

For our method, we estimate the pixel means μ and variances σ^2 using the dark-frames that have been recorded with an exposure time of 60 seconds. Ideally, we would have estimated μ and σ^2 on dark-frames whose recording temperature matches that of the noisy image. Nonetheless, the result obtained by our method is smoother than any of the previously applied methods. Our method was able to strongly reduce the graininess that was visible in all previous results. Our method provides a very smooth background, yet does not cause even faint stars to disappear. Also, the visual quality of the nebula is not deteriorated.

Milky way. Finally, we present results obtained on an image of the Milky Way, recorded at an ISO setting of 3200, with a Canon EOS 5D, see top panel of Fig. 4.5. For this image we had only six dark-frames of matching settings available to us. We use these six dark-frames for the method proposed by Gomez-Rodriguez *et al.* as well as for our method. On the left and right of the images we provide more detailed views of parts of the image. The red inset contains four hot pixels in the noisy image (the bright green pixels), which were successfully removed by both the method proposed by Gomez-Rodriguez *et al.* and ours. However, the result obtained by our method appears much less grainy, which makes individual stars more discernible.

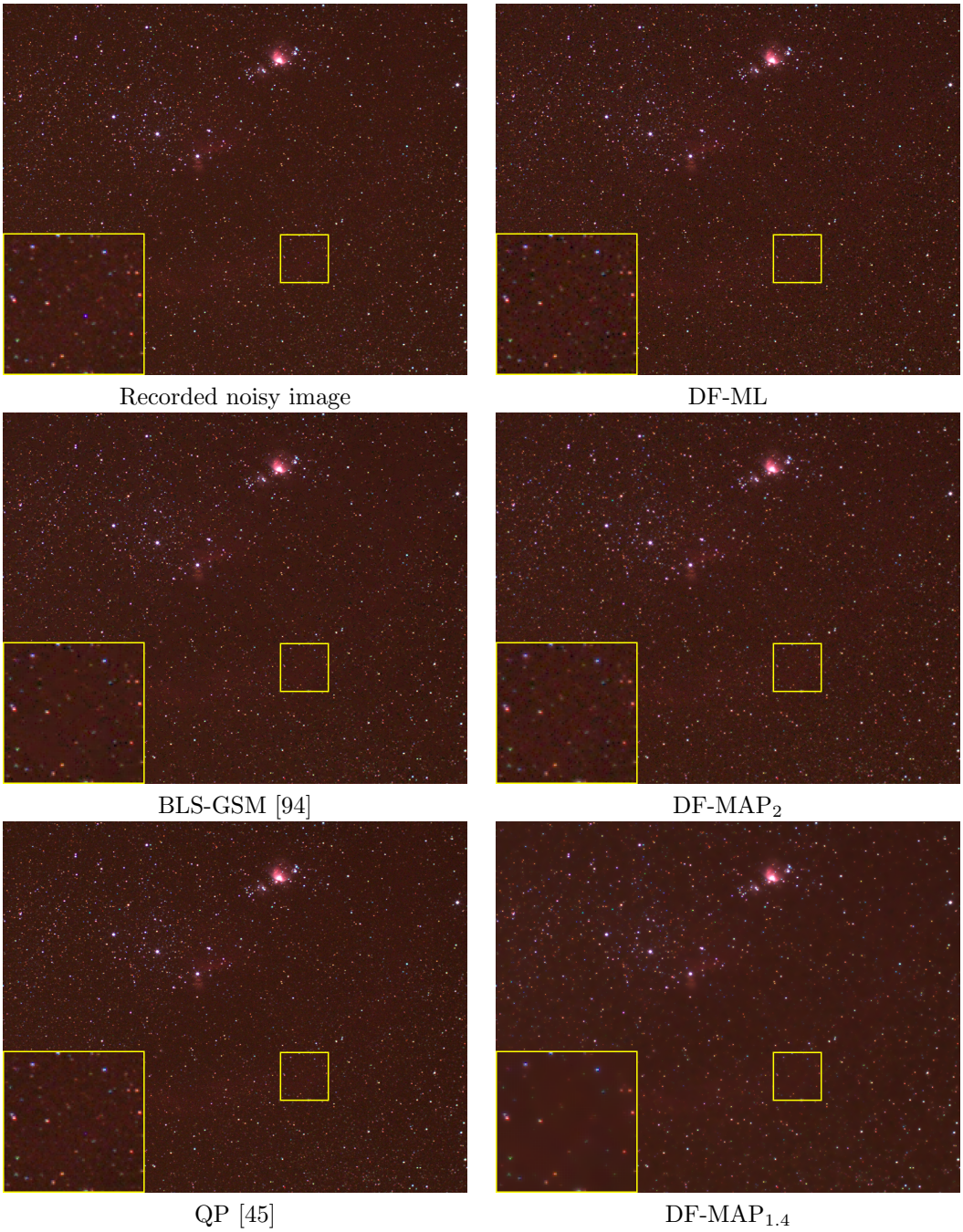
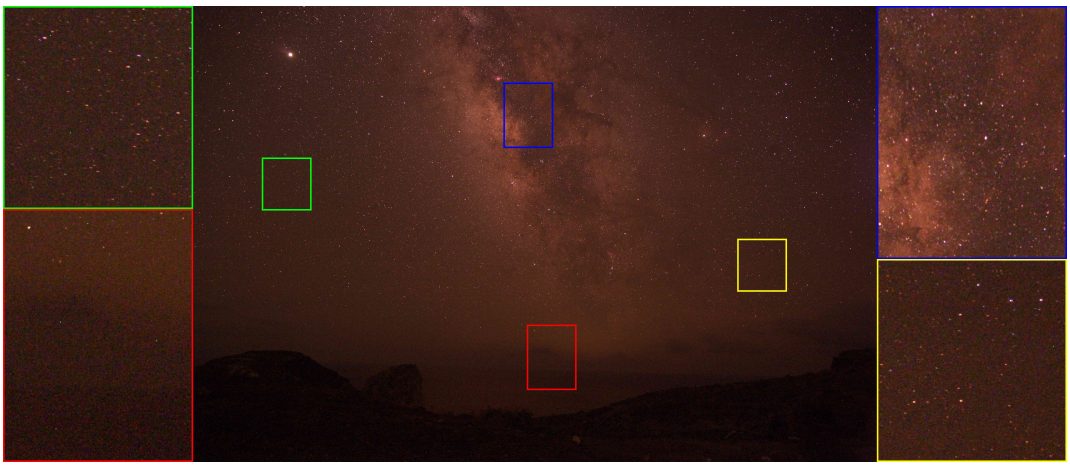
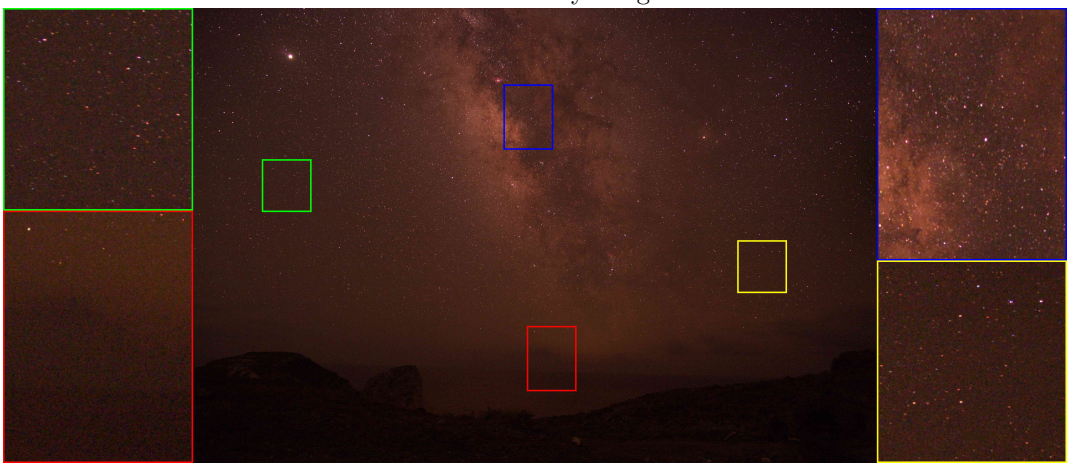


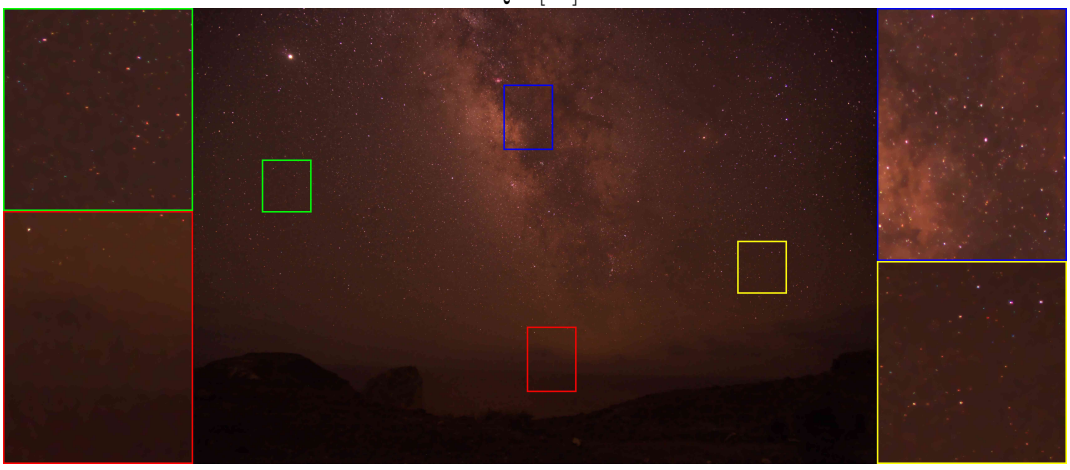
Figure 4.4: Comparison of various denoising techniques on a small section of the image of the constellation Orion.



Recorded noisy image



QP [45]



DF-MAP$_{1.4}$ (our method)

Figure 4.5: Comparison of QP [45] and DF-MAP$_{1.4}$ (our method) on a real astronomical image (taken with a Canon EOS 5D at ISO 3200 and 60 sec. exposure time with Zeiss Distagon 28mm lens at f/2.8, courtesy of Gomez-Rodriguez, Kober, and Schölkopf [45]).

4.5 Conclusions

We presented a new method for denoising astronomical images containing dark current noise. The method relies on a probabilistic description of a given camera's dark-current, as well as an image prior appropriate for astronomical images. Our method treats every pixel of a camera's sensor individually. Our image prior is similar to the one used by Gomez-Rodriguez *et al.* [45] and attempts to capture the roughness or graininess of an image. However, different from [45] we are not limited to quadratic functions, which allows us to use an image prior that is better suited for astronomical images, and moreover, we are not restricted to subtracting a convex combination of dark frames.

In laboratory conditions, we have shown that our method provides better results than state-of-the-art denoising methods that are intended for use on natural images. Our method also outperformed the recent method by Gomez-Rodriguez *et al.*, which is designed to denoise astronomical images.

On real astronomical images, we have shown that our method provides visually more appealing results than other methods. Images appear much less grainy after applying our method than when applying other methods. Fine image structure such as faint stars and nebula are preserved. It should be added that our evaluation was on single images, which is the hardest case in the sense that their noise is higher than for averages over several images as often used in astrophotography. Moreover, some of the graininess that we remove can also be removed by using a more sophisticated image acquisition pipeline including dithering (combining multiple exposures offset with respect to each other). We would expect that this would further improve our results, but make the difference to the other methods smaller.

Our method is limited in that we assume that appropriate dark-frames are provided with the image to be denoised. We assume the exposure time, ISO setting and temperature of the camera's sensor to approximately match the conditions at which the noisy image was recorded. The method proposed by Gomez-Rodriguez *et al.* overcomes this difficulty: given a library of dark-frames recorded under varying conditions, the optimization problem selects dark-frames that were recorded under the same conditions as the image. It is therefore conceivable to combine the two methods: the quadratic optimization problem described by Gomez-Rodriguez *et al.* could be used to select a set of dark-frames from a larger library. Our method would then use the selected dark-frames to infer an appropriate probabilistic dark-current model for the denoising process.

5

Image denoising with multi-layer perceptrons

Chapter abstract Image denoising can be described as the problem of mapping from a noisy image to a noise-free image. The best currently available denoising methods approximate this mapping with cleverly engineered algorithms. In this work we attempt to learn this mapping directly with plain multi layer perceptrons (MLP) applied to image patches. We will show that by training on large image databases we are able to outperform the current state-of-the-art image denoising methods. In addition, our method achieves results that are superior to one type of theoretical bound and goes a large way toward closing the gap with a second type of theoretical bound. Our approach is easily adapted to less extensively studied types of noise, such as mixed Poisson-Gaussian noise, JPEG artifacts, salt-and-pepper noise and noise resembling stripes, for which we achieve excellent results as well. We will show that combining a block-matching procedure with MLPs can further improve the results on certain images. Finally, we show that MLPs can be used to combine the results of several denoising algorithms, usually delivering results that are superior to the best result in the combination. This approach gets still closer to closing the gap with theoretical bounds.

The material of this chapter is based on the following publications:

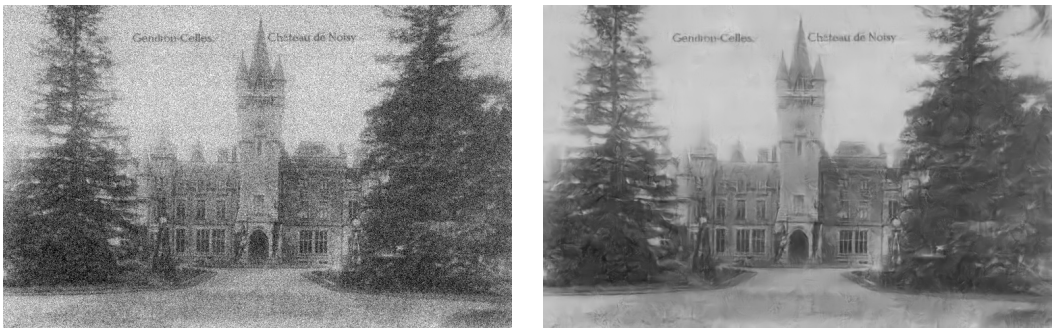
[15] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? Conference on Computer Vision and Pattern Recognition (CVPR). 2012.

[12] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 1: comparison with existing algorithms and with bounds. **Submitted** to a journal, **available** at `http://arxiv.org/abs/1211.1544`

The following publication is partially included in this chapter:

[16] H.C. Burger, C.J. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. **Submitted** to a conference.

5.1 Introduction



Noisy castle Clean castle

Figure 5.1: The goal of image denoising is to find a clean version of the noisy input image.

Images are invariably corrupted by some degree of noise. The strength and type of noise corrupting the image depends on the imaging process. In scientific imaging, one sometimes needs to take images in a low photon-count setting, in which case the images are corrupted by mixed Poisson-Gaussian noise [71]. Magnetic resonance images are usually corrupted by noise distributed according to the Rice disitribution [46]. For natural images captured by a digital camera, the noise is usually assumed to be additive, white and Gaussian-distributed (AWG noise), see for example [31, 25].

An image denoising procedure takes a noisy image as input and estimates an image where the noise has been reduced. Numerous and diverse approaches exist: Some selectively smooth parts of a noisy image [117, 126]. Other methods rely on the careful shrinkage of wavelet coefficients [111, 94]. A conceptually similar approach is to denoise image patches by trying to approximate noisy patches using a sparse linear combination of elements of a learned dictionary [1, 31]. BM3D [25] is a very successful approach to denoising and is often considered state-of-the art. The approach does not rely on a probabilistic image prior but rather exploits the fact that images are often self-similar: A given patch in an image is likely to be found elsewhere in the same image. In BM3D, several similar-looking patches of a noisy image are denoised simultaneously and *collaboratively*: Each noisy patch helps to denoise the other noisy patches. The algorithm does not rely on learning from a large dataset of natural images; excellent denoising results are achieved through the design of the algorithm. While BM3D is a well-engineered algorithm, could we also *automatically learn* an image denoising procedure purely from training examples consisting of pairs of noisy and noise-free patches?

Denoising as a function: In image denoising, one is given a noisy version of a clean image, where the noise is for instance i.i.d. Gaussian distributed with known variance (AWG noise). The goal is to find the clean image, given only the noisy image. We think of denoising as a function that maps a noisy image to a cleaner version of that image. However, the complexity of a mapping from images to images is large, so in practice we chop the image into possibly overlapping *patches* and learn a mapping from a noisy patch to a clean patch. To denoise a given image, all image patches are denoised separately by that map. The denoised image patches are then combined into a denoised image.

The size of the patches affects the quality of the denoising function. If the patches are small and the noise level is high, many clean patches are a potential explanation for a given noisy patch. In other words, adding noise to a clean patch is not injective and therefore also not invertible. It is therefore almost impossible to find a perfect denoising function. Lowering the noise and increasing the size of the patches alleviates this problem: Fewer

clean patches are a potential explanation for a given noisy image [68]. At least in theory, better denoising results are therefore achievable with large patches than with small patches.

In practice, the mapping from noisy to clean patches cannot be expressed using a simple formula. However, one can easily generate *samples*: Adding noise to a patch creates an argument-value pair, where the noisy patch is the argument of the function and the noise-free patch is the value of the function.

The aim of this chapter is to *learn* the denoising function. For this, we require a *model*. The choice of the model is influenced by the function to approximate. Complicated functions require models with high *capacity*, whereas simple functions can be approximated using a model with low capacity. The dimensionality of the problem, which is defined by the size of the patches, is one measure of the difficulty of approximation. One should therefore expect that models with more capacity are required when large image patches are used. A higher dimensionality also usually implies that more *training data* is required to learn the model, unless the problem is intrinsically of low dimension.

We see that a trade-off is necessary: Very small patches lead to a function that is easily modeled, but to bad denoising results. Very large patches potentially lead to better denoising results, but the function might be difficult to model.

This chapter will show that it is indeed possible to achieve state-of-the-art denoising performance with a plain multi layer perceptron (MLP) that maps noisy patches onto noise-free ones. This is possible because the following factors are combined:

- The capacity of the MLP is chosen large enough, meaning that it consists of enough hidden layers with sufficiently many hidden units.
- The patch size is chosen large enough, so that a patch contains enough information to recover a noise-free version. This is in agreement with previous findings [68].
- The chosen training set is large enough. Training examples are generated on the fly by corrupting noise-free patches with noise.

Training high capacity MLPs with large training sets is feasible using modern Graphics Processing Units (GPUs). Chapter 6 contains a detailed analysis of the trade-offs during training.

Contributions: We present a patch-based denoising algorithm that is *learned* on a large dataset with a plain neural network. Additional contributions of this chapter are the following.

1. We show that the state-of-the-art is improved on AWG noise. This is done using a thorough evaluation on 2500 test images,
2. excellent results are obtained on mixed Poisson-Gaussian noise, JPEG artifacts, salt-and-pepper noise and noise resembling stripes, and
3. We present a novel "block-matching" multi-layer perceptron and discuss its strengths and weaknesses.
4. We relate our results to recent theoretical work on the limits of denoising [22, 68, 69]. We will show that two of the bounds described in these papers cannot be regarded as hard limits. We make important steps towards reaching the third proposed bound.

We have previously shown that MLPs can achieve outstanding image denoising results [15]. Here, we present significantly improved results compared to our previous work as well as more thorough experiments.

5.2 Related work

The problem of removing noise from natural images has been extensively studied, so methods to denoise natural images are numerous and diverse. Estrada *et al.* [34] classify denoising algorithms into three categories:

1. The first class of algorithms rely on smoothing parts of the noisy image [101, 126, 117] with the aim of "smoothing out" the noise while preserving image details.

2. The second class of algorithms exploits the fact that different patches in the same image are often similar in appearance [25, 10].

3. The third class of denoising algorithms exploit learned image statistics. A natural image model is typically learned on a noise-free training set (such as the Berkeley segmentation dataset) and then exploited to denoise images [100, 127, 57]. In some cases, denoising might involve the careful shrinkage of coefficients. For example [111, 21, 91, 94] involve shrinkage of wavelet coefficients. Other methods denoise small images patches by representing them as sparse linear combinations of elements of a learned dictionary [31, 75, 74].

Neural networks: Neural networks belong to the category relying on learned image statistics. They have already been used to denoise images [57] and belong in the category of learning-based approaches. The networks commonly used are of a special type, known as *convolutional neural networks* (CNNs) [63], which have been shown to be effective for various tasks such as hand-written digit and traffic sign recognition [108]. CNNs exhibit a structure (local receptive fields) specifically designed for image data. This allows for a reduction of the number of parameters compared to plain multi layer perceptrons while still providing good results. This is useful when the amount of training data is small. On the other hand, multi layer perceptrons are potentially more powerful than CNNs: MLPs can be thought of as universal function approximators [24, 56, 38, 67], whereas CNNs restrict the class of possible learned functions.

A different kind of neural network with a special architecture (containing a *sparsifying logistic*) is used in [96] to denoise image patches. A small training set is used. Results are reported for strong levels of noise. It has also been attempted to denoise images by applying multi layer perceptrons on wavelet coefficients [131]. The use of wavelet bases can be seen as an attempt to incorporate prior knowledge about images.

Denoising auto-encoders [120] also use the idea of using neural networks for denoising. Denoising auto-encoders are a special type of neural network which can be trained in an unsupervised fashion. Interesting features are learned by the units in the hidden layers. For this, one exploits the fact that training pairs can be generated cheaply, by somehow corrupting (such as by adding noise to) the input. However, the goal of these networks is not to achieve state-of-the-art results in terms of denoising performance, but rather to learn representations of data that are useful for other tasks. Another difference is that typically, the noise used is not AWG noise, but salt-and-pepper noise or similar forms of noise which "occlude" part of the input. Denoising auto-encoders are learned layer-wise and then *stacked*, which has become the standard approach to deep learning [53]. The noise is applied on the output of the previously learned layer. This is different from our approach, in which the noise is always applied on the input patch only and all layers are learned *simultaneously*.

Our approach is reminiscent of deep learning approaches because we also employ several hidden layers. However, the goal of deep learning is to learn several levels of representations, corresponding to a hiearchy of features, see [6] for an overview. In this work we are mainly interested in image denoising results.

Innovations in this work: Most methods based on neural networks make assumptions about natural images. Instead, we show that state-of-the-art results can be obtained by imposing no such assumptions, but by relying on a pure *learning* approach.

5.3 Learning to denoise

In Section 5.1, we defined the denoising problem as learning the mapping from a noisy patch to a cleaner patch. For this, we require a model. In principle, different models could be used, but we will use MLPs for that purpose. We chose MLPs over other models because of their ability to handle large datasets.

5.3.1 Multi layer perceptrons (MLPs)

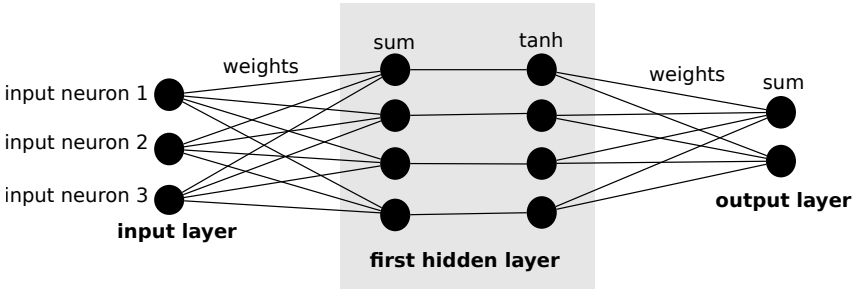


Figure 5.2: A graphical representation of a (3,4,2)-MLP.

A *multi layer perceptron* (MLP) is a nonlinear function that maps vector-valued input via several hidden layers to vector-valued output. For instance, an MLP with two hidden layers can be written as

$$f(x) = b_3 + W_3 \tanh(b_2 + W_2 \tanh(b_1 + W_1 x)). \tag{5.1}$$

The weight matrices W_1, W_2, W_3 and vector-valued biases b_1, b_2, b_3 parameterize the MLP, the function tanh operates component-wise. The *architecture* of an MLP is defined by the number of hidden layers and by the layer sizes. For instance, a (256,2000,1000,10)-MLP has two hidden layers. The input layer is 256-dimensional, i.e. $x \in \Re^{256}$. The vector $v_1 = \tanh(b_1 + W_1 x)$ of the first hidden layer is 2000-dimensional, the vector $v_2 = \tanh(b_2 + W_2 v_1)$ of the second hidden layer is 1000-dimensional, and the vector $f(x)$ of the output layer is 10-dimensional. Commonly, an MLP is also called *feed-forward neural network*. MLPs can also be represented graphically, see Figure 5.2. All our MLPs are *fully connected*, meaning that the weight matrices W_i are dense. One could also imagine MLPs which are not fully connected, using sparse weight matrices. Sparsely connected MLPs have the advantage of being potentially computationally easier to train and evaluate.

MLPs belong to the class of *parametric* models, the parameters being estimated during learning. However, the number of parameters in MLPs is often so large that they are extremely flexible.

5.3.2 Training MLPs for image denoising

To train an MLP that maps noisy image patches onto clean image patches where the noise is reduced or even removed, we estimate the parameters by training on pairs of noisy and clean image patches using *stochastic gradient descent* [64].

More precisely, we randomly pick a clean patch x from an image dataset and generate a corresponding noisy patch y by corrupting x with noise, for instance with additive white

Gaussian (AWG) noise. We then feed the noisy patch x into the MLP to compute $f(x)$, representing an estimate of the clean patch x. The MLP parameters are then updated by the *backpropagation* algorithm [102] minimizing the squared error between the mapped noisy patch $f(x)$ and the clean patch y, i.e. minimizing pixel-wise $(f(x) - y)^2$. We choose to minimize the mean squared error since it is monotonically related to the PSNR, which is the most commonly used measure of image quality. Thus minimizing the squared error will maximize PSNR values.

To make backpropagation efficient, we apply various common neural network tricks [64]:

1. Data normalization: The pixel values are transformed to have approximately mean zero and variance close to one. More precisely, assuming pixel values between 0 and 1, we subtract 0.5 and multiply by 0.2.

2. Weight initialization: We use the "normalized initialization" described by Glorot *et al.* [7]. The weights are sampled from a uniform distribution:

$$w \sim U\left[-\frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right], \tag{5.2}$$

   where n_j and n_{j+1} are the number of neurons in the input side and output side of the layer, respectively. Combined with the first trick, this ensures that both the linear and the non-linear parts of the sigmoid function are reached.

3. Learning rate division: In each layer, we divide the learning rate by N, the number of input units of that layer. This allows us to change the number of hidden units without modifying the learning rate.

The basic learning rate was set to 0.1 for most experiments. The training procedure is discussed in more detail in Chapter 6.

5.3.3 Number of hidden layers

The number hidden layers as well as the number of neurons per hidden layer control the capacity of the model. No more than a single hidden layer is needed to approximate any function, provided that layer contains a sufficient number of neurons [24, 56, 38, 67]. However, functions exist that can be represented compactly with a neural network with k hidden layers but that would require exponential size (with respect to input size) networks of depth $k - 1$ [48, 61]. Therefore, in practice it is often more convenient to use a larger number of hidden layers with fewer hidden units each. The trade-off between a larger number of hidden layers and a larger number of hidden units is discussed in Chapter 6.

5.3.4 Applying MLPs for image denoising

To denoise images, we decompose a given noisy image into overlapping patches. We then normalize the patches by subtracting 0.5 and dividing by 0.2, denoise each patch separately and perform the inverse normalization (multiply with 0.2, add 0.5) on the denoised patches. The denoised image is obtained by placing the denoised patches at the locations of their noisy counterparts, then averaging on the overlapping regions. We found that we could improve results slightly by weighting the denoised patches with a Gaussian window. Also, instead of using all possible overlapping patches (stride size 1, or patch offset 1), we found that results were almost equally good by using every third sliding-window patch (stride size 3), while decreasing computation time by a factor of 9. Using a stride size of 3, we were able to denoise images of size 350×500 pixels in approximately one minute (on CPU), which is slower than BM3D [25], but much faster than KSVD [1] and NLSC [74] and also faster than EPLL [132].

5.3.5 Efficient implementation on GPU

The computationally most intensive operations in an MLP are the matrix-vector multiplications. For these operations *Graphics Processing Units* (GPUs) are better suited than *Central Processing Units* (CPUs) because of their ability to efficiently parallelize operations. For this reason we implemented our MLP on a GPU. We used nVidia's C2050 GPU and achieved a speed-up factor of more than one order of magnitude compared to an implementation on a quad-core CPU. This speed-up is a crucial factor, allowing us to run larger-scale experiments. We describe training for various setups in Chapter 6.

5.4 Experimental setup

We performed all our experiments on gray-scale images. These were obtained from color images with MATLAB's `rbg2gray` function. Since it is unlikely that two noise samples are identical, the amount of training data is effectively infinite, no matter which dataset is used. However, the number of uncorrupted patches is restricted by the size of the dataset. Note that the MLPs could be also trained on color images, possibly exploiting structure between the different color channels. However, in this publication we focus on the gray-scale case.

Training data: For almost all our experiments, we used images from the imagenet dataset [29]. Imagenet is a hiearchically organized image database, in which each node of the hierarchy is depicted by hundreds and thousands of images. We completely disregard all labels provided with the dataset. We used 1846296 images from 2500 different object categories. We performed no pre-processing other than the transform to grey-scale on the training images.

Test data: We define six different test sets to evaluate our approach:

1. *standard test images:* This set of 11 images contains standard images, such as "Lena" and "Barbara", that have been used to evaluate other denoising algorithms [25].
2. *Berkeley BSDS500:* We used all 500 images of this dataset as a test set. Subsets of this dataset have been used as a training set for other methods such as FoE [100] and EPLL [132].
3. *Pascal VOC 2007:* We randomly selected 500 images from the Pascal VOC 2007 test set [35].
4. *Pascal VOC 2011:* We randomly selected 500 images from the Pascal VOC 2011 training set.
5. *McGill:* We randomly selected 500 images from the McGill dataset [88].
6. *ImageNet:* We randomly selected 500 images from the ImageNet dataset not present in the training set. We also used object categories not used in the training set.

We selected dataset 1) because it has become a standard test dataset, see [25] and [74]. The images contained in it are well-known and diverse: Image "Barbara" contains a lot of regular structure, whereas image "Man" contains more irregular structure and image "Lena" contains smooth areas. We chose to make a more thorough comparison, which is why we evaluated our approach as well as competing algorithms on five larger test sets. We chose five different image sets of 500 images instead of one set of 2500 images in order to see if the performance of methods is significantly affected by the choice of the dataset. EPLL [132] is trained on a subset of dataset 2), NLSC [74] is trained on a subset of 4) and our method is trained on images extracted from the same larger dataset as 6).

Types of noise: For most of our experiments, we used AWG noise with $\sigma = 25$. However, we also show results for other noise levels. Finally, we trained MLPs to remove mixed Gaussian-Poisson noise, JPEG artifacts, salt and pepper noise and noise that resembles stripes.

5.5 Results: comparison with existing algorithms

In this section, we present results achieved with an MLP on AWG noise with five different noise levels. We also present results achieved on less well-studied forms of noise. We present in more detail what steps we took to achieve these results in Chapter 6.

We compare against the following algorithms:

1. KSVD [1]: This is a dictionary-based method where the dictionary is adapted to the noisy image at hand. A noisy patch is denoised by approximating it with a sparse linear combination of dictionary elements.

2. EPLL [132]: The distribution of image patches is described by a mixture of Gaussians. The method presents a novel approach to denoising whole images based on patch-based priors. The method was shown to be sometimes superior to BM3D [25], which is often considered the state-of-the-art in image denoising.

3. BM3D [25]: The method does not explicitly use an image prior, but rather exploits the fact that images often contain self-similarities. Concretely, the method relies on a "block matching" procedure: Patches within the noisy image that are similar to a reference patch are denoised together. This approach has been shown to be very effective and is often considered the state-of-the-art in image denoising.

4. NLSC [74]: This is a dictionary-based method which (like KSVD) adapts the dictionary to the noisy image at hand. In addition, the method exploits image self-similarities, using a block-matching approach similar to BM3D. This method also achieves excellent results.

We choose these algorithms for our comparison because they achieve good results. BM3D and NLSC are usually referred to as the state-of-the-art in image denoising. Of the four algorithms, KSVD achieves the least impressive results, but these are still usually better than those achieved with BLSGSM [94], which was considered state-of-the-art before the introduction of KSVD. An additional reason for the choice of these algorithms is the diversity of the approaches. Learning-based approaches are represented through EPLL, whereas engineered approaches that don't rely on learning are represented by BM3D. Non-local methods are represented by BM3D and NLSC. Finally, dictionary-based approaches are represented by KSVD and NLSC.

5.5.1 Detailed comparison on one noise level

We will now compare the results achieved with an MLP to results achieved with other denoising algorithms on AWG noise with $\sigma = 25$. We choose the MLP with architecture (39×39, 3072, 3072, 2559, 2047, 17×17) because it delivered the best results. The MLP was trained for approximately $3.5 \cdot 10^8$ backprops, see Chapter 6 for details.

