Robust and Interpretable Denoising via Deep Learning

by

Sreyas Mohan

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Center For Data Science

New York University

May, 2022

Dr. Carlos Fernandez-Granda

Dr. Eero P Simoncelli

To my Amma and Achan

Acknowledgements

When I started my Ph.D., I used to imagine what it would be like to go through this journey. A few years later, now writing this acknowledgement, I can confidently say that (much like research) all my predictions were incorrect - there were surprises everywhere. Most instances of success I had, and most challenges I faced, were all quite unexpected. In spite of this, the last few years were one of the most enjoyable times I ever had. This is because of the people who helped, supported and cheered me all throughout. Thank you!

I am very grateful to Carlos Fernandez-Granda and Eero Simoncelli for advising and supporting me throughout my Ph.D. Thank you for teaching me to ask the right questions, nudging me out of my comfort zone to explore new topics, pushing me to work harder, and finally, for all the candid conversations that have changed my outlook towards research, and life. This thesis is a product of our joint work, and it was an absolute pleasure and honor working with Carlos and Eero.

I would like to thank Joan Bruna, Kyunghyun Cho, Bill Freeman and Yann LecCun for being on my thesis committee. I'm grateful for the conversations I have had with Joan, and for the feedback I received from Yann during the early stages of research. I met Bill for the first time during a conference, and since then he has continued to inspire me with his research, as well as kindness. I am thankful to Kyunghyun for constantly providing advice on my research and career throughout my Ph.D., and for introducing me to the world of "authentic" Korean Fried Chicken. In a post-pandemic world, my defense was the first hybrid presentation I ever did, and

honestly, it was a bit of a struggle. It is remarkable that my committee made my private defense so productive and enjoyable, and their feedback gave me new ideas to think about, and certainly improved my thesis.

None of the work presented in the thesis would have been possible if not for my wonderful collaborators and mentors. I would like to thank Joshua, Peter and Ramon from ASU who captured the microscopy data used in Chapter 5. The frustration of not being able to do well on this dataset inspired us to come up with the algorithms developed in Chapters 3 and 4. I am also grateful to my other collaborators: Zahra for Chapter 2, Dev and Mitesh for Chapter 3 and Binh and David for Chapter 5. I am fortunate to have worked on a diverse set of projects during the course of my Ph.D., and grateful to the people who made that possible: Aakash, Adria, Alejandra, Ananya, Boyang, Chaitra ,Gautier, George, Jon, Jose, Kangning, Matan and Weicheng. While these projects did not make it to my thesis, I enjoyed exploring different topics, learning new things, and ultimately, they definitely had an impact in shaping my thesis. I would also like to acknowledge our admin staff: Kathryn and Tim at NYU CDS for making sure that I was always in compliance with my degree requirements, Matthew and Serena for taking care of me while at Flatiron, and Shenglong at NYU HPC for helping me with his wizardry linux knowledge!

Outside NYU, I had a fun and productive summer at Google - I am grateful to Aamir and Yeping for making this happen, and for being wonderful hosts. I would also like to express my sincere thanks to Mitya and Cengiz for hosting me at Flatiron institute in the final summer of undergrad, which ultimately convinced me to pursue a Ph.D. and to relocate to NYC.

My time at NYU was made more enjoyable through the company of my fellow graduate students and friends. They are too many to name, but I want to make an effort to do so - after all, if you still find any typos in this thesis, they are to be blamed! I would like to acknowledge all members of Eero and Carlos' group, and the greater machine learning community at NYU for the many interesting interactions I have had along the way: Aahlad, Aishwarya, Ben, Caroline, Chris, Collin, Hope, Ilia, Irina, Jing, Katrina, Lyndon, Mark, Nan, Nikhil, Pierre, Roberta, Swap-

neel, Teddy, Vlad, and Zhouhan. I have also been forunate to have a close group of friends outside the New York area: Divya, Green, Pranoy, and Prasan. On a personal note, I want to mention Aakash, Harvineet, and Prasan for our conversations about life over chai and samosa; Brett and Matan for solving obscure math and programming puzzles with me at random times of the day; Roshan and Pranav for our annual hiking trips; Phu, Ren and Xintian for fun conversations, periodic get together, and board games; and Mandy for helping me all throughout during the later part of my dissertation and introducing me to Maqiu!

And ofcourse, there are my parents and sister to whom I owe everything. Thank you for being patient all throughout, and continuously pestering me asking when I will graduate. I hope you have your answer now!

Abstract

In the past decade, convolutional neural networks (CNN) have achieved state-of-the-art results in denoising. The goal of this work is to advance our understanding of these models and leverage this understanding to advance the current state-of-the-art. We start by showing that CNNs systematically overfit the noise levels in the training set, and propose a new architecture called *bias-free* CNNs which generalize robustly to noise levels outside the training set. Bias-free networks are also locally linear, which enables direct analysis with linear-algebraic tools. We show that the denoising map can be visualized locally as a filter that adapts to both signal structure and noise level. Denoising CNNs including bias-free CNNs are typically trained using pairs of noisy and clean data. However, in many domains like microscopy, clean data is generally not available. We develop a network architecture that performs *unsupervised denoising* for video data, i.e, we train only using noisy videos. We then build on top of the unsupervised denoising methodology and propose a new adaptive denoising paradigm. We develop *GainTuning* in which CNN models pre-trained on large datasets are adaptively and selectively adjusted for individual test images. GainTuning improves state-of-the-art CNNs on standard image-denoising benchmarks, particularly for test images differing systematically from the training data, either in noise distribution or image type. Finally, we explore the application of deep learning-based denoising in scientific discovery through a case study in electron microscopy. To ensure that the denoised output is accurate, we develop likelihood map which quantifies the agreement between real noisy data and denoised output (thus flagging denoising artifacts). In addition, we show that popular

metrics for denoising fail to capture scientifically relevant details and propose new metrics to fill this gap.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Our measurement devices are often imperfect. The microphone of our cell phone picks up background sounds, and the images captured from our camera are corrupted by random voltage fluctuations in the sensors. As in these examples, degradation with noise is a common form of corruption that occurs in our measurement devices. Reducing the noise and recovering the signal of interest from corrupted measurements is called *denoising* (See Fig 1.1 for a visual example). Achieving high-quality denoising requires (at least implicitly) quantifying and exploiting the differences between signals and noise. In the case of photographic images, the denoising problem is both an important application, as well as a useful test-bed for our understanding of natural images.

General methods for image processing are an active area of research. In the past decade, a technique called convolutional neural networks (CNN) [72] (introduced in section 1.4), has emerged as the defacto standard to deal with image data. CNN based denoisers implement a denoising strategy that is completely data driven. The denoiser is shown thousands of pairs of noisy and clean images which it uses to *learn* a strategy to denoise a previously unseen noisy image. This methodology achieves state-of-the-art results in denoising natural images [21, 150]. However, despite their success, we face several challenges while using CNN based denoisers for image denoising in practise. How can we train these methods on real noisy data when clean

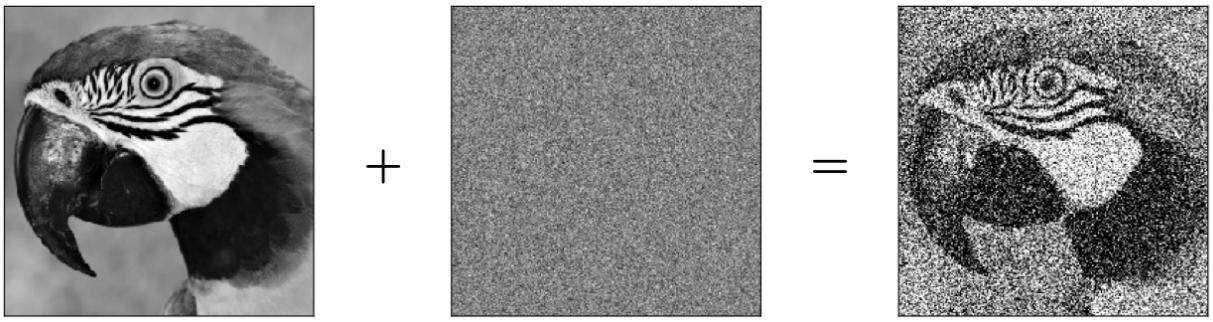


Figure 1.1: Visual example of denosing problem. The goal of denoising problem is to estimate the clean image from observed noisy image. In this example, the observed noisy image (right) is a sum of a clean image (left) with independent Gaussian noise (middle).

image are not available? Can these methods be used when the imaging conditions change from training to inference? Can we understand and interpret what strategies these denoisers learn? These challenges are outlined in Section 1.5. We will explore potential solutions in rest of the thesis.

1.2 The Denoising Problem

In this section we describe the denoising problem. Consider a signal $x \in \mathbb{R}^N$ (for example, an image with N pixels). The measurement $y \in \mathbb{R}^N$ that we observe is our signal x corrupted with noise. A common model used is additive Gaussian noise, and in this case the observation y can be written as function of noise $\eta \in \mathbb{R}^N$ as:

$$y = x + \eta, \text{ where } \eta \sim \mathcal{N}(0, \sigma^2 I_N), \tag{1.1}$$

where I_N is an $N \times N$ identity matrix and $\sigma^2 > 0$ denotes the variance of the each dimension of the Gaussian random variable (refer to figure 1.1 for an illustration of the corruption model). In addition to Gaussian noise, we will explore Poisson noise in Chapter 5, and real noise from image acquisition systems in Chapters 3, 4, and 5.

Solving the denoising problem involves finding a function $f : \mathbb{R}^N \to \mathbb{R}^N$ such that a noisy observation y can be mapped to a good estimate of x, i.e $f(y) \approx x$. We can assume that signal x and noisy observation y are realizations of a random variable $\mathcal{X}$ and $\mathcal{Y}$ respectively. A common choice to obtain denoising function f is to minimize the squared error:

$$f_{\text{opt}} := \arg\min_f \mathbb{E}_{\mathcal{X}\times\mathcal{Y}} ||x - f(y)||_2^2, \tag{1.2}$$

where the expectation $\mathbb{E}$ is taken over the joint distribution of clean and noisy images $\mathcal{X}\times\mathcal{Y}$. If we do not make assumptions on the distribution of $\mathcal{X}$, then we approximate the expectation in Equation 1.2 by the empirical expectation over a dataset of noisy and clean image pairs $\{(y_i, x_i)\}_{i=1}^n$:

$$\hat{f}_{\text{opt}} := \arg\min_f \frac{1}{n} \sum_{i=1}^{n} ||x_i - f(y_i)||_2^2, \tag{1.3}$$

where n is the number of examples in our dataset.

1.3 Classical Methods

Perhaps the simplest solution for the denoising problem is the Wiener filter. The Wiener filter assumes that the signal $\mathcal{X}$ is translation invariant and follows a Gaussian distribution [140]. In this situation, the Wiener filter is the optimal estimator in terms of mean squared error. It operates by mapping the noisy image to the frequency domain, shrinking the amplitude of all components, and mapping back to the signal domain. In the case of natural images, the high-frequency components are shrunk more aggressively than the lower-frequency components because they tend to contain less energy. This is equivalent to convolution with a lowpass filter, implying that each pixel is replaced with a weighted average over a local neighborhood. This local averaging results in over-smooths resulting leading the elimination of fine scale details and textures. Modern filtering approaches address this issue by adapting the filters to the local structure of the noisy

image (e.g. [92, 131]). In Section 2.5 we show that neural networks implement such strategies implicitly, learning them directly from the data.

In the 1990's powerful denoising techniques were developed based on multi-scale ("wavelet") transforms. These transforms map natural images to a domain where they have sparser representations. This makes it possible to perform denoising by applying nonlinear thresholding operations in order to discard components that are small relative to the noise level [19, 31, 120]. From a linear-algebraic perspective, these algorithms operate by projecting the noisy input onto a lower-dimensional subspace that contains plausible signal content. The projection eliminates the orthogonal complement of the subspace, which mostly contains noise. This general methodology laid the foundations for the state-of-the-art models in the 2000's (e.g. [26]), some of which added a data-driven perspective, learning sparsifying transforms [35], and nonlinear shrinkage functions [48, 114], directly from natural images. Ref. [96] shows that deep-learning models learn similar priors in the form of local linear subspaces capturing image features.

1.4 Convolutional Neural Network Based Denoising

For a noisy input image $y \in \mathbb{R}^N$ with N pixels, the denoising function $f : \mathbb{R}^N \to \mathbb{R}^N$ computed by a neural network is

$$f(y) = W_L R(W_{L-1} \ldots R(W_1 y + b_1) + \ldots + b_{L-1}) + b_L, \tag{1.4}$$

where W_i is called the weight matrix and b_i is bias vectors at layer i. The function $R(z) = \max(0, z)$ represents a rectified linear unit (ReLU) non-linearity [41]. ReLU is applied element wise to any tensor input. If the neural network is convolutional, W_is are restricted to be convolutional matrices. The process of finding the optimal parameters W_is and b_is by solving the optimization problem in equation 1.3 is called *training*. Neural networks that are trained on large databases of

noisy and clean natural images to minimize mean square error achieves current state-of-the-art denoising performance [52, 116, 150, 151].

When training a neural network, we need real data to approximate the expectation in Equation 1.2 and arrive at Equation 1.3. Since the expectation in Equation 1.2 is over the joint distribution of $\mathcal{X} \times \mathcal{Y}$ we needs pairs of noisy and clean images to approximate the expectation. While we have datasets with clean images [87], we have very limited data on pairs of noisy and clean images. To overcome this obstacle, a common technique used is to add synthetic additive Gaussian noise (Equation 1.1) to clean images from an available dataset, generating pairs of noise and clean images for training. However, this form of training introduces certain practical challenges. For example, in several real-world applications we only have the noisy images without access to a noise model or corresponding clean images. Can we use a CNN to denoise in this situation? If we train on a particular dataset, can we expect the CNN to generalize when the image acquisition conditions change? We outline these challenges in the next section.

1.5 Challenges in Denoising and Thesis Outline

While applying deep learning based denoising in practical situations, we encounter several challenges. We will explore three of them in this thesis:

- **We need *robustness* to changes in imaging conditions**

  Current denoising systems achieve state-of-the-art performance when the test data is similar to the training data, but their performance degrades significantly when applied to data that deviates from the training distribution. We explore a specific phenomena related to changing noise distributions in **Chapter 2**. We show that state-of-the-art CNNs trained for Gaussian denoising overfit to the noise levels (or standard deviations) they saw during training, and fails catastrophically when tested on unseen noise levels. We propose a new architecture, called *Bias-Free* CNNs that generalize seamlessly to unseen noise levels.

In **Chapter 4**, we study a more general version of the robustness problem. We introduce *GainTuning*, a framework that adapts *any* pre-trained CNN denoiser to out-of-distribution test data. GainTuning improves denoising performance significantly for test images differing systematically from the training data, either in noise level or image type. We also illustrate the potential of adaptive denoising in a scientific application, in which a CNN is trained on synthetic data, and tested on real transmission-electron-microscope images (explained in Chapter 5). In contrast to the existing methodology, GainTuning is able to faithfully reconstruct the structure of catalytic nanoparticles from these data at extremely low signal-to-noise ratios.

- **We need *interpretability* to understand how the models work and adapt them.**
  Despite the success of the CNN based denoising systems, we lack both intuition and a formal understanding of the mechanisms they implement to achieve this impressive performance. In **Chapter 2**, we develop a gradient based tool to visualize and understand the denoising mechanisms learned by the deep CNNs. We show that CNNs trained for image denoising performs adaptive averaging over nearby pixels. Further, in **Chapter 3**, we apply this tool to analyze video denoisers and show that deep CNNs trained for video-denoising performs adpative spatio-temporal averaging consistent with local motion in the input videos. This analysis reveals that the network learns to perform implicit motion compensation, even though it is only trained for video denoising.

- **We have *no ground-truth data* to train the network**
  Deep convolutional neural networks (CNNs) for denoising are typically trained with supervision, assuming the availability of clean data. However, in many applications, such as microscopy, noiseless data are not available. To address this for video denoising, in **Chapter 3**, we propose an Unsupervised Deep Video Denoiser (UDVD). UDVD is a CNN architecture designed to be trained exclusively on noisy data. The performance of UDVD

is comparable to the supervised state-of-the-art, even when trained only on a single short noisy video. We demonstrate the promise of our approach in real-world imaging applications by denoising raw video, fluorescence-microscopy and electron-microscopy data.

In addition to photographic images, denoising is also a fundamental challenge in scientific imaging. While deep CNNs provide the current state of the art in denoising natural images and produce impressive results, their potential has barely been explored in the context of scientific imaging. We explore the application of deep learning based denoising in scientific discovery through a **case study in electron microscopy**. In **Chapter 5**, we apply the methodology developed in Chapters 2, 3 and 4 to images of catalytic nanopaticles acquired through a transmission electron microscope. In scientific imaging, it is important to ensure that the denoised output is factual, and not hallucinated (for example, there could be phantom atoms in the denoised output). In Chapter 5, we develop *Likelihood Map* to aid with this task. Likelihood map quantifies the agreement between real data and the denoised output, thus flagging parts of the image that are denoising artefacts. In addition, we show that popular metrics for denoising do not capture scientifically relevant details, and develop new metrics to fill this gap. Finally, we summarize and conclude the thesis in **Chapter 6**.

2 | Bias-free denoising: Generalization to unseen noise variance

This chapter is adapted from the paper "Robust and Interpretable Denoising Via Bias-Free Convolutional Neural Networks" published in International Conference of Learning Representations (ICLR) 2021 [96]. This is a joint work with Zahra Kadkhodaie, Eero P Simoncelli and Carlos Fernendez-Granda.

ABSTRACT

We study the generalization properties of deep convolutional neural networks for image denoising in the presence of varying noise levels. We provide extensive empirical evidence that current state-of-the-art architectures systematically overfit to the noise levels in the training set, performing very poorly at new noise levels. We show that strong generalization can be achieved through a simple architectural modification: removing all additive constants. The resulting "bias-free" networks attain state-of-the-art performance over a broad range of noise levels, even when trained over a narrow range. They are also locally linear, which enables direct analysis with gradient-based tools. We show that the denoising map can be visualized locally as a filter that adapts to both image structure and noise level.

2.1 Overview

In the past decade, convolutional neural networks [72] have achieved state-of-the-art results in image denoising [21, 150]. Despite their success, these solutions are mysterious: we lack both intuition and formal understanding of the mechanisms they implement. Network architecture and functional units are often borrowed from the image-recognition literature, and it is unclear which of these aspects contributes to, or limits, the denoising performance. The goal of this chapter is advance our understanding of deep-learning models for denoising. The contributions of this chapter are twofold: First, we study the generalization capabilities of deep-learning models across different noise levels. Second, we provide novel tools for analyzing the mechanisms implemented by neural networks to denoise natural images.

An important advantage of deep-learning techniques over traditional methodology is that a single neural network can be trained to perform denoising at a wide range of noise levels. Currently, this is achieved by simulating the whole range of noise levels during training [150]. Here, we show that this is not necessary. Neural networks can be made to *generalize automatically across noise levels* through a simple modification in the architecture: removing all additive constants. We find this holds for a variety of network architectures proposed in previous literature. We provide extensive empirical evidence that the main state-of-the-art denoising architectures systematically overfit to the noise levels in the training set, and that this is due to the presence of a net bias. Suppressing this bias makes it possible to attain state-of-the-art performance while training over a very limited range of noise levels.

The data-driven mechanisms implemented by deep neural networks to perform denoising are *almost completely unknown*. It is unclear what priors are being learned by the models, and how they are affected by the choice of architecture and training strategies. Here, we provide novel tools to visualize and interpret these strategies through a local analysis of the Jacobian of the denoising map. The analysis reveals locally adaptive properties of the learned models, akin to

existing nonlinear filtering algorithms.

2.2 Network Bias Impairs Generalization

Similar to the setup described in Section **??**, we assume a measurement model in which images are corrupted by additive noise: $y = x + n$, where $x \in \mathbb{R}^N$ is the original image, containing N pixels, n is an image of i.i.d. samples of Gaussian noise with variance σ^2, and y is the noisy observation. The denoising problem consists of finding a function $f : \mathbb{R}^N \to \mathbb{R}^N$, that provides a good estimate of the original image, x. Commonly, one minimizes the mean squared error : $f = \arg\min_g E||x - g(y)||^2$, where the expectation is taken over some distribution over images, x, as well as over the distribution of noise realizations. In deep learning, the denoising function g is parameterized by the weights of the network, so the optimization is over these parameters. If the noise standard deviation, σ, is unknown, the expectation must also be taken over a distribution of σ. This problem is often called *blind denoising* in the literature. In this chapter, we study the generalization performance of CNNs *across* noise levels σ, i.e. when they are tested on noise levels not included in the training set.

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

$$f(y) = W_L R(W_{L-1} ... R(W_1 y + b_1) + ... b_{L-1}) + b_L = A_y y + b_y, \tag{2.1}$$

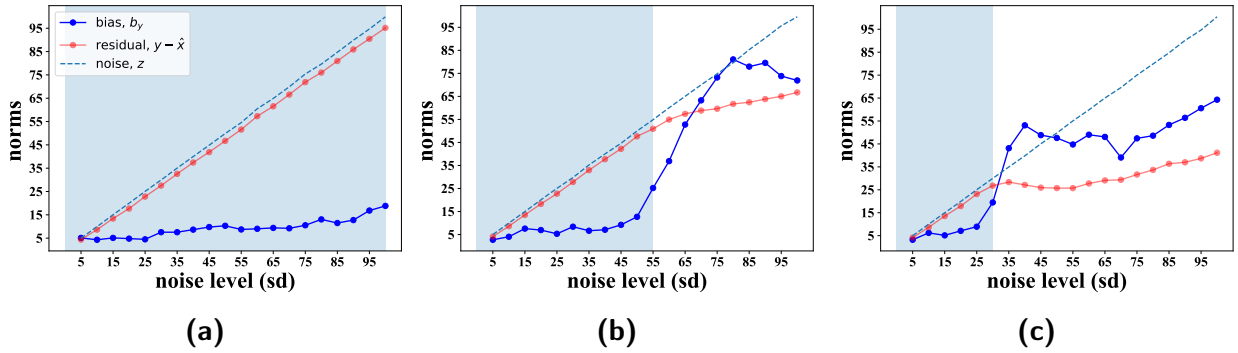


Figure 2.1: First-order analysis of the residual of a denoising convolutional neural network as a function of noise level. The plots show the norms of the residual and the net bias averaged over 100 20×20 natural-image patches for networks trained over different training ranges. The range of noises used for training is highlighted in blue. **(a)** When the network is trained over the full range of noise levels ($\sigma \in [0, 100]$) the net bias is small, growing slightly as the noise increases. **(b-c)** When the network is trained over the a smaller range ($\sigma \in [0, 55]$ and $\sigma \in [0, 30]$), the net bias grows explosively for noise levels beyond the training range. This coincides with a dramatic drop in performance, reflected in the difference between the magnitudes of the residual and the true noise. The CNN used for this example is DnCNN [150]; using alternative architectures yields similar results as shown in Figure A.1.

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

Based on Equation 2.1 we can perform a first-order decomposition of the error or *residual* of the neural network for a specific input: $y - f(y) = (I - A_y)y - b_y$. Figure 2.1 shows the magnitude of the residual and the constant, which is equal to the net bias b_y, for a range of noise levels. Over the training range, the net bias is small, implying that the linear term is responsible for most of the denoising (see Figures A.2 and A.3 for a visualization of both components). However, when the network is evaluated at noise levels outside of the training range, the norm of the bias increases dramatically, and the residual is significantly smaller than the noise, suggesting a form of overfitting. Indeed, network performance generalizes very poorly to noise levels outside the training range. This is illustrated for an example image in Figure 2.2, and demonstrated through extensive experiments in Section 2.4.

2.3 Proposed Methodology: Bias-Free Networks

Section 2.2 shows that CNNs overfit to the noise levels present in the training set, and that this is associated with wild fluctuations of the net bias b_y. This suggests that the overfitting might be ameliorated by removing additive (bias) terms from every stage of the network, resulting in a *bias-free* CNN (BF-CNN). Note that bias terms are also removed from the batch-normalization used during training. This simple change in the architecture has an interesting consequence. If the CNN has ReLU activations the denoising map is locally homogeneous, and consequently *invariant to scaling*: rescaling the input by a constant value simply rescales the output by the same amount, just as it would for a linear system.

Lemma 2.1. *Let $f_{\mathrm{BF}} : \mathbb{R}^N \to \mathbb{R}^N$ be a feedforward neural network with ReLU activation functions and no additive constant terms in any layer. For any input $y \in \mathbb{R}$ and any nonnegative constant α,*

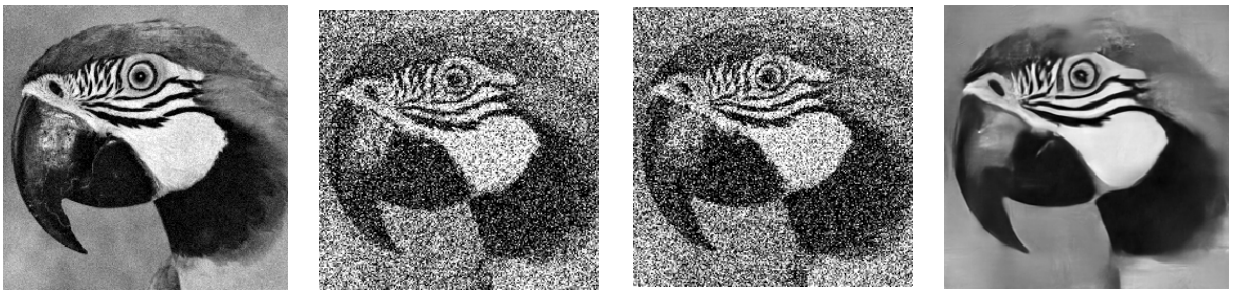
$$f_{\mathrm{BF}}(\alpha y) = \alpha f_{\mathrm{BF}}(y). \tag{2.2}$$

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

$$f_{\mathrm{BF}}(\alpha y) = W_L \mathcal{R}(W_{L-1} \cdots \mathcal{R}(W_1 \alpha y)) = \alpha W_L \mathcal{R}(W_{L-1} \cdots \mathcal{R}(W_1 y)) = \alpha f_{\mathrm{BF}}(y). \tag{2.3}$$

□

Note that networks with nonzero net bias are not scaling invariant because scaling the input may change the activation pattern of the ReLUs. Scaling invariance is intuitively desireable for



Noisy training image, $\sigma = 10$ (max level) | Noisy test image, $\sigma = 90$ | Test image, denoised by CNN | Test image, denoised by BF-CNN

Figure 2.2: Denoising of an example natural image by a CNN and its bias-free counterpart (BF-CNN), both trained over noise levels in the range $\sigma \in [0, 10]$ (image intensities are in the range $[0, 255]$). The CNN performs poorly at high noise levels ($\sigma = 90$, far beyond the training range), whereas BF-CNN performs at state-of-the-art levels. The CNN used for this example is DnCNN [150]; using alternative architectures yields similar results (see Section 2.4).

a denoising method operating on natural images; a rescaled image is still an image. Note that Lemma 2.1 holds for networks with skip connections where the feature maps are concatenated or added, because both of these operations are linear.

In the following sections we demonstrate that removing all additive terms in CNN architectures has two important consequences: (1) the networks gain the ability to generalize to noise levels not encountered during training (as illustrated by Figure 2.2 the improvement is striking), and (2) the denoising mechanism can be analyzed locally via gradient-based tools that reveal intriguing ties to more traditional denoising methodology such as nonlinear filtering.

2.4 Bias-Free Networks Generalize Across Noise Levels

In order to evaluate the effect of removing the net bias in denoising CNNs, we compare several state-of-the-art architectures to their bias-free counterparts, which are exactly the same except for the absence of any additive constants within the networks (note that this includes the batch-normalization additive parameter). These architectures include popular features of existing neural-network techniques in image processing: recurrence, multiscale filters, and skip connec-

tions. More specifically, we examine the following models (see Section A.1 for additional details):

- DnCNN [150]: A feedforward CNN with 20 convolutional layers, each consisting of 3×3 filters, 64 channels, batch normalization [54], a ReLU nonlinearity, and a skip connection from the initial layer to the final layer.
- Recurrent CNN: A recurrent architecture inspired by [151] where the basic module is a CNN with 5 layers, 3×3 filters and 64 channels in the intermediate layers. The order of the recurrence is 4.
- UNet [116]: A multiscale architecture with 9 convolutional layers and skip connections between the different scales.
- Simplified DenseNet: CNN with skip connections inspired by the DenseNet architecture [52, 153].

We train each network to denoise images corrupted by i.i.d. Gaussian noise over a range of standard deviations (the *training range* of the network). We then evaluate the network for noise levels that are both within and beyond the training range. Our experiments are carried out on 180×180 natural images from the Berkeley Segmentation Dataset [87] to be consistent with previous results [21, 117, 150]. Additional details about the dataset and training procedure are provided in Section A.2.

Figures 2.3, A.4 and A.5 show our results. For a wide range of different training ranges, and for all architectures, we observe the same phenomenon: the performance of CNNs is good over the training range, but degrades dramatically at new noise levels; in stark contrast, the corresponding BF-CNNs provide strong denoising performance over noise levels outside the training range. This holds for both PSNR and the more perceptually-meaningful Structural Similarity Index [**wang2004ssim**] (see Figure A.5). Figure 2.2 shows an example image, demonstrating visually the striking difference in generalization performance between a CNN and its corresponding

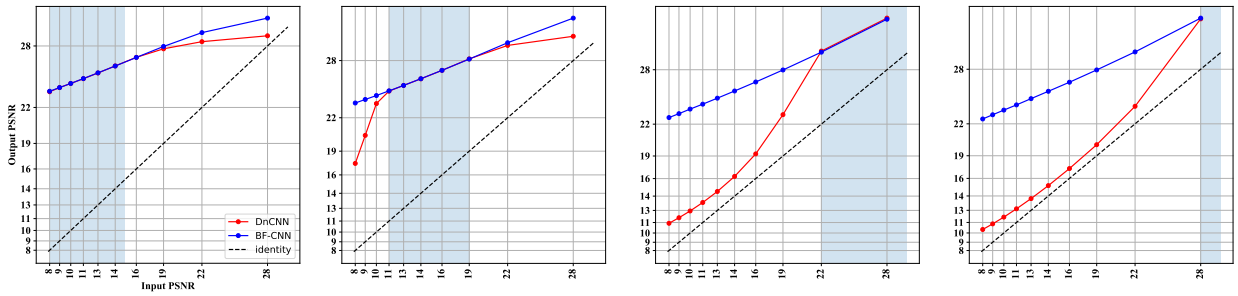


Figure 2.3: Comparison of the performance of a CNN and a BF-CNN with the same architecture for the experimental design described in Section 2.4. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR. Both networks are trained over a fixed ranges of noise levels indicated by a blue background. In all cases, the performance of BF-CNN generalizes robustly beyond the training range, while that of the CNN degrades significantly. The CNN used for this example is DnCNN [150]; using alternative architectures yields similar results (see Figures A.4 and A.5).

BF-CNN. Our results provide strong evidence that removing net bias in CNN architectures results in effective generalization to noise levels out of the training range.

2.5 Revealing the Denoising Mechanisms Learned by BF-CNNs

In this section we perform a local analysis of BF-CNN networks, which reveals the underlying denoising mechanisms learned from the data. A bias-free network is strictly linear, and its net action can be expressed as

$$f_{\mathrm{BF}}(y) = W_L R(W_{L-1} \ldots R(W_1 y)) = A_y y, \tag{2.4}$$

where A_y is the Jacobian of $f_{\mathrm{BF}}(\cdot)$ evaluated at y. The Jacobian at a fixed input provides a local characterization of the denoising map. In order to study the map we perform an analysis of the Jacobian. Our approach is similar in spirit to visualization approaches– proposed in the context of image classification– that differentiate neural-network functions with respect to their input

(e.g. [97, 121]).

The linear representation of the denoising map given byEquation 3.3 implies that the ith pixel of the output image is computed as an inner product between the ith row of A_y, denoted $a_y(i)$, and the input image:

$$f_{\mathrm{BF}}(y)(i) = \sum_{j=1}^{N} A_y(i,j)y(j) = a_y(i)^T y. \tag{2.5}$$

The vectors $a_y(i)$ can be interpreted as *adaptive filters* that produce an estimate of the denoised pixel via a weighted average of noisy pixels. Examination of these filters reveals their diversity, and their relationship to the underlying image content: they are adapted to the local features of the noisy image, averaging over homogeneous regions of the image without blurring across edges. This is shown for two separate examples and a range of noise levels in Figures 2.4, A.6, A.7 and A.8 for the architectures described in Section 2.4. We observe that the equivalent filters of all architectures adapt to image structure.

Classical Wiener filtering [140] denoises images by computing a local average dependent on the noise level. As the noise level increases, the averaging is carried out over a larger region. As illustrated by Figures 2.4, A.6, A.7 and A.8, the equivalent filters of BF-CNNs also display this behavior. The crucial difference is that the filters are adaptive. The BF-CNNs learn such filters implicitly from the data, in the spirit of modern nonlinear spatially-varying filtering techniques designed to preserve fine-scale details such as edges (e.g. [131], see also [92] for a comprehensive review, and [22] for a recent learning-based approach).

2.6 Discussion

In this chapter, we show that removing constant terms from CNN architectures ensures strong generalization across noise levels, and also provides interpretability of the denoising method via

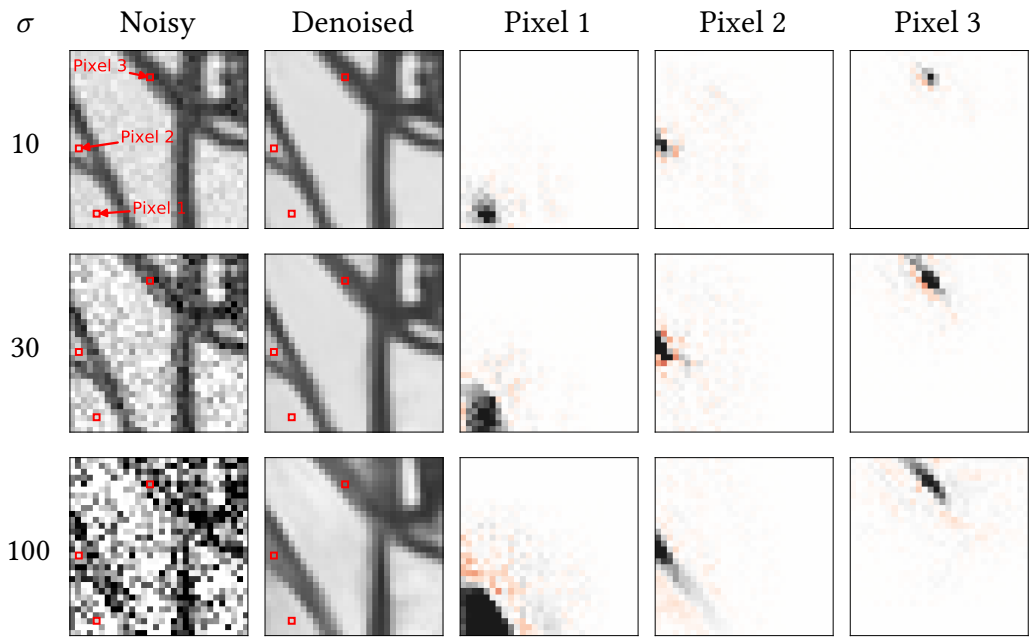


Figure 2.4: Visualization of the linear weighting functions (rows of A_y in Equation 3.3) of a BF-CNN for three example pixels of an input image, and three levels of noise. The images in the three rightmost columns show the weighting functions used to compute each of the indicated pixels (red squares). All weighting functions sum to one, and thus compute a local average (note that some weights are negative, indicated in red). Their shapes vary substantially, and are adapted to the underlying image content. As the noise level σ increases, the spatial extent of the weight functions increases in order to average out the noise, while respecting boundaries between different regions in the image, which results in dramatically different functions for each pixel. The CNN used for this example is DnCNN [150]; using alternative architectures yields similar results (see Figure A.6).

linear-algebra techniques. We provide insights into the relationship between bias and generalization through a set of observations. Theoretically, we argue that if the denoising network operates by projecting the noisy observation onto a linear space of "clean" images, then that space should include all rescalings of those images, and thus, the origin. This property can be guaranteed by eliminating bias from the network. Empirically, in networks that allow bias, the net bias of the trained network is quite small within the training range. However, outside the training range the net bias grows dramatically resulting in poor performance, which suggests that the bias may be the cause of the failure to generalize. In addition, when we remove bias from the architecture, we preserve performance within the training range, but achieve near-perfect generalization, even to

noise levels more than 10x those in the training range. These observations do not fully elucidate how our network achieves its remarkable generalization- only that bias prevents that generalization, and its removal allows it.

Finally, our analysis uncovers interesting aspects of the denoising map, but these interpretations are very local: small changes in the input image change the activation patterns of the network, resulting in a change in the corresponding linear mapping. Extending the analysis to reveal global characteristics of the neural-network functionality is a challenging direction for future research.

The training scheme we used in chapter was supervised - it required access to ground-truth data to enable learning. However, in many practical situations like microscopy and astronomy, ground-truth clean data is often difficult or impossible to obtain. In the next chapter, we will explore techniques to train deep CNNs for denoising without ground-truth clean data.

3 | Unsupervised denoising: Learning without ground truth data

This chapter is adapted from the paper "Unsupervised Deep Video Denoising" published in International Conference of Computer Vision (ICCV) 2021 [119]. This is a joint work with Dev Sheth, Joshua Vincent, Ramon Manzorro, Peter A Crozier, Mitesh Khapra, Eero P Simoncelli and Carlos Fernandez-Granda. In Chapters 1 and 2, we discussed state-of-the-art deep CNNs for image denoising, and introduced bias-free networks which generalize seamlessly to noise levels not seen during training. These networks were, however, trained using clean images as target. In this chapter, we will explore unsuperivised denoising or training CNNs for denoising *only* with noisy data. We will review unsupervised denoising methods for static images, and develop a new method for sequence of images or videos.

ABSTRACT

Deep convolutional neural networks (CNNs) for video denoising are typically trained with supervision, assuming the availability of clean videos. However, in many applications, such as microscopy, noiseless videos are not available. To address this, we propose an Unsupervised Deep Video Denoiser (UDVD), a CNN architecture designed to be trained exclusively with noisy data. The performance of UDVD is comparable to the supervised state-of-the-art, even when trained

only on a single short noisy video. We demonstrate the promise of our approach in real-world imaging applications by denoising raw video, fluorescence-microscopy and electron-microscopy data. In contrast to many current approaches to video denoising, UDVD does not require explicit motion compensation. This is advantageous because motion compensation is computationally expensive, and can be unreliable when the input data are noisy. A gradient-based analysis reveals that UDVD automatically adapts to local motion in the input noisy videos. Thus, the network learns to perform implicit motion compensation, even though it is only trained for denoising.

3.1 Overview

Video denoising is a fundamental problem in image processing, as well as an important pre-processing step for computer vision tasks. Convolutional neural networks (CNNs) [72] provide current state-of-the-art solutions for this problem [23, 27, 29, 34, 129, 130, 143, 147]. These networks are typically trained using a database of clean videos, which are corrupted with simulated noise. However, in applications such as microscopy, noiseless ground truth videos are often not available. To address this issue, we propose a method to train a video denoising CNN without access to supervised data, which we call Unsupervised Deep Video Denoising (UDVD). UDVD is inspired by the "blind-spot" technique, recently introduced for unsupervised still image denoising [8, 66, 68, 74], in which a CNN is trained to estimate each *noisy* pixel from the surrounding spatial neighborhood *without including the pixel itself*. Here, we propose a blind-spot architecture that processes the surrounding spatio-temporal neighborhood to denoise videos.

We show that UDVD is competitive with the current supervised state-of-the-art on standard benchmarks, despite not having access to ground-truth clean videos during training (see Figure 3.1). Moreover, when combined with aggressive data augmentation and early stopping, it can produce high-quality denoising even when trained exclusively on a single *brief* noisy video sequence (as few as 30 frames), outperforming unsupervised video denoising techniques (e.g.

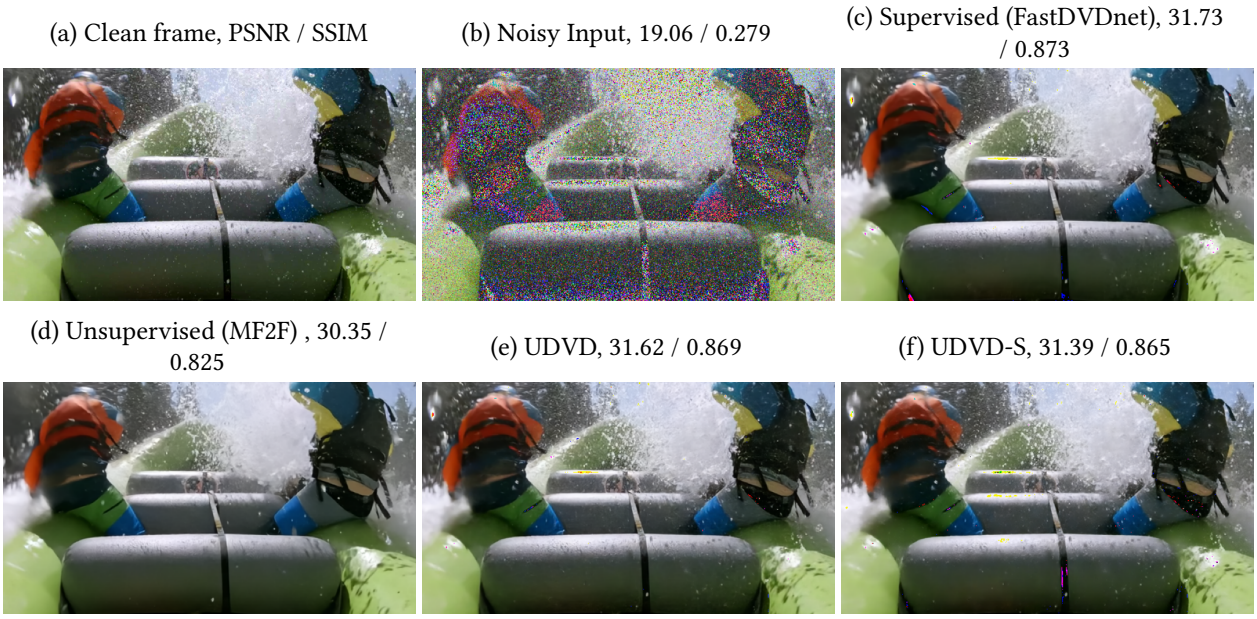


Figure 3.1: Unsupervised denoising matches the performance of supervised denoising. Frame from a video in the Set8 dataset denoised using different approaches. (a) Clean frame. (b) Frame corrupted with Gaussian noise of standard deviation 30 (relative to intensity range [0-255]). (c) FastDVDnet [130], a supervised method trained on the DAVIS dataset. (d) MF2F [29], an unsupervised method which fine-tunes a pre-trained FastDVDnet on the noisy video (e) Our proposed unsupervised method (UDVD), which uses five frames to denoise each frame, trained on the DAVIS dataset. (f) UDVD trained only on the noisy video itself. Performance is quantified using PSNR / SSIM [138], respectively. The corresponding videos, as well as additional examples, are included in Section C of the supplementary material.

F2F[34] and MF2F [29]) which are pre-trained with supervision. Finally, methods based on pre-training are not suitable for imaging applications where clean data is unavailable. In contrast, we demonstrate that UDVD can effectively denoise three different real-world datasets: raw videos from surveillance cameras, fluorescence-microscopy videos of cells, and electron-microscopy videos of catalytic nanoparticles.

The state-of-the-art performance of UDVD is unexpected. Nearly all existing approaches to video denoising [2, 14, 80, 84], including those based on deep CNNs [34, 44, 129, 143, 145], use estimates of optical flow to adaptively compensate for the motion of objects in the video. Conventional wisdom suggest that ignoring such motion should lead to denoising results in which moving content is blurred. Contrary to this intuition, UDVD and some recent state-of-the-art su-

pervised methods for video denoising [23, 27, 130] yield excellent empirical performance without explicit estimation of optical flow. *How can is this achieved?* We use a gradient-based analysis to show that both UDVD and supervised CNNs perform spatio-temporal *adaptive* filtering, which is aligned with underlying motion. Thus, these CNNs are *automatically performing implicit motion compensation*. To quantify this, we demonstrate that it is possible to estimate optical flow accurately from the network gradients, even though the network architectures are not designed to account for optical flow, and the models receive no optical-flow information during training.

3.2 Background and Related Work

Traditional and CNN-based video denoising. Traditional techniques for single image denoising include nonlinear filtering [92, 131], sparse prior methods [19, 25, 31, 36, 107, 120], and nonlocal means [71]; many of which have been extended to videos [2, 14, 80, 84]. In order to exploit the spatio-temporal structure of the video, these methods typically employ motion compensation based on estimates of optical flow.

In the last five years, data-driven methods based on deep CNNs [72] have outperformed all other techniques in image [21, 46, 150] and video denoising [129, 130, 143, 147]. The CNNs are trained to minimize the mean squared error between the network output and ground truth using large databases of natural figures/ch3/videos. Many deep-learning techniques also perform explicit motion compensation. DVDnet [129] applies an image-denoising CNN to each input frame, estimates the optical flow from the denoised frames using DeepFlow [139] (a CNN pretrained for this purpose), warps the frames using the flow estimate to align their content, and finally processes the registered frames with a CNN. Ref. [143] applies a similar pipeline, but jointly trains an optical-flow module with the denoising CNN.

Video denoising without motion compensation. Three recent methods perform video denoising without explicit motion estimation. VNLnet [27] uses a non-local search algorithm to

find self-similar patches in the input video, and then uses a CNN to process the patches. ViDeNN [23] consists of a first stage that denoises each frame using a CNN, and a second stage that exploits temporal structure by using the frames, $(t-1)$, t and $t+1$ to produce the denoised tth frame. FastDVDnet [130] uses UNet [116] blocks, trained end to end, to denoise each frame using five contiguous frames. These methods achieve state-of-the-art performance without any explicit motion compensation, similar to our proposed UDVD. In this chapter we show that such CNNs actually performs *implicit* motion estimation, which can be revealed through a gradient-based analysis.

Unsupervised denoising. Noise2Noise (N2N) is an unsupervised image-denoising technique where a CNN is trained on pairs of noisy images corresponding to the same clean image [74]. Frame2Frame (F2F) [34] exploits this approach to fine-tune a pretrained image-denoising CNN with noisy data. The idea is to register contiguous frames using the optical flow (obtained from TV-L1 [148]), and treat them as noisy realizations of the same clean image. This scheme is extended to have a trainable flow estimation module in [145], additional optical-flow consistency in [44] and to use multiple noisy frames as input in Multi-Frame2Frame (MF2F) [29].

Using the N2N framework to perform unsupervised video denoising requires warping adjoining frames, which in turn requires explicit motion compensation, and accurate occlusion estimation. In addition, the assumption that contiguous frames can be registered may not hold, particularly if the motion speeds in the video are large relative to the frame rate or local intensity changes are not due to translation. In order to bypass these issues, we develop a blind-spot network that trains denoising CNNs by fitting the noisy data directly. The CNN is trained to estimate each noisy pixel value using the surrounding spatio-temporal neighborhood, but without taking into account the noisy pixel itself in order to avoid the trivial identity solution. This "blind spot" can be enforced through architecture design [68], or by masking [8, 66]. For still images, several variations of this approach have been shown to provide effective denoising for natural images and noisy images from fluorescence microscopy [60, 67, 108].

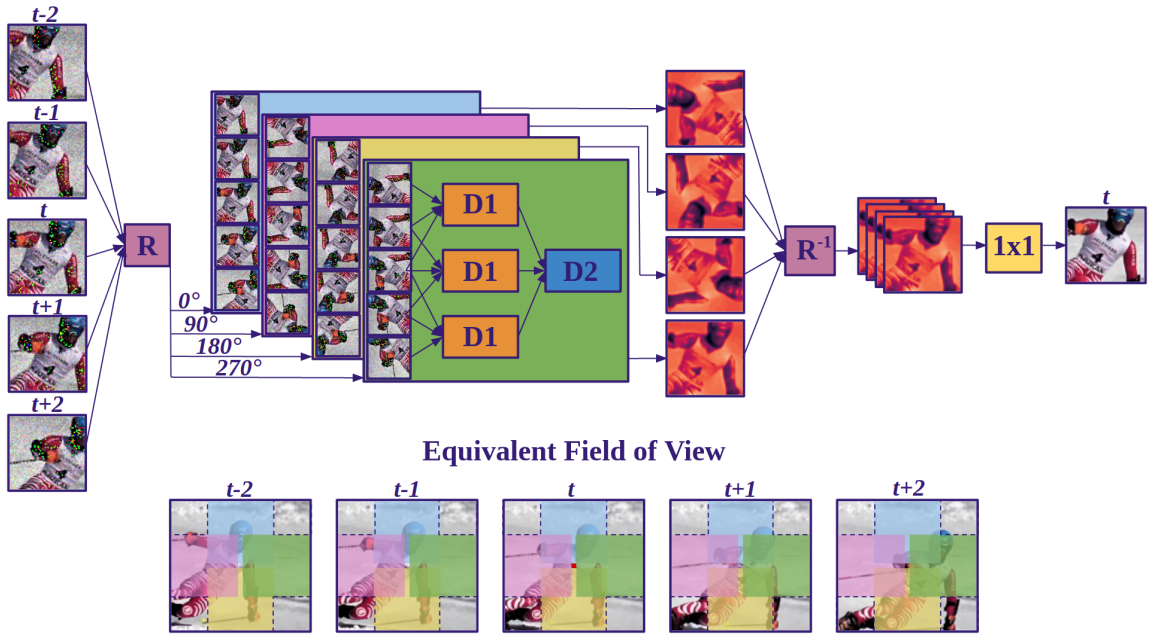


Figure 3.2: Unsupervised Deep Video Denoising (UDVD) Network Architecture. The network takes 5 consecutive noisy frames as input and produces a denoised central frame as output. We rotate the input frames by multiples of 90° and process them in four separate branches with shared parameters, each containing asymmetric convolutional filters that are *vertically causal*. As a result, the branches produce outputs that only depend on the pixels above (0° rotation, blue region), to the left (90°, pink region), below (180°, yellow region) or to the right (270°, green region) of the output pixel. Each branch consists of a cascade of 2 Unet-style blocks (D1 and D2) to combine information over frames. These outputs are then *derotated* and linearly combined (using a 1×1 convolutions) followed by a ReLU nonlinearity to produce the final output. The resulting "field of view" is depicted at the bottom with each color representing the contribution of the corresponding branch.

3.3 Unsupervised Deep Video Denoising

In this section we describe our proposed architecture (see Figure 3.2 for a detailed diagram).