Comparison on 11 standard test images: Table 5.1 summarizes the comparison of our approach (MLP) to the four other denoising algorithms. Our approach achieves the best result on 7 of the 11 test images and is the runner-up on one image. However, our method is clearly inferior to BM3D and NLSC on images "Barbara" and "House". These two images contain a lot of regular structure (see Figure 5.3) and are therefore ideally suited for algorithms like BM3D and NLSC, which adapt to the noisy image. However, we outperform

Figure 5.3: We outperform BM3D on images with smooth surfaces and non-regular structures. BM3D outperforms us on images with regular structure. The image "Barbara" contains a lot of regular structure on the pants as well the table-cloth.

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP
Barbara	29.49dB	28.52dB	**30.67**dB	*30.50*dB	29.52dB
Boat	29.24dB	29.64dB	29.86dB	*29.86*dB	**29.95**dB
C.man	28.64dB	29.18dB	29.40dB	*29.46*dB	**29.60**dB
Couple	28.87dB	29.45dB	*29.68*dB	29.63dB	**29.75**dB
F.print	27.24dB	27.11dB	**27.72**dB	27.63dB	*27.67*dB
Hill	29.20dB	29.57dB	*29.81*dB	29.80dB	**29.84**dB
House	32.08dB	32.07dB	*32.92*dB	**33.08**dB	32.52dB
Lena	31.30dB	31.59dB	*32.04*dB	31.87dB	**32.28**dB
Man	29.08dB	29.58dB	29.58dB	*29.62*dB	**29.85**dB
Montage	30.91dB	31.18dB	**32.24**dB	*32.15*dB	31.97dB
Peppers	29.69dB	30.08dB	30.18dB	**30.27**dB	**30.27**dB

Table 5.1: Results on 11 standard test images for $\sigma = 25$.

KSVD on both of these images even though KSVD is also an algorithm that is well-suited for these types of images. We also note that we outperform both KSVD and EPLL on every image.

(a) (b)

Figure 5.4: The MLP outperforms BM3D on image (a). Locations where BM3D is worse than the MLP on image 198054 are highlighted (b).

Comparison on larger test sets We now compare our approach to EPLL, BM3D and NLSC on the five larger test sets defined in section 5.4. Each dataset contains 500 images, giving us a total of 2500 test images.

- Comparison to EPLL: We outperform EPLL on 2487 (99.48%) of the 2500 images, see Figure 5.5. The average improvement over all datasets is 0.35dB. On the VOC

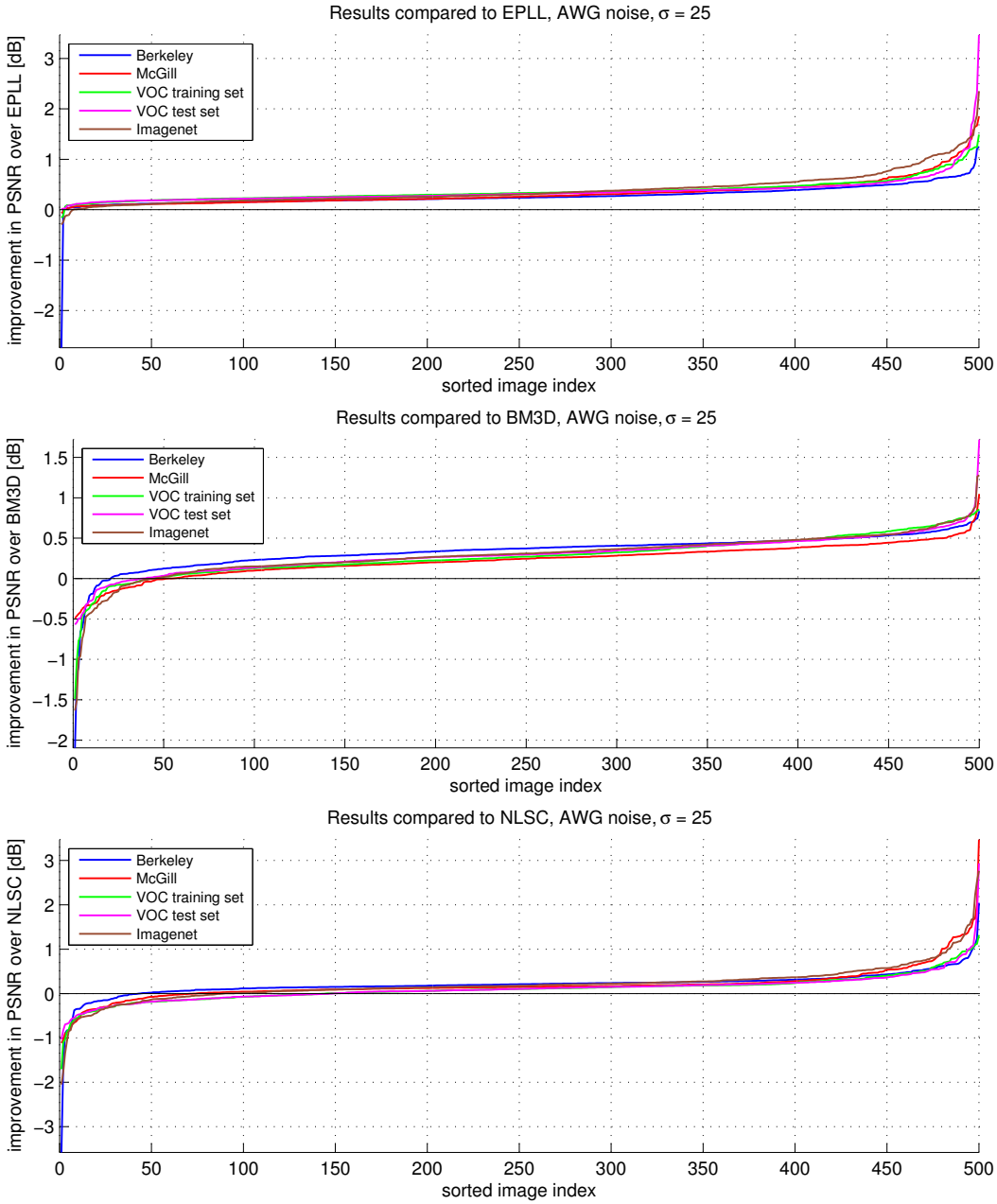


Figure 5.5: Results compared to EPLL (top), BM3D (middle) and NLSC (bottom) on five datasets of 500 images, $\sigma = 25$.

2007 test set, we outperform EPLL on every image. The best average improvement over EPLL was on the subset of the ImageNet dataset (0.39dB), whereas the smallest improvement was on the Berkeley dataset (0.27dB). This is perhaps a reflection of the fact that EPLL was trained on a subset of the Berkeley dataset, whereas our approach was trained on the ImageNet dataset. For EPLL, the test set contains the training set. For our method, this is not the case, but it is plausible that the ImageNet dataset contains some form of regularity across the whole dataset.

- Comparison to BM3D: We outperform BM3D on 2304 (92.16%) of the 2500 images, see Figure 5.5. The average improvement over all datasets is 0.29dB. The largest average improvement was on the Berkeley dataset (0.34dB), whereas the smallest average improvement was on the McGill dataset (0.23dB).

  Figure 5.4 highlights the areas of the image in the lower row of Figure 5.3 where BM3D creates larger errors than the MLP. We see that it is indeed in the areas with complicated structures (the hair and the shirt) that the MLP has an advantage over BM3D.

- Comparison to NLSC: We outperform NLSC on 2003 (80.12%) of the 2500 images, see Figure 5.5. The average improvement over all datasets was 0.16dB. The largest average improvements were on the ImageNet subset and Berkeley dataset (0.21dB), whereas the smallest average improvements were on the VOC 2011 training set and VOC 2007 test set (0.10dB and 0.11dB respectively). This is perhaps a reflection of the fact that the initial dictionary of NLSC was trained on a subset of the VOC 2007 dataset [74].

In summary, our method outperforms state-of-the-art denoising algorithms for AWG noise with $\sigma = 25$. The improvement is consistent across datasets. We notice that our method tends to outperform BM3D on images with smooth areas such as the sky and on images which contain irregular structure, such as the hair of the woman in Figure 5.3. The fact that our method performs well on smooth surfaces can probably be explained by the fact that our method uses large input patches: This allows our method to handle low frequency noise. Methods using smaller patches (such as BM3D) are blind to lower frequencies. The fact that our method performs better than BM3D on images with irregular structures is explained by the block-matching approach employed by BM3D: The method cannot find similar patches in images with irregular textures. Further examples of images where our method outperforms BM3D are shown in Figure 5.6 and where BM3D outperforms our method in Figure 5.7.

5.5.2 Comparison on different noise variances

We have seen that our method achieves state-of-the-art results on AWG noise with $\sigma = 25$. We now evaluate our approach on other noise levels. We use $\sigma = 10$ (low noise), $\sigma = 50$ (high noise), $\sigma = 75$ (very high noise) and $\sigma = 170$ (extremely high noise) for this purpose. We describe in Chapter 6 which architectures and patch sizes are used for the various noise levels.

Comparison on 11 standard test images: Table 5.2 compares our method against KSVD, EPLL, BM3D and NLSC on the test set of 11 standard test images for $\sigma = 10$. Our method outperforms KSVD on ten images, EPLL on all images, BM3D on four images and NLSC on three images. Our method achieves the best result of all algorithms on two images. Like for $\sigma = 25$, BM3D and NLSC perform particularly well for images "Barbara" and "House".

Table 5.3 performs the same comparison for $\sigma = 50$. Our method outperforms all others on 8 of the 11 images. BM3D and NLSC still perform significantly better on the image "Barbara". We outperform KSVD and EPLL on every image.

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP
Barbara	34.40dB	33.59dB	**34.96**dB	**34.96**dB	34.07dB
Boat	33.65dB	33.64dB	*33.89*dB	**34.02**dB	33.85dB
C.man	33.66dB	33.99dB	34.08dB	**34.15**dB	*34.13*dB
Couple	33.51dB	33.82dB	**34.02**dB	*33.98*dB	33.89dB
F.print	32.39dB	32.12dB	32.46dB	*32.57*dB	**32.59**dB
Hill	33.37dB	33.49dB	*33.60*dB	**33.66**dB	33.59dB
House	35.94dB	35.74dB	*36.71*dB	**36.90**dB	35.94dB
Lena	35.46dB	35.56dB	**35.92**dB	35.85dB	*35.88*dB
Man	33.53dB	33.94dB	33.97dB	*34.06*dB	**34.10**dB
Montage	35.91dB	36.45dB	**37.37**dB	*37.24*dB	36.51dB
Peppers	34.20dB	34.54dB	34.69dB	**34.78**dB	*34.72*dB

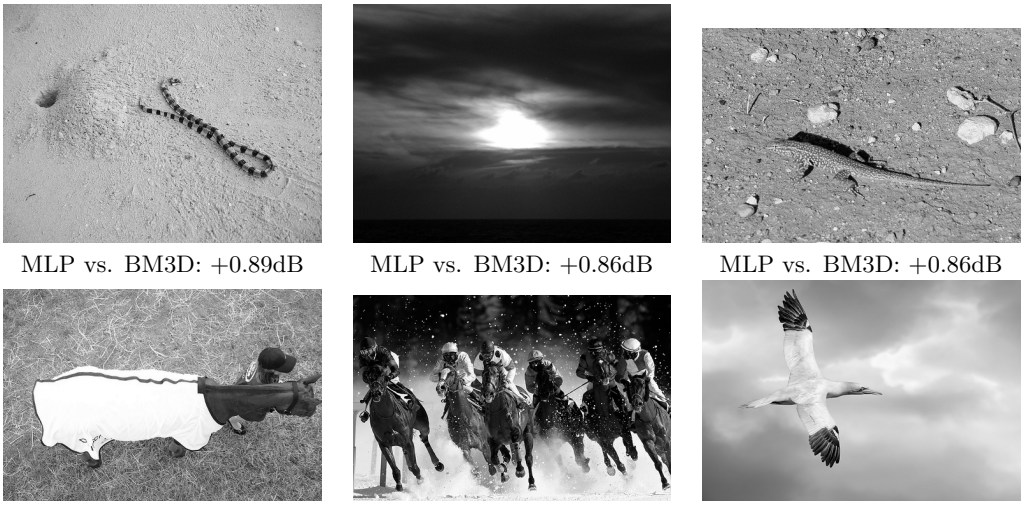
Table 5.2: Results on 11 standard test images for $\sigma = 10$.

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP
Barbara	25.22dB	24.83dB	**27.21**dB	*27.13*dB	25.37dB
Boat	25.90dB	26.59dB	26.72dB	*26.73*dB	**27.02**dB
C.man	25.42dB	26.05dB	26.11dB	*26.36*dB	**26.42**dB
Couple	25.40dB	26.24dB	*26.43*dB	26.33dB	**26.71**dB
F.print	23.24dB	23.59dB	**24.53**dB	*24.25*dB	24.23dB
Hill	26.14dB	26.90dB	*27.14*dB	27.05dB	**27.32**dB
House	27.44dB	28.77dB	*29.71*dB	**29.88**dB	29.52dB
Lena	27.43dB	28.39dB	*28.99*dB	28.88dB	**29.34**dB
Man	25.83dB	26.68dB	*26.76*dB	26.71dB	**27.08**dB
Montage	26.42dB	27.13dB	27.69dB	*28.02*dB	**28.07**dB
Peppers	25.91dB	26.64dB	26.69dB	*26.73*dB	**26.74**dB

Table 5.3: Results on 11 standard test images for $\sigma = 50$.

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP
Barbara	22.65dB	22.95dB	**25.10**dB	*25.03*dB	23.48dB
Boat	23.59dB	24.86dB	*25.04*dB	24.95dB	**25.43**dB
C.man	23.04dB	24.19dB	*24.37*dB	24.24dB	**24.72**dB
Couple	23.43dB	24.46dB	*24.71*dB	24.48dB	**25.09**dB
F.print	20.72dB	21.44dB	**22.83**dB	*22.48*dB	22.41dB
Hill	24.21dB	25.42dB	*25.60*dB	25.57dB	**25.97**dB
House	24.53dB	26.69dB	27.46dB	*27.64*dB	**27.75**dB
Lena	24.87dB	26.50dB	27.16dB	*27.17*dB	**27.66**dB
Man	23.76dB	25.07dB	*25.29*dB	25.15dB	**25.63**dB
Montage	23.58dB	24.86dB	*25.36*dB	25.20dB	**25.93**dB
Peppers	23.09dB	24.52dB	*24.71*dB	24.46dB	**24.87**dB

Table 5.4: Results on 11 standard test images for $\sigma = 75$.



MLP vs. BM3D: +0.89dB | MLP vs. BM3D: +0.86dB | MLP vs. BM3D: +0.86dB

MLP vs. BM3D: +0.84dB | MLP vs. BM3D: +0.82dB | MLP vs. BM3D: +0.82dB

Figure 5.6: Images where the MLP outperforms BM3D, for $\sigma = 25$. The images contain smooth areas, irregular structures, or both.

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP
Barbara	18.08dB	20.79dB	19.74dB	*20.99*dB	**21.37**dB
Boat	18.42dB	*21.60*dB	20.49dB	21.48dB	**22.47**dB
C.man	18.00dB	20.48dB	19.65dB	*20.50*dB	**21.28**dB
Couple	18.26dB	*21.48*dB	20.39dB	21.29dB	**22.16**dB
F.print	16.75dB	17.06dB	17.46dB	*18.51*dB	**18.57**dB
Hill	18.69dB	*22.63*dB	20.98dB	22.62dB	**23.33**dB
House	18.20dB	*22.52*dB	21.19dB	21.95dB	**23.80**dB
Lena	18.68dB	22.96dB	21.38dB	*23.20*dB	**24.24**dB
Man	18.49dB	*22.10*dB	20.59dB	21.72dB	**22.85**dB
Montage	17.91dB	*20.48*dB	19.69dB	20.40dB	**20.93**dB
Peppers	17.47dB	*20.26*dB	19.58dB	19.53dB	**20.81**dB

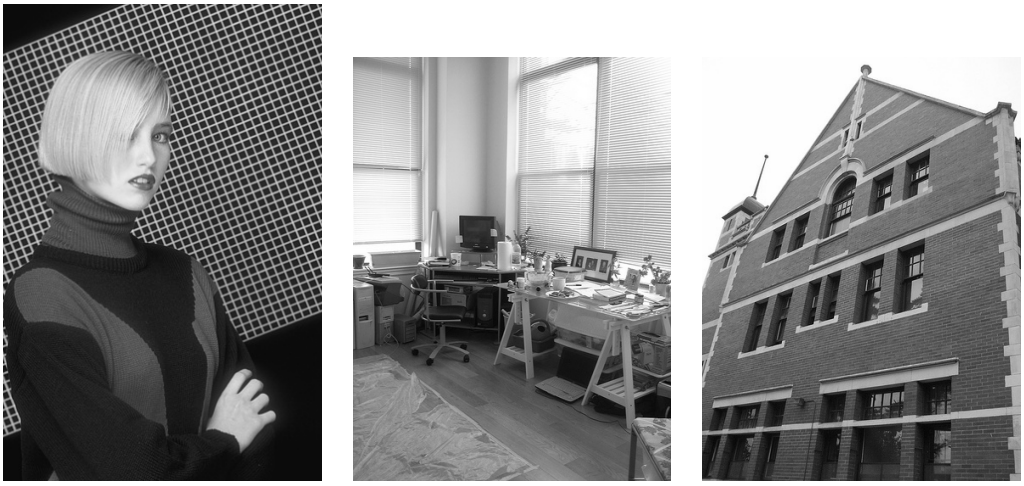
Table 5.5: Results on 11 standard test images for $\sigma = 170$.

For $\sigma = 75$, our method outperforms all others on 9 of the 11 images, see Table 5.4. BM3D and NLSC still perform significantly better on the image "Barbara".

For $\sigma = 170$, our method outperforms all other methods on all images, see Table 5.5. It was suggested by Levin and Nadler [68] that image priors are not useful at extremely high noise levels. However, our results suggest otherwise: Our method is the best-performing method on this noise level. The second best performing method, EPLL, is also a prior-based method. The improvement of our method over BM3D (which is not prior-based) is often very high (almost 3dB on image "Lena").

Comparison on 2500 test images: Figure 5.8 (top) compares the results achieved with an MLP on $\sigma = 10$ to BM3D. We outperform BM3D on 1876 (75.04%) of the 2500 images. The average improvement over all images is 0.1dB. The largest average improvement is on the McGill dataset (0.27dB), whereas the smallest average improvement is on the VOC training set (0.02dB). The improvement in PSNR is very small on the VOC training set, but we observe an improvement on 301 (60.2%) of the 500 images.

Figure 5.8 (middle) compares the results achieved with an MLP on $\sigma = 50$ to BM3D. We outperform BM3D on 2394 (95.76%) of the 2500 images. The average improvement over all



MLP vs. BM3D: -2.09dB | MLP vs. BM3D: -1.03dB | MLP vs. BM3D: -0.75dB

MLP vs. BM3D: -1.09dB | MLP vs. BM3D: -0.66dB | MLP vs. BM3D: -0.54dB

Figure 5.7: Images where BM3D outperforms the MLP, for $\sigma = 25$. All images contain regular structures. The chair (bottom right) contains a regular dotted pattern.

datasets is 0.32dB. The largest average improvement is on the Berkeley dataset (0.36dB), whereas the smallest average improvement is on the McGill dataset (0.27dB). This is an even greater improvement over BM3D than on $\sigma = 25$, see Figure 5.5.

Figure 5.8 (bottom) compares the results achieved with an MLP on $\sigma = 75$ to BM3D. We outperform BM3D on 2440 (97.60%) of the 2500 images. The average improvement over all datasets is 0.36dB. The average improvement is almost the same for all datasets, ranging from 0.34 to 0.37dB.

Adaptation to other noise levels: How do the MLPs perform on noise levels they have not been trained on? Figure 5.9 summarizes the results achieved by MLPs on noise levels they have not been trained on and compares these results to BM3D. The results are averaged over the 500 images in the Berkeley dataset. We varied σ between 5 and 100 in steps of 5. We see that the MLPs achieve better results than BM3D on the noise levels they have been trained on. However, the performance degrades quickly for noise levels they have not been trained on. Exceptions are the MLPs trained on $\sigma = 50$ and $\sigma = 75$, which also outperform BM3D on $\sigma = 45$ and $\sigma = 55$ (for the MLP trained on $\sigma = 50$) and $\sigma = 70$ and $\sigma = 80$ (for the MLP trained on $\sigma = 75$).

We conclude than our method is particularly well suited for medium to high noise levels. We outperform the previous state-of-the-art on all noise levels, but for $\sigma = 10$, the improvement is rather small (0.1dB). However, our method has to be trained on each noise level in order to achieve good results.

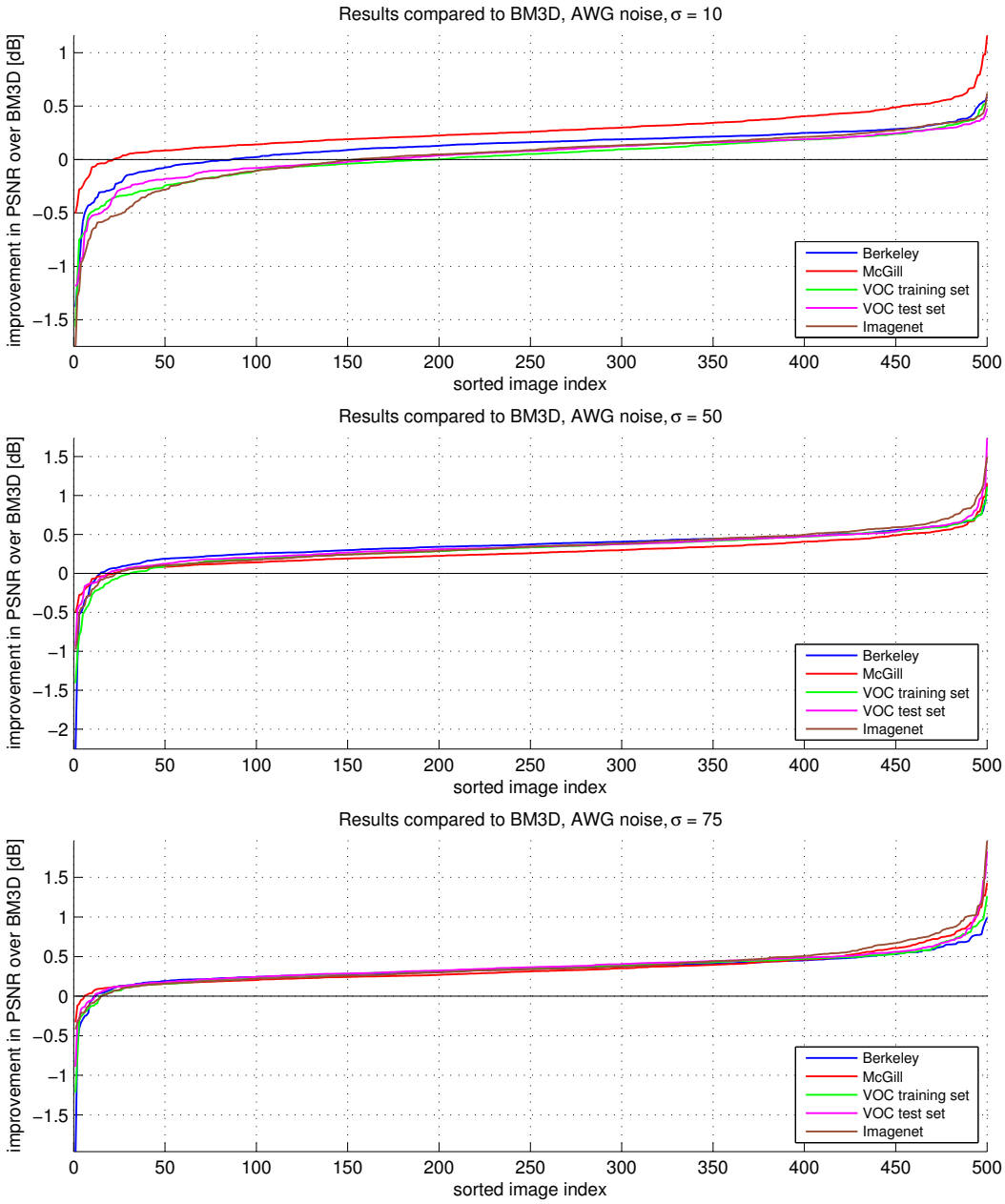


Figure 5.8: Results compared to BM3D on five datasets of 500 images and different noise levels. Top: $\sigma = 10$, middle: $\sigma = 50$, bottom: $\sigma = 75$.

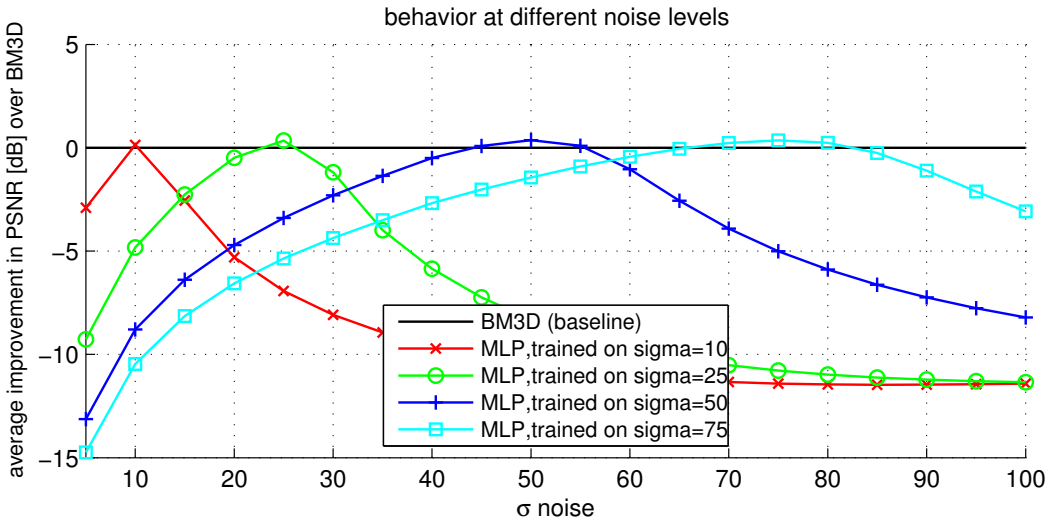


Figure 5.9: Results achieved on different noise levels. Results are averaged over the 500 images in the Berkeley dataset.

5.6 Results: Comparison with theoretical bounds

It has been observed that recent denoising algorithms tend to perform approximately equally well [22], which naturally raises the question of whether recent state-of-the-art algorithms are close to an inherent limit on denoising quality. Two approaches to estimating bounds on denoising performance have been followed [22, 68]. We will relate the results obtained by our algorithm to these bounds.

Figure 5.10: Images "Mandrill" and "Parrot". For $\sigma = 25$, the theoretical bounds estimated by Chatterjee and Milanfar [22] are very close to the result achieved by BM3D: 25.61dB and 28.94dB, respectively. Our results outperform these bounds and are 26.01dB and 29.25dB respectively.

5.6.1 Clustering-based bounds

Chatterjee and Milanfar [22] derive bounds on image denoising capability. The authors make a "cluster" assumption about images: Each patch in a noisy image is assigned to one of a finite number of clusters. Clusters with more patches are denoised better than clusters with fewer patches. According to their bounds, improvements over existing denoising algorithms are mainly to be achieved on images with simple geometric structure (the authors use a synthetic "box" image as an example), whereas current denoising algorithms (and BM3D in particular) are already very close to the theoretical bounds for images with richer geometric structure.

Figure 5.10 shows two images with richer structure and on which BM3D is very close to the estimated theoretical bounds for $\sigma = 25$ [22]. Very little, if any, improvement is expected on these images. Yet, we outperform BM3D by 0.4dB and 0.31dB on these images, which is a significant improvement.

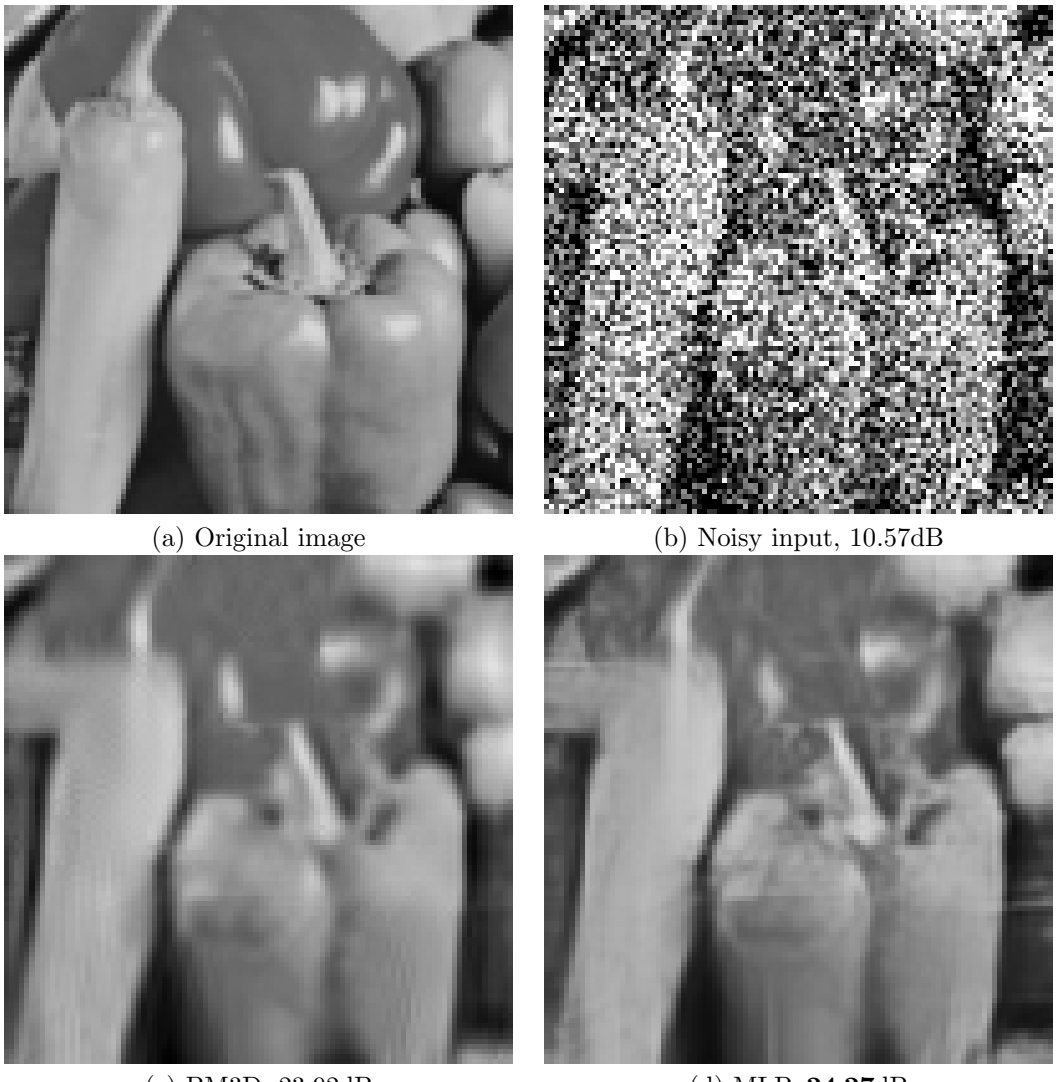
The MLP does not operate according to the cluster assumption (it operates on a single patch at a time) and it performs particularly well on images with rich geometric structure. We therefore speculate that the cluster assumption might not be a reasonable assumption to derive ultimate bounds on image denoising quality.

5.6.2 Bayesian bounds

Levin and Nadler [68] derive bounds on how well any denoising algorithm can perform. The bounds are dependent on the patch size, where larger patches lead to better results. For

	worst	best	mean
BLSGSM [94]	22.65dB	23.57dB	23.15dB
KSVD [1]	21.69dB	22.59dB	22.16dB
NLSC [74]	21.39dB	22.49dB	21.95dB
BM3D [25]	*22.94*dB	23.96dB	23.51dB
BM3D, step1 [25]	21.85dB	22.79dB	22.35dB
EPLL [132]	22.94dB	*24.07*dB	*23.56*dB
MLP	**23.32**dB	**24.34**dB	**23.85**dB

Table 5.6: Comparison of results achieved by different methods on the down-sampled and cropped "Peppers" image for $\sigma = 75$ and 100 different noisy instances.



(a) Original image (b) Noisy input, 10.57dB

(c) BM3D, 23.92dB (d) MLP, **24.27**dB

Figure 5.11: For image (a) and $\sigma = 75$, the best achievable result estimated by Levin and Nadler [68] is only 0.07dB better than the result achieved by BM3D (exact dB values are dependent on the noisy instance). On average, our results are 0.34dB better than BM3D.

large patches and low noise, tight bounds cannot be estimated. On the image depicted in Figure 5.11a (a down-sampled and cropped version of the image "Peppers") and for noise level $\sigma = 75$, the theoretically best achievable result using patches of size 12×12 is estimated to be 0.07dB better than BM3D (23.86dB for BM3D and 23.93 for the estimated bound).