Multi-frame blind-spot architecture. Our CNN maps five contiguous noisy frames to a denoised estimate of the middle frame. Building on the "blind spot" idea proposed in [68] for single-image denoising, we design the architecture so that each output pixel is estimated from a spatio-temporal neighbourhood that does not include the pixel itself. We rotate the input frames by multiples of 90° and process them through four separate branches containing asymmetric con-

volutional filters that are *vertically causal*. As a result, the branches produce outputs that only depend on the pixels above (0° rotation), to the left (90°), below (180°) or to the right (270°) of the output pixel. These partial outputs are then *derotated* and combined using a three-layered cascade of 1×1 convolutions and nonlinearities to produce the final output. The resulting field of view does not include the pixel being denoised, as depicted at the bottom of Figure 3.2.

UDVD processes the video in two stages as shown in Figure 3.2, similar to previously proposed networks for supervised video denoising [23, 129, 130]. A first stage, consisting of three UNets [116] (D1 in the diagram) with shared parameters, maps each group of three contiguous frames (i.e. $(t-2, t-1, t)$, $(t-1, t, t+1)$ and $(t, t+1, t+2)$) to a separate feature map. These features are then mapped to a single output using another UNet (D2). See Section B.1.3 for a detailed description of the architecture.

Bias-free architecture. Inspired by Section 2.4, we remove all additive terms from the convolutional layers in UDVD. This provides automatic generalization to varying noise levels not encountered during training, and facilitates our proposed analysis to interpret the denoising mechanisms learned by the network (see Section 3.5 and 3.6).

Using the missing pixel. The denoised value generated by the proposed architecture at each pixel is computed without using the noisy observation at that location. This avoids overfitting – i.e. learning the trivial identity map that minimizes the mean-squared error cost function – but ignores important information provided by the noisy pixel. In the special case of Gaussian additive noise, we can use this information via a precision-weighted average between the network output and the noisy pixel value. Following [67, 68], the weights in the average are derived by assuming a Gaussian distribution for the error in the blind-spot estimates of the color pixel values. Specifically, we model the distribution of the three color channels of a pixel $x \in \mathcal{R}^3$ given the noisy neighbourhood Ω_y as $p(x|\Omega_y) = \mathcal{N}(\mu_x, \Sigma_x)$, where $\mu_x \in \mathcal{R}^3$ and $\Sigma_x \in \mathcal{R}^3$ represent the mean vector and covariance matrix. Let $y = x + \eta$, $\eta \sim \mathcal{N}(0, \sigma^2 I_3)$ be the observed noisy pixel. We integrate the information in the noisy pixel with the UDVD output by computing the mean

of the posterior $p(x|y, \Omega_y)$, given by

$$E[x|y] = (\Sigma_x^{-1} + \sigma^{-2} I)^{-1} (\Sigma_x^{-1} \mu_x + \sigma^{-2} y). \tag{3.1}$$

See Suppl. A for more details. The CNN architecture is trained to estimate the mean and covariance of this distribution at each pixel by maximizing the log likelihood of the noisy data: $\mathrm{L}(\mu_x, \Sigma_x) = \frac{1}{2}[(y - \mu_x)^T (\Sigma_x + \sigma^2 I)^{-1} (y - \mu_x)] + \frac{1}{2} \log |\Sigma_x + \sigma^2 I|$. When the noise process is unknown, we simply minimize the MSE between the denoised output and noisy video, and ignore the center pixel (see Section B.1.2 for more details).

Data augmentation and early stopping. In supervised denoising with simulated noise, training can rely on the generation of a virtually unlimited set of fresh noise realizations, which prevents overfitting. In the unsupervised setting, this is not possible, which makes it more challenging to train models that can denoise short video sequences. To address this, we (a) leverage data augmentation strategies: spatial flipping and time reversal, and (b) perform early stopping by monitoring the mean squared error between the network output and noisy frames on a held-out set of frames. These strategies make it possible to train UDVD with short video sequences (as few as 30 frames), while achieving denoising performance that is on par with or superior to both unsupervised and supervised networks trained on much larger datasets (see Figure 3.1, Table 3.2 and Section B.4.2).

3.4 Datasets

We demonstrate the broad applicability of our approach by validating it on domains with different signal and noise structure: natural videos, raw videos, fluorescence microscopy, and electron microscopy.

Natural videos. We perform controlled experiments on natural videos by adding iid Gaussian

		Traditional		Supervised CNN			Unsupervised CNN (UDVD)		
test set	σ	VNLB	VBM4D	VNLnet	DVDnet	FastDVDnet	1 frame	3 frames	5 frames
DAVIS	30	33.73	31.65	-	**34.08**	34.06	32.80	33.48	**33.92**
	40	32.32	30.05	32.32	**32.86**	32.80	31.48	32.20	**32.68**
	50	31.13	28.80	31.43	**31.85**	31.83	30.47	31.20	**31.70**
Set8	30	31.74	30.00	-	**31.79**	31.60	30.91	31.62	**32.01**
	40	30.39	28.48	**30.55**	**30.55**	30.37	29.63	30.42	**30.82**
	50	29.24	27.33	29.47	**29.56**	29.42	28.65	29.47	**29.89**

Table 3.1: Denoising results on natural video datasets. All networks are trained on the DAVIS train set. Performance values are PSNR of each trained network averaged over held-out test data. UDVD, operating on 5 frames, outperforms the supervised methods on Set8 and is competitive on the DAVIS test set. Unsupervised denoisers with more temporal frames show a consistent improvement in denoising performance. DVDnet and FastDVDnet are trained using varying noise levels ($\sigma \in [0, 55]$) and VNLnet is trained and evaluated on each specified noise level. All UDVD networks are trained *only* at $\sigma = 30$, showing that they generalize well on unseen noise levels. See Sections C and F in the supplementary material for additional results. The PSNR values for all methods except UDVD are taken from [130].

noise to the DAVIS dataset [106]. The training/validation/test split is 60/30/30 videos, respectively. We use three additional datasets for testing - Set8 [130] composed of 4 videos from the Derfs Test Media collection and 4 videos captured with a GoPro camera, Derfs [29] with 7 videos, and the first 10 videos from Vid3oC [62] dataset (See Section B.4.1 for details).

Raw videos. We evaluate UDVD on a dataset of raw videos i.e with frame color channels interleaved according to the sensor mosaic containing real noise introduced in [147]. The dataset contains 11 unique videos, each containing 7 frames, captured at five different ISO levels using a surveillance camera. Each video has 10 different noise realizations per frame, which are averaged to obtain an estimated clean version of the video.

Fluorescence microscopy. We apply our approach to fluorescence-microscopy recordings of live cells in [132]. We use two videos: Fluo-C2DL-MSC (CTC-MSC) depicting mesenchymal stem cells, and Fluo-N2DH-GOWT1 (CTC-N2DH) depicting GOWT1 cells. This dataset illustrates the challenges of applying supervised approaches to real data: there is no ground-truth clean data.

Electron microscopy. We also apply our methodology to a transmission electron microscopy dataset from [95]. The data consist of a 40-frame video depicting a platinum nanoparticle sup-

ported on a cerium oxide base. The average image intensity is 0.45 electrons/pixel, which results in an extremely low signal-to-noise ratio. As with the fluorescence-microscopy data, no ground-truth clean images are available. See Chapter 5 for more details.

3.5 Experiments and Results

Comparison with other approaches on natural videos. We train UDVD on the DAVIS training set (see Suppl. A for the training procedure). Following [8, 66, 68, 96, 129, 130, 150], we add iid Gaussian noise with standard deviation $\sigma = 30$ on the clean videos during training. UDVD is evaluated on the DAVIS test set and on Set8 by comparing to the clean ground-truth videos via PSNR. We compare UDVD with several popular methods: Bayesian processing of spatio-temporal patches (VNLB [71]), an extension of the popular image-denoising algorithm BM3D (VBM4D [84]) and supervised CNNs (VNLnet [27], DVDnet [129], FastDVDnet [130]). As shown in Table 3.1, UDVD achieves comparable performance to the supervised state-of-the-art on the DAVIS test set and slightly outperforms these methods on an independent test set (Set8) at multiple noise levels. It also outperforms traditional unsupervised techniques such as VNLB and VBM4D (see Figure 3.1 and Section B.3 for visual examples).

Unsupervised denoising from limited data. In order to validate our approach on a more challenging setting that is closer to the practical applications of unsupervised denoising, we trained and tested UDVD on individual videos from our test sets. As shown in Table 3 and 4 in Suppl. D, when combined with data augmentation and early stopping (using the last 5 frames of each video as a held-out validation set), this version of UDVD (called UDVD-S) achieves comparable results, or often outperforms supervised FastDVDnet and unsupervised UDVD trained on a large dataset (DAVIS) (see Table 3.2 for results on 4 different datasets).

To the best of our knowledge, all the existing unsupervised video denoising techniques are based on the F2F [34] framework, where a backbone CNN pre-trained with supervision is fine-

	$\sigma = 30$				$\sigma = 90$			
	DAVIS	Set8	Derfs	Vid3oC	DAVIS	Set8	Derfs	Vid3oC
UDVD-S	33.68 / 78.16	**32.90** / 81.85	**33.95** / 81.91	34.65 / 84.60	**29.05** / **53.53**	**28.07** / **55.35**	**29.42** / **59.25**	**29.94** / **63.79**
UDVD*	33.78 / 79.88	31.90 / **82.53**	32.58 / 81.44	34.24 / 83.96	28.87 / 51.22	27.25 / 51.84	28.26 / 52.44	29.23 / 60.08
FastDVDnet*	**33.91** / 76.99	31.81 / 80.21	32.45 / 81.64	35.05 / 84.44	28.01 / 47.53	26.54 / 50.16	27.36 / 52.87	28.42 / 55.99
MF2F	**33.91** / **80.01**	31.84 / 80.55	32.87 / **82.22**	**35.18** / **85.71**	28.81 / 51.24	27.25 / 52.78	28.29 / 55.06	29.67 / 61.28

Table 3.2: Results for UDVD trained on individual noisy videos. The top row shows **PSNR/VMAF**[76] values (averaged over the entire dataset) for UDVD trained on each individual video sequence with early stopping (labelled UDVD-S) using the last 5 frames of a video as a held-out set. We augmented the dataset with spatial flipping and time reversal (see Suppl. D for an ablation study). With the augmentations and early stopping, UDVD-S is comparable to (and often outperforms) UDVD or FastDVDnet trained on the full DAVIS dataset (indicated by *) and MF2F, which fine-tunes a pre-trained CNN on each individual video. See Suppl. D for results on individual video sequences.

tuned on the video to be denoised. We compared UDVD-S against the most recent such method – MF2F [29] which fine-tunes a FastDVDnet [130] trained with supervision on natural videos using an objective involving optical flow computed on consecutive noisy frames (see Section 3.2). Without any pre-training, UDVD-S outperforms MF2F in almost all videos in Tables B.3 and B.4, and datasets in Table 3.2. Note that (a) we trained MF2F using all the 5 training schemes provided in the paper and reported the best results in Table 3.2, and (b) the metric we used to measure performance in Table 3.2 is the average PSNR of all denoised frames, unlike in Ref. [29] where the first 10 frames of each video were excluded.

Use of temporal information. UDVD estimates each frame from k surrounding contiguous frames. To validate the effect of using more temporal information, we tested $k \in \{1, 3, 5\}$. As shown in Table 3.1, performance improves substantially and monotonically with k (see Section B.2 for more noise levels). This is in agreement with the literature on supervised learning [130]. The performance gains arising from a longer temporal context are more substantial at higher noise levels (see Table 3.1). This is consistent with our analysis in Section 3.6 which shows that, at low noise levels, UDVD($k = 5$) tends to ignore the distant frames, but relies on them more at higher noise levels (see Figure 3.4 and Section B.7).

Generalization across noise levels. UDVD generalizes strongly across noise levels not en-

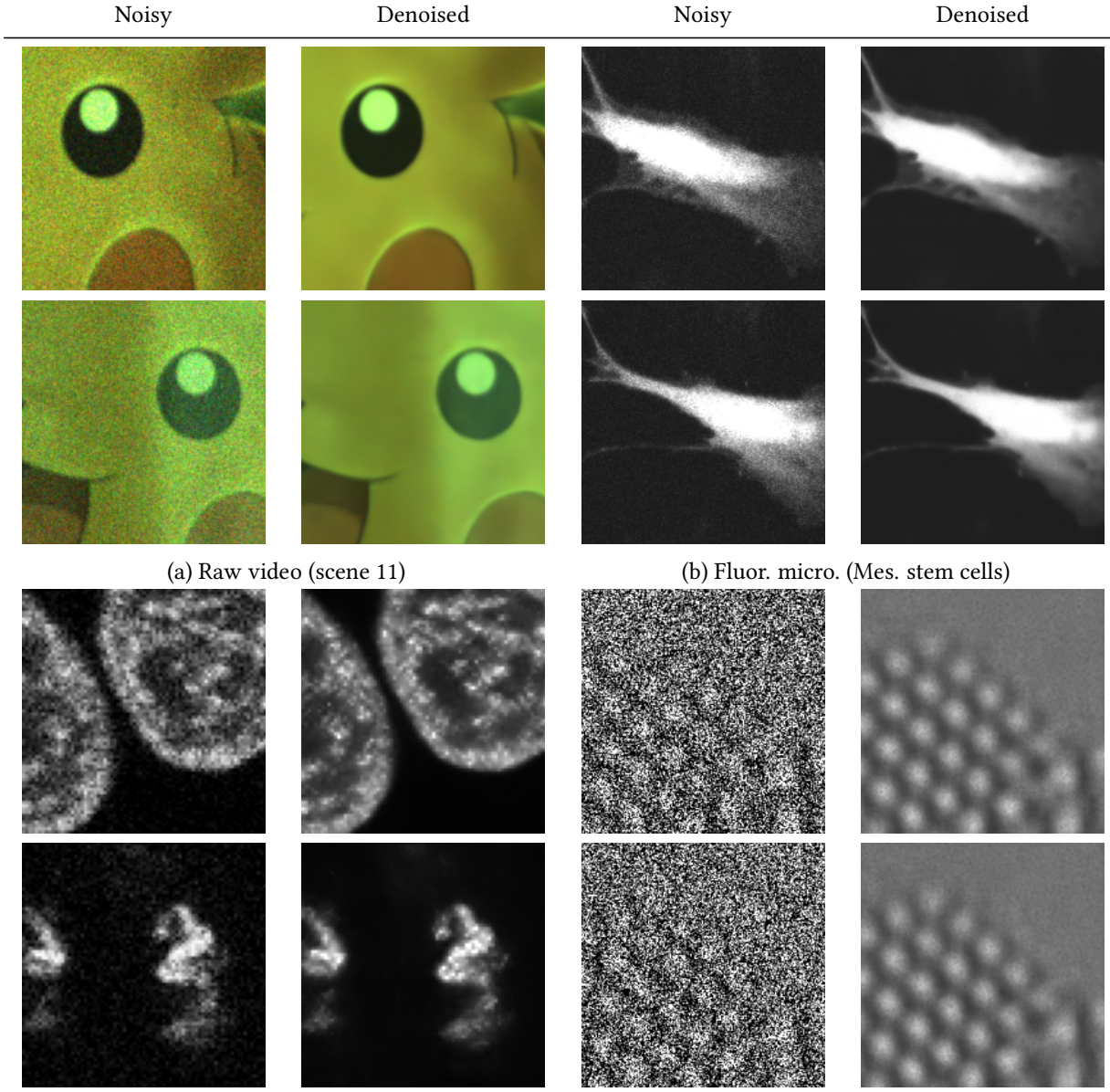
CNNISO	1600	3200	6400	12800	25600	**mean**
UDVD	**48.04**	**46.24**	**44.70**	**42.19**	**42.11**	**44.69**
RViDeNet [147]	47.74	45.91	43.85	41.20	41.17	43.97

Table 3.3: Raw video denoising. PSNR values evaluated on the test set of the raw video dataset (Section 3.4) when denoised with (a) UDVD trained only the noisy test videos and (b) RViDeNet trained with supervision on a large dataset. The columns correspond to different ISO levels, with larger levels resulting in noisier data.

countered during training. The results in Table 3.1 are obtained with a network trained only at a fixed noise level of $\sigma = 30$. This generalization ability is consistent with bias-free networks for image denoising (Chapter 2). See Section B.6 for more discussion and results.

Raw videos with real noise. We train UDVD on the first 9 realizations of the 5 videos from the test set of the raw video dataset (see Section 3.4), holding out the last realization for early stopping. We compare our performance with RViDeNet [147] which is pre-trained on a simulated dataset and then fine-tuned with supervision on 6 training videos from the raw video dataset. UDVD outperforms RViDeNet at all noise levels (see Table 3.3 and Fig 3.3). Note that UDVD was directly trained on the mosaiced raw videos. Existing unsupervised video denoising methods, like MF2F, cannot be applied directly on this dataset as their pre-trained backbone expects an input in the RGB domain.

Real-world microscopy data. We train UDVD on the fluorescence-microscopy data described in Section 3.4 following the same procedure as for the natural videos, including data augmentation. For the electron-microscopy data, we trained on the first 35 frames of the video, and used the remaining 5 as a validation set to perform early stopping based on mean-squared error. UDVD is able to effectively denoise the fluorescence-microscopy and the electron-microscopy datasets described in Section 3.4. This can be appreciated qualitatively in Figure 3.3 and Section B.5.



(a) Raw video (scene 11)

(b) Fluor. micro. (Mes. stem cells)

(c) Fluor. micro. (GOWT1 cells)

(d) Electron micro. (nanoparticle)

Figure 3.3: Denoising real-world data. Results from applying UDVD to the raw video, fluorescence-microscopy and electron-microscopy datasets described in Section 3.4. Qualitatively, UDVD succeeds in removing noise while preserving the underlying signal structure, even for the highly noisy electron-microscopy data. Raw videos are converted to RGB for visualization.

3.6 Automatic Motion Compensation

Most previous approaches for video denoising rely on explicit motion compensation [2, 14, 80, 84]. This requires estimating the optical flow, which is the local translational motion of features in the image arising from the motion of objects and surfaces in a visual scene relative to the camera. Several CNN-based denoisers build motion estimation into the architecture [129, 143]. In particular, motion compensation is critical to the F2F and MF2F frameworks for unsupervised denoising, which use motion compensation to register contiguous images [29, 34, 44]. In contrast, recent supervised video denoising networks like FastDVDnet [130] and ViDeNN [23], as well as our unsupervised UDVD, do not perform any explicit motion compensation. Despite this, they achieve state-of-the-art results. The empirical performance of these approaches suggests that the networks must somehow be exploiting temporal information successfully. Here, we study this phenomenon through an analysis of the denoising mapping, which reveals that these networks perform an implicit form of motion compensation.

Gradient-based analysis. We use the approach developed in Section 2.5 to analyze CNNs trained for image denoising. Let $y \in \mathbb{R}^{nT}$ be a flattened video sequence containing T noisy frames with n pixels each, processed by a CNN. We define the denoising function $f_i : \mathbb{R}^{nT} \rightarrow \mathbb{R}$ as the map between the noisy video and the denoised value $d_i := f_i(y)$ of the CNN output at the ith pixel. A first-order Taylor decomposition of the denoising function may be written as:

$$d_i := f_i(y) = \langle \nabla f_i(y), y \rangle + b, \tag{3.2}$$

where $\nabla f_i(y) \in \mathbb{R}^{nT}$ denotes the gradient of f_i at y. The constant $b := f_i(y) - \langle \nabla f_i(y), y \rangle$ is the net bias of the network, a combined function of all additive constants in the convolutional and batch-normalization layers of the CNN.

Our proposed architecture is bias-free (i.e., all additive constants are removed from the archi-

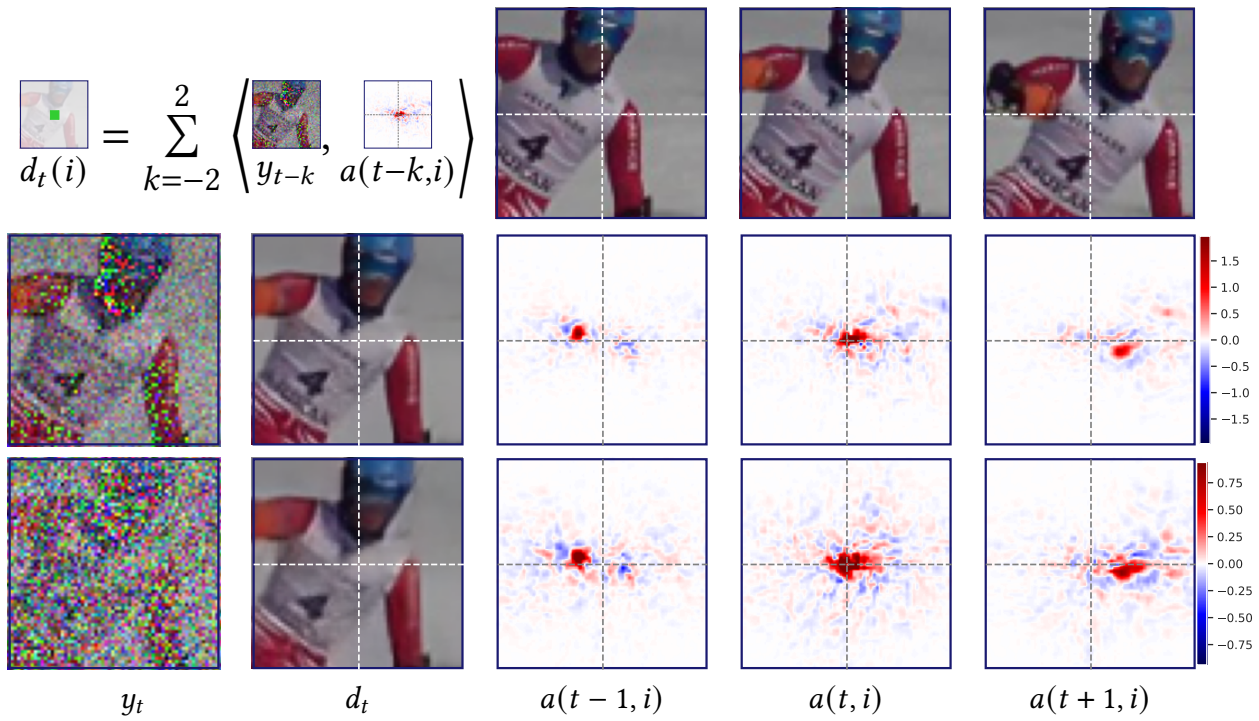


Figure 3.4: Video denoising as spatiotemporal adaptive filtering. Visualization of the equivalent linear weights ($a(k,i)$, Eq. 3.3) used to compute two example denoised pixels using UDVD. The left two columns show noisy frames y_t at two noise levels, and the corresponding denoised frames, d_t. Three successive clean frames $\{x_{t-1}, x_t, x_{t+1}\}$ are shown in top row, for reference. Corresponding weights $a(k,i)$ for pixel i (intersection of the dashed white lines) in these three frames, are shown in the last three columns. The weights are seen to adapt to underlying video content, with their mode shifting to track the motion of the skier. As the noise level σ increases (bottom row), their spatial extent grows, averaging out more of the noise while respecting object boundaries. For each denoised pixel, the sum of weights (over all pixel locations and frames) is approximately one, and thus can be interpreted as computing a local average (but note that some weights are negative, depicted in blue).

tecture, as proposed in Chapter 2, and thus $b = 0$. As a result, the denoised value at the ith pixel may be written as:

$$d(i) = \langle \nabla f_i(y), y \rangle = \sum_{k=1}^{T} \langle a(k,i), y_k \rangle, \tag{3.3}$$

where y_k denotes each of the T flattened frames that compose the noisy video, and the weights $a(k,i)$ correspond to the gradient of f_i with respect to y. Each vector $a(k,i)$ can be interpreted as an *equivalent filter* that produces an estimate of the denoised video at pixel i via a weighted average of the noisy observations over space and time.

Interpreting equivalent filters. Visualizing these equivalent filters reveals that UDVD learns to denoise by performing averaging over an adaptive spatiotemporal neighborhood of each pixel. As illustrated in Figures 3.4, B.4, B.5 and B.6 , when the noise level increases, the averaging is carried out over larger regions. This intuitive behavior is also seen in classical linear Wiener filters [140], where the filters are larger for higher levels of noise. The crucial difference is that in the case of CNNs, the equivalent filters are *adapted* to the local video content: they respect object boundaries in space and time, taking into account their motion. This is apparent in Figure 3.4: equivalent filters in adjoining frames are automatically shifted spatially to compensate for the movement of the skier (additional examples in Section B.7). We find that this implicit motion compensation is not unique to UDVD: CNNs trained in a supervised fashion have the same property (see also Section B.7).

Optical-flow estimation. In order to validate our observation that CNNs exclusively trained for denoising implicitly detect and exploit video motion, we use the equivalent filters of the networks to estimate the optical flow. To estimate the optical flow from the t^{th} frame to the $(t+1)^{th}$ frame at the ith pixel, we compute the difference between the position of the centroid of the equivalent filter corresponding to the pixel at times t, $a(t,i)$, and time $t+1$, $a(t+1,i)$. To increase the stability of the estimated flow, we compute the filter centroid through a robust weighted average that only

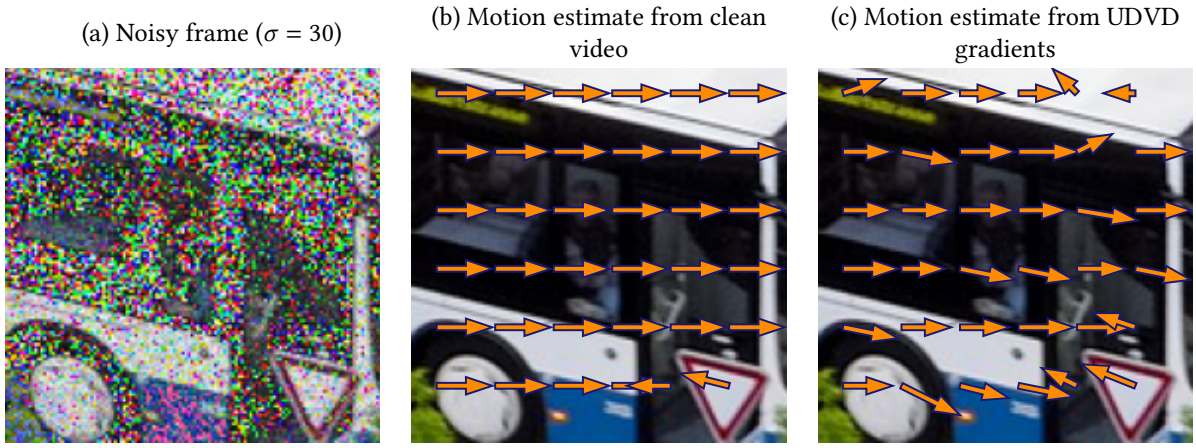


Figure 3.5: CNNs trained for denoising automatically learn to perform motion estimation. (a) Noisy frame from a video in the DAVIS dataset. (b) Optical flow direction at multiple locations of the image obtained using a state-of-the-art algorithm applied *to the clean video*. (c) Optical flow direction estimated from the shift of the adaptive filter obtained by differentiating the network, which is trained exclusively with noisy videos and no optical flow information. Optical flow estimates are well-matched to those in (b), but deviate according to the aperture problem at oriented features (see black vertical edge of bus door), and in homogeneous regions (see bus roof, top right).

includes entries with relatively large values (within 20% of maximum value in the filter).

The optical-flow estimates obtained from the gradients of the trained UDVD network are surprisingly precise, even at very high noise levels. Figure 3.5, and additional figures in Section B.7, show that the results are similar to those obtained by applying an algorithm for optical-flow estimation (DeepFlow [139]) on the corresponding *clean* video. This demonstrates that the CNNs are able to implicitly estimate motion from data, despite the fact that they were not trained on that problem, and *even in the presence of substantial noise corruption*, a setting that is quite challenging for optical-flow estimation techniques. We also observe that the optical-flow estimates obtained from UDVD gradients tend to be less accurate for pixels near strongly oriented features where local motion is only partially constrained (known as the *aperture problem*) or in homogeneous regions, where the local motion is unconstrained (the *blank wall problem*).

3.7 Conclusion

In this chapter we propose a method for unsupervised deep video denoising that achieves comparable performance to state-of-the-art supervised approaches. Combined with data-augmentation techniques and early stopping, the method achieves effective denoising even when trained exclusively on short individual noisy sequences, which enables its application to real-world noisy data. In addition, we perform a gradient-based analysis of denoising CNNs, which reveals that they learn to perform implicit adaptive motion compensation. This suggests several interesting research directions. For example, denoising may be a useful pretraining task for optical-flow estimation or other computer-vision tasks requiring motion estimation. In the next chapter, we will build on the framework of unsupervised denoising to develop a semi-supervised denoising paradigm.

4 Adaptive denoising: Generalizing pre-trained denoisers to out-of-distribution data

This chapter is adapted from the paper "Adaptive Denoising via GainTuning" published in Neural Information Processing Systems (NeurIPS) 2021 [94]. This is a joint work with Joshua Vincent, Ramon Manzorro, Peter A Crozier, Carlos Fernandez-Granda, and Eero P Simoncelli. In this chapter, we will explore a more general version of the robustness problem we studied in Chapter 2. The framework will rely on unsupervised denoising techniques discussed in Chapter 3.

ABSTRACT

Deep convolutional neural networks (CNNs) for image denoising are typically trained on large datasets. These models achieve the current state-of-the-art, but they do not generalize well to data that deviate from the training distribution. Recent work has shown that it is possible to train denoisers on a single noisy image. These models adapt to the features of the test image, but their performance is limited by the small amount of information used to train them. Here we propose “GainTuning”, a methodology by which CNN models pre-trained on large datasets can be adaptively and selectively adjusted for individual test images. To avoid overfitting, GainTuning

optimizes a single multiplicative scaling parameter (the "Gain") of each channel in the convolutional layers of the CNN. We show that GainTuning improves state-of-the-art CNNs on standard image-denoising benchmarks, boosting their denoising performance on nearly every image in a held-out test set. These adaptive improvements are even more substantial for test images differing systematically from the training data, either in noise level or image type. We illustrate the potential of adaptive GainTuning in a scientific application to transmission-electron-microscope images, using a CNN that is pre-trained on synthetic data. In contrast to the existing methodology, GainTuning is able to faithfully reconstruct the structure of catalytic nanoparticles from these data at extremely low signal-to-noise ratios.

4.1 Overview

Like many problems in image processing, the recovery of signals from noisy measurements has been revolutionized by the development of convolutional neural networks (CNNs) [21, 149, 150]. These models are typically trained on large databases of images, either in a supervised (like in Chapters 1 and 2) [21, 96, 149–151] or an unsupervised (like in Chapter 3) fashion [8, 66, 68, 141]. Once trained, these solutions are evaluated on noisy test images. This approach achieves state-of-the-art performance when the test images and the training data belong to the same distribution. However, when this is not the case, the performance of these models is often substantially degraded [96, 133, 151]. This is an important limitation for many practical applications, in which it is challenging (or even impossible) to gather a training dataset that is comparable in noise and signal content to the images encountered at test time. Overcoming this limitation requires *adaptation* to the test data.

A recent unsupervised method (Self2Self) has shown that CNNs can be trained exclusively on individual test images, producing impressive results [110]. Despite this, the performance of Self2Self is limited by the small amount of available training information, and is generally inferior

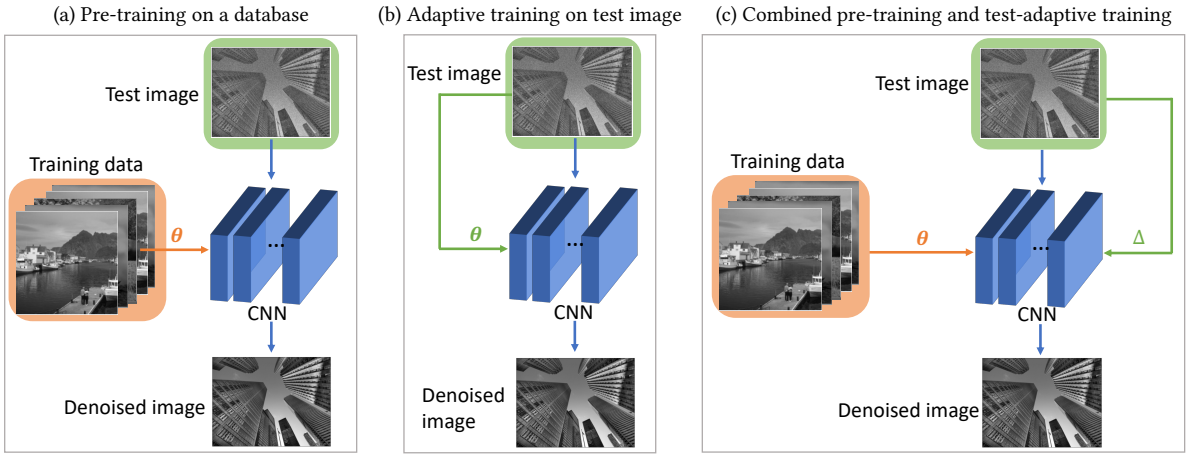


Figure 4.1: Proposed denoising paradigm. (a) Typically, CNNs are trained on a large dataset and evaluated directly on a test image. (b) Recent unsupervised methods perform training on a single test image. (c) We propose GainTuning, a framework which bridges the gap between both of these paradigms: a CNN pre-trained on a large training database is adapted to the test image.

to CNN models trained on large databases.

In this chapter, we propose *GainTuning*, a framework to bridge the gap between models pre-trained on large datasets, and models trained exclusively on test images. In the spirit of two recent methods [123, 133], GainTuning adapts pre-trained CNN models to individual test images by minimizing an unsupervised denoising cost function, thus fusing the generic capabilities obtained from the training data with specific refinements matched to the structure of the test data. Rather than adapt the full parameter set (filter weights and additive constants) to the test image, GainTuning instead optimizes a single multiplicative scaling parameter (the "Gain") for each channel within each layer of the CNN. The dimensionality of this reduced parameter set is a small fraction ($\approx$ 0.1% in our examples) of that of the full parameter set. We demonstrate through extensive examples that this prevents overfitting to the test data. The GainTuning procedure is general, and can be applied to any CNN denoising model, regardless of the architecture or pre-training process.

GainTuning provides a novel method for adapting CNN denoisers trained on large datasets to a single test image. GainTuning improves state-of-the-art CNNs on standard image-denoising

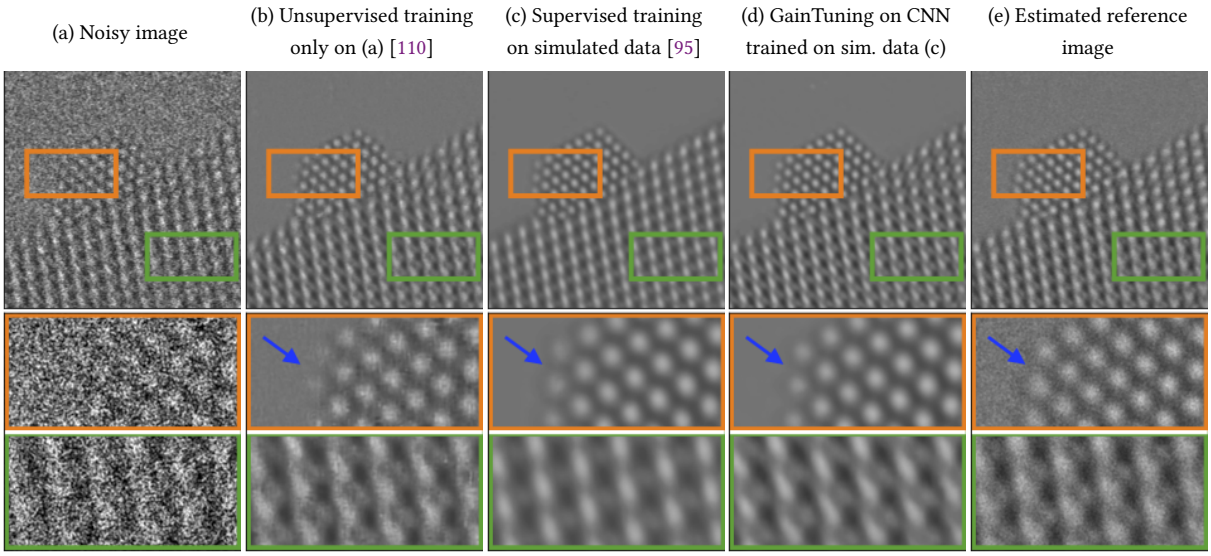


Figure 4.2: Denoising results for real-world data. (a) An experimentally-acquired atomic-resolution transmission electron microscope image of a CeO2-supported Pt nanoparticle. The image has a very low signal to noise ratio (PSNR of $\approx 3dB$). (b) Denoised image obtained using Self2Self [110], which fails to reconstruct three atoms (blue arrow, second row). (c) Denoised image obtained via a CNN trained on a simulated dataset, where the pattern of the supporting atoms is not recovered faithfully (third row). (d) Denoised image obtained by adapting the CNN in (c) to the noisy test image in (a) using GainTuning. Both the nanoparticle and the support are recovered without artefacts. (e) Reference image, estimated by averaging 40 different noisy images of the same nanoparticle.

benchmarks, boosting their denoising performance on nearly every image in held-out test sets. Performance improvements are even more substantial when the test images differ systematically from the training data. We showcase this ability through controlled experiments in which we vary the distribution of the noise and image structure of the test data. Finally, we evaluate GainTuning in a real scientific-imaging application where adaptivity is crucial: denoising transmission-electron-microscope data at extremely low signal-to-noise ratios. As shown in Figure 4.2, both CNNs pre-trained on simulated images and CNNs trained only on the test data produce denoised images with substantial artefacts. In contrast, GainTuning achieves effective denoising, accurately revealing the atomic structure in the real data.

4.2 Related Work

Denoising via deep learning. As discussed in Chapter 1, CNN-based methods have clearly outperformed previous state-of-the-art denoising methods [19, 25, 31, 35, 48, 107, 120]. Denoising CNNs are typically trained in a supervised fashion, minimizing mean squared error (MSE) over a large database of example ground-truth clean images and their noisy counterparts [21, 96, 150]. Unsupervised methods have also been developed, which do not rely on ground-truth images. There are two main strategies to achieve this: use of an empirical Bayes objective, such as Stein's unbiased risk estimator (SURE) [31, 81, 91, 114, 123, 124], and architectural "blind-spot" methods [8, 66, 68, 141] (see Section 4.4 for a more detailed description).

Generalization to out-of-distribution noise. Previous studies, including Chapter 2, have shown that CNN denoisers fail to generalize when the noise encountered at test time differs from that of the training data [96, 151]. Chapter 2 proposes the use of a modified CNN architecture without additive bias terms, which is able to generalize to noise with variance well beyond that encountered in the training set. Here, we show that augmenting a generic architecture with GainTuning yields comparable performance to removing bias.

Generalization to out-of-distribution images. In order to adapt CNNs to operate on test data with characteristics differing from the training set, recent publications propose fine-tuning the networks using an additional training dataset that is more aligned with the test data [42, 133]. This is a form of transfer learning, a popular technique in classification problems [30, 144]. However, it is often challenging to obtain relevant additional training data. Here, we show that GainTuning can adapt CNN denoisers to novel test images.

Feature normalization. Normalization techniques such as batch normalization (BN) [55] are a standard component of deep CNNs. BN consists of two stages: (1) centering and normalizing the features corresponding to each channel, (2) scaling and shifting the normalized features using two learned parameters per channel (a scaling factor and a shift). The scaling parameter is analogous

to the gain parameter introduced in GainTuning. However, in BN this parameter is adjusted during training and fixed during test time, whereas GainTuning adjusts it adaptively, for each test image.

Gain normalization. Motivated by gain control properties observed in biological sensory neurons [18], adaptive local normalization of response gains has been previously applied in object recognition [56], density estimation [4], and compression [5]. In contrast to these approaches, which adjust gains based on local responses, GainTuning adjusts a global gain for each channel by optimizing an unsupervised objective function.

Adapting CNN denoisers to test data. Two recent publications have developed methods of adapting CNN denoisers to test data [124, 133]. Ref. [124] include the noisy test images in the training set. In a recent extension, the authors fine-tune a pre-trained CNN on a single test image using the SURE cost function [123]. Ref. [133] does the same using a novel cost function based on noise resampling (see Section 4.4 for a detailed description). As shown in Section C.4 fine-tuning the full set of CNN parameters using only a single test image can lead to overfitting. Ref. [123] avoids this using early stopping, selecting the number of fine-tuning steps beforehand. Ref. [133] uses a specialized architecture with a reduced number of parameters. Here, we show that several unsupervised cost functions can be used to perform adaptation without overfitting, as long as we only optimize a subset of the model parameters (specifically, the gain of each channel). .

Adjustment of channel parameters to improve generalization in other tasks. Adjustment of channel parameters, such as gains and biases, has been shown to improve generalization in multiple machine-learning tasks, such as the vision-language problems [28, 103], image generation [20], style transfer [39], and image restoration [47]. In these methods, the adjustment is carried out while training the model by minimizing a supervised cost function. In image classification, recent studies have proposed performing adaptive normalization [58, 100, 118] and optimization [136] of channel parameters during test time, in the same spirit as GainTuning.

4.3 Proposed Methodology: GainTuning

In this section we describe the GainTuning framework. Let f_θ be a CNN denoiser parameterized by weight and bias parameters, θ. We assume that we have available a training database and a test image $\mathbf{y}_{\text{test}}$ that we aim to denoise. First, the networks parameters are optimized on the training database

$$\theta_{\text{pre-trained}} = \arg\min_{\theta} \sum_{y \in \text{training database}} \mathcal{L}_{\text{pre-training}}(y, f_\theta(y)). \tag{4.1}$$

The cost function $\mathcal{L}_{\text{pre-training}}$ used for pre-training can be supervised, if the database contains clean and noisy examples, or unsupervised, if it only contains noisy data.

A direct method of adapting the pre-trained CNN to the test data is to finetune all the parameters, as is done in all prior work on test-time adaptation [42, 123, 133]. Unfortunately this can lead to *overfitting* the test data (see Section C.4). Due to the large number of degrees of freedom, the model is able to minimize the unsupervised cost function without denoising the noisy test data effectively. This can be avoided to some extent by employing CNN architectures with a small number of parameters [133], or by only optimizing for a short time ("early stopping") [123]. Unfortunately, using a CNN with reduced parameters can limit performance (see Section 4.5), and it is unclear how to choose a single criterion for early stopping that can operate correctly for all test images. Here, we propose a different strategy: tuning a single parameter (the *gain*) in each channel of the CNN. GainTuning can be applied to any pre-trained CNN.

We denote the gain parameter of the c^{th} channel of the the l^{th} layer as $\gamma[l, c]$, and the conventional parameters of that channel by $\theta_{\text{pre-trained}}[l, c]$ (a vector containing the filter weights). The adapted GainTuning parameters are the product of these:

$$\theta_{\text{GainTuning}}(\gamma)[l, c] = \gamma[l, c]\ \theta_{\text{pre-trained}}[l, c]. \tag{4.2}$$

We estimate the gains by minimizing an unsupervised loss that only depends on the noisy image:

$$\hat{\gamma} = \arg\min_{\gamma} \quad \mathcal{L}_{\text{GainTuning}}(\mathbf{y}_{\text{test}}, \theta_{\text{GainTuning}}(\gamma)) \tag{4.3}$$

The final denoised image is $f_{\theta_{\text{GainTuning}}(\hat{\gamma})}(\mathbf{y}_{\text{test}})$. Section 4.4 describes several possible choices for the cost function $\mathcal{L}_{\text{GainTuning}}$. Since we use only one scalar parameter per channel, the adjustment performed by GainTuning is very low-dimensional ($\approx$ 0.1% of the dimensionality of θ). This makes optimization quite efficient, and prevents overfitting (see Section C.4). Further, in Section C.4 we show that performing GainTuning provides better performance when compared to fine-tuning only the last few layers of the pre-trained network.

4.4 Cost Functions for GainTuning

A critical element of GainTuning is the use of an unsupervised cost function, which is minimized in order to adapt the pre-trained CNN to the test data. Here, we describe three different choices, each of which are effective for the GainTuning framework, but which have different benefits and limitations.

Blind-spot loss. This loss measures the ability of the denoiser to reproduce the noisy observation, while excluding the identity solution. To achieve this, the CNN must estimate the jth pixel y_j of the noisy image y as a function of the other pixels $y_{\{j\}^c}$, *excluding the pixel itself*. As long as the noise degrades pixels *independently*, the network to learn a nontrivial denoising function that exploits the relationships between pixels arising from the underlying clean image(s). The resulting loss can be written as

$$\mathcal{L}_{\text{blind-spot}}(\mathbf{y}, \theta) = \mathbb{E}\left[(f_\theta(\mathbf{y}_{\{j\}^c})_j - \mathbf{y}_j)^2\right]. \tag{4.4}$$

Here the expectation is over the data distribution and the selected pixel. This "blind spot" can

be enforced through architecture design [68], or by masking [8, 66] (see also [110] and [141] for related approaches). The blind-spot loss has a key property that makes it very powerful in practical applications: it makes no assumption about the noise distribution beyond pixel-wise independence. When combined with GainTuning it achieves effective denoising of real electron-microscope data at very low SNRs (see Figure 4.2 and Section 4.5.4).

Stein's Unbiased Risk Estimator (SURE). Let $\mathbf{x}$ be an N-dimensional ground-truth random vector $\mathbf{x}$ and let $\mathbf{y} := \mathbf{x} + \mathbf{n}$ be a corresponding noisy observation, where $\mathbf{n} \sim \mathcal{N}(0, \sigma_n^2 \mathbf{I})$. SURE provides an expression for the MSE between $\mathbf{x}$ and a denoised estimate $f_\theta(\mathbf{y})$, which *only depends on the noisy observation* $\mathbf{y}$:

$$\mathbb{E}\left[\frac{1}{N}\left\|\mathbf{x} - f_\theta(\mathbf{y})\right\|^2\right] = \mathbb{E}\left[\frac{1}{N}\left\|\mathbf{y} - f_\theta(\mathbf{y})\right\|^2 - \sigma^2 + \frac{2\sigma^2}{N}\sum_{k=1}^{N}\frac{\partial(f_\theta(\mathbf{y})_k)}{\partial \mathbf{y}_k}\right] := \mathcal{L}_{\text{SURE}}(\mathbf{y}, \theta). \tag{4.5}$$

The last term in Equation 4.5 is the divergence of f_θ, which can be approximated using Monte Carlo techniques [112] (Section C.3). The divergence is the sum of the partial derivatives of each denoised pixel with respect to the corresponding input pixel. Intuitively, penalizing it forces the denoiser to not rely as heavily on the jth noisy pixel to estimate the jth clean pixel. This is similar to the blind-spot strategy, with the added benefit that the jth noisy pixel is not ignored completely. To further illustrate this connection, consider a linear convolutional denoising function $f_\theta(\mathbf{y}) = \theta \circledast \mathbf{y}$, where the center-indexed parameter vector is $\theta = [\theta_{-k}, \theta_{-k+1}, \ldots, \theta_0, \ldots, \theta_{k-1}, \theta_k]$. The SURE cost function (Equation 4.5) reduces to

$$\mathbb{E}_{\mathbf{n}}\left[\frac{1}{N}\left\|\mathbf{y} - \theta \circledast \mathbf{y}\right\|^2\right] - \sigma^2 + 2\sigma^2\theta_0 \tag{4.6}$$

The SURE loss equals the MSE between the denoised output and the noisy image, with a penalty on the "self" pixel. As this penalty is increased, the self pixel will be ignored, so the loss tends towards the blind-spot cost function. When integrated into the GainTuning framework, the SURE

loss is limited to additive Gaussian noise, for which it outperforms the blind-spot loss. Extensions of SURE to many other stochastic observation models have been developed [113], and may offer alternative objectives for GainTuning.

Noise Resampling. Ref. [133] introduced a novel procedure for adaptation which we call *noise resampling*. Given a pre-trained denoiser f_θ and a test image $\mathbf{y}$, first one obtains an initial denoised image by applying f_θ to $\mathbf{y}$, $\hat{\mathbf{x}} := f_{\theta_{\text{pre-trained}}}(\mathbf{y})$. This denoised image is then artificially corrupted $\hat{\mathbf{x}}$ by simulating noise from the same distribution as the data of interest to create synthetic noisy examples. Finally, the denoiser is fine-tuned by minimizing the MSE between $\hat{\mathbf{x}}$ and the synthetic examples. If we assume additive noise, the resulting loss is of the form

$$\mathcal{L}_{\text{noise resampling}}(\mathbf{y}, \theta) = \mathbb{E}_n \left[\| (f_\theta(\hat{\mathbf{x}} + \mathbf{n}) - \hat{\mathbf{x}} \|^2 \right]. \tag{4.7}$$

Noise resampling is reminiscent of Refs. [99, 142], which add noise to an already noisy image. When integrated in the GainTuning framework, we find the noise-resampling loss results in effective denoising in the case of additive Gaussian noise, although it generally underperforms the SURE loss.

4.5 Experiments and Results

We performed three different types of experiment to evaluate the performance of GainTuning **In-distribution** (test examples held out from the training set); **out-of-distribution noise** (noise level or distribution of test examples differs from training set); and **out-of-distribution signal** (test images differ in features or context from the training set). We also apply GainTuning to **real data** from a transmission electron microscope.

Our experiments make use of four **datasets**: The BSD400 natural image database [87] with test sets Set12 and Set68 [150], the Urban100 images of urban environments [53], the IUPR dataset

	Model		Set12			BSD68		
			$\sigma = 30$	40	50	30	40	50
GainTuning	DnCNN	Pre-trained	29.52	28.21	27.19	28.39	27.16	26.27
		GainTuning	**29.62**	**28.30**	**27.29**	**28.47**	**27.23**	**26.33**
	UNet	Pre-trained	29.34	28.05	27.05	28.27	27.05	26.15
		GainTuning	29.46	28.15	27.13	28.34	27.12	26.22
Baseline	LIDIA	Pre-trained	29.46	27.95	26.58	28.24	26.91	25.74
		Adapted	29.50	28.10	26.95	28.23	26.97	26.02
	Self2Self		29.21	27.80	26.58	27.83	26.67	25.73

Figure 4.3: GainTuning achieves state-of-the-art performance. (Left) The average PSNR on two test set of generic natural images improves after GainTuning using SURE loss function for different architectures across multiple noise levels. The CNNs are trained on generic natural images (MartinFTM01). (Right) Histograms of improvement in PSNR achieved by DnCNN over test images from Set12 (top) and BSD68 (bottom) at $\sigma = 30$.

of scanned documents [17], and a set of synthetic piecewise constant images [73] (see Section C.1). We demonstrate the broad applicability of GainTuning by using it in conjunction with multiple **architectures for image denoising**: DnCNN [150], BFCNN [96], UNet [116] and Blind-spot net [68] (see Section A.1 and Section B.1.3). Finally, we compare our results to several **benchmarks**: (1) models trained on the training database, (2) CNN models adapted by fine-tuning all parameters (as opposed to just the gains), (3) a model trained only on the test data, (4) LIDIA, a specialized architecture and adaptation strategy proposed in [133]. We provide details on training and optimization in Section C.2.

4.5.1 GainTuning surpasses state-of-the-art performance for in-distribution data

Experimental set-up. We use [87], a standard natural-image benchmark, corrupted with Gaussian white noise with standard deviation σ sampled uniformly from $[0, 55]$ (relative to pixel intensity range $[0, 255]$). Following [150], we evaluate performance on two independent test sets: Set12 and BSD68, corrupted with Gaussian noise with $\sigma \in \{30, 40, 50\}$.

Comparison to pre-trained CNNs. GainTuning consistently improves the performance of pre-

trained CNN models. Figure 4.3 shows this for two different models, DnCNN [150] and UNet [116] (see also Section C.5.1). The SURE loss outperforms the blind-spot loss, and is slightly better than noise resampling (Table C.6). The same holds for other architectures, as reported in Section C.5.1. On average the improvement is modest, but for some images it is quite substantial (up to 0.3 dB in PSNR for $\sigma = 30$, see histogram in Figure 4.3).

Comparison to other baselines. GainTuning outperforms fine-tuning based on optimizing all the parameters for different architectures and loss functions (see Section C.4). GainTuning clearly outperforms a Self2Self model, which is trained exclusively on the test data (Figure 4.3). It also outperforms the specialized architecture and adaptation process introduced in [133], with a larger gap in performance for higher noise levels.

4.5.2 GainTuning generalizes to new noise distributions

Experimental set-up. The same set-up as Section 4.5.1 is used, except that the test sets are corrupted with Gaussian noise with $\sigma \in \{70, 80\}$ (both beyond the training range of $\sigma \in [0, 55]$).

Comparison to pre-trained CNNs. Pre-trained CNN denoisers fail to generalize in this setting. GainTuning consistently improves their performance (see Figure 4.4).

The SURE loss again outperforms the blind-spot loss, and is slightly better than noise resampling (see Section C.5.2). The same holds for other architectures, as reported in Section C.5.2. The improvement in performance for all images is substantial (up to 12 dB in PSNR for $\sigma = 80$, see histogram in Figure 4.4).

Comparison to other baselines. GainTuning achieves comparable performance to a gold-standard CNN trained with supervision at all noise levels (Figure 4.4). GainTuning matches the performance of a bias-free CNN [96] specifically designed to generalize to out-of-distribution noise (Figure 4.4). GainTuning outperforms fine-tuning based on optimizing all the parameters for different architectures and loss functions (see Section C.4). GainTuning clearly outperforms a Self2Self model trained exclusively on the test data (Section C.5.2), and the LIDIA adaptation

Out-of-distribution test noise

Test set	σ	Trained on $\sigma \in [0, 55]$		Bias Free Model [96]	Trained on $\sigma \in [0, 100]$
		Pre-trained	Gaintuning		
Set12	70	22.45	25.54	25.59	25.50
	80	18.48	24.57	24.94	24.88
BSD68	70	22.15	24.89	24.87	24.88
	80	18.72	24.14	24.38	24.36

Out-of-distribution test image

	Training data	Test data	Pre-trained	Gaintuning
(a)	Piecewise constant	Natural images	27.31	28.60
(b)	Natural images	Urban images	28.35	28.79
(c)	Natural images	Scanned documents	30.02	30.73

Figure 4.4: GainTuning generalizes to out-of-distribution data. Average performance of a CNN trained to denoise at noise levels $\sigma \in [0, 55]$ improves significantly on test image with noise outside the training range, $\sigma = 70, 80$ (top) and on images with different characteristics than training data (bottom) after GainTuning. Capability of GainTuning to generalize to out-of-distribution noise is comparable to that of Bias-Free CNN [96], which is an architecture explicitly designed to generalize to noise levels outside the training range, and to that of a denoiser trained with supervision at all noise levels. (Right) Histogram showing improvement in performance for each image in the test set. The improvement is substantial across most images, reaching nearly 12dB improvement in one example. For these examples, the denoiser was DnCNN (with additive bias terms) and the GainTuning loss function was SURE. See Section C.5.2 for experiments with other CNN architectures and loss functions.

method [133].

Gaussian to Poisson generalization: Section C.5.2 and Figure C.4 show that GainTuning can effectively adapt a CNN pre-trained for Gaussian noise removal to restore images corrupted with Poisson noise as well.

4.5.3 GainTuning generalizes to out-of-distribution image content

Experimental set-up. We evaluate the performance of GainTuning on test images that have different characteristics from the training images. We perform the following controlled experiments:

(a) Simulated piecewise constant images → Natural images. We pre-train CNN denoisers

on simulated piecewise constant images. These images consists of constant regions (of different intensities values) with the boundaries having varied shapes such as circle and lines with different orientations (see Section C.1 for some examples). Piecewise constant images provide a crude model for natural images [73, 88, 105]. We use GainTuning to adapt a CNN trained on this dataset to generic natural images (Set12). This experiment demonstrates the ability of GainTuning to adapt from a simple simulated dataset to a significantly more complex real dataset.

(b) Generic natural images → Images with high self-similarity. We apply GainTuning to adapt a CNN trained on generic natural images to images in Urban100 dataset. Urban100 consists of images of buildings and other structures typically found in an urban setting, which contain substantially more repeating/periodic structure (see Section C.1) than generic natural images.

(c) Generic natural images → Images of scanned documents. We apply GainTuning to adapt a CNN trained on generic natural images to images of scanned documents in IUPR dataset (see Section C.1).

All CNNs were trained for denoising Gaussian white noise with standard deviation $\sigma \in [0, 55]$ and evaluated at $\sigma = 30$.

Comparison to pre-trained CNNs. GainTuning consistently improves the performance of pre-trained CNNs in all the three experiments. Figure 4.4 shows this for DnCNN when GainTuning is based on SURE loss. We obtain similar results with other architectures (see Section C.5.3). In experiment (a), all test images show substantial improvements over the pre-trained results (average increase of roughly 1.3dB, and best case more than 3 dB, at $\sigma = 30$). We observe similar trends for experiments (b) and (c) as well, with improvements being better on an average for experiment (c). Note that we obtain similar performance increases when both *image and noise are out-of-distribution* as discussed in Section C.5.4.

Comparison to other baselines. In experiment (a), GainTuning outperforms methods that optimize all parameters over different architectures and loss functions (Section C.4). However, Self2Self trained only on test data outperforms GainTuningin this case, because the test images contain content that differs substantially from the training images. Self2Self provides the strongest form of adaptation, since it is trained exclusively on the test image, whereas the denoising properties of GainTuning are partially due to the pretraining (see Sections 4.7, C.5.3). We did not evaluate LIDIA [133] for this experiment. For experiments (b) and (c), training all parameters clearly outperforms GainTuning for case (b), but has similar performance for (c). GainTuning outperforms LIDIA on experiments (b) and (c). Self2Self trained exclusively on test data outperforms GainTuning (and LIDIA) on (b) and (c) (see Sections 4.7, C.5.3).

4.5.4 Application to Electron microscopy

Scientific motivation. Transmission electron microscopy (TEM) is a popular imaging technique in materials science [122, 128]. Recent advancements in TEM enable to image at high frame rates [37, 38]. These images can for example capture the dynamic, atomic-level rearrangements of catalytic systems [24, 45, 69, 75, 125], which is critical to advance our understanding of functional materials. Acquiring image series at such high temporal resolution produces data severely degraded by shot noise. Consequently, there is an acute need for denoising in this domain.

The need for adaptive denoising. Ground-truth images are not available in TEM, because measuring at high SNR is often impossible. Prior work has addressed this by using simulated training data (see Chapter 5) [95, 135], whereas others have trained CNNs directly on noisy real data (Chapter 3) [119].

Dataset. We use the training set of 5583 simulated images and the test set of 40 real TEM images from [95, 135]. The data correspond to a catalytic platinum nanoparticle on a CeO_2 support (Section 5.4)).

Comparison to pre-trained CNN. A CNN [68] pre-trained on the simulated data fails to re-

construct the pattern of atoms faithfully (green box in Figure 4.2 (c), (e)). GainTuning applied to this CNN using the blind-spot loss correctly recovers this pattern (green box in Figure 4.2 (d), (e)) reconstructing the small oxygen atoms in the CeO_2 support. GainTuning with noise resampling failed to reproduce the support pattern (probably because it is absent from the initial denoised estimate) (see Ref. [94]).

Comparison to other baselines. GainTuning clearly outperforms Self2Self, which is trained exclusively on the real data. The denoised image from Self2Self shows missing atoms and substantial artefacts (see Figure 4.2). We also compare GainTuning dataset to blind-spot methods using the 40 test frames [68, 119]. GainTuning clearly outperforms these methods. Finally, GainTuning outperforms fine-tuning based on optimizing all the parameters, which overfits heavily (see Section C.4).

4.6 Analysis

In this section, we perform a qualitative analysis of the properties of GainTuning.

Which images benefit most from GainTuning adaptation? Figure 4.5 shows the images in the different test datasets for which GainTuning achieves the most and the least improvement in PSNR. The result is quite consistent over multiple architectures: the improvement in performance achieved by GainTuning is larger if the test image contains highly repetitive patterns. This makes intuitive sense; the repetitions effectively provide multiple examples from which to learn these patterns during the unsupervised refinement.

Generalization via GainTuning. We investigate the generalization capability of GainTuning. We observe that a CNN adapted to a particular image via GainTuning generalizes effectively to other similar images. Figure 4.6 shows that GainTuning can achieve generalization to images that are similar to the test image used for adaptation on two examples: (1) adapting a network to an image of a scanned document generalizes to other scanned documents, and (2) adapting

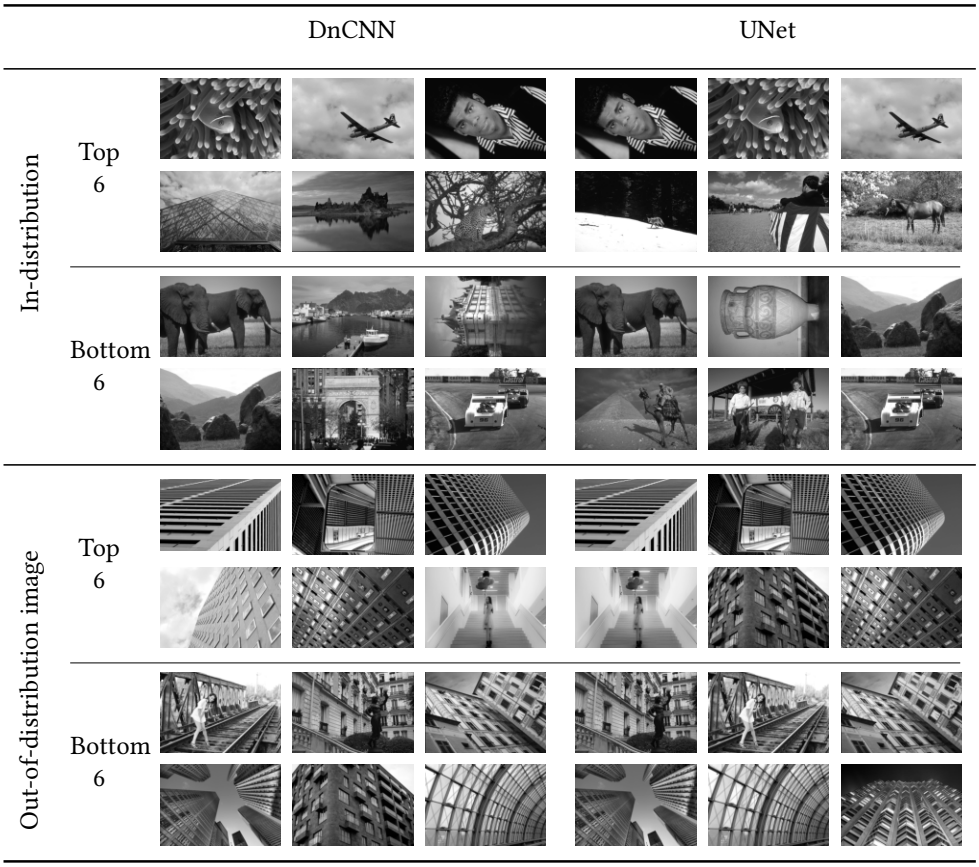


Figure 4.5: What kind of images benefit the most from adaptive denoising? We visualize the images which achieve the top 6 and bottom 6 (left top to the right bottom of each grid) improvement in performance (in PSNR) after performing GainTuningİmages with the largest improvement in performance often have highly repetitive patterns or large regions with constant intensity. Images with least improvement in performance tend to have more heterogeneous structure. Note that, in general, the distribution of improvements in performance is often skewed towards the images with lowest improvement in performance (See Figures 4.3, 4.4, and 4.8).

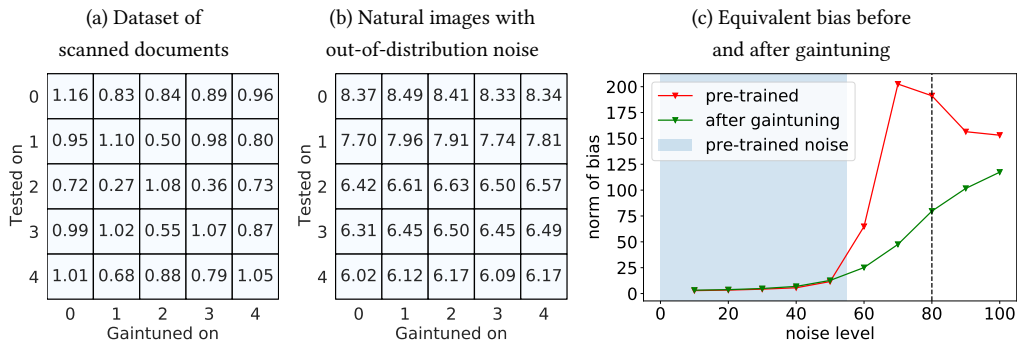


Figure 4.6: Analysis of GainTuning. GainTuning can achieve generalization to images that are similar to the test image used for adaptation. We show this through two examples: (a) adapting a network to an image of a scanned document generalizes to other scanned documents, and (b) adapting a a network to an image with out-of-distribution noise generalizes to other images with similar noise statistics. The $(i, j)^{\text{th}}$ entry of the matrix in (a) and (b) represents the improvement in performance (measured in PNSR) when a CNN GainTuned on image j is used to denoise image i. We use 5 images with the largest improvement in performance across the dataset for (a) and (b). Finally, (c) shows that generalization to noise levels outside the training range is enabled by reducing the *equivalent bias* of the pre-trained CNN (see equation (2.1)).

a a network to an image with out-of-distribution noise generalizes to other images with similar noise statistics.

How does GainTuning adapt to out-of-distribution noise? Generalization to out-of-distribution noise provides a unique opportunity to understand how GainTuning modifies the denoising function. Section 2.5 shows that the first-order Taylor approximation of denoising CNNs trained on multiple noise levels tend to have a negligible constant term, and that the growth of this term is the primary culprit for the failure of these models when tested on new noise levels. GainTuning reduces the amplitude of this constant term, facilitating generalization.

Let $y \in \mathbb{R}^N$ be a noisy image processed by a CNN. Using the first-order Taylor approximation, the function $f : \mathbb{R}^N \to \mathbb{R}^N$ computed by a denoising CNN may be expressed as an affine function

$$f(z) = f(y) + A_y(z - y) = A_y z + b_y, \tag{4.8}$$

where $A_y \in \mathbb{R}^{N \times N}$ is the Jacobian of $f(\cdot)$ evaluated at input y, and $b_y \in \mathbb{R}^N$ represents the *net*

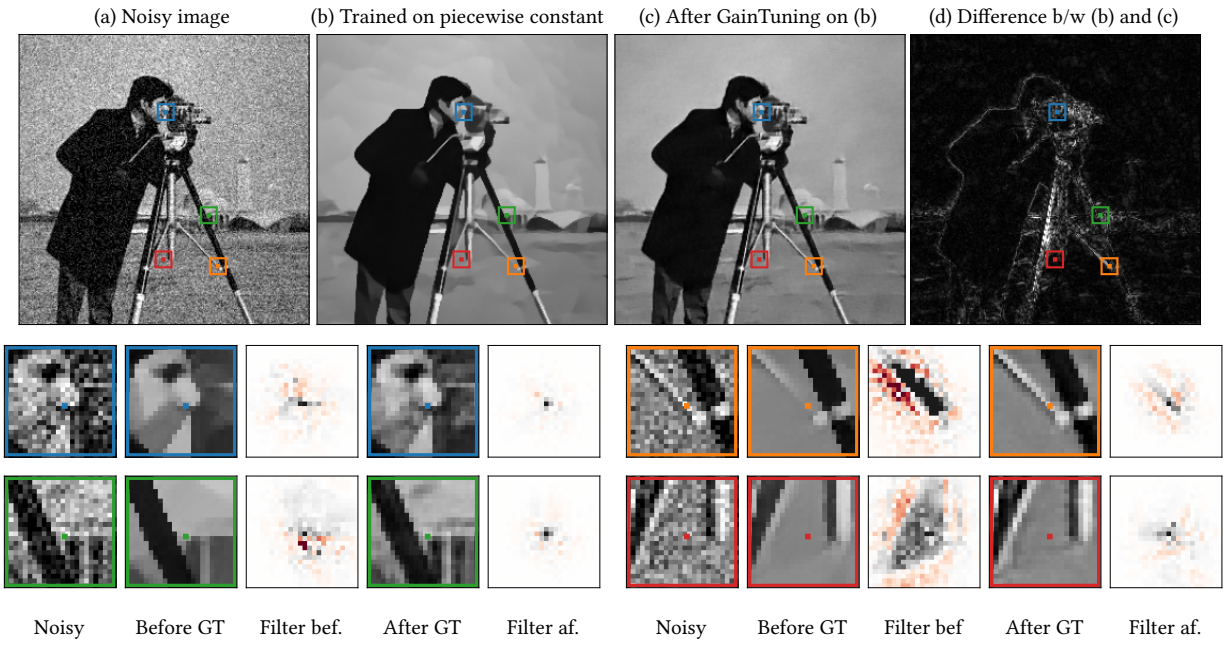


Figure 4.7: Adaptation to new image content. (Top) A Bias-free CNN [96] pre-trained on piecewise constant images applied to a natural test image (a) oversmooths the image and blurs the details (b), but is able to recover more detail after applying GainTuning using SURE loss function (c). (Bottom) The CNN estimates a denoised pixel (colored pixel at the center of each image) as a linear combination of the noisy input pixels. The weighting functions (filters) of pre-trained CNN are dispersed, consistent with the training set. However, after GainTuning, the weighting functions are more precisely targeted to the local features, resulting in better recovery of details in the denoised image (c).

bias. In [96], it was shown that the bias tends to be small for CNNs trained to denoise natural images corrupted by additive Gaussian noise, but is a primary cause of failures to generalize to noise levels not encountered during training. Figure 4.6 shows that GainTuning reduces the net bias of CNN, facilitating the generalization to new noise levels.

How does GainTuning adapt to out-of-distribution images? In order to understand how GainTuning adapt to out-of-distribution images, we examine the adaptation of a CNN pre-trained on piecewise constant to natural images. Piecewise constant images have large areas with constant intensities, therefore, CNNs trained on these images tends to average over large areas. This is true even when the test image contains detailed structures. We verify this by forming the affine approximation of the network (eq. 4.8) and visualizing the equivalent linear filter [96], as

explained below:

Let $y \in \mathbb{R}^N$ be a noisy image processed by a CNN. We process the test image using a Bias-Free CNN [96] so that the net bias b_y is zero in its first-order Taylor decomposition (4.8). When $b_y = 0$, (4.8) implies that the ith pixel of the output image is computed as an inner product between the ith row of A_y, denoted $a_y(i)$, and the input image:

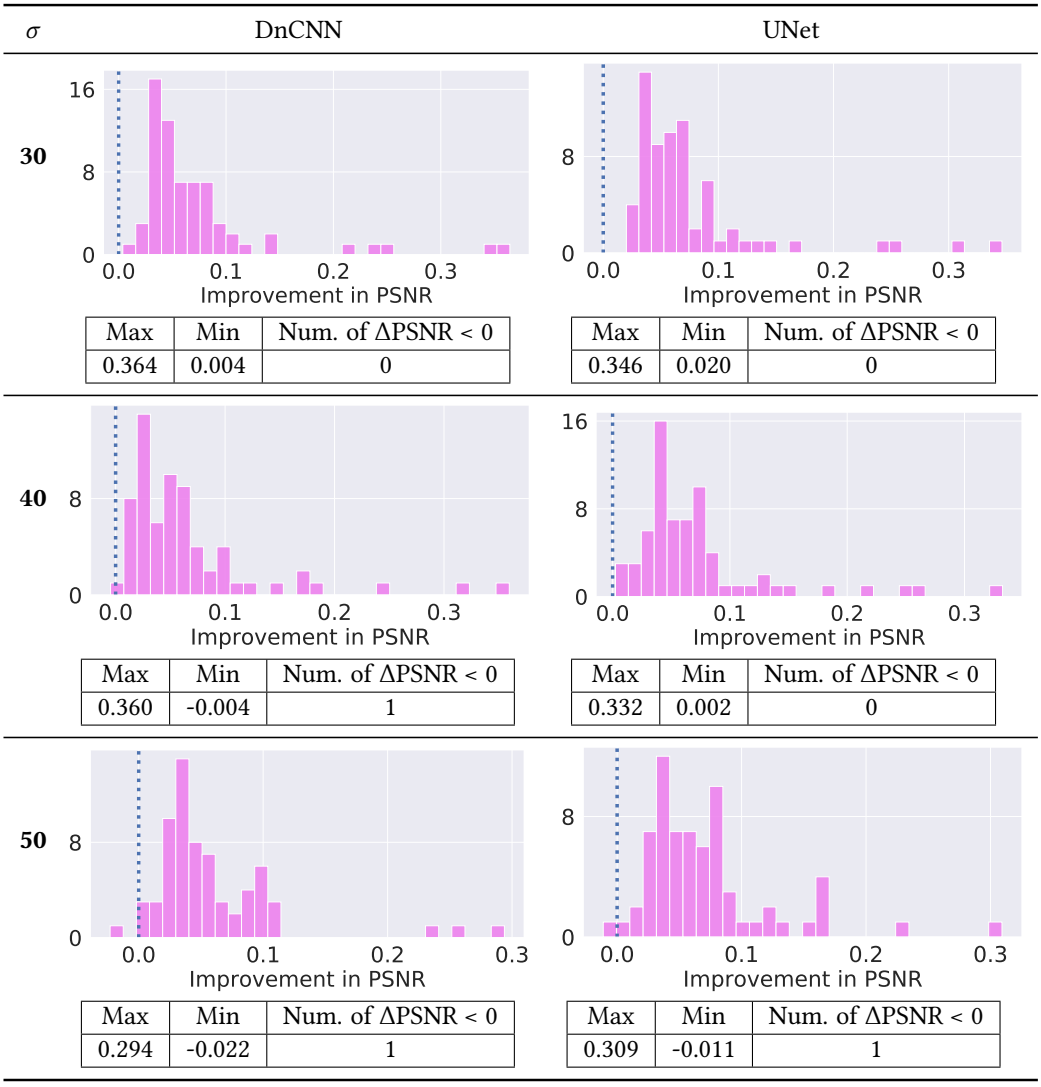
$$f(y)(i) = \sum_{j=1}^{N} A_y(i,j)y(j) = a_y(i)^T y. \tag{4.9}$$

The vectors $a_y(i)$ can be interpreted as *adaptive filters* that produce an estimate of the denoised pixel via a weighted average of noisy pixels. As shown in Figure 4.7 the denoised output of CNN pre-trained on piece wise constant images is over-smoothed and the filters average over larger areas. After GainTuning the model learns to preserve the fine features much better, which is reflected in the equivalent filters.

4.7 Limitations

As shown in Section 4.5, GainTuning improves the state of the art on benchmark datasets, adapts well to out-of-distribution noise and image content, and outperforms all existing methods on an application to real world electron-microscope data. A crucial component in the success of GainTuning is restricting the parameters that are optimized at test time. However, this constraint also limits the potential improvement in performance one can achieve, as seen when fine-tuning for test images from the Urban100 and IUPR datasets, each of which contain many images with highly repetitive structure. In these cases, we observe that fine-tuning all parameters, and even training only on the test data using Self2Self can outperform GainTuning. This raises the question of how to effectively leverage training datasets for such images.

In addition, when the pre-trained denoiser is highly optimized, and the test image is within



Max	Min	Num. of ΔPSNR < 0
0.364	0.004	0

Max	Min	Num. of ΔPSNR < 0
0.346	0.020	0

Max	Min	Num. of ΔPSNR < 0
0.360	-0.004	1

Max	Min	Num. of ΔPSNR < 0
0.332	0.002	0

Max	Min	Num. of ΔPSNR < 0
0.294	-0.022	1

Max	Min	Num. of ΔPSNR < 0
0.309	-0.011	1

Figure 4.8: Distribution of PSNR improvement on in-distribution test set. Distribution of improvement on BSD68 dataset at noise levels $\sigma = \{30, 40, 50\}$ (in-distribution). When the network is well optimized, and the test image is in-distribution, GainTuning can sometimes degrade the performance of the network. This degradation is atypical (in this figure, there are only 3 occurrences of degradation out of 408 experiments), and very small (in this figure, the maximum degradation is 0.022)

distribution, GainTuning occasionally causes a slight degradation of performance. This is atypical (3 occurrences in 412 GainTuning experiments using DnCNN and SURE), and the decreases are quite small (maximum PSNR degradation of about 0.02dB, compared to maximum improvement of nearly 12dB; see Figure 4.8).

4.8 Conclusions

We've introduced GainTuning an adaptive denoising methodology for adaptively fine-tuning a pre-trained CNN denoiser on individual test images. The method, which is general enough to be used with any denoising CNN, improves the performance of state-of-the-art CNNs on standard denoising benchmarks, and provides even more substantial improvements when the test data differ systematically from the training data, either in noise level, noise type, or image type. We demonstrate the potential of adaptive denoising in scientific imaging through an application to electron microscopy. Here, GainTuning is able to jointly exploit synthetic data and test-time adaptation to reconstruct meaningful structure (the atomic configuration of a nanoparticle and its support), which cannot be recovered through alternative approaches. A concrete challenge for future research is to combine the unsupervised denoising strategy of Self2Self, which relies heavily on dropout and ensembling, with pre-trained models. More generally, it is of interest to explore whether GainTuning can provide benefits for other image-processing tasks.

5 | Application to electron microscopy data

This chapter is adapted from the papers "Deep Denoising for Scientific Discovery: A Case Study in Electron Microscopy" published in Transactions of Computational Imaging [95], and "Developing and Evaluating Deep Neural Network-based Denoising for Nanoparticle TEM Images with Ultra-low Signal-to-Noise" published in Microscopy and Microanalysis, 2021 [135]. This is a joint work with Ramon Manzorro, Joshua L. Vincent, Binh Tang, Dev Yashpal Sheth, Eero P. Simoncelli, David S. Matteson, Peter A. Crozier, and Carlos Fernandez-Granda.

In Chapters 2 to 4, we developed learning based denoising strategies primarily focusing on natural images. In this chapter, we will apply these techniques to a dataset containing images of catalytic nanoparticles acquired through an electron microscope. In Chapter 3, we directly applied the unsupervised denoising technique to this dataset. We take a different approach in this chapter - we first establish a baseline by creating a simulated dataset containing ground truth labels to train networks with supervision. We show that such simulation based approach can outperform unsupervised methods in extreme low data regime. The network used for GainTuning experiments on the electron microscopy data in Chapter 4 is also developed in this chapter. Further, we use the analysis tools developed in Chapter 2 to understand why certain network architectures that achieve state-of-the-art performance on natural images fail to perform well on this dataset. Guided by this analysis, we propose a modified network architecture which achieves

state-of-the-art performance.

ABSTRACT

Denoising is a fundamental challenge in scientific imaging. Deep convolutional neural networks (CNNs) provide the current state of the art in denoising natural images, where they produce impressive results. However, their potential has been inadequately explored in the context of scientific imaging. Denoising CNNs are typically trained on real natural images artificially corrupted with simulated noise. In contrast, in scientific applications, noiseless ground-truth images are usually not available. To address this issue, we propose a simulation-based denoising (SBD) framework, in which CNNs are trained on simulated images. We test the framework on data obtained from transmission electron microscopy (TEM), an imaging technique with widespread applications in material science, biology, and medicine. SBD outperforms existing techniques by a wide margin on a simulated benchmark dataset, as well as on real data. We analyze the generalization capability of SBD, demonstrating that the trained networks are robust to variations in imaging parameters and of the underlying signal structure. Our results reveal that state-of-the-art architectures for denoising photographic images may not be well adapted to scientific-imaging data. For instance, substantially increasing their field-of-view dramatically improves their performance on TEM images acquired at low signal-to-noise ratios. We also demonstrate that standard performance metrics for photographs (such as PSNR and SSIM) may fail to produce scientifically meaningful evaluation. We propose several metrics to remedy this issue for the case of atomic resolution electron microscope images. In addition, we propose a technique, based on likelihood computations, to visualize the agreement between the structure of the denoised images and the observed data. Finally, we release a publicly available benchmark dataset of TEM images, containing $18,000$ examples.

5.1 Overview

Imaging technology is an essential tool in many scientific domains. Electron microscopy enables the visualization of atomic structures [156], fluorescence microscopy makes it possible to study cellular processes [77], and telescopes reveal galaxies and other astronomical objects that are light years away [89]. In all these modalities, images are corrupted by noise associated with stochastic processes occurring during signal generation and detection, degrading the information content of the image data. The general goal of denoising is to estimate and restore the information missing from these noisy observations, thus facilitating the extraction of useful scientific information.

In the past decade, convolutional neural networks (CNNs) [72] have achieved state-of-the-art performance in image denoising [21, 150]. However, the potential of this methodology has barely been explored in the context of scientific imaging. In the vast majority of the existing work, noisy data are generated by adding Gaussian noise to clean photographs. The CNNs are then trained to approximate the ground-truth images from these measurements, usually by minimizing mean squared error [150]. Unfortunately, *this paradigm is not adequate for most scientific domains*, where large, labeled datasets of ground-truth clean data are typically not available. To address this issue, we propose a simulation-based denoising (SBD) framework, in which CNNs are trained on simulated images. We validate our methodology through a case study in transmission electron microscopy.

Transmission electron microscopy (TEM) is a powerful and versatile characterization technique used to probe the atomic-level structure and composition of a wide range of materials, such as catalysts or semiconductors [122, 128]. The technique has had a huge impact in structural biology, as recognized with the award of the 2017 Nobel Prize in Chemistry [3]. Recent advancements in direct electron detection systems enable experimentalists to image dynamic events at frame rates in the kilohertz range [37, 38]. Imaging at these time scales is critical to advance our

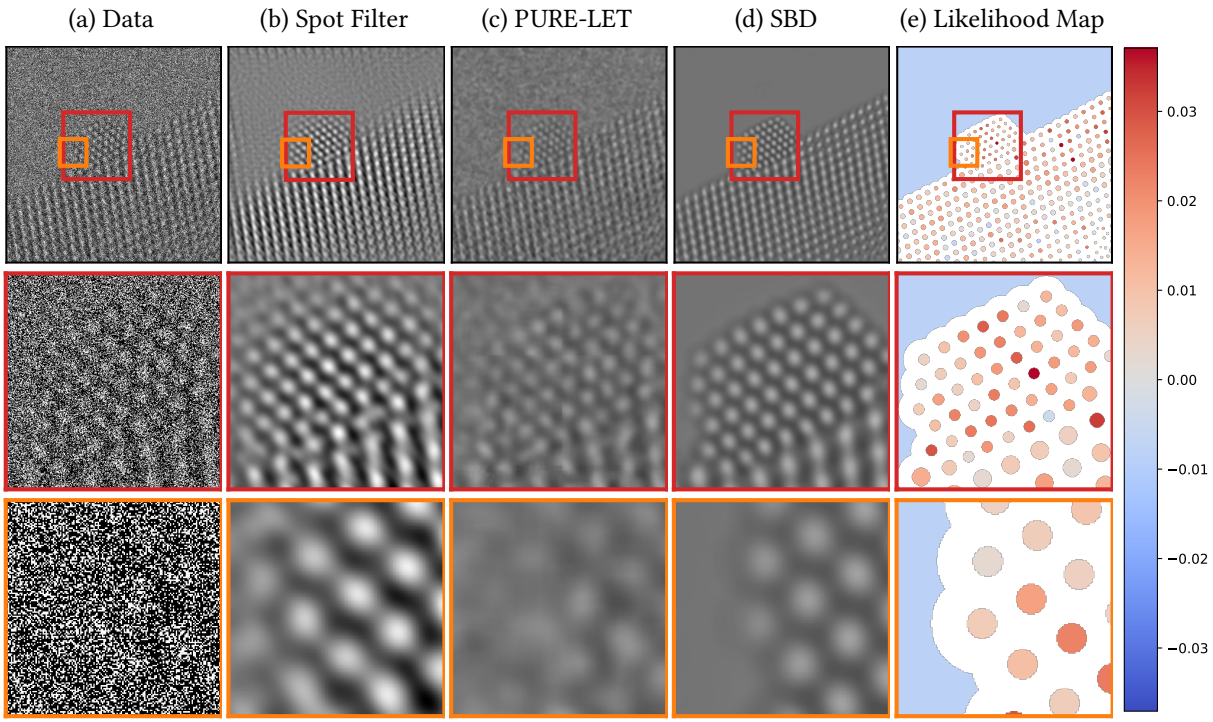


Figure 5.1: Denoising results for real data. (a) An experimentally-acquired atomic-resolution transmission electron microscope image of a CeO2-supported Pt nanoparticle. A description of the experimental data acquisition is given in Section 5.4. The average image intensity is 0.45 electrons/pixel (i.e., a large fraction of pixels register zero electrons), which results in an extremely low signal-to-noise ratio. (b) Denoised image obtained via Fourier-based filtering by a domain expert. (c) Denoised image obtained via the wavelet-based PURE-LET method [82]. (d) Denoised image obtained by the proposed simulation-based denoising (SBD) framework. (e) Likelihood map quantifying to what extent the atomic structure identified from the SBD denoised image is consistent with the data (see Section 5.3). Regions in red are more likely to correspond to atomic columns in the nanoparticle. Regions in blue are more likely to belong to the vacuum.

understanding of functional materials. In catalytic systems, for example, the chemical transformation process is accompanied by dynamic, atomic-level structural rearrangements which may occur over a time scale spanning tens of milliseconds [24, 45, 69, 75, 125]. Acquiring image series at such high temporal resolution necessarily produces datasets that are severely degraded by shot noise, rendering traditional imaging processing approaches ineffective. It is typically not feasible to reduce the noise content by increasing the intensity of the incident electron beam, since the high-energy beam can also damage the material when exposed to high doses. Consequently, there is an acute need for novel denoising technology in this domain.

In order to apply the proposed SBD framework to TEM data, we generate a large simulated dataset of TEM images, containing 18,000 examples, and use it to train CNNs for noise removal. This approach outperforms existing techniques by a wide margin on held-out simulated data, as well as on real TEM measurements (see Figures 5.1, 5.9 and 5.13 and Sections 5.5.2 and 5.5.4). We perform a thorough analysis of the generalization capability of our models, demonstrating that the CNNs are robust to variations of imaging parameters and of underlying signal structure. Our results indicate that architectures optimized for natural photographic images may have fundamental shortcomings when applied to domain-specific data. For instance, we show that substantially increasing the field-of-view of denoising CNNs has almost no effect on photographs, but produces a significant boost in performance for TEM images. We also demonstrate that standard performance metrics for photographs, such as peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) [138], often fail to produce a scientifically-meaningful evaluation of the denoising results. For example, the presence or absence of a single atomic column often results in a negligible change in these metrics. This is highly problematic, because detecting these columns is one of the main motivations for our case study. To remedy this issue, we propose several scientifically-motivated metrics to evaluate our results (see Section 5.5.3). In addition, we propose a likelihood-based visualization of the agreement between the observed measurements and structures of interest (such as atomic columns) in the denoised image. This visualization can be

used to flag denoising artefacts, which may be mistaken for scientifically-relevant structure (see Figure 5.5). Finally, to encourage further development of deep-learning methodologies for scientific imaging, we have made our benchmark dataset of TEM images publicly available online[1]. More details on applying the proposed methodology to TEM data, and domain-specific insights derived from the denoised images are described in Ref. [135].

5.2 Related work

Denoising in scientific imaging

A wide variety of denoising methods have been applied across different scientific imaging modalities, including traditional linear filters [101], nonlinear filters [57, 92, 131], wavelet-based methods [19, 90, 107, 155], and sparsity-based approaches [9, 90]. Deep convolutional networks have been shown to outperform all of these approaches in photographic images [21, 150]. The rapidly growing literature on this methodology focuses almost exclusively on photographic images. We are aware of only a few very recent exceptions. In the medical domain, CNN-based denoising has been applied to low-dose computer tomography [61], positron-emission tomography [43] and scintillation-camera data [93]. Refs. [40, 134] apply CNNs to denoise simulated electron microscopy data, without validating on real data. Ref.[32] train CNN to denoise by adding synthetic noise on high-quality electron micrograph data. Ref. [86] trains CNNs to denoise Raman scattering microscopy data, using measurements gathered at a higher signal-to-noise ratio (SNR) as ground-truth images. These results showcase the potential of deep denoising for scientific imaging, but also the challenge of gathering adequate datasets to train the deep networks. In this work, we propose to address this challenge by training denoising CNNs on carefully-designed simulated datasets, and validate our approach on experimental measurements.

[1] https://sreyas-mohan.github.io/electron-microscopy-denoising/

Unsupervised denoising

Unsupervised denoising is a promising approach for applications where ground-truth images are not available. Unsupervised methods based on wavelets have achieved performance comparable to their supervised counterparts on photographic images [49, 81, 115].
Noise2Noise [74], a deep-learning approach that requires access to pairs of noisy images corresponding to the same underlying signal, has been applied to cryo-electron microscopy [15]. More recent methods can be trained directly on noisy images [8, 66, 68]. Several recent works apply this approach to fluorescence microscopy data [59, 67, 108, 152]. In the case of our TEM data, standard unsupervised methods do not perform as well as the proposed supervised approach (see Section 5.5.4.1). This is possibly due to the limited number of training data (see Figure 5.15) and to the low input SNR. The SNR of our TEM data (around 3 dB) is orders of magnitude lower than that reported in typical unsupervised denoising works (e.g. around 27 dB for [152]).

Deep Learning for TEM

Deep CNNs have been applied to other image-processing tasks in TEM beyond denoising, see Ref. [33] for a comprehensive review. Ref. [126] proposes a CNN-based method for TEM image super-resolution, wherein CNNs are trained on pairs of low-resolution and high-resolution images acquired experimentally. Ref. [51] applies CNNs to perform segmentation and systematically studies the influence of the design of the training dataset and network architecture on the generalization capabilities of these models. In this work, we provide a similar analysis for denoising. Refs. [83] and [111] propose a CNN-based method to identify structures of interest in TEM images. They train on carefully designed simulated data and show that the model generalizes to real data. Our work provides further evidence that CNNs trained on simulated data can generalize effectively to real measurements.

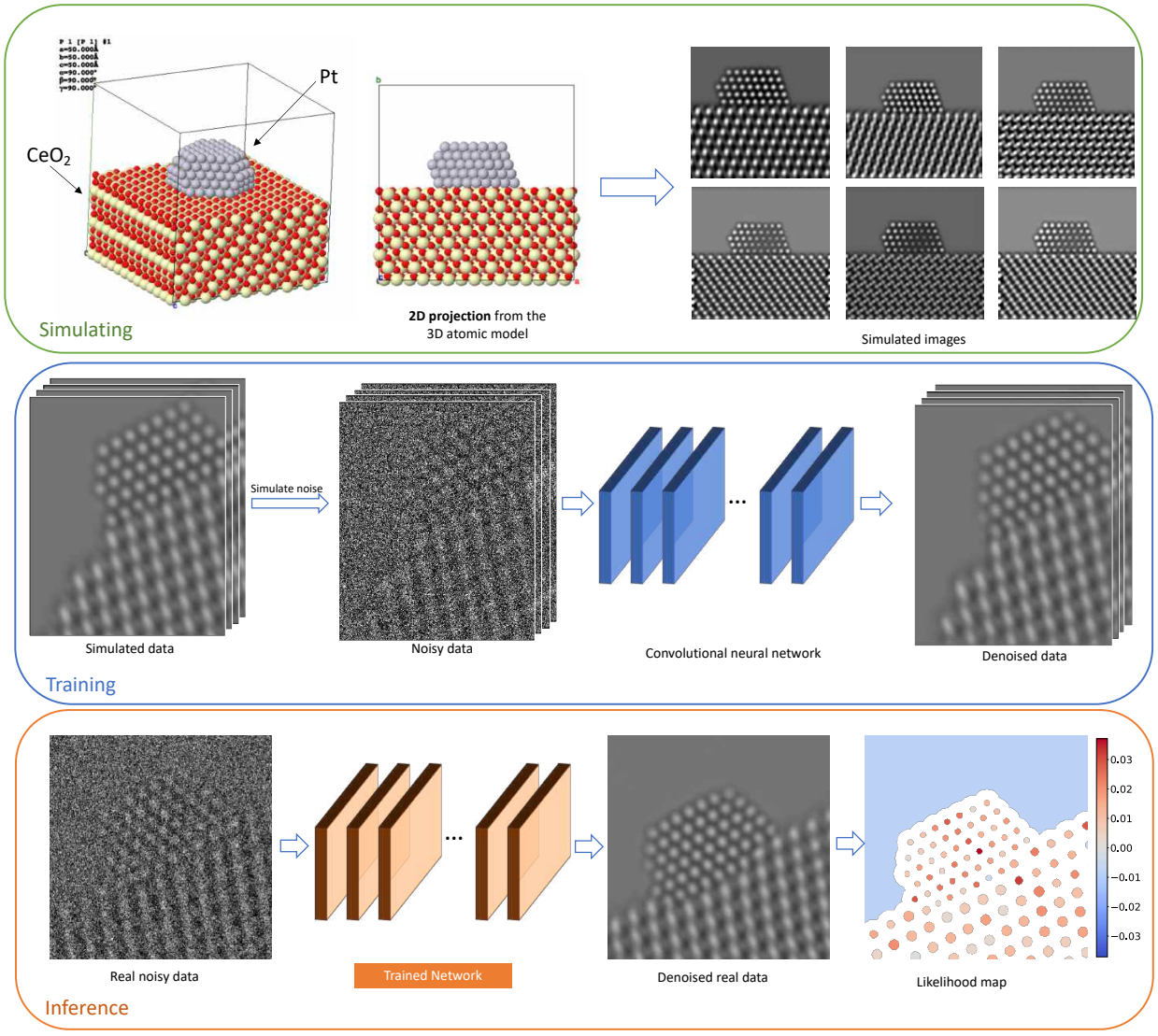


Figure 5.2: Simulation-based denoising framework. (Top) A training dataset is generated by simulating TEM images of different structures at varying imaging conditions. Here we focus on structures of Pt nanoparticles supported on CeO_2. (Middle) A CNN is trained using the simulated images, paired with noisy counterparts obtained by simulating the relevant noise process. (Bottom) The trained CNN is applied to real data to yield a denoised image. After analyzing the image to extract structures of interest, a likelihood map is generated to quantify the agreement between this structure and the noisy data.

5.3 Methodology

5.3.1 Simulation-based denoising

Current state-of-the-art deep-learning techniques for denoising photographic images require a training set of ground-truth images [150]. Typically these clean images are corrupted with additive Gaussian noise, and the CNNs are trained to minimize the mean squared error between the network output and the original images. The main obstacle to leveraging this approach in scientific imaging is the lack of ground-truth data; in many applications there is no such thing as a *clean image*. We address this by using a dataset of simulated images to train the CNNs. We call this framework simulation-based denoising (SBD).

Simulation-based denoising (SBD) consists of three stages: simulation of the training set, training of the CNNs using the simulated data, and inference on the real data (see Figure 5.2 for an overview of the methodology). In order to generate the training set, we simulate clean images $x_1, \ldots, x_N \in \mathbb{R}^M$ (where M is the number of pixels) according to a predefined physical model. These clean images are then corrupted using a noise model, which can follow a predefined model or be learned from the data, to generate the simulated noisy data. We provide a detailed account of how we generate the simulated dataset for our case study in Sections 5.4 and D.1.1, and of the noise model in Section 5.4.3. Let $Y(x_i)$ denote the random vector representing the noisy image corresponding to the clean simulated image x_i and let $y(x_i)$ represent a realization of $Y(x_i)$. We parameterize the denoising function as a CNN $f_\theta : \mathbb{R}^M \to \mathbb{R}^M$ where the parameters θ are the weights of the network. To find a good denoising function f_θ, we minimize a loss function $\mathcal{L} : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ which quantifies how close the estimate from the CNN $f_\theta(y(x_i))$ is to the clean image x_i. In our case study, we use mean squared error, which is a standard choice in CNN-based denoising [150]. More concretely, during the training stage, we compute the parameters

by solving

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}\left[\sum_{i=1}^{N} \mathcal{L}(f_\theta(Y(x_i)), x_i)\right] = \arg\min_{\theta} \mathbb{E}\left[\sum_{i=1}^{N} \|f_\theta(Y(x_i)) - x_i\|_2^2\right] \tag{5.1}$$

We perform minimization iteratively using a variant of stochastic gradient descent. We approximate the expectation in Equation 5.1 by sampling new realizations of the noisy image $Y(x_i)$ every time we compute the gradient. Once the network is trained, it can be directly applied to new noisy images to perform denoising.

A crucial difference between SBD and previous methodology for deep denoising is that the training set needs to be explicitly *designed*. In order to ensure effective generalization to real data, we must include sufficient variation of imaging parameters and image structure in the training dataset. In addition, particular care is needed to enforce invariance to small changes in the geometry of the image. Figure 5.3 shows that a denoising CNN can easily overfit the specific alignment and scale of the training data. This issue can be addressed by augmenting the training set with rotated and scaled versions of the simulated images. Determining how to optimally sample the space of possible simulation parameters when generating data to train CNNs for denoising is an important methodological question for future research.

5.3.2 Exploiting non-local signal structure

Images in scientific applications often have pixel-intensity distributions that differ significantly from those of natural images. Our case study shows that it is crucial to take this into account in order to achieve successful denoising. Current state-of-the-art networks for denoising photographic images have very small fields of view. For example, the field of view of DnCNN [150] and DURR [151] are 41×41 pixels, and 45×45 pixels respectively. Unlike photographic images, the TEM images in our case study exhibit very prominent global regularities, due to periodicity in the atomic structure of the imaged materials. In addition, electron-microscopy images are

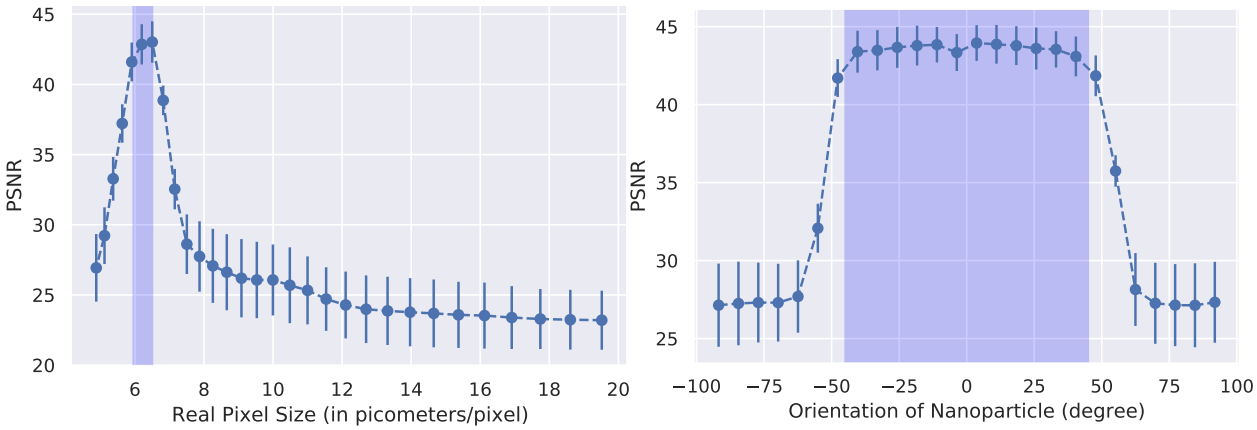


Figure 5.3: Overfitting scaling and orientation. The plots show the performance of the proposed network in PSNR for simulated images that have different scalings (on the left, measured in terms of the corresponding pixel size in picometers) and orientations (on the right). The CNN was trained on data augmented with rescaled and rotated images corresponding to the regions shaded in purple. When tested out of those regions, the denoising performance deteriorates significantly. This shows that careful data augmentation is required to ensure invariance to scaling and orientation.

often measured at very low SNRs (in our case, the SNR for the real TEM data is about 3 dB, but most works for photographic images focus on an SNR above 22 dB, see e.g. [150]). As the SNR decreases, denoising CNNs tend to average over larger regions of the surrounding pixels, as demonstrated in Ref. [96] (qualitatively, this is the same behavior observed in a classical linear Wiener filter [140]). These considerations motivate using networks with large field of view to denoise TEM data.

Here we propose to denoise TEM data using deep networks with very large fields of view: 221×221 pixels and 893×893 pixels, a 25-fold and 400-fold increase with respect to generic denoising architectures respectively. In order to obtain a large field of view efficiently (i.e. without dramatically increasing the number of parameters in the network), we propose using a UNet network architecture [116]. We use 4 downsampling operations to achieve the 221×221 field of view and 6 downsampling operations to achieve the 893×893 field of view (see Section **??** for a detailed description of the architecture). Table 5.1 compares the influence of the field of view in denoising photographic and TEM images. For photographic images the performance of the network remains almost constant as we increase the field of view. In contrast, for TEM

(a) TEM Images

Model	Parameters	FoV	PSNR	SSIM
SBD + DnCNN [150]	668K	41×41	30.47 ± 0.64	0.93 ± 0.01
SBD + Small UNet [151]	233K	45×45	30.87 ± 0.56	0.93 ± 0.01
SBD + UNet (32 base channels)	352K	221×221	36.39 ± 0.77	0.98 ± 0.01
SBD + UNet (64 base channels)	1.41M	221×221	37.24 ± 0.76	0.99 ± 0.01
SBD + UNet (128 base channels)	5.61M	221×221	38.05 ± 0.81	0.99 ± 0.01
SBD + UNet (128 base channels)	70.15M	893×893	42.87 ± 1.45	0.99 ± 0.01

(b) Photographic Images

Model	Parameters	FoV	PSNR		SSIM	
			$\sigma = 30$	$\sigma = 70$	$\sigma = 30$	$\sigma = 70$
UNet	102K	49×49	29.67 ± 2.84	26.16 ± 2.79	0.83 ± 0.06	0.70 ± 0.09
UNet	352K	221×221	29.65 ± 2.76	26.08 ± 2.68	0.83 ± 0.05	0.70 ± 0.08
UNet	4.4M	893×893	29.54 ± 2.82	26.07 ± 2.80	0.83 ± 0.06	0.70 ± 0.09

Table 5.1: Field of view of CNN architectures and performance. Mean PSNR and SSIM ($\pm$ standard deviation) of different CNN architectures on the (a) held-out simulated test set of TEM data described in Section 5.5.2 and (b) validation set of the DIV2K photographic image dataset [1]. For TEM images, increasing the field of view (FoV) of the UNet from 45×45 pixels to 221×221 produces a dramatic increase of around 6 dB in PSNR, even if the number of parameters remains similar. Increasing the number of parameters while keeping the field of view constant produces a modest gain in performance. In contrast, changing the field of view has almost no effect on denoising photographic images. The networks used for photographic images were trained on 512×512 patches extracted from training images of DIV2K [1] corrupted with additive Gaussian noise with standard deviation $\sigma \in [0, 100]$.

images increasing the field of view produces a dramatic improvement in performance (6 dB and 10 dB, when the field of view is 221×221 and 893×893 respectively). Increasing the number of parameters, while keeping the field of view constant, has a very modest effect, which suggests that the increase in field of view is the primary reason for the improvement.