We tested an MLP trained on $\sigma = 75$ as well as other methods (including BM3D) on the same image and summarize the results in Table 5.6. We used 100 different noisy versions of the same clean image and report the worst, best and average results obtained. For BM3D, we obtain results that are in agreement with those obtained by Levin and Nadler [68], though we note that the difference between the worst and best result is quite large: Approximately 1dB. The high variance in the results is due to the fact that the test image is relatively small and the noise variance quite high. The results obtained with BLSGSM and KSVD are also in agreement with those reported by Levin and Nadler [68]. NLSC achieves results that are much worse than those obtained by BM3D on this image and this noise level. EPLL achieves results that are on par with those achieved by BM3D.

BM3D achieves a mean PSNR of 23.51dB and our MLP achieves a mean PSNR of 23.85dB, an improvement of 0.34dB. Visually, the difference is noticeable, see Figure 5.11. This is a much greater improvement than was estimated to be possible by Levin and Nadler [68], using patches of size 12×12. This is possible because of the fact that we used larger patches. Levin and Nadler [68] were unable to estimate tight bounds for larger patch sizes because of their reduced density in the dataset of clean patches.

Levin and Nadler [68] describe BM3D as a method that uses patches of size 12×12. However, BM3D is a two-step procedure. It is true that BM3D uses patches of size 12×12 (for noise levels above $\sigma = 40$) in its first step. However, the second step of the procedure effectively increases the support size: In the second step, the patches "see beyond" what they would have seen in the first step, but it is difficult to say by how much the support size is increased by the second step. Therefore, a fairer comparison would have been to compare the estimated bounds against only the first step of BM3D. If only the first step of BM3D is used, the mean result is 22.35dB. Therefore, if the constraint on the patch sizes is strictly enforced for BM3D, the difference between the theoretically best achievable result and BM3D is larger than suggested by Levin and Nadler [68].

5.6.3 Bayesian bounds with unlimited patch size

More recently, bounds on denoising quality achievable using any patch size have been suggested by Levin *et al.* [69]. This was done by extrapolating bounds similar to those suggested by Levin and Nadler [68] to larger patch sizes (including patches of infinite size). For $\sigma = 50$ and $\sigma = 75$, the bounds lie 0.7dB and 1dB above the results achieved by BM3D, respectively. The improvements of our approach over BM3D on these noise levels (estimated on 2500 images) are 0.32dB and 0.36dB, respectively. Our approach therefore reaches respectively 46% and 36% of the remaining possible improvement over BM3D. Furthermore, Levin *et al.* [69] suggest that increasing the patch size suffers from a law of diminishing returns. This is particularly true for textured image content: The larger the patch size, the harder it becomes to find enough training data. Levin *et al.* [69] therefore suggest that increasing the patch size should be the most useful for smooth image content. The observation that our method performs much better than BM3D on images with smooth areas (see middle row in Figure 5.3) is in agreement with this statement. The fact that image denoising is cursed with a law of diminishing returns also suggests that the remaining available improvement will be increasingly difficult to achieve. However, Levin *et al.* [69] suggest that patch-based denoising can be improved mostly in flat areas and less in textures ones. Our observation that the MLP performs particularly well in areas with complicated structure (such as on the bottom image in Figure 5.3 or both images in Figure 5.10) shows that large improvements over BM3D on images with complicated textures are possible.

5.6.4 Bayesian bounds correlate with the results achieved with MLPs

We have previously seen (Figures 5.3, 5.6, and 5.7) that we outperform BM3D on smooth regions and regions with irregular structure, whereas BM3D outperforms our method on images with repeating structure. An explanation for this phenomenon is that BM3D is based on a block-matching procedure and cannot find matching blocks in images with irregular textures and therefore does not perform very well. In this section, we provide an explanation for this phenomenon that is related to the Bayesian denoising bounds with finite patch sizes.

The Bayesian denoising estimate given by Levin and Nadler [68] is given by:

$$\hat{\mu}(y_c) = \frac{\frac{1}{N}\sum_i p(y|x_i)x_{i,c}}{\frac{1}{N}\sum_i p(y|x_i)}, \tag{5.3}$$

where x_i is a clean image patch and $x_{i,c}$ is the center pixel of that patch. The pixel y_c to be denoised lies in the center of a larger patch y. The conditional probability $p(y|x_i)$ is given by:

$$p(y|x_i) = \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{||x_i - y||^2}{2\sigma^2}} \tag{5.4}$$

for AWG noise. It is shown by Levin and Nadler [68] that the Bayesian denoising estimate $\hat{\mu}(y_c)$ indeed converges to the true Bayesian minimum mean squared error (MMSE) estimator if N tends to infinity. Hence, a noisy pixel y_c can be denoised by comparing how well clean patches x_i can explain the noisy patch y and performing a weighted average for large N. Achieving good results with this approach is extremely time-consuming, but can be parallelized. In addition to the *single-pixel* Bayesian denoising estimate proposed by Levin and Nadler [68], we propose the *full-patch* Bayesian denoising estimate

$$\hat{\mu}(y) = \frac{\frac{1}{N}\sum_i p(y|x_i)x_i}{\frac{1}{N}\sum_i p(y|x_i)}, \tag{5.5}$$

where $\hat{\mu}(y)$ is a whole patch.

Experiments: We use two images of size 100×100 pixels and corrupted with AWG noise, $\sigma = 75$, see Figure 5.12. We observe the evolution of both the single pixel and the full patch Bayesian estimate for growing N, see Figure 5.13. Following [68], we use the LabelMe dataset [103] as a source of clean image patches for the Bayesian denoising estimates and use patches of size 12×12. As a comparison, we provide the results achieved with BM3D [25] and with two MLPs trained on $\sigma = 75$. For BM3D, we provide (i) the results achieved with only the first step of the algorithm, and (ii) the results achieved with both steps. The first MLP we use in the comparison has an input patch size of 39×39 and an output patch size of 17×17, whereas the second MLP uses input and output patches of the size 12×12 and is therefore more comparable to the denoising approach using the Bayesian full-patch estimate.

Results: BM3D outperforms the MLP on the pants image, whereas the MLP outperforms BM3D on the peppers image. The MLPs are barely able to recover the regular stripes on the pants, whereas BM3D recovers the stripes quite well, see Figure 5.14. The fact that BM3D outperforms the MLP on that image was expected: The image is very regular and contains many patches that look similar. The MLP outperforms BM3D on the peppers image. We see that the Bayesian full patch estimate $\hat{\mu}(y)$ always outperforms the Bayesian single-pixel estimate $\hat{\mu}(y_c)$. This can be explained by the averaging in areas of overlapping patches: Estimation errors are averaged out to some extent.

For the Bayesian estimates, we observe that the PSNR tends to increase with growing N, though this is not always the case. We note that there is no guarantee that the PSNR always increases: The PSNR is merely guaranteed to approach the true MMSE as N tends

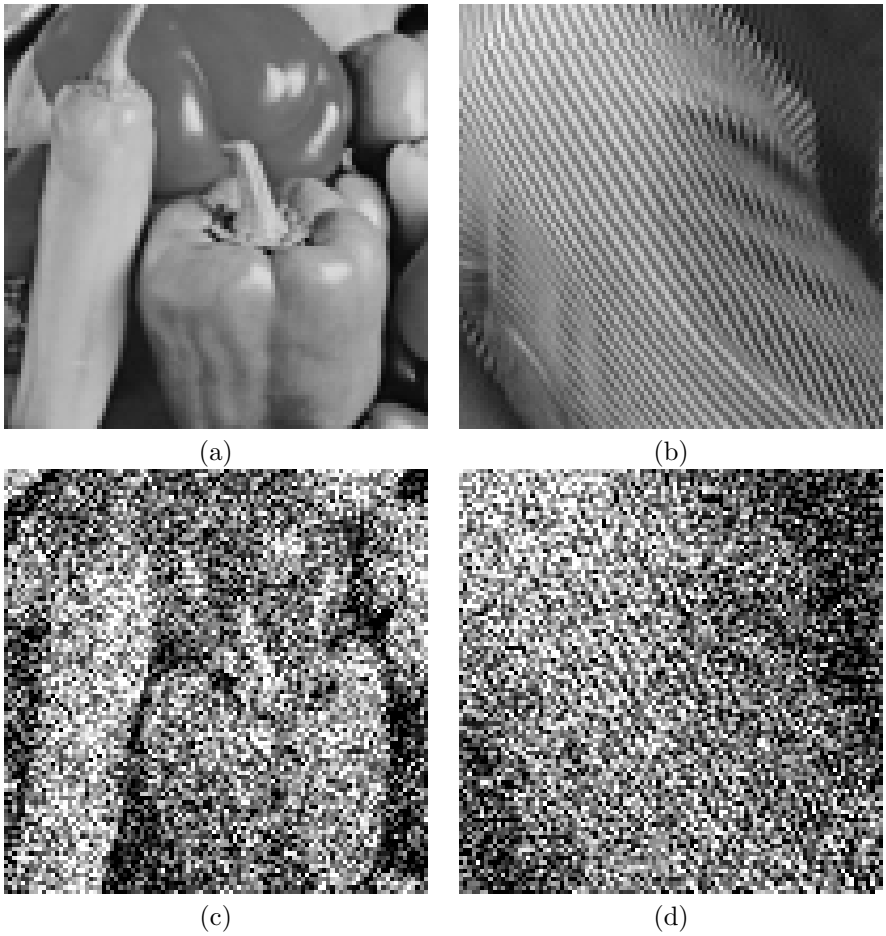


Figure 5.12: Image (a) is a down-sampled and cropped version of the "peppers" image and has already been used by Levin and Nadler [68]. Image (b) shows a part of the pants in image "Barbara". Images (c) and (d) show the noisy versions of these images, using $\sigma = 75$.

to infinity, but it is possible for the PSNR to decrease with growing N. This can happen when the "best" clean image patches are presented first: In an extreme scenario, the clean version of the image to be denoised is contained in the dataset of clean image patches. If the clean version of the noisy image is visited first, the denoising result will be very good. The remaining patches in the dataset will tend to decrease the PSNR value.

For the peppers image, the Bayesian bound estimates slightly outperform BM3D. This is in agreement with the findings of Levin and Nadler [68], though we were able to observe this effect after only $N = 10^8$ clean image patches instead of $N = 10^{10}$ used in [68]. The MLP using the larger input patches (39×39) performs slightly better than the MLP using the smaller (12×12) patches. Both MLPs perform slightly better than the Bayesian bounds. For the MLP using 39×39 patches, this can be explained by the fact that it is not bounded by the bounds estimated for 12×12 patches. For the MLP using 12×12 patches, this can presumably be explained by the fact that an insufficient number of clean patches were used to accurately estimate the MMSE. We also note that using only the first step of BM3D yields results that are significantly worse than those obtained using both steps of BM3D. The results obtained using only the first step of BM3D are quickly outperformed using the Bayesian estimates.

For the pants image, the Bayesian estimates are poor and barely progress with growing N. The MLPs give somewhat better results than the Bayesian estimates, but BM3D out-

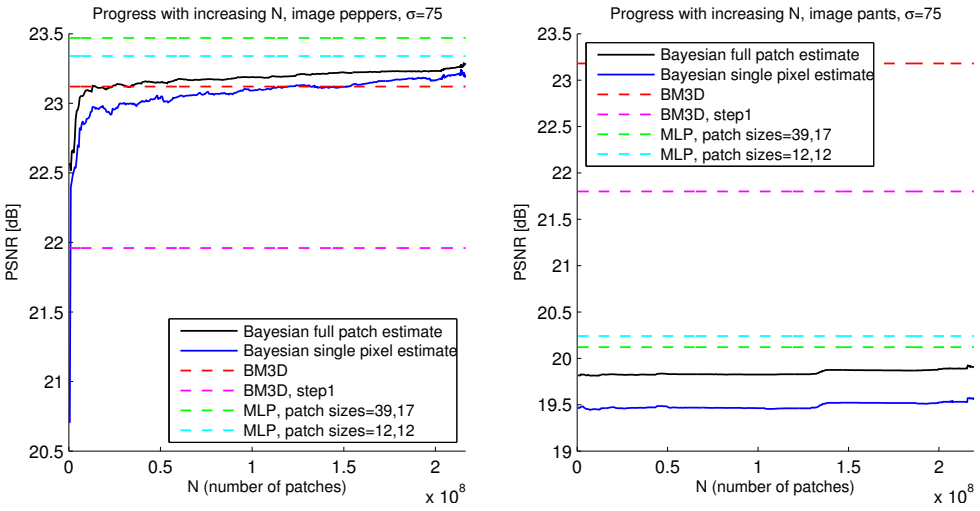


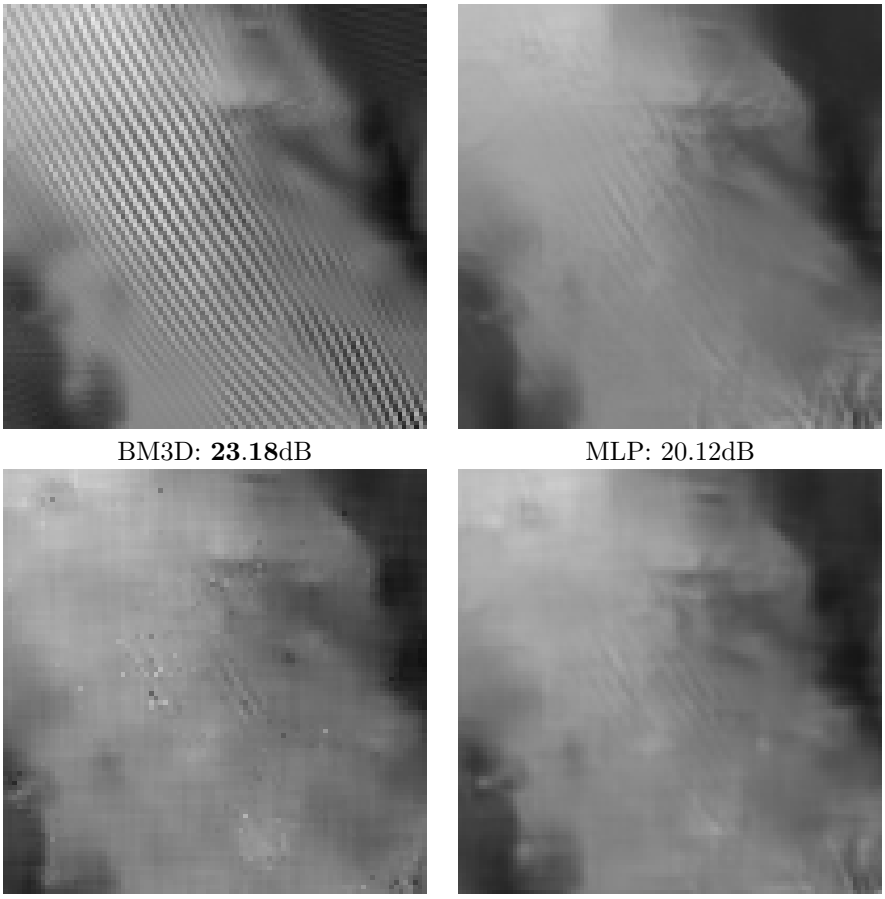
Figure 5.13: Estimated MMSE with growing number of clean patches. For the pants image, there is barely any progress. The full-patch estimate always provides better results than the single pixel estimate.

performs the MLPs by approximately 3dB, which is a significant improvement. Even when only the first step of BM3D is used, the results are still much better than those obtained with the MLPs. It is interesting that for this image, the MLP using smaller patches slightly outperforms the MLP using larger patches.

Conclusion: The pants image is difficult to denoise using the Bayesian approach because it is not well represented by the distribution of natural image patches. The same argument holds true for the MLPs: The MLPs are trained on a large dataset of natural image patches. They do not perform well on image patches that do not resemble those in the training dataset. The MLPs are therefore a method that is similar to the Bayesian approach. We expect the results achieved by both approaches to correlate (and indeed observe such a correlation on the peppers and pants images).

BM3D performs particularly well on the pants image compared to both the MLP and the Bayesian approach because of a combination of two reasons: i) The patches in the pants image are uncommon in the set of natural image patches, and ii) many patches in the pants image are similar to other patches in the same image. BM3D is therefore ideally suited for images such as the pants image.

The fact that the MLP using smaller patches achieves better results than the MLP using larger patches on the pants image is counterintuitive, but is in agreement with the findings of Levin *et al.* [69] suggesting denoising algorithms with variable patch sizes: Large patches for smooth areas and smaller patches for textured areas. Large patches for textures areas can be a disadvantage because one cannot find enough clean patches resembling such patches.



BM3D: **23**.**18**dB | MLP: 20.12dB

Bayesian single pixel estimate $\hat{\mu}(y_c)$: 19.56dB | Bayesian full patch estimate $\hat{\mu}(y)$: 19.90dB

Figure 5.14: Results obtained with various methods on image "pants", corrupted with AWG noise, $\sigma = 75$. The Bayesian bound estimates and the MLP are barely able to recover the stripes, whereas BM3D recovers the stripes well.

5.7 Results: comparison on non-AWG noise

Virtually all denoising algorithms assume the noise to be AWG. However, images are not always corrupted by AWG noise. Noise is not necessarily additive, white, Gaussian and signal independent. For instance in some situations, the imaging process is corrupted by Poisson noise (such as photon shot noise). Denoising algorithms which assume AWG noise might be applied to such images using some image transform [78]. Rice-distributed noise, which occurs in magnetic resonance imaging, can be handled similarly [36].

In most cases however, it is more difficult or even impossible to find Gaussianizing transforms. In such cases, a possible solution is to create a denoising algorithm specifically designed for that noise type. MLPs allow us to effectively learn a denoising algorithm for a given noise type, provided that noise can be simulated. In the following, we present results on three noise types that are different from AWG noise. We make no effort to adapt our architecture or procedure in general to the specific noise type but rather use an architecture that yielded good results for AWG noise (four hidden layers of size 2047 and input and output patches of size 17×17).

5.7.1 Stripe noise

It is often assumed that image data contains structure, whereas the noise is uncorrelated and therefore unstructured. In cases where the noise also exhibits structure, this assumption is violated and denoising results become poor. We here show an example where the noise is additive and Gaussian (with $\sigma = 50$), but where 8 horizontally adjacent noise values have the same value.

Since there is no canonical denoising algorithm for this noise, we choose BM3D as the competitor. An MLP trained on 82 million training examples outperforms BM3D for this type of noise, see left column of Figure 5.15.

5.7.2 Salt and pepper noise

When the noise is additive Gaussian, the noisy image value is still correlated to the original image value. With salt and pepper noise, noisy values are not correlated with the original image data. Each pixel has a probability p of being corrupted. A corrupted pixel has probability 0.5 of being set to 0; otherwise, it is set to highest possible value (255 for 8-bit images). We show results with $p = 0.2$.

A common algorithm for removing salt and pepper noise is median filtering. We achieved the best results with a filter size of 5×5 and symmetrically extended image boundaries. We also experimented with BM3D (by varying the value of σ) and achieved a PSNR of 25.55dB. An MLP trained on 88 million training examples outperforms both methods, see middle column of Figure 5.15.

The problem of removing salt and pepper noise is reminiscent of the in-painting problem, except that it is not known which pixels are to be in-painted. If one assumes that the positions of the corrupted pixels are known, the pixel values of the non-corruped pixels can be copied from the noisy image, since these are identical to the ground truth values. Using this assumption, we achieve 36.53dB with median filtering and 38.64dB with the MLP.

5.7.3 JPEG quantization artifacts

Such artifacts occur due to the JPEG image compression algorithm. The quantization process removes information, therefore introducing noise. Characteristics of JPEG noise are blocky images and loss of edge clarity. This kind of noise is not random, but rather completely determined by the input image. In our experiments we use JPEG's quality setting $Q = 5$, creating visible artifacts.

Figure 5.15: Comparison of our method to other on stripe noise (left), salt-and-pepper noise (middel) and JPEG quantization artifacts (right). BM3D is not designed for stripe noise.

image	peak	GAT+BM3D	UWT/BDCT [80]	UWT/BDCT [71]	MLP
Barbara	1	20.83dB	-	20.79dB	**21.44**dB
Barbara	20	**27.52**dB	-	27.33dB	26.08dB
Cameraman	1	20.34dB	20.35dB	20.48dB	**21.66**dB
Cameraman	20	26.83dB	25.92dB	**26.93**dB	**26.93**dB
Lena	1	22.96dB	22.83dB	-	**24.26**dB
Lena	20	29.39dB	28.46dB	-	**29.89**dB
Fluo.cells	1	24.54dB	25.13dB	25.25dB	**25.56**dB
Fluo.cells	20	29.66dB	29.47dB	31.00dB	29.98dB
Moon	1	22.84dB	-	**23.49**dB	23.48dB
Moon	20	25.28dB	-	**26.33**dB	25.71dB

Table 5.7: Comparison of MLPs against two competing methods on mixed Poisson-Gaussian noise. The MLPs perform particularly well when the noise is strong (peak = 1), but are also competitive on lower noise.

A common method to enhance JPEG-compressed images is to shift the images, re-apply JPEG compression, shift back and average (see Nosratinia [87]). This method achieves a PSNR of 28.42dB on our image. We also compare against the state-of-the-art in JPEG de-blocking [37].

An MLP trained on 58 million training examples with that noise outperforms both methods, see right column of Figure 5.15. In fact, the method described by Nosratinia [87] achieves an improvement of only 1.09dB over the noisy image, whereas our method achieves an improvement of 2.09dB. SA-DCT [37] achieves an improvement of 1.63dB.

5.7.4 Mixed Poisson-Gaussian noise

In photon-limited imaging, observations are usually corrupted by mixed Poisson-Gaussian noise [80, 71]. Observations are assumed to come from the following model:

$$z = \alpha p + n, \tag{5.6}$$

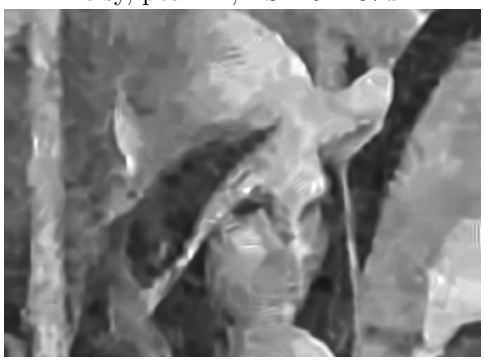
where p is Poisson-distributed with mean x and n is Gaussian-distributed with mean 0 and variance σ^2. One can regard x to be the underlying "true" image of which one wishes to make a noise-free observation. To generate a noisy image from a clean one, we follow the setup used by Mäkitalo and Foi [80] and Luisier *et al.* [71]: We take the clean image and scale it to a given peak value, giving us x. Applying (5.6) gives us a noisy image z.

Two canonical approaches exist for denoising in the photon-limited setting: (i) Applying a variance stabilizing transform on the noisy image, running a denoising algorithm designed for AWG noise (such as BM3D) on the result and finally applying the inverse of the variance stabilizing transform, and (ii) designing a denoising algorithm specifically for mixed Poisson-Gaussian noise. GAT+BM3D [80] is an example of the first approach, whereas UWT/BDCT PURE-LET [71] is an example of the second approach. In the case where a variance-stabilizing transform such as the Anscombe transformation or the generalized Anscombe transformation (GAT) [114] is applied, the difficulty lies in the design of the inverse transform [76, 77, 78, 79]. Designing a denoising algorithm specifically for Poisson-Gaussian noise is also a difficult task, but can potentially lead to better results.

Our approach to denoising photon-limited data is to train an MLP on data corrupted with mixed Poisson-Gaussian noise. We trained an MLP on noisy images using a peak value of 1 and another MLP for peak value 20, both on 60 million examples. For the Gaussian noise, we set σ to the peak value divided by 10, again following the setup used by Mäkitalo and Foi [80] and Luisier *et al.* [71]. We compare our results against GAT+BM3D [80], which is considered state-of-the-art. We compare on further images in Table 5.7. For

noisy, peak=1, PSNR: 2.87dB

noisy, peak=20, PSNR: 14.53dB



GAT+BM3D [80]
PSNR: 22.90dB

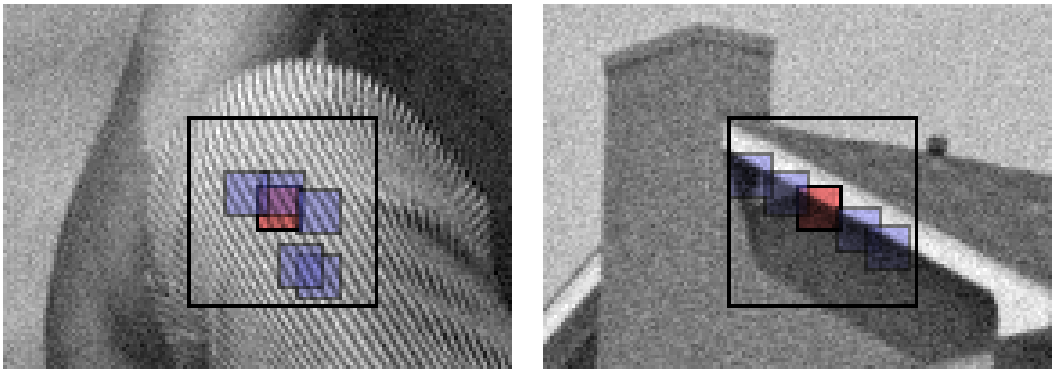
GAT+BM3D [80]
PSNR: 29.36dB

MLP: **24.26**dB

MLP: **29.89**dB

Figure 5.16: Comparison of our method to GAT+BM3D [80] on images corrupted with mixed Poisson-Gaussian noise, which occurs in photon-limited imaging.

UWT/BDCT PURE-LET [71], we noticed a discrepancy between the results reported by Mäkitalo and Foi [80] and by Luisier *et al.* [71] and therefore report both. We see that the MLPs outperform the state-of-the art on image "Lena" in both settings.



(a) (b)

Figure 5.17: Block matching: The goal of the procedure is to find the patches most similar to the reddish ("reference") patch. The neighbors (blueish patches) have to be found within a search region (represented by the larger black bounding box). Patches can overlap. Here, the procedure was applied on (a) the "Barbara" image and (b) the "House" image, both corrupted with AWG noise with $\sigma = 10$. This figure is inspired by Figure 1 in [25].

5.8 Combining BM3D and MLPs: Block-matching MLPs

Many recent denoising algorithms rely on a block-matching procedure. This most notably includes BM3D [25], but also NLSC [74]. The idea is to find patches similar to a reference patch and to exploit these "neighbor" patches for better denoising. More precisely, the procedure exploits the fact that the noise in the different patches is *independent*, whereas the (clean) image content is *correlated*. Figure 5.17 shows the effect of the procedure on two images.

Since this technique has been used with so much success, we ask the question: Can MLPs be combined with a block matching procedure to achieve better results? In particular, can we achieve better results on images where we perform rather poorly compared to BM3D and NLSC, namely images with repeating structure? To answer this question, we train MLPs that take as input not only the *reference* patch, but also its k nearest *neighbors* in terms of ℓ_2 distance. We will see that such *block-matching MLPs* can indeed achieve better on images with repeating structure. However, they also sometimes achieve worse results than plain MLPs and do not achieve better results on average.

5.8.1 Differences to previous MLPs

Previously, we trained MLPs to take as input one noisy image patch and to output one denoised image patch. The best results were achieved when the input patch size was 39×39 and the output patch was of size 17×17. Now, we train MLPs to take as input k noisy patches of size 13×13 or 17×17 and to output one noisy patch of the same size. The block matching procedure has to be performed for each training pair, slowing down the training procedure by approximately a factor of 2. One could also imagine MLPs taking as input k patches and providing k patches as output, but we have been less successful with that approach. In all our experiments, we used $k = 14$. The architecture of the MLP we used had four hidden layers; the first hidden layer was of size 4095 and the remaining three were of size 2047. We discuss the training procedure in Chapter 6.

image	BM3D [25]	NLSC [74]	MLP	BM-MLP
Barbara	**30.67**dB	*30.50*dB	29.52dB	29.75dB
Boat	29.86dB	29.86dB	**29.95**dB	*29.92*dB
C.man	29.40dB	29.46dB	*29.60*dB	**29.67**dB
Couple	29.68dB	29.63dB	**29.75**dB	*29.73*dB
F.print	**27.72**dB	27.63dB	*27.67*dB	27.63dB
Hill	29.81dB	29.80dB	*29.84*dB	**29.87**dB
House	*32.92*dB	**33.08**dB	32.52dB	32.75dB
Lena	32.04dB	31.87dB	**32.28**dB	*32.17*dB
Man	29.58dB	29.62dB	*29.85*dB	**29.86**dB
Montage	**32.24**dB	*32.15*dB	31.97dB	32.11dB
Peppers	30.18dB	*30.27*dB	30.27dB	**30.53**dB

Table 5.8: Block-matching MLP compared to plain MLPs and other algorithms for $\sigma = 25$

We note that our block-matching procedure is different from the one employed by BM3D in a number of ways: (i) We always use the same number of neighbors, whereas BM3D chooses all patches whose distance to the reference patch is smaller than a given threshold, up to a maximum of 32 neighbors, (ii) BM3D is a two-step approach, where the denoising result of the first step is merely used to find better neighbors in the second step. We find neighbors directly in the noisy image. (iii) When the noise level is higher than $\sigma = 40$, BM3D employs "coarse pre-filtering" in the first step: patches are first transformed (using a 2D wavelet or DCT transform) and then hard-thresholded. This is already a form of denoising and helps to find better neighbors. We employ no such strategy. (iv) BM3D has a number of hyper-parameters (patch and stride sizes, type of 2D transform, thresholding and matching coefficients). The value of the hyper-parameters are different for the two steps of the procedure. We have fewer hyper-parameters, in part due to the fact that our procedure consists of a single step. We also choose to set the search stride size to the canonical choice of 1.

5.8.2 Block-matching MLPs vs. plain MLPs

Results on 11 standard test images: Table 5.8 summarizes the results achieved by an MLP using block matching with $k = 14$, patches of size 13×13 and $\sigma = 25$. We omit KSVD and EPLL from the comparison because the block-matching MLP and the plain MLP both outperform the two algorithms on every image. The mean result achieved on the 11 images is 0.07dB higher for the block-matching MLP than for the plain MLP. The block-matching MLP outperforms NLSC on 8 images, whereas the plain MLP outperforms NLSC on 7 images. The block-matching MLP outperforms the plain MLP on 7 images. The improvement on the plain MLP is the largest on images Barbara, House and Peppers (approximately 0.25dB on each). The largest decrease in performance compared to the plain MLP is observed on image Lena (a decrease of 0.11dB). We see that the block-matching procedure is most useful on images with repeating structure, as found in the images "Barbara" and "House". However, both BM3D and NLSC achieve results that are far superior to the block-matching MLP on image "Barbara".

Results on larger test sets: The block-matching MLP outperforms the plain MLP on 1480 (59.2%) of the 2500 images, see Figure 5.18. The average improvement over all datasets is 0.01dB. The largest improvement was on the VOC training set (0.03dB). On the McGill dataset, the block-matching MLP was worse by 0.01dB. The block-matching MLP and the plain MLP therefore achieve approximately equal results on average.

On image 198023 in the Berkeley dataset, the MLP with block-matching outperforms the plain MLP by 0.42dB. This is an image similar to the "Barbara" images in that it contains

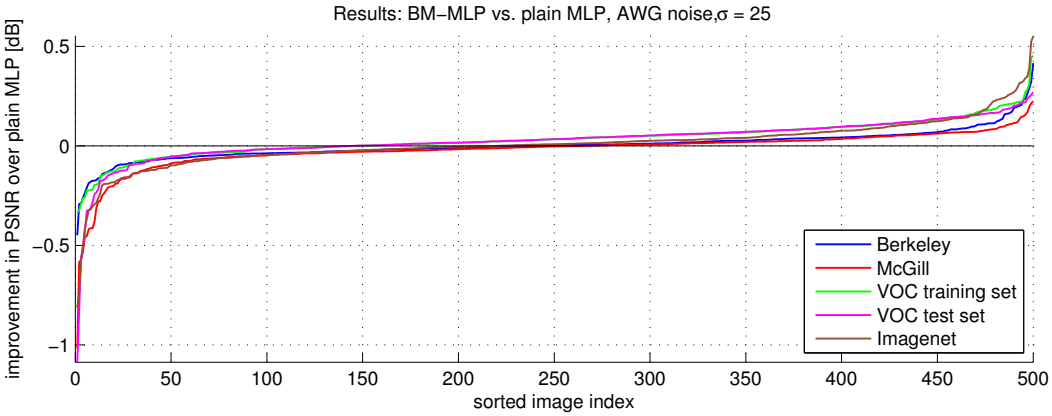


Figure 5.18: Results of the block-matching MLP compared to the plain MLP on five datasets of 500 images

(a) (b)

Figure 5.19: The MLP with block-matching outperforms the plain MLP on this image. (a) Clean image (b) regions where the block-matching MLP is better are highlighted.

a lot of regular structure, see Figure 5.19.

On image 004513 in the VOC test set, see Figure 5.3, the MLP with block-matching performs 1.09dB worse than the plain MLP. This can be explained by the fact that the block-matching MLP uses smaller patches, making it blind to low frequency noise, resulting in a decrease in performance on images with smooth surfaces.

Conclusion: On average, the results achieved with a block-matching MLP are almost equal to those achieved by a plain MLP. Plain MLPs perform better on images with smooth surfaces whereas the block-matching MLPs provide better results on images with repeating structure. However, combining MLPs with the block-matching procedure did not allow us to outperform BM3D and NLSC on image "Barbara". We emphasize that the block-matching MLPs use less information as input than the plain MLPs, yet still achieve results that are comparable on average. Block-matching is a search procedure and therefore cannot be learned by a feed-forward architecture with few layers.

5.9 Combining BM3D and MLPs: Ensembling MLPs

In Section 5.8 we described block-matching MLPs, which are an attempt to enhance the results achieved with MLPs using BM3D's block-matching procedure. In this section, we show that it is possible to combine MLPs with BM3D more directly, using *ensembling* MLPs. It has already been shown [58] that image denoising methods have complementary strengths and weaknesses. In the same paper, the outputs of several denoising algorithms are combined using a method based on regression tree fields (RTFs). We attempt a similar approach, using an MLP and ask the question: Is it possible to achieve results that are better than the best of the inputs? We will see that ensembling MLPs are indeed able to achieve results that are often superior to the best of the inputs.

5.9.1 Ensembling with MLPs

Plain MLPs take as input one noisy patch and provide one denoised patch as output. Ensembling MLPs (E-MLPs) take as input several patches and provide as output one denoised patch. The input patches we use are the following: (i) one patch from the original noisy image (ii) one patch from the BM3D-denoised image, and (iii) one patch from the MLP-denoised image. All patches are taken from corresponding image locations. The idea is that the ensembling MLP should be able to adaptively decide which of the inputs is better. The noisy patch is also provided as input in order to avoid loss of information: Applying a denoising algorithm almost inevitably destroys information contained in the noisy image. Figure 5.20 graphically illustrates the method.

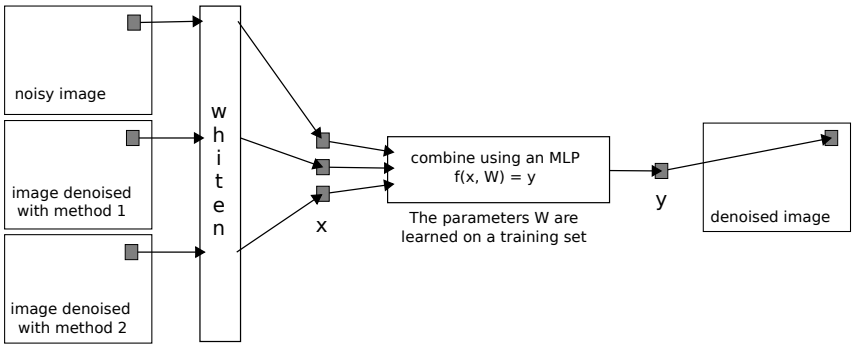


Figure 5.20: Illustration of how we combine denoising results of different algorithms using MLPs. We call this method ensembling MLPs (E-MLPs).

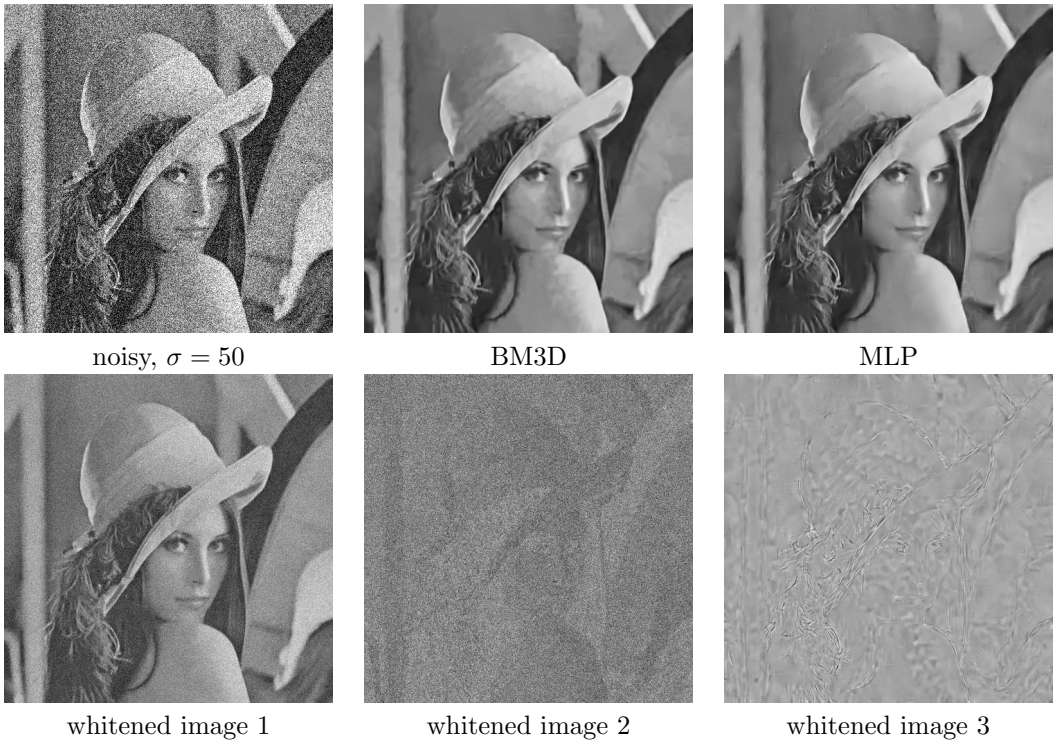


Figure 5.21: Effect of the whitening transform for ensembling MLPs. Top row: The noisy input image, BM3D and MLP. Bottom row: The three images obtained after applying the whitening transform. The third image mostly contains the difference between the BM3D and MLP images.

Training: We train an E-MLP with architecture ($3\times25\times25$, 4×2047, 17×17), (three input patches of size 25×25) on a training set of approximately 8×10^4 images. For each image, we have (i) the noisy image, corrupted with AWG noise, $\sigma = 50$, (ii) the result of BM3D and (iii) the result of an MLP trained on $\sigma = 50$. We chose BM3D and MLP as denoising algorithms because of their good performance and relatively short running times (on GPU, for the MLP) and because of their complementary strengths and weaknesses: BM3D is good on regular structures, the MLP on more complex structures. However, one could also imagine using a different set of denoising algorithms. The three patches extracted from the three images are then *whitened*, using an empirically found 3×3 whitening matrix. This way, the three input patches are approximately uncorrelated. We were unable to achieve good results without this whitening transform, presumably because the patches from the BM3D and the MLP images are too similar, which might make optimization difficult [64]. The effect of the whitening transform is illustrated in Figure 5.21. Training otherwise proceeds exactly like for the plain MLPs and the BM-MLPs.

5.9.2 Results with ensembling MLPs

Results on 11 standard test images: Table 5.9 lists the results obtained on 11 standard test images with an E-MLP and with other algorithms. The E-MLP outperforms all other methods on all images except image "Barbara". We see that the output of the E-MLP is better than the best input on all images except image "Barbara". We can already answer our question in the positive: E-MLPs are able to achieve results than are better than the best of its inputs. While BM3D is still the best method on image "Barbara", the E-MLP achieves an improvement of approximately 1.6dB over the plain MLP. The improvements

image	KSVD [1]	EPLL [132]	BM3D [25]	NLSC [74]	MLP	E-MLP: MLP and BM3D
Barbara	25.22dB	24.83dB	**27.21**dB	*27.13*dB	25.37dB	26.95dB
Boat	25.90dB	26.59dB	26.72dB	26.73dB	*27.02*dB	**27.11**dB
C.man	25.42dB	26.05dB	26.11dB	26.36dB	*26.42*dB	**26.75**dB
Couple	25.40dB	26.24dB	26.43dB	26.33dB	*26.71*dB	**26.78**dB
F.print	23.24dB	23.59dB	*24.53*dB	24.25dB	24.23dB	**24.57**dB
Hill	26.14dB	26.90dB	27.14dB	27.05dB	*27.32*dB	**27.40**dB
House	27.44dB	28.77dB	29.71dB	*29.88*dB	29.52dB	**30.00**dB
Lena	27.43dB	28.39dB	28.99dB	28.88dB	*29.34*dB	**29.46**dB
Man	25.83dB	26.68dB	26.76dB	26.71dB	*27.08*dB	**27.13**dB
Montage	26.42dB	27.13dB	27.69dB	28.02dB	*28.07*dB	**28.34**dB
Peppers	25.91dB	26.64dB	26.69dB	26.73dB	*26.74*dB	**27.18**dB

Table 5.9: Ensembling BM3D and MLP with an MLP, $\sigma = 50$. The results are usually better than the best of the two inputs.

MLP, PSNR: 25.37dB | BM3D: **27.21**dB | E-MLP: 26.95dB

MLP, PSNR: 29.34dB | BM3D: 28.99dB | E-MLP: **29.46**dB

Figure 5.22: Results obtained with an ensembling MLP (E-MLP) on images Barbara and Lena, corrupted with AWG noise, $\sigma = 50$. On image Barbara, BM3D produces better results on regular textures, *e.g.* the pants and the scarf, whereas the MLP produces better results in smooth areas, *e.g.* the floor. The E-MLP combines the strengths of both methods: The regular textures look better than when using the MLP and the floor looks better than when using BM3D. On image Lena, the result obtained with the E-MLP is visually closer to the result obtained with the MLP, but is superior in terms of PSNR. Hence, the result obtained with BM3D presumably contains useful complementary information.

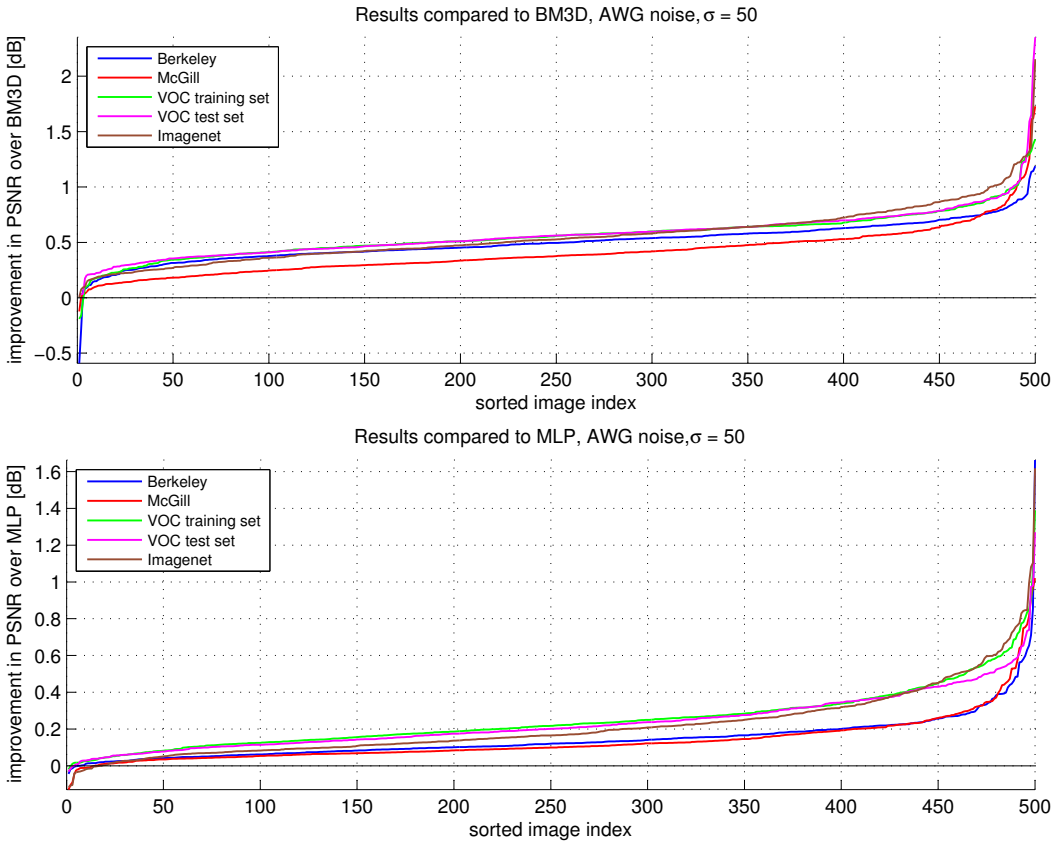


Figure 5.23: E-MLP vs. BM3D (top) and E-MLP vs. plain MLP (bottom) on five datasets of 500 images and $\sigma = 50$.

method name:	FoE [105]	BM3D [25]	EPLL [132]	NLSC [74]	$\text{RTF}_{\text{Plain}}$ [58]	MLP
PSNR:	24.47dB	25.09dB	25.18dB	25.09dB	24.76dB	**25.45**dB

method name:	RTF_{BM3D} [58]	RTF_{All} [58]	E-MLP (combines MLP and BM3D)
PSNR:	25.38dB	25.51dB	**25.58**dB

Table 5.10: Results obtained with an ensembling MLP (E-MLP) and other methods on the dataset of images used in [58], with $\sigma = 50$. Top: Stand-alone methods, bottom: methods combining the results of other methods. The MLP is the best stand-alone method on this dataset and this noise level. The E-MLP outperforms all RTF-based methods on this dataset and this noise level.

obtained using the ensembling MLP can be visually appreciated in Figure 5.22.

Results on larger test sets: Figure 5.23 compares the results achieved with the E-MLP to BM3D and the plain MLP on the 2500 images. The E-MLP achieves an average improvement of 0.52dB over BM3D. The smallest average improvement is the McGill dataset, with 0.41dB, while the largest improvement is on the VOC test set, with 0.57dB improvement. There are only 6 images out of 2500 (0.24%) where the ensembling MLP is worse than BM3D.

The ensembling MLP achieves an average improvement of 0.2dB over the plain MLP. The smallest improvement is on the McGill dataset, with 0.13dB improvement, the largest improvement on the VOC training set, with 0.25dB improvement. Of the 2500 images, there are 40 images (1.6%) where the results have become worse than the plain MLP (however by only 0.12dB in the worst case).

Comparison against the RTF-based method [58]: Jancsary *et al.* [58] propose a denoising method based on regression tree fields (RTFs). The RTFs produce a Gaussian conditional random field (GCRF) on which it is possible to perform inference in order to obtain an image-dependent prediction. The method can be used either by itself ($\text{RTF}_{\text{Plain}}$) or in combination with other denoising methods. The method combined with BM3D is denoted as RTF_{BM3D} and the method combined with BM3D, FoE [105], EPLL [132] and NLSC [74] is denoted as RTF_{All}. Results for all three methods are reported on a subset of the Berkeley dataset, with images down-scaled by a factor of 2 (*i.e.* the images are quite small: approximately 160×240 pixels). Table 5.10 compares our method to the RTF-based methods and others on the same dataset, for $\sigma = 50$. We see that our E-MLP achieves the best results of all methods on this dataset. We note that it might be possible to improve the results of the E-MLP further by including the results of NLSC, FoE and EPLL.

Conclusion: We can conclude than ensembling MLPs are an effective way to combine the strengths of complementary denoising algorithms. In our example using BM3D and a plain MLP as input methods, the results of ensembling are almost always better than the best of the input methods. We also see that this ensembling approach achieves better results than the BM-MLP of Section 5.8. Ensembling MLPs also bring us still closer to the denoising bounds estimated by Levin *et al.* [69]. Levin *et al.* [69] estimate the bounds to lie 0.7dB above BM3D for $\sigma = 50$. With plain MLPs, we achieved an improvement of 0.32dB over BM3D. The ensembling MLP achieves an improvement of 0.52dB over BM3D. This leaves an estimated room for an improvement of only 0.18dB.

5.10 Code

We make available a MATLAB toolbox allowing to denoise images with our trained MLPs on CPU at `http://people.tuebingen.mpg.de/burger/neural_denoising/`. The script

`demo.m` loads the image "Lena", adds AWG noise with $\sigma = 25$ on the image and denoises with an MLP trained on the same noise level. Running the script produces an output similar to the following

```
>> demo
Starting to denoise...
Done! Loading the weights and denoising took 121.4 seconds
PSNRs: noisy: 20.16dB, denoised: 32.26dB
```

and display the clean, noisy and denoised images. Denoising an image is performed using the function `fdenoiseNeural`:

```
>> im_denoised = fdenoiseNeural(im_noisy, noise_level, model);
```

The function takes as input a noisy image, the level of noise and a struct containing the step size and the width of the Gaussian window applied on denoised patches.

```
>> model = {};
>> model.step = 3;
>> model.weightsSig = 2;
```

5.11 Discussion and Conclusion

In this chapter, we have described a learning-based approach to image denoising. We have compared the results achieved by our approach against other algorithms and against denoising bounds, allowing us to draw a number of conclusions.

Comparison against state-of-the-art algorithms:

- **KSVD:** We compared our method against KSVD [31] on 11 test images and for all noise levels. KSVD outperforms our method only on image Barbara with $\sigma = 10$.
- **EPLL:** We outperform EPLL [132] on more than 99% of the 2500 test images on $\sigma = 25$, and by 0.35dB on average. For all other noise levels and 11 test images, we always outperform EPLL.
- **NLSC:** We outperform NLSC [74] more approximately 80% of the 2500 test images on $\sigma = 25$, and by 0.16dB on average. The higher the noise level, the more favorably we perform against NLSC. NLSC has an advantage over our method on images with repeating structure, such as Barbara and House. However, at high noise levels, this advantage disappears.
- **BM3D:** We outperform BM3D [25] on approximately 92% of the 2500 test images on $\sigma = 25$, and by 0.29dB on average. Otherwise, the same conclusions as for NLSC hold: The higher the noise level, the more favorably we perform against BM3D. BM3D has an advantage over our method on images with repeating structure, such as Barbara and House. However, at high noise levels, this advantage disappears.

Our method compares the least favorably compared to other methods on the lowest noise level ($\sigma = 10$), but we still achieve an improvement of 0.1dB over BM3D on that noise level.

Comparison against denoising bounds:

- **Clustering-based bounds** Our results exceed the bounds estimated by Chatterjee and Milanfar [22]. This is possible because we violate the "patch cluster" assumption made by the authors. We conclude that the patch cluster assumption is not a reasonable assumption to make in order to estimate denoising bounds. In addition,

Chatterjee and Milanfar [22] suggest that there is almost no room for improvement over BM3D on images with complex textures. We have seen that is not the case: Our approach is often significantly better than BM3D on images with complex textures.

- **Bayesian patch-based bounds** Levin and Nadler [68] estimate denoising bounds in a Bayesian setting, for a given patch size. Our results are superior to these bounds. This is possible because we use larger patches than is assumed by Levin and Nadler [68]. The same authors also suggest that image priors should be the most useful for denoising at medium noise levels, but not so much at high noise levels. Yet, our method achieves the greatest improvements over other methods at high noise levels.

  Similar bounds estimated for patches of infinite size are estimated by Levin *et al.* [69]. We make important progress toward reaching these bounds: Our approach reaches almost half the theoretically possible gain over BM3D. Levin *et al.* [69] agree with Chatterjee and Milanfar [22] that there is little room for improvement on patches with complex textures. We have seen that this is not the case.

  We have seen that the reason why MLPs do not perform as well as BM3D on images with regular textures is supported by estimated Bayesian bounds with finite patch size. The amount of training data (*i.e.* the number of clean patches) resembling such regularly textured patches is too small in order to denoise highly regular textures as well as BM3D.

Comparison on other noise types: We have seen that our method can be adapted to other types of noise by merely switching the training data. We have shown that we achieve good results are on stripe noise, salt-and-pepper noise, JPEG quantization artifacts and mixed Poisson-Gaussian noise. In the latter two cases we seem to be competitive with the state-of-the-art.

Block-matching MLPs: We have also seen that results can sometimes be improved a little further using a block-matching procedure. However, this comes at the cost of a more complicated training procedure and longer training and test times. In addition, the block-matching procedure is highly *task-specific*: It has been shown to work well on AWG noise, but it is not clear that it is useful for all kinds of noise. In addition, plain MLPs could potentially be used for other low-level vision tasks. It is not clear that the block-matching procedure is useful for other tasks. We here face an often encountered conundrum: Is it worth exploiting task-specific knowledge? This often leads to better results, at the cost of more engineering.

Ensembling MLPs: On some images, our method outperforms BM3D by more than 1.5dB and NLSC by more than 3dB, see Section 5.5. Our method therefore seems to have a clear advantage over other methods on some images. However, we have seen that our approach sometimes achieves results that are much worse than the previous state-of-the-art. This happens especially on images with a lot of regular structure, such as the image "Barbara". Our attempt to ameliorate the situation using a block-matching procedure was only partially successful. However, we showed that combining BM3D with MLPs via ensembling MLPs achieves much better results. The results achieved with this method are significantly better than those achieved with plain MLPs. In addition, there are almost no images where this method performs worse than BM3D. The average results achieved using this method for $\sigma = 50$ are close to the bounds estimated by Levin *et al.* [69]: A room for improvement over BM3D of 0.7dB is estimated, and an improvement of 0.52dB is achieved using this method. Training an ensembling MLP requires somewhat more engineering than training a plain MLP, but less so than training a block-matching MLP. The ensembling MLP can be regarded as a learning-based approach: Each denoising algorithm used as input can be regarded as a feature extractor.

Computation time: Denoising an image using an MLP takes approximately a minute on CPU and less than 5 seconds on GPU. This is not as fast as BM3D, but much faster than approaches that require learning a dictionary, such as KSVD or NLSC which can take almost an hour per image (on CPU).

Training procedure: Chapter 6 describes our training procedure in detail and shows the importance of various factors influencing the quality of the results, such as the size of the training corpus, the architecture of the multi-layer perceptrons and the size of the input and output patches. We show that some setups lead to surprisingly bad results and provide an explanation for the phenomena.

Understanding denoising: Also not discussed in this chapter is the operating principle of the multi-layer perceptrons: How do they achieve denoising? Trained neural networks are often seen as "black boxes", but we will see in Chapter 6 that in this case, the behavior can be understood, at least to some extent.

Multi-scale extension: We showed in Chapter 3 that many denoising algorithms do not handle low frequencies well and can be improved using a multi-scale procedure in which the denoising algorithm is applied at various scales and the resulting images subsequently recombined using a procedure resembling a Laplacian pyramid. This meta-procedure is especially helpful at high noise levels. Can our method also be improved using this multi-scale meta-procedure? We answer this question in the negative: We were not able to improve the results using a multi-scale procedure, even at the highest noise level ($\sigma = 170$). This result indicates that the input patches of our MLPs are large enough to adequately handle low frequencies.

6

Training and understanding multi-layer perceptrons for image denoising

Chapter abstract In Chapter 5, we described image denoising as the problem of mapping from a noisy image to a noise-free image and showed that multi-layer perceptrons can model such a mapping effectively. Trained multi-layer perceptrons can achieve outstanding image denoising performance for various types of noise (additive white Gaussian noise, mixed Poisson-Gaussian noise, JPEG artifacts, salt-and-pepper noise and noise resembling stripes). In this chapter, we discuss in detail which trade-offs have to be considered during the training procedure. We will show how to achieve good results and which pitfalls to avoid. By analysing the activation patterns of the hidden units we are able to make observations regarding the functioning principle of multi-layer perceptrons trained for image denoising.

The material of this chapter is based on the following publications:

[13] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising with multi-layer perceptrons, part 2: training trade-offs and analysis of hidden activation patterns. **Submitted** to a journal, **available** at `http://arxiv.org/abs/1211.1552`

6.1 Introduction

In Chapter 5, we show that multi-layer perceptrons (MLPs) mapping a noisy image patch to a denoised image patch are able to achieve outstanding image denoising results, even surpassing the previous state-of-the-art [25]. In addition, the MLPs outperform one type of theoretical bound in image denoising [22] and come a long way toward closing the gap to a second type of theoretical bound [69]. Related work in image denoising is also discussed in Chapter 5. This chapter explains the technical trade-offs to achieve those results.

Achieving good results with MLPs was possible through the use of larger patch sizes: It is known that larger patch sizes help make the denoising problem less ambiguous [68]. However, large patches also make the denoising problem more difficult (the function is higher dimensional). This required us to train high-capacity MLPs on a large number of training samples. Training such MLPs is therefore time-consuming, though modern GPUs alleviate the problem somewhat.

Training neural networks, especially large ones, is usually performed using stochastic gradient descent and is sometimes considered more of an art than a science. While there exist "tricks" to make training efficient [64, 7], it is still quite possible that some experimental setups will lead to poor results. In these cases, it is often poorly understood why the results are bad. One might sometimes attribute these bad results to "bad luck" such as an unlucky weight initialization. This becomes a problem especially for time-consuming large-scale experiments, where multiple restarts are simply not possible. It is therefore crucial to understand which setups are likely to lead to good results and which to bad results before launching an experiment.

A common criticism regarding neural networks is that they are "black boxes": Given a neural network, one can merely observe its output for a given input. The inner workings or logic are usually not open for inspection. Under certain circumstances, this is not the case: Convolutional neural networks [63] are usually easier to interpret for humans because the hidden representations can be represented as images [65]. More recently, Erhan *et al.* [33] have proposed an *activation maximization* procedure to find an input maximizing the activation of a hidden unit, and have shown that this procedure allows for better qualitative evaluation of a network.

Contributions: This chapter aims to address the above two issues for MLPs trained to denoise image patches. In the first part of this chapter, we provide a detailed description of a large and varied set of large-scale experiments. We will discuss various trade-offs encountered during the training procedure. Certain settings of training parameters can lead to initially good results, but later lead to a *catastrophic* degradation in performance. This phenomenon is highly undesirable and we will provide guidelines on how to avoid it, as well as an explanation of such phenomena.

In the second part of this chapter, we show that surprisingly, it is possible to gain insight into the operating principle or inner workings of an MLP trained on image denoising. This is the least difficult for MLPs with a single hidden layer, but we will show that MLPs with more hidden layers are also interpretable through analysis of the activation patterns of the hidden units. We also gain insight about denoising auto-encoders [120] due to their similarity to our MLPs.

Notation and definitions: For an MLP with four hidden layers, each containing 2047 hidden units, input patches of size 39×39 pixels and output patches of size 17×17 pixels, we use the following notation $(39 \times 39, 2047, 2047, 2047, 2047, 17 \times 17) \equiv (39, 4 \times 2047, 17)$. If the input and output patches are of the same size, we use the following notation $(17, 4 \times 2047)$ to denote an MLP with four hidden layers of size 2047 and input and output patches of size 17×17 pixels.

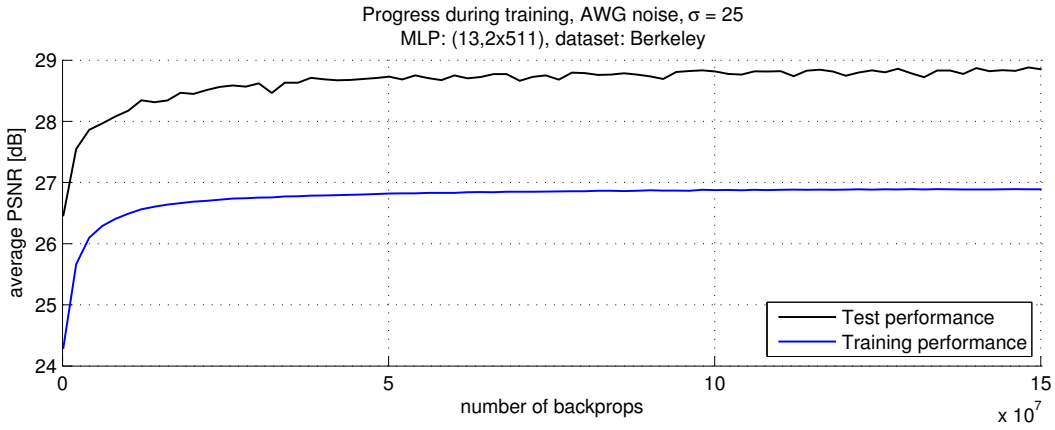


Figure 6.1: No overfitting even after many updates due to an abundance of training data.

We will periodically halt the training procedure of an MLP and report the *test performance*, by which we mean the average PSNR achieved on the 11 standard test images defined in Chapter 5. When we report the *training performance*, we mean the average PSNR achieved on the last 2×10^6 training samples. The test performance therefore refers to image denoising performance, whereas the training performance refers to patch denoising performance.

6.2 Training trade-offs to achieve good results with MLPs

In Chapter 5 we showed that it is possible to achieve state-of-the-art image denoising results with MLPs. This section will show what steps are necessary to achieve these results. We do so by tracking the evolution of the results for different experimental setups during the training process. In particular, we will vary the size of the training dataset as well as the architecture of the MLPs. We will mostly use AWG noise with $\sigma = 25$. Each experiment is the result of many days and sometimes even weeks of computation time on a modern GPU (we used nVidia's C2050).

6.2.1 Long training times do not result in overfitting

In this section, we will use a much smaller training set as the one defined in Chapter 5. We will use the 200 training images from the BSDS300 dataset, which is a subset of the BSDS500 dataset.

We train an MLP with architecture $(13, 2 \times 511)$. We report both the training performance and the test performance. The reason why the test performance is superior to the training performance is that the test performance refers to the image denoising performance (as opposed to the patch denoising performance). The image denoising performance is better than the patch denoising performance because of the averageing procedure in areas where patches overlap. We observe that the training and test performance improve steadily during the first few million updates. Results still improve after 10^8 updates, albeit more slowly. On the test set, results occasionally briefly become worse. We also see that there is no overfitting even though we are using a rather small training set. This is due to the abundance of training data (the probability that a noisy patch is seen twice is zero). These results suggest that overfitting is not an issue.

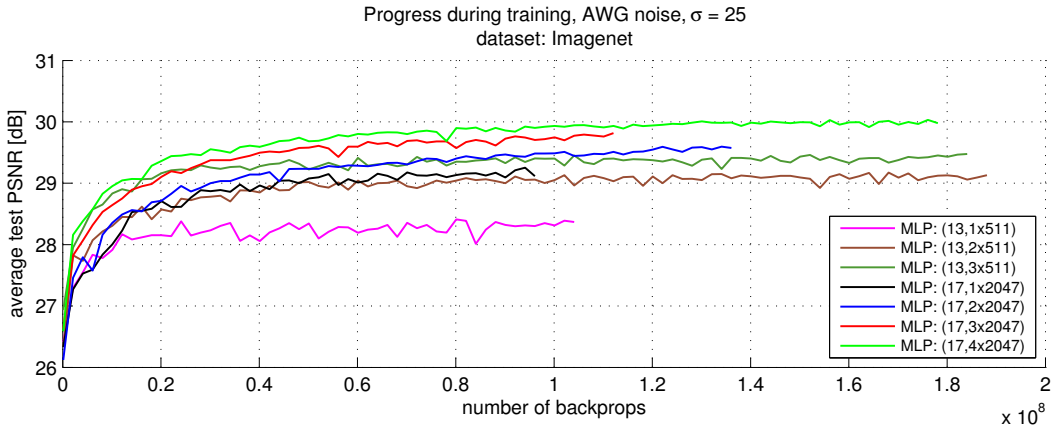


Figure 6.2: More hidden layers help. Three small hidden layers outperform one large hidden layer.

Figure 6.3: More hidden layers usually help. Too many hidden layers with few hidden units cause a catastrophic degradation in performance.

6.2.2 Larger architectures are usually better

We now use the full training set—as defined in Chapter 5—and train various MLPs. The size of the patches was either 13×13 or 17×17. When the patch size was 13×13, we used hidden layers with 511 units. When the patch size was 17, we used hidden layers with 2047 units. We varied the number of hidden layers, see Figure 6.2.

Adding hidden layers seems to always help. Larger patch sizes and wider hidden layers seem to be beneficial. However, the MLP using patches of size 13×13 and three hidden layers of size 511 outperforms the MLP using patches of size 17×17 and a single hidden layer of size 2047.

Is it always beneficial to add hidden layers? To answer this question, we train MLPs with patches of size 13×13 and hidden layers of size 511 with four and five hidden layers, see Figure 6.3. The MLPs with four and five hidden layers perform well during the beginning of the training procedure, but experience a significant decrease in performance later on. The MLP achieving the best performance overall has three hidden layers. We therefore conclude

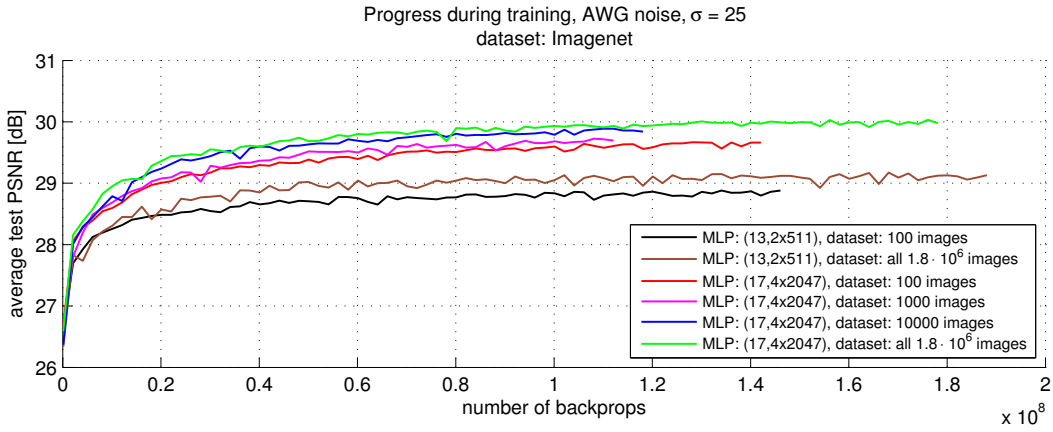


Figure 6.4: Using a larger corpus of training data helps.

that it is not always beneficial to add hidden layers.