In order to gain some insight into the denoising mechanisms learned by our models, we apply the gradient-based analysis proposed by Ref. [96]. We visualize the linear term in the first-order Taylor decomposition of the denoising map with respect to its input for specific pixels. In more detail, we compute the gradient of a pixel in the denoised image $f(y)_i$ with respect to the input noisy image y. This vector (or image) $\nabla_y(f(y)_i)$, makes it possible to visualize the influence of different regions of the noisy image on the denoised pixel $f(y)_i$. This approach is similar to visualization methods proposed in the context of image classification (e.g. [97, 121]). Our analysis reveals that the network learns to simultaneously exploit local structure as well as non-local periodicities in the data (see Figure 5.4). This demonstrates the remarkable flexibility of data-driven denoising based on deep learning.

5.3.3 Likelihood maps

In most applied domains, the goal of denoising is to uncover image structure of scientific interest. In our case study, this corresponds to the location and intensity of projected columns of atoms in a catalytic nanoparticle that is surrounded by a vacuum. Quantifying to what extent such structure is consistent with the observed measurements is therefore of great interest. We propose to achieve this by computing the likelihood of the data *with respect to meaningful features identified in the denoised image*. The general procedure, and its implementation in the case of our case study, are as follows:

1. Identify a region of interest $\mathcal{R}$. In our case study, this would correspond to an atomic column, located for example via blob detection [79], or to the vacuum.

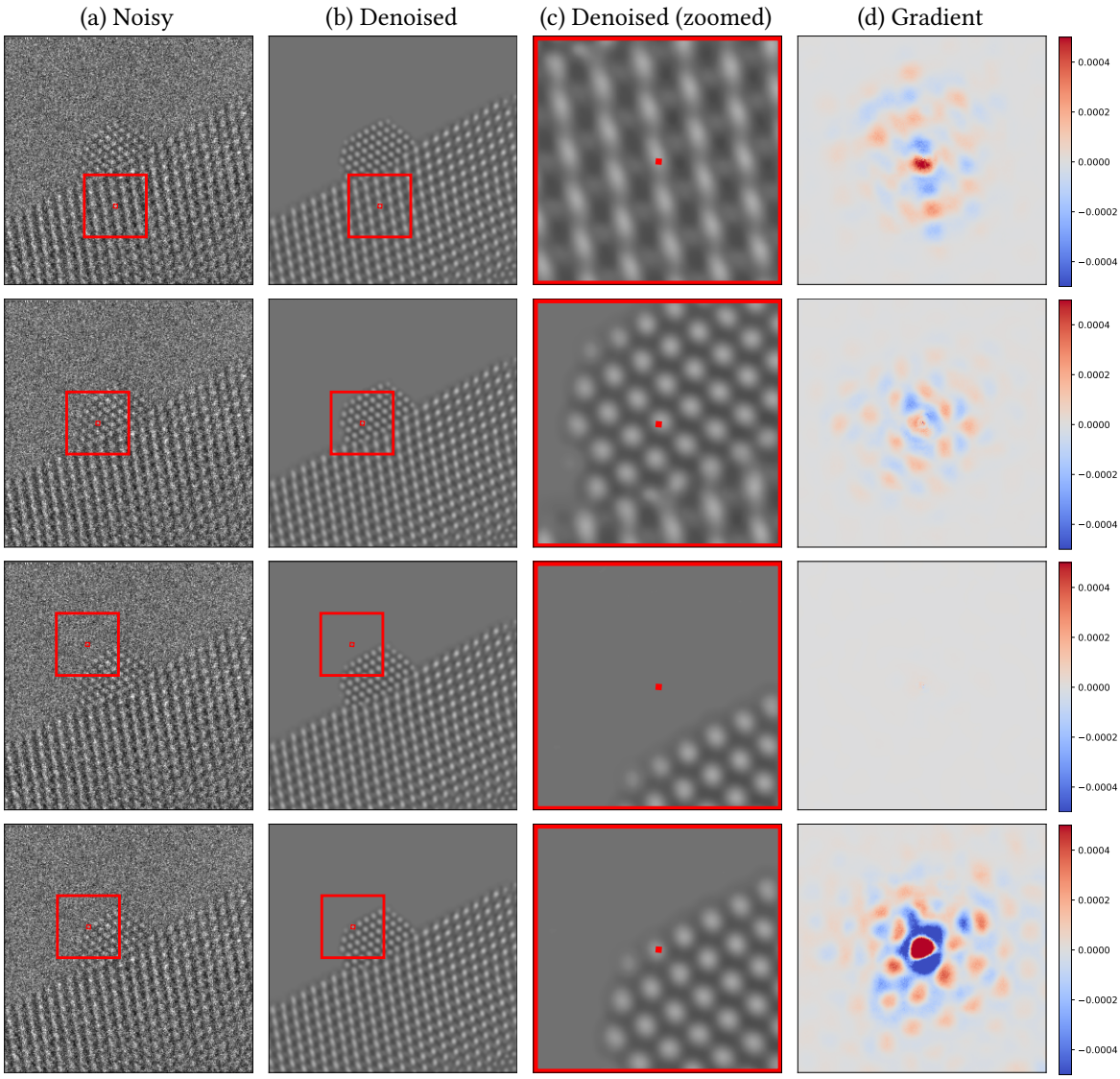


Figure 5.4: Gradient analysis of the learned denoising function on real data. (a) To compute the red pixel in the denoised image (b), the proposed CNN can use noisy pixels in a neighbourhood of up-to 893×893, of which a 300×300 area (red box) (c) is highlighted. The gradient of the denoised pixel with respect to its input indicates what regions in the noisy image have a greater influence on the estimate (according to a first-order Taylor approximation to the denoising map). The gradient (d) weights nearby pixels more heavily, but also has significant magnitude at pixels located on different atoms. This suggests that the CNN combines local and non-local information to estimate the pixel. The colorbar is shared across all the images.

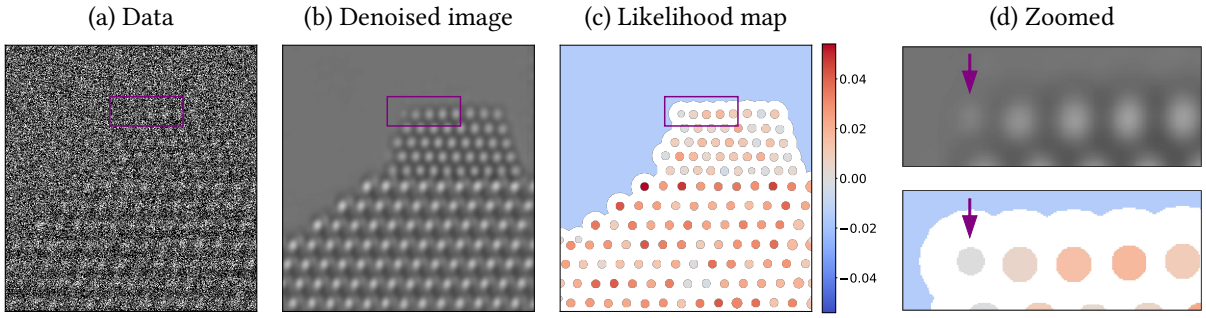


Figure 5.5: Likelihood map. When the simulated noisy image in (a) is denoised using the proposed framework (b), a spurious atom appears at the left edge of the nanoparticle (see zoomed image (d)). The value of the likelihood map (c) at that location is very low, indicating that the presence of an atom is less consistent with the observed data than its absence.

2. Fit a low-dimensional model to the denoised image within the region of interest. The low-dimensional model provides an estimate of the image value x_i at each pixel location $i \in \mathcal{R}$. In our case, we assume that the intensity of each atomic column and the vacuum are constant, so the estimate is obtained by averaging over all denoised pixels in $\mathcal{R}$.

3. Compute the likelihood of the noisy data in $\mathcal{R}$ with respect to the estimated pixel values. In our case, the noise is approximately independent and individually distributed (iid) Poisson (see Section 5.4.3), so the likelihood is given by

$$\mathcal{L}(\mathcal{R}) := \prod_{i \in \mathcal{R}} p_{x_i}(y_i), \tag{5.2}$$

where y_i denotes the noisy value in the ith pixel, and p_{x_i} is a Poisson probability mass function (pmf) with rate parameter x_i. Note that in the low-dimensional model, which assumes constant intensities, x_i is constant for all pixels in $\mathcal{R}$.

This technique makes it possible to consider different hypotheses about the underlying image structure and compare their agreement with the observed data. In our case study, we evaluate the hypotheses that a detected atomic column is (1) truly there, or (2) an artefact introduced by the denoising procedure. The likelihood under hypothesis (1) is computed as above. The likelihood

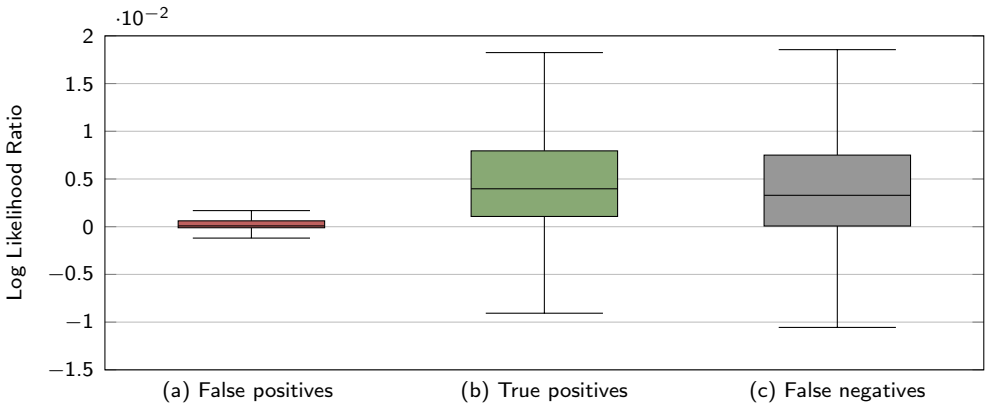


Figure 5.6: Distribution of likelihood ratio. The figure shows the distribution of log-likelihood ratio of over $25,000$ regions of interest computed from the surface of 1550 denoised images using the dataset described in Section 5.5.3.2. The empirical distribution is visualized as a box plot indicating the median, 25^{th} quartile, 75^{th} quartile, minimum and maximum value of the distribution. The regions containing spurious atoms (false positives, (a)) have a much lower log-likelihood ratio than the regions containing accurately recovered atoms (true positives, (b)). Regions where existing atoms were not detected (false negatives, (c)) have a higher log-likelihood ratio, comparable to that of the regions with accurately recovered atoms. The occurrence of missing and spurious atoms in denoised images is relatively low: out of the $25,732$ regions of interest, only $2,457$ and $2,368$ were false positives and false negatives respectively.

under hypothesis (2) is computed by setting the estimate x_i equal to the average intensity of the noisy pixels identified as belonging to the vacuum region. To visualize the consistency of the two hypotheses with the measured data, we plot the difference in their log likelihood for each region of interest. This is equivalent to performing a log-likelihood ratio test. We call this visualization a *likelihood map*.

Figures 5.1 and 5.5 show likelihood maps for the real data and for a simulated example. In the simulated example (Figure 5.5), a spurious atom is detected at the left end of the zoomed region. However, the likelihood map at that location is very low, which indicates that the presence of an atom is not consistent with the observed data at that location. Figure 5.6 shows the distribution of log-likelihood ratio of over $25,000$ regions of interest extracted from the surface of over $1,550$ denoised images obtained from the dataset described in Section 5.5.3.2. As shown in Figure 5.6, the log-likelihood ratio values of spurious atoms (false postives, (a)) are much lower than those of correctly-identified atoms (true positives, (c)). When the network fails to detects atoms (false

negatives, (b)), we observe that the log-likelihood ratio in such regions tends to be high. It is worth noting that the occurrence of spurious and missing atoms in the denoised images is relatively low: out of the $25,732$ regions identified, only $2,457$ and $2,368$ regions correspond to spurious and missing atoms respectively.

Visualizing the likelihood is useful to quantify the agreement between the output of deep-learning models and the observed data, but it is important to note that the approach suffers from sampling bias. We focus on regions of the input that have been selected because they resemble structures of scientific interest. The data in those regions are therefore more likely to be in agreement with the presence of such structures, just by the sheer fact that they have been selected. This is a manifestation of the notorious multiple-comparisons problem [10, 11]. Overcoming this issue is an important challenge for future research.

5.4 Dataset

The TEM image data used in this work correspond to images from a widely utilized catalytic system, which consist of platinum (Pt) nanoparticles supported on a larger cerium (IV) oxide (CeO_2) nanoparticle. This bi-functional catalytic system is ubiquitously used in clean energy conversion and environmental remediation applications, in addition to a broad range of other chemical reactions [98, 102, 146]. From a general point of view, this system can be considered as a model for supported nanoparticle catalysts, since a large number of heterogeneous catalysts are based on metallic nanoparticles supported over different oxides. Thus, results and conclusions extracted from the current work are relevant to a great number of similar samples in the field of catalysis (e.g., oxide crystals supporting metal nanoparticles).

5.4.1 Real Data

The real data used to test the proposed SBD framework consist of a series of images of the Pt/CeO_2 catalyst. The images were acquired in a N_2 gas atmosphere using an aberration-corrected FEI Titan transmission electron microscope (TEM), operated at 300 kV and coupled with a Gatan K2 IS direct electron detector. The detector was operated in electron counting mode with a time resolution of 0.025 sec/frame and an incident electron dose rate of 5,000 $e^-/Å^2/s$. The electromagnetic lens system of the microscope was tuned to achieve a highly coherent parallel beam configuration with minimal low-order aberrations (e.g., astigmatism, coma), and a third-order spherical aberration coefficient of approximately -13 μm.

5.4.2 Simulation Dataset

The simulated TEM image dataset was generated using the multi-slice TEM image simulation method, as implemented in the Dr. Probe software package [7] (see Section D.1.1 for more details on the simulation process). Images were simulated with 1024 x 1024 pixels and then binned to match the approximate pixel size of the experimentally acquired image series. To equate the intensity range of the simulated images with those acquired experimentally, the intensities of the simulated images were scaled by a factor which equalized the vacuum intensity in a single simulation to the average intensity measured over a large area of the vacuum in a single 0.025 second experimental frame (i.e., 0.45 counts per pixel in the vacuum region).

In the type of phase-contrast TEM imaging performed in this work, multiple electron-optical and specimen parameters can give rise to complex, non-linear modulations of the image contrast. These parameters include the objective lens defocus, the specimen thickness, the orientation of the specimen, and its crystallographic shape/structure. Various combinations of these parameters may cause the contrast of atomic columns in the image to appear as black, white, or an intermediate mixture of the two (see, e.g., Figure D.1). When designing the simulated dataset for the

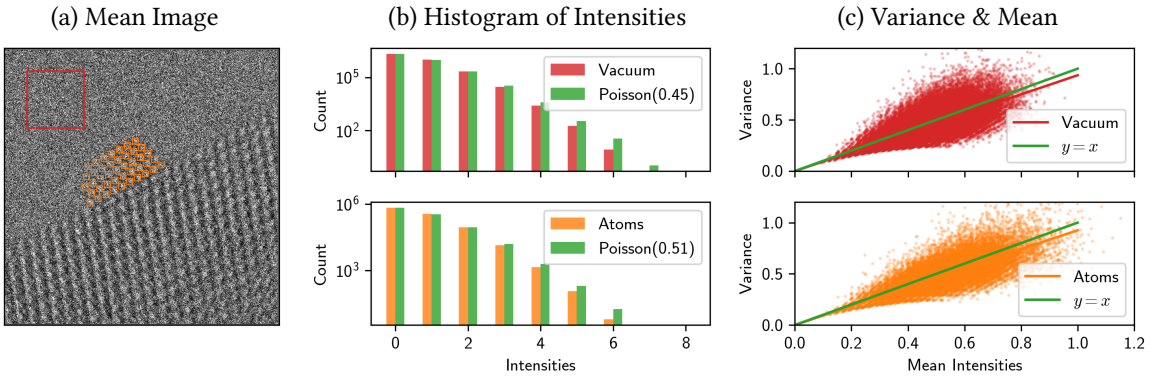


Figure 5.7: Analysis of the noise in the real data. The analysis shows that the noise is approximately iid Poisson. (a) Pixel-wise mean over 40 frames of the real data described in Section 5.4. (b) Histogram of noisy pixel intensities from highlighted regions in the image compared to a simulated Poisson distribution. (c) The plot of empirical mean and standard deviation of pixels approximately follows a line with unit slope, as expected from iid Poisson samples (the spread is due to averaging over only 40 frames).

SBD framework, it is necessary to include images simulated under widely varied conditions, in order to cover the breadth of possibilities which may arise during a typical experiment. A skilled microscopist attempts to acquire images under conditions in which the image contrast can be interpreted, which limits the overall size of the parameter space under consideration. However, various instances of defocus, tilt, thickness, and shape/structure inevitably arise. To generate our dataset we systematically varied these parameters to produce a large number of potential combinations (approximately 18,000), as described in Sections D.1.2 and D.1.3.

5.4.3 Noise model

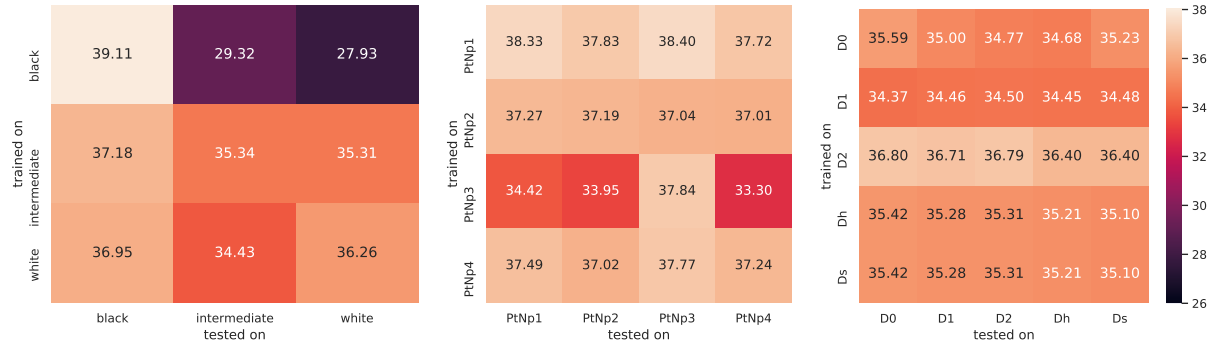
The TEM images were acquired on a direct electron detector operating in electron counting mode. In such conditions, the electron dose rate per pixel is sufficiently low enough that individual electron arrivals can be detected and registered. It is well known that the statistical fluctuations of such counting processes for discrete events are governed by shot noise, which can be modeled with a Poisson distribution [6]. We expect that other sources of noise, including fixed pattern noise, dark noise, and thermal noise are minimal after applying a gain correction and a

dark reference to the raw image, and by cooling the detector to -20 °C, respectively.

We empirically verified that the noise follows Poisson statistics through the analysis shown in Figure 5.7. Our real dataset, described in Section 5.4.1 consists of 40 noisy frames acquired sequentially across time in 0.025 sec intervals. Figure 5.7(a) shows the mean image over all 40 frames. The region of the image containing no material (red box) corresponds to the vacuum, where the electron beam intensity is uniformly illuminating the detector. Fluctuations of the intensity in this region therefore purely arise from the noise. We validate that the histogram of these pixels aggregated over the identified spatial region closely follows a Poisson distribution (Figure 5.7(b)). Pixels aggregated over spatial domains corresponding to the Pt atomic columns (orange boxes) show similar behavior. Further, if the distribution is indeed Poisson, the mean and variance of the noisy pixels should be approximately the same. The empirical mean and variance, computed by averaging over the 40 time frames at every spatial location, follows a linear trend, thus further confirming that the noise distribution has Poisson properties (Figure 5.7(c)). The spread in the scatter plot is due to the limited number of time frames over which we average.

5.5 Experiments and Results

In this section, we evaluate the performance of our proposed methodology and show that we outperform other methods by a large margin (more than 12 dB in PSNR on held-out simulated data). We also perform a thorough analysis of the generalization capability of our models, demonstrating that the CNNs are robust to variations in imaging parameters and in underlying signal structure. Furthermore, we demonstrate that standard performance metrics for photographs, such as peak signal-to-noise ratio (PSNR) and SSIM [138], may fail to produce a scientifically-meaningful evaluation of the denoising results, and we propose a few alternative metrics to remedy this. Finally, we show that our approach achieves effective denoising of real experimental data.



(a) Contrasts (b) Structures (c) Defects

Figure 5.8: Generalization across different imaging parameters and signal structures. In order to study the generalization ability of the proposed method we divided the simulated dataset described in Section 5.4 into subsets, based on the atomic column contrast, the structure/size of the supported Pt nanoparticle, and the defects of the Pt surface structure. The tables show the test PSNR for networks trained and tested on different combinations of the subsets. (a) Networks trained on white and intermediate contrast generalize well to all other contrasts. The network trained on black contrast does not generalize as well. (b) Networks trained on a type of nanoparticle structure generalize well to all other types. (c) Networks trained on one type of defect or no defects generalize well to different types. Figures D.1, D.5, D.4 and D.2 show the effect of these parameters on the images.

We use CNNs with the proposed UNet architecture with 128 base channels and 6 scales in all of our experiments (see Sections 5.3.2 and **??** for more details). The networks were trained on 400×400 patches extracted from the training images and augmented with horizontal flipping, vertical flipping, random rotations between $-45°$ and $+45°$, and random resizing by a factor of 0.75-0.82. The models were trained using the Adam optimizer [63], with a default starting learning rate of 10^{-3}, which was reduced by a factor of 2 every time the validation PSNR plateaued. Training was terminated via early stopping based on validation PSNR. The details of training, validation and test data for each experiment are provided in the corresponding section. Since the models are trained on 400×400 patches, when applying them to larger images we divide the images into overlapping 400×400 patches, denoise them, and then combine them via averaging.

5.5.1 Generalization to unseen structures and acquisition conditions

In order to study the generalization ability of the proposed approach across different imaging parameters and signal structures we divided the simulated dataset described in Section 5.4 into different subsets. These subsets were classified based on (1) the character of the atomic column contrast, (2) the structure/size of the supported Pt nanoparticle, and (3) the defects of the Pt surface structure. The contrast was classified into three divisions, black, intermediate, or white contrast, by a domain expert (see Figure D.1 in the supplementary material). The nanoparticle structure was classified into four categories, "PtNp1" through "PtNp4". PtNp1 and PtNp2 correspond to supported Pt nanoparticles of size 2 nm, which differ in the presence or absence of an atomic column located at the interface between the Pt and the CeO_2 support. PtNp3 corresponds to a Pt nanoparticle 1 nm in size. PtNp4 corresponds to a Pt nanoparticle 3 nm in size. Finally, the defects were divided into five categories: "D0", "D1", "D2", "Dh", and "Ds" in accordance with the atomic-scale structural models presented in D.1.1 and in particular in Figure D.5. D0 is the initial structure, D1/D2 a structure in which 1/2 atomic columns have been removed respectively, Dh a structure in which a column has been reduced to half its original occupancy, and Ds a structure in which a column has been reduced to a single atom. The generalization ability of the proposed CNN was evaluated by systematically training on each of the subsets and testing on the rest. The number of images in each subset was fixed to be equal in order to ensure a fair comparison.

The performance of SBD is robust to variations in imaging parameters and in the underlying signal structure, as shown in Figure 5.8. We only observe a significant decrease in performance when the network is trained on black-contrast images and tested on other contrasts (interestingly the network generalizes well from white and intermediate contrasts to black contrasts).

Methods	PSNR	SSIM
Raw	3.56 ± 0.03	0.00 ± 0.00
Low Pass Filter [101]	21.59 ± 0.07	0.44 ± 0.03
Adaptive Wiener Filter [78]	22.42 ± 1.08	0.63 ± 0.02
VST + NLM [13]	26.55 ± 0.16	0.73 ± 0.01
VST + BM3D [85]	25.27 ± 0.15	0.80 ± 0.01
PURE-LET [82]	28.36 ± 0.88	0.93 ± 0.01
SBD + DnCNN [150]	30.47 ± 0.64	0.93 ± 0.01
SBD + Small UNet [151]	30.87 ± 0.56	0.93 ± 0.01
SBD + Proposed Architecture	**42.87** ± 1.45	**0.99** ± 0.01

Table 5.2: Results on simulated test data. Mean PSNR and SSIM (± standard deviation) of different denoising methods on the held-out simulated test set described in Section 5.5.2. SBD approaches achieve the best results. SBD combined with the proposed architecture outperforms all other techniques by about 12 dB. The performance of SBD applied to additional architectures is reported in Table 5.1.

5.5.2 Comparison of SBD with other methods

The imaging parameters of the real data, described in Section 5.4, are well described by the white contrast category defined in Section D.1.2. We therefore used the subset of simulated dataset corresponding to this contrast (5583 images) to compare our proposed methodology to other models. 90% of the data were used for training. The remaining 559 images were evenly split into validation and test sets. We compare our proposed UNet architecture (see Section D.2) with two state-of-the-art architectures for photographic-image denoising [150, 151] (see Sections A.1.1 and A.1.3), and with several classical denoising methods:

- **Low-pass Filter (LPF)**: We apply a standard low-pass filter [101] designed by a domain expert. The cut-off frequency was determined based on inspection of the data in the Fourier domain.
- **Adaptive Wiener Filter**: We apply an adaptive low-pass Wiener filter [78] to account for variations in local image statistics. The mean and variance around each pixel are estimated from a local neighborhood. We selected a neighborhood with radius 13 pixels based on expert evaluation of the denoised images in the validation set.

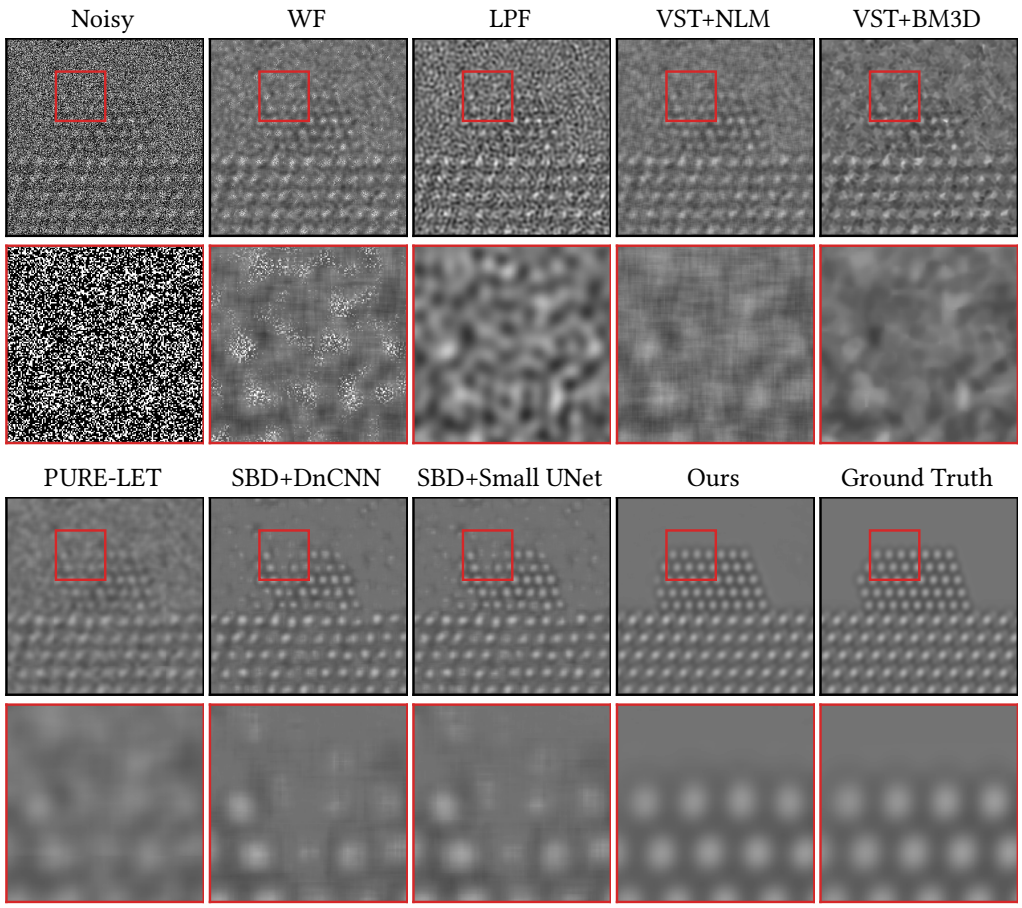


Figure 5.9: Denoising results for simulated data. Comparison of SBD and the baseline methods described in Sections 5.5.2 and **??**. The second row zooms in on the region in red box. Our proposed approach produces images of much higher quality than the other approaches, and is able to accurately recover the atomic structure of the nanoparticle. For example, the vacuum region in images denoised by several of the baselines contain visible artefacts, including missing atoms. See Figure D.6 in the supplementary material for an additional example.

- **VST + BM3D and VST + Non-Local Means (NLM)**: BM3D [85] and NLM [13] are popular denoising techniques for natural images with additive Gaussian noise, which can be adapted to Poisson noise by applying a nonlinear variance-stabilizing transformation (VST) [152]. More specifically, we use the Anscombe transformation proposed in Ref. [85].

- **PURE-LET**: PURE-LET [82] is a transform-domain thresholding algorithm adapted to mixed Poisson–Gaussian noise. The method requires the input image to have dimensions of the

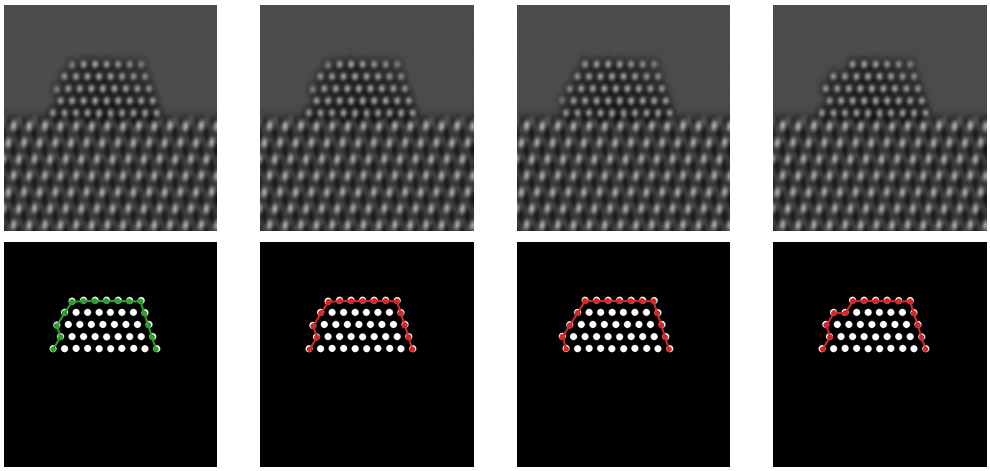
form $(2^k, 2^k)$. To apply this method on our TEM images, we extracted 128×128 overlapping patches, denoised them and combined them via averaging.

For all methods, hyperparameters were chosen based on the validation data. Performance was measured in terms of SSIM [138] and peak signal-to-noise ratio (PSNR).

The results demonstrate that SBD is an effective denoising methodology for TEM data. Our proposed CNN outperforms all other methods by a margin of 12 dB in PSNR on the simulated test data, as shown in Table 5.2, and Figures 5.9 and D.6. SBD recovers the overall shape of the nanoparticle, the interface between the nanoparticle and the support, and the different periodic patterns of the CeO_2 support and Pt nanoparticle. Contrast features, such as subtle patterns of bright, intermediate and dark features associated with the atomic structure of the CeO_2 crystal, are well reproduced in the images denoised via SBD, but are mostly absent from the results of the baseline approaches.

5.5.3 Beyond PSNR: Towards scientifically-meaningful evaluation metrics

Domain scientists denoise images in order to extract scientifically relevant information. In our case, the atoms on the surface of nanoparticles are of particular interest, because the atomic configuration at the surface regulates the nanoparticle's ability to catalyze chemical reactions. It is therefore of critical importance to understand how different denoising methods recover these atoms. We can verify visually that SBD achieves a largely successful recovery in held-out simulated data, whereas the baseline methods described in Section 5.5.2 do not (see Figure 5.9 for example). However, visual inspection is a limited and non-quantitative evaluation tool. Unfortunately, standard metrics like PSNR and SSIM are insensitive to changes in the atomic structure of the nanoparticle surface, because these changes have a small effect on the overall intensity of the images. We demonstrate the lack of sensitivity through a synthetic example in Figure 5.10: when



	(a) Reference	(b) All atoms preserved	(c) One extra atom	(d) One missing atom
PSNR	-	41.89	41.90	41.85
SSIM	-	0.946	0.948	0.949
Precision	-	1.00	0.933	0.933
Recall	-	1.00	0.933	0.933
F1	-	1.00	0.933	0.933
Jaccard Index	-	1.00	0.875	0.875

Figure 5.10: Scientifically-meaningful metrics for atom detection. In order to compare metrics in terms of their sensitivity to changes in atomic structure we perturb and compare the reference image in (a) to several modified images. In (b) the atomic structure is the same. In (c) a spurious atom is added to the image in the top left. In (d) an atom is removed from the top left. The images in (b), (c), and (d) are further corrupted by adding iid Gaussian noise with a small deviation of around $2/255$. The PSNR and SSIM [137] of (b), (c) and (d) with respect to (a) are essentially constant, indicating that these metrics are not sensitive to changes in atomic configuration. In contrast, our proposed metrics reflect these changes more accurately: (b) is assigned a score of 1 in all metrics, whereas (c) and (d) are consistently assigned lower values.

we add or remove an atom in the surface the PSNR and SSIM remain roughly constant. Motivated by the need for scientifically-relevant performance evaluation, we propose several metrics explicitly designed to account for changes in surface atomic configuration in Section 5.5.3.1. We report an evaluation of SBD using these metrics on a challenging test dataset in Section 5.5.3.2.

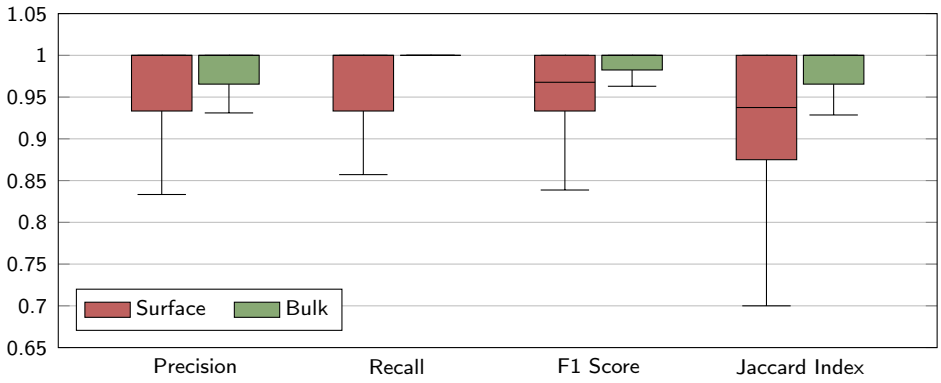


Figure 5.11: Performance of SBD in terms of our proposed metrics. We compute all our proposed metrics (see Section 5.5.3.1) on over $7,000$ denoised images corresponding to 25 unique noisy images sampled from the 308 clean images described in Section 5.5.3.2. The empirical distribution on the surface (red) and bulk (green) is visualized as box plots indicating the median, 25^{th} quartile, 75^{th} quartile, minimum and maximum value of the distribution. SBD has a near perfect performance in the bulk with all metric values hovering around 1. On the surface, SBD achieves a median score of 1 for precision and recall, and about 0.95 for F1 score and Jaccard index.

5.5.3.1 Evaluation metrics

To define metrics that evaluate detection of surface atoms, we assume that there is a predefined approach to perform detection based on the denoised images. In our case of interest, we apply a blob detection algorithm (Laplacian of Gaussian [79]) to locate the centers, and compute the α-shape of all the atom centers using Delaunay triangulation [109]. Let A and B be the set of surface atoms of interest in the denoised image and the ground-truth clean image respectively. We propose the following four metrics to measure the fidelity of the recovered structure:

- **Precision** is the fraction of atoms in the denoised image that are also present in the clean image.

$$P(A, B) = \frac{|A \cap B|}{|B|}. \tag{5.3}$$

- **Recall** is the fraction of atoms in the clean image that are correctly recovered in the denoised image.

$$R(A, B) = \frac{|A \cap B|}{|A|}. \tag{5.4}$$

- **F1 score** combines precision and recall by giving them equal importance.

$$F(A,B) = 2\frac{P(A,B)R(A,B)}{P(A,B)+R(A,B)}. \tag{5.5}$$

- **Jaccard index** is an alternative measure consisting of the ratio between the size of the intersection between the recovered atoms and the ground truth divided by the size of their union.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{5.6}$$

When performing intersection and union operations, we consider two atoms to be the same if the distance between their centers is less than a threshold of 10 pixels. For comparison, the physical distance between neighboring atoms is about 0.16nm, and 10 pixels correspond to a distance of 0.061nm. All our metrics take values between 0 and 1 (1 is best). Figure 5.10 shows a synthetic example comparing three images with different atomic configurations: all the images have similar PSNR and SSIM values, but the precision, recall, F1 and Jaccard index show substantial differences.

5.5.3.2 Evaluating atom detection accuracy

To evaluate the performance of the proposed approach to recover atoms at the surface, we designed a new dataset with 308 images, where the imaging parameters are set based on the real dataset described in Section 5.4.1. This new dataset is similar to the one used for the generalization experiments in Section 5.5.1, but here we add more diverse surface defects. We created a series of 44 Pt/CeO_2 structural models with atomic-level surface defects such as the removal of an atom from a column, removal of two atoms, removal of all but one atom and addition of an atom at a new site (see Figure D.7 for a visual overview). We hypothesize that these defects emulate dynamic atomic-level reconfigurations that could potentially be observed in real experiments.

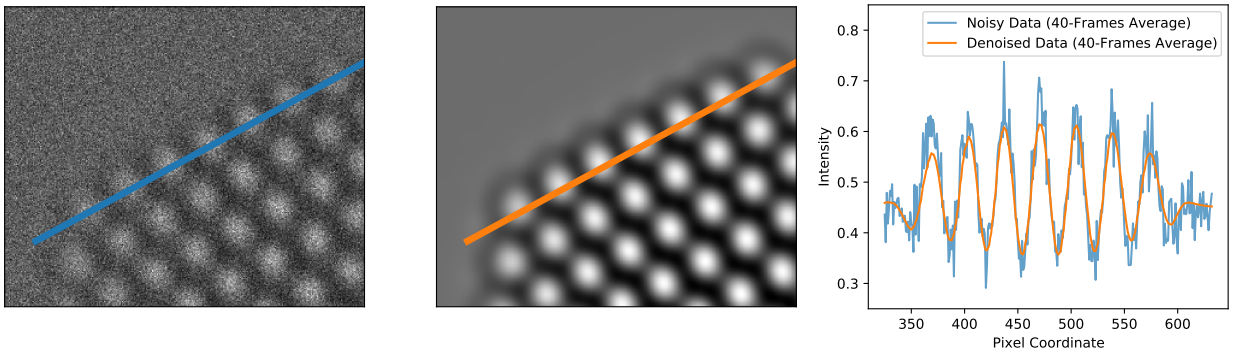


Figure 5.12: Validation on real data. The real data consist of 40 frames which are approximately stationary and aligned. Their temporal average (left) therefore provides a reasonable estimate for the true intensity profile. In the image on the right, we compare the average intensity profile on the surface atomic columns of the platinum nanoparticle for the denoised data (middle) and the temporal average (left). The profiles are very similar (except for some spurious fluctuations in the temporal average), which suggests that the proposed approach achieves effective denoising on the real data.

To match the image contrast of our real data, we simulated images under defocus values ranging from 6 nm to 10 nm, all with a tilt of 3° in x and -1° in y and a support thickness of 40 Å. SBD recovers all the atoms in the bulk almost perfectly, as reflected in the different metrics. On the surface, SBD achieves a median score of 1 for precision and recall, and more than 0.95 on F1 and Jaccard index (see Figure 5.11).

5.5.4 Performance on real data

In the experiments reported in Sections 5.5.2 and 5.5.3 we used a network trained on all simulated images from the white contrast category defined in Section 5.5.1. However, the real data described in Section 5.4 more closely corresponds to a subset of white contrast images satisfying the following conditions: structure limited to PtNP2, thickness between 40 Å - 60 Å and, defocus between 5 nm and 10 nm. We used 236 images from this subset for training, and another such 15 images for validation. We also trained two state-of-the-art architectures for photographic image denoising - DnCNN [150] and DURR [151] on these data.

Results on real experimental data obtained using SBD trained on this relevant subset of white

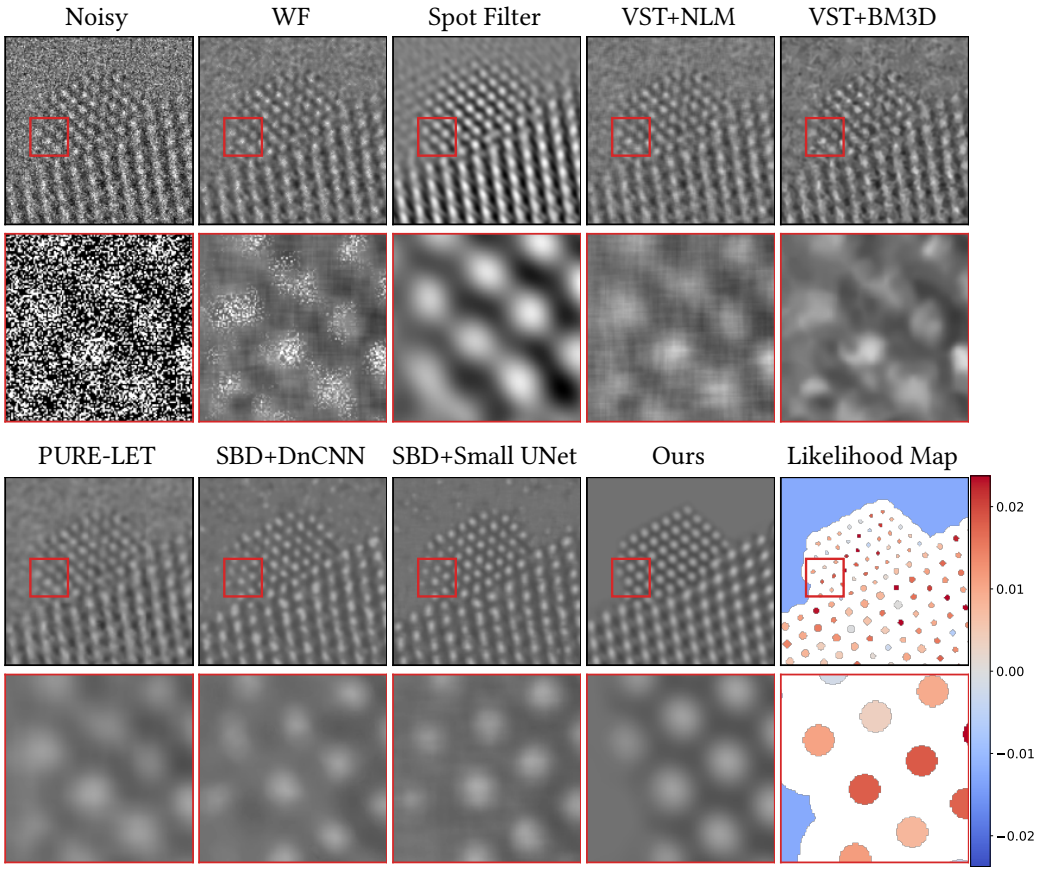


Figure 5.13: Denoising results for real data. Comparison SBD and the baseline methods described in Sections 5.5.2 and ?? when applied on the real data described in Section 5.4.1. The second row zooms in on the region in red box. In contrast to the other methods, SBD combined with the proposed architecture is able to precisely recover the structure of the nanoparticle and has very few artefacts, particularly in the vacuum region. The likelihood map quantifies the agreement between recovered structures in the denoised images, such as atomic columns and the vacuum, and the observed data (see Section 5.3.3 for more details). See Figure D.8 for an additional example

contrast are shown in Figures 5.1, 5.13, and D.8. SBD produces denoised images that are of much higher quality than those of the baseline methods described in Section 5.5.2, which contain obvious artefacts. Further, we validate the denoising results of SBD by comparing to an estimated reference image obtained by temporal averaging. Our real dataset consists of 40 frames that are approximately stationary and aligned. Therefore, their temporal average provides a good estimate for the ground-truth images. As shown in Figure 5.12, the denoised intensity values of the atomic column approximately match those of the estimated reference image.

In the rest of this section, we compare the performance of SBD and unsupervised denoising techniques on the real experimental data, and analyze the effect of the design of the training dataset on the denoised output produced by SBD.

5.5.4.1 Comparison to unsupervised deep denoising methods

Unsupervised denoising techniques can be used to train a denoising CNN using only noisy images (see Section 5.2 for a discussion on this methodology). We apply the following unsupervised methods to the real data described in Section 5.4.1:

- **Noise2Noise** [**lehtinen2018noise2noise**] is a strategy used to train CNN by using pairs of noisy images which correspond to the same clean image. We applied this method to our data by treating images captured in consecutive time steps as different noisy realizations of an underlying clean image. The results (shown in Figure 5.14(b)) contain visible artefacts and missing atoms.

- **Blind-spot net** [68] is a CNN which is constrained to predict the intensity of a pixel as a function of the noisy pixels in its neighbourhood, without using the pixel itself. This method is competitive with the current supervised state-of-the-art CNN on photographic images. However, when applied to our real dataset it produces denoised images with visible artefacts (see Figure 5.14(c)). A possible explanation is the limited amount of data (40 noisy images) we train on. To validate this hypothesis, we trained a blind-spot net on simulated training sets of different sizes. The performance on held-out data is indeed poor when the training set is small, but it improves to the level of supervised approaches as we use more training data (see Figure 5.15).

- **Blind-spot net with early stopping**. In Ref. [119] it is shown that early stopping based on noisy held-out data can boost the performance of blind-spot nets. Here we used 35 images for training the blind-spot net and the remaining 5 images as a held-out validation

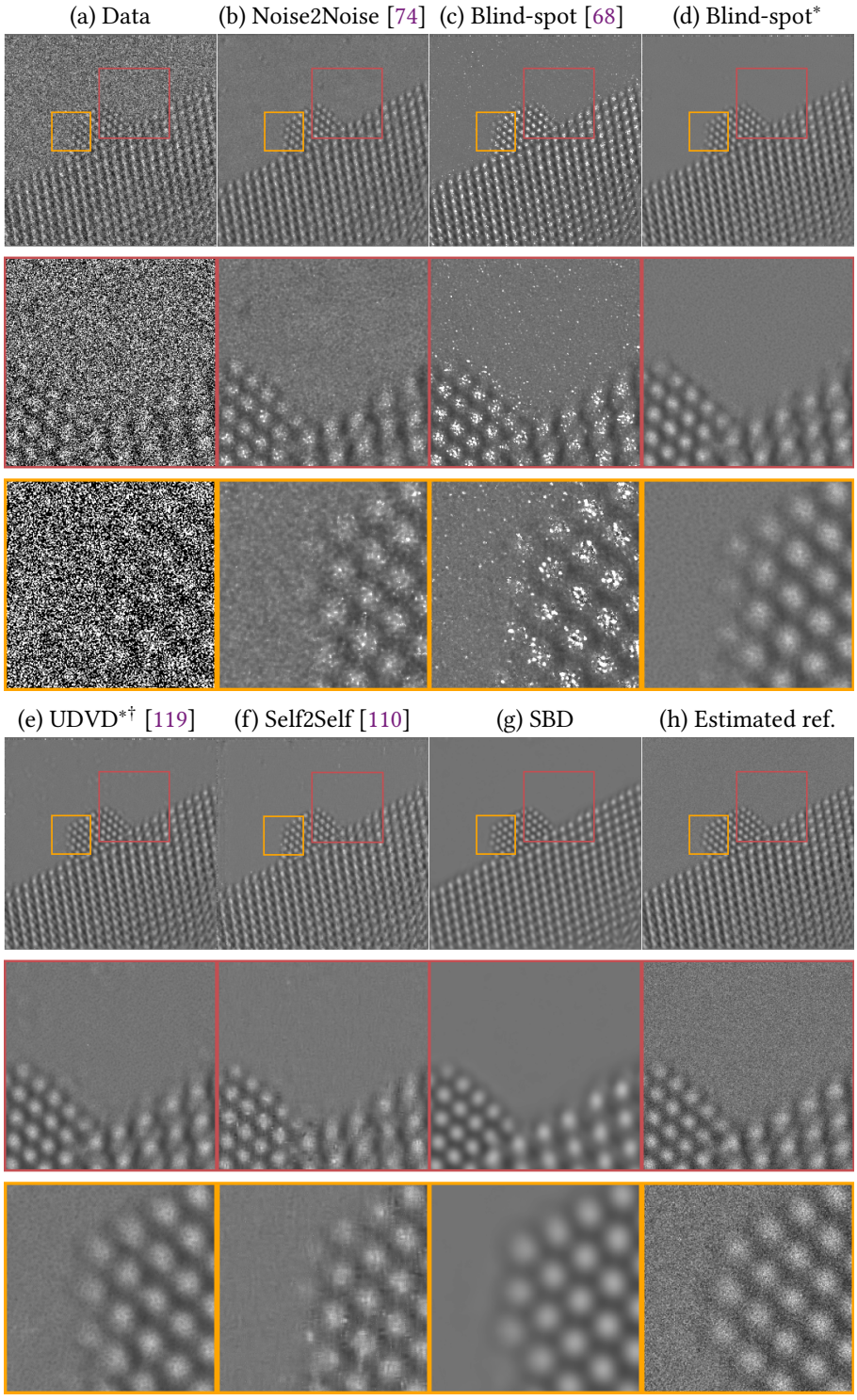


Figure 5.14: Comparison of unsupervised denoising methods with SBD on real data. The real data described in Section 5.4.1 denoised using SBD and unsupervised methods described in Section 5.5.4.1. The second and third rows zoom in on the region in red and green boxes respectively. Our proposed method denoises the real data more effectively than the unsupervised approaches. SBD is able to precisely recover the structure of the nanoparticle and has very few artefacts (compare visually to the estimated reference image obtained via time averaging; there are three missing atoms for most unsupervised methods in the third row). A $*$ indicates that the method used early stopping and † indicates that the method uses 5 noisy frames as input.

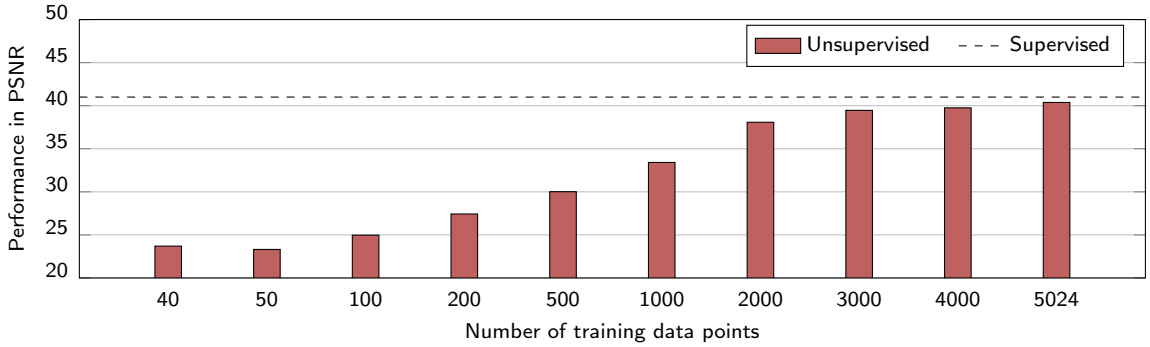


Figure 5.15: Training set size and unsupervised denoising. Performance of blind-spot net [68] on a held-out set of images, measured in PSNR, when trained on simulated training sets of different sizes. The held-out set and the training data comprise white contrast images. The performance is poor when the training set is small, but improves to the level of supervised approaches (denoted by dashed line) when trained using more training data. Our real dataset described in Section 5.4.1 contains only 40 images, so it is likely that the limited amount of data explains the poor performance of the blind-spot net in Figure 5.14.

set. We chose the model parameters that minimized the mean squared error between the noisy validation images and the corresponding denoised estimates. The results (shown in Figure 5.14(d)) are significantly better than those of the standard blind-spot network. However, there are still noticeable artefacts, which include missing atoms.

- **Unsupervised Deep Video Denoising (UDVD)** [119] is an unsupervised method for denoising video data based on the blind-spot approach. It estimates a denoised frame using 5 consecutive noisy frames around it. Our real data consists of 40 frames acquired sequentially. UDVD produces better results than blind-spot net, but still contains visible artefacts, including missing atoms (see Figure 5.14(e)). Note that, UDVD uses 5 noisy images as input, and thus has more context to perform denoising than the other methods (including SBD).

- **Self2Self** [110] is an unsupervised method specifically designed for denoising based on a single noisy image. This approach achieves near state-of-the-art performance in noisy photographic images corrupted with moderate amounts of noise [110]. However, when applied to our data, Self2Self produces images with clear artefacts; some of the atoms are

missing and the shape of atoms are distorted (see Figure 5.14(f)).

It is important to note that the backbone architectures of all these methods are UNets with large fields of view, like the one used for SBD. In our experiments, we trained the blind-spot nets and UDVD on 600×600 patches extracted from the real data. We used Adam optimizer [63] with a starting learning rate of 1×10^{-4} which was reduced in half for every 2000 epochs. We trained for a total of 5000 epochs. When performing early stopping, we picked the checkpoint with the best mean squared error on the validation set. Following Ref. [110], Self2Self was trained for $150,000$ steps with the Adam optimizer and a starting learning rate of 10^{-4}.

As shown in Figure 5.14, the unsupervised denoising methods produce higher-quality reconstructions than those of the baseline methods discussed in Section 5.5.2 (see Figure 5.13). However, they still suffer from visible artefacts, particularly on the surface of the nanoparticle, limiting their practical utility. UDVD is the method that achieves best performance, but it requires multiple noisy frames as input. In contrast, SBD can denoise the image effectively from a single noisy input frame (see Figure 5.14(g)), as long as the simulated training data correspond closely to the real noisy image. Using a single frame is important in some applications, such as our case of interest, where the ultimate goal is to identify dynamic changes in the atomic structure of the nanoparticle.

5.5.4.2 A word of caution: Effect of training data on SBD

Figures 5.13, D.8 and 5.14 show that SBD achieves impressive results on real data, but it is important to point out that this requires a careful design of the training dataset. Our real data broadly corresponds to images in the white contrast category, defined in Section 5.5.1. However, when a network trained on white contrast images (Section 5.5.2) is evaluated on the real data, it produces unnatural streak patterns in the bulk (see third row in Figure 5.16). When visually comparing this to the pattern in the bulk of the reference image computed by time averaging, it is evident that this is an artefact of denoising. This can be remedied by training the network

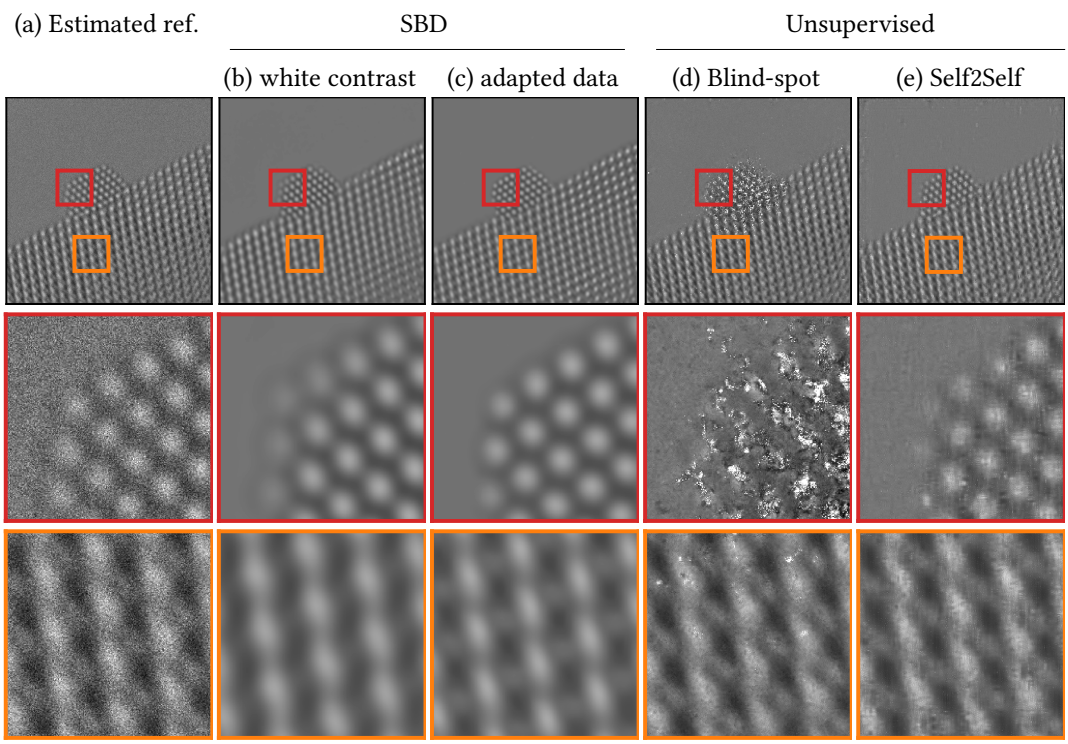


Figure 5.16: Effect of training data on SBD. The real data consists of 40 frames which are approximately stationary and hence their temporal mean (a) can be used as an estimate for the reference image. The second and third row zooms into the red and yellow region respectively. When the network is trained on white contrast (i.e, the training data does not align well with real data), the denoised image of the real data (b) shows an unnatural streak pattern in the bulk (compare row 3 in (b) and (a)). Interestingly, when the training data is more reflective of the real data, the patterns in the bulk are recovered better (row 3 in (c) and (a)). Further, note that unsupervised methods denoise the contrast patterns in the bulk well (row 3 in (d), (e) and (a)) but they suffer from significant artefacts on the nanoparticles (row 2 in (d), (e) and (a)).

on the more restricted subset of images described in Section 5.5.4 (see third row in Figure 5.16), whose imaging parameters are more suited to the real acquisition conditions. Since unsupervised denoising methods directly train on the real data, they do not suffer from this problem of mismatch between training and test data. The patterns recovered by unsupervised methods in the bulk are close to the estimated reference image (see Figure 5.16). However, as discussed in Section 5.5.4.1, they show significant artefacts on the surface of nanoparticle. As shown in Section 4.5.4, performing GainTuning on the network trained on white contrast can also remedy this problem.

5.6 Discussion and Conclusions

Our case study is a proof of concept that CNNs trained on simulated data can be remarkably effective when applied to real imaging data. It provides several insights and suggests future research directions that are relevant, beyond electron microscopy, to other domains where the images of interest can be simulated, such as medical imaging [61, 93], other types of microscopy [40, 134], or astronomy [104]. We show that the design of the training dataset is critical, so an important question is how to design simulated training datasets in a principled systematic way. Answering it will require a deeper understanding of the generalization ability of CNNs with respect to variations in the statistics of the input images. We also demonstrate that architectures tailored to photographic imaging can perform poorly when applied to other data. Designing CNNs for other domains requires an understanding of the image features that are exploited for denoising. Gradient visualization is shown to be useful here, but more advanced visualization techniques are needed. In addition, we demonstrate that standard metrics used to quantify performance in photographs may not be sensitive to scientifically relevant features, and propose several new metrics to address this problem. Although SBD outperforms other methods by a large margin, some artefacts such as phantom atoms still appear. Our proposed likelihood maps help to flag such events, but may still fail to do so in regions of unusually low SNR. Developing more sophisticated methods for uncertainty quantification is therefore a key research direction. It would also be of great interest to develop unsupervised or self-supervised denoising approaches that are effective with small amounts of data at low SNRs. Finally, to encourage further development of deep-learning methodologies for scientific imaging, we release a denoising benchmark dataset of TEM images, containing 18,000 examples.

6 | Conclusion

In this thesis, we explored the topic of deep learning based denoising. The contribution of the thesis is broadly in two aspects: (1) novel network architectures and algorithms advancing the current state-of-the-art in denoising, and (2) new tools for analyzing neural networks advancing our understanding of deep learning models for denoising. While this thesis concentrated on denoising, many of our contributions have relevance beyond denoising. We finish by summarizing some of our specific contributions, commenting on their applications outside denoising, and listing our future research directions and a few related open problems:

- **Bias-free CNNs**. In Chapter 2 we introduced bias-free CNNs which generalize to noise levels outside the training range. We showed that this generalization capability is facilitated by the local linearity of the network architecture (Section 2.4). Since our introduction of bias-free networks in denoising, it has been applied in photometric stereo [50], reflection removal [154], and tone mapping [70]. Our observations in Chapter 2 do not fully elucidate how our network achieves its remarkable generalization- only that bias prevents that generalization, and its removal allows it. Understanding how bias-free networks achieve this generalization is a direction for future research.

- **Unsupervised Deep Video Denoising (UDVD)**. In Chapter 3, we extended the idea of blind-spot denoising for static images [8, 66] to develop an unsupervised denoiser which can be trained only with noisy videos. UDVD achieved performance comparable to supervised state-of-the-art, even when trained on only a single short noisy video sequence. This

enabled the application of deep learning models to domains like microscopy where clean data is generally not available. However, evaluation of denoising models in the absence of ground-truth data is an unsolved problem. Current metrics used for evaluation (eg. PSNR and SSIM [138]) measures the closeness of the denoised image with ground truth data. Likelihood maps introduced in Section 5.3.3 takes a step in this direction by measuring the fidelity of the denoised image with noisy data under strict assumptions about the signal structure and noise likelihood. A more general solution towards unsupervised evaluation of denoising models remains an interesting open challenge.

- **GainTuning**. In Chapter 4 we introduced GainTuning, a framework to adapt any pre-trained denoiser to an out-of-distribution data point. GainTuning achieved state-of-the-art performance on test data points which systematically differ in signal or noise distribution from the training data. The core idea behind GainTuning is simple: optimize only a small subset of parameters (the "Gains") during inference to adapt the model to the test data. A mechanism similar to GainTuning achieved out-of-distribution generalization in classification [136], and we have preliminary results indicating that such a mechanism might be effective in audio compression. This suggests that GainTuning may have applications in other areas of signal processing and machine learning like compression and segmentation. Despite the success in denoising, and potential applications in other areas of machine learning, GainTuning in its current form requires multiple backward passes to compute the gain parameters. Making this routine more efficient, or completely replacing optimization during test time is a direction for future research.

- **Analysis of models**. In Chapter 2 we proposed a gradient-based analysis to visualize the equivalent action of a denoising network. We extended this analysis to video denoisers in Chapter 3. Our analysis on image and video denoisers revealed that these models average over noisy pixels in the relevant spatio-temporal neighbourhood to compute the

denoised pixel value. This indicates that CNN denoisers encode the underlying structure of the signals. We believe that this property makes denoising a good candidate as an auxiliary task for other problems that exploits underlying signal structure. This hypothesis is supported by recent results in segmentation [16] and compression [127]. Our observations and insights in Chapter 2 and 3 uncovers interesting aspects of the denoising map, but these interpretations are very local: small changes in the input image change the activation patterns of the network, resulting in a change in the corresponding linear mapping. Extending the analysis to reveal global characteristics of the neural-network functionality is a challenging direction for future research.

A | Bias-free denoising

A.1 Description of denoising architectures

In this section we describe the denoising architectures used for our computational experiments in more detail.

A.1.1 DnCNN

We implement BF-DnCNN based on the architecture of the Denoising CNN (DnCNN) [150]. DnCNN consists of 20 convolutional layers, each consisting of 3×3 filters and 64 channels, batch normalization [54], and a ReLU nonlinearity. It has a skip connection from the initial layer to the final layer, which has no nonlinear units. To construct a bias-free DnCNN (BF-DnCNN) we remove all sources of additive bias, including the mean parameter of the batch-normalization in every layer (note however that the scaling parameter is preserved).

A.1.2 Recurrent CNN

Inspired by [151], we consider a recurrent framework that produces a denoised image estimate of the form $\hat{x}_t = f(\hat{x}_{t-1}, y_{\text{noisy}})$, at time t where f is a neural network. We use a 5-layer fully convolutional network with 3×3 filters in all layers and 64 channels in each intermediate layer to implement f. We initialize the denoised estimate as the noisy image, i.e $\hat{x}_0 := y_{\text{noisy}}$. For the

version of the network with net bias, we add trainable additive constants to every filter in all but the last layer. During training, we run the recurrence for a maximum of T times, sampling T uniformly at random from $\{1, 2, 3, 4\}$ for each mini-batch. At test time we fix $T = 4$.

A.1.3 UNet

Our UNet model [116] has the following layers:

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

The structure is the same as in [151], but without recurrence. For the version with bias, we add trainable additive constants to all the layers other than *conv9*. This configuration of UNet assumes even width and height, so we remove one row or column from images in with odd height or width.

A.1.4 Simplified DenseNet

Our simplified version of the DenseNet architecture [52] has 4 blocks in total. Each block is a fully convolutional 5-layer CNN with 3×3 filters and 64 channels in the intermediate layers with ReLU nonlinearity. The first three blocks have an output layer with 64 channels while the last block has an output layer with only one channel. The output of the i^{th} block is concatenated with the input noisy image and then fed to the $(i+1)^{th}$ block, so the last three blocks have 65 input channels. In the version of the network with bias, we add trainable additive parameters to all the layers except for the last layer in the final block.

A.2 Datasets and training procedure

Our experiments are carried out on 180×180 natural images from the Berkeley Segmentation Dataset [87]. We use a training set of 400 images. The training set is augmented via downsampling, random flips, and random rotations of patches in these images [150]. A test set containing 68 images is used for evaluation. We train the DnCNN and it's bias free model on patches of size 50×50, which yields a total of 541,600 clean training patches. For the remaining architectures, we use patches of size 128×128 for a total of 22,400 training patches.

We train DnCNN and its bias-free counterpart using the Adam Optimizer [63] over 70 epochs with an initial learning rate of 10^{-3} and a decay factor of 0.5 at the 50^{th} and 60^{th} epochs, with no early stopping. We train the other models using the Adam optimizer with an initial learning rate of 10^{-3} and train for 50 epochs with a learning rate schedule which decreases by a factor of 0.25 if the validation PSNR decreases from one epoch to the next. We use early stopping and select the model with the best validation PSNR.

A.3 Additional results

In this section we report additional results of our computational experiments:

- Figure A.1 shows the first-order analysis of the residual of the different architectures described in Section A.1, except for DnCNN which is shown in Figure 2.1.
- Figures A.2 and A.3 visualize the linear and net bias terms in the first-order decomposition of an example image at different noise levels.
- Figure A.4 shows the PSNR results for the experiments described in Section 2.4.
- Figure A.5 shows the SSIM results for the experiments described in Section 2.4.
- Figures A.6, A.7 and A.8 show the equivalent filters at several pixels of two example images for different architectures (see Section 2.5).

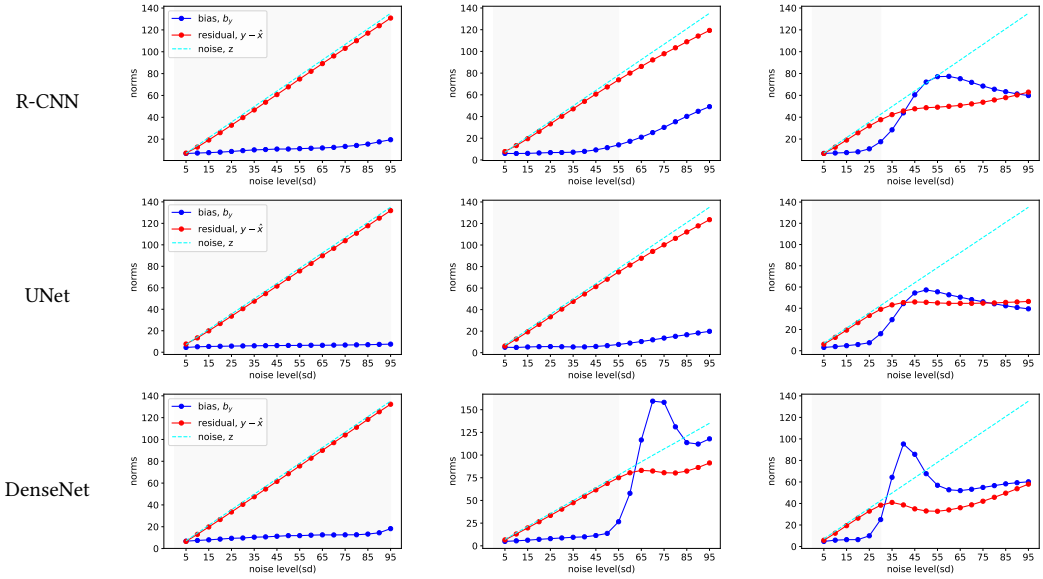


Figure A.1: First-order analysis of the residual of Recurrent-CNN (Section A.1.2), UNet (Section A.1.3) and DenseNet (Section A.1.4) as a function of noise level. The plots show the magnitudes of the residual and the net bias averaged over 68 images in Set68 test set of Berkeley Segmentation Dataset [87] for networks trained over different training ranges. The range of noises used for training is highlighted in gray. (left) When the network is trained over the full range of noise levels ($\sigma \in [0, 100]$) the net bias is small, growing slightly as the noise increases. (middle and right) When the network is trained over the a smaller range ($\sigma \in [0, 55]$ and $\sigma \in [0, 30]$), the net bias grows explosively for noise levels outside the training range. This coincides with the dramatic drop in performance due to overfitting, reflected in the difference between the residual and the true noise.

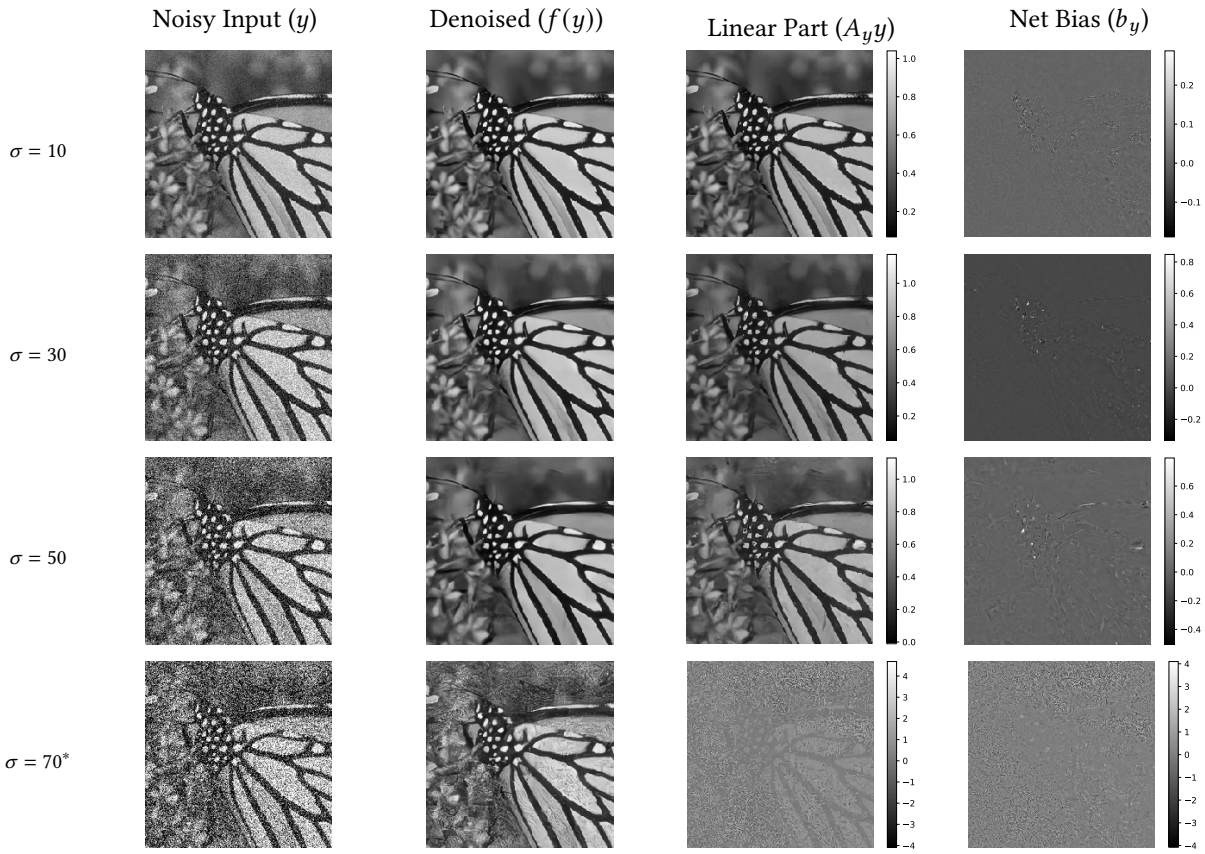


Figure A.2: Visualization of the decomposition of output of DnCNN trained for noise range $[0, 55]$ into linear part and net bias. The noise level $\sigma = 70$ (highlighted by $*$) is outside the training range. Over the training range, the net bias is small, and the linear part is responsible for most of the denoising effort. However, when the network is evaluated out of the training range, the contribution of the bias increases dramatically, which coincides with a significant drop in denoising performance.

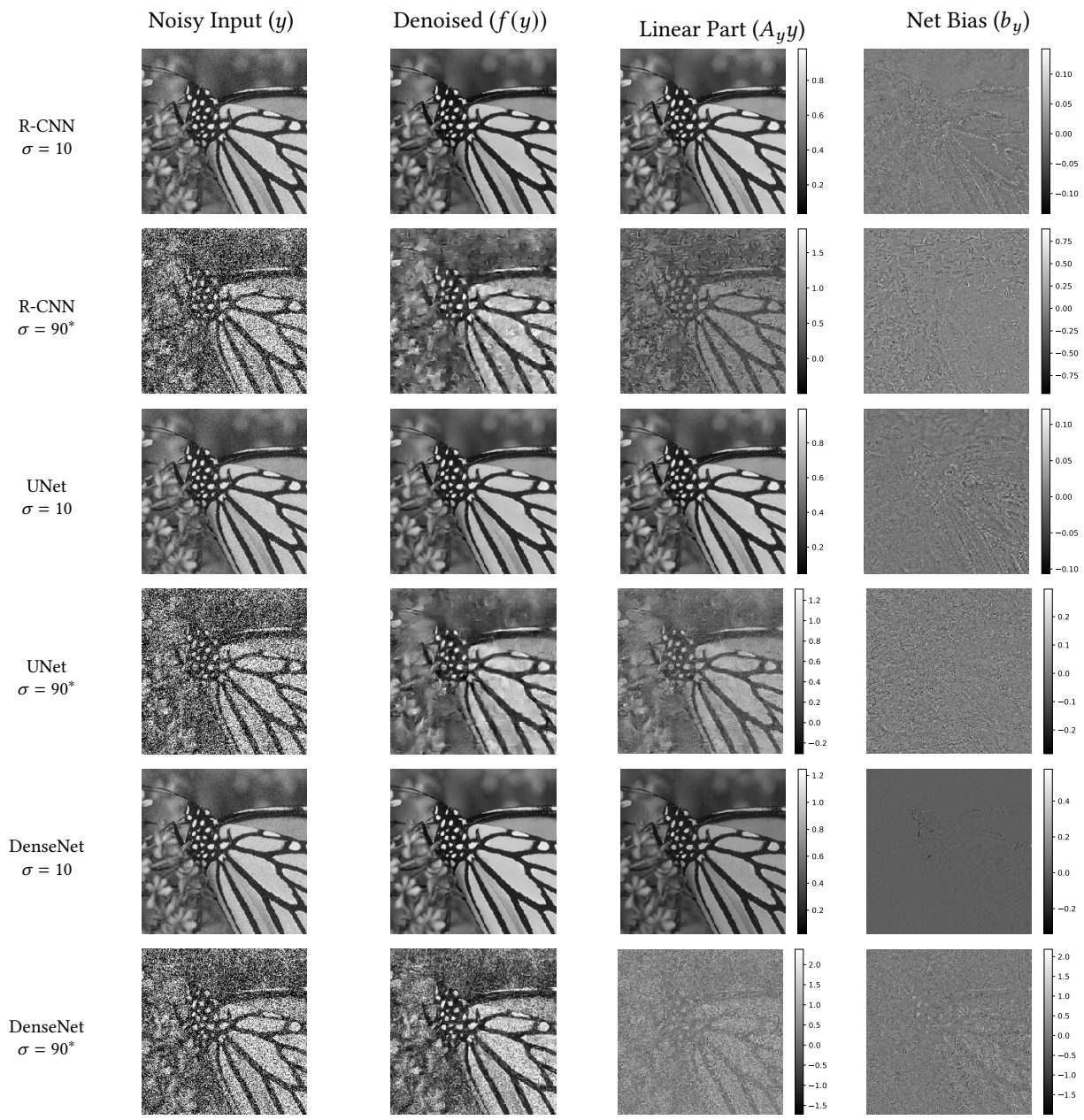
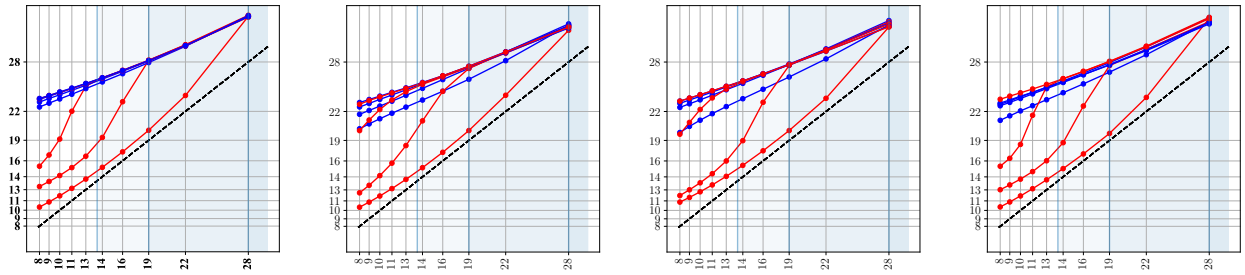
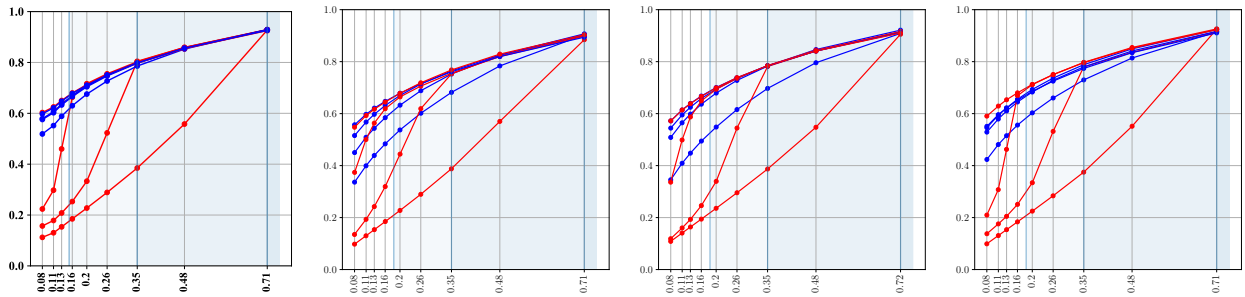


Figure A.3: Visualization of the decomposition of output of Recurrent-CNN (Section A.1.2, UNet (Section A.1.3) and DenseNet (Section A.1.4) trained for noise range $[0, 55]$ into linear part and net bias. The noise level $\sigma = 90$ (highlighted by $*$) is outside the training range. Over the training range, the net bias is small, and the linear part is responsible for most of the denoising effort. However, when the network is evaluated out of the training range, the contribution of the bias increases dramatically, which coincides with a significant drop in denoising performance.



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

Figure A.4: Comparisons of architectures with (red curves) and without (blue curves) a net bias for the experimental design described in Section 2.4. The performance is quantified by the PSNR of the denoised image as a function of the input PSNR of the noisy image. All the architectures with bias perform poorly out of their training range, whereas the bias-free versions all achieve excellent generalization across noise levels. **(a)** Deep Convolutional Neural Network, DnCNN [150]. **(b)** Recurrent architecture inspired by DURR [151]. **(c)** Multiscale architecture inspired by the UNet [116]. **(d)** Architecture with multiple skip connections inspired by the DenseNet [52].



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

Figure A.5: Comparisons of architectures with (red curves) and without (blue curves) a net bias for the experimental design described in Section 2.4. The performance is quantified by the SSIM of the denoised image as a function of the input SSIM of the noisy image. All the architectures with bias perform poorly out of their training range, whereas the bias-free versions all achieve excellent generalization across noise levels. **(a)** Deep Convolutional Neural Network, DnCNN [150]. **(b)** Recurrent architecture inspired by DURR [151]. **(c)** Multiscale architecture inspired by the UNet [116]. **(d)** Architecture with multiple skip connections inspired by the DenseNet [52].

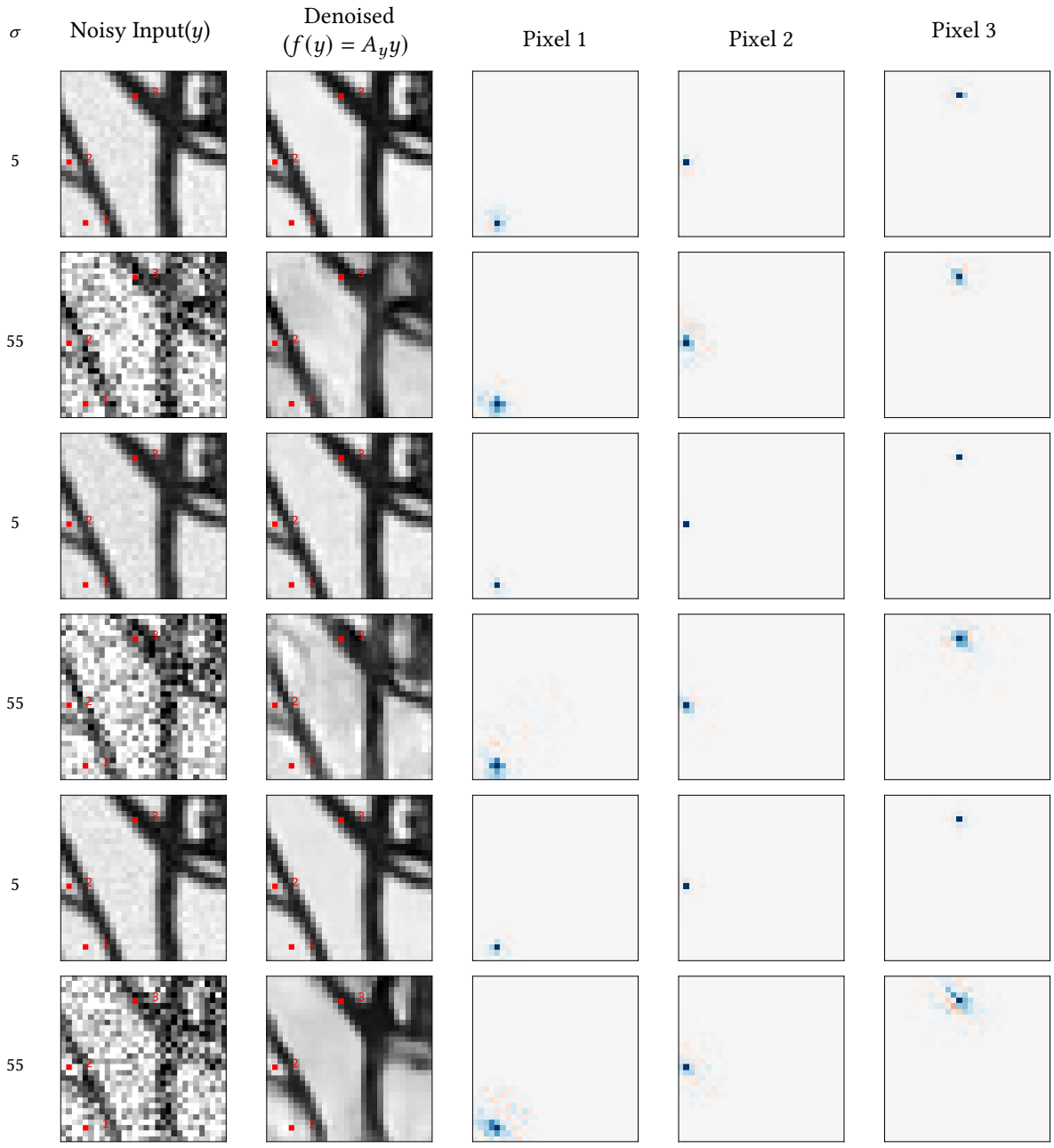


Figure A.6: Visualization of the linear weighting functions (rows of A_y) of Bias-Free Recurrent-CNN (top 2 rows) (Section A.1.2), Bias-Free UNet (next 2 rows) (Section A.1.3) and Bias-Free DenseNet (bottom 2 rows) (Section A.1.4) for three example pixels of a noisy input image (left). The next image is the denoised output. The three images on the right show the linear weighting functions corresponding to each of the indicated pixels (red squares). All weighting functions sum to one, and thus compute a local average (although some weights are negative, indicated in red). Their shapes vary substantially, and are adapted to the underlying image content. Each row corresponds to a noisy input with increasing σ and the filters adapt by averaging over a larger region.

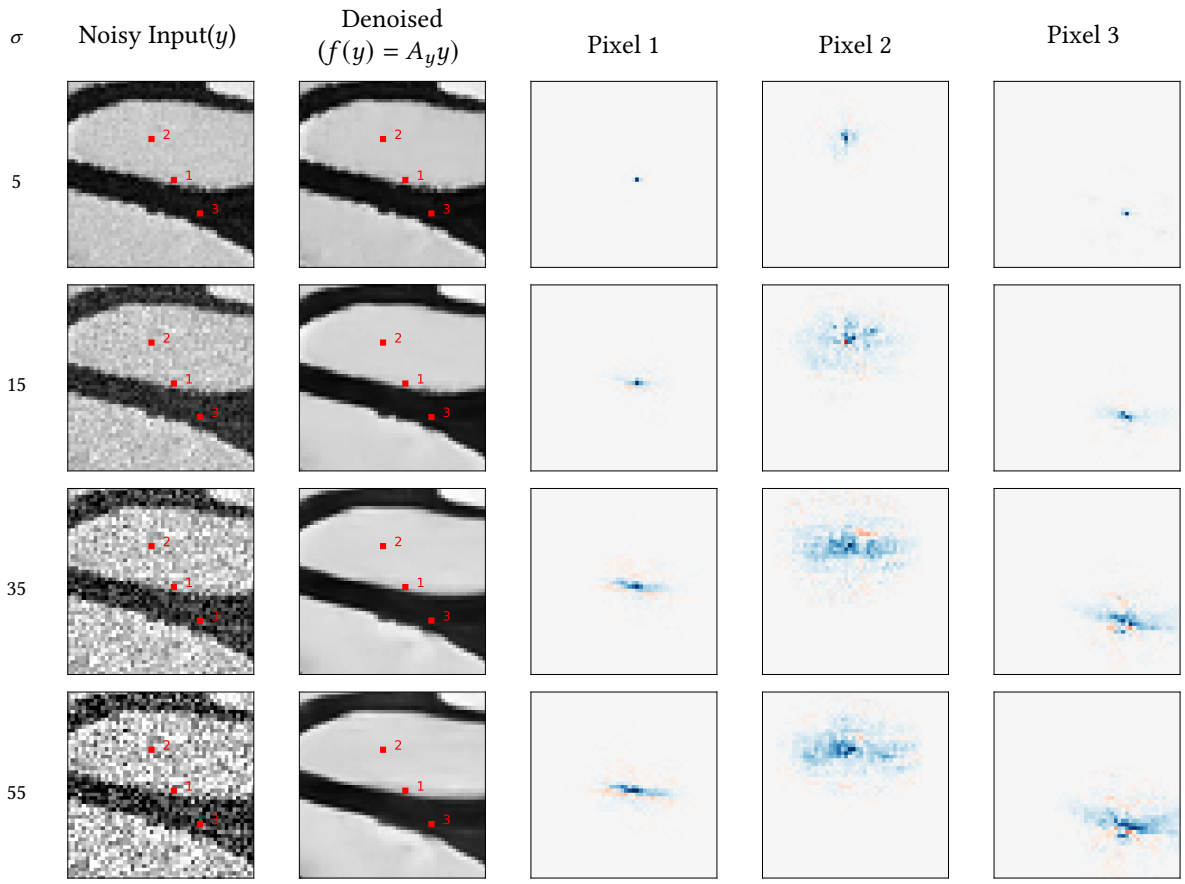


Figure A.7: Visualization of the linear weighting functions (rows of A_y) of a BF-DnCNN for three example pixels of a noisy input image (left). The next image is the denoised output. The three images on the right show the linear weighting functions corresponding to each of the indicated pixels (red squares). All weighting functions sum to one, and thus compute a local average (although some weights are negative, indicated in red). Their shapes vary substantially, and are adapted to the underlying image content. Each row corresponds to a noisy input with increasing σ and the filters adapt by averaging over a larger region.

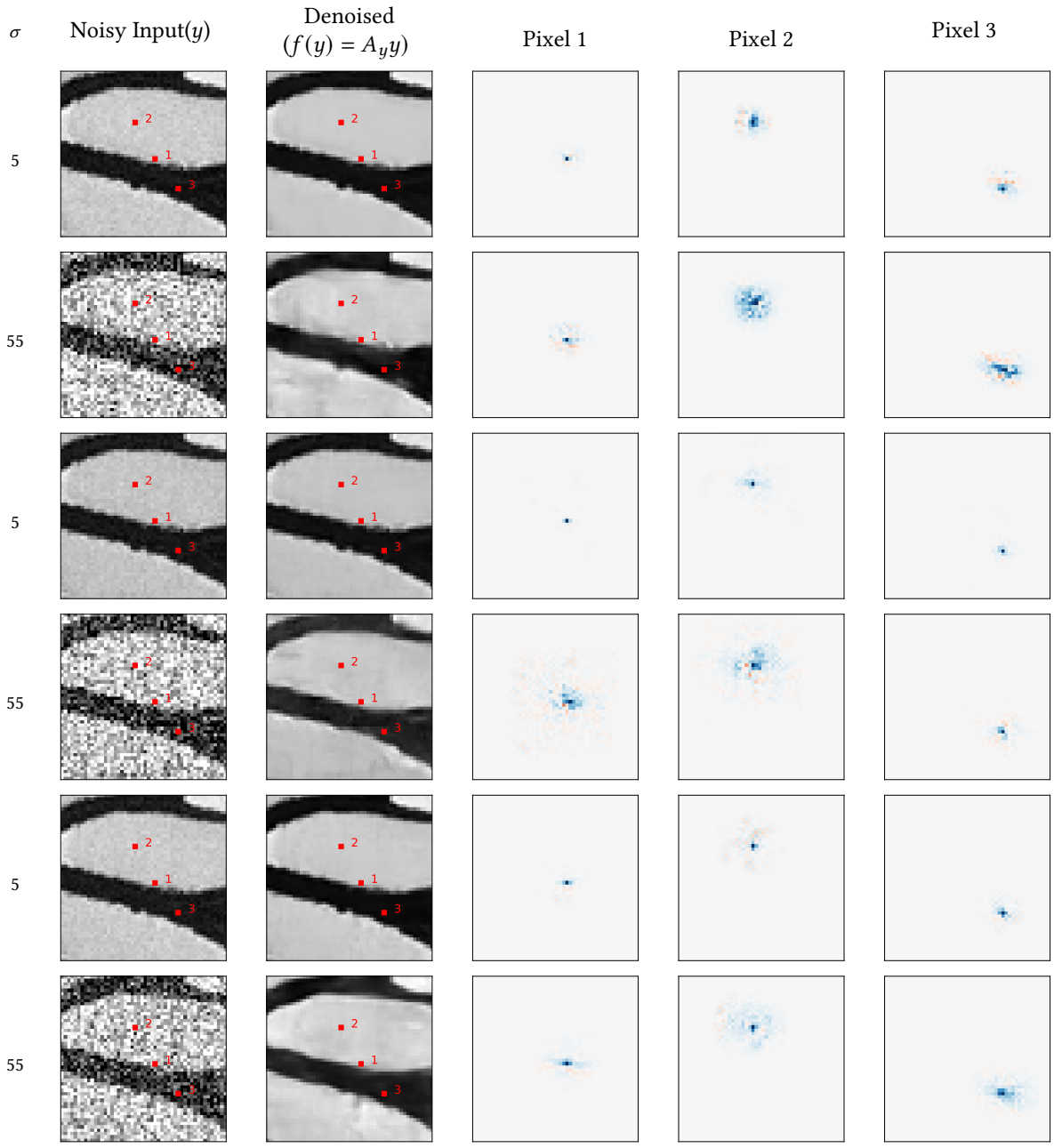


Figure A.8: Visualization of the linear weighting functions (rows of A_y) of Bias-Free Recurrent-CNN (top 2 rows) (Section A.1.2), Bias-Free UNet (next 2 rows) (Section A.1.3) and Bias-Free DenseNet (bottom 2 rows) (Section A.1.4) for three example pixels of a noisy input image (left). The next image is the denoised output. The three images on the right show the linear weighting functions corresponding to each of the indicated pixels (red squares). All weighting functions sum to one, and thus compute a local average (although some weights are negative, indicated in red). Their shapes vary substantially, and are adapted to the underlying image content. Each row corresponds to a noisy input with increasing σ and the filters adapt by averaging over a larger region.

B | Unsupervised denoising

B.1 Implementation Details of Unsupervised Deep Video Denoising

B.1.1 Restricting field of view

In UDVD, we rotate the input frames by multiples of 90° and process them through four separate branches (with shared parameters) containing asymmetric convolutional filters that are *vertically causal.* As a result, the branches produce outputs that only depend on the pixels above (0° rotation), to the left (90°), below (180°) or to the right (270°) of the output pixel. We use a UNet [116] style architecture for each branch of UDVD. The field of view of the UNet is constrained by restricting the field of view of the convolutional, downsampling and upsampling layers that are used to build the UNet.

Convolutional Layers: We restrict the receptive field of each convolutional layer to extend only upwards following the strategy proposed in [68]. Let the filter size be $h \times w$. We zero-pad the top region of the input tensor with $k = *h/2$ zero rows before convolution and remove the bottom k rows after convolution. This is equivalent to convolving with a filter, where all weights below the center row are zero, so that the field of view only extends upwards.

Downsampling and Upsampling Layers: Following [68] we restrict the receptive field of the downsampling layer by creating an offset of one pixel (zero-pad with a row of zeros on the top and remove a row of pixels from below) before performing max-pooling using a 2×2 kernel. This operation restricts the field of view of the downsampling and upsampling operation pair.
Note that we do not use BatchNorm [55] layers in UDVD as computing the spatial mean and variance would modify the field of view to include the center pixel.

B.1.2 Adding the Noisy Pixel Back

The denoised generated by the proposed architecture at each pixel is computed without using the noisy observation at that location. This avoids overfitting – i.e. learning the trivial identity map that minimizes the mean-squared error cost function – but ignores important information provided by the noisy pixel. In the case of Gaussian additive noise, we can use this information via a precision-weighted average between the network output and the noisy pixel value. Following [68], the weights in the average are derived by assuming a Gaussian distribution for the error in the blind-spot estimates of the (color) pixel values. The CNN architecture is trained to estimate the mean and covariance of this distribution at each pixel by maximizing the log likelihood of the noisy data. We explain this in detail in the following paragraphs.

UDVD estimates the value of a pixel based on the noisy pixels in its neighbourhood. We model the distribution of the three color channels of a pixel $x \in \mathcal{R}^3$ given the noisy neighbourhood Ω_y as $p(x|\Omega_y) = \mathcal{N}(\mu_x, \Sigma_x)$, where $\mu_x \in \mathcal{R}^3$ and $\Sigma_x \in \mathcal{R}^3$ represent the mean vector and covariance matrix. Let $y = x + \eta$, $\eta \sim \mathcal{N}(0, \sigma^2 I_3)$ be the observed noisy pixel. We integrate the information in the noisy pixel with the UDVD output by computing the mean of the posterior $p(x|y, \Omega_y)$, given by

$$p(x|y, \Omega_y) \propto p(y|x)\, p(x|\Omega_y) \tag{B.1}$$

where $p(x|\Omega_y)$ is the prior and $p(y|x)$ is the noise model. Since both the prior and the noise

model are Gaussian, we can write the optimal posterior estimate as,

$$E[x|y] = (\Sigma_x^{-1} + \sigma^{-2}I)^{-1}(\Sigma_x^{-1}\mu_x + \sigma^{-2}y). \tag{B.2}$$

Note that the posterior mean has a very intuitive interpretation. When the signal variance is high compared to noise variance (i.e. the uncertainty in our estimation is high) the posterior mean favours noisy pixel value. We estimate μ_x and Σ_x as a function of the neighbourhood Ω_y using the network architecture discussed earlier. If x is a grayscale image, then the output of the network consists of two channels - one for μ_x and one for σ_x. When the input image has k channels, the output consists of k channels for μ_x and $k(k-1)/2$ for the upper-triangular entries of Σ_x

One can estimate μ_x and Σ_x directly from the noisy data by maximizing the likelihood. Using our distributional assumptions, the noisy pixels y follows a Gaussian distribution, $y \sim \mathcal{N}(\mu_y, \Sigma_y)$, where $\mu_y = \mu_x$ and $\Sigma_y = \Sigma_x + \sigma^2 I$. Therefore, the loss function or the negative log likelihood is:

$$\mathcal{L}(\mu_x, \Sigma_x) = \frac{1}{2}[(y-\mu_x)^T(\Sigma_x + \sigma^2 I)^{-1}(y-\mu_x)] + \frac{1}{2}\log|\Sigma_x + \sigma^2 I|. \tag{B.3}$$

If σ is unknown during training and has to be estimated, we use a separate neural network with the same architecture to do so. In such cases, we add a small regularization term equal to -0.1σ for numerical stability, following [68].

For the experiments with real data, the noise distribution is unknown, so we simply ignore the central pixel.

B.1.3 Architecture and Training

Architecture: The overall architecture is explained in Section 3.3. The network architecture for the D1 and D2 blocks is described in Table B.1. D1 has $k_1 = 9$ input channels and $k_2 = 32$ output

Name	N_{out}	Function
Input	k_1	
enc_conv_0	48	Convolution 3×3
enc_conv_1	48	Convolution 3×3
enc_conv_2	48	Convolution 3×3
pool_1	48	MaxPool 2×2
enc_conv_3	48	Convolution 3×3
enc_conv_4	48	Convolution 3×3
enc_conv_5	48	Convolution 3×3
pool_2	48	MaxPool 2×2
enc_conv_6	96	Convolution 3×3
enc_conv_7	96	Convolution 3×3
enc_conv_8	48	Convolution 3×3
upsample_1	48	NearestUpsample 2×2
concat_1	96	Concatenate output of pool_1
dec_conv_0	96	Convolution 3×3
dec_conv_1	96	Convolution 3×3
dec_conv_2	96	Convolution 3×3
dec_conv_3	96	Convolution 3×3
upsample_2	96	NearestUpsample 2×2
concat_2	$96+k_1$	Concatenate output of Input
dec_conv_4	96	Convolution 3×3
dec_conv_5	96	Convolution 3×3
dec_conv_6	96	Convolution 3×3
dec_conv_7	k_2	Convolution 3×3

Table B.1: Network architecture used for UDVD. The convolution and pooling layers are the blind-spot variants described in Section B.1.1. k_1 and k_2 represent the number of input and output channels of the base network respectively.

channels. D2 has $k_1 = 96$ input channels and $k_2 = 96$ output channels. The architecture of D1 and D2 are analogous to the blocks in FastDVDnet [130] to facilitate fair comparison with the supervised models. As described in Figure 3.2, D2 is followed by a derotation and the output is passed to a series of three cascaded 1×1 convolutions and non-linearity for reconstruction with 4 and 96 intermediate output channels, as in [68]. The final convolutional layer is linear and has 9 output channels, 3 representing the RGB value of the denoised image and 6 representing its covariance matrix. We use the same architecture for fluorescence microscopy and electron microscopy with the number of input channels to UDVD modified to 5 and number of output channels modified to 1.