A possible explanation for the degradation of performance shown in Figure 6.3 is that MLPs with more hidden layers become more difficult to learn. Indeed, each hidden layer adds non-linearities to the model. It is therefore possible that the error landscape is complex and that stochastic gradient descent gets stuck in a poor local optimum from which it is difficult to escape. In Figure 6.2, we see that an MLP with patches of size 17×17 and four hidden layers of size 2047 does not experience the effect shown in Figure 6.3, which is an indication that deep and narrow networks are more difficult to optimize than deep and wide networks.

6.2.3 A larger training corpus is always better

We have seen that longer training times lead to better results. Therefore, seeing more training samples helps the MLPs achieve good results.

We now ask the question: What is the effect of the number of images in the training corpus? To this end, we have trained MLPs with identical architectures on training sets of different sizes, see Figure 6.4. We used either the full ImageNet training set or various subsets (100, 1000 and 10000 images) of the same training set. We see that significant gains can be obtained from using more training images. In particular, using even 10000 training images delivers results that are clearly worse than results obtained when training on the full ($\sim 1.8 \cdot 10^6$ image) training set. We also never observe a degradation in performance by using more training images.

6.2.4 The trade-off between small and large patches

We ask the question: Is it better to use small or large patches? We first restrict ourselves to situations where the input and output patches are of the same size.

Figure 6.5 shows the results obtained with MLPs with four hidden layers of size 2047 and various patch sizes. We see that up to a patch size of 17×17, an increase in patch size leads to better results. This is in agreement with the findings of Levin and Nadler [68]: Using a larger support size makes the denoising problem less ambiguous.

However, increasing the patch size further leads to worse results. The results obtained using patches of size 21×21 are worse than those obtained using patches of size 17×17. Using patches of size 25×25 leads to results that are still worse and even leads to a degradation in performance after approximately 10^8 updates. For patches of size 29×29 we observe still worse results and a deterioration of results after approximately $5 \cdot 10^7$ updates.

Figure 6.5: Larger patches lead to better results, up to a point.

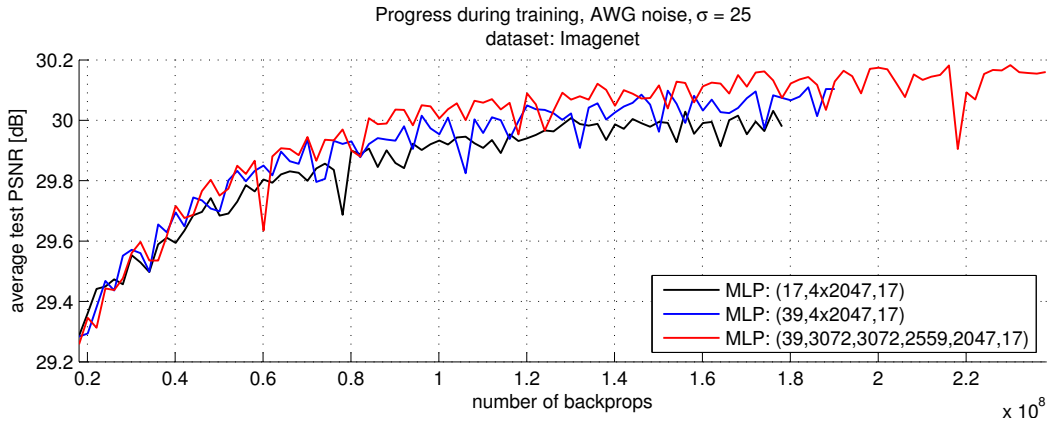


Figure 6.6: Larger input patches help.

The performance later recovers slightly, but never reaches the levels achieved before the degradation in performance. For this observation, we provide an explanation similar to the one provided in section 6.2.2: Larger patch sizes increase the dimensionality of the problem and therefore also the difficulty. The model is therefore more difficult to optimize when large patches are used, and stochastic gradient descent may fail.

Therefore, when the input and output patches are of the same size, an ideal patch size exists (for our architectures, it seems to be approximately 17×17). Patches that are too small result in a denoising *function* that does not deliver good results, whereas patches that are too large results in a *model* that is difficult to optimize.

Larger input than output patches: What happens when we remove the restriction that the input patches be of the same size as the output patches? We expect bad results when the output patches are larger than the input patches: This would require hallucinating part of the patch. A more interesting question is: What happens when the output patches are smaller than the input patches?

Figure 6.6 shows that using input patches that are larger than the output patches delivers slightly better results. Using an architecture with even more hidden units leads to even slightly better results.

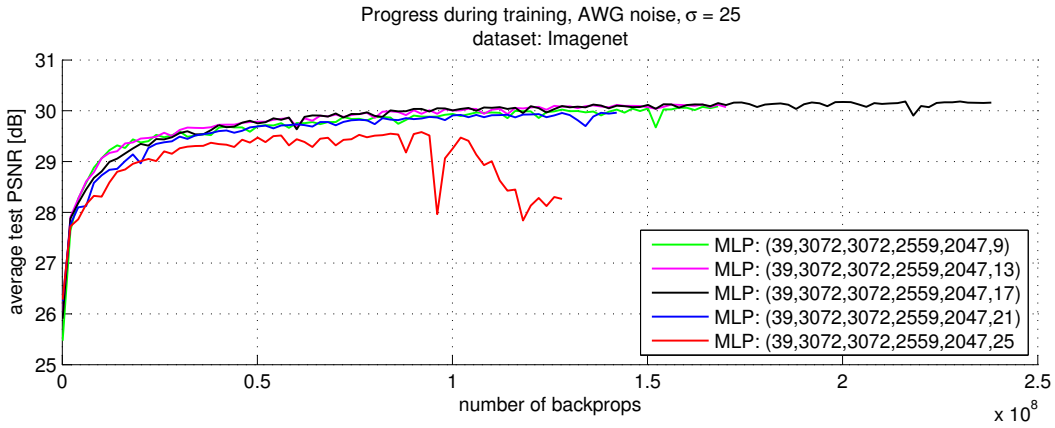


Figure 6.7: For a given input patch size, there exists an ideal output patch size. Output patches that are too large can create problems.

Figure 6.8: Too many hidden layers combined with large output patches creates problems.

We now keep the size of the input patches fixed at 39×39 pixels and vary the size of the output patches, see Figure 6.7. We observe that increasing the size of the output patches helps only up to a point, after which we observe a degradation in performance. The ideal output patch size seems to be the same as when the input and output patches are of the same size (17×17). Our explanation is again that output patches that are too large result in a model that is difficult to optimize.

Finally, we investigate if the patch size has an effect on the best choice of architecture. Figure 6.8 shows the results obtained with different patch sizes and architectures. We see again that with hidden layers of size 511, using more than three hidden layers creates a degradation of performance when combined with patches of size 13×13. With hidden layers of size 1023, four hidden layers combined with input patches of size 39×39 and output patches of size 17×17, no degradation in performance is observed. Using the same patch sizes with six hidden layers of size 1023 quickly results in a degradation in performance. However, using the same architecture, but using output patches of size 9×9 results in no degradation in performance and even yields the best results in this comparison. We therefore conclude that it is the combination of deep and narrow networks combined with large output patches that are the most difficult to optimize.

Conclusions concerning MLP architectures: We have learned that hidden layers with more units are always beneficial. Similarly, larger input patches are also always helpful. However, too many hidden layers may lead to problems in the training procedure. Problems are more likely to occur if the hidden layers contain few hidden units or if the size of the output patches is large.

6.2.5 Important gains in performance through "fine-tuning"

In all previous experiments, we observed that the test error fluctuates slightly. We attempt to avoid or at least reduce this behavior using a "fine-tuning" procedure: We initially train with a large learning rate and later switch to a lower learning rate. The large learning rate is supposed to encourage faster learning, whereas the low learning rate is supposed to encourage more stable results on the test data. Figure 6.9 shows that we can indeed reduce

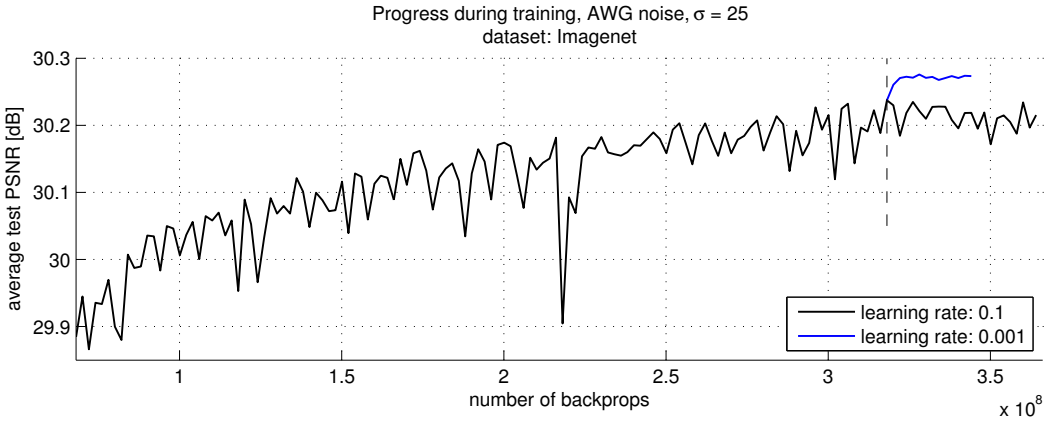


Figure 6.9: Fine-tuning improves results and reduces fluctuations in the test error. The vertical dashed line indicates where the learning rate was switched from 0.1 to 0.001. The two curves therefore only disagree starting at the dashed vertical line.

fluctuations in the test error using a fine-tuning procedure. In addition, the switch to a lower learning rate leads to an improvement of approximately 0.05dB on the test set. We conclude that it is important to use a fine-turning procedure to obtain good results.

6.2.6 Other noise variances: smaller patches for lower noise

Figure 6.10 shows the improvement of the test results (the average result obtained on the 11 standard test images) during training for different values of σ. The test results achieved by the MLPs is compared against the test results achieved by BM3D. We used input patches of size 39×39, output patches of size 17×17 and hidden layers of sizes 3072, 3072, 2559 and 2047. We also experimented with smaller patches ("smaller patches" in Figure 6.10): Input patches of size 21×21 and output patches of size 9×9. In that case, we also used a somewhat smaller architecture: Four hidden layers of size 2047.

Most MLPs never reach the test results achieved by BM3D because of the relatively bad performance on image "Barbara". For $\sigma = 50$, we approach the results achieved by BM3D faster than for $\sigma = 25$ and for $\sigma = 25$, we approach the results achieved by BM3D faster than for $\sigma = 10$. For $\sigma = 75$, we approach the results achieved by BM3D the fastest and even slightly outperform the results. We see that the gap between our results and those of BM3D becomes smaller when the noise is stronger. The slower convergence for lower noise levels can be explained by the fact that the overall error is lower (or equivalently: the PSNR values are higher), which causes the updates during the training procedure to be smaller.

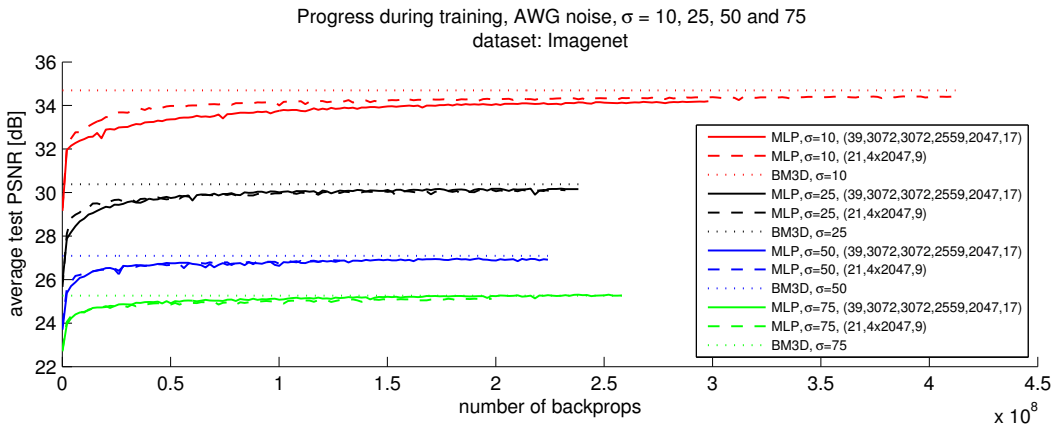


Figure 6.10: Progress on different noise levels compared to BM3D. The higher the noise level, the faster the progress.

For $\sigma = 10$, better results are achieved with smaller patches. For $\sigma = 25$, $\sigma = 50$ and $\sigma = 75$, better results are achieved with larger patches. The reason larger patches achieve better results for $\sigma = 25$, $\sigma = 50$ and $\sigma = 75$ is that larger patches are necessary when the noise becomes stronger [68]. This implies that it is not necessary to use large patches when the noise is weaker. Indeed, using patches that are too large can cause the optimization to become difficult, see section 6.2.4. Therefore, the ideal patch sizes are influenced by the strength of the noise. We used 21×21 and 9×9 patches for $\sigma = 10$ and 39×39 and 17×17 patches for the other noise levels.

6.3 Training trade-offs for block-matching MLPs

We have seen in Chapter 5 that MLPs can be combined with a block-matching procedure and that doing so can lead to improved results on some images. In this section, we discuss the training procedure of block-matching MLPs in more detail. We write $(39, 14\times13, 4\times2047, 13)$ to denote a block-matching MLP with a search window of size 39×39 pixels, taking as input 14 patches of size 13×13 pixels, four hidden layers with 2047 hidden units each, and an output patch size of 13×13 pixels.

6.3.1 Block-matching MLPs can learn faster

We see in Figure 6.11 that progress during training with the block-matching MLPs is similar to progress with the best MLPs that do not use block-matching. We see an improvement over the plain MLPs particularly at the beginning of the training procedure. Later on, the advantage of the block-matching procedure over plain MLPs is less evident. The block-matching procedure using patches of size 13×13 and a search window of size 39×39 performs slightly better than the block matching procedure using patches of size 17×17 and a search window of size 59×59. The search window size of 39×39 is the same as the size of the patches the best-performing plain MLP takes as input. This means that the block-matching MLP achieving the better results always uses less information as input than the plain MLP achieving the best results, yet still achieves similar results.

Figure 6.12 compares the progress of the winning plain MLP to the block-matching MLP using patches of size 13×13 on image "Barbara" against the remaining 10 of the 11 standard test images. We see that on image "Barbara", the block-matching MLP has a clear advantage, particularly at the beginning of the training procedure. On the remaining images,

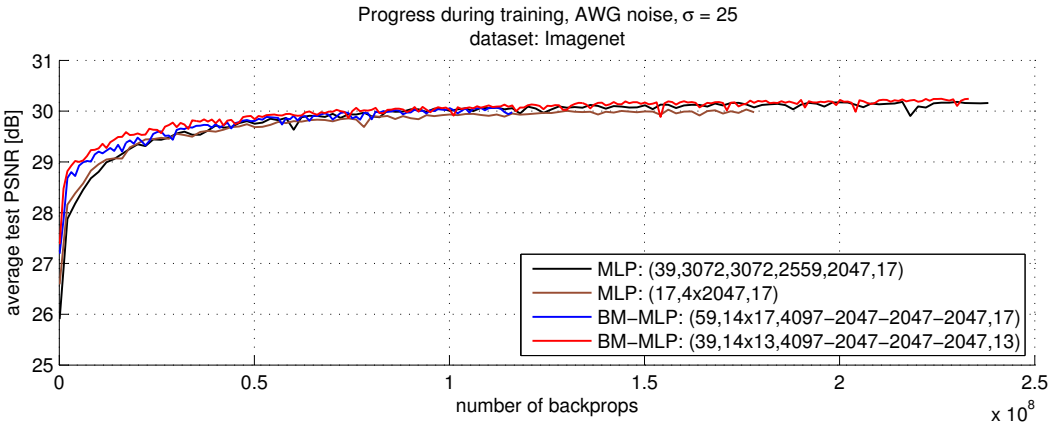


Figure 6.11: Block matching helps at the beginning of the training procedure.

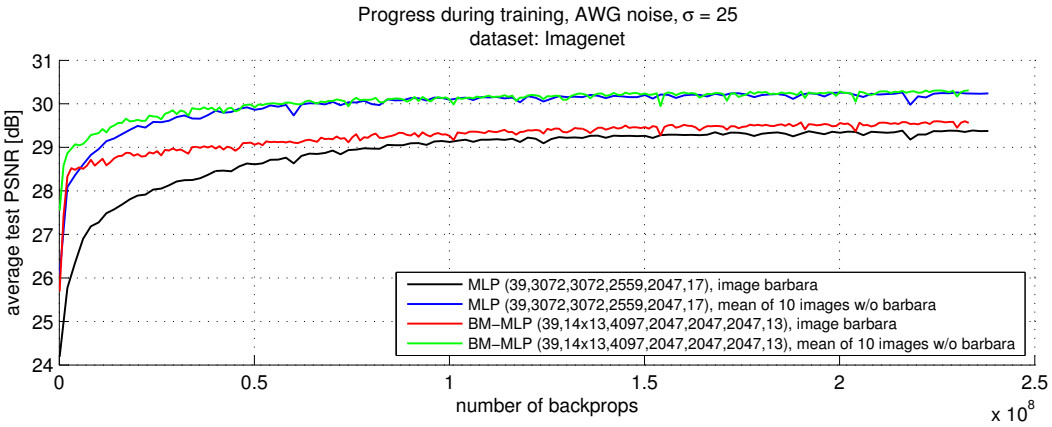


Figure 6.12: Block matching helps particularly for image Barbara.

the advantage is less clear. Still, the results at the beginning of the training procedure are better for the block-matching MLP.

This answers our question: The block-matching procedure helps on images with regular structure. However, the improvement is rather small at the end of the training procedure.

6.3.2 Are block-matching MLPs useful on all noise levels?

We train MLPs in combination with the block matching procedure on noise levels $\sigma = 10$, $\sigma = 50$ and $\sigma = 75$. We again use $k = 14$ and patches of size 13×13.

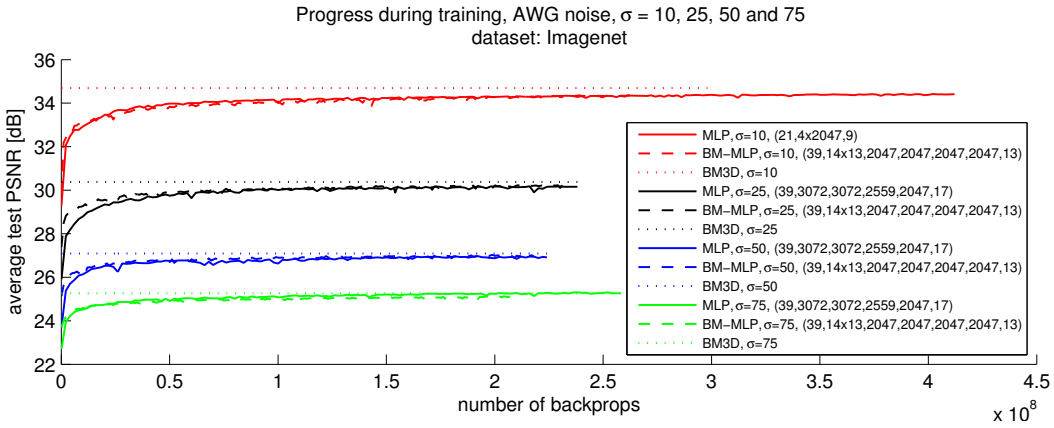


Figure 6.13: Progress on different noise levels compared to BM3D. Block-matching is the most useful at $\sigma = 25$ and $\sigma = 50$.

Figure 6.13 shows the progress during training for the different noise levels. For $\sigma = 10$, the block-matching procedure seems to present no advantage over the best MLP without block-matching procedure. For $\sigma = 25$ and $\sigma = 50$, the block-matching procedure provides better results at the beginning of the training procedure. In the later stages of the training procedure, it is not clear if the block-matching procedure achieves superior results. For $\sigma = 75$, the block-matching procedure presents no clear advantage at the beginning of the training procedure and also achieves worse results than the plain MLP in the later stages of the training procedure. A possible explanation for the deterioration of the results achieved with block-matching compared to plain MLPs at increasing noise levels is that it becomes more difficult to find patches similar to the reference patch. A possible solution would be to employ a coarse pre-filtering step such as the one employed by BM3D.

6.4 Analysis of hidden activation patterns

We have seen in Chapter 5 that our method can achieve good results on medium to high noise levels. We have also shown which steps are important and which are to be avoided in order to achieve good results. We now ask the question: Can we gain insight into how the MLP works? An MLP is a highly non-linear function with millions of parameters. It is therefore unlikely that we will be able to perfectly describe its behavior. This section describes a set of experiments that will nonetheless provide some insight about how the MLP works.

Definitions: Weights connecting the input to one unit in the first hidden layer can be represented as a patch. We refer to these weights as *feature detectors* because they can be

interpreted as filters. The weights connecting one unit in the last hidden layer of an MLP to the output can also be represented as a patch and we will refer to these as *feature generators*.

When feeding data into an MLP, we are interested not only in the weights, but also in the *activations*, by which we mean the values taken by the hidden units, due to the input. We will attempt to find inputs maximizing the activation of a specific hidden unit and refer to such an input as an *input pattern*. Conversely, we refer to the output caused by the activation of a single hidden unit as an *output pattern*.

The input pattern maximizing the activation of a hidden unit in the first hidden layer is the same as the feature detector corresponding to the hidden unit. Also, the output pattern corresponding to a hidden unit in the last hidden layer is the same as the feature generator associated to the same hidden unit.

6.4.1 MLPs with a single hidden layer

We start by analyzing an MLP with a single hidden layer. We use an MLP with the architecture $(17\times17, 2047, 17\times17)$ for that purpose. Such an MLP is identical to a denoising auto-encoder with AWG noise [120].

Weights as patches: The feature detectors of this MLP can be represented as patches of size 17×17 pixels. The feature generators have the same size of the feature detectors.

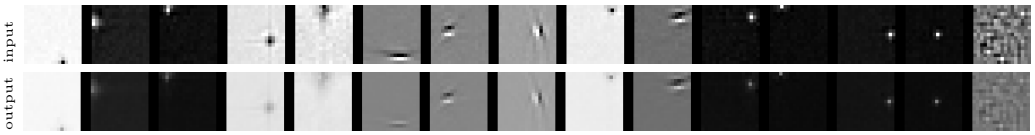


Figure 6.14: Random feature detectors (top) and the corresponding feature generators (bottom) in a trained MLP with one hidden layer.

Figure 6.14 shows some feature detectors (top row) and the feature generators corresponding to each feature detector (bottom row). Scaling of the pixel values was performed separately for each pair of feature detector and feature generator. The feature detectors are similar in appearance to the corresponding feature generators, up to a scaling factor. The feature detectors can be classified into three main categories: 1) feature detectors resembling Gabor filters 2) feature detectors that focus on just a small number of pixels (resembling a dot), and 3) feature detectors that look noisy. Most feature detectors belong to the first and second category. The Gabor filters occur at different scales, shifts and orientations. Similar dictionaries have also been learned by other denoising approaches. It should be noted that MLPs are not shift-invariant, which explains why some patches are shifted versions of each other. Similar features have been observed in denoising auto-encoders [120].

In Figure 6.15, the feature detectors have been sorted according to their standard deviation. We see that the feature detectors that look noisy have the lowest standard deviation. The noisy feature detectors therefore merely look noisy because of the normalization according to which they are displayed. Because the noisy-looking feature detectors have different mean values, we can interpret them as various DC-component detectors.

Denoising auto-encoders are sometimes trained with "tied" weights: The feature detectors are forced to be identical to the output bases. We observe that the learned feature detectors and feature generators look identical up to a scaling factor without the tying of the weights. This suggests that the intuition behind weight tying is reasonable. However, our observation also suggests that better results might be achieved if the feature detectors and feature generators are tied, but allowed to have different scales.

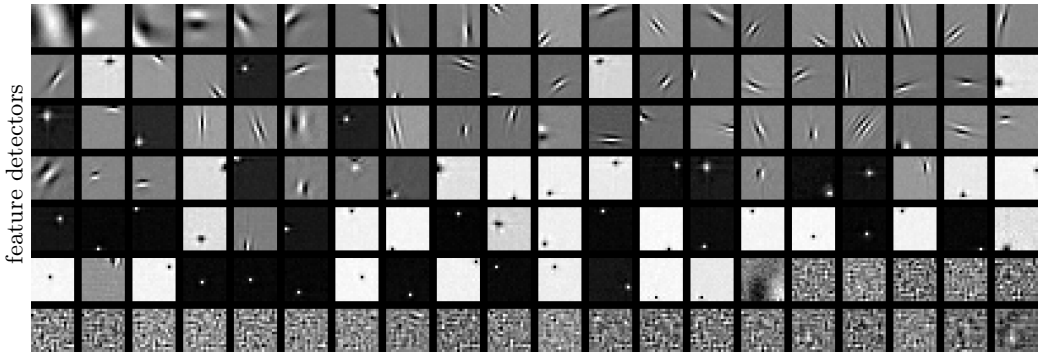
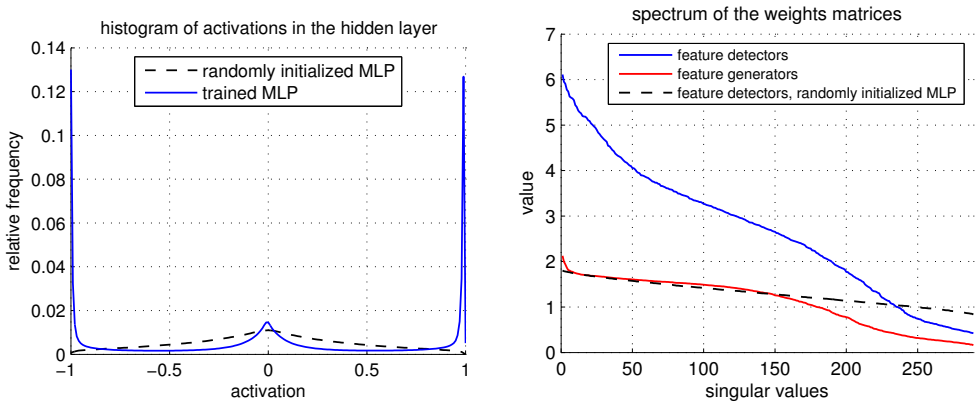


Figure 6.15: Selection of feature detectors in an MLP with a single hidden layer sorted according to their standard deviation. We chose every 15th feature detector in the sorted list. The sorting is from left to right and from top to bottom: The top-left patch has the highest standard deviation, the bottom-left patch the lowest.

Activations: The MLP learned a dictionary in the output layer resembling the dictionaries learned by sparse coding methods, such as KSVD. This suggests that the activations in the last hidden layer might be sparse. We therefore ask the question: What is the behavior of the activations in the hidden layer?

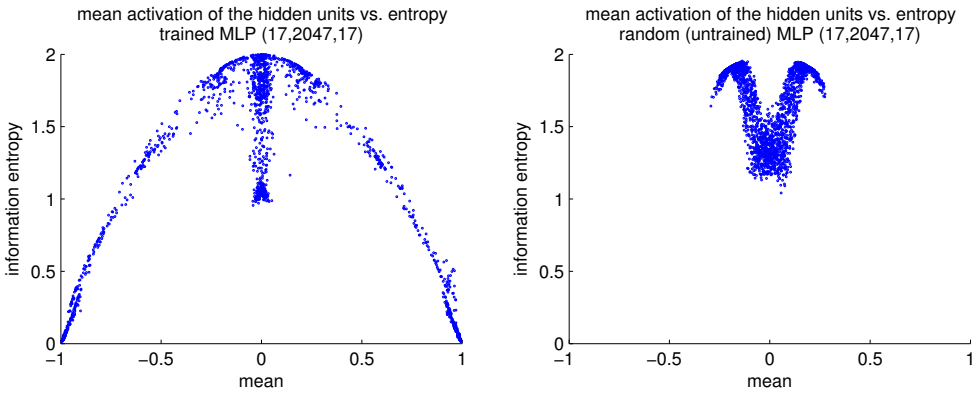


(a) (b)

Figure 6.16: (a) Histograms of the activations in the hidden layer of a one hidden layer MLP. (b) Spectrum of the feature detectors and feature generators.

Figure 6.16a shows a histogram of the activations of all hidden units in both a trained MLP and a random MLP, evaluated on the 500 images in the Berkeley dataset. The activations are centered around zero in the case of the random MLP. The activations in the trained MLP however are almost completely binary: The activations are either close to -1 or close to 1, but seldom in between. This is an indication that the training process is completed: The activities lie on the saturated parts of the tanh transfer function, where the derivative is close to zero. The gradient that is back-propagated to the first layer is therefore mostly zero. This also answers our question: The activations are not sparse. We will provide a further interpretation for this observation later in this section.

Entropy: To measure the usefulness of neurons, we estimate the information entropy of their activation distributions. We plot the mean activations of hidden units against their entropy $H(X) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i)$ with four bins of equal size in Figure 6.17a. We



(a) (b)

Figure 6.17: (a) In a trained MLP, units with small mean tend to have high entropy. This means that these units are highly active; only their mean activation is close to 0. (b) In an untrained MLP, no units have high mean.

repeat the experiment for an untrained MLP in Figure 6.17b. We see that units with high entropy tend to have a low absolute mean, and that units with low absolute mean have high entropy. The reverse is also true: units with low entropy have a high absolute mean and units with a high absolute mean have low entropy. The entropy of the units in a random MLP is higher than in a trained MLP. This is explained by the fact that the random MLP has no units with high absolute mean. These observations allow us to conclude that the units that have a mean close to 0 also have a binary behavior: They are either 1 or -1 and seldom have a value in between. In fact, we can say that these units take value 1 in approximately 50% of the cases and value -1 in the remaining cases. They therefore have a high entropy.

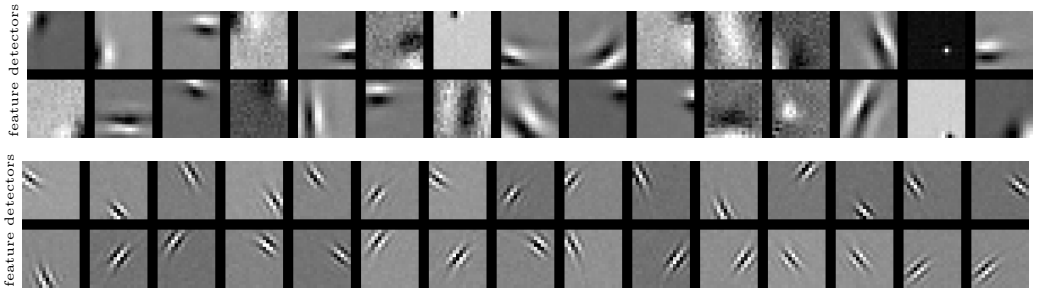


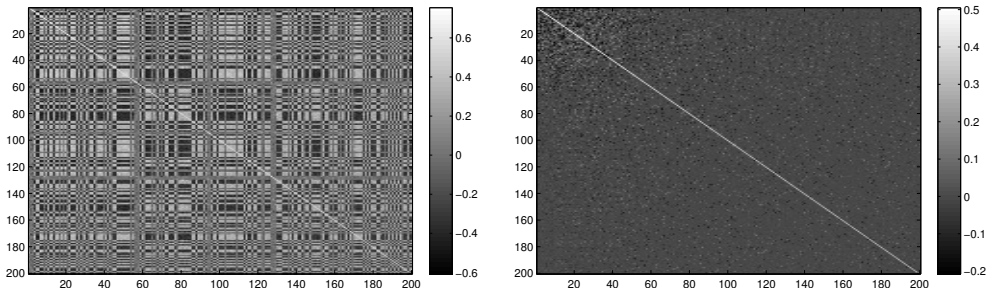
Figure 6.18: Feature detectors of the units with the highest (top) and lowest (bottom) entropy.

Figure 6.18 shows the feature detectors of the units with the highest and lowest entropy. The feature detectors with the lowest entropy all resemble high-frequency Gabor filters of different positions and orientations. A possible explanation for their low entropy is that these filters are highly selective. Only few patches cause these filters to activate.

Weight spectrum: We perform an SVD-decomposition of the weight matrices of both the trained and the random MLP and plot the spectrum of the singular values, see Figure 6.16b. For the random MLP, we omit the spectrum of the feature generators because its shape is identical to the spectrum of the feature detectors. This is due to the initialization procedure and symmetrical architecture.

The similar shape of the spectra in the trained MLP was expected: the feature detectors and feature generators are similar in appearance, see Figure 6.14. The larger singular values for the feature detectors is a reflection of the fact that the norms of the feature detectors is larger than the norm of the feature generators (also seen in Figure 6.14).