Training Details: Following the convention in image and video denoising, we train UDVD on 128×128 patches extracted from our dataset [68, 96, 129, 130, 150] (this is also consistent with the training methodology of the supervised baselines). For the natural video and fluorescence microscopy datasets, no data augmentation was applied. For electron microscopy dataset, we applied spatial flipping, time reversal and time subsampling (i.e. skipping frames).

Optimization Details: All networks were trained using Adam [64] optimizer with a starting learning of 10^{-4}. The learning rate was decreased by a factor of 2 at checkpoints $[20, 25, 30]$ during a total training of 40 epochs. We did not experiment with other learning rate schedules such as cosine scheduling, which is a popular choice in unsupervised image denoising [68].

B.2 Ablation Study on Number of Input Frames

We perform an ablation study on the number of frames k UDVD uses as input, $k \in \{1, 3, 5\}$. UDVD with $k = 1$ is equivalent to the blind-spot network proposed for image denoising in [68]. In this section we describe the architectural and training details for UDVD with $k \in \{1, 3, 5\}$ and

	DAVIS				Set8			
	Supervised CNN	Unsupervised CNN (UDVD)			Supervised CNN	Unsupervised CNN (UDVD)		
σ	5 frames	1 frame	3 frames	5 frames	5 frames	1 frame	3 frames	5 frames
20	35.86	34.13	34.91	35.16	33.37	32.39	33.09	33.36
30	34.06	32.80	33.48	33.92	31.60	30.91	31.62	32.01
40	32.80	31.48	32.20	32.68	30.37	29.63	30.42	30.82
50	31.83	30.47	31.20	31.70	29.42	28.65	29.47	29.89
60	31.01	29.65	30.39	30.90	29.08	27.86	28.70	29.13
70	30.21	28.96	29.70	30.22	28.37	27.20	28.06	28.49
80	29.28	28.37	29.10	29.63	27.60	26.65	27.50	27.94

Table B.2: Performance of UDVD. Table shows the mean PSNR values of a state-of-the-art supervised video denoiser (FastDVDnet [130]) and UDVD with the denoised frame being predicted from $k \in \{1, 3, 5\}$ surrounding frames. The performance of UDVD monotonically increases with k and is comparable for supervised denoising across all noise levels. All the three UDVD networks reported here are trained for only $\sigma = 30$. FastDVDnet is trained for $\sigma \in [5, 55]$.

present some additional results.

Architectural Details: UDVD with $k = 1$ contains only one UNet style network in each branch with architecture described in Table B.1 and Section B.1.3. There are 3 input channels and 9 output channels (3 for the RGB channels in each denoised pixel and 6 for the corresponding covariance matrix). UDVD with $k = 3$ has a similar architecture as for $k = 1$ but has 9 input channels instead (3 channels for each frame). The architecture for $k = 5$ is described in Section B.1.3.

Training Details: UDVD with $k \in \{1, 3, 5\}$ was trained on the DAVIS dataset with $\sigma = 30$. The training details were as described in Section B.1.3.

Results: As shown in Table 3.1 and Table B.2 performance improves substantially and monotonically with k (the number of surrounding frames used to denoise each frame) across a wide range of noise levels. This difference in performance can also be observed visually. Fig B.1 shows an example where the texture details of the brick wall and the fence are not well recovered when using only a single noisy frame. The texture is estimated better when using 5 noisy frames to

predict the denoised output.

B.3 Denoising Results on Natural Video Datasets

In this section we provide additional comparisons between UDVD and supervised CNN-based methods.

1. Table B.2 shows the performance of UDVD trained at $\sigma = 30$, and FastDVDnet trained for $\sigma \in [5, 55]$ when evaluated on the DAVIS test set and Set8 corrupted with $\sigma \in \{20, 40, \ldots, 80\}$. UDVD achieves comparable performance to FastDVDnet on DAVIS test set and slightly outperforms it on Set8 at all noise levels.

2. Examples of noisy videos, and denoised counterparts obtained using UDVD are included in the official github repository[1] (`hypermooth.mp4, rafting.mp4, motorbike.mp4` and `snowboard.mp4`).

B.4 UDVD-S: Denoising Using Only a Single Video

UDVD, combined with aggressive data augmentation and early stopping, achieves state-of-the-art performance even when trained on only a single short video. In this section, we analyze the contribution of each of the data augmentation and early stopping scheme to the performance of UDVD-S through an ablation study. We also provide more details about our comparison to MF2F [29].

B.4.1 Details of test sets.

We evaluate UDVD-S and baselines on the following four datasets:

[1] https://github.com/sreyas-mohan/udvd

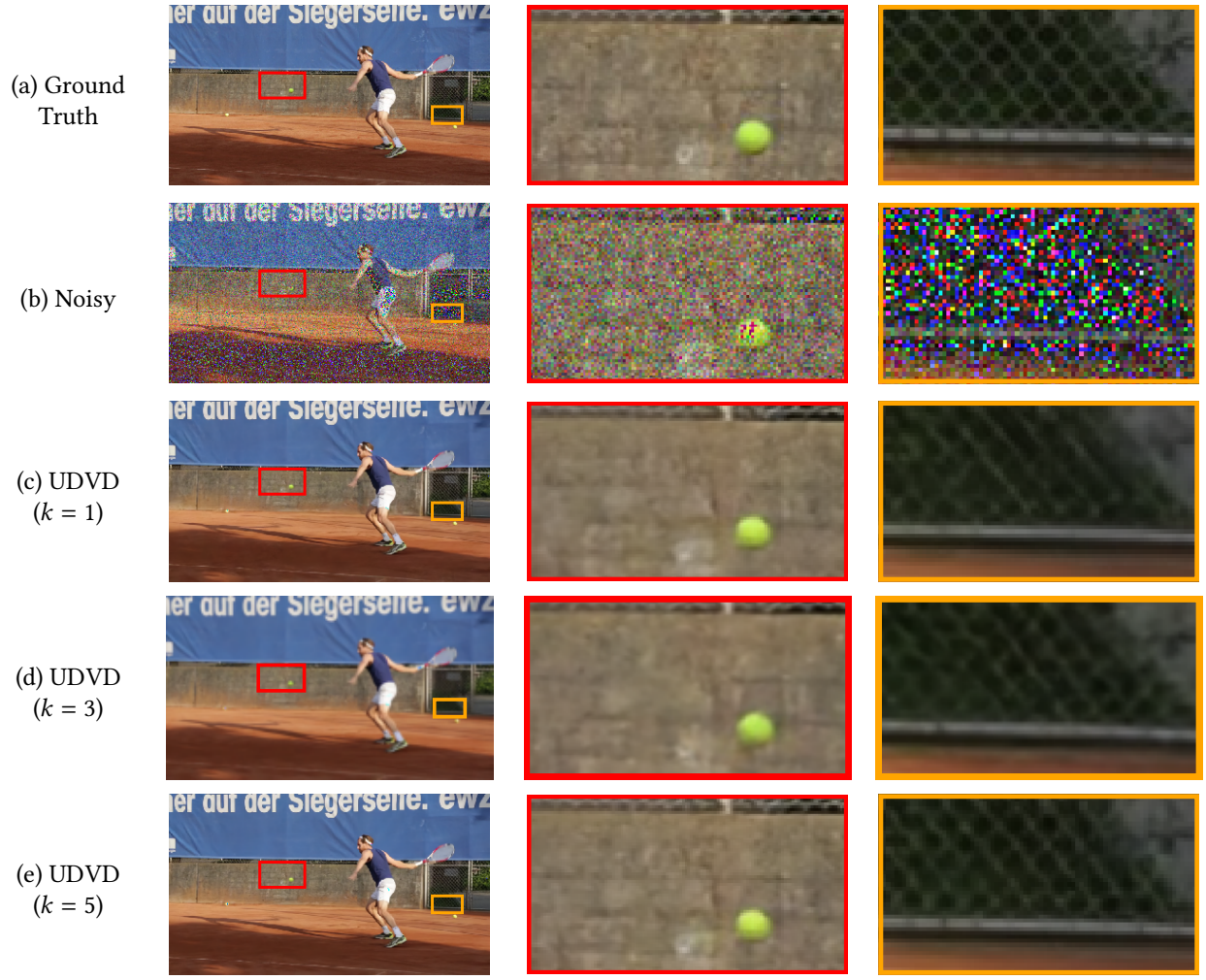


Figure B.1: Comparison of blind image and video denoising. Example from the DAVIS dataset. (a) Ground truth frame. (b) Noisy frame. (c) Reconstruction using a single frame. The texture details of the brick wall and the fence are not recovered well. Reconstruction using (d) 3 and (e) 5 surrounding frames produces an improved texture estimate.

		$\sigma = 30$									
		ten-v	snow	hyper	raft	motor	trac	sunf	touch	park	**mean**
No. of frames		75	59	37	29	32	85	85	85	85	-
No Aug	(without ES)	33.37	29.10	29.72	27.26	27.28	32.52	35.07	32.65	30.20	30.80
No Aug	(with ES)	34.35	30.67	32.42	30.72	29.21	33.08	37.04	33.63	30.40	32.39
F	(without ES)	34.00	30.60	30.15	30.16	28.44	33.09	36.86	33.56	30.37	31.91
F	(with ES)	34.68	30.76	32.41	30.76	29.33	33.35	37.13	33.74	30.53	32.52
F+TR	(without ES)	34.18	30.73	31.06	30.31	28.98	33.53	**37.29**	33.51	**30.56**	32.24
F+TR	(with ES)	*34.70*	*30.78*	**32.60**	*30.80*	**29.36**	**33.54**	**37.29**	**33.88**	**30.56**	**32.61**
UDVD*		**34.82**	**30.83**	32.34	30.82	29.24	31.73	35.33	33.48	28.98	31.95
FastDVDnet*		34.58	30.78	32.48	**30.94**	29.35	31.39	35.06	33.71	28.73	31.89
MF2F - 8 sigmas		34.45	30.44	30.93	29.70	28.81	31.61	34.43	33.41	28.79	31.40
MF2F - online no teacher		34.50	30.42	30.54	29.45	28.40	32.11	35.19	33.47	28.89	31.44
MF2F - online with teacher		34.48	30.44	31.13	*29.91*	*28.92*	32.08	35.20	33.44	28.91	31.61
MF2F - offline no teacher		*34.66*	30.49	30.20	29.38	28.36	*32.19*	35.50	33.58	28.98	31.48
MF2F - offline with teacher		34.63	*30.52*	*31.16*	29.55	*28.92*	31.93	*35.52*	*33.61*	*29.04*	*31.65*

Table B.3: Results for UDVD and MF2F trained on individual noisy videos for $\sigma = 30$. The top block show PSNR values for UDVD trained on each individual video sequence with and without data augmentation (spatial flipping(F) and time-reversal(TR)) and early stopping (ES). Early stopping was performed using the last 5 frames of each video as the held-out set. The last block shows the result of running MF2F [29] with all the 5 different fine-tuning scheme proposed in Ref. [29]. With the augmentations and early stopping, UDVD-S, on average outperforms UDVD and FastDVDnet trained on the full DAVIS dataset (indicated by *) and MF2F which fine-tunes a pre-trained FastDVDNet on each individual video. The best performing method for each video is highlighted in bold and the best performing method in each block is highlighted in italics. The tennis-vest video is from DAVIS and the rest of the 8 videos are from Set8.

1. **DAVIS [106]**: We take all the 30 sequences from the test set of the DAVIS Challenge 2017.

2. **Set8 [130]**: Following FastDVDNet [130], we use 4 sequences from the GoPro set (*hypersmooth, motorbike, rafting, snowboard*) and 4 sequences from the Derfs Test Media Collection (*park_joy, sunflower, touchdown, tractor*).

3. **Derfs**: Following [29], we use 7 sequences from the Derfs Test Media Collection, which are *park_joy, sunflower, touchdown, tractor* (shared with Set8), and *blue_sky, old_town_cross, pedestrian_area*. We use the first 85 frames from each sequences with a spatial-resolution of 960×540 [130].

		$\sigma = 90$									
		ten-v	snow	hyper	raft	motor	trac	sunf	touch	park	**mean**
No. of frames		75	59	37	29	32	85	85	85	85	-
No Aug	(without ES)	24.13	22.89	22.04	20.99	20.06	24.84	25.98	25.67	23.35	23.33
No Aug	(with ES)	30.15	25.49	27.48	26.05	23.79	28.18	31.91	29.87	25.46	27.60
F	(without ES)	27.21	24.42	24.05	23.32	21.84	27.42	29.53	28.01	25.03	25.65
F	(with ES)	30.35	**25.60**	27.72	*26.16*	23.89	**28.71**	32.17	29.93	25.59	27.79
F+TR	(without ES)	27.11	24.77	24.25	23.55	21.98	27.80	30.22	28.56	25.44	25.96
F+TR	(with ES)	**30.40**	25.59	**27.75**	*26.16*	**23.92**	28.63	**32.18**	**29.96**	**25.62**	**27.80**
UDVD*		28.78	25.16	26.78	25.81	23.57	26.42	29.04	28.71	24.23	26.50
FastDVDnet*		29.44	25.25	27.30	**26.35**	23.68	27.42	30.29	29.61	24.72	27.12
MF2F - 8 sigmas		28.79	25.04	27.14	26.21	23.56	26.89	29.19	29.04	24.35	26.69
MF2F - online no teacher		28.35	25.12	26.67	26.07	23.39	27.28	30.01	29.49	24.64	26.78
MF2F - online with teacher		*29.44*	*25.25*	*27.30*	**26.35**	*23.68*	*27.42*	30.09	29.53	24.71	*27.08*
MF2F - offline no teacher		28.70	25.17	26.64	26.02	23.41	27.42	*30.29*	29.60	*24.72*	26.89
MF2F - offline with teacher		28.79	*25.25*	27.22	26.31	23.62	27.34	*30.29*	*29.61*	24.69	27.01

Table B.4: Results for UDVD and MF2F trained on individual noisy videos for $\sigma = 90$. The top block show PSNR values for UDVD trained on each individual video sequence with and without data augmentation (spatial flipping(F) and time-reversal(TR)) and early stopping (ES). Early stopping was performed using the last 5 frames of each video as the held-out set. The last block shows the result of running MF2F [29] with all the 5 different fine-tuning scheme proposed in Ref. [29]. With the augmentations and early stopping, UDVD-S, on average outperforms, UDVD or FastDVDnet trained on the full DAVIS dataset (indicated by *) and MF2F which fine-tunes on a pre-trained FastDVDNet on each individual video. The best performing method for each video is highlighted in bold and the best performing method in each block is highlighted in italics. The tennis-vest video is from DAVIS and the rest of the 8 videos are from Set8.

4. **Vid3oC [62]**: We use the first 10 sequences (*000 to 009*) out of the 50 available sequences.

B.4.2 Ablation study

We train UDVD-S on 128×128 patches extracted from the noisy video. (see Section B.1.3) for more details). For each patch, we apply each of the following data augmentations at random:

1. **Spatial flipping**: We flip all the 5 input patches vertically or horizontally. This operation only rearranges the pixel location and does not combine the pixel together in anyway, making sure that the noise statistics is still preserved after the augmentation.

2. **Time reversal**: We reverse the order of frames in the input to generate a new video. Like spatial flipping, this operation also preserves the noise statistics.

In addition to data augmentation, we employ early stopping by choosing the model parameters which produced the best error on a a held-out set of frames. We pick the last 5 frames of each video as our held out set. Tables B.3 and B.4 show an ablation study over data augmentations and early stopping for 9 different videos at two different noise levels, $\sigma = 30$ and $\sigma = 90$. Across videos and noise levels, data augmentation and early stopping significantly increase the performance of our method.

B.5 Denoising Results on Real-world Datasets

Raw videos: The estimated ground truth, noisy raw data [147], and the denoised videos obtained with UDVD can be found on the official github repository (`raw_video.mp4`). The videos were converted to RGB for illustration.

As discussed in Section 3.5, UDVD was directly trained on the mosaiced raw videos. Existing unsupervised video denoising methods, like MF2F [29], cannot be applied directly on this dataset as their pre-trained backbone expects an input in the RGB domain. In Ref. [29], the authors convert mosaiced videos into the RGB domain, apply MF2F [29] and transform the denoised RGB videos back.

Fluorescence and electron microscopy data: The noisy fluorescence microscopy and electron microscopy data, and the denoised videos obtained with UDVD can be found on the official github repository (`fluoro_1.mp4, fluoro_2.mp4` and `electron.mp4`).

B.6 Generalization Across Noise and Frame Rate

Ideally, a denoiser should be able to denoise videos corrupted at a wide range of noise levels. This is usually achieved by training the CNN on examples corrupted with a range of noise strength [129, 130, 150]. The range of noise levels on which the network is trained is called the *training range* of the network.

Generalization outside the training range: The authors of [96] showed that CNNs trained for image denoising generalize well on test images corrupted with noise in the training range, but fails catastrophically when corrupted with noise strength outside the training range. The authors provided evidence that the overfitting is due to additive terms in the convolutional layers (and BatchNorm [55]) and showed that a CNN with no additive terms, called a *bias-free* CNN generalizes well outside the training range. UDVD uses a bias-free architecture and generalizes well to noise levels outside its training range (Fig B.2).

Generalization across frame rates: To test generalization across frame rates, we simulated faster videos by skipping frames of videos in Set8. Fig B.2 shows that UDVD generalizes robustly to faster videos and maintains a significant gain in performance over single-image denoising even when tested on videos where a large number of frames have been skipped (i.e. at a very low frame rate).

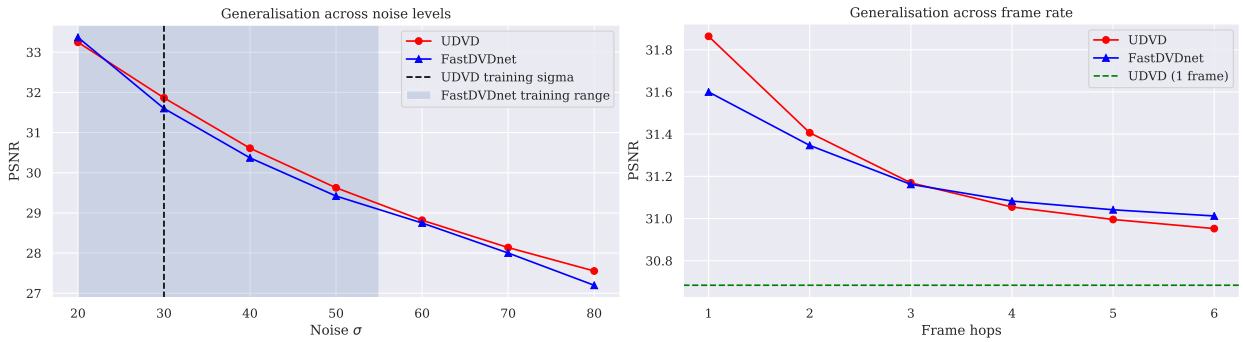


Figure B.2: Generalization across noise levels and frame rates. (left) UDVD trained at only $\sigma = 30$ generalizes well to noise levels not seen during training. The plotted points represent mean PSNR values evaluated on Set8. (right) UDVD generalizes well to faster videos (created by skipping frames) and consistently outperforms a baseline image denoiser (UDVD with a single input frame, shown as a green dashed line).

B.7 Analysis of CNN-based Video Denoising

B.7.1 Natural Videos

In Section 3.6 we examined the equivalent filters and illustrated that UDVD learns to denoise by performing an average over a spatiotemporal neighbourhood of each pixel. Here we examine equivalent filters for more videos and a supervised CNN (FastDVDnet) and show that similar observations hold.

Adaptive filtering: Fig B.4 and B.5 shows filters computed at a pixel for 2 different videos at 4 different noise levels. The filters adapt to the underlying signal content. They span larger areas as the noise level increases. These observations also holds for FastDVDnet, which is trained with supervision (Fig B.6)

Contribution of neighbouring frames for denoising: UDVD tends to ignore temporally distant frames at lower noise levels as shown in Fig B.4 and B.5. This phenomenon is quantified in Fig B.3 by plotting the contribution of each frame to the denoised pixel by averaging over **5000**

pixels from **250** random patches of size 128×128. At higher noise levels, UDVD seems to use distant frames more. This is consistent with the ablation study, which shows that for higher noise levels using more surrounding frames improves the denoising performance. Similar results hold for supervised CNN FastDVDnet, as shown in Fig B.6.

Local Averaging: The weighting functions or equivalent filters perform an approximate averaging operation. They are mostly non-negative (although they do have some negative entries as depicted in blue in Fig B.4, B.5, **??** and **??**) and they approximately sum up to one (see Fig B.3).

B.7.2 Real-world Data

Equivalent filters for the raw video, the fluorescence-microscopy and the electron-microscopy data are shown in Fig B.7. The fluorescence -microscopy data have a low noise level. As expected from the results on natural videos (see Section B.3), the weighting functions are mostly confined to the middle frame (as quantified in Fig B.3). In the electron-microscopy dataset the weighting functions shows that the network relies on adjacent frames to estimate the denoised (as quantified in Fig B.3).

B.7.3 Motion Estimation

Figures B.4 and B.5 show that the equivalent filters in adjoining frames are automatically shifted spatially to account for the movement of objects in the videos. We extracted motion information using the shift as explained in Section 6. Figures B.8, B.9, **??** and **??** show additional examples for UDVD and FastDVDnet. The estimated optical flow is mostly consistent with the estimated obtained by DeepFlow [139] applied on the clean videos. The motion estimates obtained from the equivalent filters tends to be less accurate for pixels near strongly correlated features or highly homogeneous regions where the local motion is ambiguous.

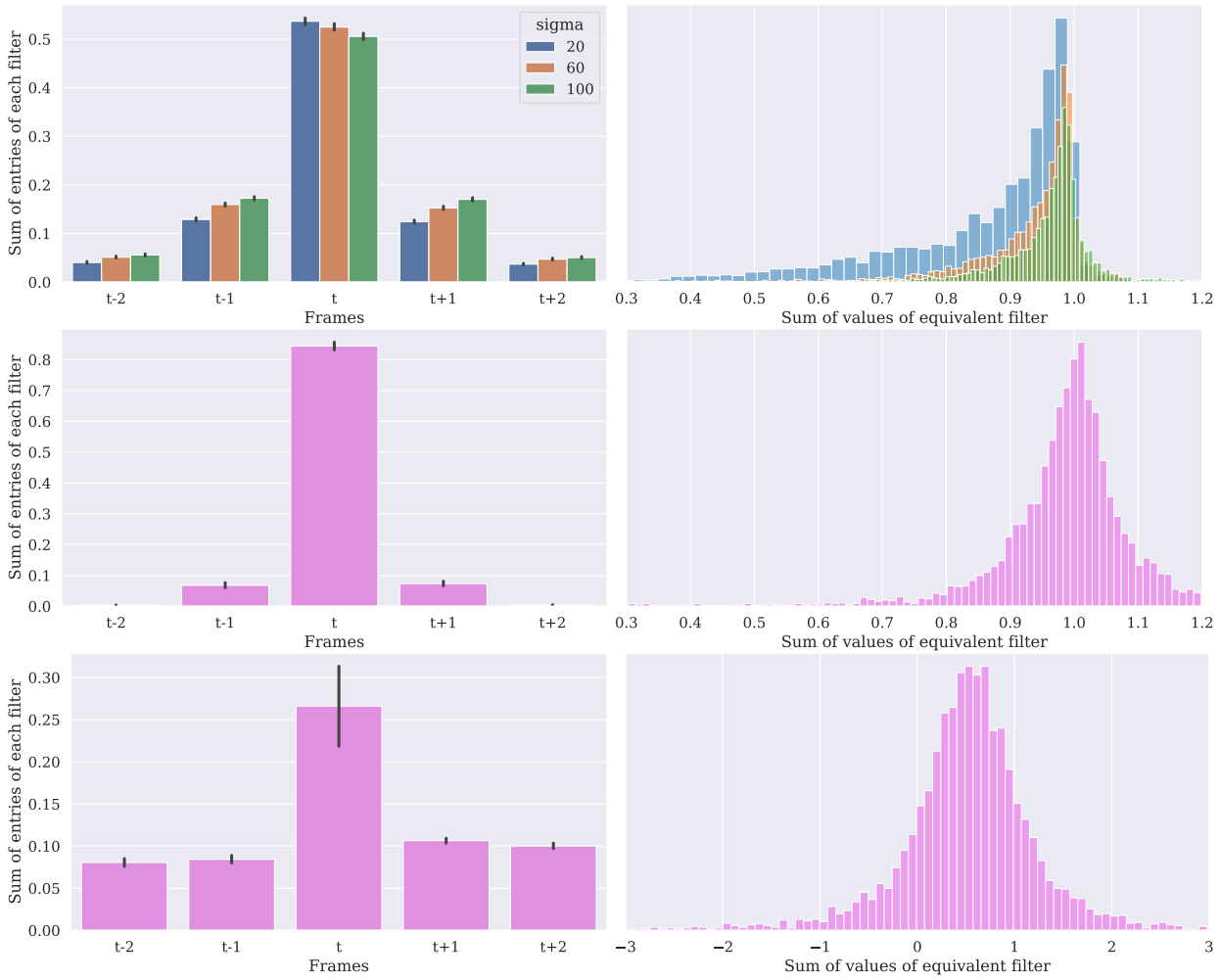


Figure B.3: Quantitative analysis of equivalent filters. *Left column:* The graphs show the sum of the entries of the equivalent filters in each frame, averaged over 5000 pixels from 250 random patches of size 128×128. For all datasets, the central frame dominates. For the DAVIS dataset (top), the contribution from the other frames increases with the noise level. For the fluorescence-microscopy data (mid) the contribution of the other frames is rather low, due to the high signal-to-noise ratio. For the electron-microscopy dataset the contribution of the other frames is larger (bottom). *Right column:* Histogram of the sum of all entries in the equivalent filters (over all 5 frames) for 5000 pixels from 250 random patches of size 128×128 from the DAVIS test set (top), the fluorescence-microscopy dataset (mid) and the electron-microscopy dataset (bottom). For the DAVIS and fluorescence-microscopy datasets, the filters sum to 1 in most cases. The peak of electron microscopy deviates significantly from 1. This could be due to the noise model, which has non-Gaussian characteristics (it is Poisson with low counts).

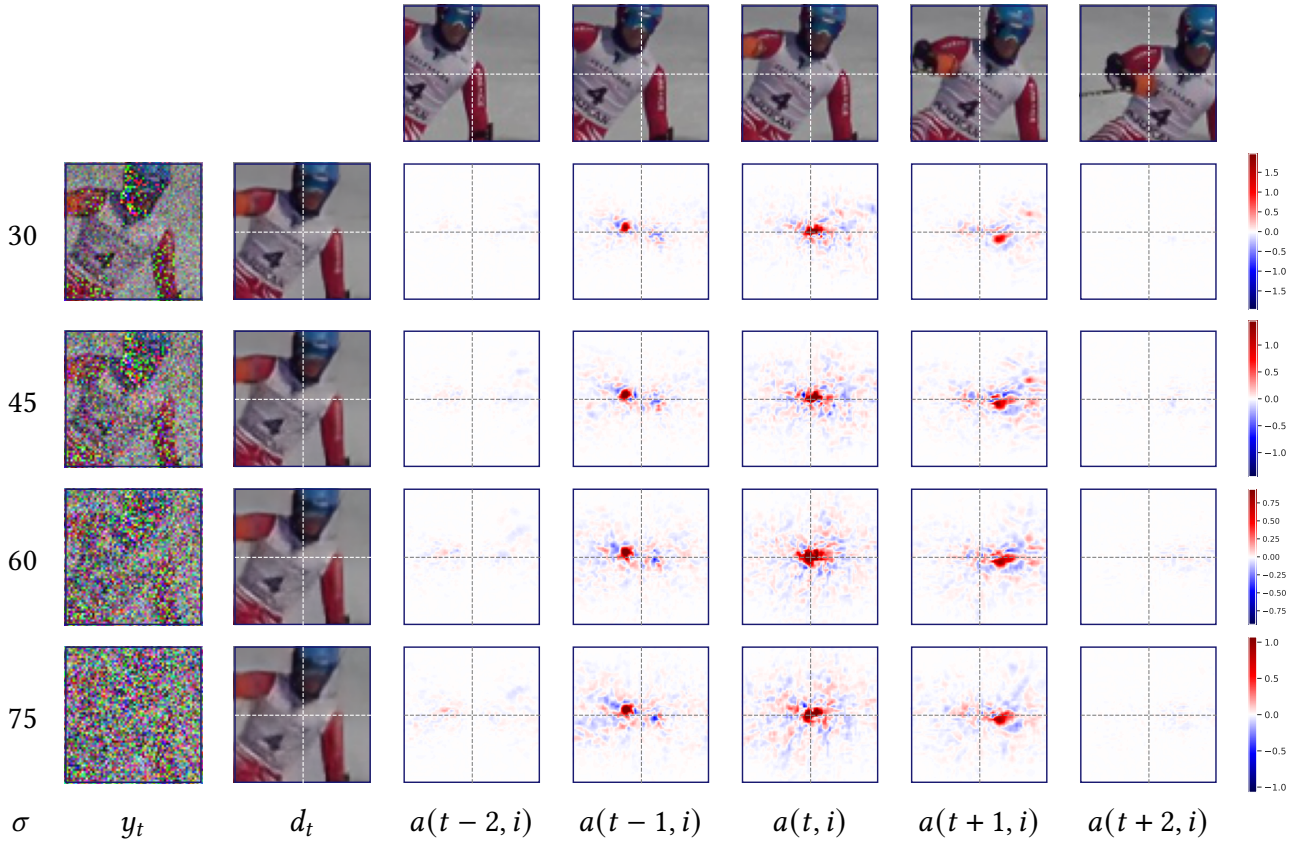


Figure B.4: Video denoising as spatiotemporal adaptive filtering; `giant-slalom` video from the DAVIS dataset. Visualization of the linear weighting functions ($a(k,i)$, Section 3.6) of UDVD. The left two columns show the noisy frame y_t at four levels of noise, and the corresponding denoised frame, d_t. Weighting functions $a(k,i)$ corresponding to the pixel i (at the intersection of the dashed white lines), for five successive frames, are shown in the last five columns. The weighting functions adapt to underlying image content, and are shifted to track the motion of the skier. As the noise level σ increases, their spatial extent grows, averaging out more of the noise while respecting object boundaries. The weighting functions corresponding to the five frames approximately sum to one, and thus compute a local average (although some weights are negative, depicted in blue) as explained in Section B.7.1.

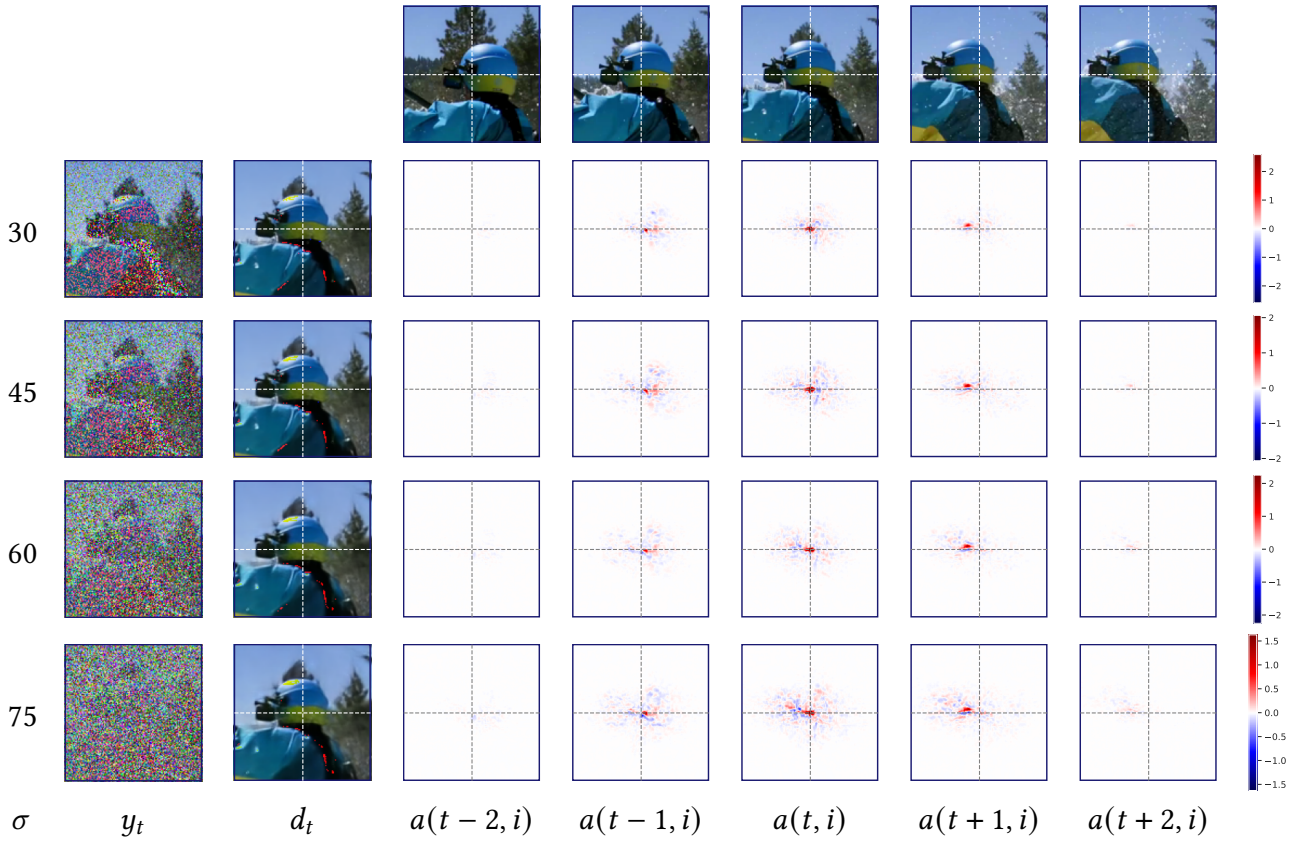


Figure B.5: Video denoising as spatiotemporal adaptive filtering; `rafting` video from the GoPro dataset. Visualization of the equivalent filters, as described in Fig B.4.

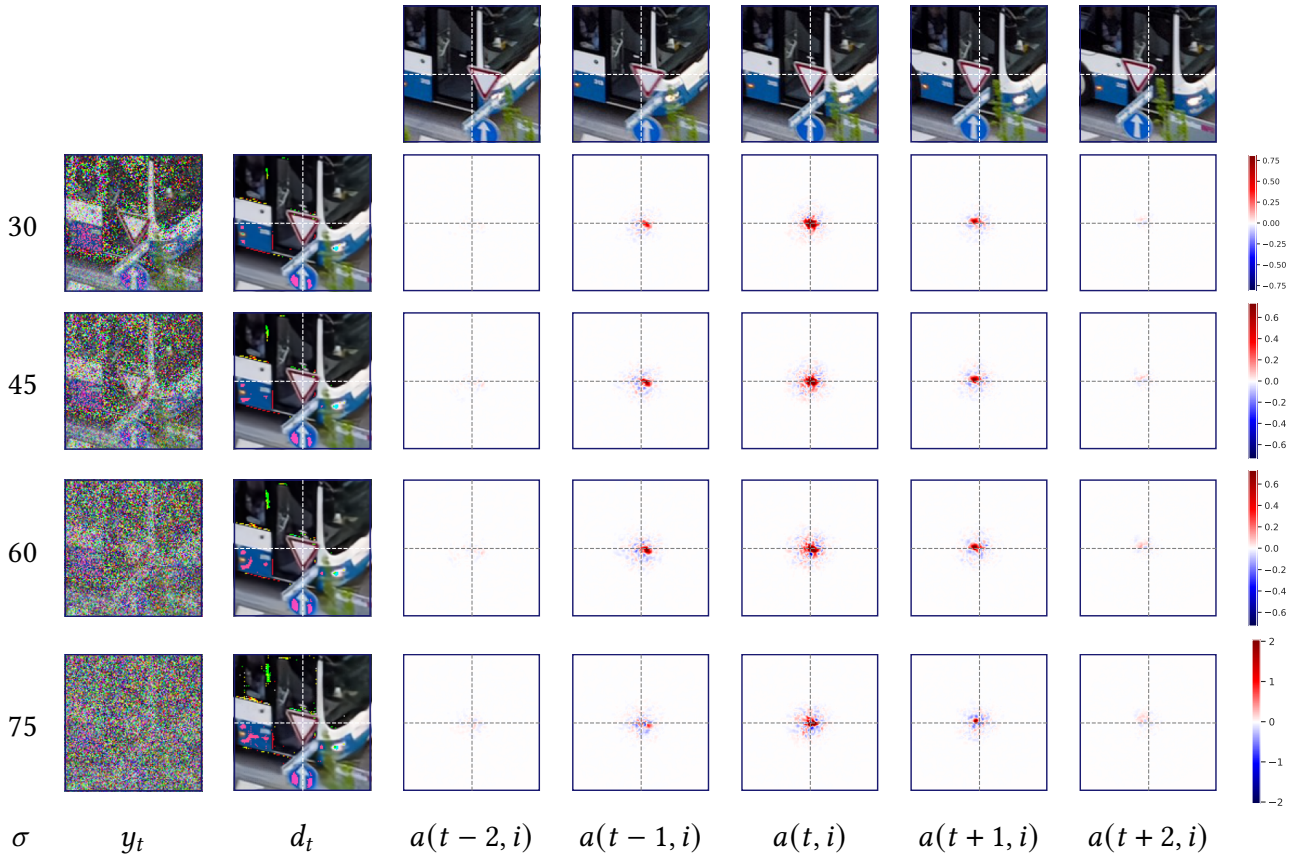


Figure B.6: Video denoising using FastDVDnet as spatiotemporal adaptive filtering; bus video from the DAVIS dataset. Visualization of the linear weighting functions ($a(k,i)$, Section 3.6) of FastDVDnet which is trained with supervision. The left two columns show the noisy frame y_t at four levels of noise, and the corresponding denoised frame, d_t. Weighting functions $a(k,i)$ corresponding to the pixel i (at the intersection of the dashed white lines), for five successive frames, are shown in the last five columns. The weighting functions adapt to underlying image content, and are shifted to track the motion of the stop sign. As the noise level σ increases, their spatial extent grows, averaging out more of the noise while respecting object boundaries. The behavior is very similar to the corresponding filters of UDVD as shown in Fig **??**.

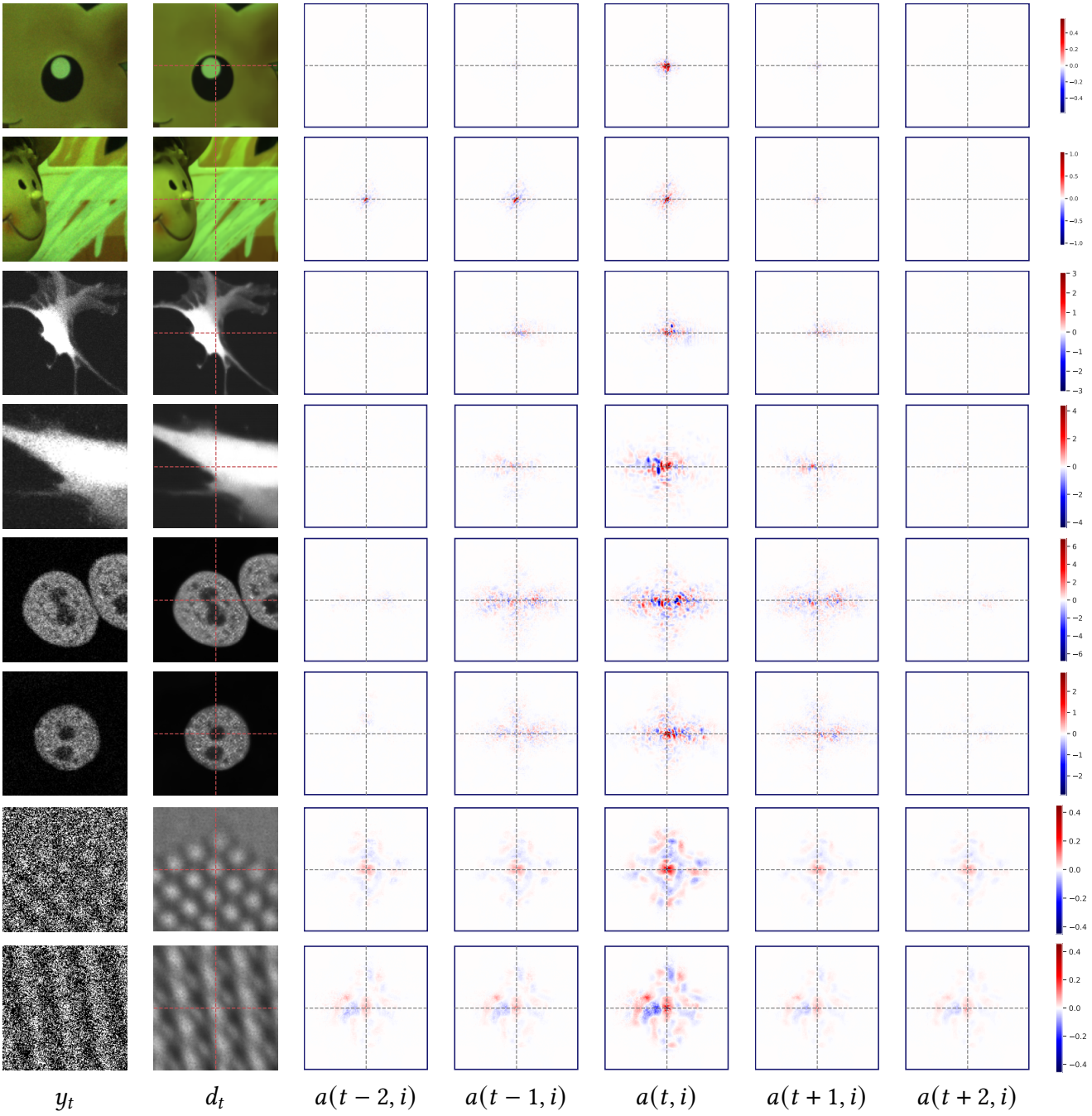


Figure B.7: Equivalent filters of UDVD when applied to real-world data. Visualization of the linear weighting functions ($a(k,i)$, Section 3.6) of UDVD trained to denoise raw video, fluorescence and electron microscopy data. The left two columns show the noisy frame y_t and the corresponding denoised frame, d_t. Weighting functions $a(k,i)$ corresponding to the pixel i (at the intersection of the dashed white lines), for five successive frames, are shown in the last five columns. In raw video data and fluorescence-microscopy data, the contributions from neighbouring frames are smaller. For electron-microscopy data they are larger (see also Fig B.3).

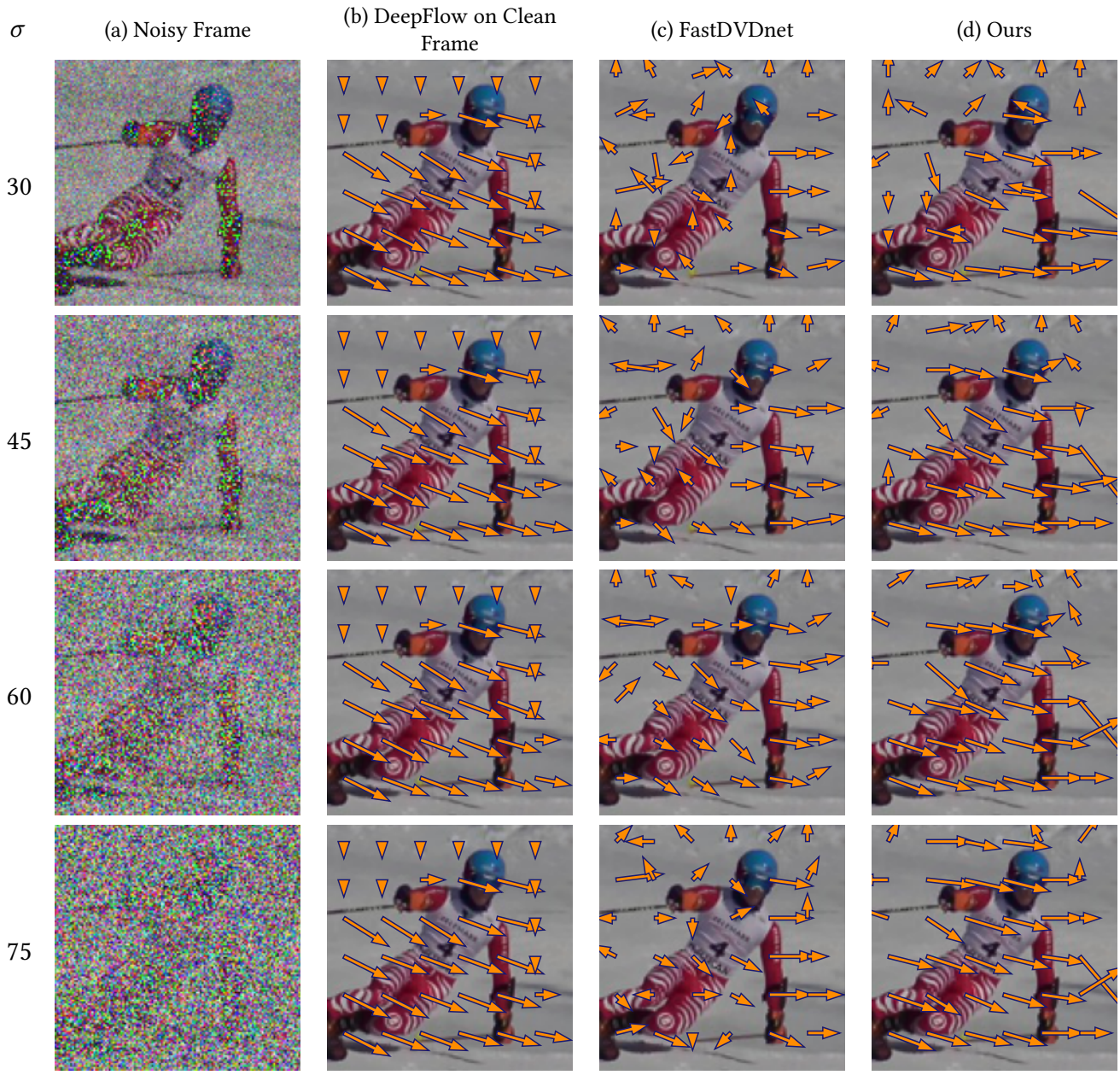


Figure B.8: CNNs trained for denoising automatically learn to perform motion estimation. (a) Noisy frame from `giant-slalom` video in the DAVIS dataset. (b) Optical flow direction at multiple locations of the image obtained using a state-of-the-art algorithm applied *to the clean video*. Optical flow direction estimated from the shift of the adaptive filter obtained from the gradients of (c) FastDVDnet and (d) UDVD, both of which are trained with no optical flow information. FastDVDnet is trained with supervision. Optical flow estimates are well-matched to those in (b), but are not as accurated at oriented features, and in homogeneous regions where local motion is not well defined (e.g. in the background). Each row corresponds to a different noise levels. At higher noise levels, the networks perform averages over more frames, improving the motion estimation results.

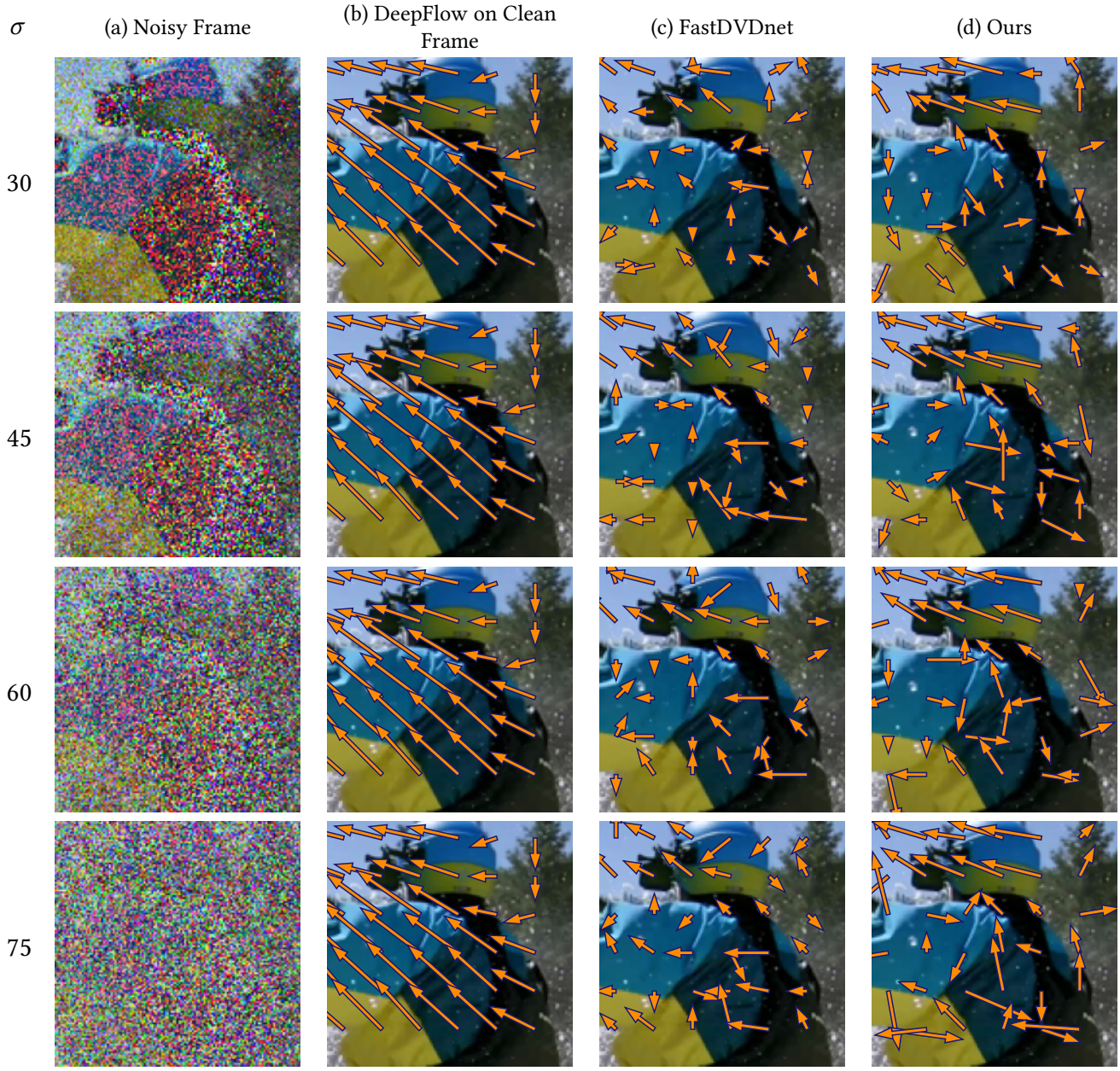


Figure B.9: CNNs trained for denoising automatically learn to perform motion estimation; `rafting` video from Set8. Motion estimated from the gradients of UDVD and FastDVDnet. See description of Figure B.8.

C | GainTuning

C.1 Datasets

We perform controlled experiments on datasets with different signal and noise structure to evaluate the broad applicability of GainTuning (see Figure C.1 for a visual summary of datasets). We describe each dataset below:

Generic natural images. We use 400 images from BSD400 [87] dataset for pre-training CNNs. We evaluate on two test sets, Set12 and Set68, with 12 and 68 images, respectively [150].

Images of urban scenes. We evaluate generalization capabilities of GainTuning using a dataset of images captured in urban settings, Urban100 [53]. These images often contain repeating patterns and structures, unlike generic natural images (see Figure C.1). We evaluate GainTuning on the first 50 images from this dataset.

Images of scanned documents. We use images of scanned documents from the IUPR dataset [17]. We resized the images in IUPR dataset by a factor of 6, and used the first 50 images from the dataset for evaluation.

Simulated piecewise constant images. We use a dataset of simulated piecewise constant images. These images have constant regions with boundaries consisting of various shapes such as circles and lines with different orientations. The constant region has an intensity value sampled from a uniform distribution between 0 and 1 (see Figure C.1). These piecewise constant images provide a crude model for natural images [73, 88, 105], and a CNN pre-trained on this dataset pro-

vides a substrate for testing the ability of GainTuning to adapt to the complexity of real-world images.

C.2 Details of pre-training and GainTuning

In this section, we describe the implementation details of the pre-training process and our proposed GainTuning framework.

While performing GainTuning, we introduce a scalar multiplicative parameter (gain) in every channel of the convolutional layers in the denoising CNN. We do not introduce gain parameters in the last layer of the network. We describe the general optimization process for GainTuning here, and describe any additional modifications for specific datasets in the supplementary material of Ref. [94].

Data. We perform GainTuning on patches extracted from the noisy image. We extracted 400×400 patches for the electron microscopy dataset, and 50×50 patches for all other datasets. We do not perform any data augmentation on the extracted patches.

BatchNorm layers during GainTuning. If the denoising CNN contains batch normalization (BN) layers (only DnCNN [150] and BFCNN [96] in our experiments), we freeze their statistics while performing GainTuning. That is, we do not re-estimate the mean and standard deviation parameter for each layer from the test data. Instead, we re-use the original values estimated from pre-training dataset.

Optimization for GainTuning. We use Adam [64] optimizer. We empirically find that training for 100 steps with a starting learning rate of 10^{-4} which is reduced to 10^{-5} after the 20^{th} step performs well across most situations (see sections below for hyper-parameters used in different experiments). Here, we define each step as a pass through 5000 random patches extracted from the test image. When performing experiments which compare optimizing all parameters to optimizing only gain during the adaptation process, we kept the learning rate constant at 10^{-5} for

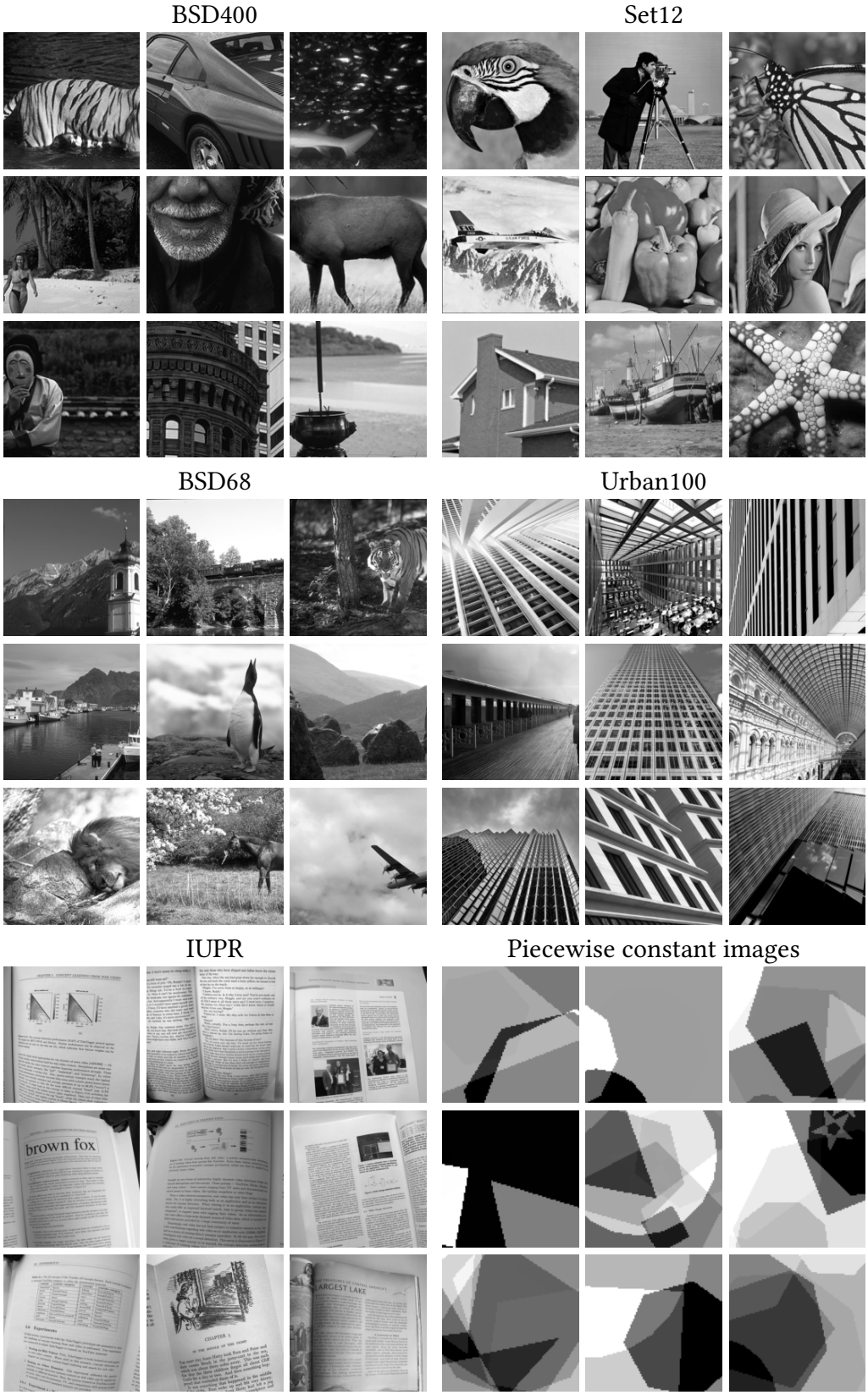


Figure C.1: Example images from different dataset. Nine images chosen at random from each dataset.

both options, and trained for 1000 steps.

C.3 Approximation for SURE

Let $\mathbf{x}$ be an N-dimensional ground-truth random vector $\mathbf{x}$ and let $\mathbf{y} := \mathbf{x}+\mathbf{n}$ be a corresponding noisy observation, where $\mathbf{n} \sim \mathcal{N}(0, \sigma_n^2\mathbf{I})$. Stein's Unbiased Risk Estimator (SURE) provides an expression for the mean-squared error between $\mathbf{x}$ and the denoised estimate $f_\theta(\mathbf{y})$ (where f_θ denotes an arbitrary denoising function), which *only depends on the distribution of noisy observations* $\mathbf{y}$:

$$\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\frac{1}{N}\left\|\mathbf{x}-f_\theta(\mathbf{y})\right\|^2\right] = \mathbb{E}_{\mathbf{y}}\left[\frac{1}{N}\left\|\mathbf{y}-f_\theta(\mathbf{y})\right\|^2 - \sigma^2 + \frac{2\sigma^2}{N}\sum_{k=1}^{N}\frac{\partial(f_\theta(\mathbf{y})_k)}{\partial \mathbf{y}_k}\right] \tag{C.1}$$

The last (divergence) term in the equation is costly to compute. Therefore, we make use of a Monte Carlo approximation of SURE introduced by Ref. [112]:

$$\sum_{k=1}^{N}\frac{\partial(f_\theta(\mathbf{y})_k)}{\partial \mathbf{y}_k} \approx \frac{1}{\epsilon N}\langle \tilde{\mathbf{n}}, f_\theta(\mathbf{y}+\epsilon\tilde{\mathbf{n}}) - f_\theta(\mathbf{y})\rangle \tag{C.2}$$

where $\langle \mathbf{x}, \mathbf{y}\rangle$ represents the dot product between $\mathbf{x}$ and $\mathbf{y}$, $\tilde{n}$ represents a sample from $\mathcal{N}(0,1)$, and ϵ represents a fixed, small, positive number. We set $\epsilon = \sigma\times 1.4\times 10^{-4}$ for our computational experiments [123]. Equation (C.2) has been used in the implementation of several traditional [112], and deep learning based [91, 123, 124] denoisers.

C.4 GainTuning prevents overfitting

We perform controlled experiments to compare test-time updating of (1) all parameters of a CNN, and (2) only the gain parameters. We briefly describe each experiment and our findings in this section.

Comparison across different cost functions. We fine-tune (a) all parameters, and (b) only gain

parameters of a DnCNN [150] model when the test image is (1) in-distribution, (2) corrupted with out-of-distribution noise and (c) contains image features which are different from the training set. Fine-tuning only the gain parameters outperforms fine-tuning all parameters in all of these situations for different choices of cost functions (see Figures C.2, C.3 and C.4)

Comparison across different architectures. We fine-tune (a) all parameters, and (b) only gain parameters of a DnCNN [150], BFCNN [96] and, UNet [116] model when the test image is (a) in-distribution, (b) corrupted with out-of-distribution noise and (c) contains image features which are different from the training set. Fine-tuning only the gain parameters often outperforms fine-tuning all parameters in all of these situations for different choices of cost functions (see Figure C.5). Figure C.5 shows results for a CNN trained on generic natural images and tested on images of urban scenes. In this case, training all parameters of the CNN outperforms training only the gains (see Section 4.7 for a discussion). Interestingly, training gains is comparable to training all parameters when we corrupt the images from urban scenes with a noise level that is also outside the training range (see Figure C.6).

GainTuning does not require early stopping. Optimizing all parameters of a CNN during adaptation often results in overfitting (see Figure C.5). In contrast, optimizing only the gain parameters for longer periods of time results improves performance without overfitting (FigureC.7).

Real electron microscopy data. We fine-tune (a) all parameters, and (b) gain parameters to adapt a CNN to real images of nanoparticle acquired through an electron microscope. The CNN was pre-trained on the simulated data described in Section C.1. Optimizing only the gain parameters outperforms optimizing all parameter and does not require early stopping (Figure C.8)

GainTuning outperforms fine-tuning last few layers of the CNN. We compared GainTuning to selectively fine-tuning last n layers for DnCNN with $n = 20$ layers. GainTuning outperformed fine-tuning last layers by a substantial margin (see Table C.1 for details). Note that gains only constitute 1.15K or 0.17% of the parameters, while fine-tuning only the last 2 layers is 37K or 5.63% parameters (about 33x more than the number of gains). The in-distribution and out-

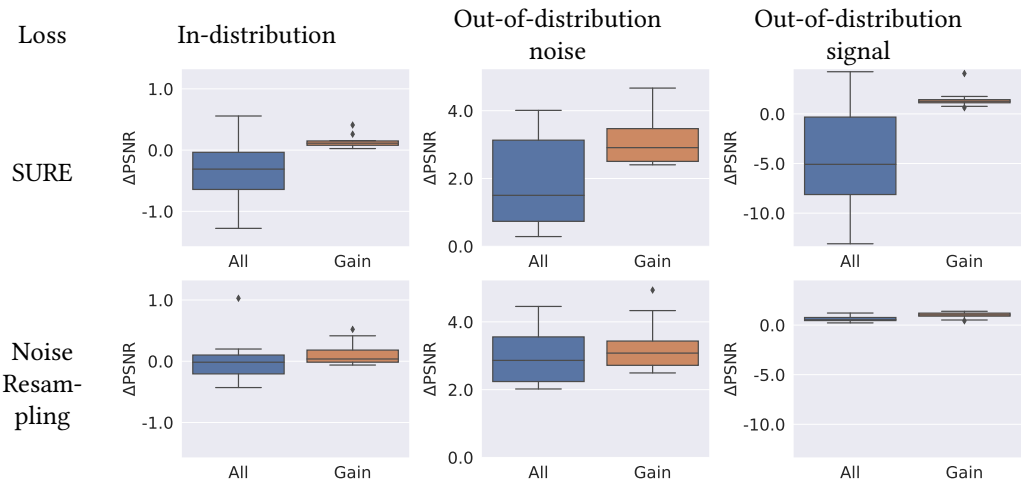


Figure C.2: GainTuning prevents overfitting. Comparison of adaptive training of all network parameters, and GainTuning (training of gains only), using two different unsupervised objectives - SURE (top) and noise resampling (bottom). The distributions of performance improvements are shown as box plots. See Figure C.3 for corresponding scatterplots. For *in-distribution*, we evaluate a CNN pre-trained on natural images corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$ on natural images (Set12) at $\sigma = 30$. For *out-of-distribution noise* we evaluate natural images (Set12) at $\sigma = 70$. For *out-of-distribution signal* we evaluate a CNN trained on piecewise constant images at $\sigma \in [0, 55]$ on natural images (set12) at $\sigma = 30$. Please refer to Section C.5 for details.

of-distribution noise consists of adapting a DnCNN trained on natural images with $\sigma \in [0, 55]$ for natural images (Set12) with $\sigma = 30$ and $\sigma = 70$ respectively. We adapted a CNN trained on piecewise constant images with $\sigma \in [0, 55]$ to natural images (Set12) with $\sigma = 30$ for out-of-distribution signal experiments.

C.5 Performance of GainTuning

C.5.1 In-distribution test image

Different architectures. We evaluated DnCNN [150], UNet [116] and BFCNN [96] architectures for this task. All models were trained on denoising Gaussian white noise of standard deviation $\sigma \in [0, 55]$ from generic natural images. Results of DnCNN and UNet are presented in Figure 4.3 in the main paper. Results for the BFCNN architecture are provided in Table C.2.

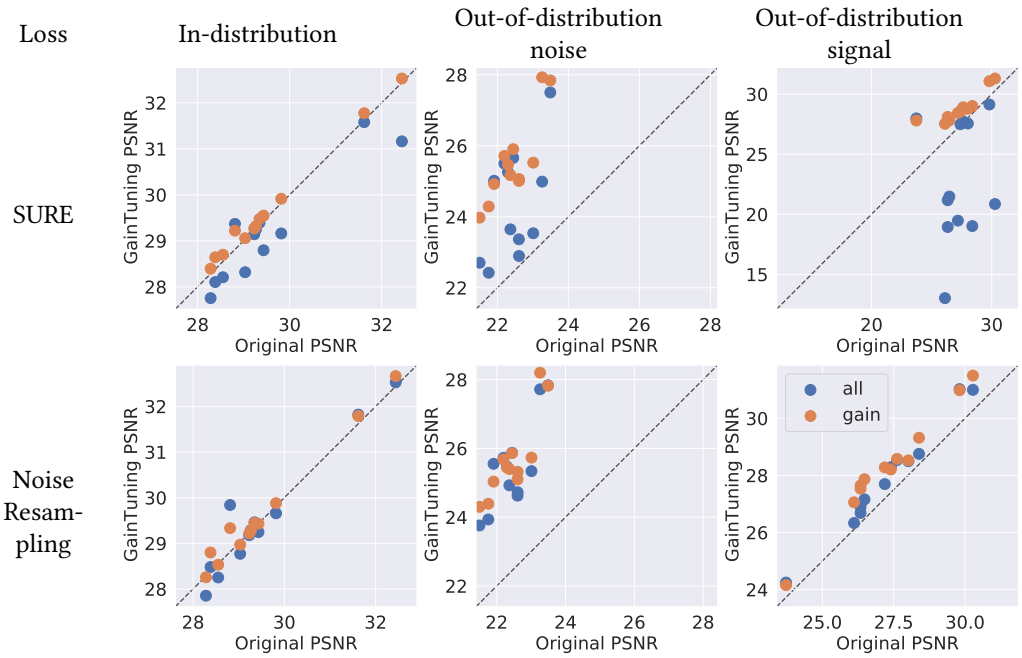


Figure C.3: GainTuning prevents overfitting. Performance obtained from adaptively training all network parameters (blue points), compared to GainTuning (orange points) using the SURE loss, plotted against performance of the originally trained network. Each data point corresponds to one image in the dataset. The dashed line represents the identity (i.e., no improvement). Training all parameters (blue points) often leads to degraded performance, but training only the gains (orange points), leads to an improvement. For *in-distribution* test images, we evaluate a CNN pre-trained on natural images corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$ on natural images (Set12) at $\sigma = 30$. For *out-of-distribution noise* we test on natural images (Set12) at $\sigma = 70$. For *out-of-distribution signal* we test a CNN trained on piecewise constant images at $\sigma \in [0, 55]$ on natural images (set12) at $\sigma = 30$. Please refer to Section C.5 for details.

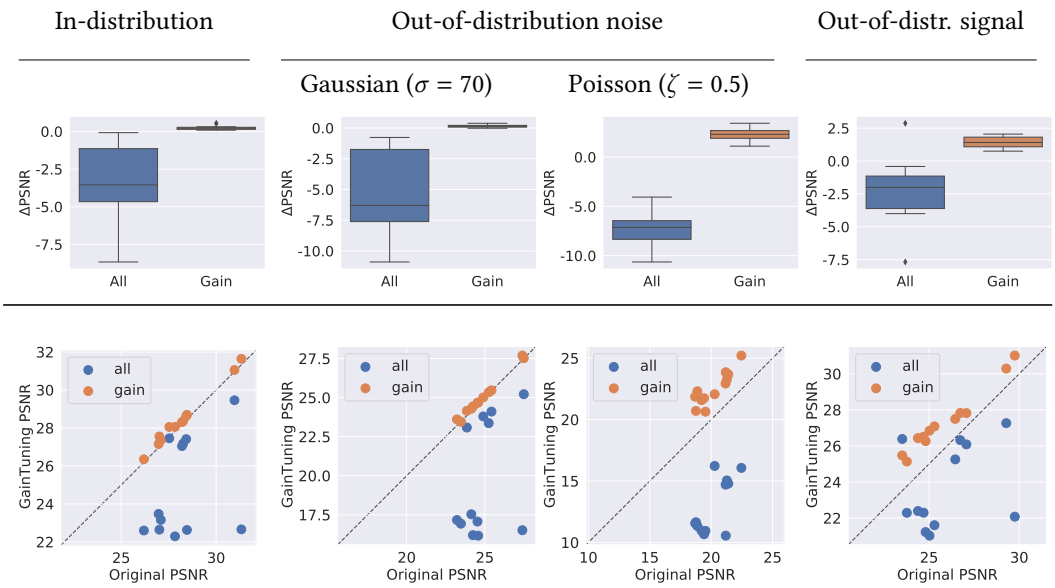


Figure C.4: GainTuning prevents overfitting. Comparison of adaptive training of all network parameters, and GainTuning (training of gains only) using **blind-spot** cost function. The distribution of the gain in performance is visualized as a box plot. For *in-distribution*, we evaluate a CNN pre-trained on natural images corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$ on natural images (Set12) at $\sigma = 30$. For *out-of-distribution noise* we evaluate natural images (Set12) at $\sigma = 70$ (Gaussian noise), and $\zeta = 0.5$ for Poisson noise. For *out-of-distribution signal* we evaluate a CNN trained on piecewise constant images at $\sigma \in [0, 55]$ on natural images (Set 12) at $\sigma = 30$. We used network architecture in [68] for our experiments.

Different cost functions. We provide the results of evaluating DnCNN architecture with different cost functions in Table C.6.

C.5.2 Out-of-distribution noise

Different Architectures. We summarize the results using DnCNN in Table 4.4 in the main paper. Figure C.5 shows that the UNet architecture is also able to generalize to out-of-distribution noise.

Different Loss Functions. We provide the results of evaluating DnCNN architecture with different cost functions in Table C.6.

Comparison to baselines. Table C.3 summarizes the result of evaluating a DnCNN trained on generic natural images for $\sigma \in [0, 55]$ on a test set of generic natural images corrupted with $\sigma = \{70, 80\}$, which is outside the training range of the network. GainTuning is able to generalize

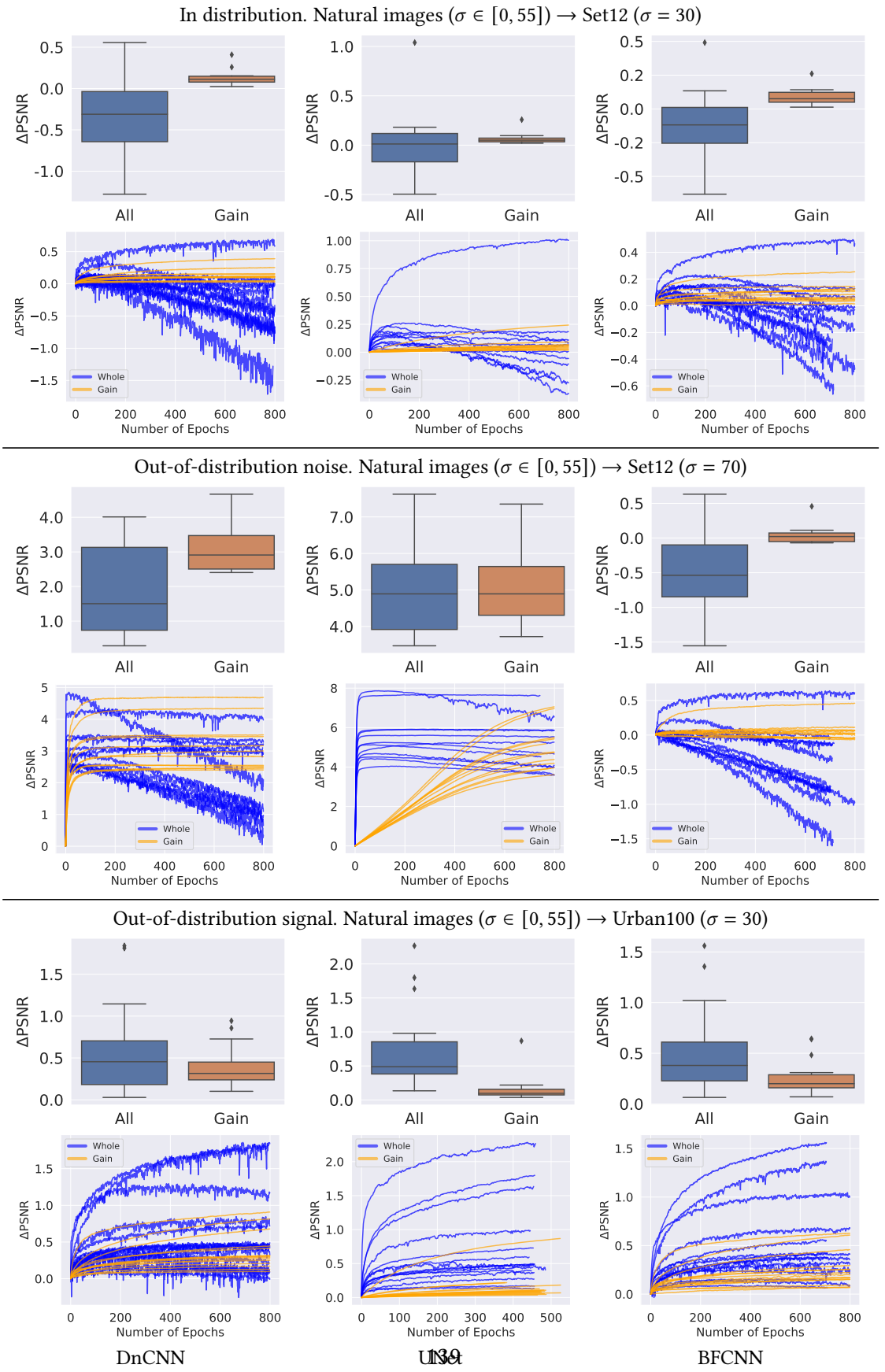


Figure C.5: GainTuning prevents overfitting. We compare training all parameters of the network (blue) and only the gain parameters (orange) during the adaptation process. All architectures are trained using the SURE cost function.

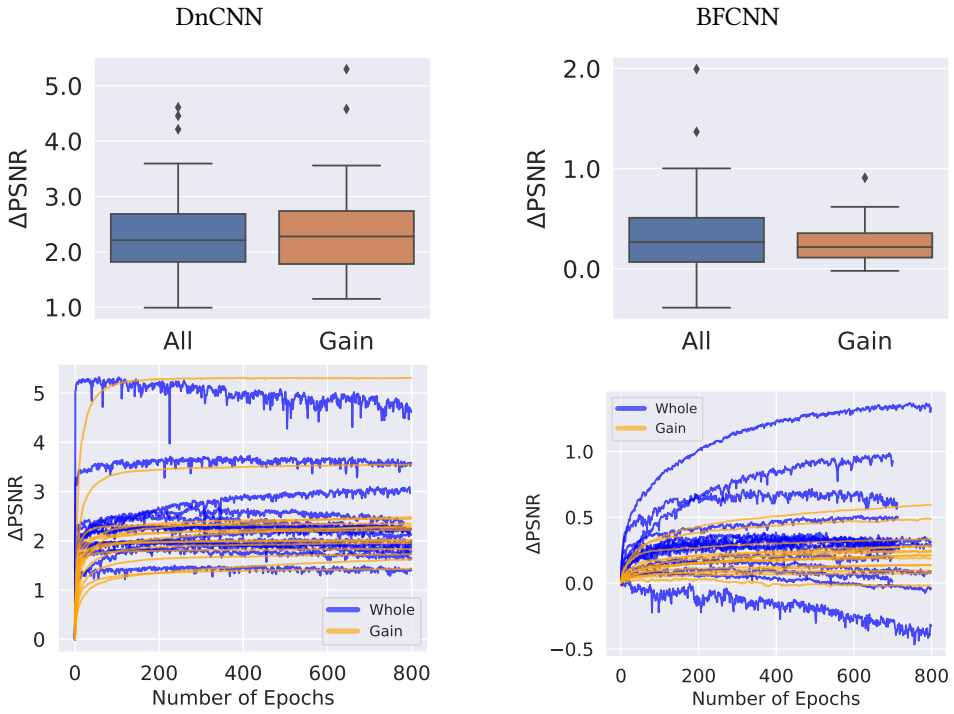


Figure C.6: Out-of-distribution noise and signal. We compare training all parameters of the network (blue), and only the gain parameters (orange) during the adaptation process. The CNN is pre-trained on generic natural images corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$. We apply GainTuning to adapt it to images of urban scenes (high self-similarity, hence different signal characteristics from natural images) corrupted with $\sigma = 70$ (which is outside the training range of noise). All architectures are trained using the SURE cost function.

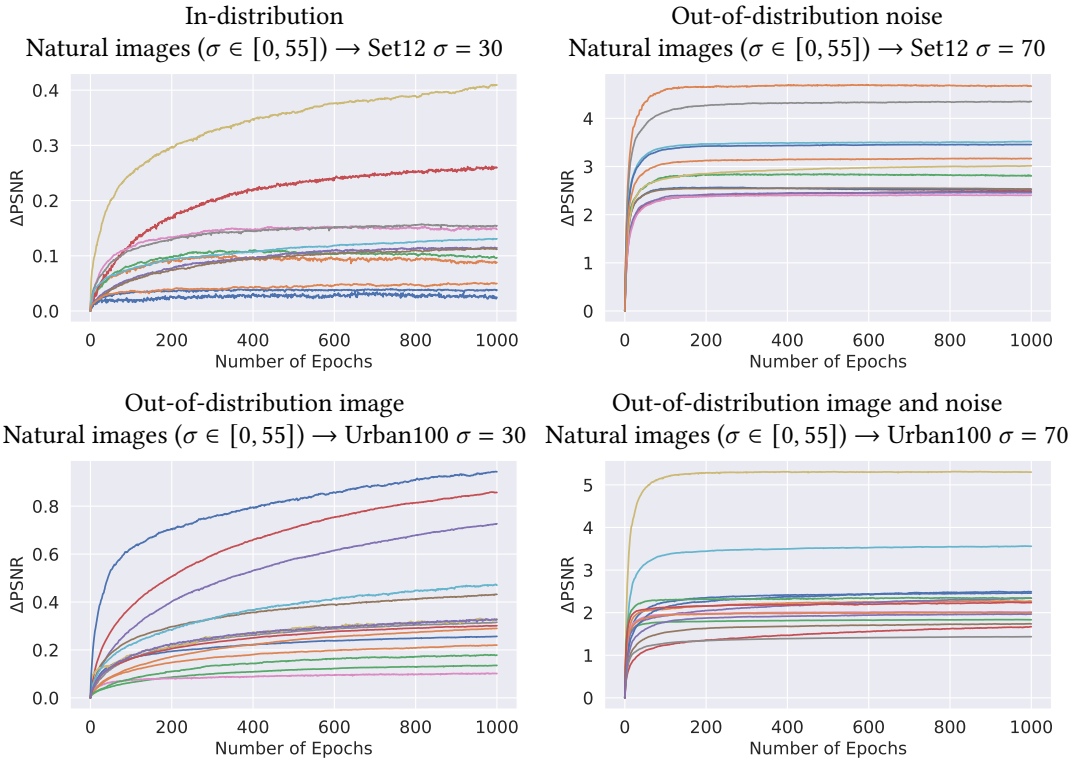


Figure C.7: GainTuning does not require early stopping. We plot the improvement in performance achieved by GainTuning with the number of iterations. Each iteration step is a pass through 10000 random 50×50 patch extracted from the image. The performance achieved by optimizing only the gain parameters remains constant or monotonically increases with iteration, while training all parameters often overfits (see Figure C.5)

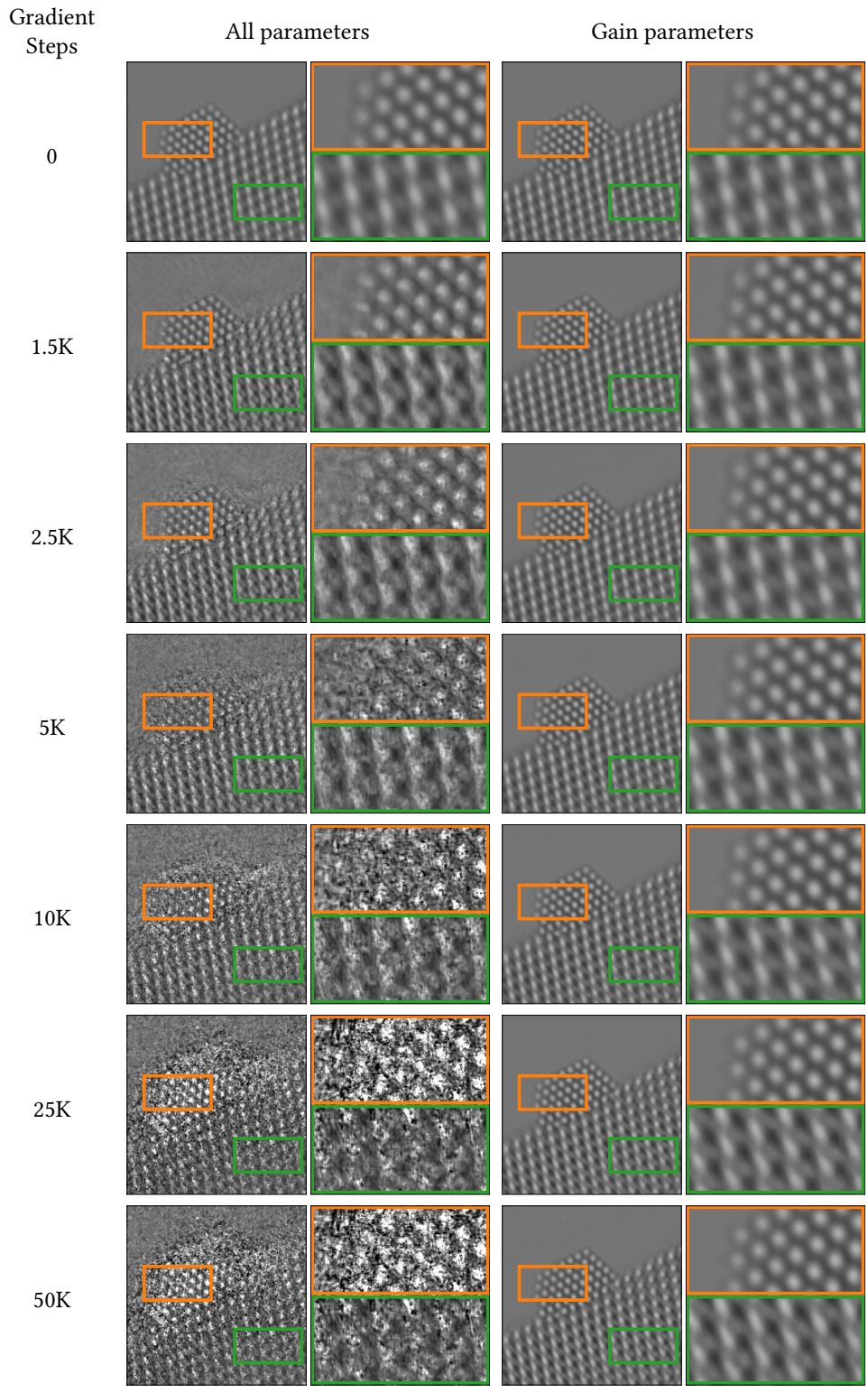


Figure C.8: GainTuning prevents overfitting in TEM data. We compare training all parameters and only the gain parameters while adapting a CNN pre-trained on simulated TEM data to real TEM data. Training all parameters clearly overfits to the noisy image. Each gradient step is updated over two random patches of size 400×400.

	Fine-tuning						
	All params	Last n layers					Only gains
		$n = 10$	$n = 4$	$n = 3$	$n = 2$	$n = 1$	
Num. params (% of total params)	668,225 (100%)	334,081 (49.95%)	111,745 (16.72%)	74,689 (11.18%)	37,633 (5.63%)	577 (0.09%)	1,152 (0.17%)
in-distribution	-0.33	0.09	0.05	0.04	0.04	0.06	**0.14**
out-of-distr. noise	1.92	1.92	2.05	2.06	2.10	2.13	**3.11**
out-of-distr. signal	-4.48	0.92	1.12	1.06	0.93	0.83	**1.45**

Table C.1: GainTuning vs selectively fine-tuning last few layers. We compared GainTuning to selectively fine-tuning last n layers for a DnCNN with $n = 20$ layers. Table entries indicate the change in performance (i.e., the performance in PSNR after fine-tuning minus the PSNR of the pre-trained network - larger positive values are better). Across different tasks, GainTuning outperformed fine-tuning last layers by a significant margin. The in-indistribution and out-of-distribution signal consists of adapting a DnCNN trained on natural images with $\sigma \in [0, 55]$ for natural images (Set12) with $\sigma = 30$ and $\sigma = 70$ respectively. We adapted a CNN trained on piecewise constant images with $\sigma \in [0, 55]$ to natural images (Set12) with $\sigma = 30$ for out-of-distribution signal experiments.

effectively to this out-of-distribution test set. GainTuning achieves comparable performance to a network trained with supervision on a large range of noise levels ($\sigma \in [0, 100]$), and a bias-free model which is explicitly designed to generalize to noise levels outside the training range. GainTuning also outperforms LIDIA [133] (a specialized architecture and adaptation procedure). and Self2Self [110] (a method trained exclusively on the test image).

C.5.3 Out-of-distribution image

Different Architectures. We summarize the results using DnCNN in Table 4.4 in the main paper. Figures C.5 show that the UNet and BFCNN architectures are also able to generalize to test data with different characteristics from the training data when adapted using GainTuning .

Different Loss Functions. We provide the results of evaluating the DnCNN architecture with different cost functions in Table C.6.

Model	σ	Set12		Set68	
		Pre-trained	GainTuning	Pre-trained	GainTuning
BFCNN	30	29.52	29.61	28.36	28.45

Table C.2: Results for BFCNN. Results for BFCNN [96] architecture trained on BSD400 dataset corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$. Results for other architectures are provided in Section 4.5.1.

Test set	σ	Trained on $\sigma \in [0, 55]$		Baselines				
				Bias Free Model [96]	Trained on $\sigma \in [0, 100]$	LIDIA [133]		S2S [110]
		Pre-trained	GainTuning			Pre-trained	Adapted	
Set12	70	22.45	25.54	25.59	25.50	23.69	25.01	24.61
	80	18.48	24.57	24.94	24.88	22.12	24.17	23.64
BSD68	70	22.15	24.89	24.87	24.88	23.28	24.57	24.29
	80	18.72	24.14	24.38	24.36	21.87	23.97	23.65

Table C.3: GainTuning for out-of-distribution noise. We evaluate a DnCNN trained on generic natural images for $\sigma \in [0, 55]$ on a test set of generic natural images corrupted with $\sigma = \{70, 80\}$, which is outside the training range of the network. GainTuning is able is generalize effectively to this out-of-distribution test set. GainTuning achieves comparable performance to a network trained with supervision on a large range of noise levels ($\sigma \in [0, 100]$) an bias-free models which is an architecture explicitly designed to generalize to noise levels outside the training range. GainTuning also outperforms LIDIA [133], a specialized architecture and adaptation procedure, and Self2Self [110], a method trained exclusively on the test image.

Comparison to baselines. Results of comparison to LIDIA [133], a specialized architecture to perform adaptation, and Self2Self [110] a method trained exclusively on the test image is summarized in Table C.5. While GainTuning outperforms LIDIA, it does not match the performance of Self2Self (see Section 4.7 for a discussion on this).

C.5.4 Out-of-distribution noise and image

We evaluated the ability of GainTuning to adapt to test images which have different characteristics from those in the training set, and are additionally corrupted with a noise distribution that is different from the noise in the training set. Figure C.6 shows that GainTuning is successful

(a) ζ	CNN trained on Gauss. $\sigma \in [0, 55]$		(d) Bias-free CNN trained on Gauss. $\sigma \in [0, 55]$	Improvement after GainTuning	
	(b) Pre-trained	(c) GainTuning		(e) Maximum	(f) Minimum
1	17.58	21.07	17.91	4.79	2.25
0.5	20.19	22.50	20.12	3.43	1.11
0.1	25.28	25.99	24.88	1.16	0.34

Table C.4: CNN trained on Gaussian noise generalizes to Poisson noise. Results on applying GainTuning to a CNN pre-trained on additive Gaussian noise (which has spatially uniform variance) to test data corrupted by Poisson noise (where the variance depends on the underlying pixel values and is hence spatially variant). We evaluate on Poisson noise with three different scaling ζ values (a), where a larger value of ζ implies that the image is more noisy (if x is a clean image, the noisy image y is sampled from $\zeta\text{Pois}(x/\zeta)$ where $\text{Pois}(\lambda)$ is the PMF of Poisson distribution with parameter λ). Applying GainTuning on the CNN improves its performance (b) by a significant margin (c). GainTuning on the pre-trained CNN also outpeforms its bias-free counterpart (d), which is designed to generalize well to Gaussian noise outside the training range. The maximum improvement in PSNR (e) obtained by applying GainTuning to the pre-trained CNN (b) is substantial, and the minimum improvement in PSNR (f) is non-trivial. The CNN used here is [47] and was pre-trained on BSD400 dataset. GainTuning was performed to adapt to Set12 with Poisson noise.

in this setting. The CNN was pre-trained on natural images corrupted with Gaussian white noise of standard deviation $\sigma \in [0, 55]$. We used GainTuning to adapt this CNN to a test set of images taken in urban setting (see Section C.1 for a discussion on how it is different from natural images), corrupted with Gaussian noise of standard deviation $\sigma = 70$ (which is outside the training range of $[0, 55]$).

C.5.5 Different loss functions

GainTuning can be used in conjunction with any unsupervised denoising cost function. We explore three different choices - SURE, noise resampling, and blind-spot cost functions (see Section 4.4), and summarize our finding in Table C.6.

SURE loss outperforms other choices in most experiments. Noise resampling has comparable performance to SURE when the test data is in-distribution, or when it is corrupted with out-of-distribution noise. However, noise resampling generally under-performs SURE when the test images have different features from the training images. A possible explanation for this is that

	Training Data	Test Data	DnCNN [150]		Baselines		
					LIDIA [133]		S2S [110]
			Pre-trained	GainTuning	Pre-trained	Adapted	
(a)	Piecewise constant	Natural images	27.31	28.60	-	-	29.21
(b)	Natural images	Urban images	28.35	28.79	28.54	28.71	29.08
(c)	Natural images	Scanned documents	30.02	30.73	30.05	30.23	30.86

Table C.5: GainTuning for out-of-distribution images. GainTuning generalizes robustly when the test image has different characteristics than the training data. We demonstrate this through three different experiments. (a) GainTuning provides an average of 1.3 dB in performance while adapting a CNN trained on simulated piecewise constant dataset to natural images. This controlled setting demonstrates the capability of GainTuning to adapt from a simple simulated training set to a significantly more complex real dataset. (b) GainTuning provides an average of 0.45 dB improvement in performance when a CNN trained on natural images is adapted to a dataset of images taken in urban settings. These images display a lot of repeating structure (see Section C.1) and hence has different characters than generic natural images. Similarly, (c) GainTuning provides an average of 0.70 dB improvement in performance when a CNN pre-trained on natural images is adapted to images of scanned documents. While GainTuning outperforms LIDIA [133], a specialized architecture designed for adapting, it does not match the performance of Self2Self (see Section 4.7 for a discussion on this). As noted in Section 4.5.3, we did not train LIDIA for (a).

noise resampling relies on the initial denoised image to fine-tune and, therefore, it may not be able to exploit features which are not present in the initial estimate. In contrast, the SURE cost function is computed on the noisy test image itself, thereby enabling it to adapt to features that the pre-trained network may be agnostic to.

Finally, adapting using blind-spot cost function often under-performs both SURE and noise resampling. The difference in performance is reduced at higher noise levels (see also Section 4.5.4 where we use blind-spot cost function for experiments with real TEM data with very high noise). The reason for this could be that at higher noise levels, the information contained in a single pixel becomes less relevant for computing the corresponding denoised estimate (in fact, the regularization penalty on "self pixel' for SURE cost function (Section 4.4) increases as the noise level increases). Therefore, the loss of performance incurred by the blind-spot cost function is diminished. At lower noise levels (particularly when the images are in-distribution), adapting using blind-spot cost function will force the pre-trained network to give up using the "self pixel", which results in a degraded performance. An alternative to adapting a generic pre-trained network using blind-spot architecture is to use a CNN that is architecturally constrained to include a blind-spot. In Table C.7, we show that adapting such a CNN using blind-spot loss improves the performance its performance. However, the overall performance of this architecture is in general lower than the networks which also use the "self pixel". We refer interested readers to Ref. [67, 68, 141] for approaches to incorporate the noisy pixel into the denoised estimate.

			GainTuning with		
		Pre-training	SURE	Noise resampling	Blind-spot (Noise2Self [8])
in distribution	Set12	29.52	29.62	**29.63**	29.50
	BSD68	28.39	**28.46**	28.40	28.36
out-of-distribution noise	Set12	18.48	**24.57**	24.11	22.93
	BSD68	18.72	**24.14**	23.65	22.50
out-of-distribution image	Piecewise constant → Natural images	27.31	**28.60**	28.29	27.39
	Natural images → Urban100	28.35	**28.79**	**28.79**	28.29
	Natural images → Scanned documents	30.02	**30.73**	30.57	29.23

Table C.6: Different loss functions for GainTuning. Comparison of the performance of GainTuning when used in conjunction with three different loss functions. SURE loss outperforms other choices in most experiments. Noise resampling has comparable performance to SURE when the test data is in-distribution, or when it is corrupted with out-of-distribution noise. However, noise resampling generally under-performs SURE when the test images have different features from the training images. This maybe because such features are absent from the initial denoised estimate (see Section 4.4 for a description of the different loss functions). Finally, optimizing using blind-spot cost functions often under-performs both SURE and noise resampling, but the difference in performance is reduced as the test noise increases (see also Section 4.5.4 where we use blind-spot cost function for experiments with real TEM data with very high noise). This may be because, at lower noise levels, the information contained in a pixel is often crucially important to compute its denoised estimate, and blind-spot cost function ignores this information (see Section 4.4). Here, we implemented blind-spot cost function through masking [8], see Table C.7 for results where the implemented blind-spot cost function as an architectural constraint [68].

	in-distribution		out-of-distribution image	
	Set12	BSD68	Urban100 (urban scenes)	IUPR (scanned documents)
Pre-trained	27.92	26.47	26.59	28.25
GainTuning	27.92	26.61	26.85	28.40

Table C.7: GainTuning using architecturally constrained blind-spot cost function. We perform GainTuning using blindspot network [68] which is architecturally constrained to estimate a denoised pixel exclusively from its neighbouring pixels (excluding the pixel itself). The network was pre-trained on generic natural images corrupted with Gaussian noise of standard deviation $\sigma \in [0, 55]$. Performing GainTuning on this always increases its performance, unlike GainTuning on a generic architecture trained with supervision and adapted using blind-spot loss implemented via masking. However, note the overall performance of this architecture is in general lower than the networks which also use the "self pixel". We refer interested readers to Ref. [67, 68, 141] for approaches to incorporate the information in noisy pixel back into the denoised output, thus potentially improving the performance. Our blind-spot architecture generalizes robustly to out-of-distribution noise (since it is bias-free [96]), and therefore we do not include an out-of-distribution noise comparison in this table.

D | Application to electron microscopy data

D.1 Data simulation

D.1.1 Simulation process

The simulated TEM image dataset was generated using the multi-slice image simulation method, as implemented in the Dr. Probe software package [7]. In the multi-slice approach, the modeled specimen is sectioned into many thin slices (here they are 0.167 Angstroms thick), and quantum mechanical calculations are performed to simulate the incident electron wave function propagating through and interacting with each slice of the material [65]. The resultant wave function exiting the last slice is then convolved with a point spread function that emulates the effect of imaging it in the electron microscope. All of the image simulations were performed using an accelerating voltage of 300 kV with a beam convergence angle of 0.2 mrad and a focal spread of 4 nm. The third-order spherical aberration coefficient (C_s) was set to be -13 μm. The fifth-order spherical aberration coefficient ($C5$) was set as 5 mm. All other aberrations (e.g., 2-fold and 3-fold astigmatism, coma, star aberration, etc.) were approximated to be negligible. The defocus ($C1$) was varied systematically between 0 nm and 20 nm, as discussed below. Image calculations were computed using a non-linear model including partial temporal coherence by explicit averaging

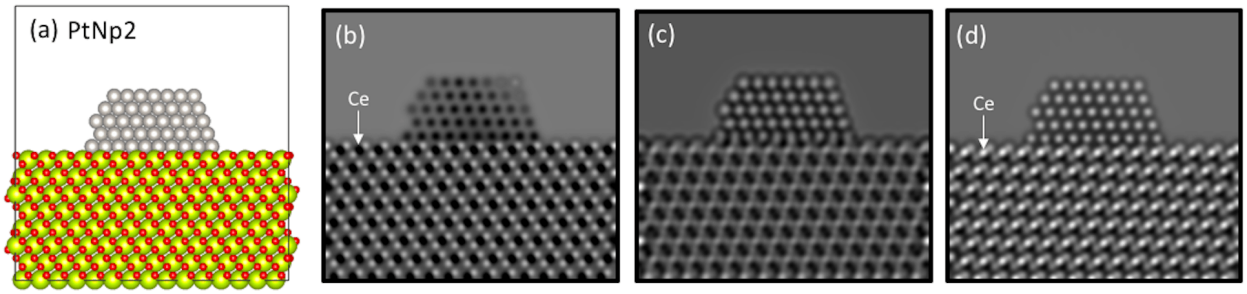


Figure D.1: Demonstration of contrast reversal with changes in defocus. (a) Image of the Pt/CeO_2 atomic structural model.(b) to (d) Simulated images under different electron-optical focusing conditions, emphasizing variations on the Ce and Pt column contrast. In (b), the image shows black contrast for both Ce and Pt columns. In (c), the Pt columns reverse contrast and now appear white, while Ce columns become challenging to discriminate. Finally, in (d) all of the atomic columns appear with white contrast.

and partial spatial coherence, which is treated by a quasi-coherent approach with a dampening envelope applied to the wave function. An isotropic vibration envelope of 50 pm was applied during the image calculation. Images were simulated with 1024 x 1024 pixels and then later binned to desired sizes to match the pixel size of the experimentally acquired image series. Finally, to equate the intensity range of the simulated images with those acquired experimentally, the intensities of the simulated images were scaled by a factor which equalized the vacuum intensity in a single simulation to the average intensity measured over a large area of the vacuum in a single 0.025 second experimental frame (i.e., 0.45 counts per pixel in the vacuum region).

D.1.2 Experimental parameters

In phase-contrast TEM imaging (the technique employed here), multiple electron-optical and specimen parameters can give rise to complex, non-linear modulations of the image contrast. These parameters can include the objective lens defocus, the specimen thickness, the orientation of the specimen, and its crystallographic shape/structure. Due to the complex image formation mechanisms, atomic columns of the same material imaged may appear black or white (or somewhere in between, i.e., intermediate) depending on the exact combination of these various factors.

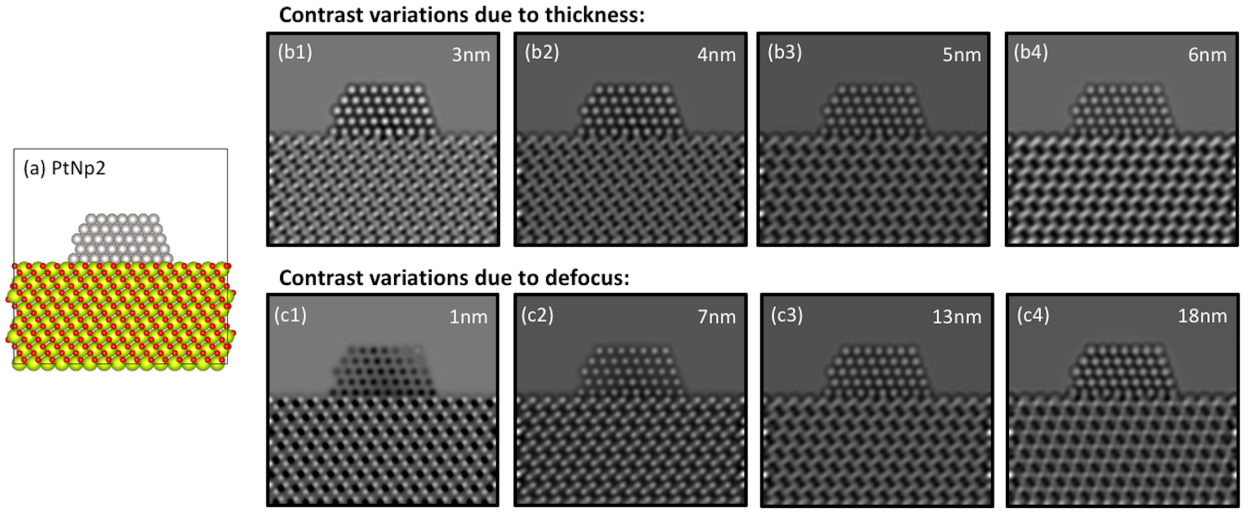


Figure D.2: Image contrast variations due to thickness and defocus. (a) Image of the Pt/CeO_2 atomic structural model. (b1) to (b4) Simulated images at a defocus value of 13 nm, where the variations of the contrast are due to the thickness of the model, increased from 3 nm to 6nm. (c1) to (c4) Contrast variations on simulated images due to defocus: the thickness of the model has been kept constant at 5 nm and the defocus has been tuned to 1 nm, 7 nm, 13 nm, and 18 nm.