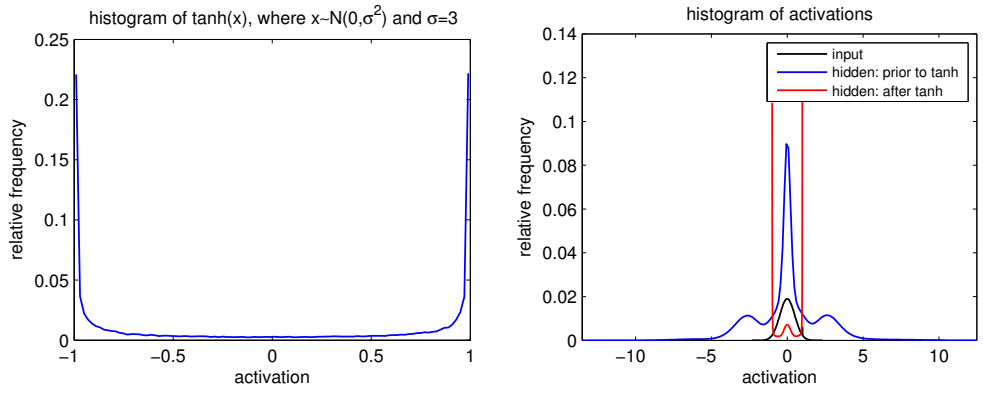
The spectrum for both the feature detectors and the feature generators is relatively flat, which is an indication that the feature detectors are diverse: Strong correlations between feature detectors would cause small singular values. The fact that there are no singular values with value zero means that the output bases matrix has full rank. The spectrum of the random MLP is even flatter: it also has full rank. This means that the output bases of both the trained and the random MLP are able to approximate any patch.



(a) (b)

Figure 6.19: (a) Many hidden units are strongly correlated when image data is given as input. (b) The hidden units are not strongly correlated when noise is given as input.

Activation correlations: Figure 6.19a shows the covariance matrix between the 200 hidden units of the trained MLP with the highest variance, when image data is provided as input. We see that activations between units are highly correlated. This is a reflection of the fact that many of the features detected by the filters tend to occur simultaneously in image patches. Figure 6.19b shows that this observation does not hold when noise is provided as input.



(a) (b)

Figure 6.20: (a) Applying a squashing function on a normally distributed vector with high variance creates a binary distribution. (b) AWG noise with $\sigma = 25$ will cause mostly binary activations in the hidden layer.

How do the binary codes arise? We observed in Figure 6.16a that the codes in the hidden layer are almost completely binary. This observation is surprising: The binary distribution was not explicitly enforced and the distribution of activations is usually different [7]. A possible explanation would be if the activities prior to the application of the tanh-function have high variance. Applying the tanh-function on a normally-distributed vector with high variance indeed creates a binary distribution, see Figure 6.20a.

Is this explanation plausible? A supporting argument is that the feature detectors shown in Figure 6.14 have high norm compared to their corresponding output bases. The high norm of the filters could cause high activations in the hidden layers.

We now feed AWG noise with $\sigma = 25$ into the MLP. The histograms of the activations prior and after application of the tanh-layer are shown in Figure 6.20b. We observe that the activations before the tanh-layer indeed have high variance and that the activations after the tanh-layer are indeed mostly binary. We conclude that the binary activities in the hidden layer are due to activities with high variance prior to the tanh-layer, which are in turn due to feature detectors with high norm.

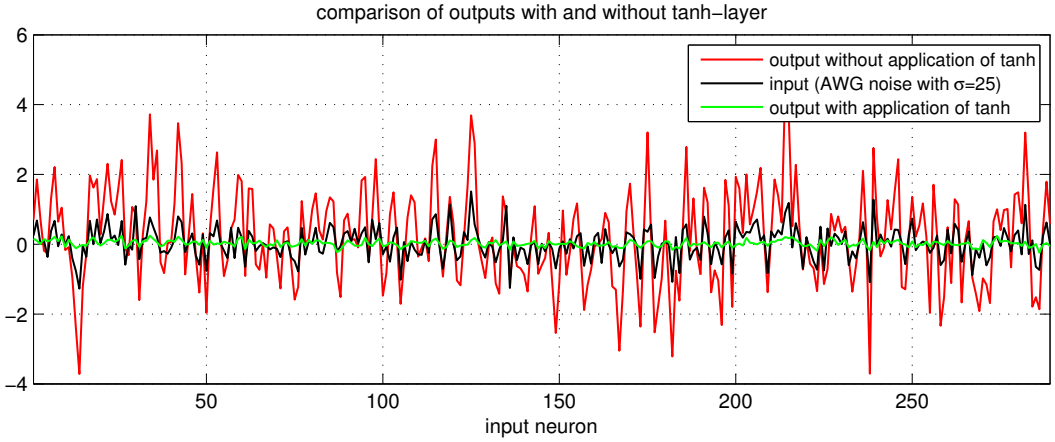


Figure 6.21: The tanh-layer has a denoising effect due to saturation.

How is denoising achieved? We have made a number of observations regarding the behavior of the MLP but have not yet explained why the MLP is able to denoise. Is the binarization effect observed in Figure 6.20 an important factor? To answer this question, we feed an image patch containing only AWG noise with $\sigma = 25$ through the MLP. We compare the output when the tanh-layer is applied to when the tanh-layer is not applied, see Figure 6.21. Without tanh-layer, the output is more noisy than the input. With tanh-layer however, the output is less noisy than the input. We can therefore conclude that the same thresholding operation responsible for the binary codes is also at least partially responsible for the denoising effect of the MLP.

Thresholding for denoising has been thoroughly studied and dates back at least to "coring" for reducing television noise [98]. Typically, a thresholding operation is performed in some transform domain, such as a wavelet domain [94]. However, the thresholding operations typically affect small values most strongly: In the case of *hard thresholding*, values close to zero are set to zero and all other values are left unchanged. In the case of *soft thresholding*, all values are reduced by a fixed amount. Then, values close to zero are set to zero. In the MLP, the situation is reversed: Values close to zero are left unchanged. Only large absolute values are modified by the tanh-layer. We call this effect *saturation*.

We have seen that the saturation of the tanh-layer can explain why noise is reduced. However, denoising can always be trivially achieved by removing both noise and image

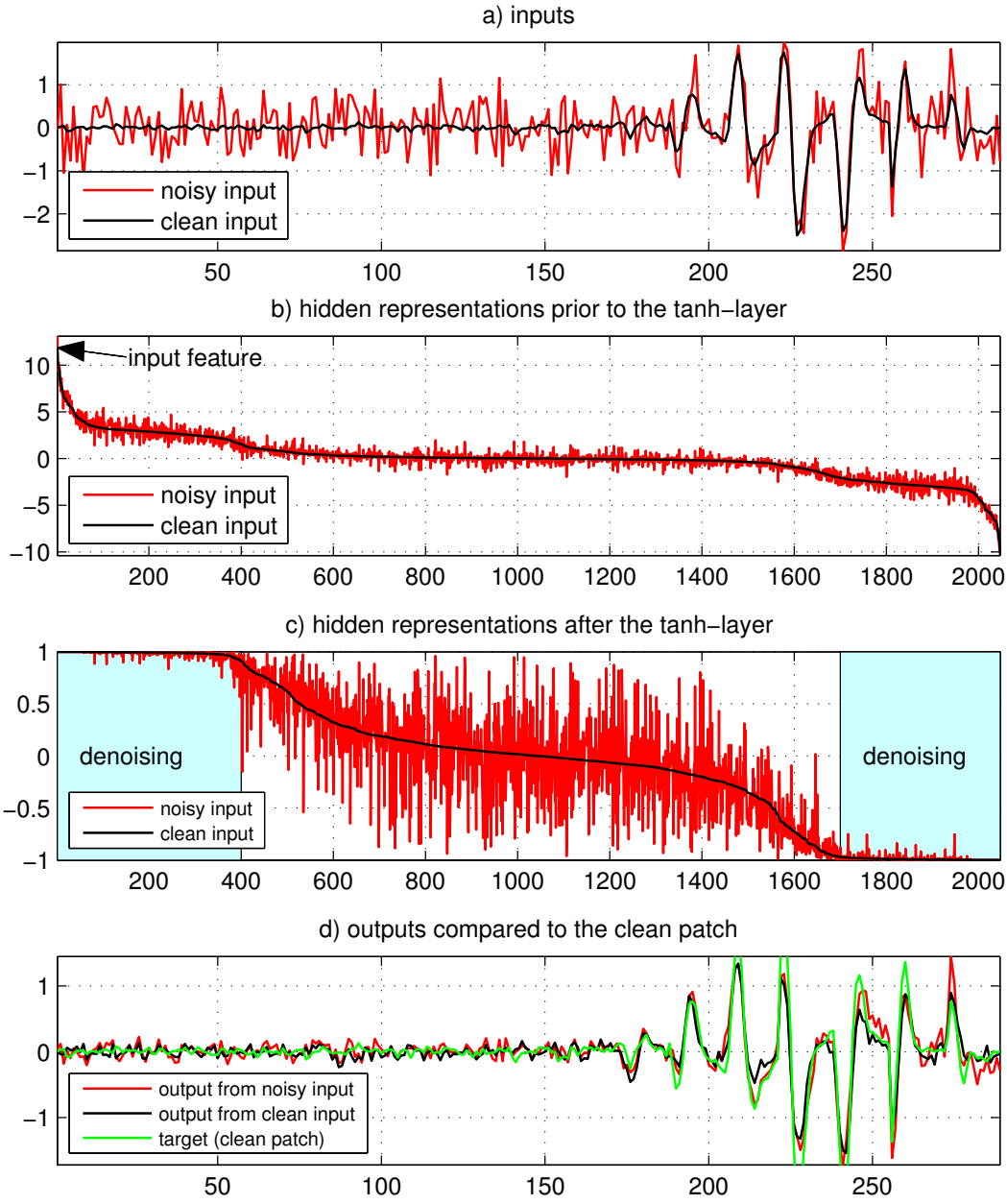


Figure 6.22: Denoising in action: The input maximizes the activity of one feature detector (hidden neuron). Other feature detectors are also strongly activated. After the tanh-layer, the noise has almost no effect on the feature detectors that are highly active. The activations in b) are sorted and the activations in c) use the same sorting. Denoising happens mostly in the blue areas in c).

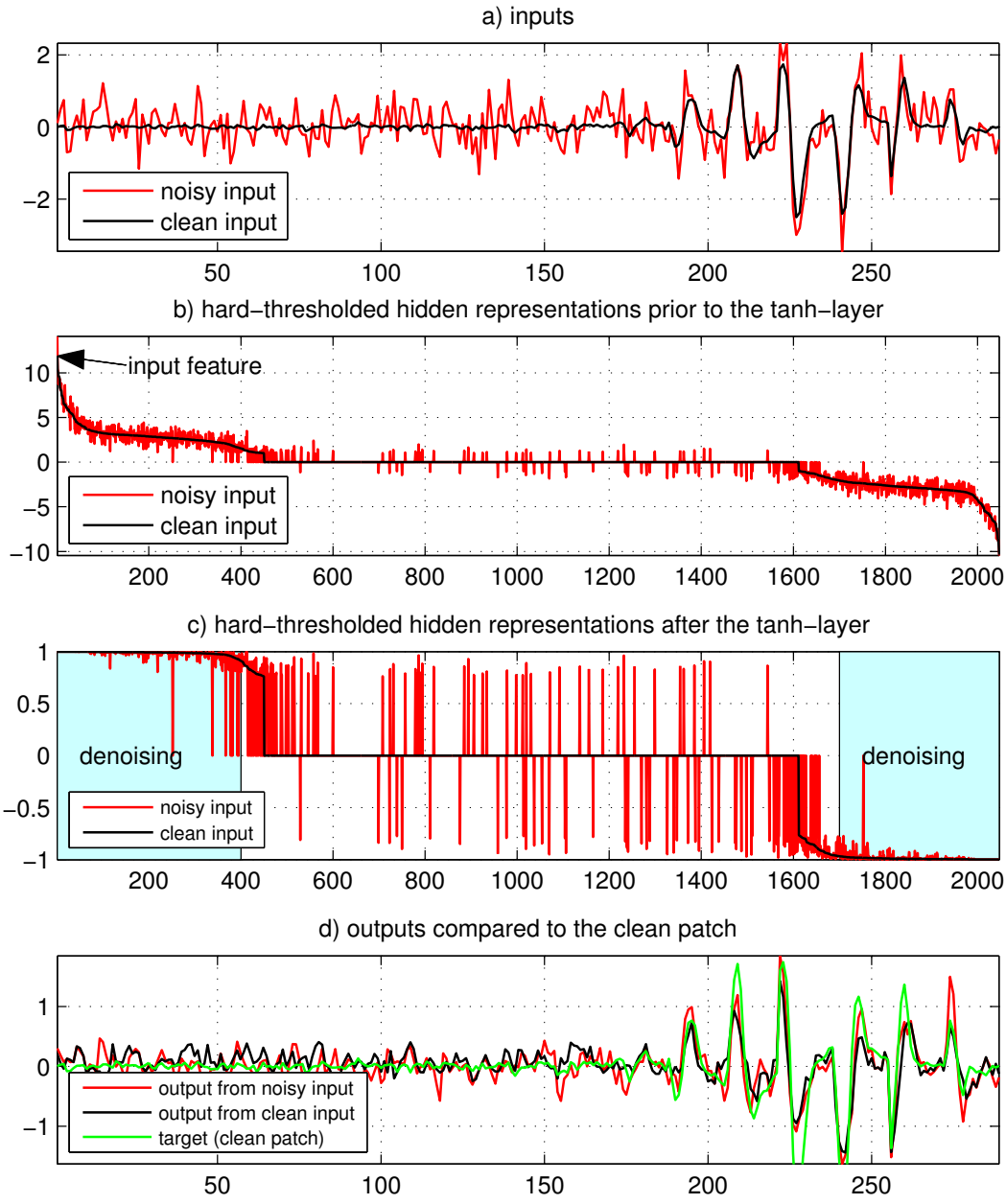


Figure 6.23: Denoising through hard-thresholding: Setting the less import feature detectors to 0 also produces a denoising effect. The activations in b) are sorted and the activations in c) use the same sorting.

information. We therefore ask the question: Why are image features preserved? We proceed by example. As input, we will use the feature detected by one of the feature detectors. As a comparison, we will use as input a noisy version of this feature, see Figure 6.22a. The clean input has the effect of maximizing the activity of its corresponding feature detector prior to the tanh-layer, see Figure 6.22b. Other feature detectors also have a high value, which should be expected, given the high covariance of the hidden units, see Figure 6.19. We see that the noisy input creates a hidden representation that looks quite different from the one created from the clean input: The noise is still clearly present. After application of the tanh-layer, the noise is almost completely eliminated on the feature detectors with high activity, see Figure 6.22c. This is due to the saturation of the tanh-layer. The outputs look similar to the clean input, see Figure 6.22d. In particular, the noise from the noisy input has been attenuated.

We repeat the experiment performed in Figure 6.22, but this time hard-threshold the hidden activities: Activities in the hidden layer prior to the tanh-layer with an absolute value smaller than 1 are set to 0. Doing so still produces a denoising effect, see Figure 6.23. This observation brings us to the conclusion that the feature detectors with a high activity are the more important ones. This is convenient, because the noise on the feature detectors with high activities disappears due to saturation.

We summarize the denoising process in a one-hidden-layer MLP as follows. Noise is attenuated through the saturation of the tanh-layer. Image features are preserved due to the high activation values of the corresponding feature detectors.

Relation to stacked denoising autoencoders (SDAEs): MLPs with a single hidden-layer which are trained on the denoising task are exactly equivalent to denoising autoencoders. Denoising autoencoders can be stacked into SDAEs [120]. The difference between SDAEs and MLPs with multiple hidden layers is that SDAEs are trained sequentially: One layer is trained at a time and each layer is trained to denoise the output provided by the previous layer (or the input data in the case of the first layer). While our MLPs are trained to optimize denoising performance, SDAEs are trained to provide a useful initialization for a different supervised task.

It has been suggested by Bengio *et al.* [8] that deep learning is useful due to an *optimization* effect: Greedy layer-wise training helps to optimize the training criterion. However, later work contradicts this interpretation: Erhan *et al.* [32] suggest that SDAEs and other deep pre-trained architectures such as deep belief nets (DBNs) are useful due to a *regularization* effect: Supervised training of an architecture (especially a deep one) using stochastic gradient descent is difficult because of an abundance of local minima, many of them poor (in the sense that they do not generalize well). The unsupervised pre-training phase imposes a restriction on the regions of parameter space that stochastic gradient descent can explore during the supervised phase and reduces the number of local minima that stochastic gradient descent can fall into. Pre-training thus initializes the architecture in such a way that stochastic gradient descent finds a better basin of attraction (again in the sense of generalization).

The fact that activations in the hidden layers of a SDAE are almost completely binary (see Figure 6.16) and relatively high entropy (see Figure 6.17a) was not mentioned by Erhan *et al.* [32], but is in agreement with the regularization interpretation: The fact that the denoising task forces the hidden representations to be binary is a restriction and therefore also a form of regularization. In addition, information about the input should be preserved in order for the hidden representations to be useful. Information about the input is preserved by virtue of the denoising task: The hidden representations have to contain sufficient information to reconstruct the uncorrupted input. The fact that the hidden units have relatively high information entropy is an additional indication that information is preserved.

We have not answered the question if the binary restriction is better than a more classical form of regularization, such as ℓ_1 or ℓ_2 regularization. However, Erhan *et al.* [32] suggest

that pre-training achieves a form of regularization that is different from and indeed more useful than ℓ_1 or ℓ_2 regularization on the parameters (ℓ_2 regularization on the weights is approximately equivalent to ℓ_2 regularization on the activations). Another argument is that binary vectors are easier to manipulate (e.g. classify) than vectors with small norm.

Relation to restricted Boltzmann machines and deep belief nets: The binary activations in the hidden layer of our MLP are reminiscent of restricted boltzmann machines (RBMs) and deep belief nets (DBNs), which usually employ stochastic binary activations during the unsupervised training phase [53]. An additional similarity is that it has been shown that DBNs and stacked denoising autoencoders extract similar features when trained on either hand-written digits or natural image patches [33].

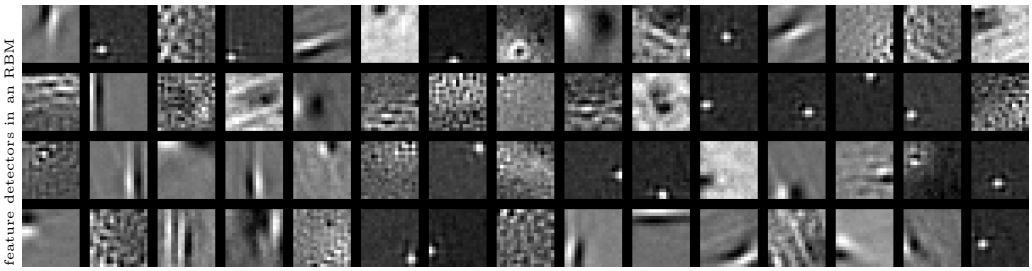
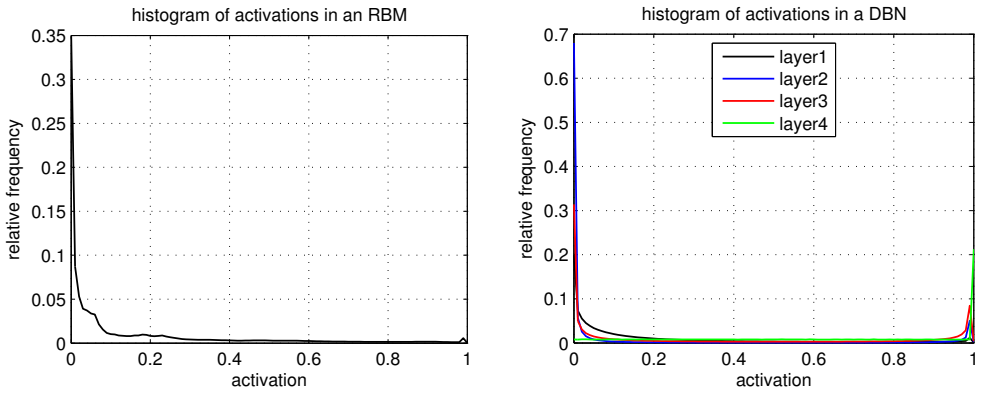


Figure 6.24: Some filters learned by an RBM trained on natural image data (in an unsupervised fashion) with Gaussian visible units, patches of size 17×17 and 512 stochastic binary hidden units. The filters resemble those learned by an MLP on the denoising task.

We trained an RBM with Gaussian visible units on image patches of size 17×17 using contrastive divergence [51, 52]. Figure 6.24 shows that the filters learned by the RBM are similar in appearance to the filters learned by our one-hidden layer MLP, which is in agreement with the findings of Erhan *et al.* [33].



(a) (b)

Figure 6.25: (a) Histogram of activations in the hidden layer of an RBM trained in an unsupervised fashion on natural image patches. (b) Histogram of activations in the hidden layers of a DBN trained in an unsupervised fashion on handwritten digits [54].

The activations of the RBM are binary and stochastic during the unsupervised pre-training phase. It is possible to use the weights learned during pre-training for a supervised task, in which case the hidden units are allowed to take real values. After unsupervised learning of our RBM, we observe the distribution of the real-valued activations in the hidden layer, see Figure 6.25a. The activations lie between 0 and 1 instead of between -1 and 1

for our MLP because of the use of the logistic function instead of tanh. We see that the activations are sparse and do not show the binary behavior exhibited by our MLP.

We also used the code provided by Hinton and Salakhutdinov [54] to train a deep belief net (DBN) on hand-written digits. After pre-training, the activation in all layers is also sparse, see Figure 6.25b. We see that sparsity occurs in the hidden layers even when not explicitly enforced, as proposed by Hinton [52].

Summary: MLPs with one hidden layer denoise by detecting features in the noisy input patch. Each feature detector responds maximally to a single feature, but usually many features are detected simultaneously (see Figure 6.19). The denoised output corresponds to a weighted sum of each feature detector, see Figure 6.14, where the weight depends on the response of the feature detector. The features are mostly Gabor filters of different scales, locations and orientations. Similar features are observed when training other models on natural image data, such as RBMs, see Figure 6.24. The features are informative in the sense that many hidden units have high information entropy, see Figure 6.17b. Noise is removed through saturation of the tanh-layer. Saturation is achieved through feature detectors with high norm, which in turn leads to activations with high variance in the hidden layer before the tanh-layer and mostly binary activations after the tanh-layer, see Figure 6.20. The binary distribution of activations is surprising given the fact that it has not been explicitly enforced, but is useful for denoising and also fits well into the regularization interpretation of denoising auto-encoders proposed by Erhan *et al.* [32].

6.4.2 MLPs with several hidden layers

The behavior of MLPs with a single hidden layer is easily interpretable. However, we have seen in Section 6.2.2 that MLPs with more hidden layers achieve better results. Unfortunately, interpreting the behavior of an MLP with more hidden layers is more complicated. The weights in the input layer and in the output layer can still be represented as image patches, but the layer or layers between the input and output are not so easy to interpret. MLPs with a single hidden layer are identical to denoising autoencoders. This is not true anymore for MLPs with more hidden layers.

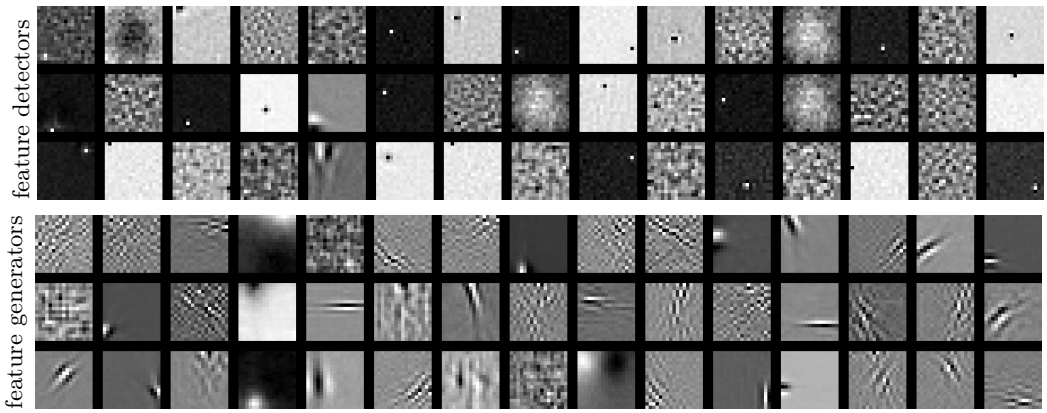


Figure 6.26: Random selection of weights in the input layer (top) and output layer (bottom) for an MLP with two hidden layers.

Two hidden layers: We will start by studying an MLP with architecture (17×172, 2047, 2047,17×172). We repeat the experiment we performed on an MLP with a single hidden layer and look at the feature detectors and feature generators of the MLP, see Figure 6.26. We notice that the feature generators look relatively similar to the output bases of the

MLP with a single hidden layer. However, the feature detectors now look different: Many look somewhat noisy (perhaps resembling grating filters) or seem to extract a feature that is difficult to interpret. Intuition would suggest that these filters are in some sense worse than those learned by the single hidden layer MLP. However, we have seen in Figure 6.2 that better results are achieved with the MLP with two hidden layers than with one hidden layer.

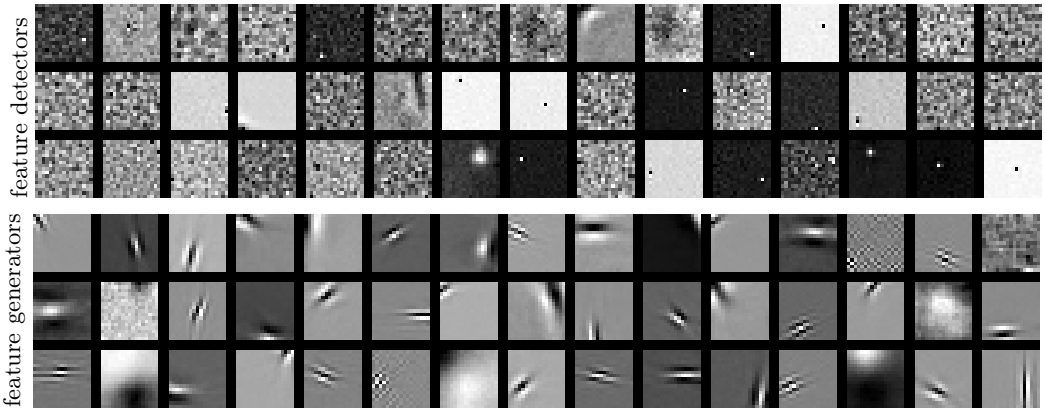


Figure 6.27: Random selection of weights in the input layer (top) and output layer (bottom) for an MLP with four hidden layers.

Four hidden layers: We look at the feature detectors and the output bases of an MLP with architecture (17×17, 2047, 2047, 2047, 2047, 17×17), see Figure 6.27. The output bases resemble those of the MLPs with one and two hidden layers. The feature detectors however look still noisier than those of the MLP with two hidden layers. The results achieved with the MLP with four hidden layers are again better than those achieved with a two hidden layer MLP, see Figure 6.2. We conclude that feature detectors that look noisy or are just difficult to interpret do not necessarily lead to worse denoising results.

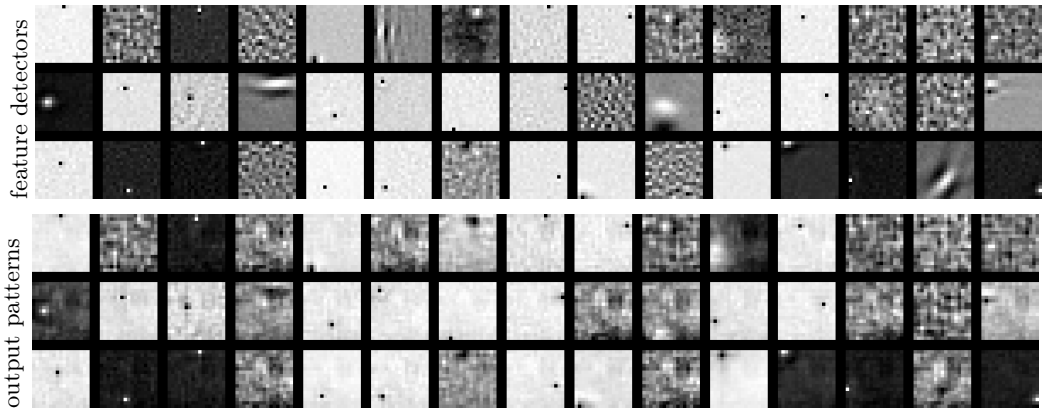


Figure 6.28: Feature detectors (top) and outputs (bottom) corresponding to each feature detector, using an MLP with two hidden layers. The detection of one feature causes the generation of a similar feature in the output.

Outputs corresponding to feature detectors: In the MLP with a single hidden layer, there was a clear correspondence between feature detectors and feature generators: The feature

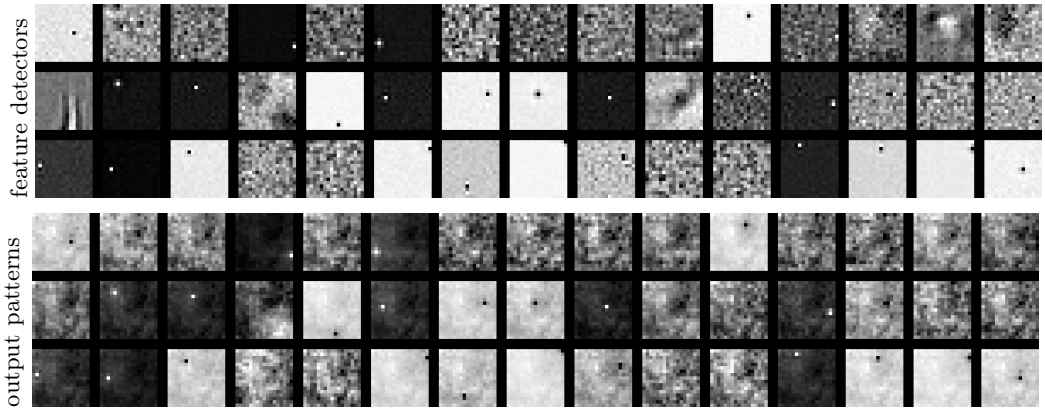


Figure 6.29: Feature detectors (top) and outputs (bottom) corresponding to each feature detector, using an MLP with four hidden layers. The detection of one feature causes the generation of a similar feature in the output.

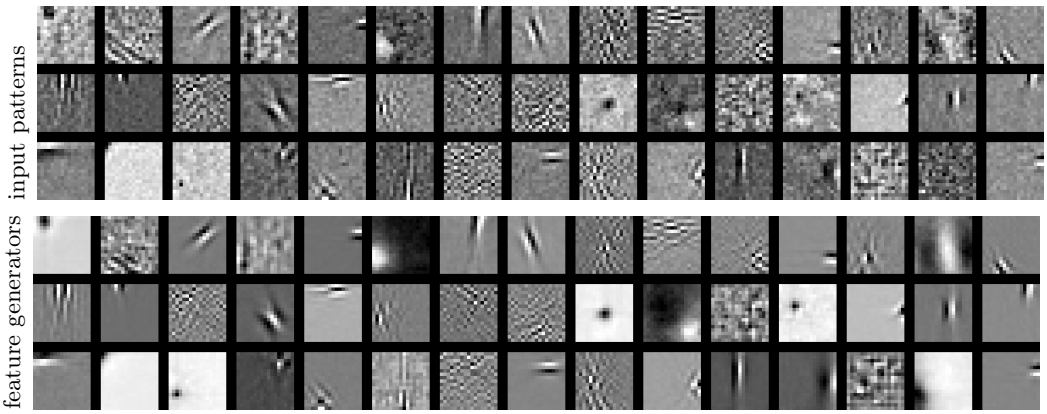


Figure 6.30: Features discovered through activation maximization (top) and corresponding feature generators (bottom), using an MLP with two hidden layers. The detection of one feature causes the generation of a similar feature in the output.

generators looked identical to their corresponding feature detectors. This correspondence is lost in MLPs with more hidden layers, due to the additional hidden layer separating feature detectors from output bases. Can we still find a connection between feature detectors and corresponding outputs? To answer this question, we activate a single unit in the first hidden layer: The unit is assigned value 1 and all other units are set to 0. We then perform a forward pass through the MLP, but completely ignore the input of the MLP. Doing so provides us with an tentative answer to the question: What output is caused by the detection of one feature? The answer is only tentative because several features are usually detected simultaneously. The activation of more hidden units can cause additional non-linear effects due to the tanh-functions in the MLP. Figure 6.28 and 6.29 show the outputs obtained with an MLP with two and four hidden layers, respectively. Also shown are the feature detectors corresponding to the hidden units causing the outputs. We observe a similar correspondence between feature detectors and outputs as in the case of a single hidden layer MLP. The effect is more visible with the MLP with two hidden layers than with the MLP with four hidden layers. The fact that the outputs do not perfectly correspond to their feature detectors can be explained by the fact that during training, features are never detected separately, but always in combination with other features.

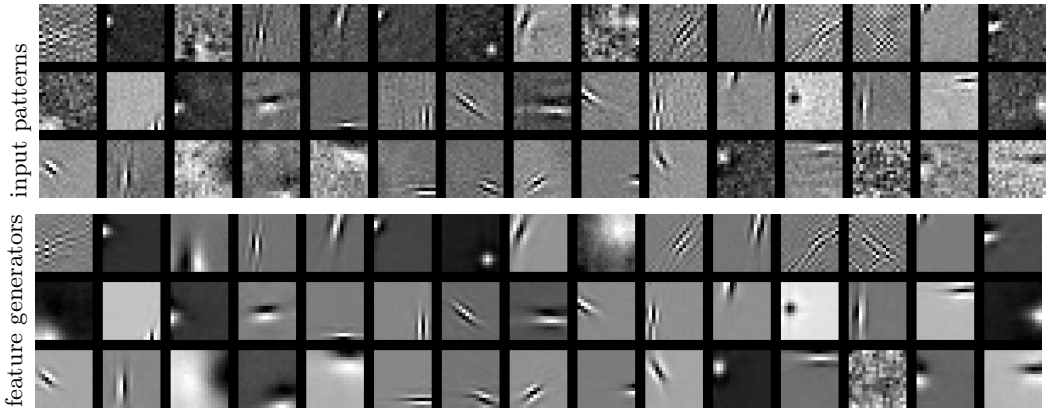


Figure 6.31: Features discovered through activation maximization (top) and corresponding feature generators (bottom), using an MLP with four hidden layers. The detection of one feature causes the generation of a similar feature in the output.

Inputs maximally activating single output bases: Which inputs cause the highest activation for each hidden neuron? Answering this question should tell us which features the MLP responds to. We answer this question using *activation maximization*, proposed by Erhan *et al*. [33]. Activation maximization is a gradient-based technique for finding an input maximizing the activation of a neuron. We use activation maximization with a step size of 0.1. We initialize the patches with samples drawn from a normal distribution with mean 0 and unit variance. We limit the norm of the patch to the norm of the initial patch.

We apply activation maximization on neurons in the last hidden layer of the MLPs with two and four hidden layers. The procedure indeed finds interesting features, see Figures 6.30 and 6.31. Even more interesting is the fact that the features found through activation maximization bear a strong resemblance to the feature generators connected to the same hidden neuron.

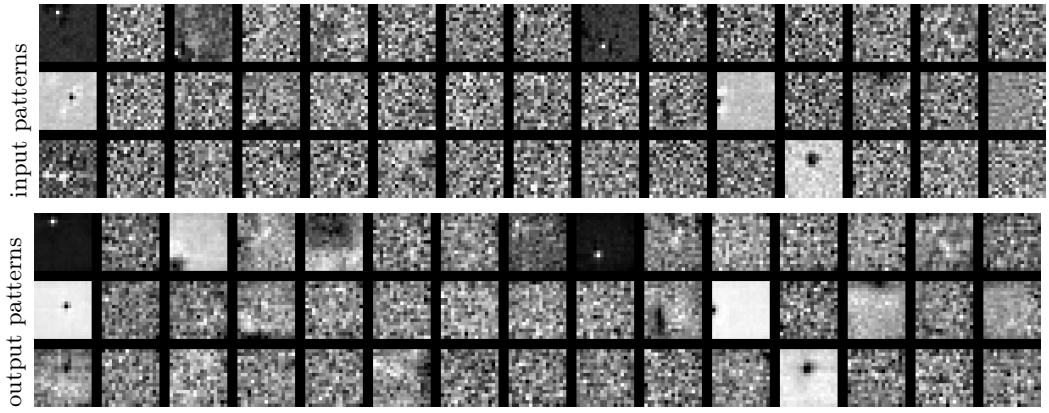


Figure 6.32: Input patterns discovered through activation maximization (top) and output patterns created using one active unit in the hidden layer (bottom), using an MLP with four hidden layers. We used the third hidden layer. We see a correspondence between the input and output patterns.

Input patterns vs. output patterns: We also observe a correspondence between the input patterns discovered through activation maximization and output patterns created by activating a single hidden neuron in deeper layers. Figure 6.32 demonstrates this correspondence

in the third hidden layer of an MLP with four hidden layers.

Summary: MLPs with more hidden layers tend to have feature detectors that are not easily interpretable. In fact, one might be tempted to conclude that they are inferior in some way to the feature detectors learned by an MLP with a single hidden layer, because many of the feature detectors look noisy. However, the denoising results obtained with MLPs with more hidden layers is superior. The visual appearance of the feature detectors is therefore not a disadvantage. The better denoising results can be explained by the higher capacity of MLPs with more hidden layers. MLPs with more hidden layers also seem to operate according to the same principle as MLPs with a single hidden layer: If a feature is detected in the noisy patch, a weighted version of the feature is added to the denoised patch.

6.4.3 MLPs with larger inputs

We now consider the MLP that provided the best results on AWG noise with $\sigma = 25$, see Figure 6.2. The MLP has architecture $(39 \times 39, 3072, 3072, 2559, 2047, 17 \times 17)$. The main difference between this MLP and the previous ones is that the input patches are larger than the output patches. An additional difference is the somewhat larger architecture.

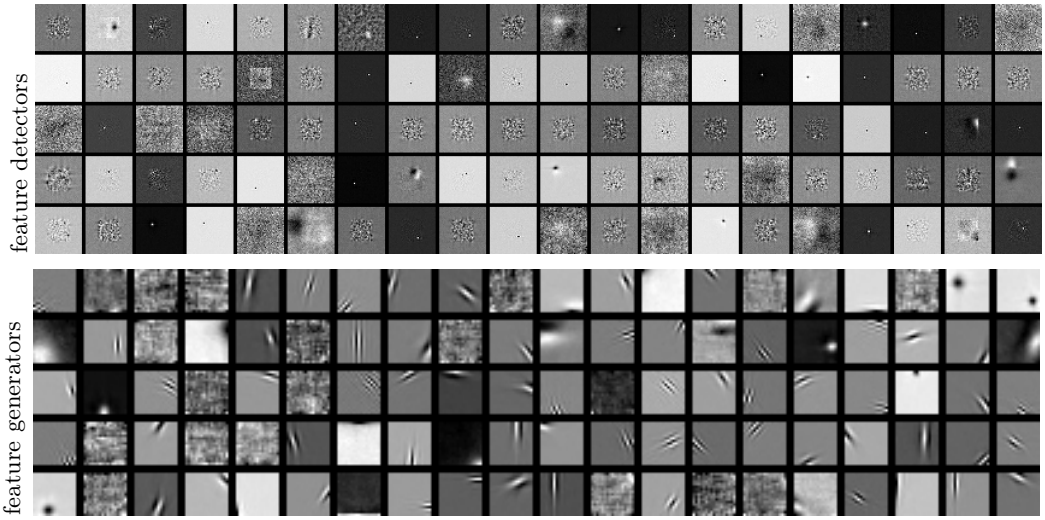
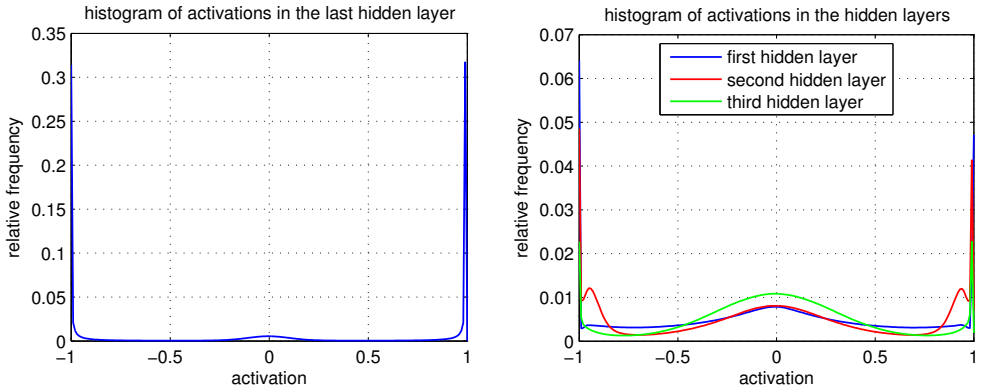


Figure 6.33: Random selection of weights in the input layer (top) and output layer (bottom) of the MLP providing the best results: $(39, 3072, 3072, 2559.2047, 17)$. This MLP has input patches of size 39×39 and output patches of size 17×17.

Feature detectors and feature generators: Figure 6.33 shows a set of feature detectors and feature generators for the MLP with larger input patches. The feature generators look similar to those learned by other MLPs. However, the feature detectors again look somewhat different: many seem to focus on the center area of the input patch. In addition, many look noisy. The fact that many feature detectors focus on the center area of the input patch can be explained by the fact that the output patches are smaller than the input patches. The target patches correspond to the center region of the input patches. Correlations between pixels fall with distance, which implies that the pixels at the outer border of the input patch should be the least important for denoising the center patch.

Activations: The activations in the last hidden layer are almost completely binary, see Figure 6.34a. This effect was also observed on an MLP with a single hidden layer, but



(a) (b)

Figure 6.34: (a) Histograms of the activations in the last hidden layer. (b) Histograms of the activations in the first three hidden layers.

is now even more pronounced. The activations in the other hidden layers are not binary: They frequently lie somewhere between -1 and 1, see Figure 6.34b and resemble a typical distribution [7]. The denoised output patches are therefore essentially constructed from binary codes weighting elements in a dictionary.

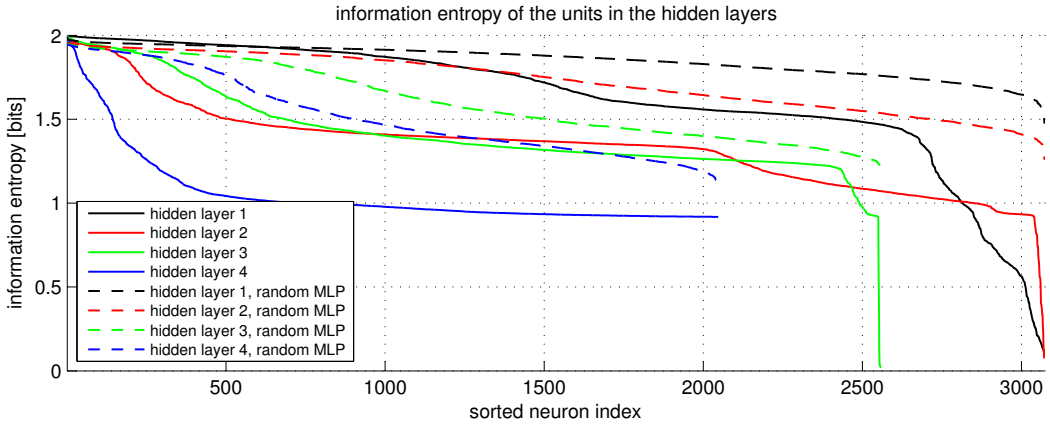


Figure 6.35: Information entropy of the units in the different hidden layers. All units in the last hidden layer have high entropy.

Entropy: An MLP with a single hidden layer had some hidden units with entropy close to zero. Is this also the case for MLPs with more hidden layers? We evaluate the information entropy of the units in the various hidden layers, see Figure 6.35. We again used four bins of equal size. We also compare against a randomly initialized MLP. We observe that the entropy is lower for the trained MLP than for the randomly initialized MLP, which was also observed on an MLP with a single hidden layer. However, this time, all the units in the last hidden layer have high information entropy. In the remaining layers, some units have low information entropy.

Figure 6.36 shows the feature detectors connected to the units with highest and lowest entropy, respectively. Figure 6.37 shows the feature generators with the highest and lowest entropy, respectively. The feature detectors with the highest entropy look different from the

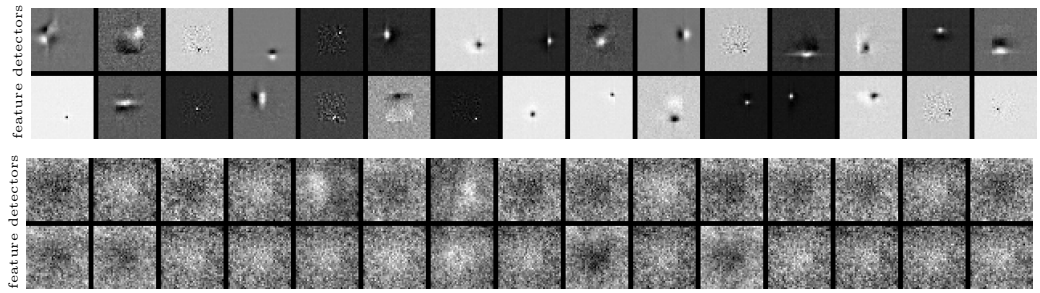


Figure 6.36: Feature detectors of the units with the highest (top) and lowest (bottom) entropy.

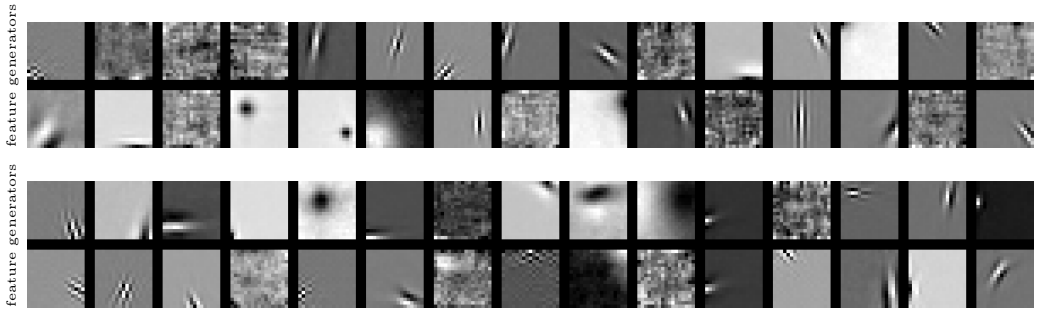


Figure 6.37: Feature generators of the units with the highest (top) and lowest (bottom) entropy.

feature detectors with the lowest entropy. The latter all look similar: All are noisy and seem to loosely focus on a region in the center of the patch. The feature detectors with the highest entropy look more clearly defined. For the feature generators, no clear difference is observed. This is perhaps due to the fact that all output bases have high information entropy. The feature detectors with the lowest entropy almost always have the same activation value and are therefore probably also not very helpful in terms of denoising results.

Approximation ability: We have seen that the MLP does not perform as well as other methods on the image "Barbara". We now ask the question: Is the dictionary formed by the last layer of the MLP the reason why some images cannot be denoised well? In other words, is it possible to approximate any image patch arbitrarily well using that dictionary, or are there images that are difficult to approximate? An additional constraint is that the code vector weighting the dictionary is not allowed to contain values below -1 or above 1 due to the tanh layer.

To answer this question, we try to approximate images patch-wise using the dictionary formed by the last layer. In other words, we try to approximate each image patch x of a clean image using our dictionary D, and proceed in a sliding-window manner. We average in the regions of overlapping patches. Formally, we solve the following problem:

$$\min ||x - D\alpha||_2 \quad \text{s.t.} -1 \preceq \alpha \preceq 1. \tag{6.1}$$

Table 6.1 lists the results obtained on the 11 standard test images, as well as one image containing only white Gaussian noise with $\mu = 127.5$ and $\sigma = 25$ (row "Noise"). We see that all images (including the noise image) can be almost perfectly approximated, though the result on image Barbara is slightly worse than on other images. We therefore conclude that the dictionary in the last layer by itself cannot be the reason why some images are not denoised well. Any image can be well approximated using the dictionary and codes with

image name	PSNR
Barbara	127.45dB
Boat	187.76dB
Cameraman	166.00dB
Couple	192.58dB
Fingerprint	198.12dB
Hill	192.05dB
House	195.68dB
Lena	192.19dB
Man	190.21dB
Montage	174.14dB
Peppers	189.49dB
Noise	161.28dB

Table 6.1: Ability of the dictionary to approximate images.

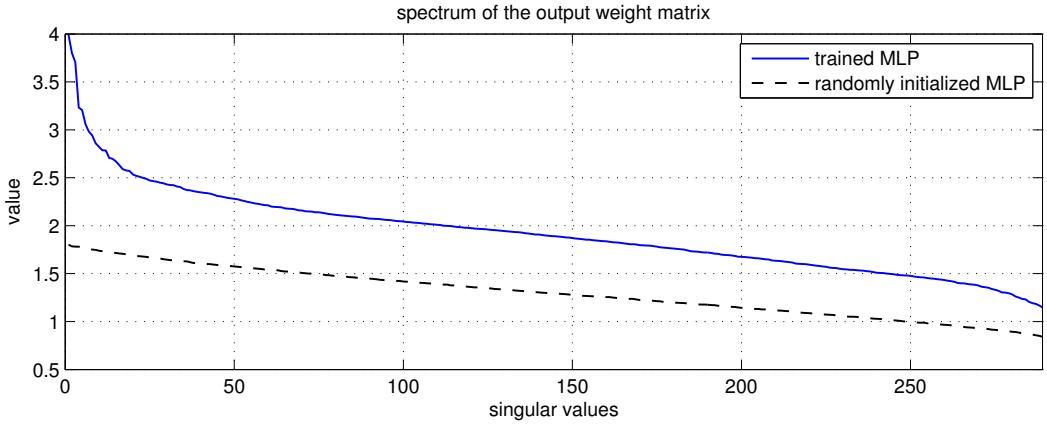


Figure 6.38: Spectrum of the output bases.

values in range from -1 to 1.

A related observation is that the weights in the last layer have no zero singular values, see Figure 6.38. This implies that the matrix has full rank and can therefore approximate any patch, when the lower- and upper-bound constraints are disregarded. We also observe that the spectrum is relatively flat, which was also the case for the MLP with a single hidden layer. This implies that the output bases are diverse.

Combining the dictionary with sparse coding: Dictionary-based methods for image denoising such as KSVD typically denoise by approximating a noisy image patch using a sparse linear combination of the elements in the dictionary. More formally, one attempts to solve the following problem:

$$\min ||\alpha||_0 \quad \text{s.t. } ||y - D\alpha||_2 \leq \epsilon \tag{6.2}$$

where y is a noisy image patch, ϵ is a pre-defined parameter and $||\cdot||_0$ refers to the ℓ_0 pseudo-norm. Approximate solutions to this problem can be found using OMP [89]. The denoised patch $\hat{x}$ is given by $\hat{x} = D\alpha$. Denoising is performed in a sliding-window manner and averaging is performed where patches overlap.

We ask the question: Can the dictionary learned by the MLP be used in combination with this sparse coding approach? We denoise the 11 standard test images with AWG noise, $\sigma = 25$ using the dictionary learned by the MLP and solve equation (6.2) approximately

image	KSVD [1]	MLP	"MLP + OMP"
Barbara	*29.49*dB	**29.52**dB	28.23dB
Boat	*29.24*dB	**29.95**dB	28.93dB
Cameraman	*28.64*dB	**29.60**dB	28.32dB
Couple	*28.87*dB	**29.75**dB	28.66dB
Fingerprint	*27.24*dB	**27.67**dB	26.88dB
Hill	*29.20*dB	**29.84**dB	28.95dB
House	*32.08*dB	**32.52**dB	30.12dB
Lena	*31.30*dB	**32.28**dB	30.65dB
Man	*29.08*dB	**29.85**dB	28.95dB
Montage	*30.91*dB	**31.97**dB	30.21dB
Peppers	*29.69*dB	**30.27**dB	29.08dB

Table 6.2: Using the MLP's dictionary in combination with OMP.

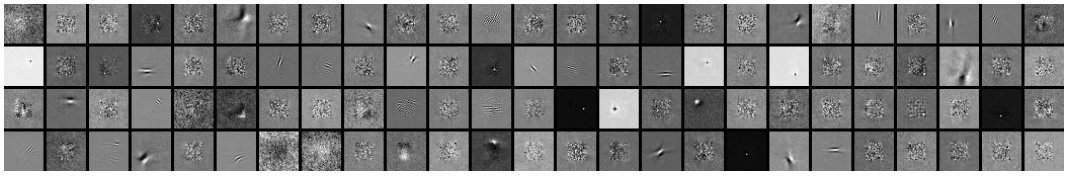
using OMP. We set ϵ similarly to KSVD [1]: $\epsilon = n((C\sigma)^2)$, where n is the dimensionality of the patches (289) and C is a hyper-parameter. We found the best value of C to be 1.05. We normalized all columns of D to have unit norm. The results of this approach are summarized in Table 6.2. The PSNR of the noisy images is approximately 20.18dB.

The denoising results of this approach are not very good. We therefore conclude that the dictionary's ability to denoise is strongly dependent on the codes provided to it. The first three hidden layers of the MLP serve as a mechanism for creating good codes for the last layer.

Inputs maximizing the activation of neurons: Which inputs cause the highest activation for each neuron? We answer this question using two approaches: (i) Activation maximization [33] and (ii) evaluating the activation values for a large number of (non-noisy) image patches.

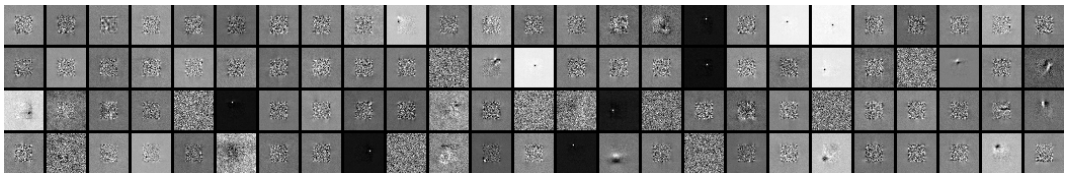
We perform activation maximization as described in section 6.4.2. We also run the MLP on a large number of noise-free natural image patches. For each neuron, we save the input maximizing its absolute activation. We used 6768 natural images, each containing many thousand patches. Figure 6.39 shows the input patterns found through activation maximization as well as the input patches found by inspecting a larger number of natural image patches. We make a number of observations.

- **Focus on the center part:** The patterns found through activation maximization mostly focus on the center part of the patches. This intuitively makes sense: The most important part of the input patch is expected to be the area covered by the output patch. In addition, pixel correlations fall with distance, so pixels that are further away are expected to be less interesting. There are exceptions however: Some patches seem to focus particularly on the patch border.
- **Gabor filters:** Many input patterns resemble Gabor filters. This is true for all hidden layers, but particularly for hidden layers two and four. We also observed this phenomenon in the output layer weights, see Figure 6.33.
- **Random looking patches:** Many input patterns look as if the pixels were set randomly. This is particularly true in hidden layer three.
- **Correlation to natural image patches:** Some input patterns found through activation maximization correlate well with patches found through exhaustive search through a set of natural image patches. For example the patches 6 and 7 from the right in the upper row of hidden layer four. In many cases however, it is not clear that the two procedures find correlating patches. The fourth hidden layer patches seem to indicate that many neurons respond to features with a highly specific location and orientation.



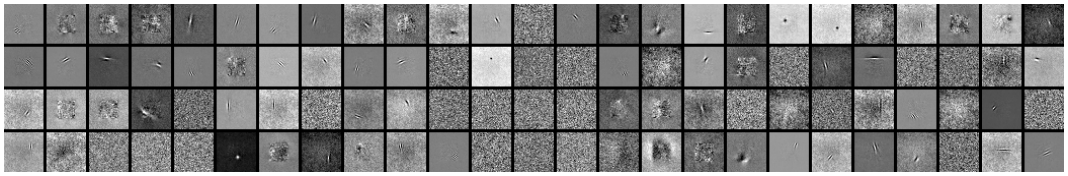
(a) Input patterns maximizing the activation of neurons in the second hidden layer.

(b) Natural image patches maximizing the absolute activation of neurons in the second hidden layer.

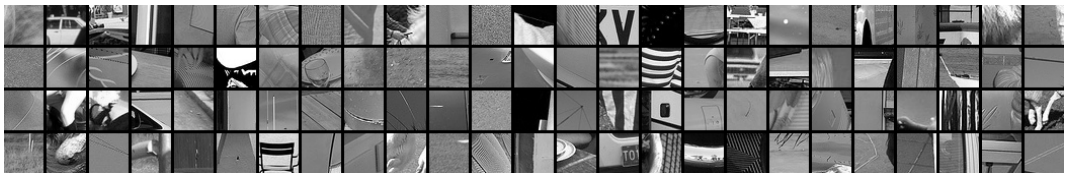


(c) Input patterns maximizing the activation of neurons in the third hidden layer.

(d) Natural image patches maximizing the absolute activation of neurons in the third hidden layer.



(e) Input patterns maximizing the activation of neurons in the fourth hidden layer.



(f) Natural image patches maximizing the absolute activation of neurons in the fourth hidden layer.

Figure 6.39: What features does the MLP respond to?

6.4.4 Comparing the importance of the feature detectors

Some of the feature detectors look random or noisy, see Figure 6.33. Are all the feature detectors useful or are the noisy looking filters less useful? We answer this question by observing the behavior of the MLP when a set of feature detectors is removed (in other words, when only a subset of feature detectors is used). We evaluate the average performance of the network on the 11 standard test images. We remove a feature detectors by replacing its weights with the average value of the feature detector.

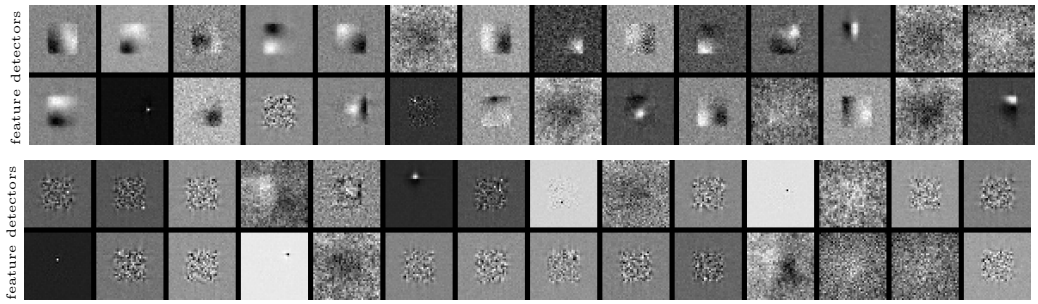


Figure 6.40: Most (top) and least (bottom) important feature detectors (using 1500 feature detectors).

We use an iterative procedure during which 1500 feature detectors are chosen for each iteration. The mean PSNR obtained is assigned to the feature detectors used during that iteration. We average over iterations. The feature detectors yielding the best results (on average) are shown in the top row of Figure 6.40 and the feature detectors yielding the worst results are shown in the bottom row.

It seems that the feature detectors yielding good results on average are more easily interpretable than the ones yielding worse results. The feature detectors yielding good results seem to focus on large-scale features, whereas the filters yielding worse results look more noisy.

6.4.5 Effect of the type and strength of the noise on the feature detectors and feature generators

All observations we have made on the feature detectors and feature generators of the MLPs were made on MLPs trained to remove AWG noise with $\sigma = 25$. We will now make a number of observations for different types and strengths of noise.

How does the strength of the noise affect the learned weights? Figure 6.41 and Figure 6.42 show the feature detectors and feature generators for $\sigma = 10$ and $\sigma = 75$, respectively. The feature generators look similar for the two noise levels. However, the feature detectors look different: For $\sigma = 10$, the feature detectors almost always focus on the area covered by the output patch, whereas for $\sigma = 75$, the feature detectors also consider pixels that are further away. This is in agreement with the findings of Levin and Nadler [68]: When the noise is stronger, larger input patches are necessary to achieve good results. We already provided a similar explanation in Section 6.2.6. This also implies that it is unnecessary to use large input patches when the noise is weak and explains why we achieved better results with smaller patches for $\sigma = 10$, see Figure 6.10.

How does the type of the noise affect the learned weights? Figures 6.43, 6.44 and 6.45 show the feature detectors and feature generators learned with stripe noise, salt-and-pepper noise and JPEG artifacts, respectively. All patches in these figures are of size 17×17. The input weights are strongly affected by the type of the noise: For horizontal stripe noise,

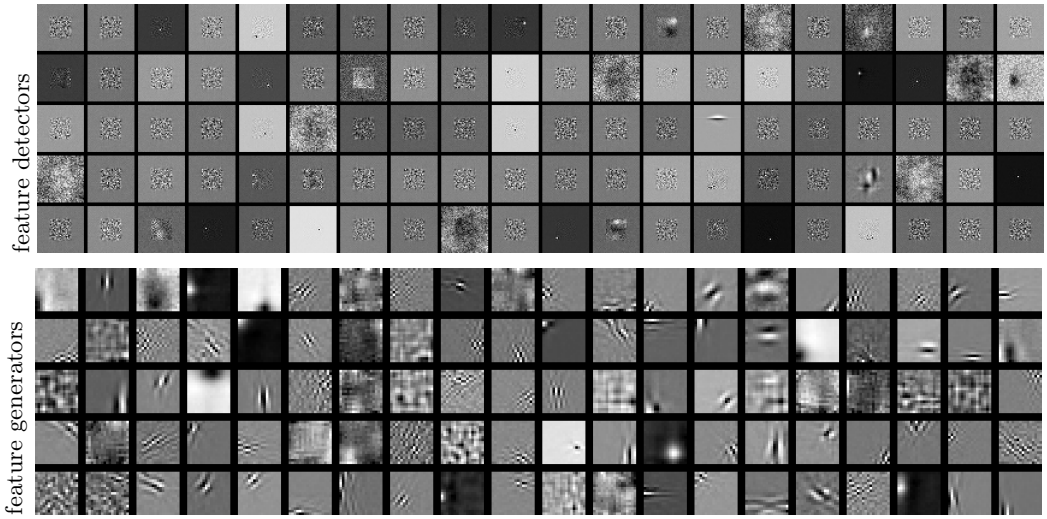


Figure 6.41: Random selection of weights in the input layer (top) and output layer (bottom) for $\sigma = 10$

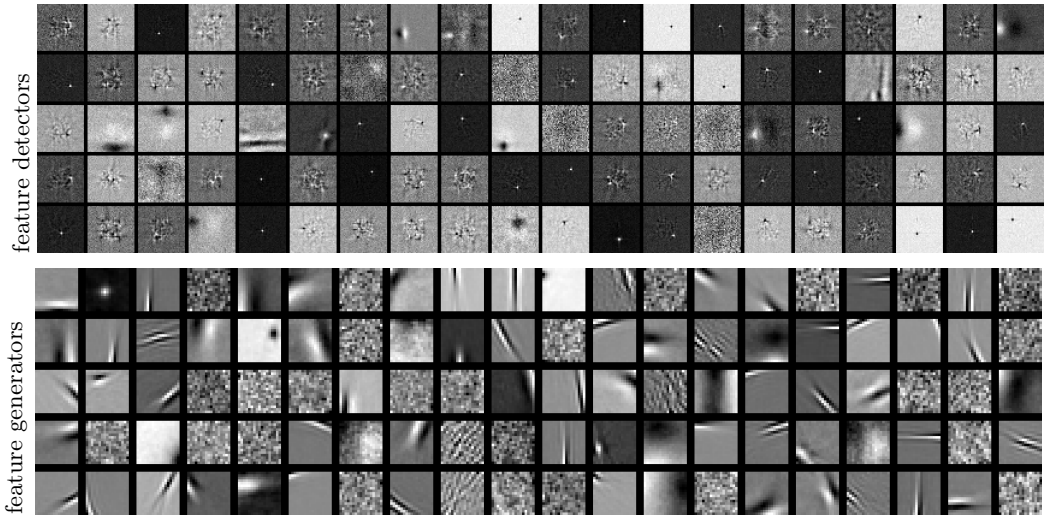


Figure 6.42: Random selection of weights in the input layer (top) and output layer (bottom) for $\sigma = 75$

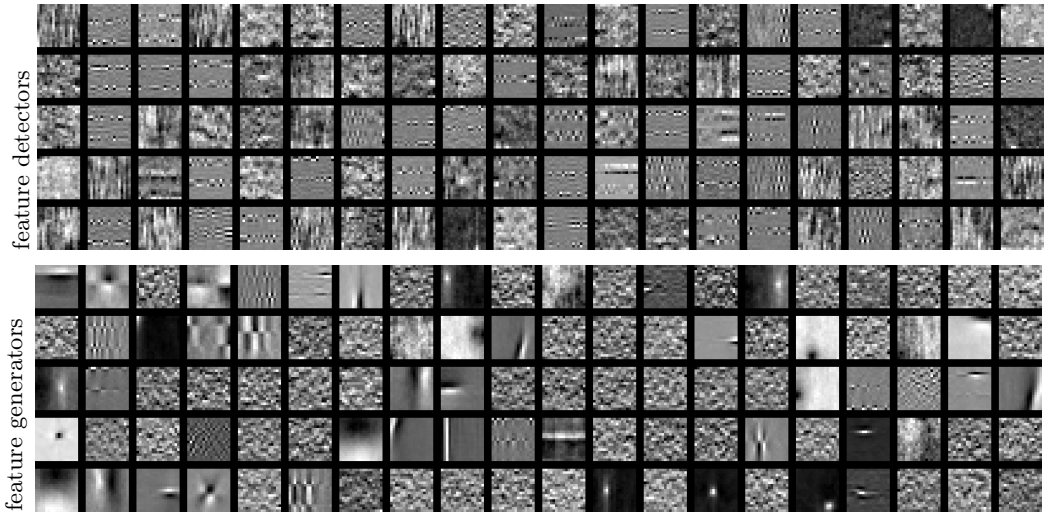


Figure 6.43: Random selection of weights in the input layer (top) and output layer (bottom) for stripe noise

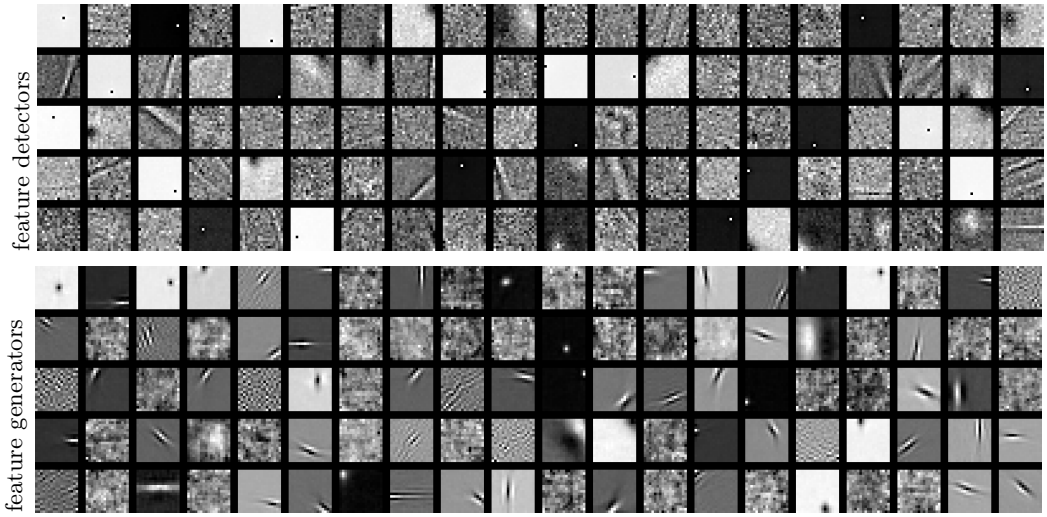


Figure 6.44: Random selection of weights in the input layer (top) and output layer (bottom) for salt and pepper noise

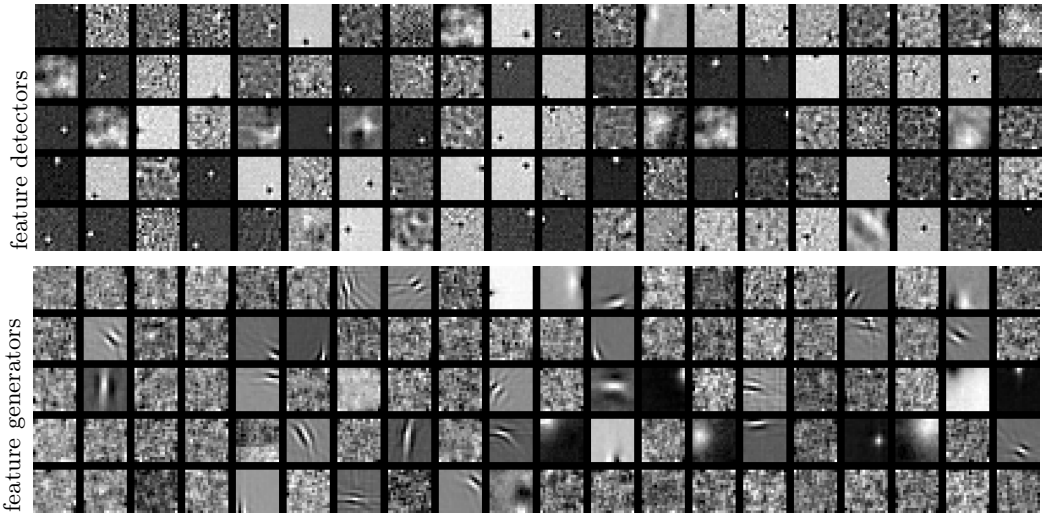


Figure 6.45: Random selection of weights in the input layer (top) and output layer (bottom) for JPEG noise

the feature detectors often have horizontal features that also look like stripes. For salt-and-pepper noise, the feature detectors are often filters focussing on long edges. For JPEG artifacts, the feature detectors are close in appearance to the output weights. The feature generators are also somewhat affected by the type of the noise. This is especially visible for stripe noise, where the feature generators seem to sometimes also contain stripes. It was also observed by Vincent *et al.* [120] that the type of the noise has a strong effect of the learned weights in denoising autoencoders.