Examples of the type of contrast reversal that may occur for a static structure imaged at constant thickness and tilt are given in Figure D.1. Additionally, images showing the type of contrast variations that may occur when the support thickness is changed, and how these compare to those which arise from changes in defocus are given in Figure D.2.

Here, we systematically varied these parameters to generate a large number of cases (approximately 18,000), corresponding to potential combinations that may arise during a real experiment (see Figure D.3). First, around 100 atomic-scale structural models of CeO_2-supported Pt nanoparticles were generated. Each model represents Pt nanoparticles of various size, shape and atomic structure (e.g., small, medium, or large size, with either faceted or defected surfaces, or some combination of both), supported on CeO_2, which itself may present either a faceted surface or one characterized by surface defects. Secondly, the thickness of the CeO_2 support was varied from 3 nm to 6 nm along 1 nm increments. One aspect to note is that the thickness variation is not equally applied to each of the aforementioned models. Third, each resultant model was tilted from 0° to 4° about the x and y axes independently in increments of 1°. Thus, variations from 0°

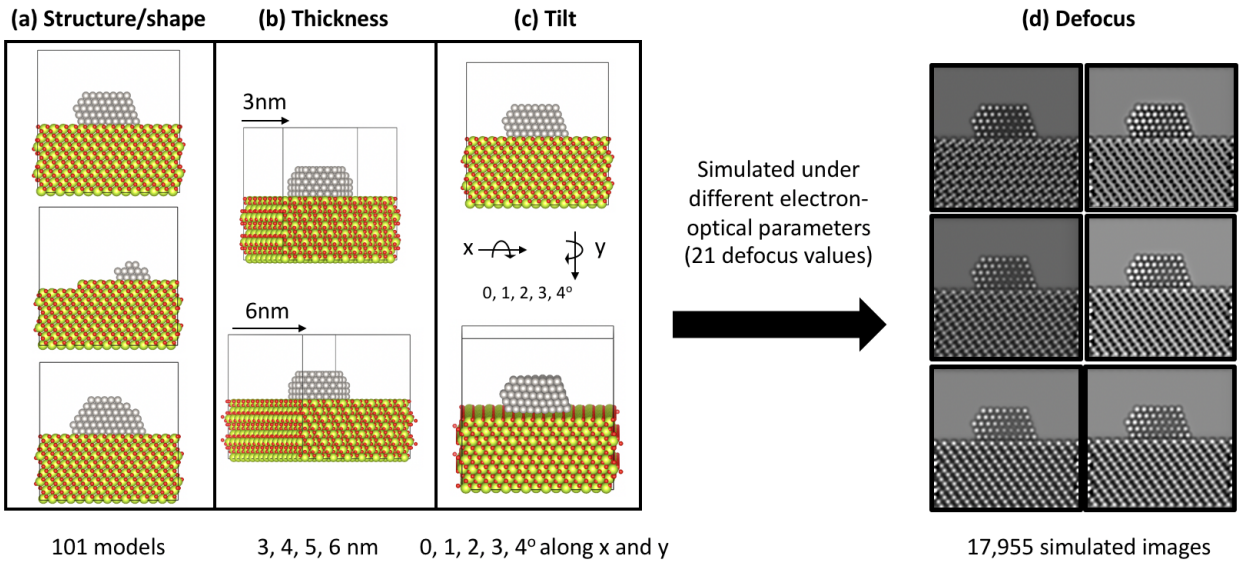


Figure D.3: Summary of parameters considered during the modelling and image simulation processes. Subset of Pt/CeO_2 atomic models presenting variations on the (a) structure and shape of the nanoparticle and the support, (b) the thickness of the CeO_2 slab and (c) the tilt of the atomic models. Color code for the models matches Pt, Ce and O with grey, yellow and red atoms respectively. (d) Simulated images under different defocus values.

in x and 0° in y, to 4° in x and 0° in y, or 0° in x and 4° in y were considered. The final parameter systematically varied in the simulated image dataset was the electron optical defocus. Every model containing a unique shape/structure, thickness, and tilt (855 total) was imaged under a range of defocus values which often arise experimentally. Namely, the defocus was varied from 0 nm to 20 nm, along increments of 1 nm. Considering all combinations of the varied parameters, a total of 17,955 simulated images were generated for training and testing the neural network.

D.1.3 Description of nanoparticle structures

The 3D atomic structural models utilized in this work consist of faceted Pt nanoparticles that oriented in a [110] zone axis and that are supported on a CeO_2 (111) surface which is itself oriented in the [110] zone axis. This crystallographic configuration corresponds to that which is often observed experimentally and is thus the focus of the current work. All of the models have been constructed with the freely available Rhodius software [12]. Each model consists of a

supercell having x and y dimensions of 5 nm x 5 nm. As discussed above, the support thickness was systematically varied for each model, and so the supercell's z dimension varies between 3 nm and 6 nm, depending on the thickness of the particular model.

While imaging these materials systems, experimentalists often aspire to visualize atomic-level structural rearrangements that can occur at the surfaces of the supported nanoparticles. Additionally, there are many millions of nanoparticles on a typical TEM sample, and the specific atomic-scale structural features comprising the surfaces of those imaged during an experiment may vary slightly from nanoparticle to nanoparticle. In order to encompass such complexity in the training dataset, a variety of Pt nanoparticles of multiple sizes/shape and surface defect character were incorporated into the 3D models. For example, four such models of CeO_2-supported Pt nanoparticles having various size and shape are shown in parts (a) to (d) of Figure D.4. The multi-slice TEM image simulations generated from the models are shown below each for two different conditions, namely in parts (a1) to (d1), images are shown for a case in which the support is 3 nm thick, the defocus is 9 nm, and the tilt is 0° in x and 0° in y; in parts (a2) to (d2), images are shown for the case in which the support is 5 nm thick, the defocus is 6 nm, and the tilt is 4° in x and 0° in y. Furthermore, the surface character of the Pt nanoparticles was varied by altering the defect structure at different surface sites. A few examples are depicted in Figure D.5. Here, in part (a), a CeO_2-supported Pt nanoparticle with faceted surfaces is shown; directly beneath it in (a1) is an image simulated under conditions in which the support is 3 nm thick, the defocus is 9nm, and there is no tilt. The arrowed sites designate locations on the Pt surface that have been subsequently altered. In part (b), the surface has been modified by removing a full atomic column from the arrowed location. In part (c), the occupancy of the arrowed corner site has been reduced by half. And in part (d), the occupancy of the arrowed corner site has been further reduced to a single atom. Parts (b1) - (d1) show the images simulated from these respective structures under the same imaging condition. Note that the surface sites altered in the structure correspond to high-energy sites (e.g., corners and edges) which are more likely to dynamically rearrange or

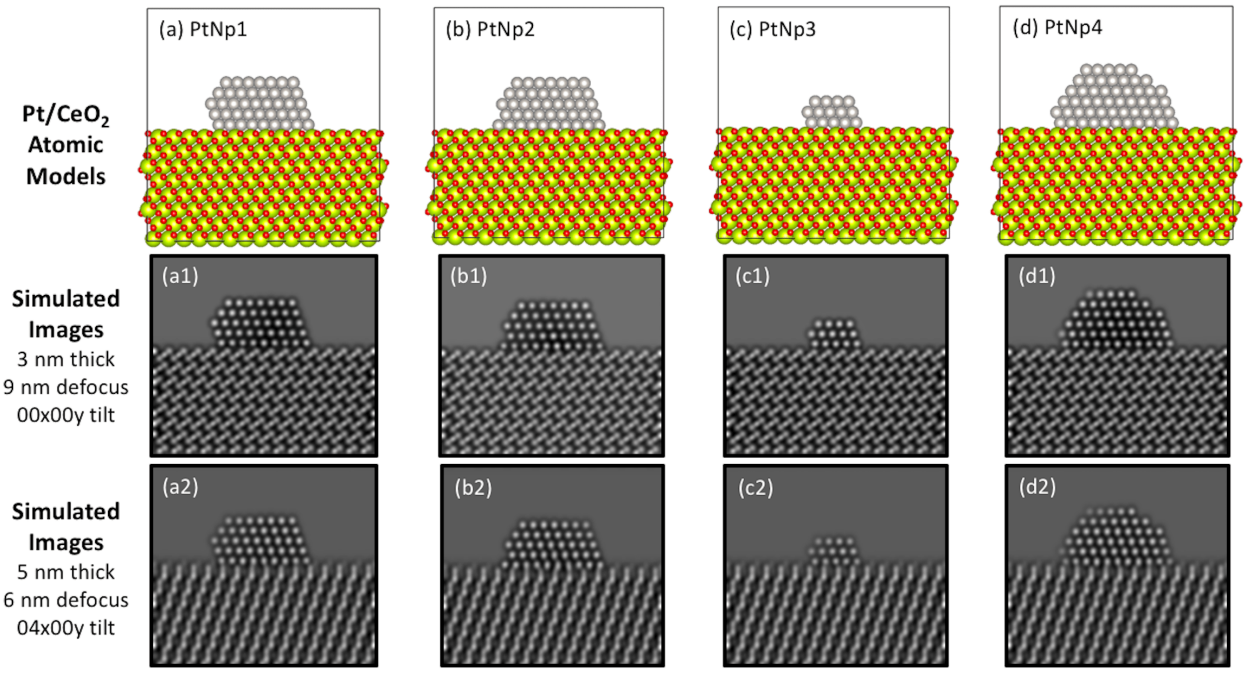


Figure D.4: Variations in the structure/size of the supported Pt nanoparticle.(a) to (d) Atomic models of Pt nanoparticles (grey atoms) with different shapes and supported over a CeO_2 slab (yellow and red atoms respectively). (a1) to (d1) Simulated images depicting the described atomic models, considering a thickness of 3 nm, 9 nm of defocus and no tilt on the model, whereas (a2) to (d2) illustrate the same model under different conditions: 5 nm thickness, 6 nm defocus and 4 degrees tilted along x axis. All the simulated images present a Cs value of -13 μm.

show variation than, say, a low-energy terrace site located in the middle of the surface.

D.2 Proposed Architecture: UNet with large field of view

We propose to use a modified version of UNet [116] with $n = 4, 6$ scales to achieve a large field of view. The network consisting of n `down-blocks` and n `up-blocks`. A `down-block` consists of a max-pooling layer, which reduces the spatial-dimension by half, followed by a `conv-block`. Similarly, an `up-block` consists of bilinear upsampling, which enlarges the size of the feature-map by a factor of two, followed by `conv-block`. A `conv-block` consists of conv-BN-ReLU-conv-BN-ReLU, where conv represents a convolutional layer and BN stands for batch normalization [54]. In our final model, we use 128 channels in each layer of `conv-block` and $n = 6$ scales.

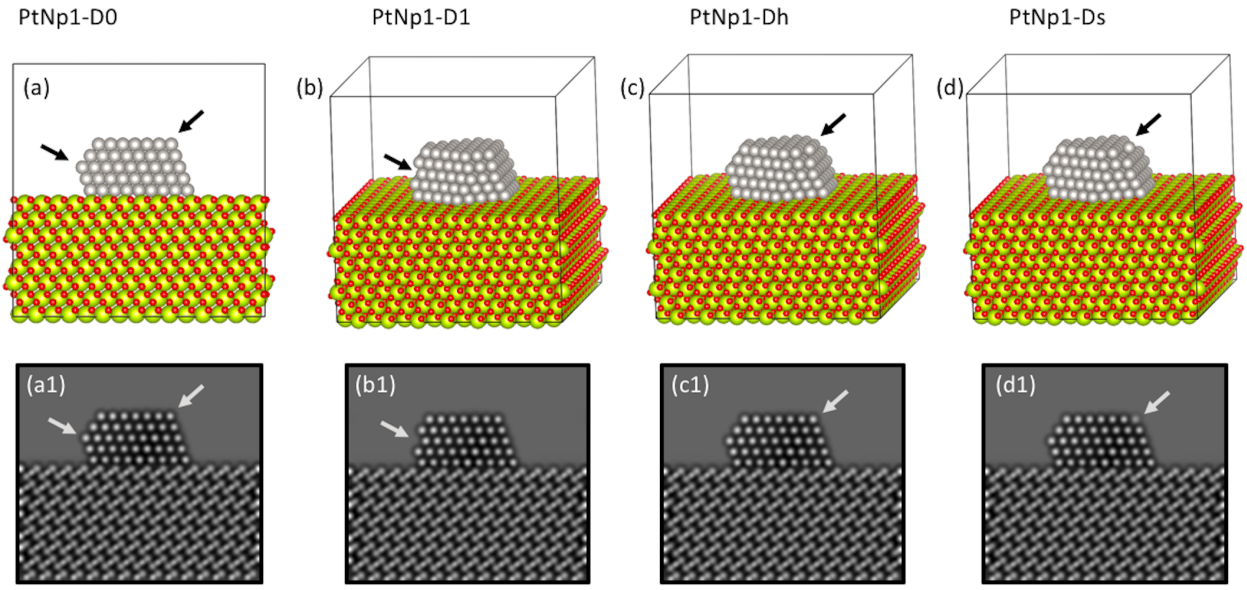


Figure D.5: Variations in the defects of the Pt surface structure. (a) Atomic model of a CeO_2-supported Pt nanoparticle without any defects. The surface has been modified by (b) removing a full atomic column, (c) removing half of the occupancy and (d) keeping a single atom. Black arrows point the sites where these defects are taking place. Models (b), (c) and (d) have been slightly tilted to observe these modifications. (a1) to (d1) Simulated images of the presented atomic columns considering a 3nm thickness, 9 nm defocus and no tilt.

D.3 Additional Results

In this section we include the following additional results:

- Figure D.6 show an additional example of simulated image denoised using the proposed approach and the methods described in Section 5.5.2.
- Figure D.7 visualizes nine nanoparticle structures used for creating the simulated dataset with surface defects described in Section 5.5.3.
- Figure D.8 show an additional example of real images denoised using the proposed approach and the methods described in Section 5.5.2.

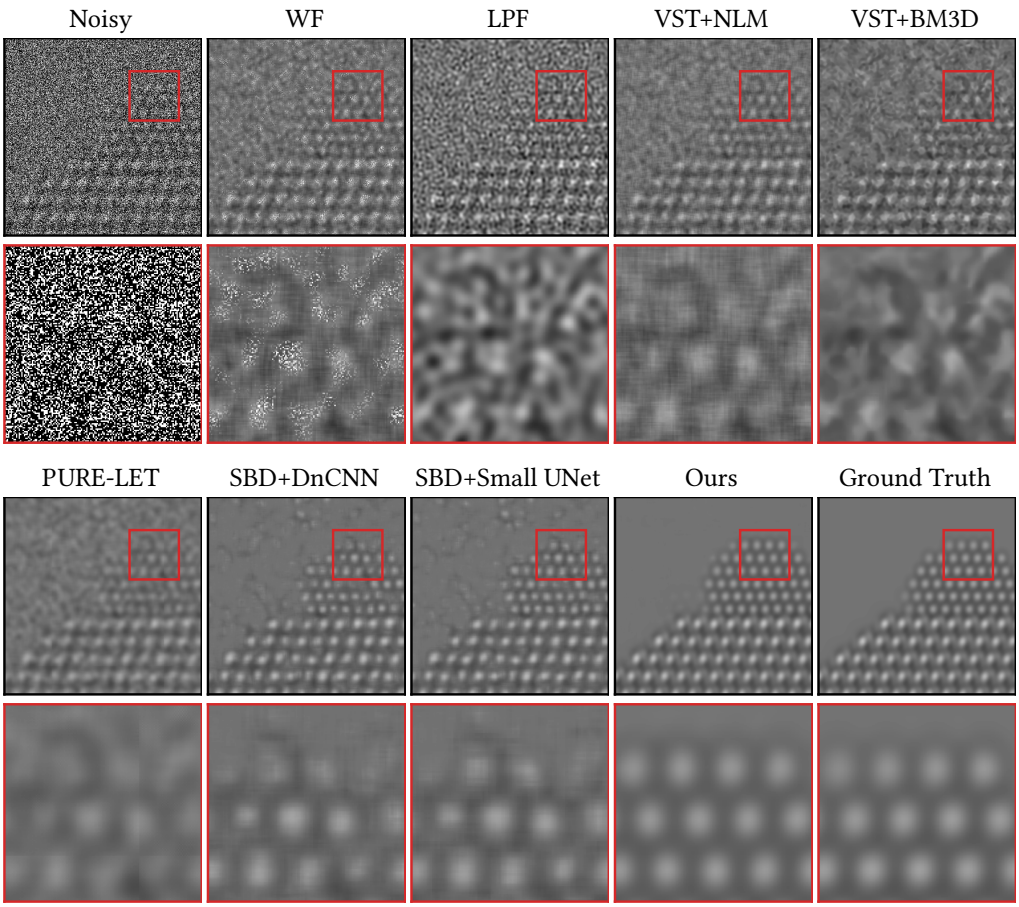


Figure D.6: Denoising results for simulated data. An additional example comparing SBD and the baseline methods described in Section 5.5.2. The second row zooms in on the region in red box. Our proposed approach produces images of much higher quality than the other approaches, and is able to accurately recover the atomic structure of the nanoparticle. For example, the vacuum region in images denoised by several of the baselines contain visible artefacts, including missing atoms.

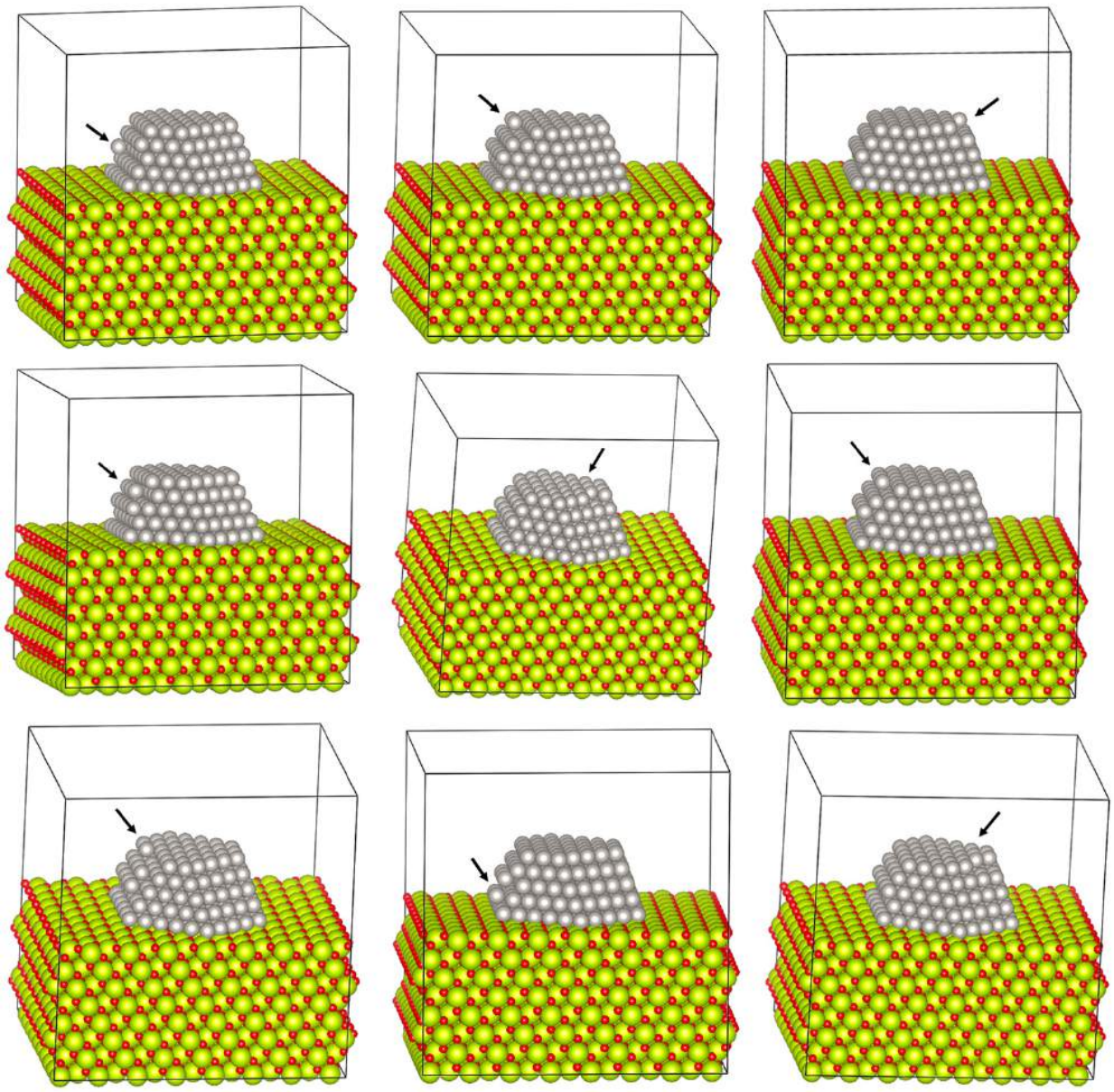


Figure D.7: Example of nanoparticle structures used for surface dataset in Section 5.5.3 A Subset of Pt/CeO_2 structual models with atomic-level surface defects like removal of an atom from a column, removal of two atoms, removal of all but one atom and the addition of a new atom at a site. See Section 5.5.3 for more details.

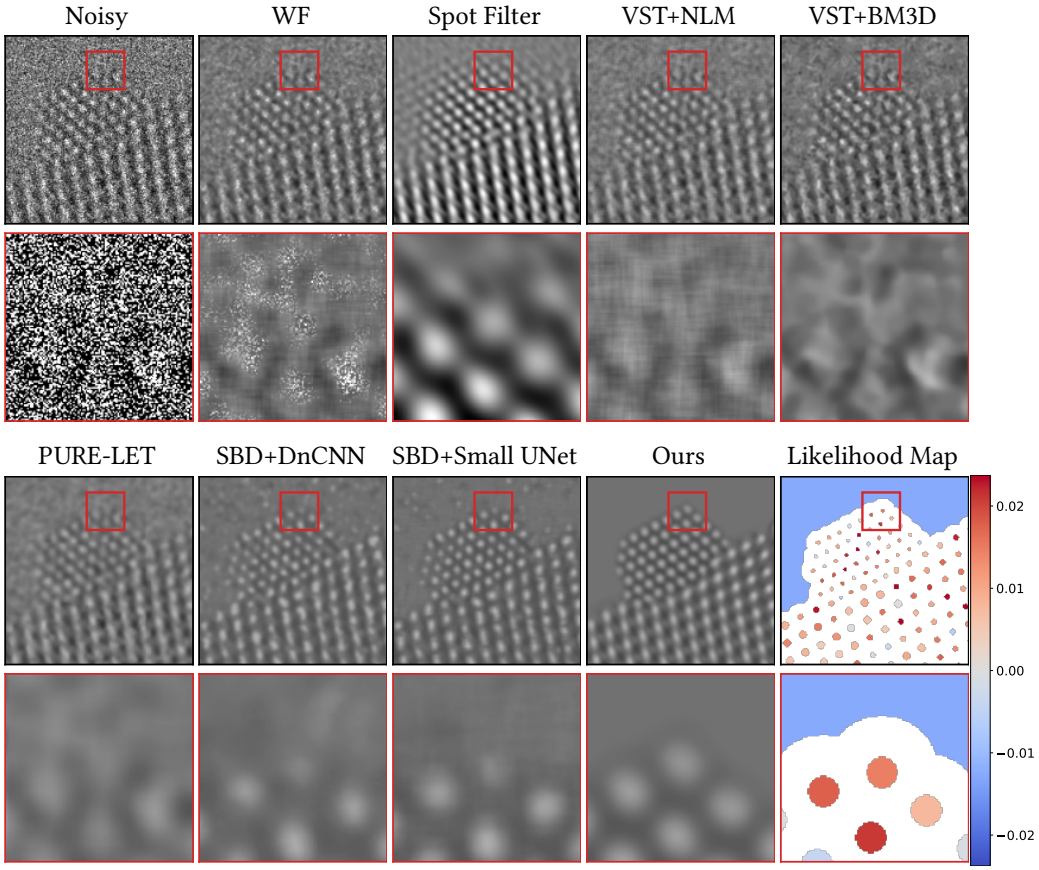


Figure D.8: Denoising results for real data. An additional example comparing SBD and the baseline methods described in Section 5.5.2 when applied on the real data described in Section 5.4.1. The second row zooms in on the region in red box. In contrast to the other methods, SBD combined with the proposed architecture is able to precisely recover the structure of the nanoparticle and has very few artefacts, particularly in the vacuum region. The likelihood map quantifies the agreement between recovered structures in the denoised images, such as atomic columns and the vacuum, and the observed data (see Section 5.3.3 for more details).

Bibliography

[1] Eirikur Agustsson and Radu Timofte. "NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.

[2] Pablo Arias and Jean-Michel Morel. "Video denoising via empirical Bayesian estimation of space-time patches". In: *Journal of Mathematical Imaging and Vision* 60.1 (2018), pp. 70–93.

[3] Xiao-Chen Bai, Greg McMullan, and Sjors H.W Scheres. "How cryo-EM is revolutionizing structural biology". In: *Trends in Biochemical Sciences* 40.1 (2015), pp. 49–57. ISSN: 0968-0004. DOI: https://doi.org/10.1016/j.tibs.2014.10.005.

[4] J Ballé, V Laparra, and E P Simoncelli. "Density modeling of images using a generalized normalization transformation". In: *Int'l Conf on Learning Representations (ICLR)*. Available at http://arxiv.org/abs/1511.06281. San Juan, Puerto Rico, May 2016.

[5] J Ballé, V Laparra, and E P Simoncelli. "End-to-end optimized image compression". In: *Int'l Conf on Learning Representations (ICLR)*. Available at http://arxiv.org/abs/1611.01704. Toulon, France, Apr. 2017.

[6] Norman Charles Barford. "Experimental measurements: precision, error and truth". In: (1967).

[7] Juri Barthel. “Dr. Probe: A software for high-resolution STEM image simulation”. In: *Ultramicroscopy* 193 (2018), pp. 1–11.

[8] Joshua Batson and Loic Royer. “Noise2Self: Blind Denoising by Self-Supervision”. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 524–533.

[9] Simon Beckouche, Jean-Luc Starck, and Jalal Fadili. “Astronomical image denoising using dictionary learning”. In: *Astronomy & Astrophysics* 556 (2013), A132.

[10] Yoav Benjamini. “Simultaneous and selective inference: Current successes and future challenges”. In: *Biometrical Journal* 52.6 (2010), pp. 708–721.

[11] Yoav Benjamini and Yosef Hochberg. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1 (1995), pp. 289–300. ISSN: 00359246. DOI: 10.2307/2346101.

[12] S Bernal et al. “The interpretation of HREM images of supported metal catalysts using image simulation: profile view images”. In: *Ultramicroscopy* 72.3-4 (1998), pp. 135–164.

[13] Antoni Buades, Bartomeu Coll, and J-M Morel. “A non-local algorithm for image denoising”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*. Vol. 2. IEEE. 2005, pp. 60–65.

[14] Antoni Buades, Jose-Luis Lisani, and Marko Miladinović. “Patch-based video denoising with optical flow estimation”. In: *IEEE Transactions on Image Processing* 25.6 (2016), pp. 2573–2586.

[15] Tim-Oliver Buchholz et al. “Cryo-care: content-aware image restoration for cryo-transmission electron microscopy data”. In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 502–506.

[16] Tim-Oliver Buchholz et al. “DenoiSeg: joint denoising and segmentation”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 324–337.

[17] Syed Saqib Bukhari, Faisal Shafait, and Thomas M Breuel. “The IUPR dataset of camera-captured document images”. In: *International Workshop on Camera-Based Document Analysis and Recognition*. Springer. 2011, pp. 164–171.

[18] Matteo Carandini and David J Heeger. “Normalization as a canonical neural computation”. In: *Nature Reviews Neuroscience* 13.1 (2012), pp. 51–62.

[19] S Grace Chang, Bin Yu, and Martin Vetterli. “Adaptive wavelet thresholding for image denoising and compression”. In: *IEEE Trans. Image Processing* 9.9 (2000), pp. 1532–1546.

[20] Ting Chen et al. “On self modulation for generative adversarial networks”. In: *arXiv preprint arXiv:1810.01365* (2018).

[21] Yunjin Chen and Thomas Pock. “Trainable nonlinear reaction diffusion: A flexible framework for fast and effective image restoration”. In: *IEEE Trans. Patt. Analysis and Machine Intelligence* 39.6 (2017), pp. 1256–1272.

[22] Sungjoon Choi et al. “Fast, Trainable, Multiscale Denoising”. In: *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2018, pp. 963–967.

[23] Michele Claus and Jan van Gemert. “ViDeNN: Deep Blind Video Denoising”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop*. 2019, pp. 1843–1852.

[24] Peter A. Crozier et al. “Dynamic restructuring during processing: approaches to higher temporal resolution”. In: *Microscopy and Microanalysis* 25.S2 (2019), pp. 1464–1465.

[25] Kostadin Dabov et al. “Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering”. In: *IEEE Transactions on Image Processing* (2017), pp. 2080–2095.

[26] Kostadin Dabov et al. “Image denoising with block-matching and 3D filtering”. In: *Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*. Vol. 6064. International Society for Optics and Photonics. 2006, p. 606414.

[27] Axel Davy et al. “A non-local CNN for video denoising”. In: *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2019, pp. 2409–2413.

[28] Harm De Vries et al. “Modulating early visual processing by language”. In: *arXiv preprint arXiv:1707.00683* (2017).

[29] Valery Dewil et al. “Self-Supervised Training for Blind Multi-Frame Video Denoising”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 2724–2734.

[30] Jeff Donahue et al. “Decaf: A deep convolutional activation feature for generic visual recognition”. In: *International conference on machine learning*. PMLR. 2014, pp. 647–655.

[31] D Donoho and I Johnstone. “Adapting to Unknown Smoothness via Wavelet Shrinkage”. In: *J American Stat Assoc* 90.432 (Dec. 1995).

[32] Jeffrey M Ede and Richard Beanland. “Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder”. In: *Ultramicroscopy* 202 (2019), pp. 18–25.

[33] Jeffrey Mark Ede. “Deep Learning in Electron Microscopy”. In: *Machine Learning: Science and Technology* (2020).

[34] Thibaud Ehret et al. “Model-Blind Video Denoising via Frame-To-Frame Training”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11361–11370.

[35] Michael Elad and Michal Aharon. “Image denoising via sparse and redundant representations over learned dictionaries”. In: *IEEE Trans. on Image processing* 15.12 (2006), pp. 3736–3745.

[36] Michael Elad and Michal Aharon. “Image denoising via sparse and redundant representations over learned dictionaries”. In: *IEEE Transactions on Image processing* 15.12 (2006), pp. 3736–3745.

[37] Peter Ercius et al. “The 4D Camera – a 87 kHz Frame-rate Detector for Counted 4D-STEM Experiments”. In: *Microscopy and Microanalysis* (2020), pp. 1–3. DOI: 10.1017/S1431927620019753.

[38] A.R. Faruqi and G. McMullan. “Direct imaging detectors for electron microscopy”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 878 (2018). Radiation Imaging Techniques and Applications, pp. 180–190. ISSN: 0168-9002. DOI: https://doi.org/10.1016/j.nima.2017.07.037.

[39] Golnaz Ghiasi et al. “Exploring the structure of a real-time, arbitrary neural artistic stylization network”. In: *arXiv preprint arXiv:1705.06830* (2017).

[40] E Giannatou et al. “Deep learning denoising of SEM images towards noise-reduced LER measurements”. In: *Microelectronic Engineering* 216 (2019), p. 111051.

[41] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier neural networks”. In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[42] Kuang Gong et al. “PET image denoising using a deep neural network through fine tuning”. In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 3.2 (2018), pp. 153–161.

[43] Kuang Gong et al. “PET image denoising using a deep neural network through fine tuning”. In: *IEEE Transactions on Radiation and Plasma Medical Sciences* 3.2 (2018), pp. 153–161.

[44] Bichuan Guo et al. “Learning Model-Blind Temporal Denoisers without Ground Truths”. In: *arXiv preprint arXiv:2007.03241* (2020).

[45] Han Guo, Philippe Sautet, and Anastassia N. Alexandrova. "Reagent-Triggered Isomerization of Fluxional Cluster Catalyst via Dynamic Coupling". In: *The Journal of Physical Chemistry Letters* 11.8 (2020). PMID: 32227852, pp. 3089–3094. DOI: 10.1021/acs.jpclett.0c00548.

[46] Shi Guo et al. "Toward Convolutional Blind Denoising of Real Photographs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1712–1722.

[47] Jingwen He, Chao Dong, and Yu Qiao. "Modulating image restoration with continual levels via adaptive feature modification layers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11056–11064.

[48] Y Hel-Or and D Shaked. "A discriminative approach for wavelet denoising". In: *IEEE Trans. Image Processing* (2008).

[49] Y. Hel-Or and D. Shaked. "A Discriminative Approach for Wavelet Denoising". In: *IEEE Transactions on Image Processing* 17 (2008), pp. 443–457.

[50] David Honzátko et al. "Defect segmentation for multi-illumination quality control systems". In: *Machine Vision and Applications* 32.6 (2021), pp. 1–16.

[51] James P Horwath et al. "Understanding important features of deep learning models for segmentation of high-resolution transmission electron microscopy images". In: *npj Computational Materials* 6.1 (2020), pp. 1–9.

[52] Gao Huang et al. "Densely connected convolutional networks". In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708.

[53] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. "Single image super-resolution from transformed self-exemplars". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 5197–5206.

[54] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[55] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[56] Kevin Jarrett et al. “What is the best multi-stage architecture for object recognition?” In: *2009 IEEE 12th international conference on computer vision*. IEEE. 2009, pp. 2146–2153.

[57] Wen Jiang et al. “Applications of a bilateral denoising filter in biological electron microscopy”. In: *Journal of structural biology* 144.1-2 (2003), pp. 114–122.

[58] Aakash Kaku et al. “Be like water: Robustness to extraneous variables via adaptive feature normalization”. In: *arXiv preprint arXiv:2002.04019* (2020).

[59] Wesley Khademi et al. “Self-Supervised Poisson-Gaussian Denoising”. In: *arXiv preprint arXiv:2002.09558* (2020).

[60] Wesley Khademi et al. “Self-Supervised Poisson-Gaussian Denoising”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 2131–2139.

[61] Byeongjoon Kim et al. “A performance comparison of convolutional neural network-based image denoising methods: The effect of loss functions on low-dose CT images”. In: *Medical physics* 46.9 (2019), pp. 3906–3923.

[62] Sohyeong Kim et al. “The Vid3oC and IntVID Datasets for Video Super Resolution and Quality Mapping”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 3609–3616.

[63] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[64] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[65] Earl J Kirkland et al. “Image simulation in transmission electron microscopy”. In: *Cornell University, Ithaca* (2006).

[66] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. “Noise2Void - Learning Denoising From Single Noisy Images”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2124–2132.

[67] Alexander Krull, Tomas Vicar, and Florian Jug. “Probabilistic Noise2Void: Unsupervised Content-Aware Denoising”. In: *arXiv preprint arXiv:1906.00651* (2019).

[68] Samuli Laine et al. “High-Quality Self-Supervised Deep Image Denoising”. In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 6970–6980.

[69] Ethan L. Lawrence et al. “Approaches to Exploring Spatio-Temporal Surface Dynamics in Nanoparticles with In Situ Transmission Electron Microscopy”. In: *Microscopy and Microanalysis* 26.1 (2020), pp. 86–94. DOI: 10.1017/S1431927619015228.

[70] Chenyang Le et al. “Perceptually Optimized Deep High-Dynamic-Range Image Tone Mapping”. In: *arXiv preprint arXiv:2109.00180* (2021).

[71] Marc Lebrun, Antoni Buades, and Jean-Michel Morel. “A nonlocal Bayesian image denoising algorithm”. In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1665–1688.

[72] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436.

[73] Ann B Lee, David Mumford, and Jinggang Huang. “Occlusion models for natural images: A statistical study of a scale-invariant dead leaves model”. In: *International Journal of Computer Vision* 41.1 (2001), pp. 35–59.

[74] Jaakko Lehtinen et al. “Noise2Noise: Learning Image Restoration without Clean Data”. In: *Proceedings of the 35th International Conference on Machine Learning*. 2018, pp. 2965–2974.

[75] Barnaby D.A. Levin, Ethan L. Lawrence, and Peter A. Crozier. "Tracking the picoscale spatial motion of atomic columns during dynamic structural change". In: *Ultramicroscopy* 213 (2020), p. 112978. ISSN: 0304-3991. DOI: https://doi.org/10.1016/j.ultramic.2020.112978.

[76] Zhi Li et al. "Toward A Practical Perceptual Video Quality Metric". In: *Netflix Technology Blog* (2016).

[77] Jeff W Lichtman and José-Angel Conchello. "Fluorescence microscopy". In: *Nature methods* 2.12 (2005), pp. 910–919.

[78] Jae S Lim. "Two-dimensional signal and image processing". In: *ph* (1990).

[79] Tony Lindeberg. "Scale selection properties of generalized scale-space interest point detectors". In: *Journal of Mathematical Imaging and vision* 46.2 (2013), pp. 177–210.

[80] Ce Liu and William T Freeman. "A high-quality video denoising algorithm based on reliable motion estimation". In: *European conference on computer vision*. Springer. 2010, pp. 706–719.

[81] F. Luisier, T. Blu, and M. Unser. "A New SURE Approach to Image Denoising: Interscale Orthonormal Wavelet Thresholding". In: *IEEE Transactions on Image Processing* 16 (2007), pp. 593–606.

[82] Florian Luisier, Thierry Blu, and Michael Unser. "Image denoising in mixed Poisson–Gaussian noise". In: *IEEE Transactions on image processing* 20.3 (2010), pp. 696–708.

[83] Jacob Madsen et al. "A deep learning approach to identify local structures in atomic-resolution transmission electron microscopy images". In: *Advanced Theory and Simulations* 1.8 (2018), p. 1800037.

[84] Matteo Maggioni et al. “Video denoising, deblocking, and enhancement through separable 4-D nonlocal spatiotemporal transforms”. In: *IEEE Transactions on image processing* 21.9 (2012), pp. 3952–3966.

[85] Markku Makitalo and Alessandro Foi. “Optimal inversion of the generalized Anscombe transformation for Poisson-Gaussian noise”. In: *IEEE transactions on image processing* 22.1 (2012), pp. 91–103.

[86] Bryce Manifold et al. “Denoising of stimulated Raman scattering microscopy images via deep learning”. In: *Biomedical optics express* 10.8 (2019), pp. 3860–3874.

[87] D. Martin et al. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. 8th Int’l Conf. Computer Vision*. Vol. 2. July 2001, pp. 416–423.

[88] GFPM Matheron. “Random sets and integral geometry”. In: (1975).

[89] Ian S McLean. *Electronic imaging in astronomy: detectors and instrumentation*. Springer Science & Business Media, 2008.

[90] William Meiniel, Jean-Christophe Olivo-Marin, and Elsa D Angelini. “Denoising of microscopy images: a review of the state-of-the-art, and a new sparsity-based method”. In: *IEEE Transactions on Image Processing* 27.8 (2018), pp. 3842–3856.

[91] Christopher A Metzler et al. “Unsupervised Learning with Stein’s Unbiased Risk Estimator”. In: *arXiv preprint arXiv:1805.10531* (2018).

[92] Peyman Milanfar. “A tour of modern image filtering: New insights and methods, both practical and theoretical”. In: *IEEE signal processing magazine* 30.1 (2012), pp. 106–128.

[93] David Minarik, Olof Enqvist, and Elin Trägårdh. “Denoising of scintillation camera images using a deep convolutional neural network: a Monte Carlo simulation approach”. In: *Journal of Nuclear Medicine* 61.2 (2020), pp. 298–303.

[94] Sreyas Mohan et al. “Adaptive Denoising via GainTuning”. In: *Advances in Neural Information Processing Systems* 34 (2021).

[95] Sreyas Mohan et al. “Deep Denoising For Scientific Discovery: A Case Study In Electron Microscopy”. In: *arXiv preprint arXiv:2010.12970* (2020).

[96] Sreyas Mohan et al. “Robust and Interpretable Blind Image Denoising via Bias-free Convolutional Neural Networks”. In: *Proceedings of the International Conference on Learning Representations*. 2020.

[97] Grégoire Montavon et al. “Explaining nonlinear classification decisions with deep taylor decomposition”. In: *Pattern Rec.* 65 (2017), pp. 211–222.

[98] Tiziano Montini et al. “Fundamentals and catalytic applications of CeO2-based materials”. In: *Chemical reviews* 116.10 (2016), pp. 5987–6041.

[99] Nick Moran et al. “Noisier2noise: Learning to denoise from unpaired noisy data”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12064–12072.

[100] Zachary Nado et al. “Evaluating prediction-time batch normalization for robustness under covariate shift”. In: *arXiv preprint arXiv:2006.10963* (2020).

[101] PD Nellist and SJ Pennycook. “Accurate structure determination from image reconstruction in ADF STEM”. In: *Journal of Microscopy* 190.1-2 (1998), pp. 159–170.

[102] Yao Nie, Li Li, and Zidong Wei. “Recent advancements in Pt and Pt-free catalysts for oxygen reduction reaction”. In: *Chemical Society Reviews* 44.8 (2015), pp. 2168–2201.

[103] Ethan Perez et al. “Film: Visual reasoning with a general conditioning layer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[104] JR Peterson et al. “Simulation of astronomical images from optical survey telescopes using a comprehensive photon Monte Carlo approach”. In: *The Astrophysical Journal Supplement Series* 218.1 (2015), p. 14.

[105] Xaq Pitkow. “Exact feature probabilities in images with occlusion”. In: *Journal of vision* 10.14 (2010), pp. 42–42.

[106] Jordi Pont-Tuset et al. “The 2017 DAVIS Challenge on Video Object Segmentation”. In: *arXiv preprint arXiv:1704.00675* (2017).

[107] Javier Portilla et al. “Image denoising using scale mixtures of Gaussians in the wavelet domain”. In: *IEEE Trans. Image Processing* 12.11 (2003).

[108] Mangal Prakash et al. “Fully Unsupervised Probabilistic Noise2Void”. In: *arXiv preprint arXiv:1911.12291* (2019).

[109] Franco P Preparata and Michael Ian Shamos. “Convex hulls: Basic algorithms”. In: *Computational geometry*. Springer, 1985, pp. 95–149.

[110] Y. Quan et al. “Self2Self With Dropout: Learning Self-Supervised Denoising From Single Image”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1887–1895. DOI: 10.1109/CVPR42600.2020.00196.

[111] Marco Ragone et al. “Atomic column heights detection in metallic nanoparticles using deep convolutional learning”. In: *Computational Materials Science* 180 (2020), p. 109722.

[112] Sathish Ramani, Thierry Blu, and Michael Unser. “Monte-Carlo SURE: A black-box optimization of regularization parameters for general denoising algorithms”. In: *IEEE Transactions on image processing* 17.9 (2008), pp. 1540–1554.

[113] M Raphan and E P Simoncelli. “Least squares estimation without priors or supervision”. In: *Neural Computation* 23.2 (Feb. 2011). Published online, Nov 2010., pp. 374–420. DOI: 10.1162/NECO_a_00076.

[114] M Raphan and E P Simoncelli. “Optimal denoising in redundant representations”. In: *IEEE Trans Image Processing* 17.8 (Aug. 2008), pp. 1342–1352. DOI: 10.1109/TIP.2008.925392.

[115] Martin Raphan and Eero P Simoncelli. “Optimal denoising in redundant representations”. In: *IEEE Transactions on image processing* 17.8 (2008), pp. 1342–1352.

[116] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention, Springer, LNCS* 9351 (2015), pp. 234–241.

[117] Uwe Schmidt and Stefan Roth. “Shrinkage fields for effective image restoration”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2774–2781.

[118] Steffen Schneider et al. “Improving robustness against common corruptions by covariate shift adaptation”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[119] Dev Yashpal Sheth et al. “Unsupervised Deep Video Denoising”. In: *arXiv preprint arXiv:2011.15045* (2020).

[120] E P Simoncelli and E H Adelson. “Noise removal via Bayesian wavelet coring”. In: *Proc 3rd IEEE Int’l Conf on Image Proc*. Vol. I. Lausanne: IEEE Sig Proc Society, Sept. 1996, pp. 379–382. DOI: 10.1109/ICIP.1996.559512.

[121] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps”. In: *arXiv preprint arXiv:1312.6034* (2013).

[122] David Smith. “CHAPTER 1: Characterization of nanomaterials using transmission electron microscopy”. English (US). In: *Hierarchical Nanostructures for Energy Devices*. Ed. by Angus I. Kirkland and Sarah J. Haigh. 37th ed. RSC Nanoscience and Nanotechnology 37. Royal Society of Chemistry, Jan. 2015, pp. 1–29. DOI: 10.1039/9781782621867-00001.

[123] Shakarim Soltanayev and Se Young Chun. “Training and Refining Deep Learning Based Denoisers without Ground Truth Data”. In: *arXiv preprint arXiv:1803.01314* (2018).

[124] Shakarim Soltanayev and Se Young Chun. “Training deep learning based denoisers without ground truth data”. In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[125] Geng Sun, Anastassia N. Alexandrova, and Philippe Sautet. “Structural Rearrangements of Subnanometer Cu Oxide Clusters Govern Catalytic Oxidation”. In: *ACS Catalysis* 10.9 (2020), pp. 5309–5317. DOI: 10.1021/acscatal.0c00824.

[126] Amit Suveer et al. “Super-Resolution Reconstruction of Transmission Electron Microscopy Images Using Deep Learning”. In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 548–551.

[127] Hossein Talebi et al. “Better compression with deep pre-editing”. In: *IEEE Transactions on Image Processing* 30 (2021), pp. 6673–6685.

[128] Franklin Tao and Peter Crozier. “Atomic-Scale Observations of Catalyst Structures under Reaction Conditions and during Catalysis”. English (US). In: *Chemical Reviews* 116.6 (Mar. 2016), pp. 3487–3539. ISSN: 0009-2665. DOI: 10.1021/cr5002657.

[129] Matias Tassano, Julie Delon, and Thomas Veit. “DVDnet: A Fast Network for Deep Video Denoising”. In: *Proceedings of the IEEE International Conference on Image Processing*. 2020, pp. 1805–1809.

[130] Matias Tassano, Julie Delon, and Thomas Veit. “FastDVDnet: Towards Real-Time Deep Video Denoising Without Flow Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1351–1360.

[131] Carlo Tomasi and Roberto Manduchi. “Bilateral filtering for gray and color images.” In: *ICCV*. Vol. 98. 1. 1998.

[132] Vladimír Ulman et al. “An objective comparison of cell-tracking algorithms”. In: *Nature Methods* 14 (2017), pp. 1141–1152.

[133] Gregory Vaksman, Michael Elad, and Peyman Milanfar. “Lidia: Lightweight learned image denoising with instance adaptation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 524–525.

[134] Rama K Vasudevan and Stephen Jesse. “Deep Learning as a Tool for Image Denoising and Drift Correction”. In: *Microscopy and Microanalysis* 25.S2 (2019), pp. 190–191.

[135] Joshua L. Vincent et al. “Developing and Evaluating Deep Neural Network-based Denoising for Nanoparticle TEM Images with Ultra-low Signal-to-Noise”. In: (2021).

[136] Dequan Wang et al. “tent: fully test-time adaptation by entropy minimization”. In: *International Conference on Learning Representations*. Vol. 4. 2021, p. 6.

[137] Zhong Lin Wang. “New developments in transmission electron microscopy for nanotechnology”. In: *Advanced Materials* 15.18 (2003), pp. 1497–1514.

[138] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Trans. Image Processing* 13.4 (2004), pp. 600–612.

[139] Philippe Weinzaepfel et al. “DeepFlow: Large displacement optical flow with deep matching”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1385–1392.

[140] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. Technology Press, 1950.

[141] Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. “Noise2Same: Optimizing A Self-Supervised Bound for Image Denoising”. In: *Advances in Neural Information Processing Systems* 33 (2020).

[142] Jun Xu et al. “Noisy-As-Clean: Learning self-supervised denoising from corrupted image”. In: *IEEE Transactions on Image Processing* 29 (2020), pp. 9316–9329.

[143] Tianfan Xue et al. “Video Enhancement with Task-Oriented Flow”. In: *International Journal of Computer Vision (IJCV)* 127.8 (2019), pp. 1106–1125.

[144] Jason Yosinski et al. “How transferable are features in deep neural networks?” In: *arXiv preprint arXiv:1411.1792* (2014).

[145] Songhyun Yu et al. “Joint Learning of Blind Video Denoising and Optical Flow Estimation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[146] Weiting Yu, Marc D Porosoff, and Jingguang G Chen. “Review of Pt-based bimetallic catalysis: from model surfaces to supported catalysts”. In: *Chemical reviews* 112.11 (2012), pp. 5780–5817.

[147] H. Yue et al. “Supervised Raw Video Denoising With a Benchmark Dataset on Dynamic Scenes”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2298–2307. DOI: 10.1109/CVPR42600.2020.00237.

[148] Christopher Zach, Thomas Pock, and Horst Bischof. “A duality based approach for realtime tv-l 1 optical flow”. In: *Joint pattern recognition symposium*. Springer. 2007, pp. 214–223.

[149] Kai Zhang, Wangmeng Zuo, and Lei Zhang. “FFDNet: Toward a Fast and Flexible Solution for CNN based Image Denoising”. In: *IEEE Trans. Image Processing* (2018).

[150] Kai Zhang et al. “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising”. In: *IEEE Trans. Image Processing* 26.7 (2017), pp. 3142–3155.

[151] Xiaoshuai Zhang et al. “Dynamically unfolding recurrent restorer: A moving endpoint control method for image restoration”. In: *arXiv preprint arXiv:1805.07709* (2018).

[152] Yide Zhang et al. "A poisson-gaussian denoising dataset with real fluorescence microscopy images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11710–11718.

[153] Yulun Zhang et al. "Residual Dense Network for Image Restoration". In: *CoRR* abs/1812.10477 (2018).

[154] Qian Zheng et al. "Single Image Reflection Removal with Absorption Effect". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13395–13404.

[155] Hai Jing Zhu, Bo Chong Han, and Bo Qiu. "Survey of astronomical image processing methods". In: *International Conference on Image and Graphics*. Springer. 2015, pp. 420–429.

[156] Jian-Min Zuo and J.C.H. Spence. *Advanced Transmission Electron Microscopy, Imaging and Diffraction in Nanoscience*. Jan. 2017. ISBN: ISBN 978-1-4939-6607-3.