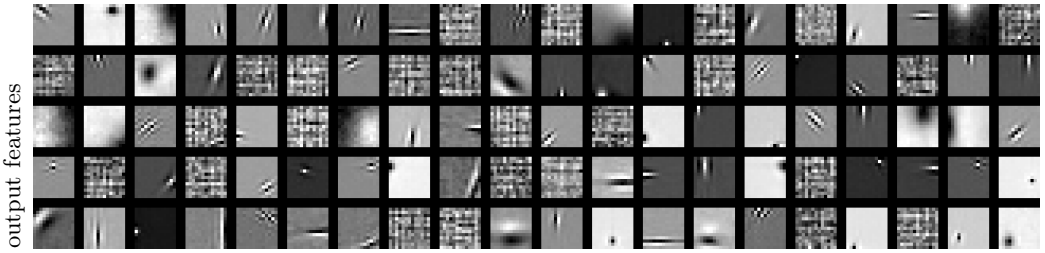


Figure 6.46: Block matching output weights

6.4.6 Block-matching filters

Figure 6.46 shows the feature generators learned by the MLP with block-matching, using $k = 14$ and patches of size 13×13. The feature generators look similar to those learned by MLPs without block-matching.

Figure 6.47 shows a selection of feature detectors learned by the MLP with block-matching. The left-most patch shows the filter applied to the reference patch, and the horizontally adjacent patches show the filters applied to the corresponding neighbor patches. The horizontally adjacent patches all connect to the same hidden neuron. We see that the filters applied to the neighbor patches are usually similar to the filters applied to the reference patch. This observation should not be surprising: The updates of the weights connecting the input patches to a hidden neuron are defined by (i) the gradient at the hidden neuron and (ii) the value of the input pixels. Hence, if the value of the input pixels are similar (this is ensured by the block-matching procedure), the weight updates are also similar.

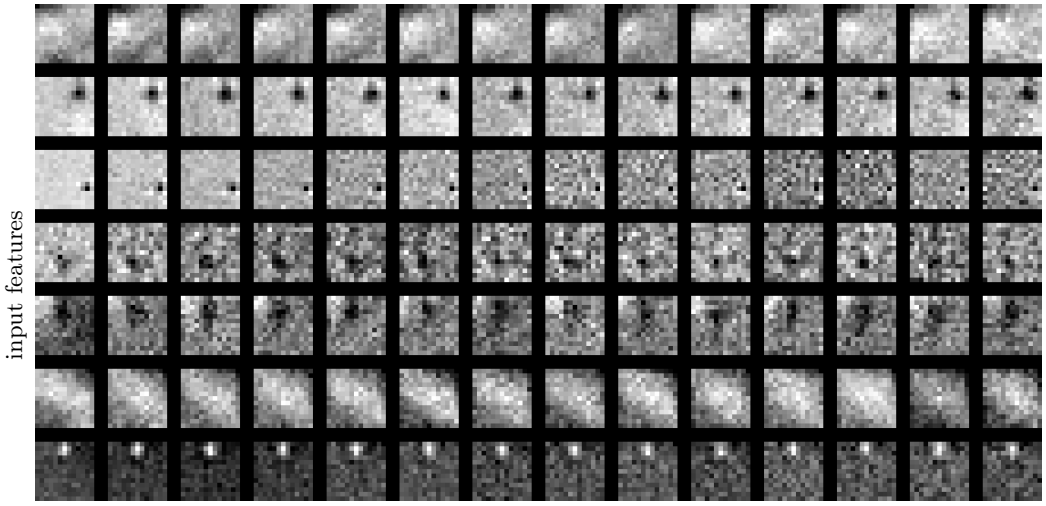


Figure 6.47: Block matching input weights

6.5 Discussion and Conclusion

In Chapter 5, we have shown that it is possible to achieve state-of-the-art image denoising results using MLPs. In this chapter, we have shown how this is possible. In the first part of this chapter, we have discussed which trade-offs are important during the training procedure. In the second part of this chapter, we have shown that it is possible to gain insight about the inner working of the trained MLPs by analysing the activation patterns on the hidden units.

How to train MLPs: We have trained MLPs with varying architectures on datasets of different sizes. We have also varied the sizes of the input as well as of the output patches. The observations made on these experiments allow us to make a number of conclusions regarding image denoising with MLPs: (i) More training data is always good, (ii) more hidden units per hidden layer is always good, (iii) there is an ideal number of hidden layers for a given problem and a given number of hidden units per hidden layer. Going above the ideal number of hidden layers can lead to catastrophic degradations in performance, (iv) increasing the output size requires higher-capacity architectures, and finally (v) fine-tuning with a lower learning rate can lead to important gains in performance.

Other image processing problems such as super-resolution, deconvolution and demosaicking might also be addressed using MLPs, in which case we expect the guidelines described in this chapter to be useful as well. Other problems unrelated to images might also benefit from these guidelines. Indeed, we expect that many difficult problems with high dimensional inputs and outputs could benefit from these insights.

Understanding denoising MLPs: The denoising procedure of MLPs with a single hidden layer can be briefly summarized as follows. Each hidden unit detects a feature in the noisy input and copies it to the output patch. Denoising is achieved through saturation of the tanh-layer. The use of activation maximization [33] and observing outputs obtained by activating a single hidden unit in an MLP allowed us to make observations concerning the internal workings of MLPs with several hidden layers. We have seen that MLPs with several hidden layers seem to work according to the same principle as MLPs with a single hidden layer: The features required to maximize the activation of a hidden unit are often remarkably similar to the output caused by the same hidden unit. This observation is true for each hidden layer.

Denoising with MLPs requires that the tanh-layer saturates, which naturally gives rise to binary representations. This is different from RBMs, which force their hidden representations to be binary. The fact that the representations are binary lends support to the regularization interpretation of denoising autoencoders proposed by Erhan *et al.* [32]. We also note that binary representations are unusual for MLPs: Other problems do not give rise to binary representations [7].

As an alternative to binary representations, we consider sparse representations. Sparse representation in higher dimensional spaces have the well-known benefit of being able to more easily rely on linear operations for a variety of tasks, see for example [73]. Sparsity has been proposed as a form of regularization to train deep belief networks, see [97]. Successful architectures for object recognition [59] also make use of sparse representations, in this case using a procedure called predictive sparse coding proposed by Kavukcuoglu *et al.* [60]. In all cases, achieving sparse representations requires sparsity inducing terms in the optimization criteria, which makes the optimization procedure more complex. We argue that binary representations have similar benefits to sparse representations, but that obtaining binary representations is easier than obtaining sparse representations, using a denoising criterion.

A further similarity between MLPs trained to denoise images and RBMs and denoising autoencoders is the similarity of the features (such as Gabor filters) learned by all three architectures. Unrelated approaches such KSVD [1] learn similar features.

7

Synopsis

In Section 2.12 we proposed to broadly classify denoising methods into two paradigms: Methods focussing on images and methods focussing on noise. Most existing methods belong to the category focussing on images. In this thesis, we have made contributions to both paradigms: The meta-procedure described in Chapter 3 belongs to the category of methods focussing on images. The method with a pixel-specific noise model for denoising astronomical images described in Chapter 4 belongs to the category of methods focussing on noise. The method based on neural networks described in Chapter 5 also belongs to the category of methods focussing on images. More precisely, according to the taxonomy defined in Section 2.12, the method belongs to the sub-category relying on external prior knowledge about images. Here, we summarize the contributions made by this thesis.

A multi-scale meta-procedure for high noise levels: We observed that many existing denoising algorithms focus mainly on the high-frequency components of images and do not handle low frequencies properly. This causes a deterioration of results at higher noise levels. To improve algorithms that do not properly handle low frequencies, we have presented in Chapter 3 a procedure in which an existing denoising algorithm is applied at several scales. The resulting denoised images are then recombined using a procedure resembling a Laplacian pyramid. We have seen that this approach improves the results achieved by many algorithms, especially at high noise levels. Some denoising algorithms cannot be improved using this strategy because they already take a multi-scale approach or use large patch sizes and therefore handle low frequencies properly. The meta-procedure belongs to the class of methods focussing on images, but cannot be assigned to a particular sub-category, because methods belonging to all sub-categories can be combined with the meta-procedure.

Denoising astronomical images: Most denoising algorithms focus on images. However in some situations, it makes sense to consider the class of denoising methods focussing on noise. This is the case when the noise has more structure than additive white Gaussian noise. In Chapter 4 we have seen that such a situation arises in digital photographs of astronomical objects, where an important source of noise is the so-called dark-current noise. Dark-current noise is quite different from additive white Gaussian noise, which is the type of noise that most existing denoising algorithms are designed to remove. In addition, astronomical images are different in appearance from "natural" images (or images of every-day scenes). To denoise such images, we have presented a denoising method relying on a probabilistic description of a given camera's dark-current, as well as on an image prior appropriate for astronomical images. Every pixel of the camera's sensor is treated individually. In laboratory conditions, we have shown that our method provides better results than most denoising methods that are intended for use on natural images. On real astronomical images, we have shown that our method provides visually more appealing results than other methods.

Denoising with multi-layer perceptrons: BM3D [25] is one of the best currently existing denoising methods. The method is engineered and does not rely on learning. Furthermore, the method relies exclusively on internal knowledge: A given noisy image is denoised using only knowledge gained from the noisy image itself. Knowledge concerning all other existing images is ignored. Other recent methods rely on similar approaches. Is it also possible to achieve excellent results using a method that lies on the opposite end of the spectrum and relies heavily on learning instead of engineering and uses external knowledge instead of internal knowledge? In Chapter 5 we present such a method. The method relies on plain neural networks (also called multi-layer perceptrons, or MLPs) that are trained to denoise image patches on a large dataset. Denoising an image is effected by applying a trained MLP patch-wise. On average, this method significantly outperforms competing methods, especially at high noise levels. Our method also achieves results that are superior to one type of theoretical bound on some images, thereby proving by example that this type of bound does not apply to all denoising algorithms. Compared to a second type of theoretical bound, our method makes significant progress towards reaching the bounds. We have also shown that our method performs particularly well in regions with complex textures, where it was theoretically expected that the least amount of improvement is possible.

A limitation of our approach is that one MLP has to be trained for each noise level: An MLP trained on one noise level does not perform well on other noise levels. However, MLPs can be trained on other types of noise, by changing the training data. We have shown that MLPs are able to perform well in JPEG artifact and Poisson noise removal, among others. It is not clear if other denoising methods can always be easily adapted to handle different kinds of noise.

Compared to other approaches, our method sometimes achieves worse results on images with regular, repeating structures. This can be explained by the fact that the methods outperforming ours on such images are specifically designed for images with regular, repeating structures. We made two attempts to improve the results our method achieves on such images: The first relying on a block-matching procedure, with limited success, and the second relying on combining the results of different denoising methods, using an MLP. The second approach is effective and yields results that are almost always better than the best of the inputs. This approach also achieves results that are still closer to reaching theoretical bounds to image denoising.

Training and understanding MLPs: Common criticisms regarding MLPs include the following: (i) It is poorly understood which settings for the training procedure will lead to good results and which to bad results, and (ii) MLPs are often seen as "black boxes" whose internal workings are unknown. We address these issues in Chapter 6. We have made a number of observations regarding which factors are important for achieving good results with MLPs. Using more training data always lead to better results. Using large input patches was also important: Small input patches make the denoising problem ambiguous. Regarding the architecture of the MLP, using more hidden units per hidden layer also always lead to better results. However, there is an ideal number of hidden layers for a given problem and a given number of hidden units per hidden layer. Going above the ideal number of hidden layers can lead to catastrophic degradations in performance. Also, there is an ideal size for the output patch. Going above this ideal size requires increasing the capacity of the MLP. Finally, we have seen that reducing the learning rate in the late stages of the training procedure can lead to important gains in performance. Training an MLP is a time-consuming process, even on today's GPUs, but denoising a single image can be performed in a reasonable amount of time (compared to other denoising algorithms) on CPU.

We were able to make a number of observations regarding the operating principle of MLPs trained for denoising. We saw that the inputs required to maximize the activation of a hidden unit is usually similar to the output caused by the same hidden unit, meaning that MLPs need to detect a feature in the noisy input in order to copy the same feature

into the output. The noise is attenuated via saturation of the last tanh-layer. Interestingly, this mode of operation gives rise to binary representations in the last hidden layer, lending support to the regularization interpretation of denoising autoencoders proposed by Erhan *et al.* [32].

Future research might attempt to use our insights in order to apply MLPs for other image processing or vision tasks, such as image super-resolution, de-mosaicking, or optical flow. A question that remains unanswered is whether a single MLP can be trained to perform well on several noise levels. An additional difficulty would be to denoise an image in which the noise level is unknown. A possible yet inconvenient solution would be to train several MLPs, one for each possible noise level and to have a procedure (either an existing approach or possibly another MLP) predict the noise level and therefore choose which MLP to use to denoise the image. Yet another goal of future research would be to remove noise generated by a real camera using MLPs, possibly incorporating de-mosaicking into the same task. Two approaches are possible: In the first, one models the noise generated by a camera and then simulates the noisy training patches, similarly to the current setup, but with more realistic noise. In the second approach, one could obtain clean and noisy image pairs directly from the camera, for example by changing the lighting conditions as well as the ISO-setting of the camera. As the ISO-setting goes up, one expects higher noise. The second approach has the advantage of treating all sources of noise, including ones that are difficult to model, with the drawback that training data is more difficult to obtain.

Bibliography

[1] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing (TIP)*, 54(11):4311–4322, 2006.

[2] R. Amiri, M. Alaee, H. Rahmani, and M. Firoozmand. Chirplet based denoising of reflected radar signals. In *Proceedings of the Third Asia International Conference on Modelling & Simulation*, AMS '09, pages 304–308. IEEE, 2009.

[3] D.F. Andrews and C.L. Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36(1):99–102, 1974.

[4] E. Arias-Castro and D.L. Donoho. Does median filtering truly preserve edges better than linear filtering? *The Annals of Statistics*, 37(3):1172–1206, 2009.

[5] V. Aurich and J. Weule. Non-linear gaussian filters performing edge preserving diffusion. In *Mustererkennung 1995, 17. DAGM-Symposium*, pages 538–545. Springer-Verlag, 1995.

[6] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[7] Y. Bengio and X. Glorot. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 13, pages 249–256, 2010.

[8] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[9] A. Buades, B. Coll, and J.M. Morel. A non-local algorithm for image denoising. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2005.

[10] A. Buades, C. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

[11] H. C. Burger and S. Harmeling. Improving denoising algorithms via a multi-scale meta-procedure. In *Proceedings of the 33rd international conference on Pattern recognition (DAGM)*, pages 206–215. Springer-Verlag, 2011.

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

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

[14] H.C. Burger, B. Schölkopf, and S. Harmeling. Removing noise from astronomical images using a pixel-specific noise model. In *International Conference on Computational Photography (ICCP)*. IEEE, 2011.

[15] H.C. Burger, C.J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012.

[16] H.C. Burger, C.J. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. *Submitted to the 35th German Conference on Pattern Recognition (GCPR)*, 2013.

[17] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31(4):532–540, 1983.

[18] V. Caselles, G. Sapiro, and D.H. Chung. Vector median filters, inf-sup operations, and coupled pde's: theoretical connections. *Journal of Mathematical Imaging and Vision*, 12(2):109–119, 2000.

[19] A. Chambolle. An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1):89–97, 2004.

[20] A. Chambolle and J. Darbon. On total variation minimization and surface evolution using parametric maximum flows. *International journal of computer vision (IJCV)*, 84(3):288–307, 2009.

[21] S.G. Chang, B. Yu, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE Transactions on Image Processing (TIP)*, 9(9):1532–1546, 2002.

[22] P. Chatterjee and P. Milanfar. Is denoising dead? *IEEE Transactions on Image Processing (TIP)*, 19(4):895–911, 2010.

[23] David J. Coffin. Decoding raw digital photos in Linux. `http://www.cybercom.net/~dcoffin/dcraw/`.

[24] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-D transform-domain collaborative filtering. *IEEE Transactions on Image Processing (TIP)*, 16(8):2080–2095, 2007.

[26] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image restoration by sparse 3D transform-domain collaborative filtering. In *Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 6812, page 6, 2008.

[27] A. Danielyan, V. Katkovnik, and K. Egiazarian. BM3D frames and variational image deblurring. *IEEE Transactions on Image Processing (TIP)*, 21(4):1715–1728, 2012.

[28] L. Demaret, F. Friedrich, H. Fuehr, and T. Szygowski. Multiscale wedgelet denoising algorithms. In *Optics and Photonics*, pages 59140X–59140X. International Society for Optics and Photonics, 2005.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009.

[30] A.A. Efros and T.K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1999.

[31] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing (TIP)*, 15(12):3736–3745, 2006.

[32] D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research (JMLR)*, 11:625–660, 2010.

[33] D. Erhan, A. Courville, and Y. Bengio. Understanding representations learned in deep architectures. Technical report, Technical Report 1355, Université de Montréal/DIRO. (Cited on page 119.), 2010.

[34] F. Estrada, D. Fleet, and A. Jepson. Stochastic image denoising. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009.

[35] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[36] A. Foi. Noise estimation and removal in mr imaging: The variance-stabilization approach. In *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 1809–1814, 2011.

[37] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE Transactions on Image Processing (TIP)*, 16(5):1395–1411, 2007.

[38] K.I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2(3):183–192, 1989.

[39] P. Gehler and M. Welling. Products of "edge-perts". In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[40] G. Gerig, O. Kubler, R. Kikinis, and F.A. Jolesz. Nonlinear anisotropic filtering of mri data. *IEEE Transcations on Medical Imaging*, 11(2):221–232, 1992.

[41] P. Getreuer. Rudin-osher-fatemi total variation denoising using split bregman. *Image Processing On Line (IPOL)*, 2012. `http://dx.doi.org/10.5201/ipol.2012.g-tvd`.

[42] P. Getreuer, M. Tong, and L. Vese. A variational model for the restoration of mr images corrupted by blur and rician noise. *Advances in Visual Computing*, pages 686–698, 2011.

[43] G. Gilboa, Y.Y. Zeevi, and N. Sochen. Texture preserving variational denoising using an adaptive fidelity term. *VLSM*, pages 137–144, 2003.

[44] M. Goesele, W. Heidrich, and H.-P. Seidel. Entropy-based dark frame subtraction. In *Proceedings of PICS: Image Processing, Image Quality, Image Capture, Systems Conference*, 2001.

[45] M. Gomez-Rodriguez, J. Kober, and B. Schölkopf. Denoising photographs using dark frames optimized by quadratic programming. In *IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2009.

[46] H. Gudbjartsson and S. Patz. The rician distribution of noisy mri data. *Magnetic Resonance in Medicine*, 34(6):910, 1995.

[47] J.A. Guerrero-Colón, L. Mancera, and J. Portilla. Image restoration using space-variant gaussian scale mixtures in overcomplete pyramids. *IEEE Transactions on Image Processing (TIP)*, 17(1):27–41, 2008.

[48] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

[49] G. Healey and R. Kondepudy. Radiometric ccd camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(3):267–276, 1994.

[50] G.E. Hinton. Products of experts. In *Ninth International Conference on Artificial Neural Networks (ICANN)*. IET, 1999.

[51] G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[52] G.E. Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9:1, 2010.

[53] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[54] G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[55] M. Hirsch, C.J. Schuler, S. Harmeling, and B. Scholkopf. Fast removal of non-uniform camera shake. In *International Conference on Computer Vision (ICCV)*. IEEE, 2011.

[56] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[57] V. Jain and H.S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[58] J. Jancsary, S. Nowozin, and C. Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *European Conference of Computer Vision (ECCV)*, 2012.

[59] K. Jarrett, K. Kavukcuoglu, M.-A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *International Conference on Computer Vision (ICCV)*. IEEE, 2009.

[60] K. Kavukcuoglu, M.-A. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical Report CBLL-TR-2008-12-01, Computational and Biological Learning Lab, Courant Institute, NYU, 2008.

[61] N. Le Roux and Y. Bengio. Deep belief networks are compact universal approximators. *Neural computation*, 22(8):2192–2207, 2010.

[62] M. Lebrun, M. Colom, A. Buades, and J.M. Morel. Secrets of image denoising cuisine. *Acta Numerica*, 21(1):475–576, 2012.

[63] Y. LeCun, L. Bottou, Y. Bengio, and Haffner P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[64] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.

[65] H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 609–616, 2009.

[66] J.S. Lee. Digital image enhancement and noise filtering by use of local statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2(2):165–168, 1980.

[67] M. Leshno, V.Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[68] A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011.

[69] A. Levin, B. Nadler, F. Durand, and W.T. Freeman. Patch complexity, finite pixel correlations and optimal denoising. In *European Conference on Computer Vision (ECCV)*, 2012.

[70] C. Liu, W.T. Freeman, R. Szeliski, and S.B. Kang. Noise estimation from a single image. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2006.

[71] F. Luisier, T. Blu, and M. Unser. Image denoising in mixed poisson–gaussian noise. *IEEE Transactions on Image Processing (TIP)*, 20(3):696–708, 2011.

[72] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, pages 689–696, 2009.

[73] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *The Journal of Machine Learning Research (JMLR)*, 11:19–60, 2010.

[74] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision (ICCV)*. IEEE, 2010.

[75] J. Mairal, M. Elad, G. Sapiro, et al. Sparse representation for color image restoration. *IEEE Transactions on Image Processing (TIP)*, 17(1):53–69, 2008.

[76] M. Mäkitalo and A. Foi. On the inversion of the anscombe transformation in low-count poisson image denoising. In *International Workshop on Local and Non-Local Approximation in Image Processing (LNLA)*, pages 26–32. IEEE, 2009.

[77] M. Mäkitalo and A. Foi. A closed-form approximation of the exact unbiased inverse of the anscombe variance-stabilizing transformation. *IEEE Transactions on Image Processing (TIP)*, 20(9):2697–2698, 2011.

[78] M. Mäkitalo and A. Foi. Optimal inversion of the anscombe transformation in low-count poisson image denoising. *IEEE Transactions on Image Processing (TIP)*, 20(1):99–109, 2011.

[79] M. Mäkitalo and A. Foi. Poisson-gaussian denoising using the exact unbiased inverse of the generalized anscombe transformation. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012.

[80] M. Mäkitalo and A. Foi. Optimal inversion of the generalized anscombe transformation for poisson-gaussian noise. *IEEE Transactions on Image Processing (TIP)*, 22(1):91–103, 2013.

[81] 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 *International Conference on Computer Vision (ICCV)*. IEEE, 2001.

[82] B. Matalon, M. Elad, and M. Zibulevsky. Improved denoising of images using modelling of a redundant contourlet transform. In *Optics and Photonics*. International Society for Optics and Photonics, 2005.

[83] A. Mittal, A. Moorthy, and A. Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing (TIP)*, 21(12):4695–4708, 2012.

[84] A.K. Moorthy and A.C. Bovik. A two-step framework for constructing blind image quality indices. *IEEE Signal Processing Letters*, 17(5):513–516, 2010.

[85] A.K. Moorthy and A.C. Bovik. Blind image quality assessment: From natural scene statistics to perceptual quality. *IEEE Transactions on Image Processing (TIP)*, 20(12):3350–3364, 2011.

[86] M.K. Ng, L. Qi, Y.F. Yang, and Y.M. Huang. On semismooth newtons methods for total variation minimization. *Journal of Mathematical Imaging and Vision*, 27(3):265–276, 2007.

[87] A. Nosratinia. Enhancement of jpeg-compressed images by re-application of jpeg. *The Journal of VLSI Signal Processing*, 27(1):69–79, 2001.

[88] A. Olmos et al. A biologically inspired algorithm for the recovery of shading and reflectance images. *Perception*, 33(12):1463, 2004.

[89] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Conference Record of The Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, pages 40–44. IEEE, 1993.

[90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(7):629–639, 1990.

[91] A. Pizurica, W. Philips, I. Lemahieu, and M. Acheroy. A joint inter- and intrascale statistical model for bayesian wavelet based image denoising. *IEEE Transactions on Image Processing (TIP)*, 11(5):545–557, 2002.

[92] J. Portilla and E.P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision (IJCV)*, 40(1):49–70, 2000.

[93] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Adaptive wiener denoising using a gaussian scale mixture model in the wavelet domain. In *International Conference on Image Processing (ICIP)*, volume 2, pages 37–40. IEEE, 2001.

[94] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing (TIP)*, 12(11):1338–1351, 2003.

[95] U. Rajashekar and E.P. Simoncelli. Multiscale denoising of photographic images. *The Essential Guide to Image Processing*, pages 241–261, 2009.

[96] M.-A. Ranzato, Y. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 2, pages 371–379, 2007.

[97] M.-A. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[98] J.P. Rossi. Digital techniques for reducing television noise. *SMPTE Journal*, 87(3):134–140, 1978.

[99] S. Roth and M.J. Black. Fields of experts: A framework for learning image priors. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2005.

[100] S. Roth and M.J. Black. Fields of experts. *International Journal of Computer Vision (IJCV)*, 82(2):205–229, 2009.

[101] L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

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

[103] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision (IJCV)*, 77(1):157–173, 2008.

[104] M.A. Saad, A.C. Bovik, and C. Charrier. Blind image quality assessment: A natural scene statistics approach in the dct domain. *IEEE Transactions on Image Processing (TIP)*, 21(8):3339–3352, 2012.

[105] U. Schmidt, Q. Gao, and S. Roth. A generative perspective on mrfs in low-level vision. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010.

[106] M. Schöberl, A. Brückner, S. Foessel, and A. Kaup. Photometric limits for digital camera systems. *Journal of Electronic Imaging*, 21(2):020501–1, 2012.

[107] C.J. Schuler, Burger H.C., S. Harmeling, and B. Schölkopf. A machine learning approach for non-blind image deconvolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013.

[108] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2011.

[109] H.R. Sheikh and A.C. Bovik. Image information and visual quality. *IEEE Transactions on Image Processing (TIP)*, 15(2):430–444, 2006.

[110] H.R. Sheikh, A.C. Bovik, and G. De Veciana. An information fidelity criterion for image quality assessment using natural scene statistics. *IEEE Transactions on Image Processing (TIP)*, 14(12):2117–2128, 2005.

[111] E.P. Simoncelli and E.H. Adelson. Noise removal via bayesian wavelet coring. In *International Conference on Image Processing (ICIP)*, volume 1, pages 379–382. IEEE, 1996.

[112] E.P. Simoncelli and W.T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing (ICIP)*, volume 3, pages 444–447. IEEE, 1995.

[113] J.L. Starck, E.J. Candès, and D.L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing (TIP)*, 11(6):670–684, 2002.

[114] J.L. Starck, F.D. Murtagh, and A. Bijaoui. Image processing and data analysis. *Image Processing and Data Analysis, by Jean-Luc Starck and Fionn D. Murtagh and Albert Bijaoui. Cambridge, UK: Cambridge University Press*, 1998.

[115] H. Talbot, H. Phelippeau, M. Akil, and S. Bara. Efficient poisson denoising for photography. In *International Conference on Image Processing (ICIP)*, volume 1, pages 3881–3884. IEEE, 2009.

[116] H. Tang, N. Joshi, and A. Kapoor. Learning a blind measure of perceptual image quality. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011.

[117] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1998.

[118] J.A. Tropp, A.C. Gilbert, and M.J. Strauss. Algorithms for simultaneous sparse approximation. part i: Greedy pursuit. *Signal Processing*, 86(3):572–588, 2006.

[119] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of ccd imaging process. In *International Conference on Computer Vision (ICCV)*. IEEE, 2001.

[120] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research (JMLR)*, 11:3371–3408, 2010.

[121] M. J. Wainwright and E. P. Simoncelli. Scale mixtures of gaussians and the statistics of natural images. In *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[122] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[123] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing (TIP)*, 13(4):600–612, 2004.

[124] Z. Wang and Q. Li. Information content weighting for perceptual image quality assessment. *IEEE Transactions on Image Processing (TIP)*, 20(5):1185–1198, 2011.

[125] Z. Wang, E.P. Simoncelli, and A.C. Bovik. Multiscale structural similarity for image quality assessment. In *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1398–1402. IEEE, 2003.

[126] J. Weickert. *Anisotropic diffusion in image processing*. ECMI Series, Teubner-Verlag, Stuttgart, Germany, 1998.

[127] Y. Weiss and W.T. Freeman. What makes a good model of natural images? In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007.

[128] R. Widenhorn, M.M. Blouke, A. Weber, A. Rest, and E. Bodegom. Temperature dependence of dark current in a ccd. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 4669, pages 193–201, 2002.

[129] L.P. Yaroslavsky. Digital picture processing: An introduction, 1985.

[130] P. Ye and D. Doermann. No-reference image quality assessment using visual codebooks. *IEEE Transactions on Image Processing (TIP)*, 21(7):3129–3138, 2012.

[131] S. Zhang and E. Salari. Image denoising using a neural network based non-linear filter in wavelet domain. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2005.

[132] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision (ICCV)*. IEEE, 2011.